WhereScape® RED

User Guide

Version 10.1.0.0 | May 2024

Table of Contents

Overview

Traditionally, data warehouses take too long to build and are too hard to change. WhereScape RED is an Integrated Development Environment, designed to support the quick and simple building and managing of data warehouses.

It has the flexibility to enable you to build a variety of architectures including:

- enterprise data warehouses
- dimensional data warehouses
- data marts
- user facing views, aggregates and summaries

In all cases, the core values of WhereScape RED are twofold: its rapid building capabilities that enable better data warehouses to be built, faster, and its integrated environment that simplifies management.

As a data warehouse specific tool, WhereScape RED embodies a simple, pragmatic approach to building data warehouses. With WhereScape RED you specify what you want to achieve by dragging and dropping objects to create a meta view, and then let WhereScape RED do the heavy lifting of creating the necessary tables, procedures, documentation, etc. Data warehouse wizards prompt for additional information at critical points to provide the maximum value from the generated objects.

Different data warehousing approaches including agile, prototyping and waterfall are supported by WhereScape RED. Agile developers will find specific functionality has been included to support common agile practices. For developers who are new to data warehousing, or are looking for a fast, pragmatic approach, WhereScape's Pragmatic Data Warehousing Methodology can be used.

The basic concepts behind WhereScape 's Pragmatic Data Warehousing Methodology are:

- minimize the impact on the source systems
- centralize management within the data warehouse
- store transactional data at the lowest practical grain within the data warehouse
- snapshot, combine and rollup transactional tables to provide additional value
- utilize star schemas, views or cubes for end user access
- allow for incremental loads from day one
- use of an iterative approach
- simplifying the reconciliation of Load tables to source system tables - data is cleaned and transformed in subsequent Stage tables instead
- design the data warehouse independently from the end user tool layer

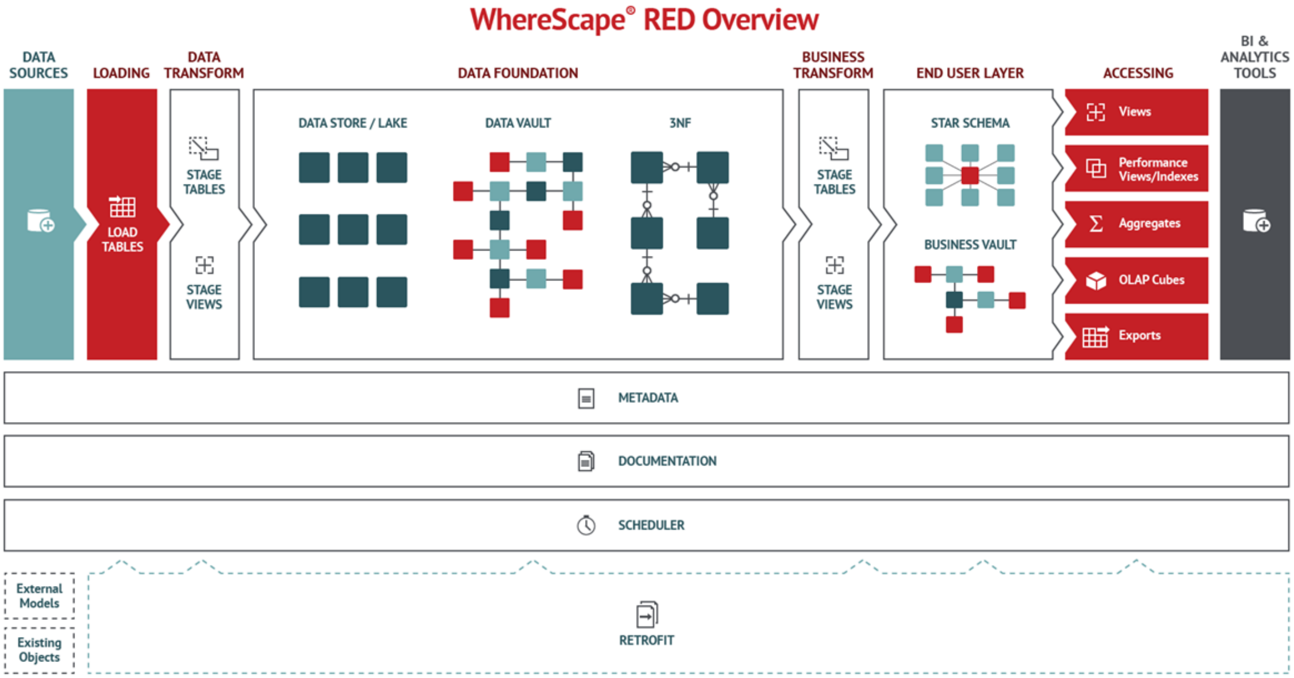
WhereScape RED supports these concepts to facilitate very rapid delivery of data warehouses. WhereScape RED controls the flow of data from the source systems through transforming and modeling layers to analysis areas.

Different styles of data warehousing (e.g. EDW 3NF, dimensional, etc.) are supported and utilize different objects, but all follow the same basic flow.

Data Flow - Enterprise Models

1. Source (OLTP) System
2. load tables
3. stage tables
4. data store tables
5. model tables, dimension tables, or detailed (transactional) fact tables
6. roll up fact table(s)
7. aggregate and/or KPI fact table(s)
8. views
9. export objects
10. Microsoft Analysis Services cubes

The diagram below shows the objects and the information flow:



Data Flow

Data is moved from source tables to Load tables via scripts, database links and ODBC links. These Load tables are created by dragging and dropping from a connection object. Load tables are generally based on source system tables. Their main purpose is to be a destination for moving data as simply and quickly as possible from the source system. Load tables generally hold a single unit of data (e.g. last night or last month), and truncated at the start of each extract. Transformations can be performed on the columns during the load process, if required.

Load tables feed Stage tables, which in turn feed Data Store, Model or Dimension tables. Data from multiple Load tables can be combined at this level.

First tier transactional tables (Fact or Model) are created and updated from Stage tables. Second tier tables (Model, Summary Rollup, Aggregate, KPI, etc.) are created and updated from lower level tables.

Cubes can be created from transactional tables or views.

Procedural code

WhereScape RED generates procedural code in the target database's native language at each stage in the data warehouse build process. The generated code is, in nearly all cases, sufficient to create a rapid prototype of the data warehouse.

While the generation of code is often seen as a key benefit of WhereScape RED, the ability to control and manage custom code is also critical to the long term management of the data warehouse environment.

In most cases 85-100% of the generated code is taken through to production with no customization required.

Scheduler

The flow of data from the source systems to data warehouse tables is controlled and managed by the WhereScape RED Scheduler. All generated code includes audit and error logging logic that is used by the Scheduler.

The Scheduler provides a single point of control for the data warehouse. From the Scheduler, the state of all jobs can be ascertained. Any warning or error messages can be investigated and if a problem occurs, the Scheduler controls the restart of the job from the point of failure.

Documentation

Documenting the data warehouse is often a task left until last, and in many cases done once (if done at all!) and not kept up to date. WhereScape RED generates user and technical documentation, including diagrams in HTML format.

Technical documentation includes copies of all current procedures.

User documentation includes a glossary of business terms, available independently of any end user tool.

Where additional specific information needs to be included in the documentation, WhereScape RED supports the inclusion of custom HTML pages in the generated output. This means in many cases the entire documentation requirements can be managed from one location, and regenerated as changes occur.

WhereScape RED and Traditional ETL Tools

WhereScape RED 's core strength is in the rapid building of data warehouse structures. Organizations that have already purchased traditional ETL tools can use WhereScape RED as a pureplay data warehouse toolset. WhereScape RED can be used to iteratively build data marts or presentation layer objects that need to be constantly updated to keep relevant for end users. In most cases, customers will find that WhereScape RED has enough ETL capabilities to build the entire data warehouse, using the database rather than a proprietary engine to perform ETL processing.

The cross over in functionality between ETL tools and WhereScape RED is not large. WhereScape RED is tightly integrated into the data warehouse database and has an embedded data warehouse building approach. For WhereScape, data movement is the start of the process—from source system to Load tables. The key benefits of the product: development productivity and an integrated environment to manage and maintain your data warehouse, comes after the data movement stage.

Where a traditional ETL tool is already in use, the output of the ETL process is a WhereScape RED Load, Stage, Dimension, Fact or Model table from which WhereScape RED builds more advanced data warehouse structures.

Design Introduction

WhereScape RED can be used to build data warehouses based on any number of design philosophies from EDW 3NF enterprise data warehouses with consumer data marts through to federated or conformed star schema based warehouses. In the absence of another approach, the following methodology can be used for the design of data warehouses.

Note:

This section can be skipped if you already have data warehouse design experience or a methodology you wish to utilize. It is meant to provide the novice designer with some tips for designing a data warehouse.

Design Approach

The concepts behind the WhereScape Pragmatic Data Warehouse Methodology are as follows:

1. Building an enterprise-wide data warehouse is a process—an evolution rather than a big bang. Start small and grow the warehouse in manageable chunks until all the pieces are in place. Once you reach that stage, changes and new source systems will continue the process.
2. You need to understand the big picture, but not get lost in it. Talk to all the various departments, business units and companies within the organization. Do so at a relatively high level and try to understand how the information from each area impacts or affects the others. Identify commonalities and areas where the same information is handled in different ways. This process should take days or weeks not months.
3. Identify the high value, high return and possibly easiest areas of the business. Drill down in these areas and break down the workload into small manageable chunks of work, for example, one to two analysis areas. Agree on the first component of the data warehouse and do that.
4. Get an understanding of the source system for this first component or analysis area. If possible, get an entity relationship diagram and talk to the people who built or support the application. Identify the tables that contain the key information you will need. The goal is a quick and initial view, a detailed specification is not required.
5. Design the first component. This design should be a first draft and can be written rather than using a design tool. Remember at this stage what the end users want is not really known, so don't set the design in concrete, or spend a large amount of time in this area.

   Note

   Experienced users of WhereScape RED often dispense with a design and go straight to building a prototype.

6. Build a prototype. In most cases, this should not take more than one or two weeks—experienced WhereScape RED developers can expect to build prototypes in hours or days. Concentrate on the detailed and descriptive data, unless you have a clear picture of the summarized requirements. Do as much as possible in terms of validating the data back to the source system. If dealing with a large or complex source system, then only deliver a segment in this prototype, e.g. one branch, one store, one product group, etc. Keep It Simple.
7. Demonstrate the prototype to a group of the key users. Then drill down to a subset of key users (we recommend no more than three) who will help you go forward with the design. If possible, give these users access to the prototype and get them using the data. Stress that data accuracy is not the issue at this stage, rather the look and feel.
8. Enhance the prototype with the feedback provided by the users. Again a quick process. If complicated requirements evolve, then create a plan to implement, doing the highest value parts first. The goal is to get quick buy in and support from the two or three key users.
9. Provide key users access to the reworked prototype and get them using the data. Have them define the business names for all the measures and attributes and to define any pre-calculated measures that they frequently use. Get them to define the hierarchies in the data. Ascertain the commonly utilized queries and reports, and see if there would be a better way of presenting these.
10. From the user feedback look at the need or possibility of using higher level fact tables, such as summaries, aggregates, snapshot or composite rollup tables.

The concepts and methodologies for designing and building a data warehouse are beyond the scope of this manual.

It is assumed that the reader understands the basic concepts of a data warehouse, and is familiar with modeling, EDW 3NF, star and snowflake schemas, dimensions, fact tables, etc.

Refer to the WhereScape web site for a basic overview of data warehouse design if required.

Installation

WhereScape RED is designed to build data warehouse analysis areas within a database environment. The installation and configuration process are very rarely a simple task as it involves several layered products external to WhereScape RED. For the more complex cases, several hours or days should be set aside to complete this task.

Therefore, it is important that the reader begins the installation process with the correct expectations. Highlighted below are the major steps in the installation and setup of WhereScape RED. In many environments some of these steps will have been completed independently of the installation process or may need to be completed by a third party.

Note:

This document does not cover the creation and configuration of the database required to run the data warehouse. It does not cover the procedures required to create and maintain database users; nor does it cover the installation of the database client software. Refer to the database vendor manuals for assistance.

Overview

WhereScape RED provides an environment to facilitate the rapid production of prototype data warehouse analysis areas. It also provides the functionality to move that prototype into a production environment and support the day-to-day running of a data warehouse.

The Installation and Administration Guide provides the information needed to:

- Install the WhereScape RED software.
- Validate the various software components required by WhereScape RED.
- Install the WhereScape RED metadata.
- Install a Scheduler.
- Optionally install third-party data warehouse applications.

Note:

The software described in this manual is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

WhereScape Forum

A web forum is available at www.wherescape.com. This forum contains information on the latest version, and bug reports that may be relevant for installation. In addition, the WhereScape Blog is available at www.wherescape.com which may provide additional information.

Installing WhereScape RED

Complete the following procedure to install the WhereScape RED software:

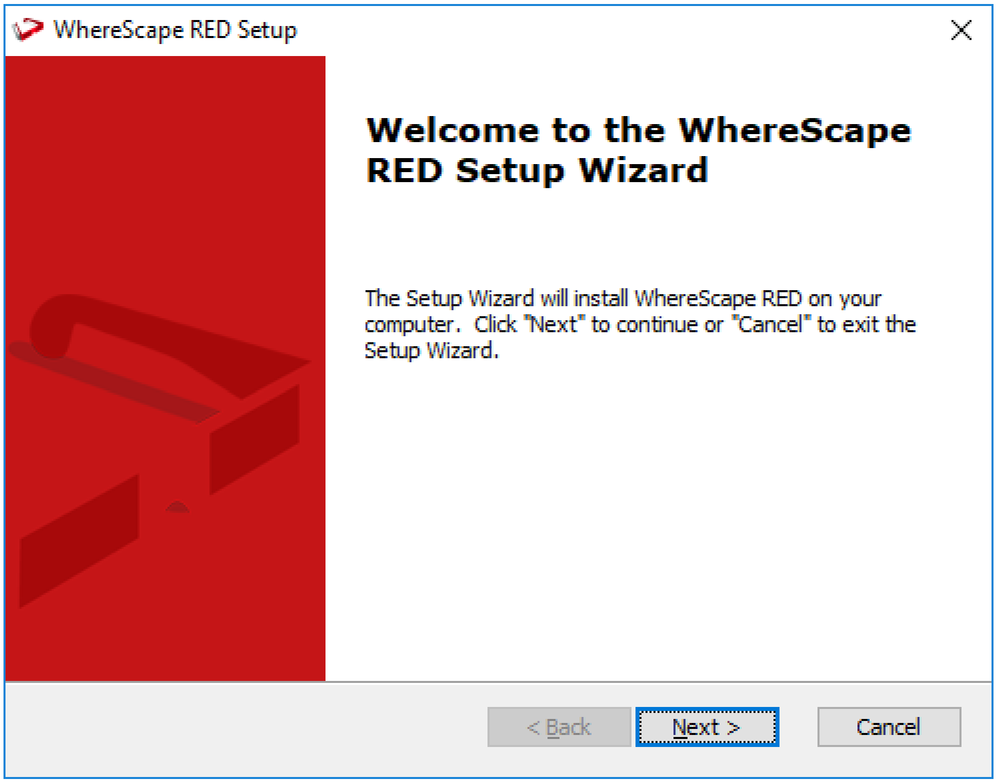
Notes:

Before you begin the install process:

- Stop any WhereScape RED schedulers running on the computer.
- Close any other applications that are running.
- The following packages are downloaded and installed by WhereScape RED as required. If working offline you need to ensure they are already installed:

Package	File Name	File Size (KB)
Visual C++ Redistributable for Visual Studio 2019 (x64)	vc_redist.x64.exe	14,708
Visual C++ Redistributable for Visual Studio 2019 (x86)	vc_redist.x86.exe	14,028
.NET Framework 4.6.1	NDP461-KB3102436-x86-x64-AllOS-ENU.exe	66,095
.NET Framework 4.6.1 (web installer)	NDP461-KB3102438.exe	1,391

1. Double-click the RED_xxxxx.exe file (For example: the downloaded file for version 6.8.4.0, would be RED_68400.exe).
   The **WhereScape RED Setup Wizard** window is displayed.



Welcome to the WhereScape RED Setup Wizard

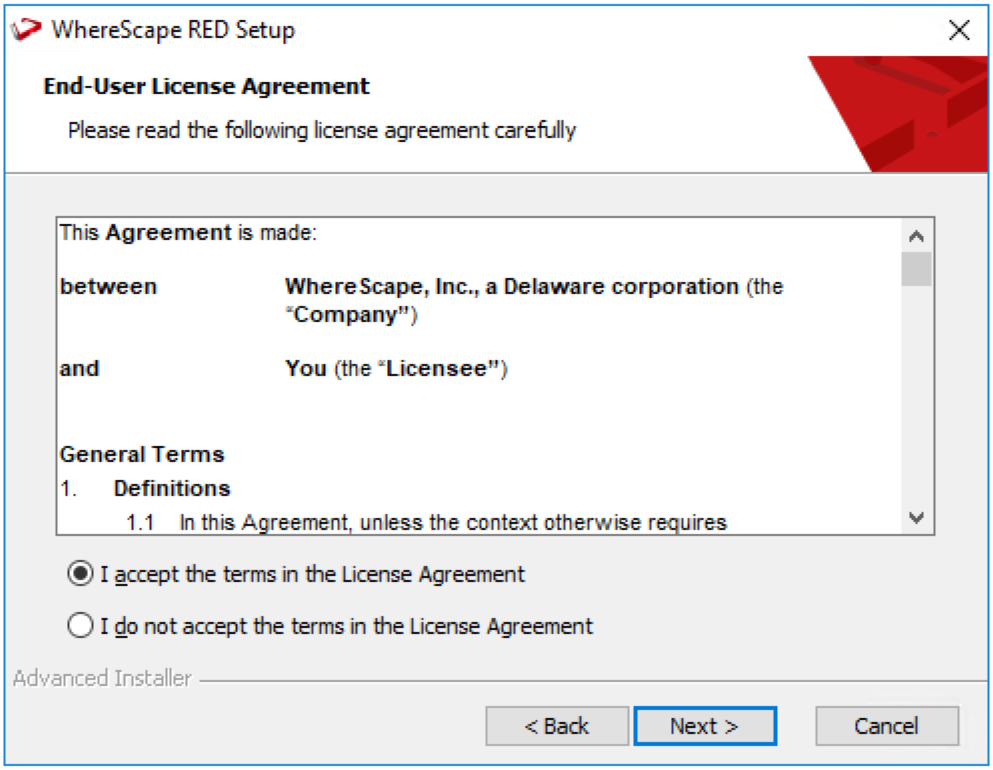
The Setup Wizard will install WhereScape RED on your computer. Click "Next" to continue or "Cancel" to exit the Setup Wizard.

2. Click **Next** to continue with the installation.

Note:

The installer checks for certain prerequisites that are needed to run some components of RED. If any of the prerequisites are missing, clicking **Next** takes you to the list of required prerequisites and you need to install them. To skip this step and jump directly to the installation of RED, click **Finish.**

The **WhereScape RED Setup End-User License Agreement** window is displayed. Read the license terms.



WhereScape RED Setup

End-User License Agreement

Please read the following license agreement carefully

This **Agreement** is made:

between **WhereScape, Inc., a Delaware corporation** (the "**Company"**)

and **You** (the "**Licensee"**)

General Terms

1. **Definitions**

1.1 In this Agreement, unless the context otherwise requires

● I accept the terms in the License Agreement

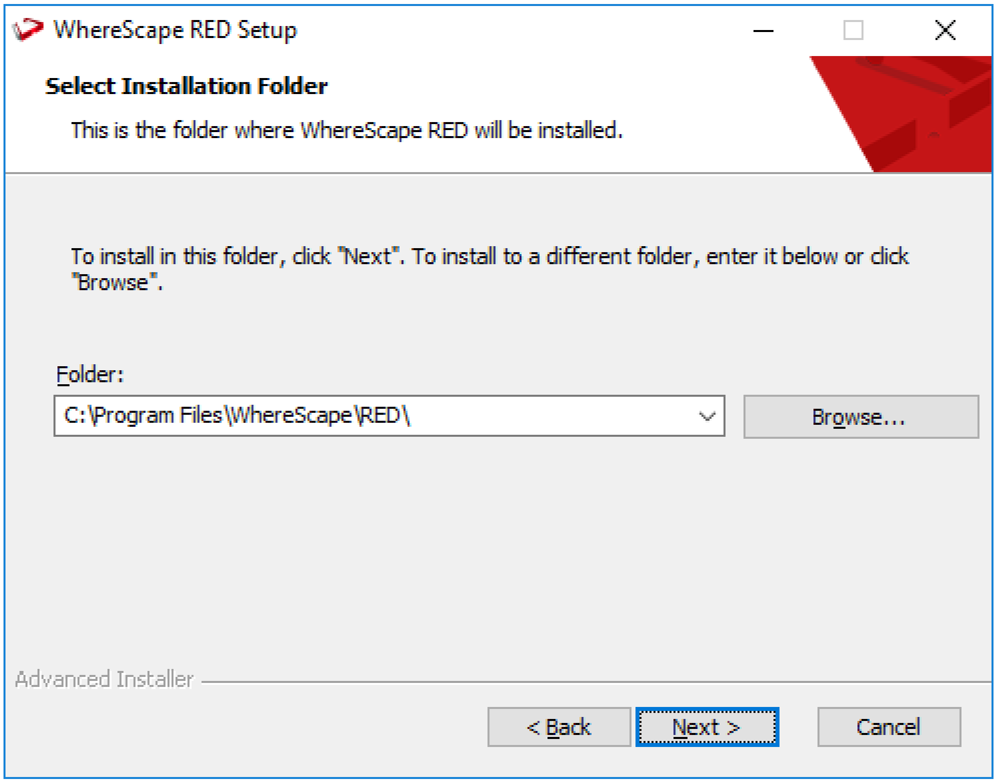
○ I do not accept the terms in the License Agreement

3. Select the **I accept the terms in the License Agreement** option and click **Next** to continue.

Note:

If you do not accept the license agreement, you cannot continue with the installation.

4. The **Select Installation Folder** window is displayed, showing the directory where the WhereScape RED software will be installed.
   To change the directory, click **Browse...** to navigate to the required **Installation Folder** and then click **Next** to continue.


WhereScape RED Setup

Select Installation Folder

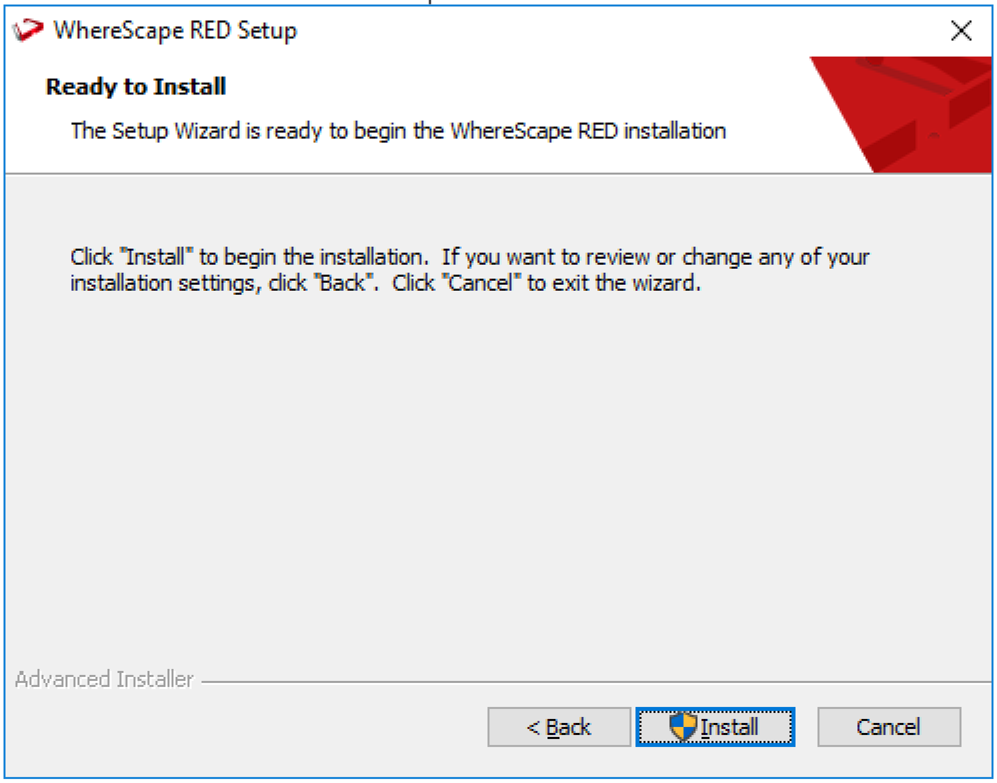
This is the folder where WhereScape RED will be installed.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder:
C:\Program Files\WhereScape\RED\

5. The **Ready to Install** window is displayed, giving you the opportunity to go back and change any of the information you have entered so far.

   Click **Install** to install the WhereScape RED software.

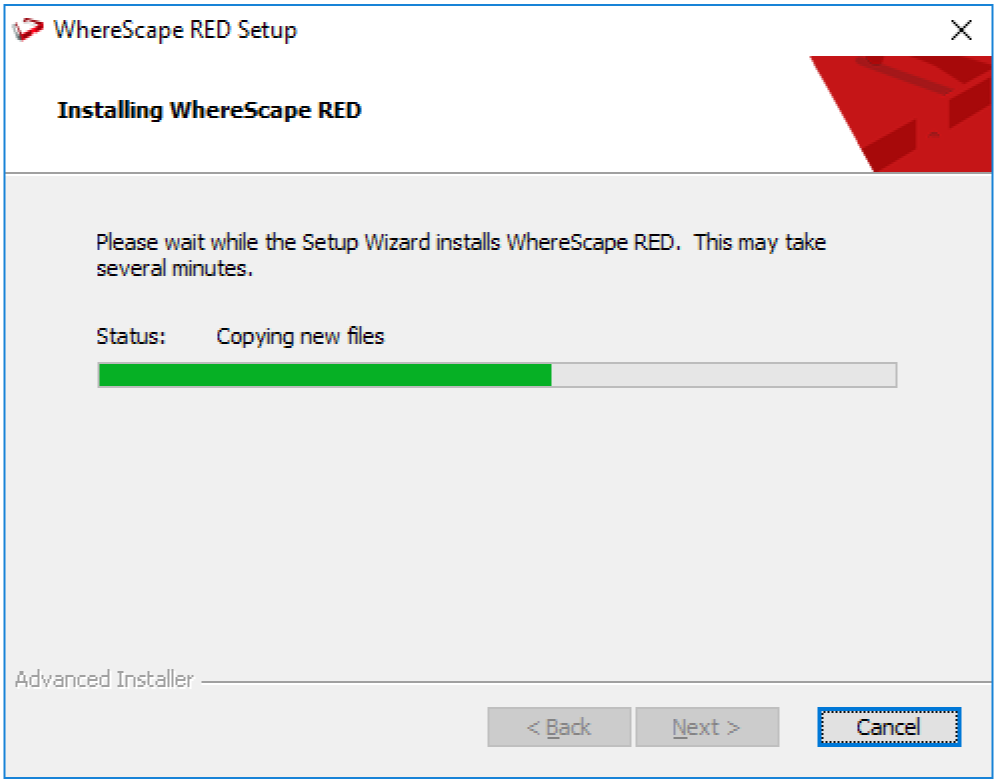

WhereScape RED Setup

Ready to Install

The Setup Wizard is ready to begin the WhereScape RED installation

Click "Install" to begin the installation. If you want to review or change any of your installation settings, click "Back". Click "Cancel" to exit the wizard.

6. The **Installing WhereScape RED** window is displayed, showing you the progress of the installation.



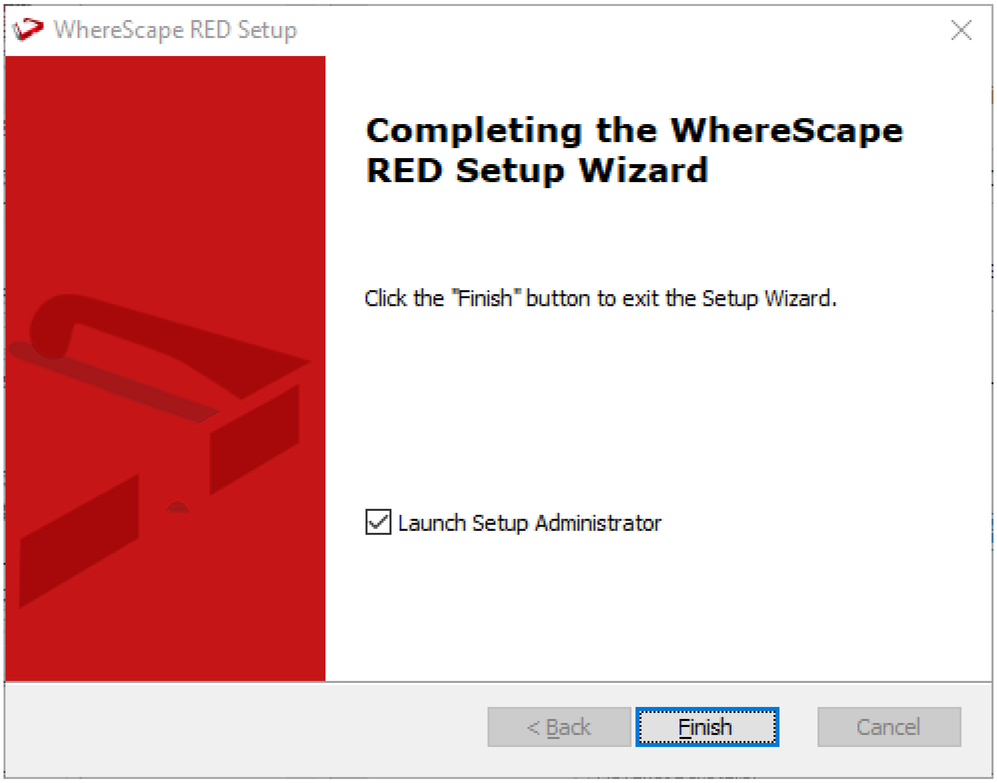
WhereScape RED Setup

Installing WhereScape RED

Please wait while the Setup Wizard installs WhereScape RED. This may take several minutes.

Status: Copying new files

7. A window is displayed after completing the installation. Click **Finish** to exit and **Launch the Setup Administrator**.



WhereScape RED Setup

Completing the WhereScape RED Setup Wizard

Click the "Finish" button to exit the Setup Wizard.

☑ Launch Setup Administrator

Note:

If a scheduler was running or a program was open (when doing an upgrade), a prompt displays advising you that a reboot is required to finish the installation.

Installing WhereScape RED from the Command Line

It is possible to install WhereScapeRED from the command line and specify a custom installation directory, using the APPDIR parameter by running a batch file.

This process can be used for all metadata database types where the exe file is configured to copy the APPDIR property into all other relevant properties.

Note:

Command Line installations will only work if the relevant version has not yet been previously installed. If the same version has already been installed, it must always be removed first.

Example

The following is an example script that loads RED with full verbose logging:

```
@echo off
cd c:\users\user\downloads\
"RED_68500.exe" APPDIR="c:\WhereScape68500\" ALLUSERS=1 RebootYesNo="No" Reboot="Suppress"
/qn+ /L*vx "c:\WhereScape\log.txt"
exit
```

Please refer to Caphyon's Advanced Installer **documentation** for more details about command-line options.

Please refer to **Microsoft Documentation** for more details about Msiexec command-line options.

Installing the RED Metadata

RED Setup Wizard Overview

WhereScape RED provides a metadata installer that enables you to setup and configure your metadata repository settings. The RED **Setup Wizard** guides you through the process of configuring your metadata repository and the other software components required to run WhereScape RED.

The RED **Setup Wizard** guides you through the following installation processes:

1. Installing the RED Metadata
2. Metadata and Enablement Pack installation
3. RED Metadata Configuration

Installation and Configuration of RED Metadata

The following sections describe the steps to install and configure the RED metadata repository and the other software components required to run RED.

Metadata and Enablement Pack Installation

Note:

Before you begin the RED Setup Wizard:

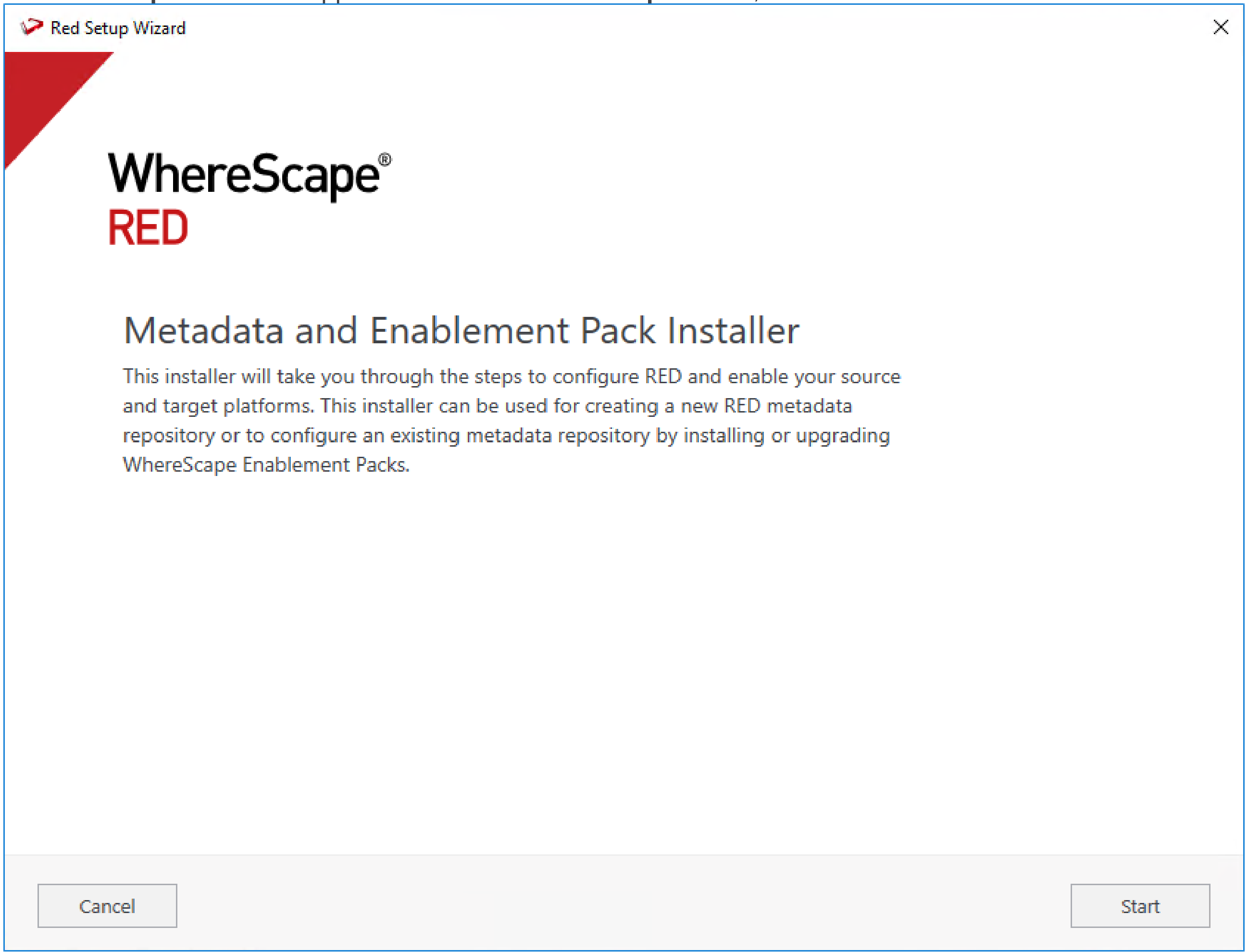
- Stop any WhereScape RED schedulers running on the computer.

Note:

- Close any other applications that are running.

The following steps describe how install the RED metadata repository and other software components to run RED with the functions and features that you configure.

1. After successfully installing WhereScape RED, open the RED Windows installation directory, double-click the **RedSetupWizard.exe** application to launch the **Setup Wizard**, and click **Start**.



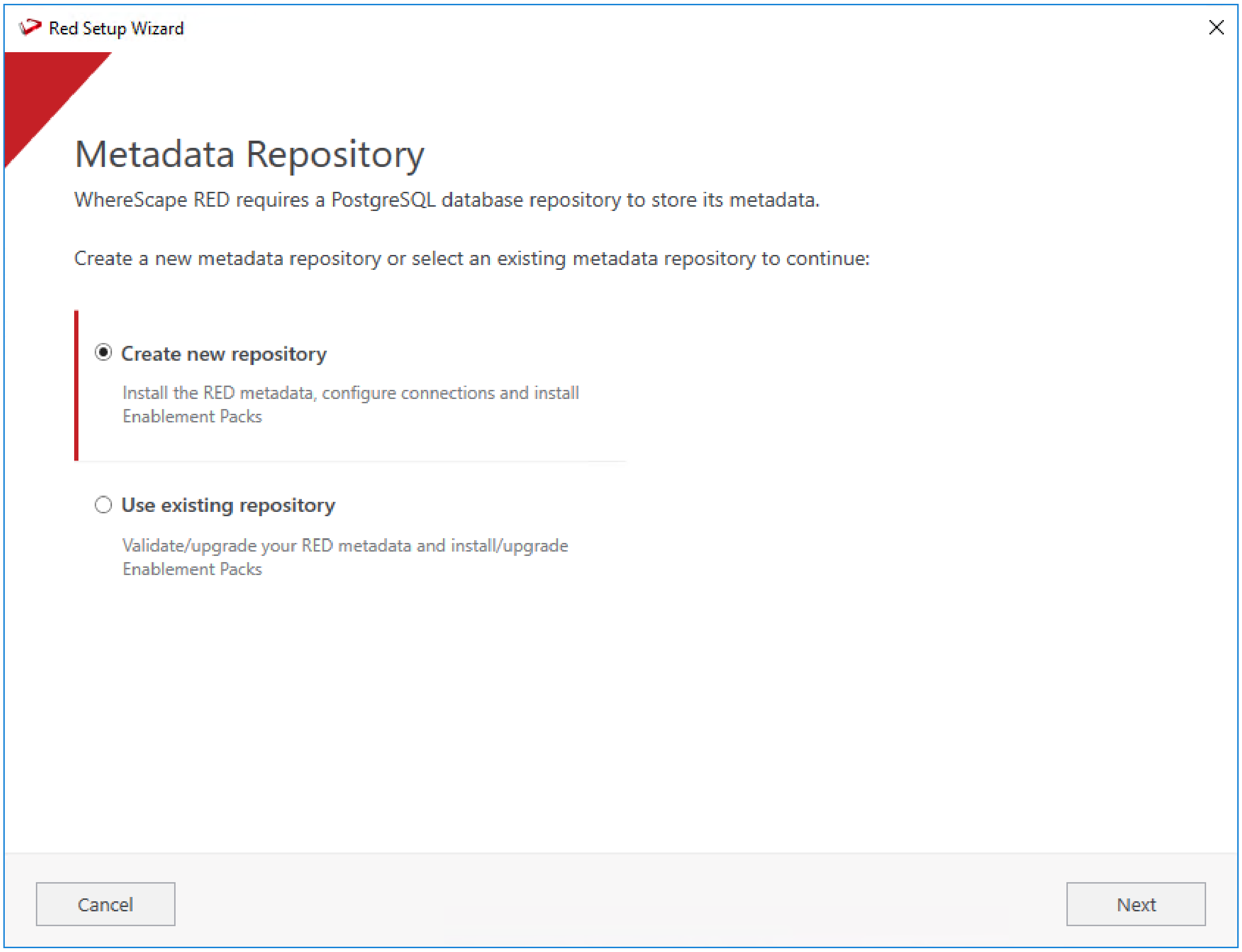
Metadata and Enablement Pack Installer

This installer will take you through the steps to configure RED and enable your source and target platforms. This installer can be used for creating a new RED metadata repository or to configure an existing metadata repository by installing or upgrading WhereScape Enablement Packs.

Note:

If a valid license is not found on the system the RedInstallLicense wizard will be launched.

2. Click **Next** to continue, the **Metadata Repository** window is displayed.
   Select if you want to create a new repository or use an existing repository to store the RED metatdata and then click **Next** to continue.



Red Setup Wizard

Metadata Repository

WhereScape RED requires a PostgreSQL database repository to store its metadata.

Create a new metadata repository or select an existing metadata repository to continue:

● **Create new repository**

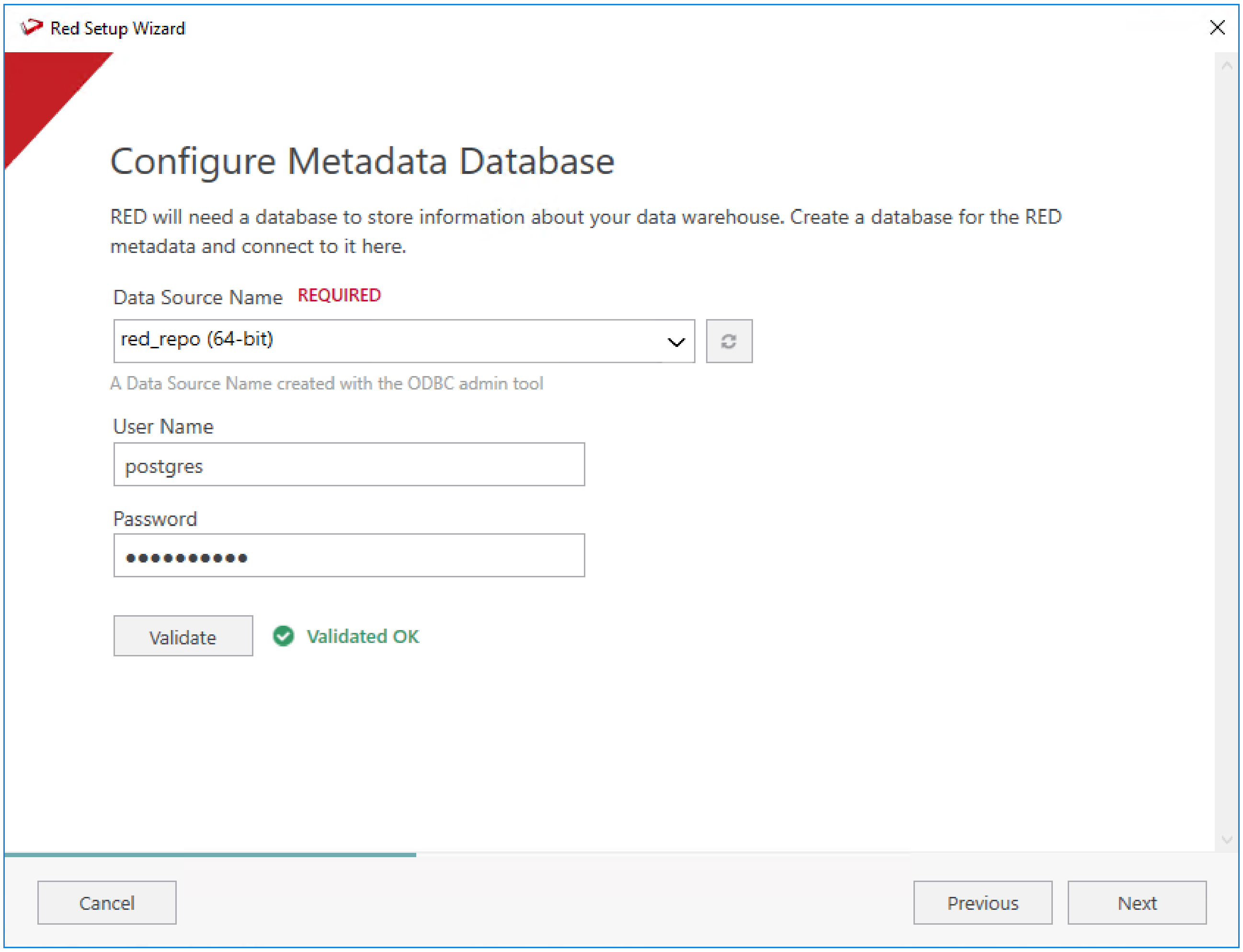
Install the RED metadata, configure connections and install Enablement Packs

○ **Use existing repository**

Validate/upgrade your RED metadata and install/upgrade Enablement Packs

RED Metadata Configuration

1. To start configuring the metadata is required to have a valid empty PostgreSQL database and an ODBC DSN.
2. Selecting the **Create new repository** option displays the **Configure Metadata Database** screen. Specify a data source to which RED connects and the log in details for the selected source database.



Configure Metadata Database

RED will need a database to store information about your data warehouse. Create a database for the RED metadata and connect to it here.

Data Source Name REQUIRED

red_repo (64-bit)

A Data Source Name created with the ODBC admin tool

User Name

postgres

Password

●●●●●●●●●●●

✓ Validated OK

Type in the required metadata connection details:

1. Select from the dropdown list the ODBC data source name (DSN) for the metadata repository connection.
2. Enter the User Name and the Password to connect to the metadata repository connection.

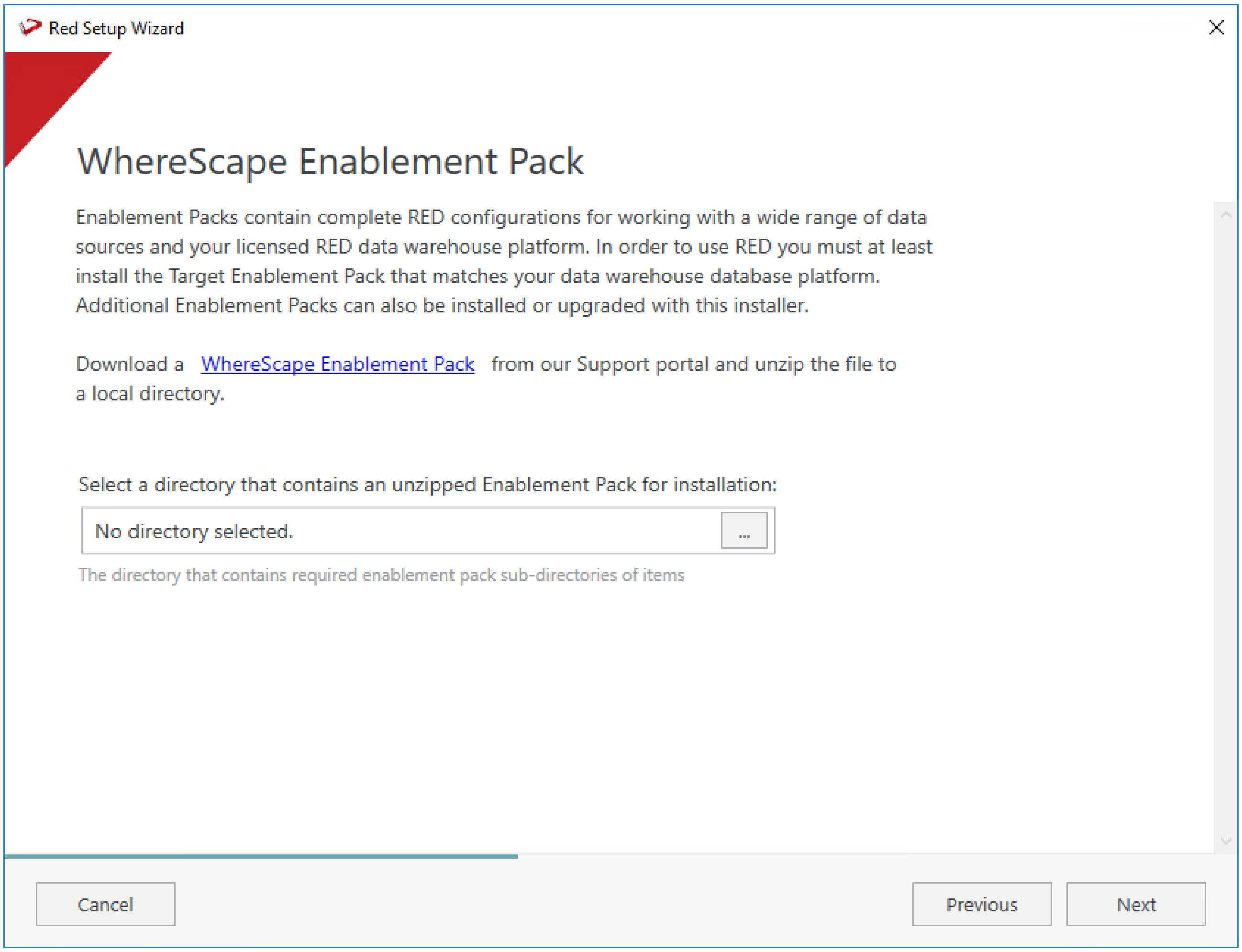
3. Click **Validate** to continue the data source and log in credentials validation. After successful validation, click **Next** to continue.

Notes:

The validation is performed against the selected data source to ensure:

- The database is not already being used by another RED repository.
- The database is not being used for another purpose.

4. The **WhereScape Enablement Pack** window displays. As the dialog explains, you can download the Enablement Pack from the Support portal and unzip the file to a local directory, click ... to select the directory that contains the unzipped Enablement Pack, once your configuration is complete, click **Next** to continue.



WhereScape Enablement Pack

Enablement Packs contain complete RED configurations for working with a wide range of data sources and your licensed RED data warehouse platform. In order to use RED you must at least install the Target Enablement Pack that matches your data warehouse database platform. Additional Enablement Packs can also be installed or upgraded with this installer.

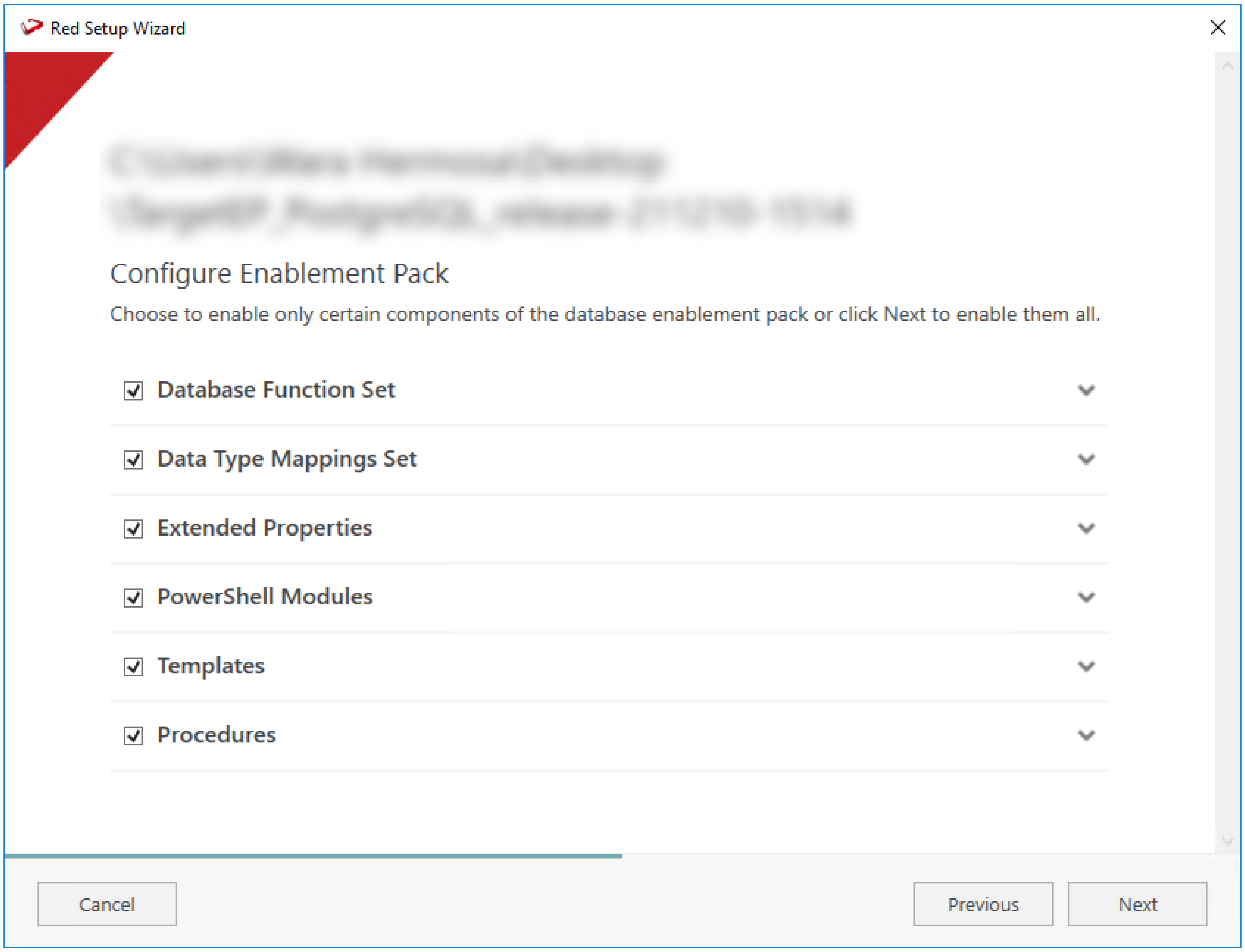
Download a WhereScape Enablement Pack from our Support portal and unzip the file to a local directory.

Select a directory that contains an unzipped Enablement Pack for installation:

No directory selected.

The directory that contains required enablement pack sub-directories of items

5. The **Configure Enablement Packs** window displays showing a list of components that you can include or exclude in RED.



[redacted]

Configure Enablement Pack

Choose to enable only certain components of the database enablement pack or click Next to enable them all.

- ☑ Database Function Set
- ☑ Data Type Mappings Set
- ☑ Extended Properties
- ☑ PowerShell Modules
- ☑ Templates
- ☑ Procedures

All components are selected by default. Click the drop-down arrow to view a sub-menu of the items available under each component.

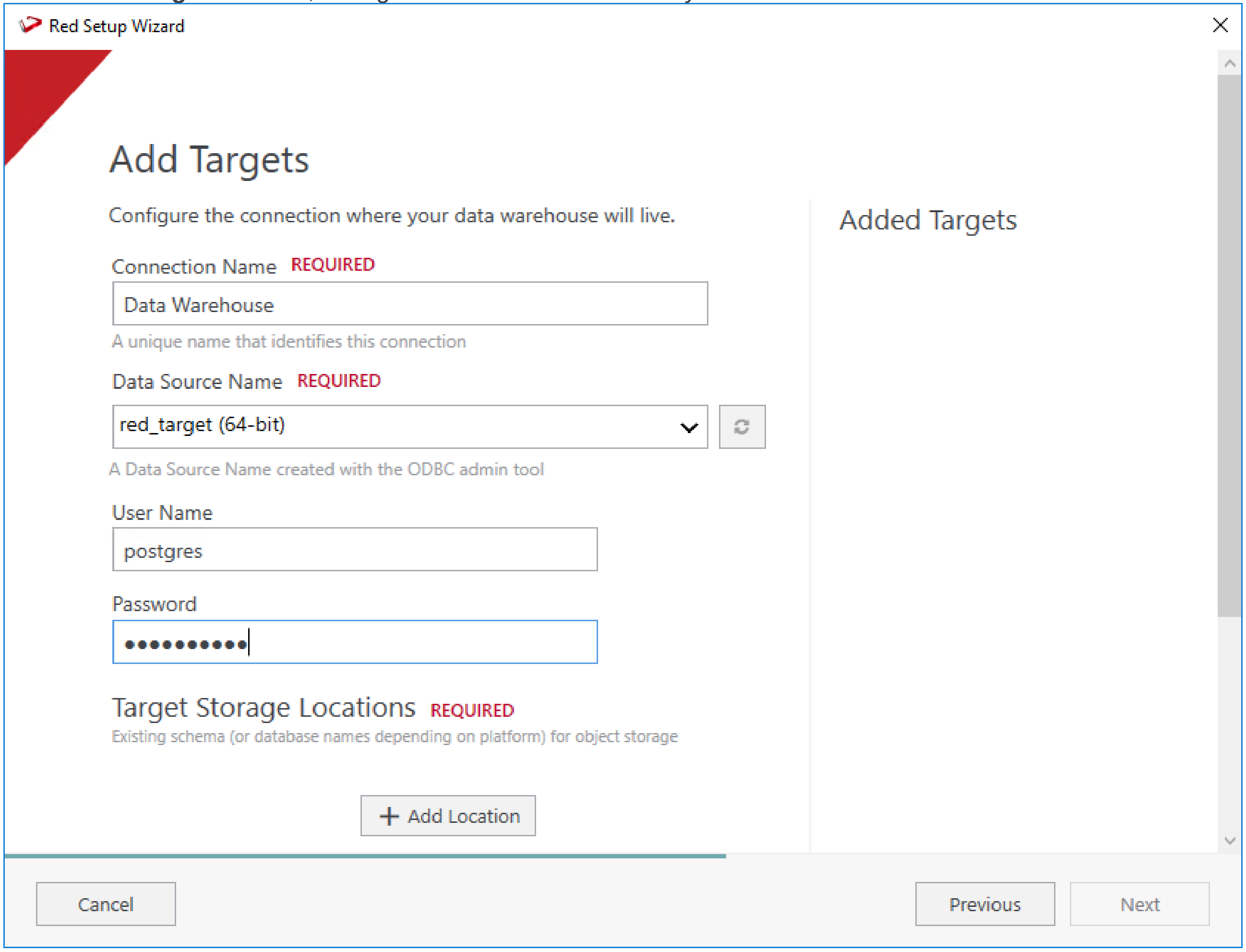
Note:

The enablement pack components vary, depending on the type of database enabled.

Option	Description
Database Function Set	Enables you to select the RED database function set that will be installed with the enablement pack. Database function sets contain a list of functions and operators that are used for building transformations.
Data Type Mappings Set	Enables you to select the RED data type mapping set that will be installed with the enablement pack. Data type mapping sets contain a list of mappings that are used when loading tables into the data warehouse.
Extended Properties	Enables you to select the RED extended property definition set that will be installed with the enablement pack. Extended properties are used to facilitate connection to a specific database technology (e.g. Snowflake, Amazon Redshift, Microsoft Azure, etc.) and also used in the script based processing of all table objects in the database target.
PowerShell Modules	Enables you to select the REDPowerShell modules that will be installed

Option	Description
	with the enablement pack. Data type mapping sets contain a list of mappings that are used when loading tables into the data warehouse
Python Modules	
Templates	Enables you to select the RED templates that will be installed with the enablement pack. Templates are used in processing the data warehouse table objects. .
Scripts	Enables you to select the RED scripts that will be installed with the enablement pack. Scripts are used in processing the data warehouse table objects. .
Procedures	Enables you to select the RED procedures that will be installed with the enablement pack. Procedures are used in processing the data warehouse table objects. .
UI Configurations	
Files Automatically Executed	Enables you to select the RED files that will be installed with the enablement pack.

6. In the **Add Targets** screen, configure the connection where your data warehouse will reside.



Red Setup Wizard

Add Targets

Configure the connection where your data warehouse will live.

Connection Name REQUIRED: Data Warehouse
A unique name that identifies this connection

Data Source Name REQUIRED: red_target (64-bit)
A Data Source Name created with the ODBC admin tool

User Name: postgres

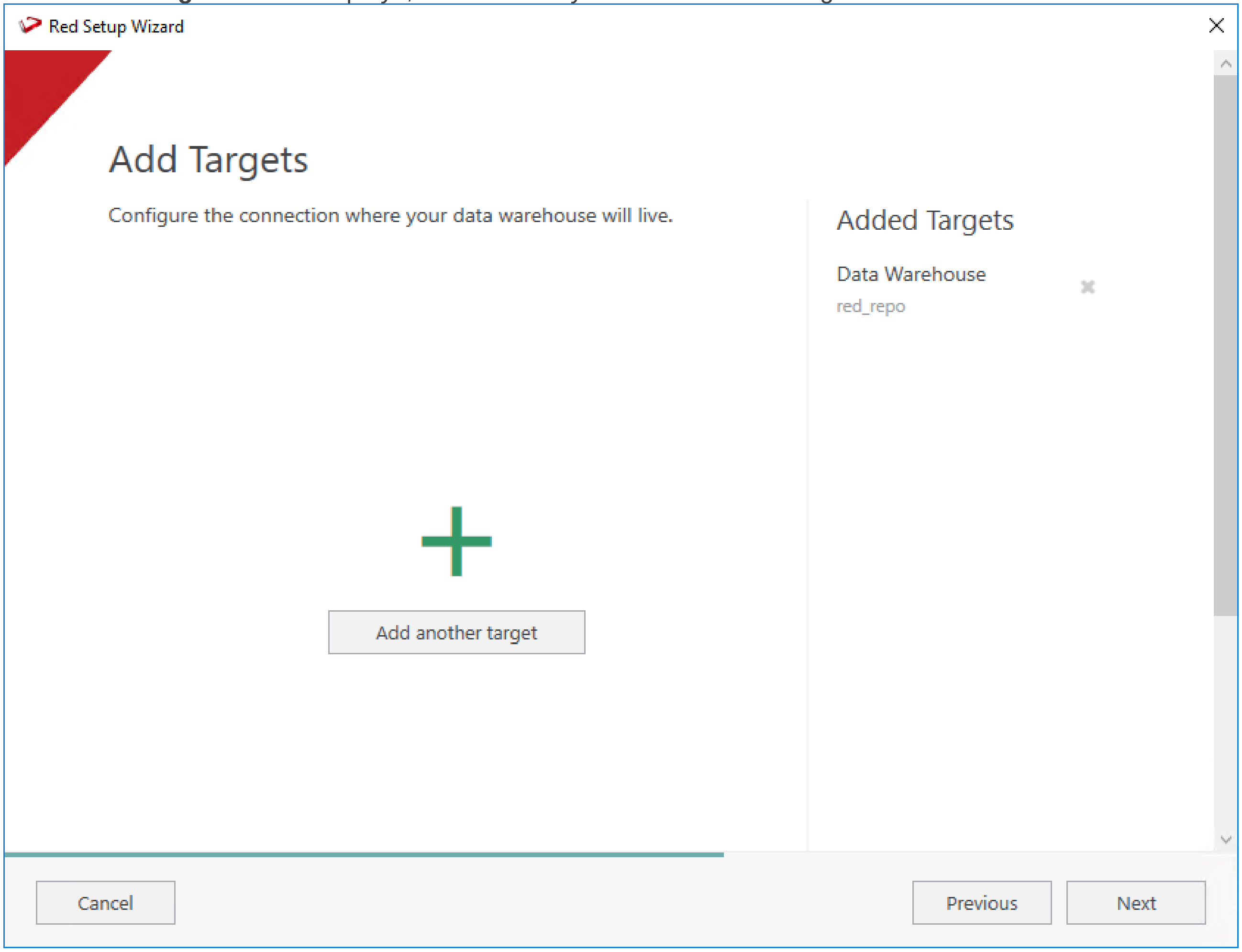
Password: ••••••••••

Target Storage Locations REQUIRED
Existing schema (or database names depending on platform) for object storage

1. Add a **Connection Name**, which needs to be a unique name to identify your connection.
2. Select the **Data Source Name** created with the ODBC admin tool.
3. Add **User Name** and **Password** associated to your connection
4. Add the location of the existing schemas for object storage, follow these steps to complete this section:

1. When you click **Add Location**the Target Storage Location populates automatically. Enables you to add and validate a schema in the specified target database. If the schema defined does not exist in the target database, it is created during the installation process.
2. After validating , click **Add** to add the target storage location, and click **Next** to continue.

7. A new **Add Targets**screen displays, which allows you to add another targets.



Add Targets

Configure the connection where your data warehouse will live.

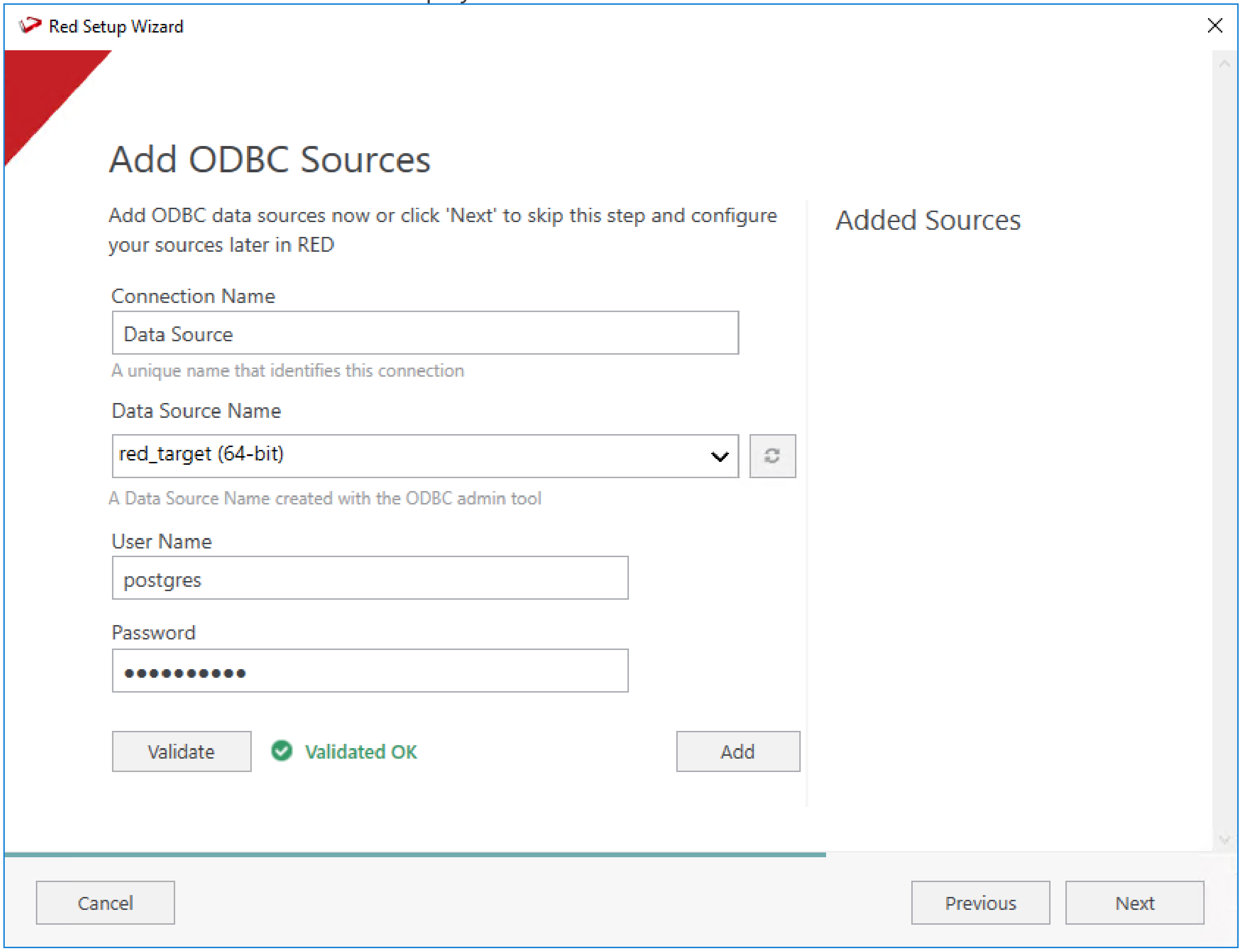
Added Targets

Data Warehouse

red_repo

After successfully adding your database target(s) and schema, click **Next** to continue.

8. The **Add ODBC Sources** window displays.



Red Setup Wizard

Add ODBC Sources

Add ODBC data sources now or click 'Next' to skip this step and configure your sources later in RED

Connection Name: Data Source

A unique name that identifies this connection

Data Source Name: red_target (64-bit)

A Data Source Name created with the ODBC admin tool

User Name: postgres

Password: ●●●●●●●●●●

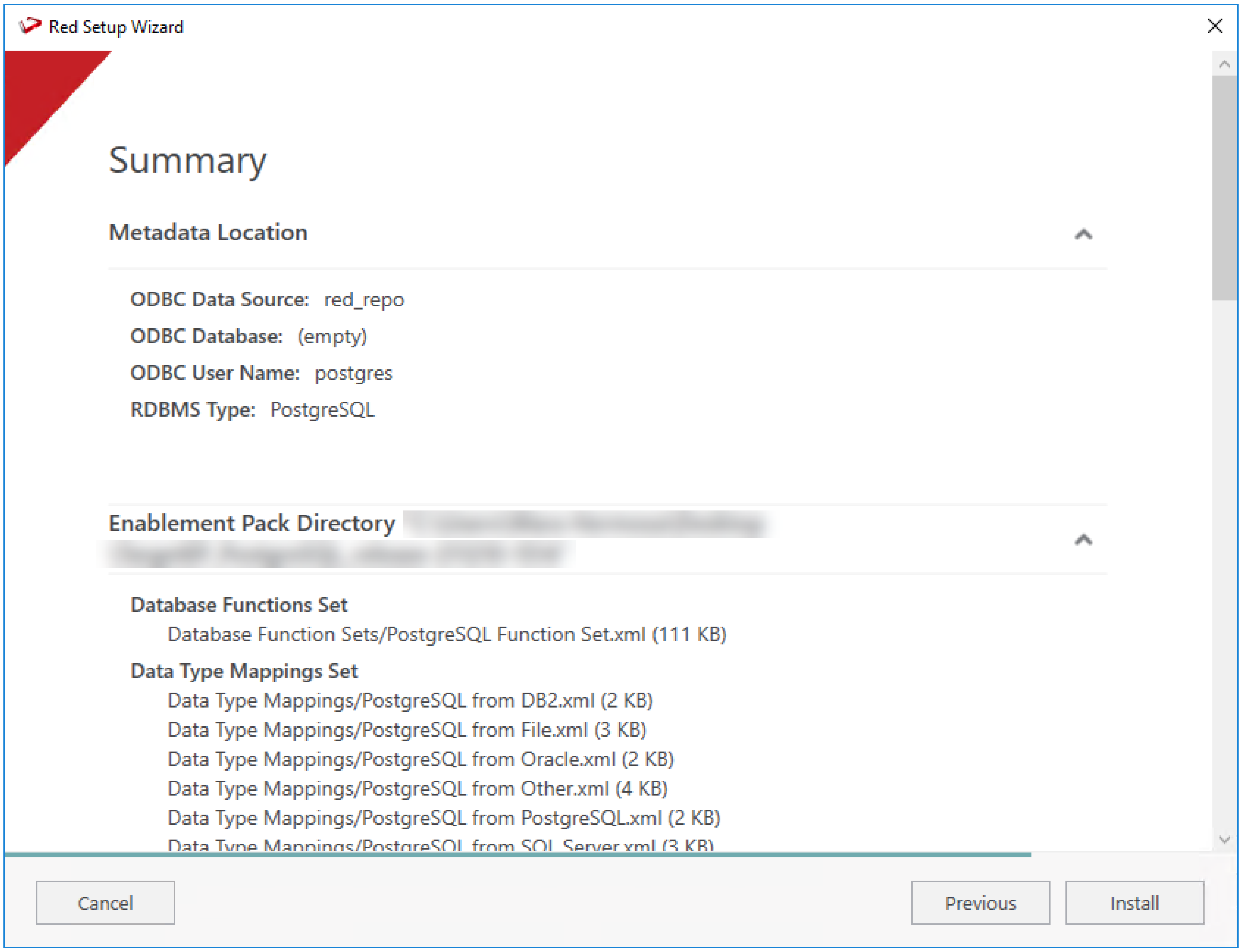
Validated OK

Added Sources

Type in the required data source connection details:

1. Add a name to your connection. Typically similar or derived from the ODBC Source Name (DSN) for quick identification.
2. The ODBC data source name (DSN) for the data source connection. The OBDC Data Source Name must be defined in the Windows **ODBC Data Source Administrator** to appear in this drop-down field.
3. Enter the log in **User Name** and **Password** to connect to the data source.
4. Click **Validate**to continue.
5. After the validation is completed, click **Add** to add several datasources, they are displayes in the right pane of the window. Click **Next** to continue.

9. In the **Summary** window you can review all the details about the **Metadata Location**, **Repository Settings**, and **Enabled Database** settings.



Summary

Metadata Location

ODBC Data Source: red_repo

ODBC Database: (empty)

ODBC User Name: postgres

RDBMS Type: PostgreSQL

Enablement Pack Directory [redacted]

Database Functions Set

Database Function Sets/PostgreSQL Function Set.xml (111 KB)

Data Type Mappings Set

Data Type Mappings/PostgreSQL from DB2.xml (2 KB)
Data Type Mappings/PostgreSQL from File.xml (3 KB)
Data Type Mappings/PostgreSQL from Oracle.xml (2 KB)
Data Type Mappings/PostgreSQL from Other.xml (4 KB)
Data Type Mappings/PostgreSQL from PostgreSQL.xml (2 KB)
Data Type Mappings/PostgreSQL from SQL Server.xml (3 KB)

Click **Install** to proceed with the installation.

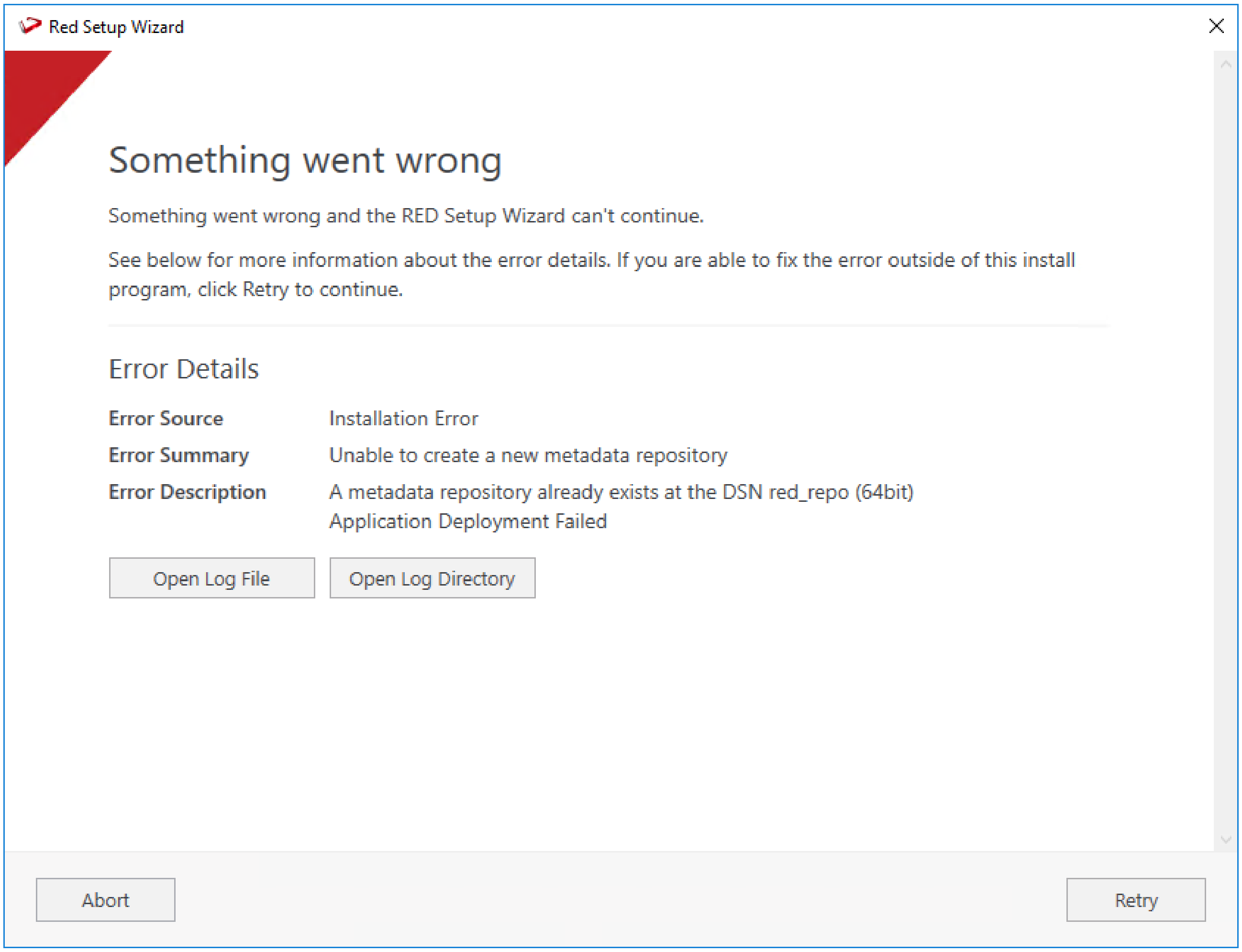
Notes:

- Clicking **Previous** enables you to step back through the Setup Wizard screens and edit the setting you have currently defined.

Once the installation is complete, a message is displayed to confirm that the installation was successful.
10. Click **Finish** to launch RED and start working with your data warehouse.

Note:

If the installation process was not successful, a message is displayed showing the error details. You can try to fix the problem using the error information and then click **Retry** to continue the installation.



Something went wrong

Something went wrong and the RED Setup Wizard can't continue.

See below for more information about the error details. If you are able to fix the error outside of this install program, click Retry to continue.

Error Details

Error Source	Installation Error
Error Summary	Unable to create a new metadata repository
Error Description	A metadata repository already exists at the DSN red_repo (64bit) Application Deployment Failed

RED Scheduler Installer

Installing the Scheduler

WhereScape maintains and distributes a customized version of Azkaban for the scheduling and orchestration of data warehouse workloads with RED. WhereScape's customized version includes support for PostgreSQL as opposed to the MySQL version available publicly. This version of Azkaban is currently only available and distributed with WhereScape RED. WhereScape's version of Azkaban is based on the Azkaban 3.x release branch. For the most part the publicly available Azkaban documentation still applies to WhereScape's version of Azkaban excluding the MySQL portions and any features beyond the 3.x version.

Note:

For more information, visit the **Azkaban official documentation**.

The WhereScape Azkaban Scheduler is made up of a single Azkaban Web Server and one or more Azkaban Executor Servers. Both of these components can run on either Windows or Linux depending on your preference for environments and the job types which are run.

You can have both Windows and Linux Azkaban Executors in your WhereScape Azkaban Scheduler ecosystem to allow for running a wide variety of jobs across your data warehouse. Every executor has a set of accepted 'tags' that allow you to tag certain job types to particular executors.

Installing the WhereScape Azkaban scheduler components on Windows is the easiest way to get started with Azkaban for RED, the Linux installation is more advanced and requires some knowledge of Linux and package management in your particular Linux distribution which is not covered in this install guide.

Before you begin

Common Prerequisites

1. An existing RED metadata repository to associate to the Azkaban Scheduler to (only one scheduler per RED repository is permitted.)
2. A PostgreSQL compatible database for the Azkaban metadata.

   Note:

   Database name must be in lowercase only and not contain any special characters that would require surrounding it with double quotes (") in queries.

   RED and Azkaban metadata can coexist in the same database as they use different schemas, ‘red’ and ‘white’ respectively, but in production environments it is recommended to keep them in separate databases for performance and administrative reasons.

3. PostgreSQL tools installed (psql,pg_dump,pg_restore.)

Next, you can find more information of how to install the Scheduler in Windows and Linux.

Windows RED Scheduler Installation

Windows Prerequisites

1. PostgreSQL ODBC driver installed and an ODBC DSN created for each of the RED and Azkaban metadata databases. PostgreSQL Unicode(x64) driver preferred.

Windows Installation

To install the WhereScape Azkaban Scheduler on a Windows machine you must first install RED then run the **RedSchedulerInstaller.ex**e from the installation directory.

1. Go to the installer path **C:\Program Files\WhereScape\RED**, right-click **RedSchedulerInstaller**, and select **Run as Administrator**.
2. The Scheduler Installation screen opens. Select all the installation options: **Install Scheduler**, **Install Executor**, and **Add Scheduler Configuration to RED**, and click **Next**.
3. In the Install Location screen, select the Installation Location. It is recommended to create a folder under the C: disk to install Azkaban. Click **Next**.
4. The Scheduler Metadata screen shows the information of the database that contains the scheduler metadata. Fill out the **Database Name**, the **User Name**, and the **Password** of the database you created for Azkaban (Follow the same steps you follow to create the PostgreSQL Database).

   Remember that the Database Server Port is 5432, by default.
5. Click **Validate** to confirm that all the entered information is correct. Click **Next**.
6. Enter the **RED Metadata database** information. Enter the **Database Name**, the **User Name**, and the **Password**. Click **Validate** to confirm all the information is correct, and click **Next.**
7. Enter the Scheduler Metadata information to upgrade or install it. These credentials are for an administrative database user with permission to perform DDL operations (If required) and click **Next.**
8. The new Scheduler runs as a web server. The Web Server Configuration screen is populated automatically, make sure that the option **Install as Windows Service** is selected before you move on.
9. In the **Executor Configuration**, click **Add an Executor Tag…**, enter the **localhost**, and click **OK**. Click **Next** to continue with the installation.
10. Under the **Email Configuration**, enter the values you need to configure the email notifications. Click **Next.**

11. A window pops up to select the **Data Source Name** from the drop-down menu which is the RED Metadata database that you created previously, its credentials, **Validate** the data, and click **Login**.
12. The **Add Scheduler Configuration** to WhereScape RED window shows the information on how the new Scheduler will be accessed. Click **Next**.
13. In the **Summary** window, review all the information about your installation and click **Install**.
14. When the installation is complete, click **Finish** to close the confirmation window.

Starting and stopping the Azkaban Servers

Windows

Each Azkaban Server service starts after installation and set to auto-start at Windows startup.

Open **Services.msc** in Windows to find your Azkaban services to **Start/Stop/Restart** them manually or change the login user.

Linux RED Scheduler Installation

Linux Prerequisites

1. Java Runtime Environment installed and on the path
2. PuTTY or equivalent tools for Windows (for remote installation and interactive task execution)

Linux Installation

Currently there is no interactive install wizard for WhereScape Azkaban on Linux but the azkaban-installer.jar file in the RED install directory under `<install_dir>\Azkaban\` can be transferred to linux and run directly using java. This process requires a pre-prepared answer file to be supplied with the parameters for the installation.

Azkaban Installation

1. Install the Azkaban Metadata
2. Install the Azkaban Web Server
3. Install one or more Azkaban Executor Servers

Install the Azkaban Metadata

Perform the Azkaban metadata upgrade by executing the below commands in your command shell in Linux or Windows.

- First set the environment variables for your Azkaban metadata user and pwd:

  ```
  AZ_DB_USER=myuser
  AZ_DB_PWD=mypwd
  ```
- Then, run the `azkaban-installer.jar` with the ‘`upgrade-schema`’ command:

  ```
  java.exe -jar "azkaban-installer.jar" upgrade-schema --database=AZ_REPO
   --host=PG_HOST --port=5432 --schema=white --username-var=AZ_DB_USER
  --password-var=AZ_DB_PWD
  ```

Install the Azkaban Web Server

Copy the scheduler .jar file from `<RED_install_dir>\Azkaban\azkaban-installer.jar` to the Linux machine that will host the main Web Server and run the following commands (see example response files later in this section):

```
java -jar azkaban-installer.jar --log-file=./install.log install-server --response-file=response.xml
```

Install one or more Azkaban Executor Servers

Copy the scheduler .jar file from `<RED_install_dir>\Azkaban\azkaban-installer.jar` to the Linux machine that will host the each Executor Server and run the following commands (see example response files later in this section):

```
java -jar azkaban-installer.jar --log-file=./install.log install-server --response-file=response.xml
```

Azkaban-installer command line help

The same installer can be run from Windows or Linux using the following command to output the installer command line help:

```
java -jar azkaban-installer.jar --help
```

Which outputs:

```
Usage: azkaban-installer [-hV] [--log-file=FILE] [COMMAND]
   --log-file=FILE   File name (including path) for the log file (default: C:
                          \Users\Red1\AppData\Local\Temp\WsSchedulerInstaller_2023050
                          2_201952_803.log)
  -h, --help             Show this help message and exit.
  -V, --version          Print version information and exit.
Commands:
  example-response-file    Generates an example well-formed XML response file
                             to standard output
  install-server           Installs or upgrades an Azkaban server as described
                             in the response file
  mandatory-executor-tags  Outputs the executor tags that should always be
                             added to executors running on the current machine.
  response-file-schema     Writes the XML schema for the response file to
                             standard output
  test-connection          Tests that a JDBC connection to the specified
                             database can be established.
  upgrade-schema           Upgrades or installs the Scheduler metadata schema.
```

Example response file for an Executor setup:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<installServerResponse>
    <serverType>EXEC_SERVER</serverType>
    <serverName>EXEC_SERVER</serverName>
    <destination>
        <destinationDirectory>\home\AzkabanSchedulerUser\WhereScape\</destinationDirectory>
    </destination>
    <metadataDatabase>
        <databaseName>schedmeta</databaseName>
        <hostName>pgserver</hostName>
        <password>schedpassword</password>
        <port>5432</port>
        <schema>white</schema>
        <userName>scheduser</userName>
    </metadataDatabase>
    <redMetadataDatabase>
        <databaseName>redmeta</databaseName>
        <hostName>pgserver</hostName>
        <password>redpassword</password>
        <port>5432</port>
        <schema>red</schema>
        <userName>reduser</userName>
    </redMetadataDatabase>
    <serverConfig>
```

```
        <executorTags>linux</executorTags>
        <port>8002</port>
        <timeZone>Pacific/Auckland</timeZone>
        <webServerExternalHostName>AzkabanServer</webServerExternalHostName>
        <webServerExternalPort>8001</webServerExternalPort>
        <adminUserName>admin</adminUserName>
        <adminPassword>admin</adminPassword>
        <dashboardUserName>dashboard</dashboardUserName>
        <dashboardPassword>dashboard</dashboardPassword>
        <publishUserName>publish</publishUserName>
        <publishPassword>publish</publishPassword>
    </serverConfig>
    <emailConfig>
        <host>emailhost</host>
        <port>587</port>
        <user>emailuser</user>
        <password>emailpassword</password>
        <sender>senderemail</sender>
        <tls>true</tls>
        <useAuth>true</useAuth>
    </emailConfig>
    <redBindir>\home\AzkabanSchedulerUser\WhereScape\</redBindir>
</installServerResponse>
```

Example Install Command:

Typical installation command assuming the `azkaban-installer.jar` has been copied to the home directory for the running user on Linux:

```
java -jar ~/azkaban-installer.jar --log-file=~/install.log install-server --response-
file=~/response.xml
```

Starting and Stopping the Azkaban Servervs

On Linux the Azkaban Servers are not started automatically after install so you will need to run the startup script.

- `<azkaban_install_dir>/<web_server_name>/azkaban-web-server/bin/start-web.sh`
- `<azkaban_install_dir>/<exec_server_name>/azkaban-exec-server/bin/start-exec.sh`

There are various ways to have these servers start when Linux starts. The simplest is to add it via crontab with the @reboot keyword.

For example:

`crontab -e` Opens crontab editor for the current user

Add a line for each server, save, and close the editor:

- `@reboot <azkaban_install_dir>/<web_server_name>/azkaban-web-server/bin/start-web.sh`
- `@reboot <azkaban_install_dir>/<exec_server_name>/azkaban-exec-server/bin/start-exec.sh`

`crontab -l` Lists the current users' cron jobs.

Add the Scheduler Configuration to RED

The Scheduler configuration can be added to RED via the RedSetupWizard.exe or manually through the RED UI in the Scheduler tab.

Azkaban Configuration

Any changes to the settings in this file will not take until the Azkaban Web or Executor Server is restarted.

Azkaban Users

If you require additional users for the Azkaban dashboard or API, other than the default users, then they can be added by the following process:

Create Azkaban accounts by editing the azkaban-users.xml file in the Web Server install directory

`<web_server_dir>\<server_name>\azkaban-users.xml`

For example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<azkaban-users>
  <role name="admin" permissions="ADMIN"/>
  <role name="read" permissions="READ"/>
  <role name="executor" permissions="EXECUTE"/>
  <user username="admin" password="admin" roles="admin"/>
  <user username="readonly" password="readonly" roles="read"/>
  <user username="executor" password="executor" roles="executor,read"/>
</azkaban-users>
```

The possible role permissions are the following:

Permissions	Values
ADMIN	Grants all access to everything in Azkaban.
READ	Gives users read only access to every project and their logs
WRITE	Allows users to upload files, change job properties or remove any project
EXECUTE	Allows users to trigger the execution of any flow
SCHEDULE	Users can add or remove schedules for any flows
CREATEPROJECTS	Allows users to create new projects if project creation is locked down

Properties

The main properties file ‘azkaban.local.properties’ for Azkaban Servers is located in the root folder of the Web or Executor Server installation directory.

Work Directory

Property: wherescape.job.workdir

Description: This setting is not present by default but can be added and set to an existing directory where all the WhereScape job work files will be created. Useful in dev environments where you have multiple schedulers running on a single machine and want to avoid conflicting temporary files.

Defaults to the following locations if the setting is not present or empty:

- Linux defaults to /tmp
- Windows defaults to defined temp directory of the user (for the system user this will be C:\Windows\Temp)

Bin Directory

Property: wherescape.red.bindir - this setting allows you to set the path that will be returned by the WSL_BINDIR environment variable in scripts.

Upgrading WhereScape RED

WhereScape Versions

WhereScape RED has a four-part version number normally shown as xx.xx.xx.xx. An example of this may be 10.1.0.0. The first number represents the major release. The second number represents the metadata repository version. The third and fourth numbers relate to application specific releases.

From the example above, we see that the current version is version 10of WhereScapeRED. We are on version 10.1 of the metadata repository.

Metadata Changes

A change from a 10.0 release to a 10.1 release would indicate a change in the metadata tables. All metadata changes are non-destructive. They simply add new columns or new meta tables. In this way, they can be applied without harming an existing metadata repository. The impact of a metadata change is that the associated applications (namely the REDUI and RedCli executables) needs to be at the same metadata version. Therefore, an 10.0.0.0 version of RED may not successfully run against a version 10.1 metadata repository.

Application changes

The final two numbers in the version represent application releases. Applications are deemed to be all of the executable images supplied with WhereScapeRED and the stored procedures. Application changes reflect enhancements and bug fixes. A change in the third number indicates a major enhancement in one of the application components and a change in the fourth number is a patch fix.

Upgrading RED

Notes:

- Ensure that you have made a backup your metadata before performing the upgrade.
- Ensure you also upgrade the WhereScapeRED Scheduler if the release notes indicates a Scheduler upgrade is required. See the **Upgrading Azkaban** section for more details.

Upgrading WhereScape RED consists of the following steps:

1. Allow any active jobs to complete. Halting active jobs will allow running tasks to complete with no new tasks starting. Aborting active jobs will kill any running tasks and stop running jobs.
2. Stop all schedulers. Windows schedulers can be stopped via Windows Services. To stop a UNIX or Linux scheduler, kill the active scheduler process.
3. Close any WhereScape programs that are running on your machine.
4. Back up your metadata.
5. Install the new version of RED on your machine.
6. Run the `RedMetadataValidate.exe` found in the installation directory of RED. This function may re-compile existing or create new metadata procedures; metadata tables may be altered, or new tables created.
7. When performing an upgrade of RED:
   - Click **OK** when prompted to validate metadata tables.
   - Click **Yes** when prompted to re-create metadata views.
   - Click **Yes** when prompted to re-compile metadata procedures.

   Warning

   If the above steps are not all completed during an upgrade, the metadata may be left in an inconsistent state.

8. Restart all schedulers. Windows schedulers can be restarted via Windows Services. For any UNIX or Linux schedulers, these can be restarted by running the start-up script in the bin directory of each Azkaban Web Server or Executor.
9. Restart any halted or aborted jobs.

Notes:

i. Metadata tables do not change between minor releases, but metadata procedure often and usually do change.
ii. RED will not let you sign into an old repository version using a newer version of RED.
iii. RED will let you sign into a new repository version using an older version of RED but it will warn you that this may potentially cause issues.
iv. Side by side installations are possible (two versions in two directories on the same machine), but be careful with schedulers. If you install the new version of RED in a new directory, you must update the registry for each service to use the new version of RED.See **Azkaban Upgrade** troubleshooting section.

Upgrading Azkaban

Typical upgrade steps

For each Azkaban Web and Executor Server instance:

1. Stop the Windows services and/or Linux processes for each Web and Exec server
2. Perform backups of the following:
    - Your Azkaban metadata repository
    - `azkaban.local.properties` for each Web and Exec server
    - Any other configuration files if you have customized them
3. Run the installer targeting the existing Azkaban install locations
4. Merge any customizations back in from your earlier backups
5. Start the Azkaban Web and Executor services and/or Linux processes

Backup your Azkaban Properties and Configuration files

The installer for Azkaban overwrites the properties and configuration files during the upgrade process. The first step is to make a copy of any properties and configuration files to merge your custom settings back in after the upgrade.

Backup these files (or the entire folder) for each Web Server and Executor Server being upgraded

- `azkaban.local.properties`
- `azkaban-users.xml`
- `log4j.properties`

The backup of the `azkaban.local.properties` file will be useful to open while you run the installer so you can apply the correct settings to the installer fields.

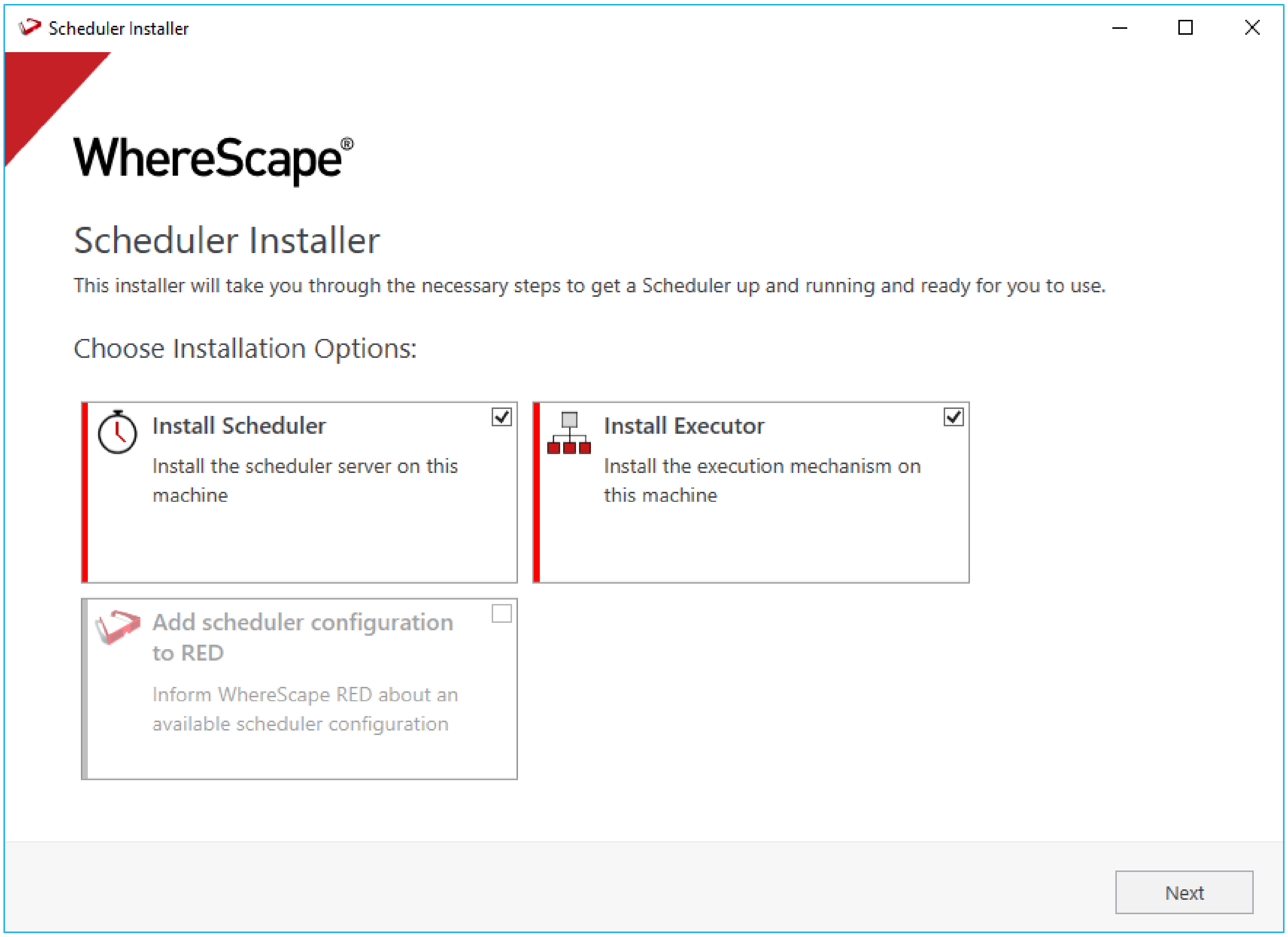
Upgrade on Windows

1. Run `RedSchedulerInstaller.exe` to start the upgrade process to follow the interactive installer wizard.

    Notes:

    For scripted upgrades, the Linux upgrade method described later can also be used on Windows.

2. To upgrade the Azkaban metadata and Web Server on this machine, select **Install Scheduler** and un-check **Add scheduler configuration to RED** as this configuration has already been added to RED during the original install. If you want to upgrade an Executor on this machine, select **Install Executor**, as required.



Scheduler Installer

This installer will take you through the necessary steps to get a Scheduler up and running and ready for you to use.

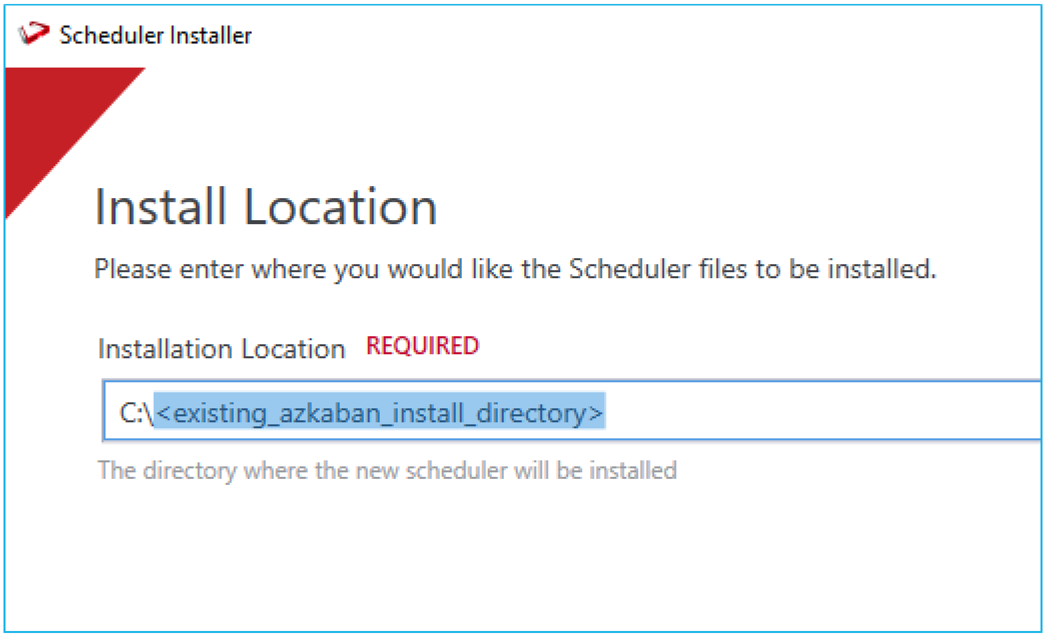
Choose Installation Options:

☑ Install Scheduler
Install the scheduler server on this machine

☑ Install Executor
Install the execution mechanism on this machine

□ Add scheduler configuration to RED
Inform WhereScape RED about an available scheduler configuration

3. Choose the existing Azkaban installation location to upgrade.



Install Location

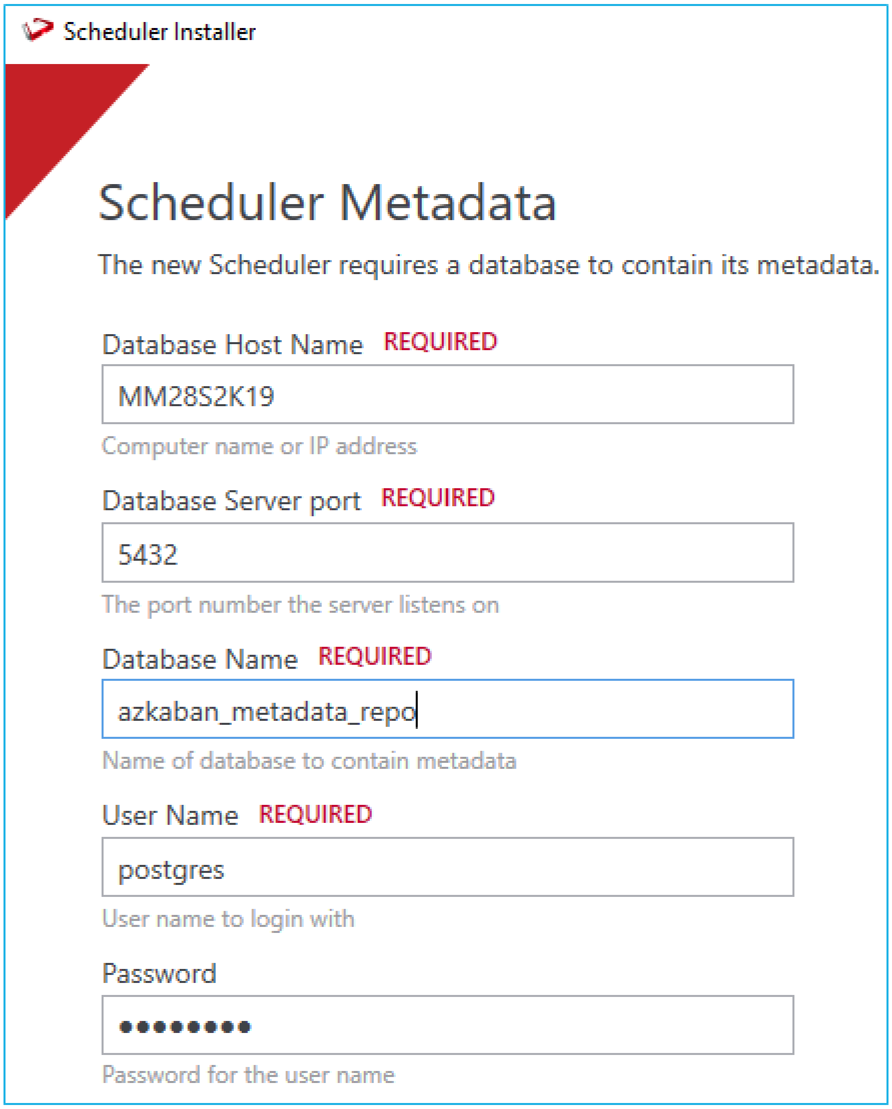
Please enter where you would like the Scheduler files to be installed.

Installation Location REQUIRED

C:\<existing_azkaban_install_directory>

The directory where the new scheduler will be installed

4. Provide the existing RED and Azkaban metadata databases. If you are unsure of the ports and users of your existing servers you can open the `azkaban.local.properties` file you backed up earlier and extract those details from it.



Scheduler Installer

Scheduler Metadata

The new Scheduler requires a database to contain its metadata.

Database Host Name REQUIRED

MM28S2K19

Computer name or IP address

Database Server port REQUIRED

5432

The port number the server listens on

Database Name REQUIRED

azkaban_metadata_repo

Name of database to contain metadata

User Name REQUIRED

postgres

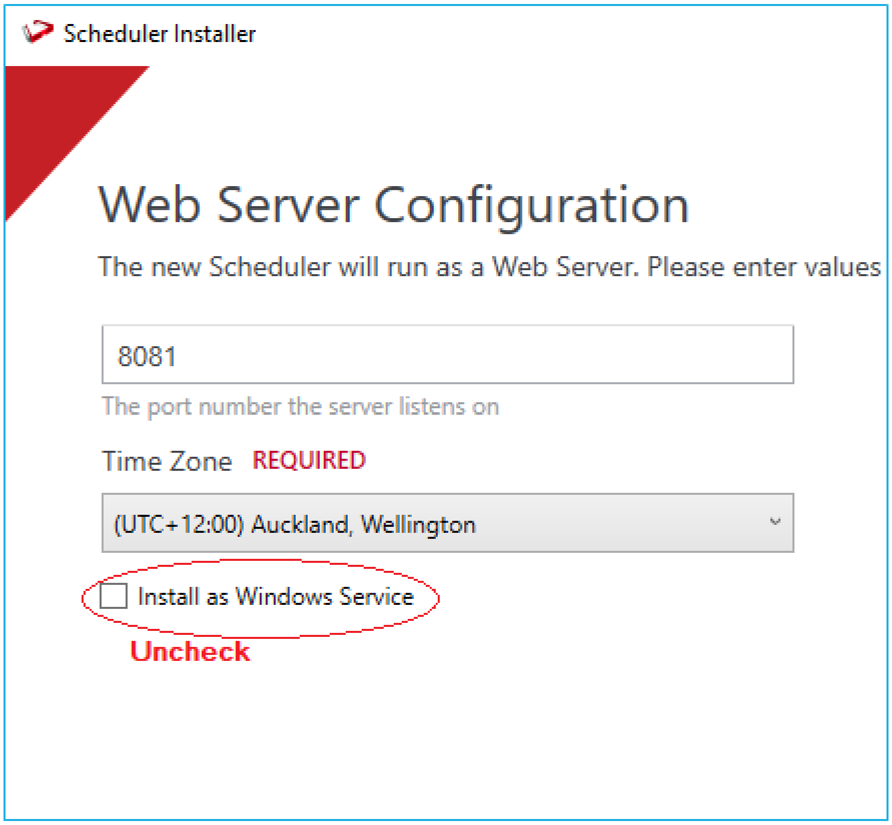
User name to login with

Password

●●●●●●●●

Password for the user name

5. Enter your existing server details into the **Web Server Configuration** and **Exec Server Configuration** screens ensure you have unchecked the options **Install as Windows service** as these were already put in place in the original installation.



Scheduler Installer

Web Server Configuration

The new Scheduler will run as a Web Server. Please enter values

8081

The port number the server listens on

Time Zone REQUIRED

(UTC+12:00) Auckland, Wellington

□ Install as Windows Service

Uncheck

6. Proceed through the rest of the installation wizard and click **Install** at the end to complete the upgrade of the Azkaban metadata and the Web and Exec Servers.
7. Post install you need to manually merge any customizations to the `azkaban.local.properties` and users file as required before starting up your services again.

Upgrade on Linux

1. Copy the scheduler .jar file from `<RED_install_dir>\Azkaban\azkaban-installer.jar` to each Linux machine that has either the Web Server or an Executor installed.
2. Perform the Azkaban metadata upgrade by executing the below commands in your Linux command shell.

- First set the environment variables for your Azkaban metadata user and pwd:

```
AZ_DB_USER=myuser
AZ_DB_PWD=mypwd
```

- Then run the `azkaban-installer.jar` with the `upgrade-schema` command:

```
java.exe -jar "azkaban-installer.jar" upgrade-schema --database=AZ_REPO --host=PG_HOST --
port=5432 --schema=white --username-var=AZ_DB_USER --password-var=AZ_DB_PWD
```

- Perform the application upgrade for each Azkaban Web or Executor Server instance.

```
java -jar azkaban-installer.jar --log-file=./install.log install-server --response-
file=response.xml
```

Post install you need to manually merge any customizations to the `azkaban.local.properties` and users file as required before starting up your services again.

The original response file for the server should still exist in each server install directory, this can be reused to reapply the same settings as before. If you have removed or deleted this file then a new one will need to be reconstructed. See the Azkaban Installation section for an answer file example.

Upgrade the Scheduler Integration Scripts

Some releases add improvements to the shipped Azkaban integration scripts, any such changes will be signaled in the release notes.

Due to specific environmental customizations users may have made to these scripts they are not upgraded automatically, instead you need to extract them from the installation folder and review and merge the changes to your existing scripts as required.

The shipped scheduler integration scripts are described below:

- `wsl_scheduler_publish` - publishes job metadata to Azkaban
- `wsl_scheduler_lookup` - retrieves the tags of all active Azkaban Executors
- `wsl_scheduler_dashboard` - opens the Azkaban Web Server dashboard

To upgrade any of these scripts follow this process:

1. In RED, open the Azkaban integration script in the script editor such as `wsl_scheduler_publish` under your Host Script objects.
2. In notepad open the matching script located in the RED installation directory under `<RED install directory>\Administrator\Scripts\wsl_scheduler_publish.ps1`
3. Copy the contents of the updated script into the RED script editor to overwrite the existing script and Save.

Notes

• It is a good idea to version the script before the manual upgrade process, right click on the script in RED and select ‘Version Control->New Version’

• If you have made any customizations to your wsl_scheduler_publish script these will need to be merged manually with the updated script changes.

• Any changes to the Azkaban integration scripts will only take effect for new jobs or when editing an existing job causing a re-publish to Azkaban.

Upgrade Troubleshooting

Upgraded RED and now Azkaban doesn't start?

If you installed RED to a different folder than when the Windows Azkaban Web Server or Executors were originally created then you need to update the `ImagePath` key in the registry for each Azkaban service.

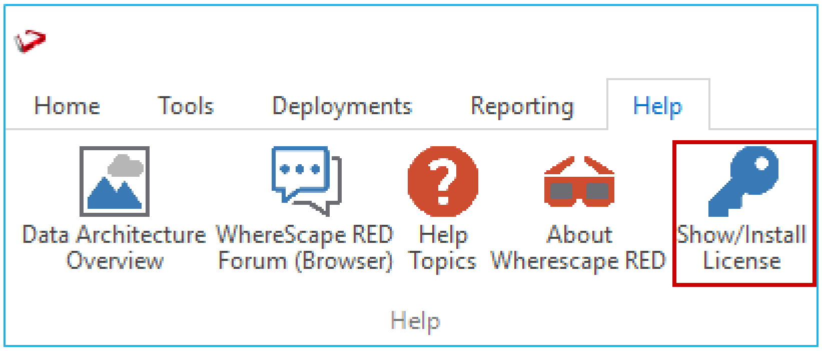
For example:

- Registry location: `Computer\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\WsAzkabanExec`
- Key: `ImagePath`
- Value: `C:\Program Files\WhereScape\RED\10001-230606-005456\WslSchedulerService.exe" --type="Exec" --scheduler-dir="C:\temp\azkaban\red1001_db" --start-service=true --service-name="WsAzkabanExec_red1001_db`

RED License Management Overview

WhereScape RED provides a license management tool that enables you to install a new license and validate or maintain existing RED licenses.

This tool is launched from the **Help** menu:



The stand alone application (**RedInstallLicense.exe**) can also be run from the RED Windows Installation directory.

This tool is also used in the RED **Setup Wizard** to install or validate licenses, prior to installing a metadata repository to work with a new version of RED. Refer to the RED Setup Wizard User Guide for details.

The succeeding sections describe the steps for using the RED license management tool.

Note:

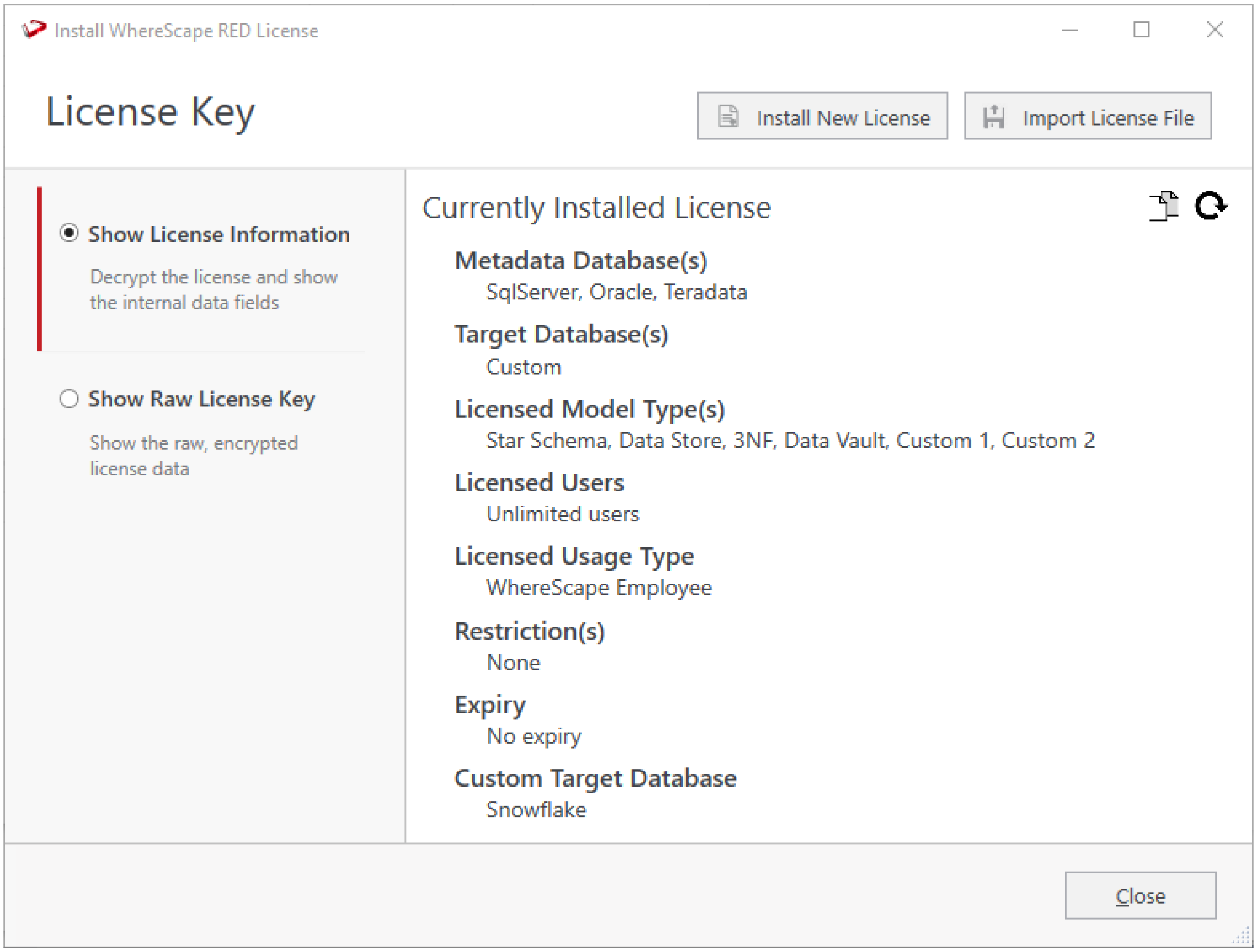
The **REDCLI** command line interface also enables you to validate a metadata repository. Refer to the REDCLI section of the RED User Guide for details.

Installing a License

The following describe the steps for installing your WhereScape RED license:

1. Log into RED and click the **Help** menu to select the **Show/Install License** option.
   Also, open the RED Windows installation directory and then double-click the **RedInstallLicense.exe** application to launch the license management tool.
2. The **License Key** window is displayed showing your License information, if you already installed RED before.
   If you are installing RED for the first time, you need to install your license information by clicking **Install**

New License.



● Show License Information

Decrypt the license and show the internal data fields

○ Show Raw License Key

Show the raw, encrypted license data

Currently Installed License

Metadata Database(s)
SqlServer, Oracle, Teradata

Target Database(s)
Custom

Licensed Model Type(s)
Star Schema, Data Store, 3NF, Data Vault, Custom 1, Custom 2

Licensed Users
Unlimited users

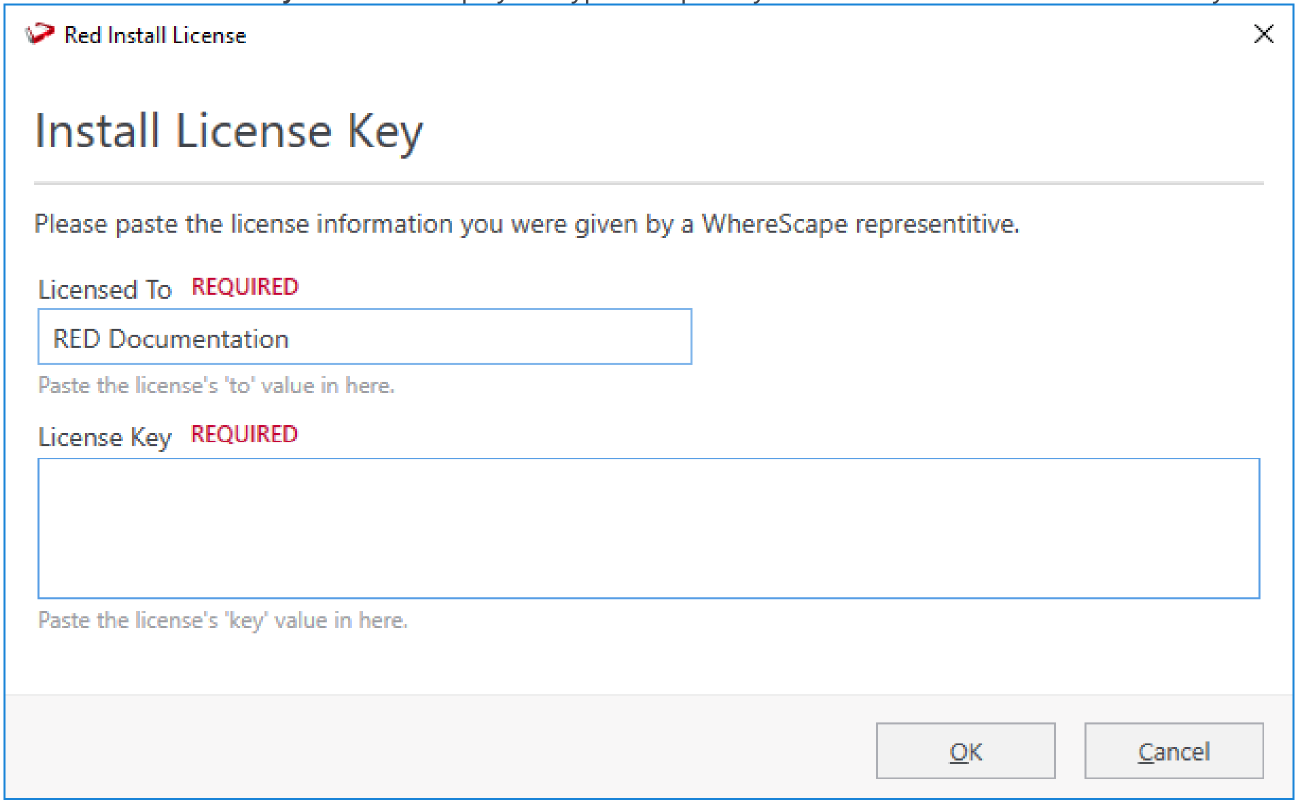
Licensed Usage Type
WhereScape Employee

Restriction(s)
None

Expiry
No expiry

Custom Target Database
Snowflake

3. The **Install License Key** window is displayed. Type in or paste your license information and license key.



Please paste the license information you were given by a WhereScape representitive.

Licensed To REQUIRED

RED Documentation

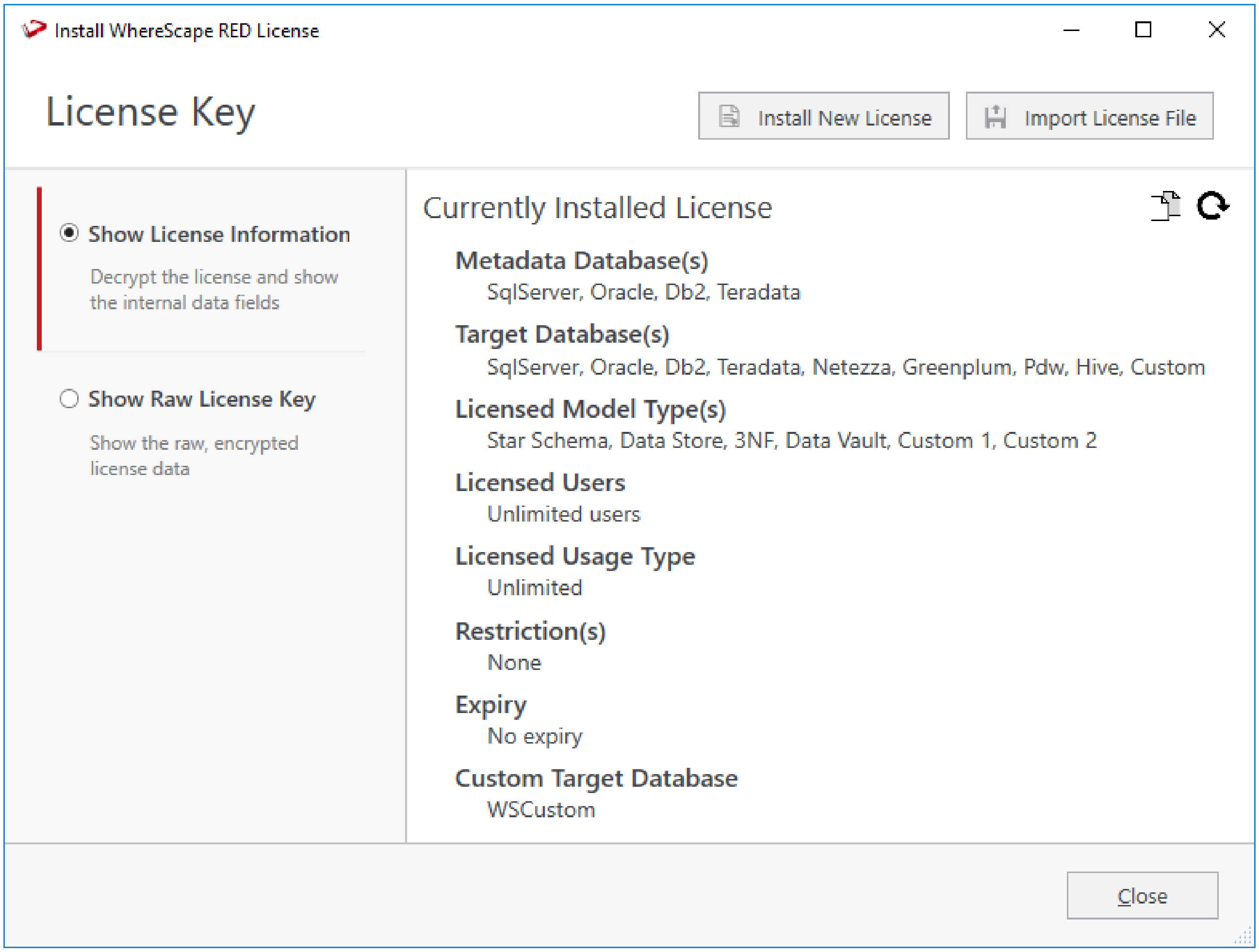
Paste the license's 'to' value in here.

License Key REQUIRED

Paste the license's 'key' value in here.

Alternatively, you can click **Import License File** to browse to a directory and select a License file to load in RED.

4. Click **OK** to continue. The **License Key** window is displayed showing your currently installed license information.



● Show License Information
Decrypt the license and show the internal data fields

○ Show Raw License Key
Show the raw, encrypted license data

Currently Installed License

Metadata Database(s)
SqlServer, Oracle, Db2, Teradata

Target Database(s)
SqlServer, Oracle, Db2, Teradata, Netezza, Greenplum, Pdw, Hive, Custom

Licensed Model Type(s)
Star Schema, Data Store, 3NF, Data Vault, Custom 1, Custom 2

Licensed Users
Unlimited users

Licensed Usage Type
Unlimited

Restriction(s)
None

Expiry
No expiry

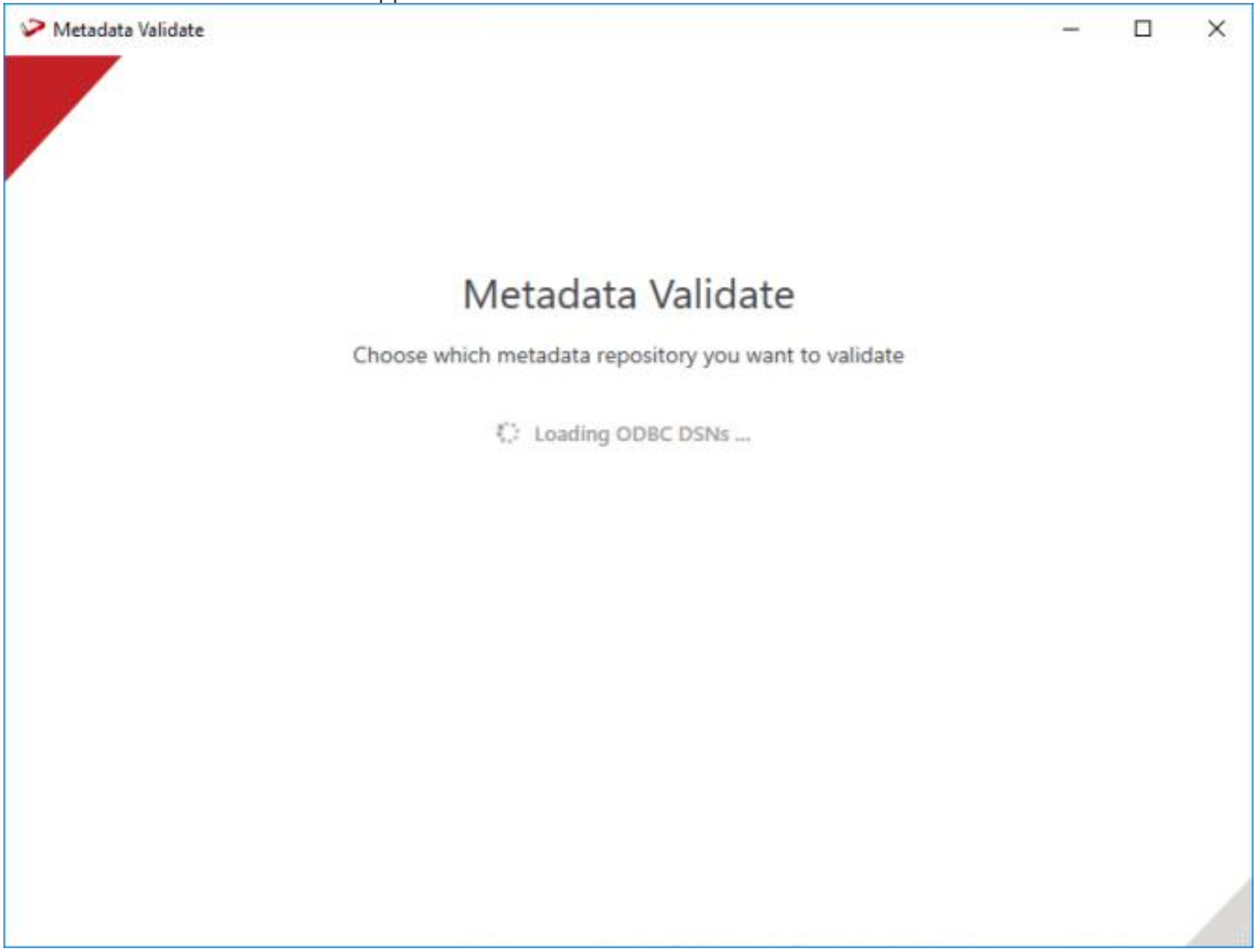
Custom Target Database
WSCustom

5. Click **Close** to exit the license management tool.

RED Metadata Validate Wizard

The following steps describe how to validate metadata inRED.

1. After successfully installing WhereScape RED, open the RED Windows installation directory, double-click the **RedMetadataValidate.exe** application to launch the **Metadata Validate Wizard**.



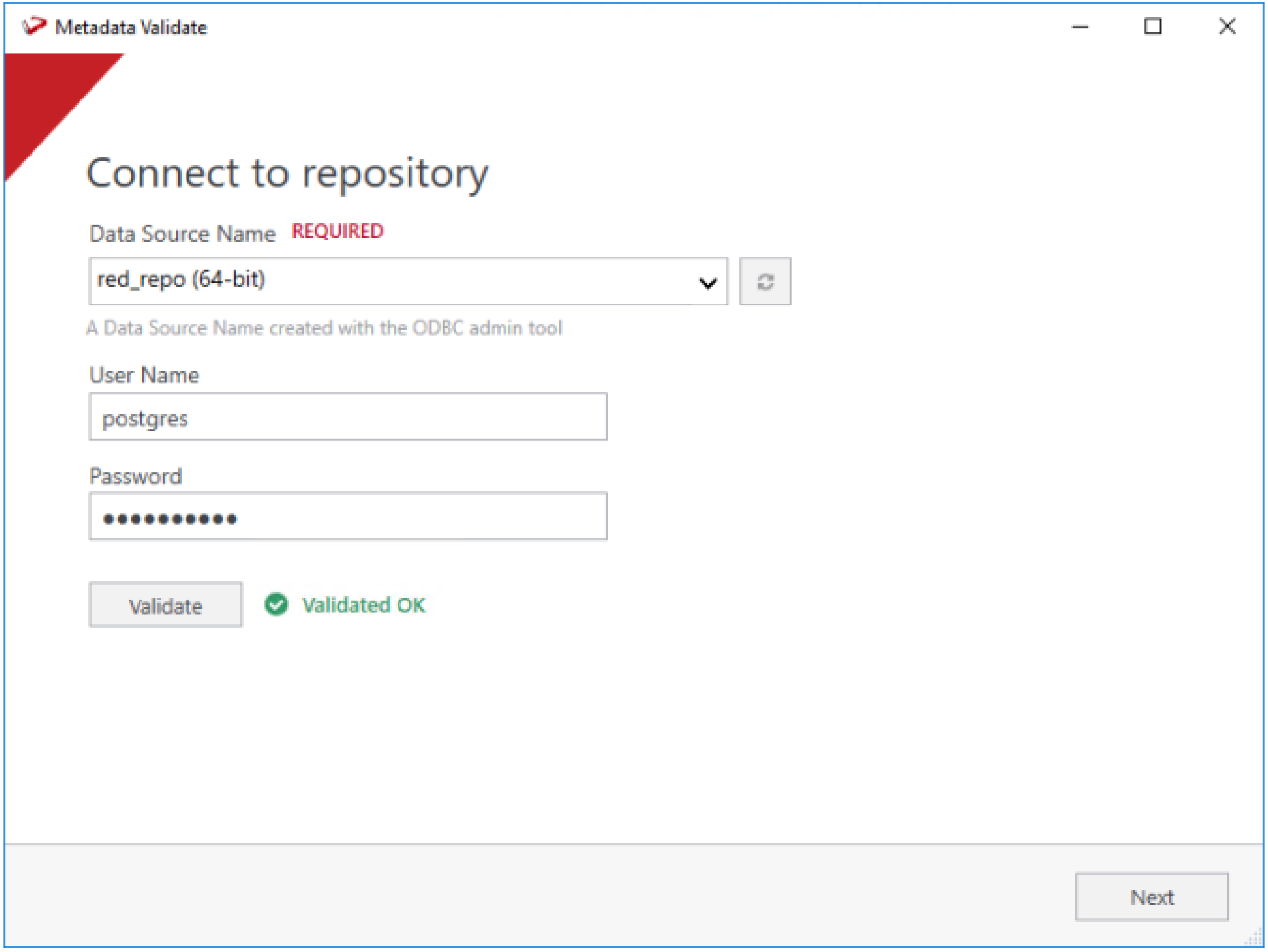
Metadata Validate

Choose which metadata repository you want to validate

Loading ODBC DSNs ...

2. When the previous window finishes loading, the **Connect to repository** screen displays. Click the drop-down to select the Data Source Name. Enter your credentials, **User Name** and **Password** to validate your

connection, and click **Next**.



Connect to repository

Data Source Name REQUIRED

red_repo (64-bit)

A Data Source Name created with the ODBC admin tool

User Name

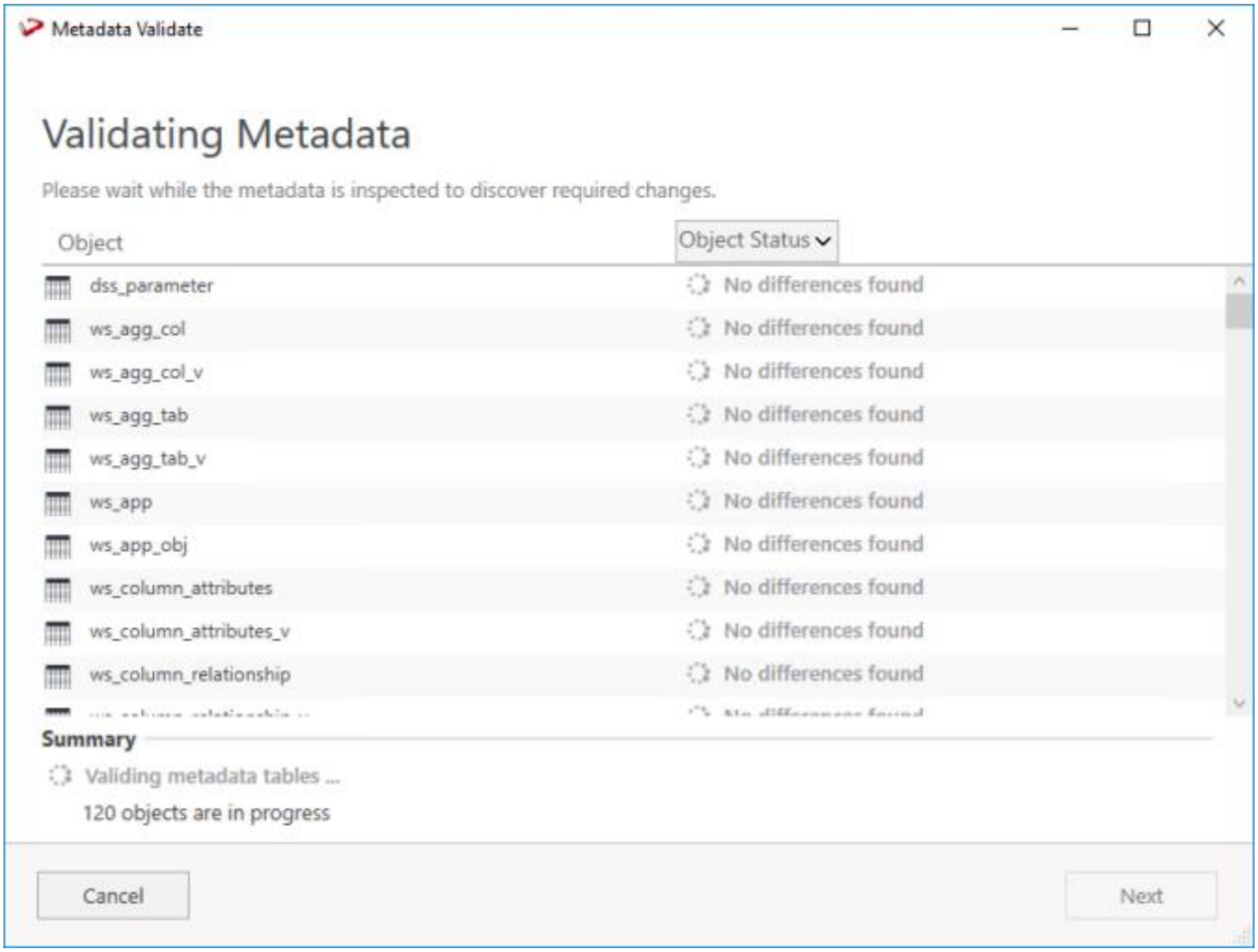
postgres

Password

●●●●●●●●●●●

✓ Validated OK

3. RED starts to validate metadata.



Validating Metadata

Please wait while the metadata is inspected to discover required changes.

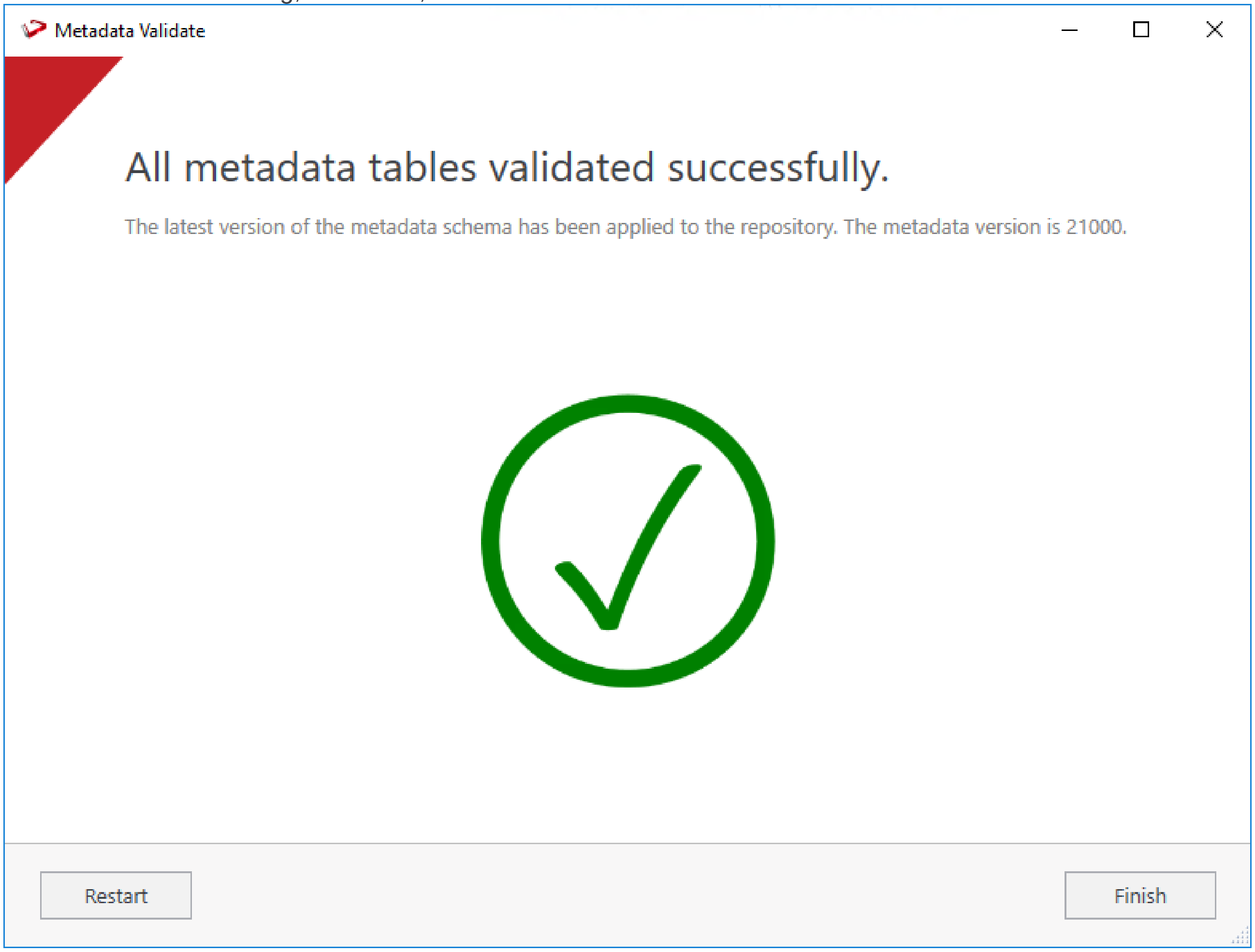
Object	Object Status
dss_parameter	No differences found
ws_agg_col	No differences found
ws_agg_col_v	No differences found
ws_agg_tab	No differences found
ws_agg_tab_v	No differences found
ws_app	No differences found
ws_app_obj	No differences found
ws_column_attributes	No differences found
ws_column_attributes_v	No differences found
ws_column_relationship	No differences found

Summary

Validing metadata tables ...

120 objects are in progress

4. After the validation is complete, the confirmation message of your successful validation displays. Click **Finish** to close the dialog, otherwise, click **Restart** to start over.



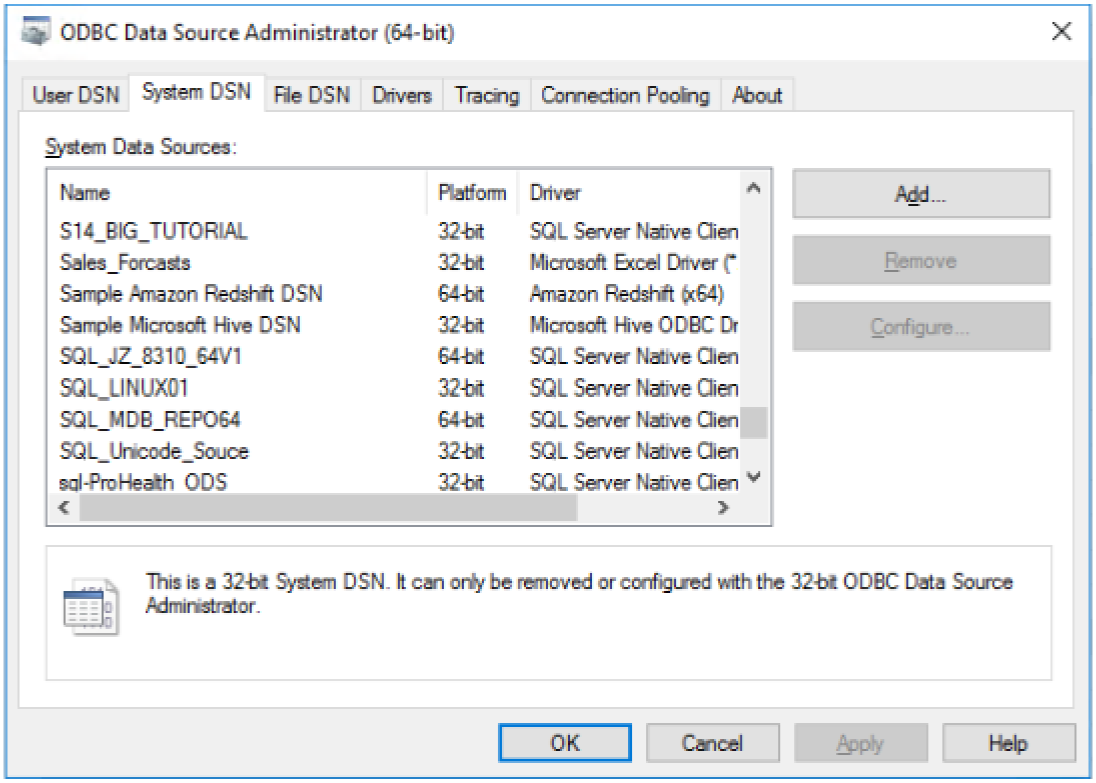
All metadata tables validated successfully.

The latest version of the metadata schema has been applied to the repository. The metadata version is 21000.

Creating an ODBC Connection

To create an ODBC Connection:

1. Click the **System DSN** tab to view the system wide ODBC entries.



System Data Sources:

Name	Platform	Driver
S14_BIG_TUTORIAL	32-bit	SQL Server Native Clien
Sales_Forcasts	32-bit	Microsoft Excel Driver (*
Sample Amazon Redshift DSN	64-bit	Amazon Redshift (x64)
Sample Microsoft Hive DSN	32-bit	Microsoft Hive ODBC Dr
SQL_JZ_8310_64V1	64-bit	SQL Server Native Clien
SQL_LINUX01	32-bit	SQL Server Native Clien
SQL_MDB_REPO64	64-bit	SQL Server Native Clien
SQL_Unicode_Souce	32-bit	SQL Server Native Clien
sql-ProHealth_ODS	32-bit	SQL Server Native Clien

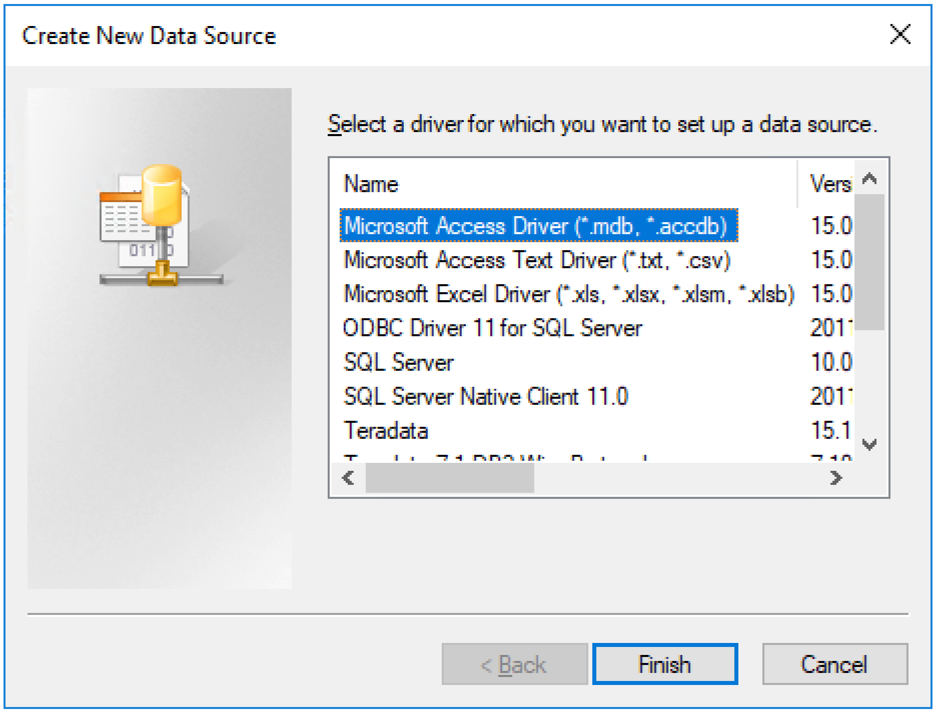
This is a 32-bit System DSN. It can only be removed or configured with the 32-bit ODBC Data Source Administrator.

- Click the **Add...** button to add a new entry.

Note:

ODBC connections must be either User DSN or System DSN. File DSN connections are not supported.

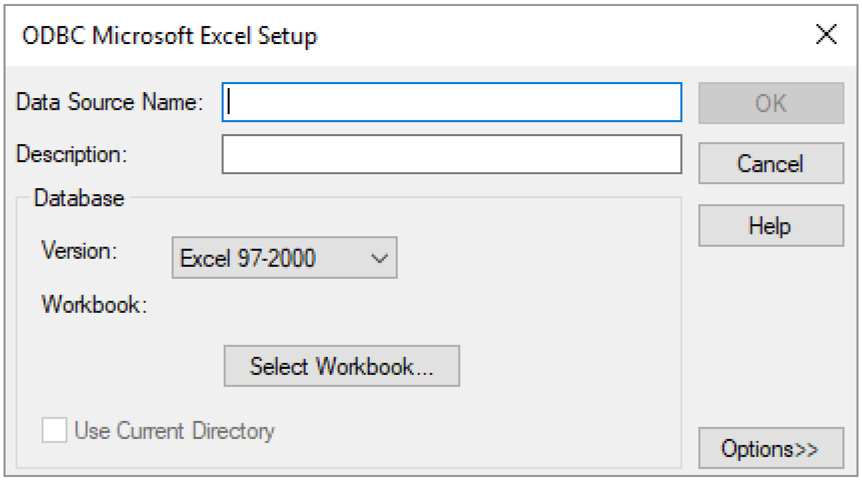
2. From the **Create a New Data Source** screen, select the relevant driver and then click **Finish**.



Select a driver for which you want to set up a data source.

Name	Vers
Microsoft Access Driver (*.mdb, *.accdb)	15.0
Microsoft Access Text Driver (*.txt, *.csv)	15.0
Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)	15.0
ODBC Driver 11 for SQL Server	201
SQL Server	10.0
SQL Server Native Client 11.0	201
Teradata	15.1

The **ODBC Text Setup** dialog is displayed:



3. Enter the relevant details and click the **OK** button to create a new ODBC data source.

PostgreSQL Database and Login Setup

Supported PostgreSQL versions

WhereScape RED and WhereScape Azkaban support PostgreSQL compatible databases from PostgreSQL version 12 to version 15.

Database

Database names for RED and WhereScape Azkaban metadata must be created in lowercase and not contain any special characters which would require surrounding in quotes (“”) in SQL queries. RED creates its metadata on a schema named ‘red’. WhereScape Azkaban creates its metadata on a schema named ‘white’.

Users and permissions

RED Users

Normal users of RED require the following grants on the metadata database:

SELECT, INSERT, UPDATE, DELETE, EXECUTE on the ‘red’ schema of the RED metadata database.

RED Installation User

For metadata installation and upgrade the user requires:

- CREATE on the database in order to create the ‘red’ schema and the RED metadata objects

Note:

Occasionally RED and Azkaban metadata may change between versions and in some cases requires the dropping of existing objects during an object upgrade, therefore the user (or role) used for an upgrade should be the owner of the objects, or a superuser will need to be used.

Azkaban User

The Azkaban user for the Web Server and Executor Servers requires:

- SELECT, INSERT, UPDATE, DELETE, EXECUTE on the ‘white’ schema of the Azkaban metadata database.

Azkaban Installation User

For metadata installation and upgrade the user require:

- CREATE on the database in order to create the ‘white’ schema and the Azkaban metadata objects.

Example RED Database setup steps

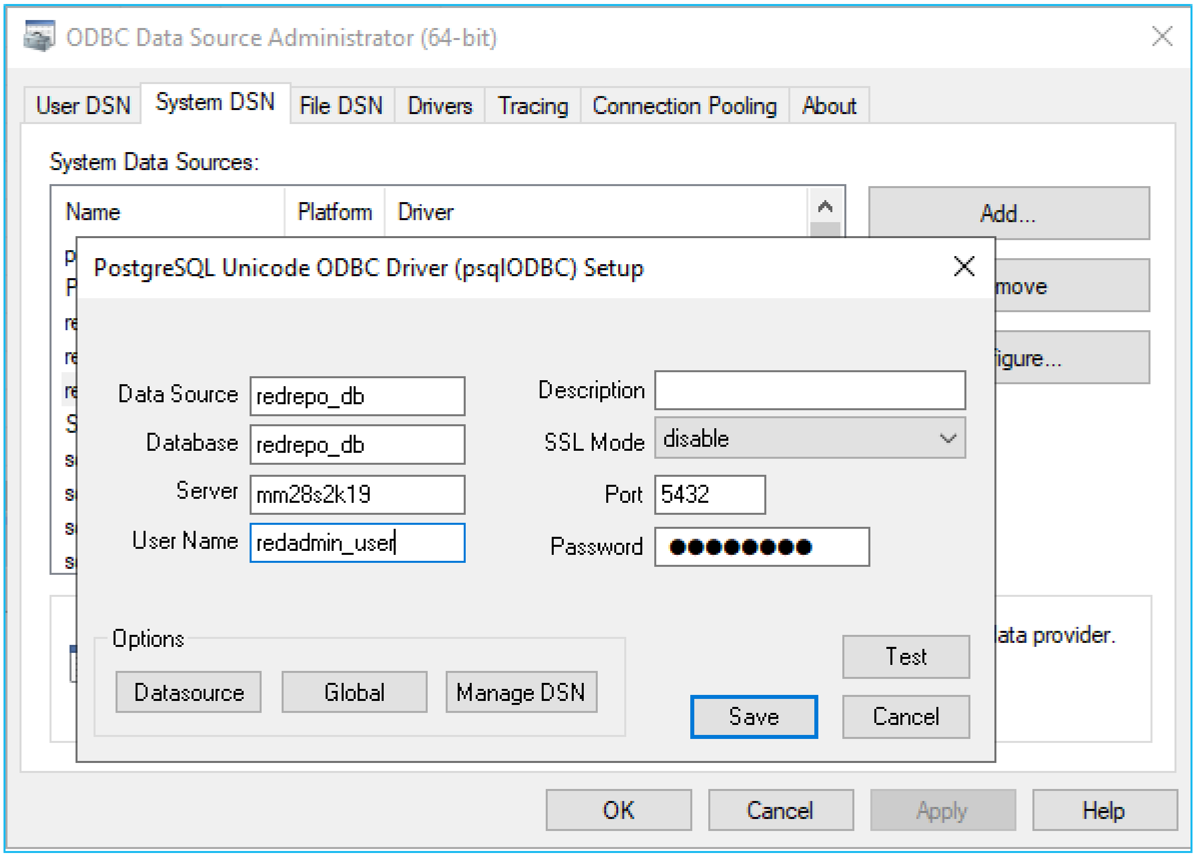
1. Create a database and admin role in PostgreSQL

```
-- Create an admin role, database and grant create

CREATE ROLE redadmin_role NOLOGIN ADMIN postgres;

CREATE DATABASE redrepo_db;

GRANT CREATE ON DATABASE redrepo_db to redadmin_role;
```

2. Create and admin user in PostgreSQL

```
-- Create the admin user

CREATE USER redadmin_user WITH PASSWORD 'mypass';

GRANT redadmin_role to redadmin_user;
```

3. Create an ODBC DSN for the database



PostgreSQL Unicode ODBC Driver (psqlODBC) Setup

Data Source: redrepo_db

Database: redrepo_db

Server: mm28s2k19

User Name: redadmin_user

Description:

SSL Mode: disable

Port: 5432

Password: ●●●●●●●●

4. Run the RedSetupWizard to install the metadata

For more information about the **Red Setup Wizard** in this section.

5. Create a RED user role and user in PostgreSQL

```
-- Note this set of statements assumes you have already created

--  the RED Metadata using the admin user.

-- Create the RED user role and give grants to the metadata objects

CREATE ROLE reduser_role NOLOGIN ADMIN postgres;

GRANT USAGE ON SCHEMA red TO reduser_role;

GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA red TO reduser_role;
GRANT EXECUTE ON ALL PROCEDURES IN SCHEMA red TO reduser_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA red TO reduser_role;



-- Create a RED user and grant the user role

CREATE USER reduser_user WITH PASSWORD 'mypass';

GRANT reduser_role to reduser_user;
```

Applications

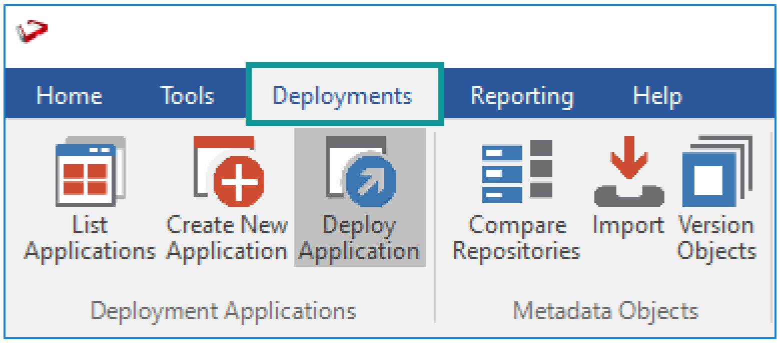
RED Application Deployment Overview

WhereScape RED provides an application deployment tool that enables you to create and deploy applications as a means to move metadata objects between repositories, e.g. from your development environment to your testing or production environment.

RED enables you to build an application which consists of different group of table objects, available from an existing repository that you can then deploy to another repository to upgrade or provide patches to existing metadata. As such, an application can be used to distribute and remotely maintain a specific data warehousing solution.

The RED **Deployments** tool is used as a method of creating and loading objects into a metadata repository. The tool generates a series of Windows files (based from the application definition and objects selected) which can then be promoted to remote sites, as a means to manage a data warehousing environment that utilizes multiple metadata repositories.

Launch this tool by selecting **Deployments** from the toolbar menu:



The stand alone application (**RedDeploy.exe**) can also be run from the RED Windows Installation directory.

The succeeding sections describe the steps for using RED**Application Deployment** tool.

Note:

The **REDCLI** command line interface also enables you to create, deploy and manage RED applications. Refer to the REDCLI section of the RED User Guide for details.

Loading an Application

Loading an application inserts various objects into the metadata repository. An application is best defined as a set of objects that are shipped to allow inclusion of those objects into a remote repository.

Note:

An application can only be loaded into a metadata repository running on the same database type as the application creator.

During the load of an application a number of cross environment mappings need to be resolved. These are:

Connections

Connections are normally unique for each metadata repository. They provide the path to the source data and this nearly always varies. Even though an application can ship a set of connections, these connections will nearly always need modification. Alternatively, a connection shipped with an application may map to an existing connection, and this can be done as part of the load process. If you are unsure whether or not a connection is required, it can be added and later modified.

Object Changes

The application may ship objects that already exist in the metadata. In such a case, the process during the load is to create a version of the existing object and then replace it with the new object. At the start of the load, a check is made to see if any objects will be replaced. If the application is a patch or an upgrade of an existing application version, then this will be normal. The load can be stopped at this point to allow for further investigation.

Table Changes

If the load replaces table objects, it may in turn need to alter the physical table in the database to match the new definition. Again, if this is a patch or an application upgrade, this may be desirable. If not, the proposed actions should be reviewed before proceeding.

Procedure Changes

New procedures, scripts, or modifications may be loaded as part of the application load. In the case of procedures, the procedures need to be recompiled. The load can perform this action, but any failures or issues needs to be resolved after the load has been completed.

The following section covers the actual application load process.

Notes:

- It is always a good idea to backup the WhereScape RED metadata before running an application load. Refer to **Backing Up the Metadata before Loading Applications** for details.
- Some database operations, such as converting existing non-partitioned tables to partitioned tables, cannot be done using a deployment application. In such cases, some manual intervention may be required to update the target databases to match the new metadata.

Backing Up the Metadata before Loading Applications

It is always a good idea to backup the WhereScape RED metadata before running an application load.

A metadata backup can be used to restore the metadata to the state before the application was loaded.

There are two options for backing up the metadata:

1. In RED, use the **Backup > Export the metadata** option to backup the metadata, using the data warehouse database's backup utility. Refer to **Backup using DB Routines** in the *WhereScape RED User Guide* for more information.
2. In RED, use the **Tools > Build application tables** option to build an application of the objects being replaced. This is achieved by selecting the new application files you're about to load in the **Previous application** box of the Define and build an application distribution window. Refer to ***Application Creation*** in the *WhereScape RED User Guide* for more information.

Note:

If the table changes in the application are being applied to the data warehouse database, then it is also a good idea to backup the data warehouse tables being changed.

Using RED Application Deployment

The following sections describe the different functions available in the application deployment tool and the detailed steps to create, deploy and manage applications.

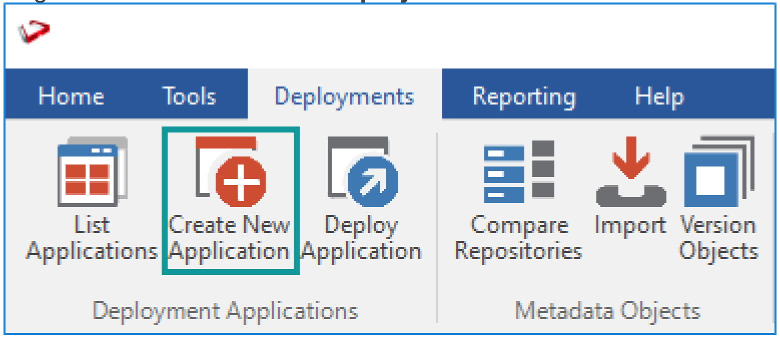
Creating a New Application

The following describe the steps for creating a deployment application in RED.

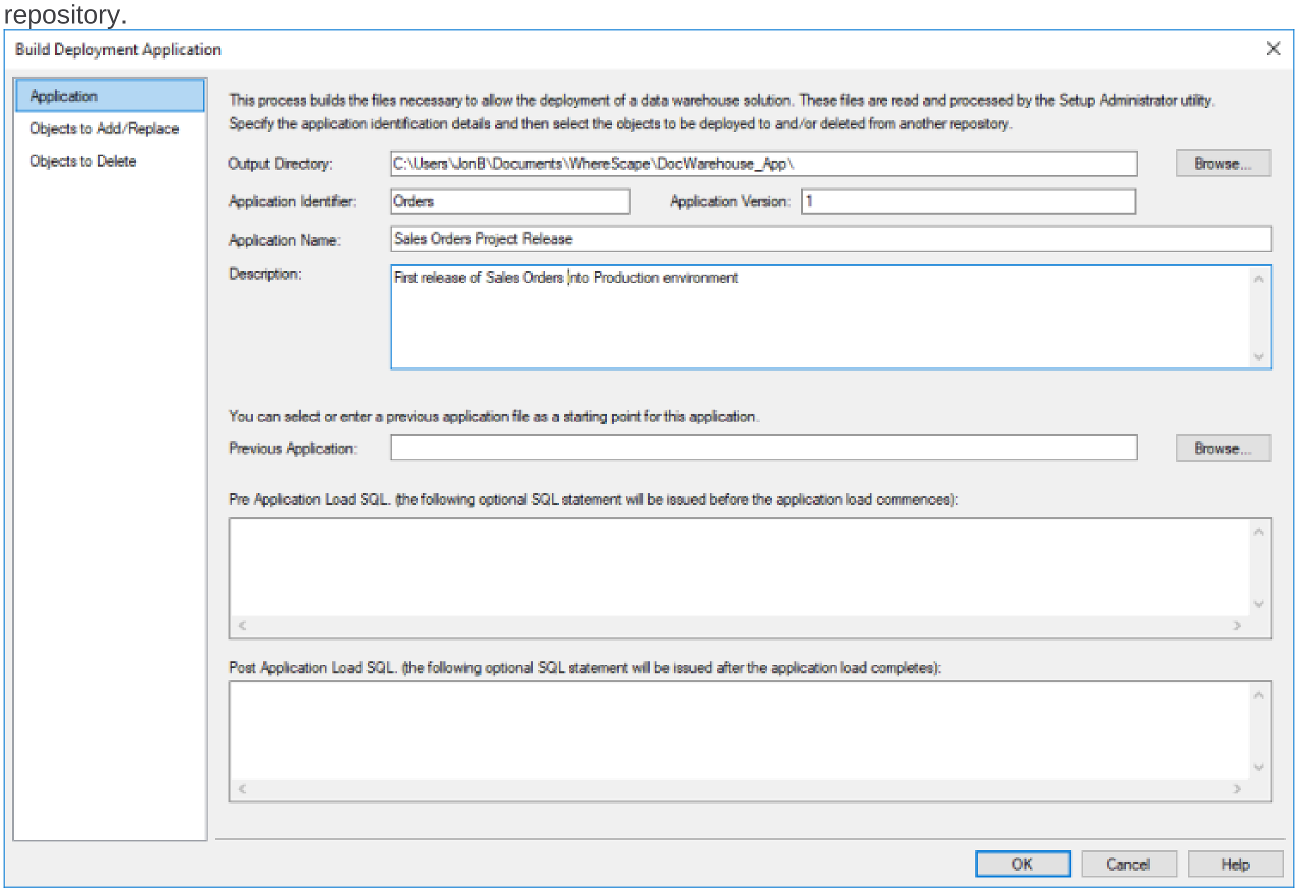
Note:

Maximum number of objects in an application:
5000 objects (including jobs)
2000 source views of views
1000 jobs

1. Log into RED and click the **Deployments** menu to select the **Create New Deployment Application** option.



2. The **Build Deployment Application** window is displayed which enables you to specify the application identification details and optional previous application to be used as starting point, and pre and post application load SQL statements to execute; if required.
   The other tabs enable you to select the objects to be deployed and/or deleted in the selected destination repository.



This process builds the files necessary to allow the deployment of a data warehouse solution. These files are read and processed by the Setup Administrator utility. Specify the application identification details and then select the objects to be deployed to and/or deleted from another repository.

Output Directory: C:\Users\JonB\Documents\WhereScape\DocWarehouse_App\

Application Identifier: Orders

Application Version: 1

Application Name: Sales Orders Project Release

Description: First release of Sales Orders into Production environment

You can select or enter a previous application file as a starting point for this application.

Previous Application:

Pre Application Load SQL. (the following optional SQL statement will be issued before the application load commences):

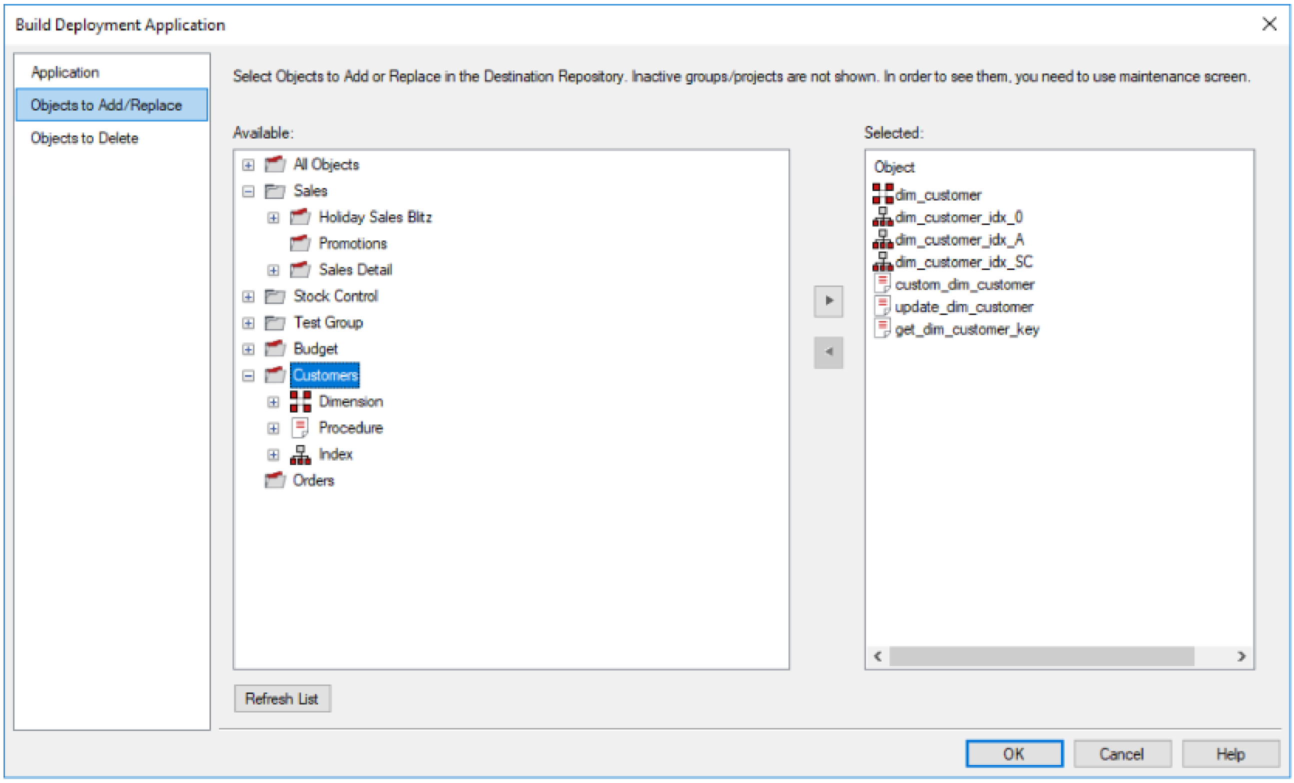
Post Application Load SQL. (the following optional SQL statement will be issued after the application load completes):

Application

Type in the required application details:

Fields	Description
Output Directory	The directory to which the application files will be written. By default, this is the WhereScape program directory. **Note:** To browse for the required folder, click the **Browse...** button. You can click **New Folder** to create a new folder in the selected directory.
Application Identifier	The application identifier is a four character code used to uniquely identify the application. This identifier is used in the naming of the files that are created to contain the application data.
Application Version	The version is a character string that provides a version number for reference purposes. This version number is displayed when applications are being loaded and is used in the naming of the files that are created to contain the application data. As such, it must contain characters that are valid in a Windows file name
Application Name	The name by which the application is known. This name is displayed during the selection of an application to load and is recorded in the metadata of the repository into which an application is loaded. It is not used, apart from documentation purposes.
Description	This description is displayed during the selection of an application to load. It is not used at any other point, apart from documentation purposes.
Previous Application	Click the **Browse** button next to **Previous Application** to choose a previously built application to use as a list of objects to include in the new application. After using a previous application as a starting point for this application, additional objects can be added or removed from the application.
Pre application load SQL	This text box enables you to enter an SQL statement that is executed before the application is loaded. For example, you may wish to drop the date dimension before loading the application because we have changed the primary key constraint. In such a case, you would enter 'drop table dim_date' in this field to have the table dropped before the application is loaded.
Post application load SQL	This text box enables you to enter an SQL statement that is executed after the application is loaded. For example, you could execute a function to populate a table.

3. After defining the application details, click the **Objects to Add/Replace** tab to select the objects that you want to add or replace in the destination repository where the application will be deployed.



Build Deployment Application

Select Objects to Add or Replace in the Destination Repository. Inactive groups/projects are not shown. In order to see them, you need to use maintenance screen.

Available:

- All Objects
- Sales
  - Holiday Sales Blitz
  - Promotions
  - Sales Detail
- Stock Control
- Test Group
- Budget
- Customers
  - Dimension
  - Procedure
  - Index
- Orders

Selected:

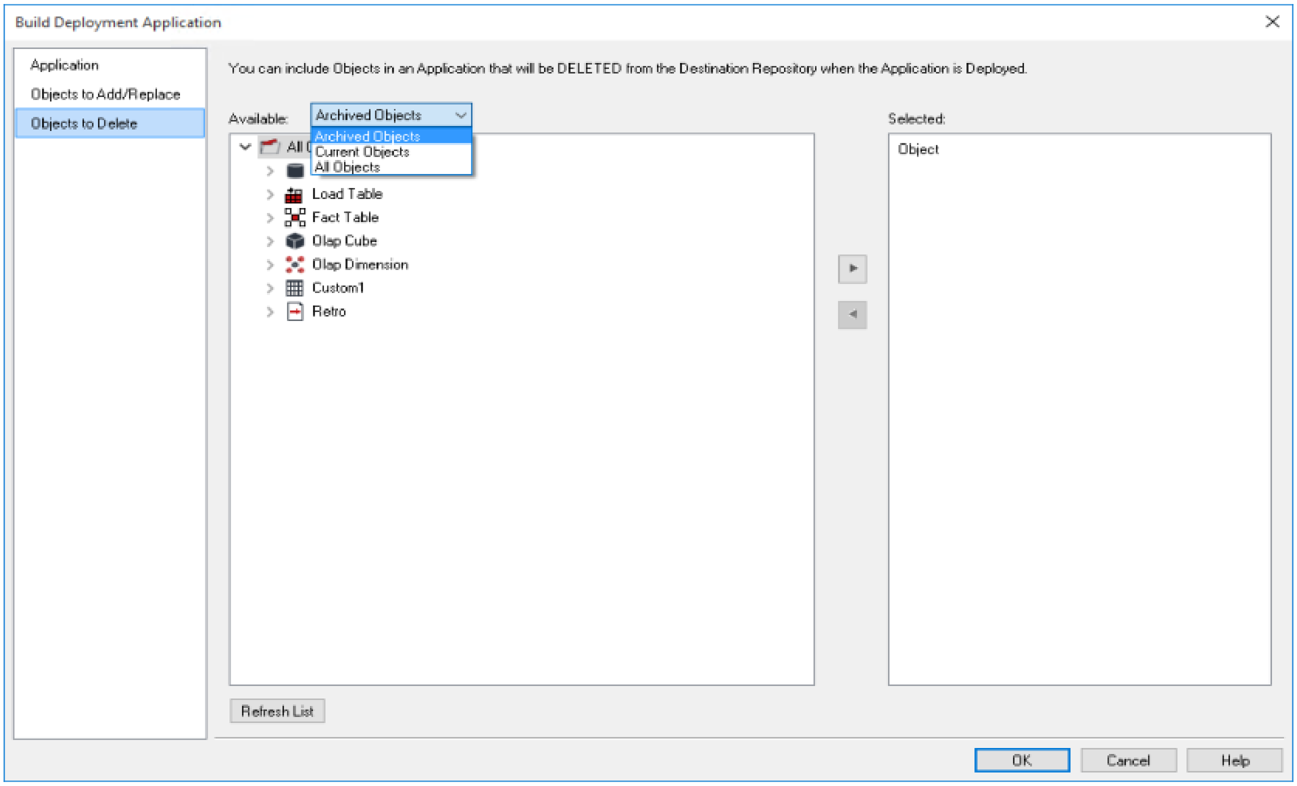
Object
- dim_customer
- dim_customer_idx_0
- dim_customer_idx_A
- dim_customer_idx_SC
- custom_dim_customer
- update_dim_customer
- get_dim_customer_key

Objects to Add/Replace
Double click a Project/Group folder to display the objects that are associated with them. Objects can be moved from the left object tree pane to the right pane by double-clicking an object or by using the **>** button. If procedures are compiled as part of the subsequent application load, the compiles occur in the order in which they are listed in the application. If procedure dependencies exist, ensure that they are ordered correctly in the application object list.

Notes:

- Projects or Groups that have been flagged as inactive are not displayed in the **Available** pane. Refer to **Maintaining Group/Project Active Flag** for details.
- You can select multiple objects using the Windows standard keyboard keys (CTRL or SHIFT) for multi selection.

4. After defining the objects to add/replace, click the **Objects to Delete** tab if you want to specify Objects in the application that will be DELETED from the selected destination repository when the application is deployed.



You can include Objects in an Application that will be DELETED from the Destination Repository when the Application is Deployed.

Available: Archived Objects

Objects to Delete

Click the **Available** drop-down field to select an option for displaying objects in left object tree pane, each option is described in the table below. Double click a Project/Group folder to display the objects that are associated with them. Objects can be moved from the left object tree pane to the right pane by double-clicking an object or by using the **>** button.

Options	Description
Archived Objects	This option includes all objects that have been versioned and deleted. This is the case in a development repository, where the object was removed after versioning. This is the default option.
Current Objects	This option includes all the currently available objects and their associated Projects or Groups.
All Objects	This option includes all the **Current Objects** plus the **Archived Objects** (deleted / versioned). Objects that were versioned and deleted will appear in **All Objects** but will not appear under any Projects or Groups.

5. Once the application is defined and the objects selected, click **OK** to generate the application files.
Check the application output folder, the following files are built when an application is created (where XXXX is the application identifier and NNNNN is the application version).

File	Purpose
App_data_XXXX_NNNNN.wst	This file contains the scripts and data required to rebuild the objects in the new metadata repository.
App_id_XXXX_NNNNN.wst	This control file identifies the application and its version.
App_obj_XXXX_NNNNN.wst	This file contains control information and each object in the application.
App_con_XXXX_NNNNN.wst	A list of all the connections either in the application or used by objects in the application.

File	Purpose
App_map_XXXX_NNNNN.wst	A list of all the project and group mappings for the objects in the application.

Refer to Deploying an Application for details on how to load your new application to a selected RED repository.

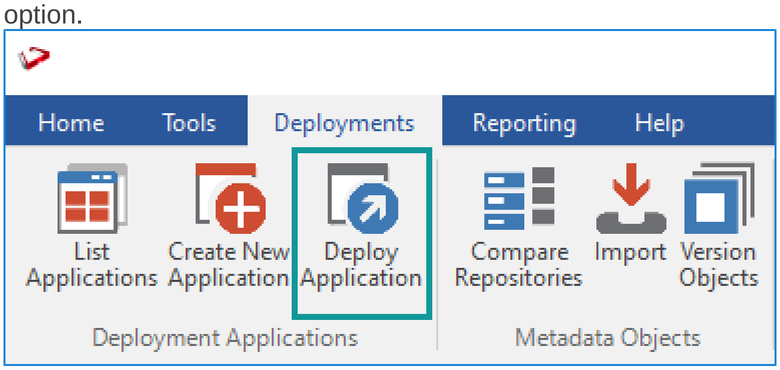
Deploying an Application

The following describe the steps for deploying an application in RED.

Notes:

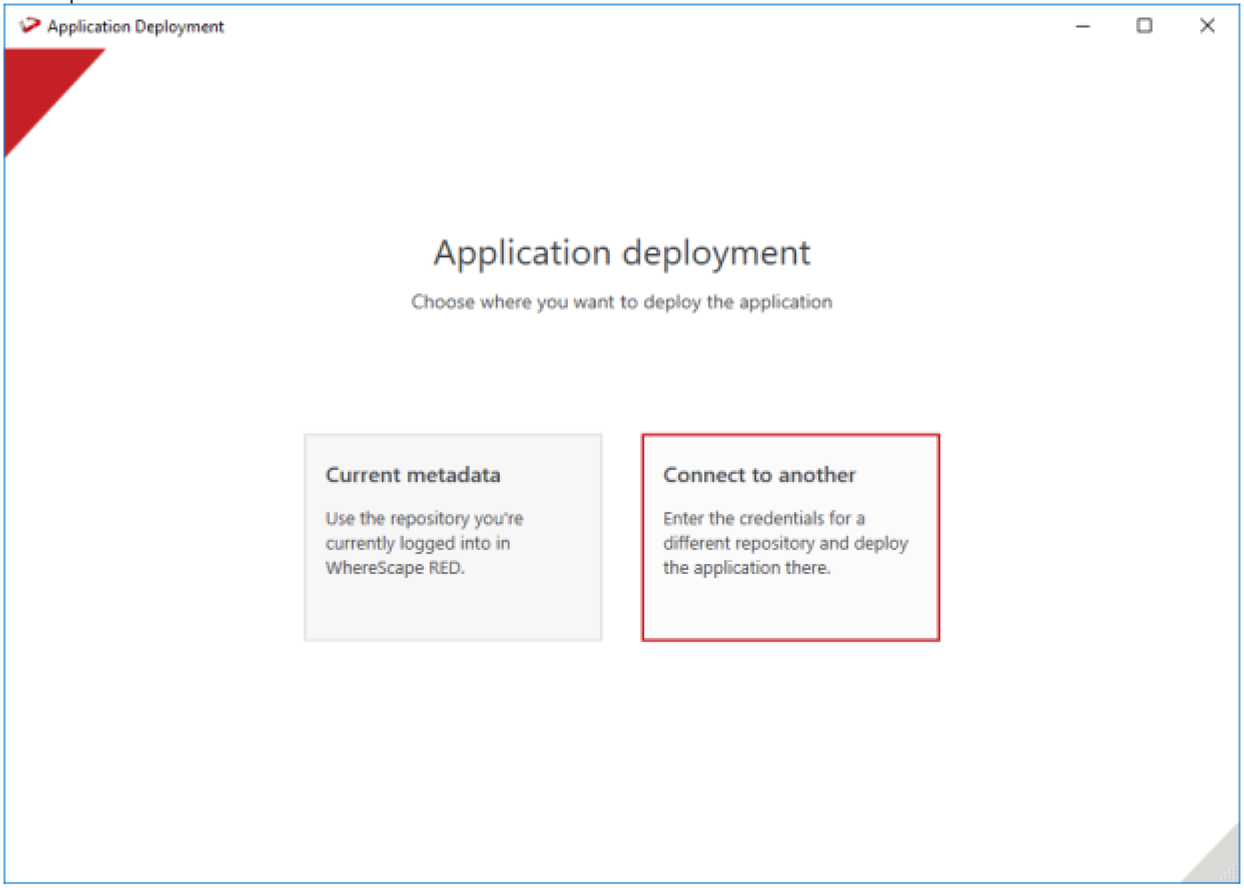
RED utilizes PostgreSQL database for its metadata repository—only applications created from PostgreSQL database can be deployed in RED version 10.0.0.0 and above.

1. Log into RED and click the **Deployments** menu to select the **Deploy Existing Deployment Application** option.



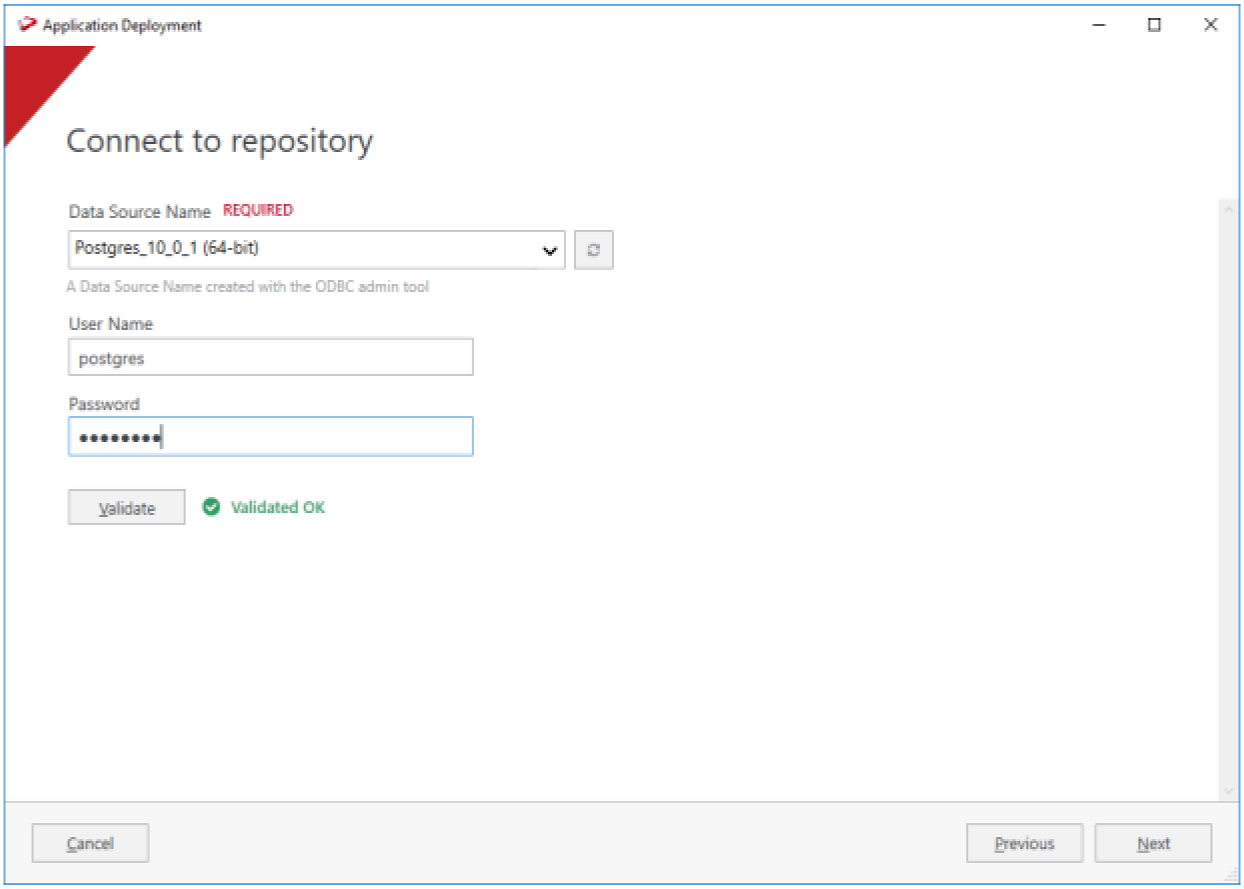
   The stand alone application (**RedDeploy.exe**) can also be run from the RED Windows Installation directory.
2. Select the metadata repository where you want to deploy the application. Click **Current metadata** to deploy the application in the RED metadata repository you are currently logged or click **Connect to another** to deploy to a different repository.
   For this example, click **Connect to another** repository option to proceed or click **Current metadata** to skip

to step 5 below.



Application Deployment
Application deployment
Choose where you want to deploy the application
Current metadata
Use the repository you're currently logged into in WhereScape RED.
Connect to another
Enter the credentials for a different repository and deploy the application there.

3. The **Connect to repository** window is displayed that enables you to specify a data source to which RED connects and the log in details for the selected target repositorydatabase.



Type in the required metadata connection details:

Field	Description
Data Source Name	The ODBC data source name (DSN) for the metadata repository connection.
Username	The log in user name to connect to the metadata repository connection.
Password	The password for the specified user name.

4. Click **Validate** to continue the data source and log in credentials validation.

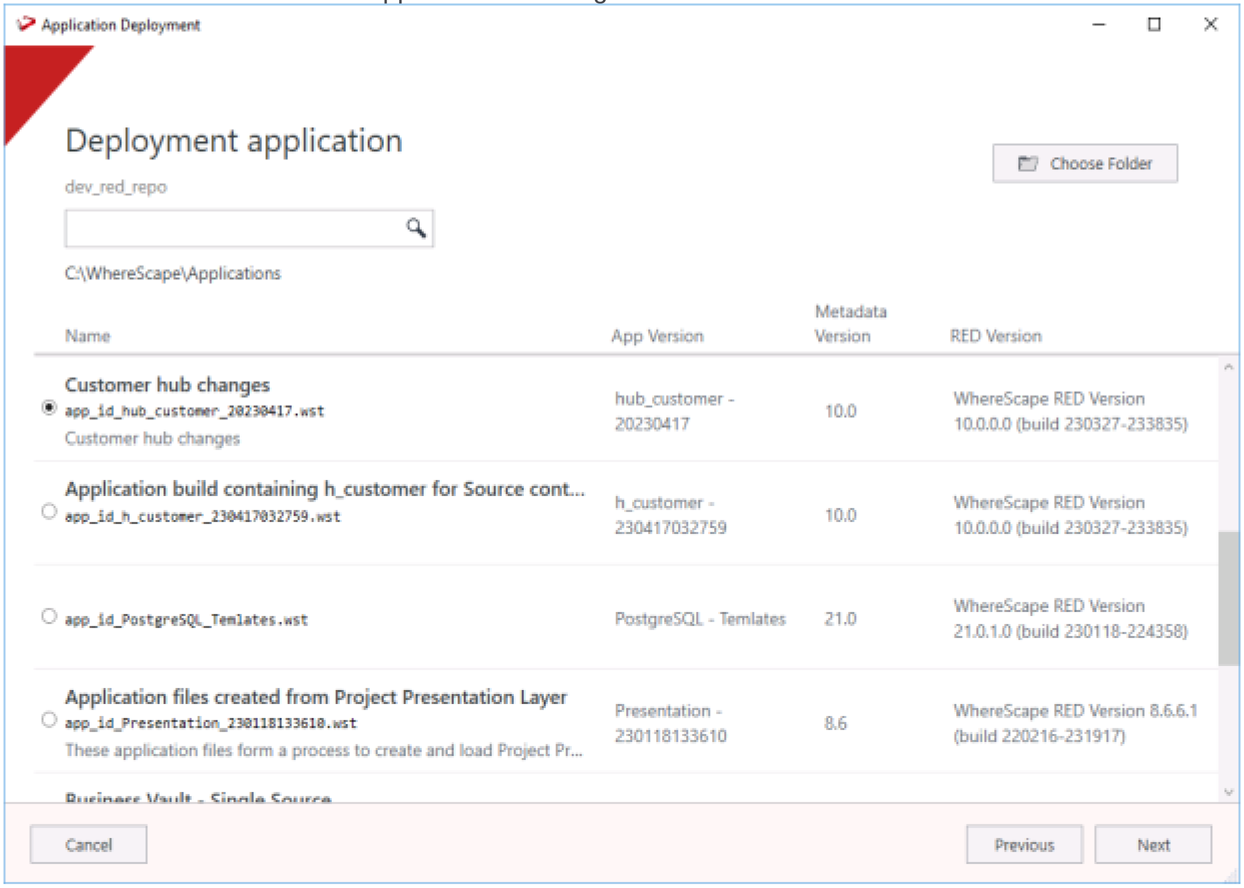
Notes:

The validation is performed against the selected data source to ensure:

- The database is not already being used by another RED repository.
- The database is not being used for another purpose.

5. After successful validation, click **Next** to continue. The **Deployment application** window is displayed with a list of application retrieved from the specified directory. You can change the directory using the **Choose**

Folder button or search for an application file using the search field.



Deployment application

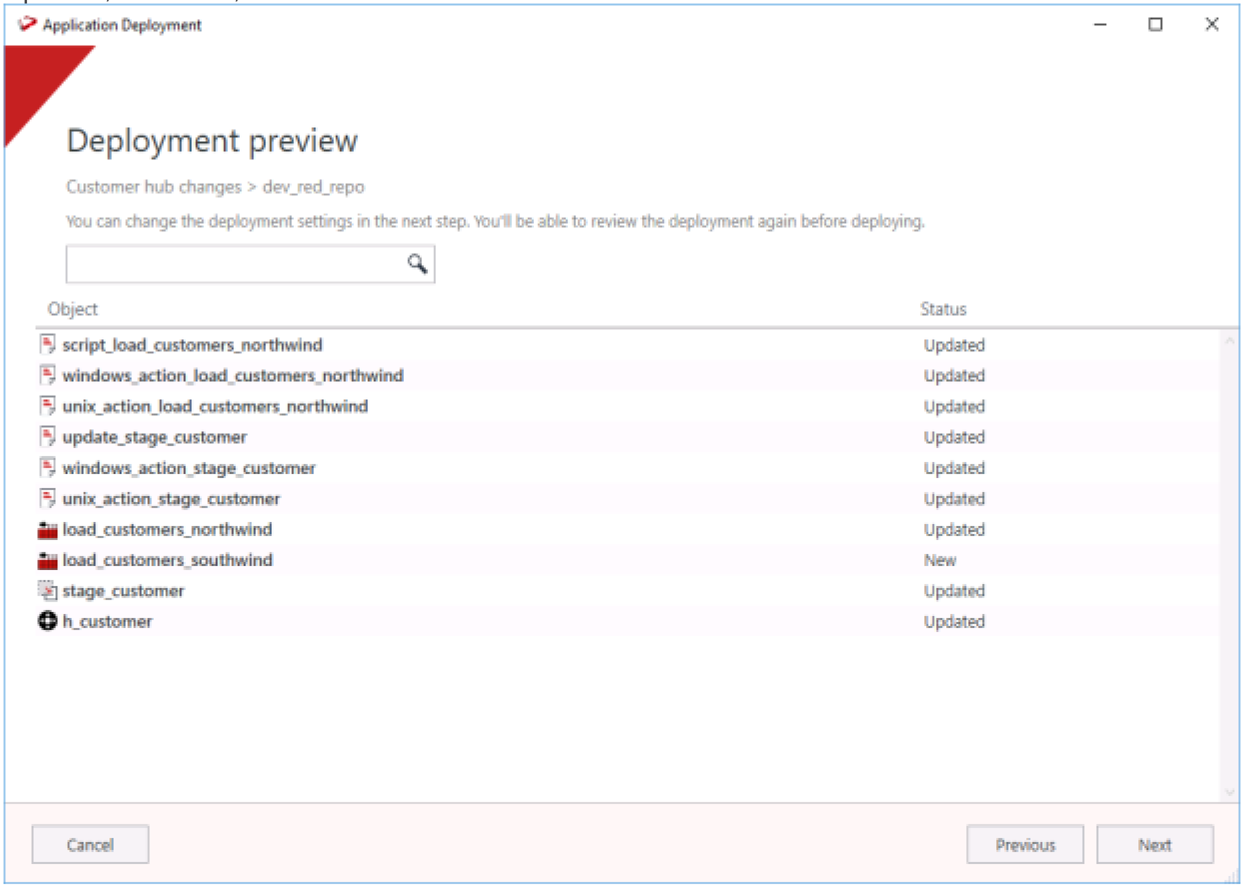
dev_red_repo

C:\WhereScape\Applications

	Name	App Version	Metadata Version	RED Version
●	Customer hub changes app_id_hub_customer_20230417.wst Customer hub changes	hub_customer - 20230417	10.0	WhereScape RED Version 10.0.0.0 (build 230327-233835)
○	Application build containing h_customer for Source cont... app_id_h_customer_230417032759.wst	h_customer - 230417032759	10.0	WhereScape RED Version 10.0.0.0 (build 230327-233835)
○	app_id_PostgreSQL_Temlates.wst	PostgreSQL - Temlates	21.0	WhereScape RED Version 21.0.1.0 (build 230118-224358)
○	Application files created from Project Presentation Layer app_id_Presentation_230118133610.wst These application files form a process to create and load Project Pr...	Presentation - 230118133610	8.6	WhereScape RED Version 8.6.6.1 (build 220216-231917)
	Business Vault - Single Source			

6. Click **Next** to continue. A warning message is displayed, if the metadata version of the application you are deploying does not match the metadata version of the repository to which the application will be deployed. WhereScape recommends upgrading the source repository and recreating the application before deploying to a production system so that the versions match.
7. Click **Next** to continue. The **Deployment preview** window is displayed, showing a list of objects that will be deployed in the selected metadata repository. The status of each object is also displayed, e.g. New,

Updated, Removed, etc.



Deployment preview

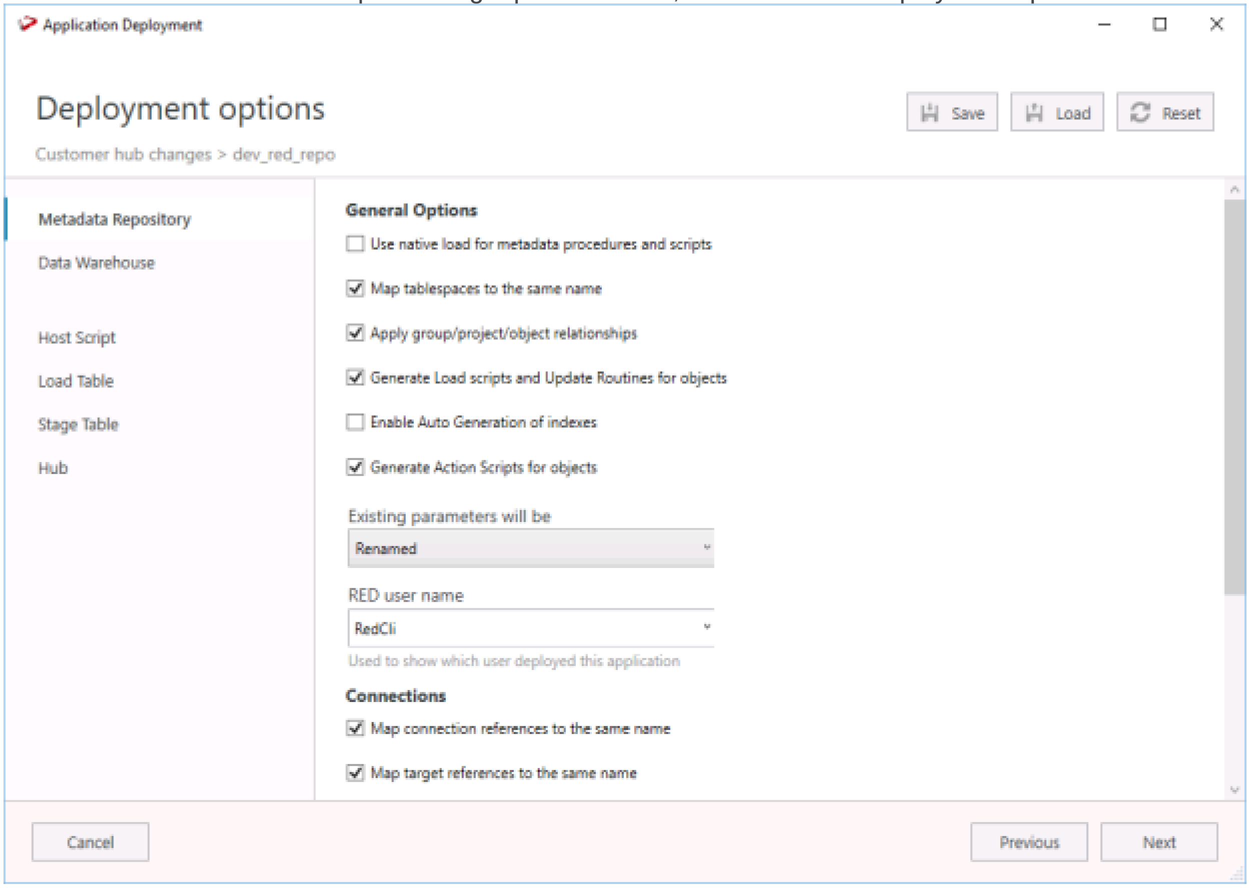
Customer hub changes > dev_red_repo

You can change the deployment settings in the next step. You'll be able to review the deployment again before deploying.

Object	Status
script_load_customers_northwind	Updated
windows_action_load_customers_northwind	Updated
unix_action_load_customers_northwind	Updated
update_stage_customer	Updated
windows_action_stage_customer	Updated
unix_action_stage_customer	Updated
load_customers_northwind	Updated
load_customers_southwind	New
stage_customer	Updated
h_customer	Updated

8. Click **Next** to continue. The **Deployment options** window is displayed, showing default deployment settings for the metadata repository, data warehouse and for the other objects included in the deployment application. Click each of the tabs in the left pane to view or change the settings available under each item. For a detailed list of the deployment options refer to the **Deployment Options** section.

You can use the buttons on top of the right pane to **Save**, **Load** or **Reset** deployment options.



General Options

☐ Use native load for metadata procedures and scripts

☑ Map tablespaces to the same name

☑ Apply group/project/object relationships

☑ Generate Load scripts and Update Routines for objects

☐ Enable Auto Generation of indexes

☑ Generate Action Scripts for objects

Existing parameters will be: Renamed

RED user name: RedCli

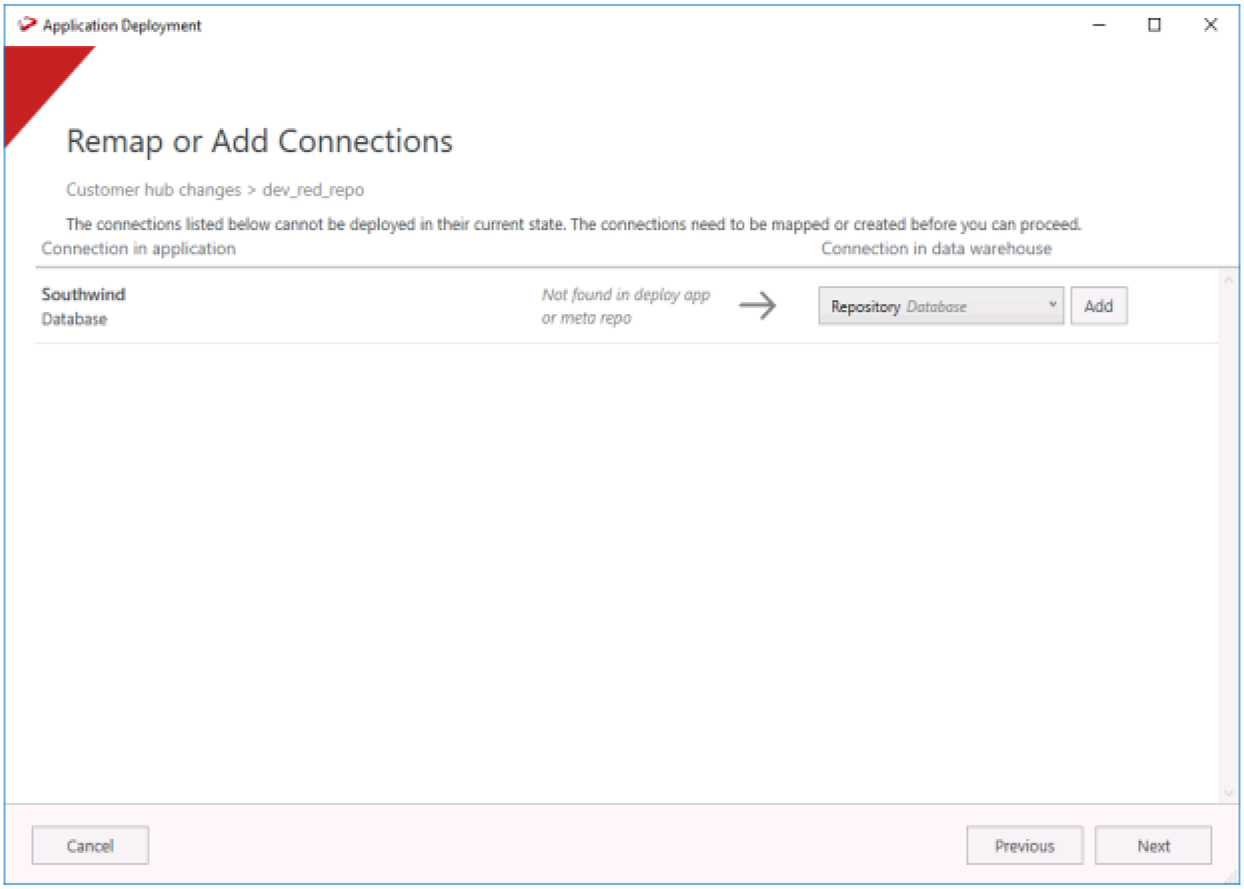
Used to show which user deployed this application

Connections

☑ Map connection references to the same name

☑ Map target references to the same name

9. Click **Next** to continue, the connections used by the deployment application are checked to ensure that they are present in the metadata repository. The results are displayed in the **Remap or Add Connections** window.



Application Deployment

Remap or Add Connections

Customer hub changes > dev_red_repo

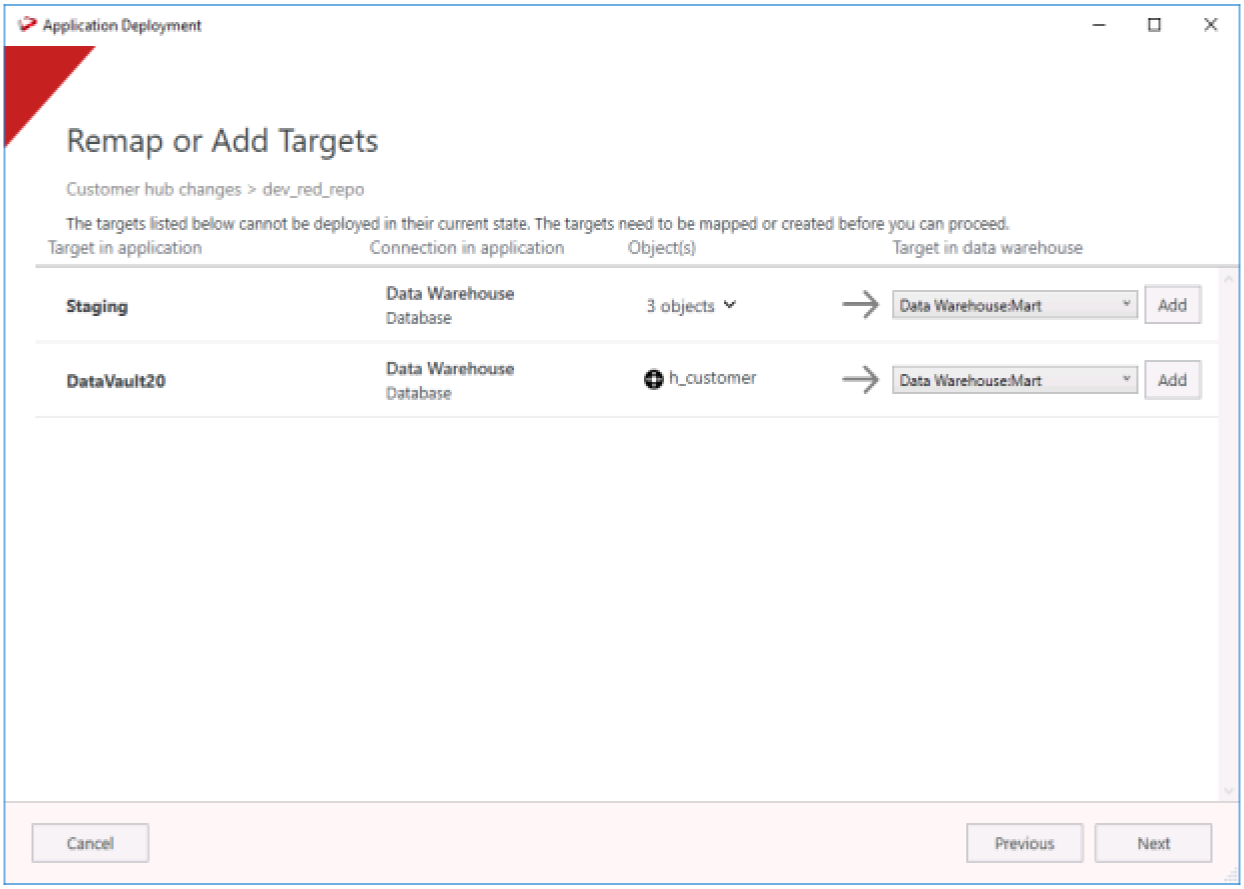
The connections listed below cannot be deployed in their current state. The connections need to be mapped or created before you can proceed.

Connection in application			Connection in data warehouse	
Southwind Database	*Not found in deploy app or meta repo*	→	Repository *Database*	Add

Note:

It is advisable to map to an existing connection in RED to achieve the automation, otherwise, after creating a connection make sure to generate the load and update scripts for the object after deployment.
If any issues are found in the connections used by the deployment application, you need to fix them before you can proceed. Follow the prompts that are displayed to map or add connections.

10. Click **Next** to continue, the targets used by the deployment application are checked to ensure that they are present in metadata repository. The results are displayed in the **Remap or Add Targets** window.



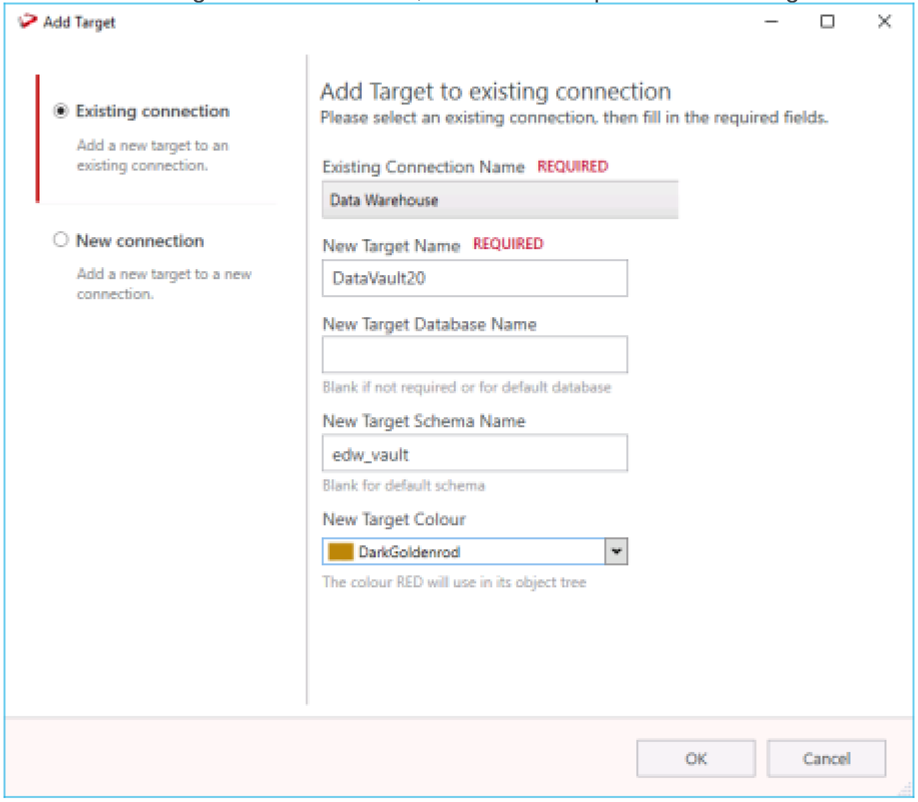
Remap or Add Targets

Customer hub changes > dev_red_repo

The targets listed below cannot be deployed in their current state. The targets need to be mapped or created before you can proceed.

Target in application	Connection in application	Object(s)	Target in data warehouse
Staging	**Data Warehouse** Database	3 objects	Data Warehouse:Mart
DataVault20	**Data Warehouse** Database	h_customer	Data Warehouse:Mart

You can add targets on this screen, click **Add** to open the Add Target wizard.


Add Target
● Existing connection
Add a new target to an existing connection.
○ New connection
Add a new target to a new connection.

Add Target to existing connection

Please select an existing connection, then fill in the required fields.

Existing Connection Name REQUIRED: Data Warehouse

New Target Name REQUIRED: DataVault20

New Target Database Name:

Blank if not required or for default database

New Target Schema Name: edw_vault

Blank for default schema

New Target Colour: DarkGoldenrod

The colour RED will use in its object tree

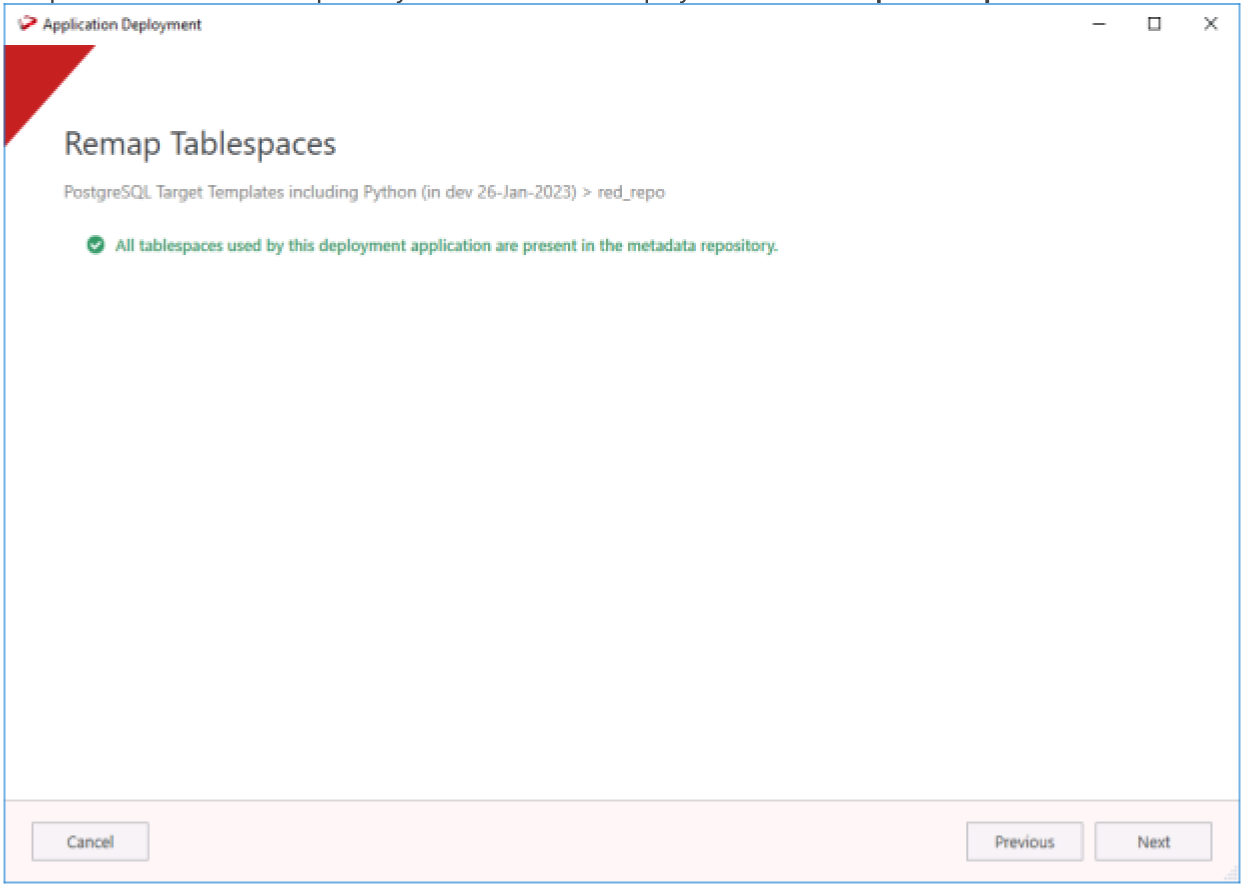
1. Select an existing connection from the drop-down list.
2. Add the target schema name and click **OK** to return to the **Remap or Add Targets**window.

3. After completing your mapping , click **Next**.

Note:

If any issues are found in the targets used by the deployment application, you need to fix them before you can proceed. Follow the prompts that are displayed to map or add targets.

11. Click **Next** to continue, the tablespaces used by the deployment application are checked to ensure that they are present in metadata repository. The results are displayed in the **Remap Tablespaces** window.



Remap Tablespaces

PostgreSQL Target Templates including Python (in dev 26-Jan-2023) > red_repo

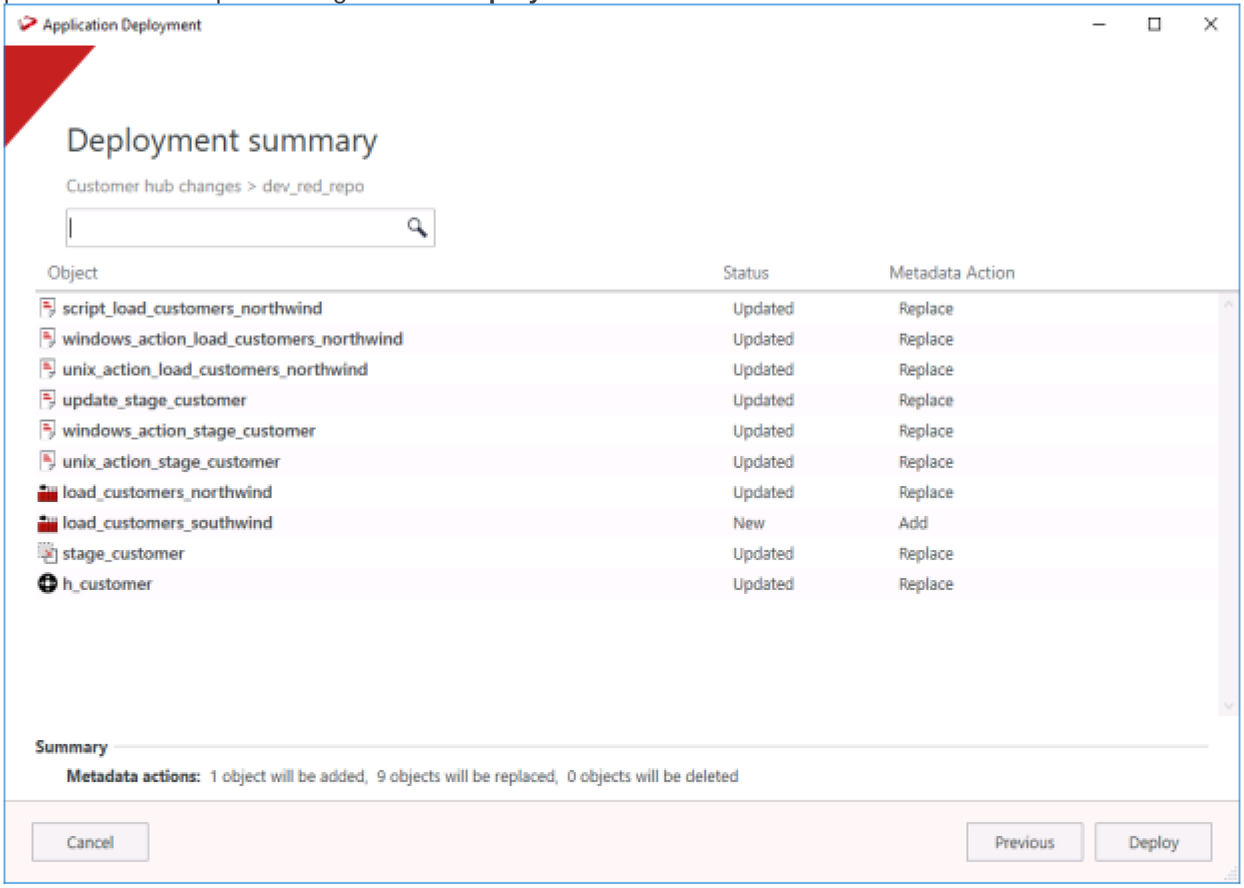
All tablespaces used by this deployment application are present in the metadata repository.

Note:

If any issues are found in the Tablespaces used by the deployment application, you need to fix them before you can proceed. Follow the prompts that are displayed to remap Tablespaces.

12. Click **Next** to continue. The **Deployment summary** window is displayed, showing a list of objects that will be deployed in the selected metadata repository. The status, metadata and database actions for each object is also displayed. It is recommended that you go through the list and review the actions that will be

performed before proceeding with the **Deploy**.



Deployment summary

Customer hub changes > dev_red_repo

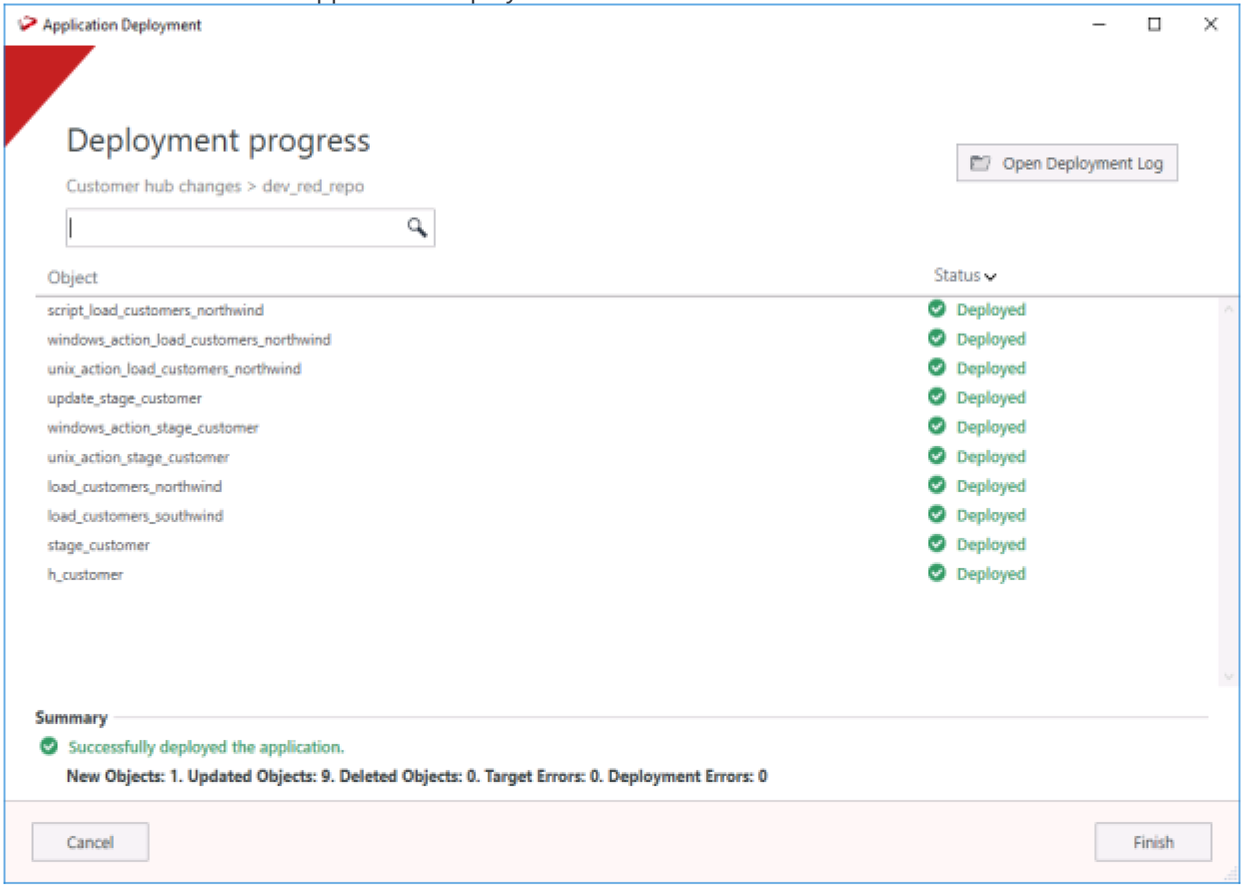
Object	Status	Metadata Action
script_load_customers_northwind	Updated	Replace
windows_action_load_customers_northwind	Updated	Replace
unix_action_load_customers_northwind	Updated	Replace
update_stage_customer	Updated	Replace
windows_action_stage_customer	Updated	Replace
unix_action_stage_customer	Updated	Replace
load_customers_northwind	Updated	Replace
load_customers_southwind	New	Add
stage_customer	Updated	Replace
h_customer	Updated	Replace

Summary

Metadata actions: 1 object will be added, 9 objects will be replaced, 0 objects will be deleted

13. After going through the deployment summary to review the actions that will be executed for the objects included in the application, you can click **Deploy** to proceed.
14. Once the deployment is complete, a window is displayed to confirm successful deployment of the objects included in the application.

Click **Finish** to close the Application Deployment tool.



Deployment progress

Customer hub changes > dev_red_repo

Object	Status
script_load_customers_northwind	✓ Deployed
windows_action_load_customers_northwind	✓ Deployed
unix_action_load_customers_northwind	✓ Deployed
update_stage_customer	✓ Deployed
windows_action_stage_customer	✓ Deployed
unix_action_stage_customer	✓ Deployed
load_customers_northwind	✓ Deployed
load_customers_southwind	✓ Deployed
stage_customer	✓ Deployed
h_customer	✓ Deployed

Summary

✓ Successfully deployed the application.

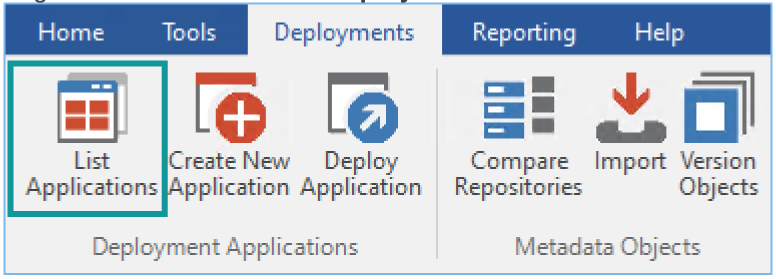
New Objects: 1. Updated Objects: 9. Deleted Objects: 0. Target Errors: 0. Deployment Errors: 0

In case of any errors when deploying the application, click **Open Deployment Log** located on the top right corner of the window to review more details and troubleshoot your deployment.

Listing Deployed Applications

The following describe the steps for listing deployed applications in a RED repository.

1. Log into RED and click the **Deployments** menu to select the **List Applications** option.



2. The **Deployed Applications** pane is displayed below the RED Builder window which lists all the applications deployed in the selected RED repository.

Deployment Options

Application Files

Option	Description
Application Identifier	The application identifier is a four character code used to uniquely identify the application. This identifier is used in the naming of the files that are created to contain the application data.
Application Version	The version is a character string that provides a version number for reference purposes. This version number is displayed when applications are being loaded and is used in the naming of the files that are created to contain the application data. As such, it must contain characters that are valid in a Windows file name.
Application Database Type	The database type from which the application was built. An application can only be loaded into a metadata repository running on the same database type from which the application was created.

General Options

Option	Description
Use Native Load for Metadata Procedures and Scripts	This option causes the metadata for procedures and scripts to be loaded into the target metadata repository, using the target repository's load utility instead of ODBC. This can have significant performance advantages when loading large applications, when loading applications into a slow target database or over a slow network.
Map Tablespace/Filegroups to the same name	Some tables may be in **filegroups**. If this option is set, existing tablespaces and filegroups in the target data warehouse database will automatically be matched to their same ones on the source repository. If the tablespace/filegroup does not exist in the target data warehouse database, a Tablespace selection window pop-ups so a tablespace/filegroup can be selected. This window appears once for each tablespace/filegroup that cannot be matched.
Apply Group/Project/Object Relationships	If this option is set to True, the project group relationships are updated in the target data warehouse database.
Generate load scripts and update routines for objects	If this option is set to True, the deployment will include the generation of load scripts and update routines for the table objects, but only if: • the code generation is template based. • a default load script and update routine template is set for the object type in the target connection • the default build options satisfy the requirements of the default template. Refer to the Connection Properties and Rebuilding Update Procedures sections of the RED User Guide for details. This option can be used by Data Vault customers who utilize WhereScape 3D to import table objects into RED via deployment applications. This setting is enabled by default for 3D applications so that when the objects are loaded into RED during deployment, their initial update scripts or routines are also generated. This option defaults to 'OFF' during the deployment of applications generated by RED.
Enable auto generation of indexes	If this option is set to True, the deployment will also include the generation of indexes for the table objects after the update routines are generated.
Existing Parameters will be	Enables you to specify the action for existing parameters. The options are:

Option	Description
	- Updated/Overwritten - Retained - Renamed
User Name	Enables you to enter a username which is included in the versioning comments.
Permanent Stage Table Options Hidden	Enables you hide the Permanent Stage Table object options in the window. **Save Settings** must be used to save the option selection. This window does not refresh upon changing this option.

Connections

Option	Description
Map Connection References to the same name	During deployment, objects need to be associated with a Connection. Setting this option to False causes the user to be prompted to choose a Connection for each object. Setting this option to True causes the deployment process to compare the name of all existing connections within the Data Warehouse with the Connection name defined in the deployment application for the current object. If the names match, then the WhereScape RED Setup Administrator deploys the object to this Connection. If no Connection name matches, then the user is prompted to choose a Connection.
Map Target References to the same name	During deployment, objects need to be associated with a target within a Connection. Setting this option to False causes the user to be prompted to choose a Target for each object. Setting this option to True causes the deployment process to compare the name of all existing Targets within the Connection that has been either selected by the user or matched automatically. If the names match, then the deployment process deploys the object to this Target. If no Target name matches, then the user is prompted to choose a Target. Choosing a Target involves choosing a Connection—the deployment process automatically selects a Connection. If required, the selected Connection can be overridden using the GUI.
Add Connections as required	If this option is set to True, all Connection objects included in the application that do not exist in the Target metadata repository which are used by objects in the application will be added to the Target metadata repository.

Jobs

Option	Description
Assign Jobs to User	All jobs in the WhereScape RED scheduler belong to a user. If this option is marked as true, the user name specified in the **User name for any jobs added by the application** drop-down is set as the user name of all jobs loaded by the application. If the specified user name does not exist, it is then created.
Existing Jobs will be	Enables you to specify the action for existing jobs. The options are: - Updated/Overwritten - Retained - Renamed

Data Warehouse Options

Option	Description

Option	Description
Apply Metadata Changes to Data Warehouse Objects	If this option is marked as True, the changes contained in the application are applied to the metadata according to the options selected for each object type.
Log Metadata DML and Data Warehouse DDL to a File	If this option is marked as True, the changes contained in the application are written to a file. • **DDL File** - Option to enter or select the file name/path for the DDL file. • **DDL File Statement Terminator** - Option to specify the statement terminator. • **Terminate lines with carriage return as well as new line** - Option to terminate DDL file lines with a carriage return, as well as a new line. • **Select encoding for DDL and Procedure files** - Option to specify the encoding to be used for the DDL and Procedure files—options are UNICODE, ASCII or UTF-8.
Compile New and Changed Procedures	If this option is marked as true, then all procedure objects loaded by the application load are compiled in the target data warehouse database. **Note:** Depending on the type of database, invalid procedures may not compile.

Data Warehouse Table/View Options

For each table and view object loaded by the application load, it is possible to specify whether the metadata changes should be applied to the data warehouse table or view; and to the indexes on these objects. The change action can be specified by object type. New objects can have a different action to existing objects of the same type. Possible actions are:

- **Created** - available for new tables and indexes of any object type
- **Recreated** - available for existing tables and indexes of any object type
- **Altered** - available for existing tables and indexes of most object types (not available for load tables or views)
- **Nothing** - available for all new and existing object types and indexes.

The following additional settings can be configured when importing Load table objects from 3D to RED:

- New Table Default Load Type, if not set in Source Connection
- New Table Default Load Script Connection, if not set in Source Connection
- New Table Default Load Script Template, if not set in Source Connection

Note:

The above default settings are only applied for new Load table objects, if these attributes are not defined in the Source Connection. They only apply for new Load table objects and Connections from 3D.

This window also enables you to set whether or not new or existing objects are to be versioned. Set to **True** to use versioning, else set to **False**.

Other options:

- **Alter allows columns to be dropped** - This option enables you to set whether or not columns are allowed to be dropped by object type, for tables being altered. Set to **True** to allow columns that are not in the metadata, to be dropped; else set to **False**.
- **Allow alter table processing with non-compliant DDL** - This option is used for custom database targets and sets whether or not to allow DDL statements that do not use a substitutable format for the table name to be used during validation, and subsequent alter table processing. Default setting is **True**.
  The DDL statements are set in the template that is defined in the custom database Connection properties

(**Connection > Target Settings > Default Table Create DDL Template**) in RED. Refer to Connection Target Settings for Custom Database in the RED User Guide for details.

WARNING:

Setting this option to TRUE can lead to the DDL being run against the actual data warehouse table.

Creating and Loading Applications from the Command Line

It is possible to create and load applications from the command line by running a batch file. The WhereScape RED Application directory contains an example batch file WSL_Application_Create_Restore_Point_and_Load.bat for creating and loading.

If you right-click this file and select **Edit** you will see the steps outlined, as well as the details on the options available.

The first step **creates** a restore point application (R) based on objects about to be loaded. This process calls the command line functionality of RED and creates the application file.

The second step **loads** an application (A) into a test WhereScape RED repository. It uses an XML Options File to specify various options, calls the command line automation functionality of Setup Administrator and loads the application (A). Ensure that you have already created and properly defined the XML Options File. Refer to **XML Options File for Application Loads** for details.

The WhereScape RED Application directory contains example XML files:

- WSL_Application_Load_SQL.xml for SQL Server

The tags in the XML Options File need to be edited because the login/connection details, etc. need to be set. Refer to **XML Options File for Application Loads** for details.

The Batch Application Create options and the Batch Application Load options (described below) are listed at the end of the batch file. The values for these variables also needs to be customized before running this file.

The process typically involves the following steps:

1. Create an application (A) in RED containing your data warehouse changes.
2. RUN WSL_Application_Create_Restore_Point_and_Load.bat from the command line; which creates a restore point application (R) and applies application (A) to a test WhereScape RED repository.
3. If the changes are incorrect, they can be undone by loading the restore point application (R).

Batch Application CREATE Options

The following options are available on RED to create applications.

Option	Description
/BA	Selects batch application create.
/U	ODBC user name.
/P	ODBC password.
/C	ODBC DSN name.
/A	DSN Architecture
/M	meta database for Teradata logon.
/N	RED User name.
/D	Directory to save application files.
/I	New application files identifier.
/V	New application files version.
/AP	Project name - all objects in a Project; and associated jobs.

Option	Description
/AG	Group name - all objects in a Group; and associated jobs.
/ALL	All objects - all objects, jobs and parameters.
/RC	Remove connections - drop all connections from the application.
/RJ	Remove jobs - drop all jobs from the application.
/RP	Remove parameters - drop all parameters from the application.
/AF	Absolute application id file name which restore point is being created for.

If using a trusted connection, the user name and password are not necessary.

Batch Application LOAD Options

See **RedCli section** for details.

Testing Applications

A testing application set consists of a Procedure and an XML script and provides the ability to define a series of tests against data warehouse objects; either comparing them to an expected value or to the results of a query.

The XML script contains the test definitions. Each test is a new XML node in the XML script and contains a name, a test query, an expected value, or a comparison query.

The procedure simply runs the tests and determines whether the tests are passed or not. This is most likely to be run as a scheduled job within WhereScape RED.

Objects and Windows

WhereScape RED makes use of an object concept when dealing with the different components that make up a data warehouse solution.

The main object types are: Connections, Load Tables, Dimensions, Stage Tables, Fact Tables, OLAP Cubes, Aggregates, Procedures, Host Scripts, Indexes, Retros and Exports.

This chapter explains and provides an overview of each of these object types and how they can be managed and organized. The full functionality of each object is covered in the succeeding chapters.

The various Windows, Panes and Views that form the WhereScape RED tool are also explained.

Object Types

WhereScape RED has a concept of objects which are combined to create real world data warehouses and data marts, fast.

Each WhereScape RED object has properties that enable the data warehouse developer to change how the object is used.

Note:

Some object types may not be available for certain types of WhereScape RED licenses.

WhereScape RED objects include:

Object Type	Icon	Description
Connection		**Connections** define the path to external objects, such as source data. Examples of connection object types are databases, analysis services cubes, operating systems or ODBC sources. Connections isolate environments simplifying, for example, the promotion of code between development and production.
Load Table		**Load tables** are the first entry point of data into the data repository, and typically hold the latest set of change data. These objects contain their definition. Load tables can be defined as database link, ODBC, external, file, script or XML. Based on their definition, they will for example, run a predefined script or create a load script at run time. Pre load actions (e.g. truncate) or post load procedures can be defined as part of a load object. In addition, transformations (either during or after the load) can be defined against columns in a load table.
Dimension Table		**Dimension tables** are the constraining elements in the star schema design and are defined by this object type. WhereScape RED automatically generates procedural code for the three standard types of slowly changing dimensions, as well as date ranged dimensions (where the current version is defined by an external system). WhereScape RED also ships with a standard time dimension which can of course be extended. Dimensions can also be defined as mapping or work tables which do not appear in the generated user documentation.
Dimension View		A **Dimension view** is a database view of a dimension table. It may be a full or partial view. A common usage is to create dimension views where multiple date dimensions exist for one fact table. Other types of views supported by WhereScape RED include fact views, other table views, work views and user defined views.
Stage Tables		**Stage tables** are used in the transformation of raw data into model or star schema format. They typically hold only the latest set of change data. In addition to custom procedures, WhereScape RED can generate

Object Type	Icon	Description
		different types of procedural code based on the complexity and size of the data set and performance requirements. Examples of procedural types that can be generated are cursor, sorted cursor, set, set + cursor or set merge procedural code. A Stage table can also be defined as a work table, which has the same properties as a stage table but does not appear in the generated user documentation.
EDW 3NF Table		An **EDW 3NF** table is a data warehouse object used to build third normal form enterprise data warehouses. In WhereScape RED, EDW 3NF objects have many of the code generating attributes of stage, dimension and fact tables. Third normal form enterprise data warehouses can be thought of as a source system for star schema data marts. Alternatively, they may be reported off directly by users and reporting tools.
Data Store Table		A **Data Store Table** is a data warehouse object used to store any type of data for later processing. In WhereScape RED, Data Store objects have many of the code generating attributes of stage, dimension and fact tables. Data objects can be thought of as a source system for the data warehouse. Alternatively, they may be reported off directly by users and reporting tools. Data Store Objects can be considered either reference or transactional in nature.
Fact Table		**Fact tables** are the central table in a star schema design. This object type enables the definition of fact tables. They support transactional, rollup, snapshot or partitioned (detail, rollup or exchange) fact tables. Changing a fact table's properties to partitioned, starts a partitioning wizard that prompts for the required information.
KPI Fact Table		This object type supports a special type of fact table. A mandatory KPI (**Key Performance Indicator**) dimension provides a set of KPIs which are stored and maintained by this object type.
Aggregate	∑	The **Aggregate** object type provides a means to speed up access by summarizing data to a higher grain. For dimensional models a rollup of the fact data will allow removal of dimensions that are no longer valid.
View		**View** objects are usually created as end user objects from any table in the data warehouse. The data or columns may be restricted or extra descriptions may be added for use by the end user or reporting tools.
OLAP Cube		The **OLAP Cube** object type uses Analysis Services cubes to deliver OLAP functionality in WhereScape RED . A cube is a set of related measures and dimensions that is used to analyze data from a variety of different front-end tools. OLAP Cubes are built from fact objects and aggregate objects in WhereScape RED.
OLAP Dimension		An **OLAP Dimension** is built by WhereScape RED for every dimension table associated with the fact (or aggregate) table the OLAP Cube is derived from. OLAP Dimensions are shared across one or more OLAP Cubes. In analysis services, a dimension is a group of attributes that represent an area of interest related to the measures in the cube and which are used to analyze the measures in the cube.
Procedure		The **Procedure** object type is used to define and hold database stored procedures. As such it may contain functions, procedures and packages that are generated, modified or custom developed.
Host Script		**Host script** objects are either Windows or UNIX scripts. These scripts are maintained within the WhereScape RED environment and can be scheduled to run in their host environments.

Object Type	Icon	Description
Index		This object type defines database **indexes** used to improve the access times on any of the table object types. (i.e. Load, Stage, Dimension, Fact, Kpi Fact and Aggregate).
Export		**Export** objects are used to manage exports from the data repository. In essence, exports are the reverse of Load tables, taking data from a table to a flat file.
Retro		**Retros** are used to load predefined data models from modeling tools and to retrofit existing tables into the WhereScape RED metadata.
Retro Copy		**Retros** can be used to copy data from an existing data warehouse into WhereScape RED metadata. Retros can be set as Retro Copy objects to enable data transfer from the existing data warehouse to the new data warehouse.
Template	</>	**Template** objects are used to generate DDL, update/custom procedures and host scripts. Once a template has been created it can be associated with a table and an operation on that table. The template is then used to generate the script used for the associated operation. Each template is assigned a type and a target database, these properties are used to assist with filtering when associating table operations to templates. **Note:** Not all operations support template script generation on all target databases. Utility type templates can contain common code for use by other templates.
Hub		A **Hub** is a table of unique business keys, they usually contain a hash key, business key(s), load date and record source. Hubs should normally have at least one Satellite.
Link		**Links** are many-to-many tables representing current and past relationships between two or more Hub entities and are used to describe associations, transactions, hierarchies and redefinitions of Hub entities in a Data Vault. Links have their own hash key and the hash keys for the Hubs that are linked, as well as a Load Date and Record Source. The attributes describing the context of a link are stored in Satellite Tables.
Satellite		**Satellites** are Data Vault objects that contain metadata which provides context for Hub and Link entities at a given time or over a period of time. Each Satellite entity can contain information on one Hub or Link. Satellite tables contain a hash key for the parent Hub or Link, a timestamp for the date of change and relevant descriptive fields. Satellites are usually created once per source system. Because descriptive attributes can change at different rates, Satellites can also be created based on rate of change.
Source Mapping		**Source mapping** objects can be defined for a number of table object types, for example Stage tables, Hubs, Links and Satellites. They enable these target tables to be sourced from more than one set of source tables. Source mapping objects don't result in data warehouse tables but contain source table and column information metadata, including any transformations and join criteria. This metadata is used to generate a procedure or script to populate the target table. Source mapping objects support more than one insert/update routine from different sources to be defined for the target table, and either executed collectively or independently. Refer to **Multi Source Processing** for details.

Object Type	Icon	Description
Scheduler Jobs	⏱	The **Scheduler Jobs** object group lists all the jobs that have been defined in WhereScape RED. Jobs, such as data loads and updates can be run in background mode and/or at a pre-determined time, using the RED Scheduler. Jobs can be added or removed from a project. Refer to **Scheduler** for details
Parameters	(x)	The **Parameters** object group lists all the parameters that have been defined in WhereScape RED. Parameters are a means of passing information between two or more procedures and between the WhereScape RED environment and procedures. Parameters can be added or removed from a project. Refer to **Parameters** for details
Custom1/ Custom2	▦	**Custom1** and **Custom2** objects are user defined objects. These Object Types can be renamed in the **Tools > Options > Object Types > Object Names** menu.

Connections are normally the first objects created. These connections are then used in the creation of Load tables through the drag and drop functionality. Subsequent objects can also be created through the use of drag and drop.

Note:

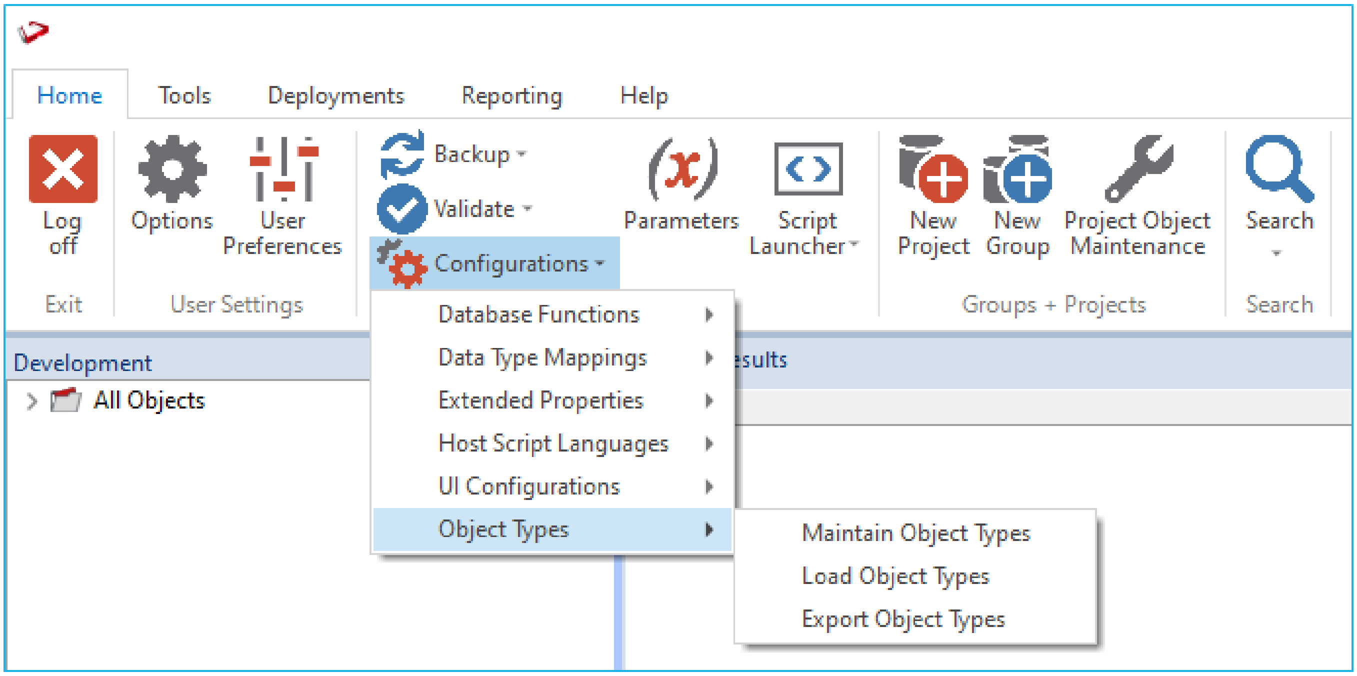
Although the object types have names that correspond with their primary usage, they can be used for other purposes. For example, the Fact object type could be used to create persistent Stage tables, if required.

Some objects are not supported by all databases, and some advanced properties are specific to the different databases.

Define Object Types

Object subtype definitions consist of the following:

Property	Description
Object type name	The parent object type
Object subtype name	The name of the object subtype
Description	A description to detail the purpose of the object subtype.
Icon	The icon to be displayed for the subtype in the repository trees and generated documentation.
Default update template	Allows the default templates for update routines to be specified by object type and subtype.
Default custom update template	Allows the default templates for custom update routines to be specified by object type and subtype.

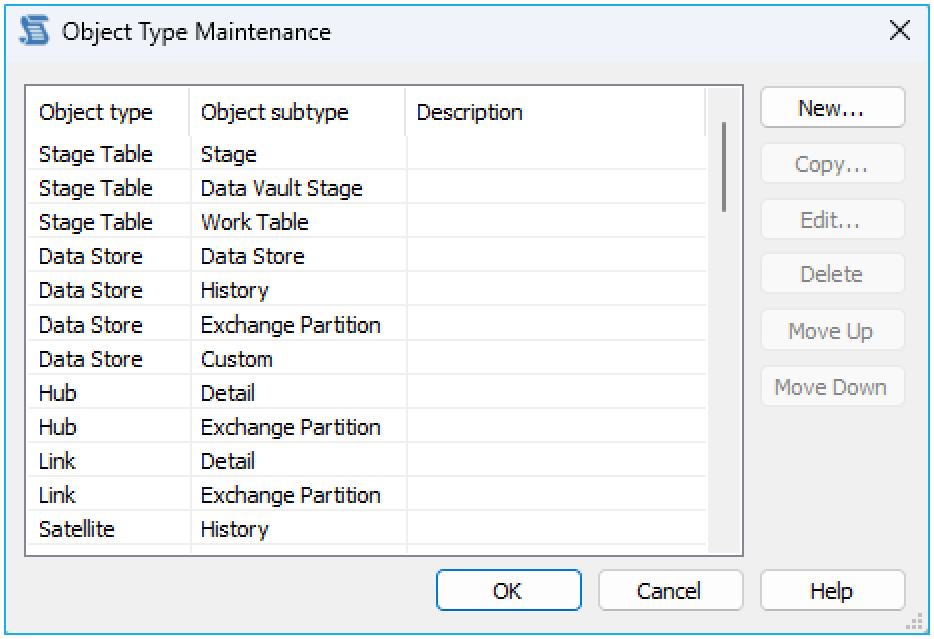


The following options are available:

- **Maintain Object Types** is used to add new or edit existing object type definition settings.
- **Load Object Types** to load a set of object type definitions into RED from an external source. This option is typically used with a WhereScape provided enablement pack.
- **Export Object Types** is used to export the available object types in RED to an object type file (`.objtype`).

Maintain Object Type Configurations

The Maintain Object Types screen displays a list of object types and subtypes configured in RED and enables you to add **New** object subtypes. Clicking an object subtype from the list enables you to **Copy**, **Edit** or **Delete** the selected subtype.



Object type	Object subtype	Description
Stage Table	Stage	
Stage Table	Data Vault Stage	
Stage Table	Work Table	
Data Store	Data Store	
Data Store	History	
Data Store	Exchange Partition	
Data Store	Custom	
Hub	Detail	
Hub	Exchange Partition	
Link	Detail	
Link	Exchange Partition	
Satellite	History	

Object Subtypes

This pane lists the built-in and user defined object subtypes that have been setup in RED. It details the name and description of each object subtype available in RED.

Function Buttons

These buttons enable you to create and manage the user defined object type definitions.

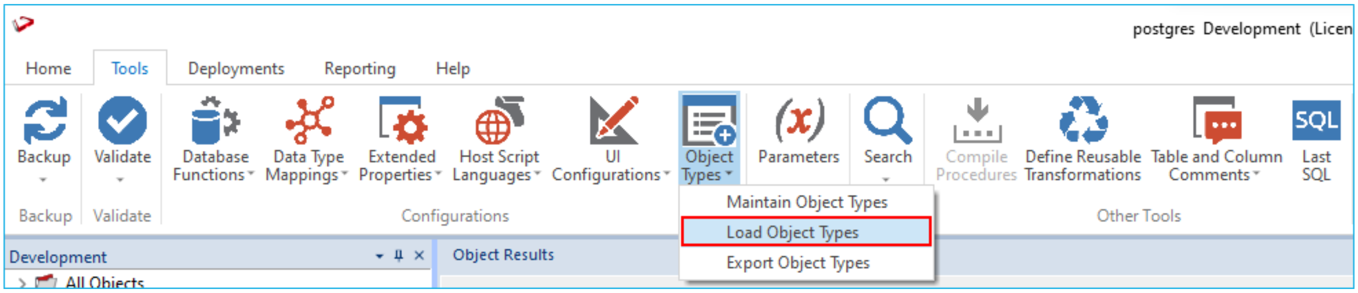
Property	Description
New	Launches the **Edit Object Type** screen which enables users to create and specify the properties of the new host script language type.
Copy	Duplicates the selected object subtype. All fields are populated with the values of the original object subtype which can be edited as required.
Edit	Launches the **Edit Object Type** screen which enables users to edit the selected host object subtype.
Delete	Deletes the selected object subtype, if it is not currently in use. The user is notified if it is in use and cannot be deleted.

Object Type Data Migration Between Repositories

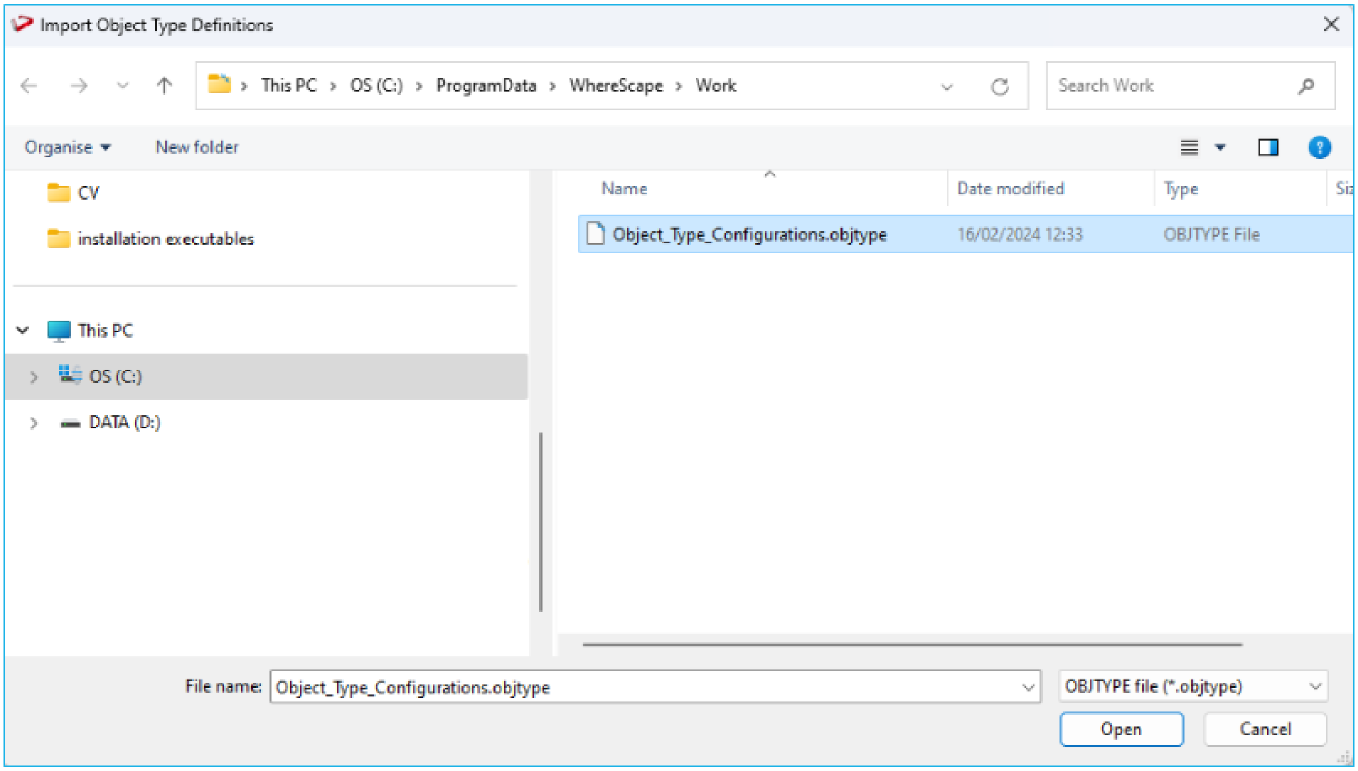
Object type configurations defined and set can be propagated to the other RED repositories. The definitions are exported from the source repository and imported to the target repository via the **Tools > Object Types** menu. Additionally the export and import of Object Types is also provided via the command line using `RedCli.exe object-type-definition` import and export commands.

Load Object Type Configurations

To load an object type file, select **Tools > Object Types > Load Object Types**



The following window displays. Select the object type (`.objtype`) file to load.

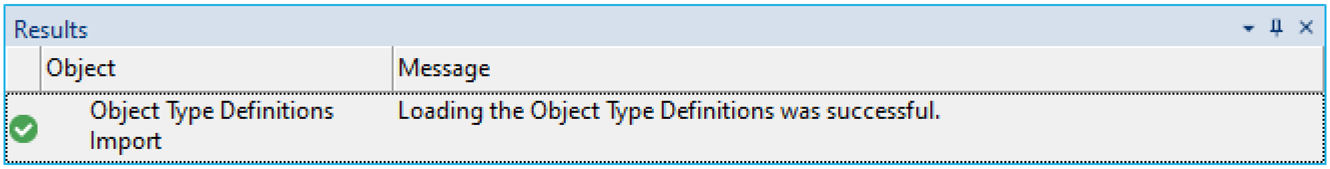


Object_Type_Configurations.objtype 16/02/2024 12:33 OBJTYPE File

File name: Object_Type_Configurations.objtype

OBJTYPE file (*.objtype)

The selected `objtype` file is loaded and listed in the **Maintain Object Types** screen and a confirmation message displays in the **Results** pane.

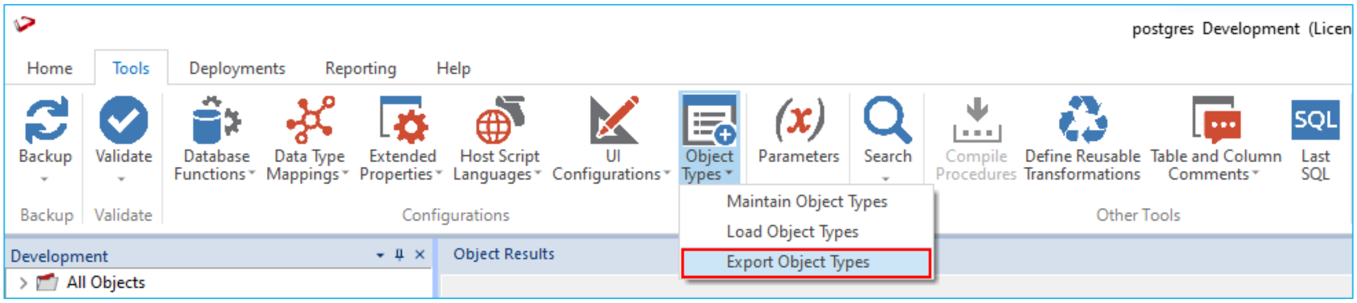


Results

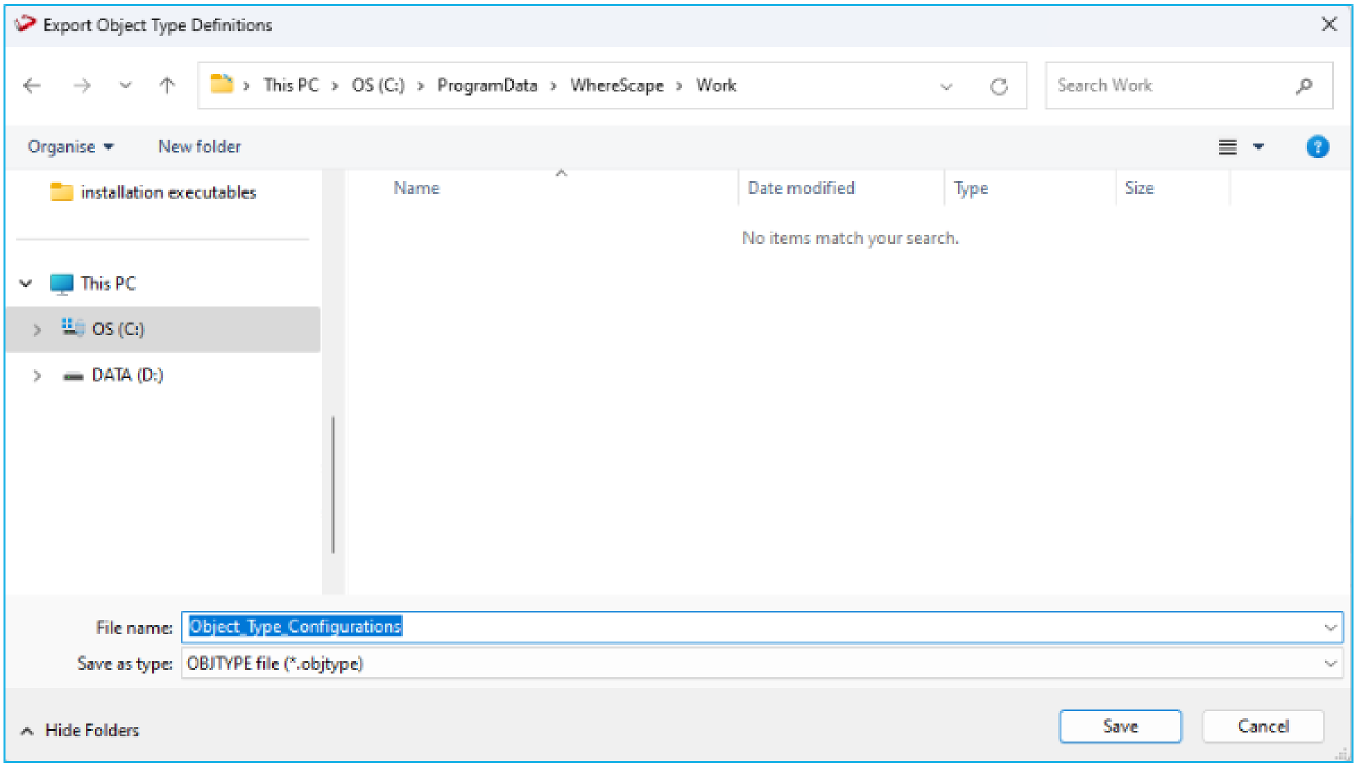
Object	Message
✓ Object Type Definitions Import	Loading the Object Type Definitions was successful.

Exporting Object Type Configurations

To export object type configurations from RED, select **Tools > Object Types > Export Object Types**.



The following window displays. Type in the **File name** and click **Save**.



Export Object Type Definitions

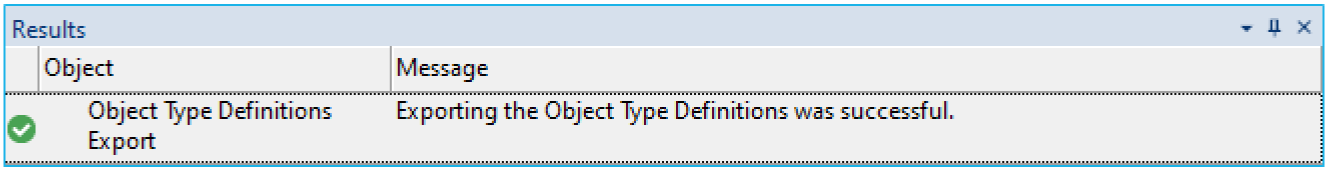
This PC › OS (C:) › ProgramData › WhereScape › Work

No items match your search.

File name: Object_Type_Configurations

Save as type: OBJTYPE file (*.objtype)

A confirmation message displays in the **Results** pane.



Results

Object	Message
✓ Object Type Definitions Export	Exporting the Object Type Definitions was successful.

Working with Objects

Most object types perform some form of action in the data warehouse. For example; Dimension, Stage, Fact and Aggregate table based objects are 'Updated' in the data warehouse via the defined update procedure.

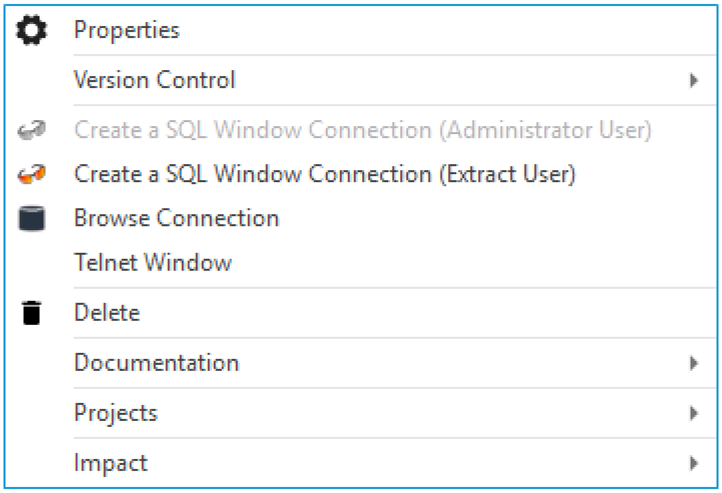
Procedures can be executed in the database. You can right-click objects listed in the left pane of the WhereScape RED **Builder** window to open a context menu which provides a number of options for manipulating the object. This can also be done for objects listed in the middle pane.

Further options are available through the menus provided in the various windows.

The operations of each of the objects is discussed in the sections below.

Connections

Connections once defined, are typically browsed and used as a source for drag and drop operations.
For database connections, a database link is normally required. This link can be created via the right-click context menu of the selected connection.


Properties
Version Control
Create a SQL Window Connection (Administrator User)
Create a SQL Window Connection (Extract User)
Browse Connection
Telnet Window
Delete
Documentation
Projects
Impact

Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected connection, pressing **V** creates a new version of the connection.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

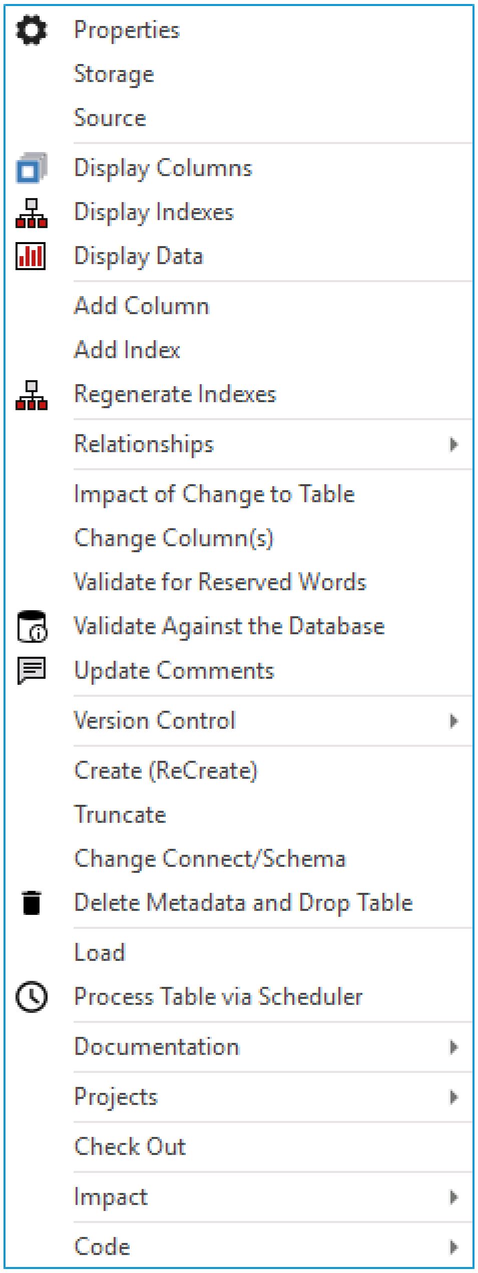
Menu Options	Description
Properties	Displays the **Properties** window for the selected connection.
Version Control	A version of a Connection is a copy of the metadata definition of the Connection at the time of the versioning. This version information can be used to create a new Connection object or can simply be left as a backup and reference point. Select this option to perform one of the following: • **New Version** - Enables you to create a new version of the selected Connection. • **Duplicate Object** - Enables you to create a new Connection object as a duplicate of the selected Connection. • **Revert to Version** - Enables you to revert to a previous version of the selected Connection. Displays a list of the available versions of the selected Connection object, from which you can select and revert.
Create a SQL Window Connection	Opens the RED **SQL Admin** tool which enables you to perform SQL queries in the database connection.
Browse Connection	Displays the **List Source Tables Connection** window which enables you change the properties of the connection you are browsing or switch to another connection. Refer to **Browsing a Connection** for details.
Telnet Window	Creates a Telnet window for UNIX connections.
Create Remote View Procedure	Creates a remote view creation procedure, if required for database connections and loads. Refer to **Remote View Extract** for details.
Create Database Link	Establishes a database link after the relevant connection has been created in RED. Refer to **Creating a Database Link** for details.
Delete	Deletes the meta data of the selected connection, including all related information. A version of the object's meta data is normally auto created (depends on the settings in **Tools > Options > Metadata Versioning**).
Documentation	Select this option to generate (or read if already generated) the WhereScape RED HTML documentation for the selected object. Select this option to perform one of the following: • **Display** the documentation available for the selected object. Refer to **Reading the**

Menu Options	Description
	Documentation for details. • **Create** the documentation for the selected object. Refer to Creating Documentation for details.
Projects	Select this option to perform one of the following: • **Remove from Project** - Select to remove the Connection from a project. Refer to Removing Objects from Projects for details. • **Add to Project** - Select to add the Connection to a project. Refer to Adding Objects to Projects for details. • **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to List Project Memberships for Objects for details. Refer to Organizing Objects for details. **Notes:** • Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane. • If there aren't any projects in the repository, the above options are unavailable.
Impact	Select this option to run reports on Load and Export objects associated with the connection. Select this option to perform one of the following: • **Load Tables Report** • **Export Tables Report** Refer to Table Reports for details.
Build	Select this option to build an application based on the connection(s) selected from the middle pane. Refer to Building Applications from Object Groups for details. This option is available from the context menu of connections listed in the middle pane.

Load Tables

Load tables once defined would normally be created and loaded, unless these actions were performed as part of the drag and drop operation.

The menu below shows the operations that can be performed on Load tables.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **L** performs an interactive load, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Menu Options	Description
Properties	Displays the **Properties** window for the Load table, albeit focused on different tabs within this window.
Storage	Displays the **Properties** window for the Load table, albeit focused on different tabs within this window.
Source	Displays the **Properties** window for the Load table, albeit focused on different tabs within this window.

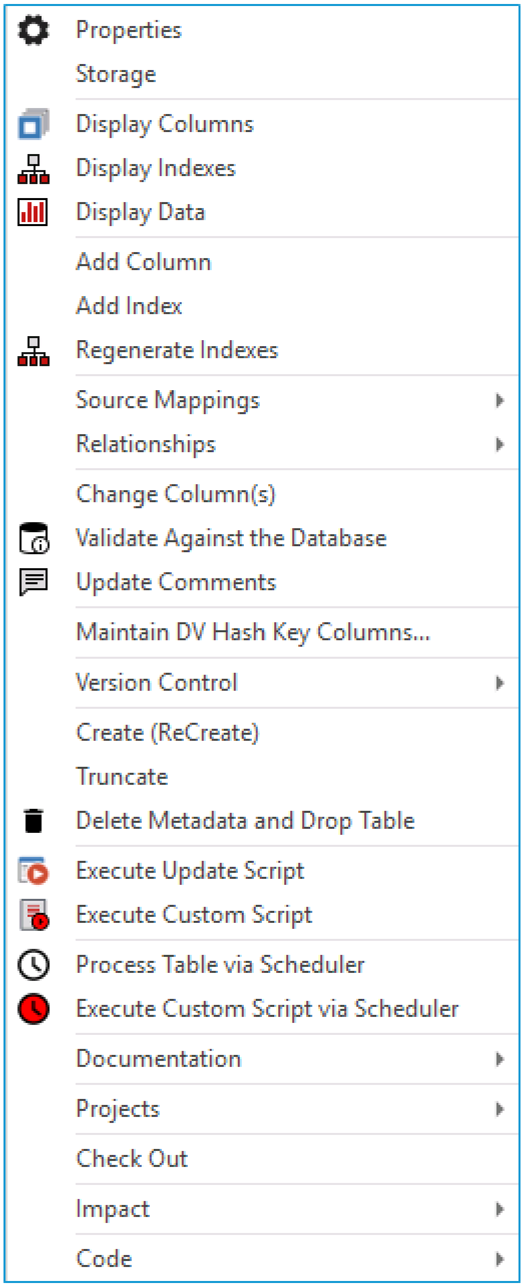
Menu Options	Description
Display Columns	Displays columns of the load table.
Display Indexes	Displays indexes of the load table.
Display Data	Displays data of the load table. **Tip:** If the data is displayed, only the first 100 rows are returned from the table. Either the SQL Admin tool (accessible via the WhereScape start menu option), or the Excel query must be used if more detailed data analysis is required.
Query via Excel	Click to query columns in Microsoft Excel. **Tip:** When a column list has been displayed in the central pane, it is sorted based on the **order** field associated with each column. Clicking the column label **Col name** will sort the columns into alphabetical order. A subsequent click will re-sort based on the **order** field.
Add Column	Select this option to add new columns. Normally columns can be added via drag and drop.
Add Index	Select this option to add new indexes. Normally indexes are created during the procedure generation phase.
Regenerate Indexes	Select this option to add missing standard indexes. Selecting this option displays a window with options to regenerate missing indexes in the metadata and recreate them or to just regenerate the missing indexes in the metadata.
Relationships	Select this option to manage enhanced relationships. Select this option to perform one of the following: • **Add Relationship** - Displays the Add Relationships dialog. • **List Relationships** - Displays a list of enhanced relationships in the **Drop Target Pane** for the selected object. • **Generate Relationships** - Generates relationships which have not yet been defined in the metadata
Impact of Change to Table	Select this option to produce a list of objects that will be potentially impacted by a change to the load table structure.
Change Column(s)	Select this option to apply changes to a selected number of columns
Validate for Reserve Words	Select this option to produce a list of table or column names where reserved words have been used; enabled for supported ODBC Drivers.
Validate Against the database	Select this option to compare the metadata for the load table with the physical table resident in the database, and where required the table is altered to match the metadata.
Update Comments	Select this option to refresh table and column comments on the table, using the table's description and columns' business definition. **Note:** You can manage the table and column comments outside the data warehouse environment via the **Table and Column Comments** menu option.
Version Control	A version of a Load table is a copy of the metadata definition of the table at the time of the versioning. This version information can be used to create a new Load table or can simply be left as a backup and reference point. Select this option to perform one of the following: • **New Version** - Enables you to create a new version of the selected Load table. • **Build Application** - Enables you to build an application file for the selected Load table. • **Duplicate Object** - Enables you to create a new Load table as a duplicate of the selected table.

Menu Options	Description
	- **Revert to Version** - Enables you to revert to a previous version of the selected Load table. Displays a list of the available versions of the selected Load table, from which you can select and revert. **Tip:** The extended property values set in the original Load table is copied to the duplicate Load table. Refer to **Extended Properties** for details.
Create (ReCreate)	Select this option to create the table in the database based on the definition stored in the metadata. To alter a table select the Validate against database option. Refer to section on table validation for details.
Truncate	Select this option to truncate the table.
Change Connect/ Schema	This option enables the rapid changing of the Connection information associated with the Load table. This information can be changed in-bulk for a number of Load tables. Refer to **Changing Load Connection and Schema** for details.
Delete Metadata and Drop Table	Select this to option perform one of the following delete options: - **Delete metadata and drop object** - This option deletes the metadata definition for the table and drops the table object from the database (default). This is a permanent delete and no recovery is provided, use with caution. - **Delete metadata only** - This option deletes the metadata definition for the table. A version of the object's metadata is normally auto created (depends on the settings in **Tools > Options > Metadata Versioning**).
Load	Select this option to perform an interactive load of the data. The method of loading depends on the type of connection. This menu option is intended for use with small data volumes as in a prototype environment. Large data volumes would normally be scheduled. The Load locks the WhereScape RED screen until completed. **Tips:** - The load option does not drop or create any indexes. Use the Process option if indexes need to be maintained.
Process Table via Scheduler	Select this option to send a request to the Scheduler to immediately process the Load table. This process drops any indexes marked as pre_drop, load the data and rebuild any required indexes. Control is immediately returned to the user and the loading occurs via the Scheduler.
Documentation	Select this option to generate (or read if already generated) the WhereScape RED HTML documentation for the selected object. Select this option to perform one of the following: - **Display** the documentation available for the selected object. Refer to **Reading the Documentation** for details. - **Create** the documentation for the selected object. Refer to **Creating Documentation** for details.
Projects	Select this option to perform one of the following: - **Remove from Project** - Select to remove the Load table from a project. Refer to **Removing Objects from Projects** for details. - **Add to Project** - Select to add the Load table to a project. Refer to **Adding Objects to Projects** for details. - **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to **List Project Memberships for Objects** for details.

Menu Options	Description
	Notes: • Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane. • If there aren't any projects in the repository, the above options are unavailable.
Check Out	Select this option to check out the object for editing and prevent any other users from being able to modify, update or delete any of their associated objects while you are making changes. Refer to **Object Check-Outs and Check-Ins** for details.
Impact	Select this option to produce a number of reports and diagrams. Select this option to perform one of the following: • **Track Back Report** • **Track Forward Report** • **Track Back Diagram** • **Track Forward Diagram** • **Dependent Jobs Report**
Code	Select this option to view a procedure associated to a table or to regenerate the table's update procedure. Select this option to perform one of the following: • **View Load script** • **View Post Load script** • **Regenerate Load script** • **Rebuild Post Load script** • **Regenerate Post Load script** **Notes:** • Only Load tables with one or more defined procedures have the Code view option. • If the Load table is Script-based load type, either option is displayed in the Code sub menu: o Generate Load Script (if a script is not yet generated) o Regenerate <script name> (if a script has already been generated). Refer to **Script based load** for details.
Build	Select this option to build a Scheduler job or an application based on the Load table(s) selected from the middle pane. Refer to **Building Scheduler Jobs from Object Groups** and **Building Applications from Object Groups** for details. This option is available from the context menu of Load tables listed in the middle pane.

Stage Tables

Following is an example of a standard menu for Stage tables.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays data, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Tip:

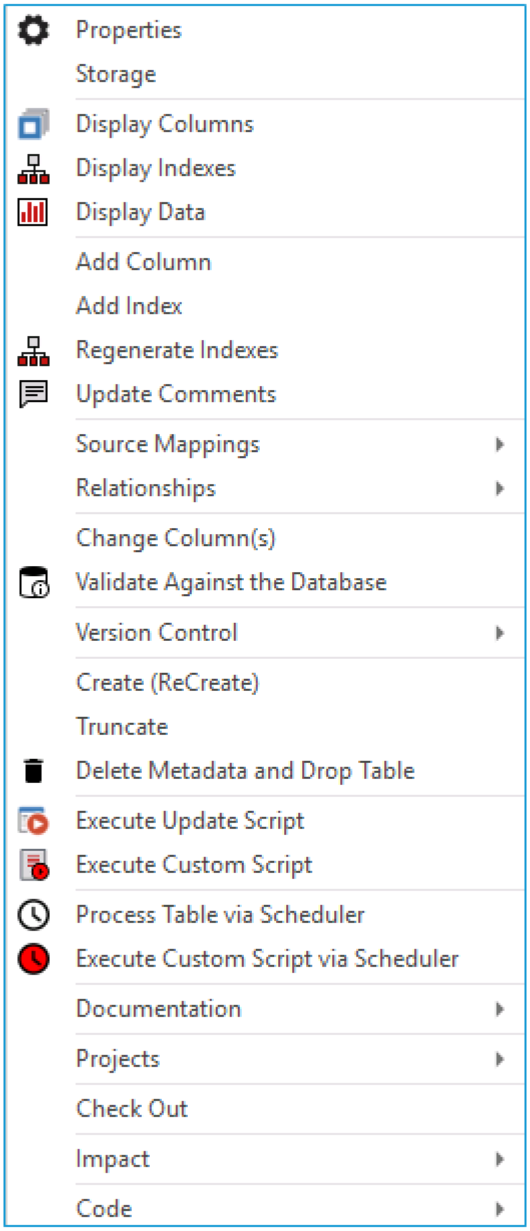
The menu options available for Stage tables is a subset of the options for **Data Store Tables**, except for the **Report Zero Keys...** element.

Menu Options	Description
Report Zero Keys...	Select this option to initiate the report Records that Failed a Dimension Join. Refer to

Menu Options	Description
	Records that Failed a Dimension Join for details.

Data Store Tables

Following is an example of a standard menu for Data Store tables.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays data, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Tip:

Tip:

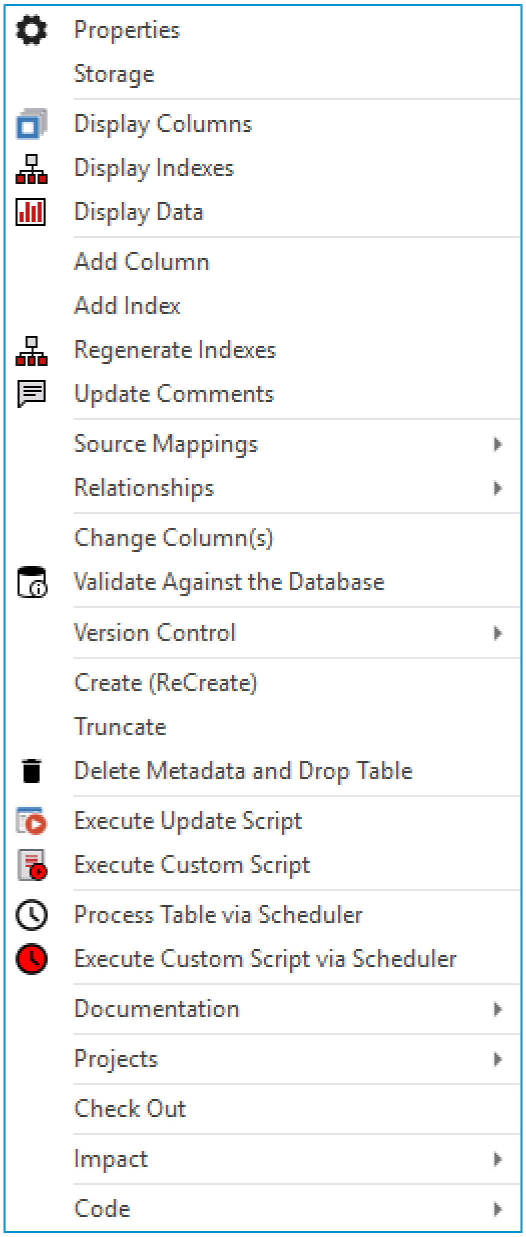
The menu options available for Data Store tables is a subset of the options for **Dimension Tables** including the Code menu options.

Menu Options	Description
Code	Select this option to view a procedure attached to a table or to rebuild or regenerate the table's update procedure. Select this option to perform one of the following: • View update_ds_product • View custom_ds_product • Rebuild update_ds_product • Regenerate update_ds_product • Rebuild custom_ds_product • Regenerate custom_ds_product • Display Procedure Build Options Choose a procedure from the list to open in the procedure editor in view mode or choose to rebuild or regenerate the update procedure.
Build	Select this option to build a Scheduler job or an application based on the Data Store table(s) selected from the middle pane. Refer to **Building Scheduler Jobs from Object Groups** and **Building Applications from Object Groups** for details. This option is available from the context menu of Data Store tables listed in the middle pane.

Hub Tables

Hub table objects in WhereScape RED contain hash key(s), business key(s), load date and record source. This table can be created via a Wizard—refer to **Creating the Hub, Link and Satellite Tables** for details.

Following is an example of the menu displayed when you right-click a Hub table object.



Tip:

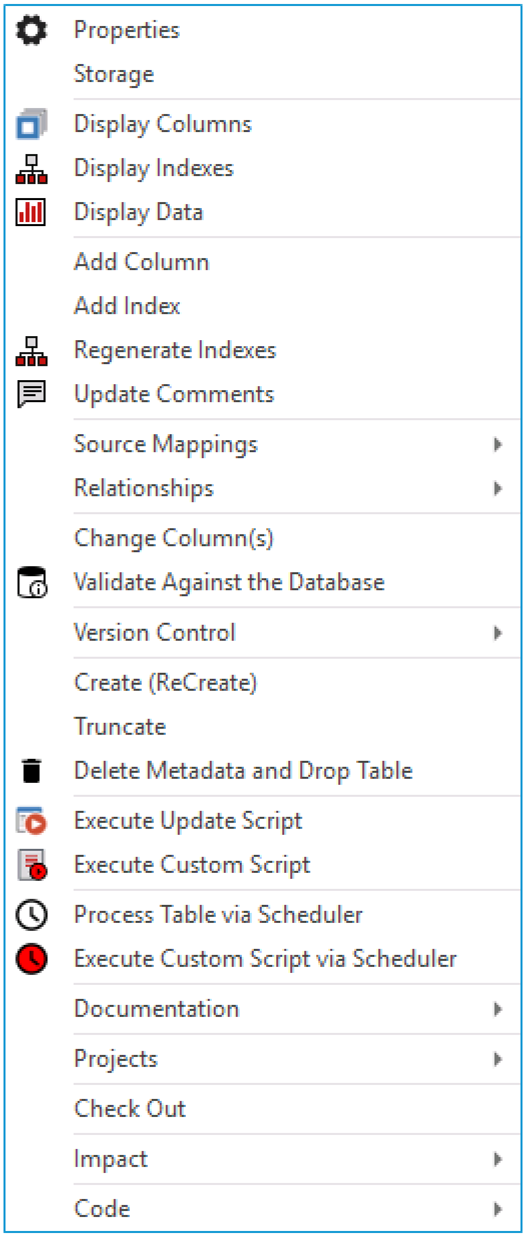
The bulk of the menu options are same as for the other table objects described in this section.

Link Tables

Link table objects are many-to-many tables representing current and past relationships between two or more Hub entities and are used to describe associations, transactions, hierarchies and redefinitions of Hub entities in a Data Vault. Links have their own hash key and the hash keys for the Hubs that are linked, as well as a Load Date and Record Source. The attributes describing the context of a link are stored in Satellite tables.

This table can be created via a Wizard—refer to **Creating the Hub, Link and Satellite Tables** for details.

Following is an example of the menu displayed when you right-click a Link table object.



Tip:

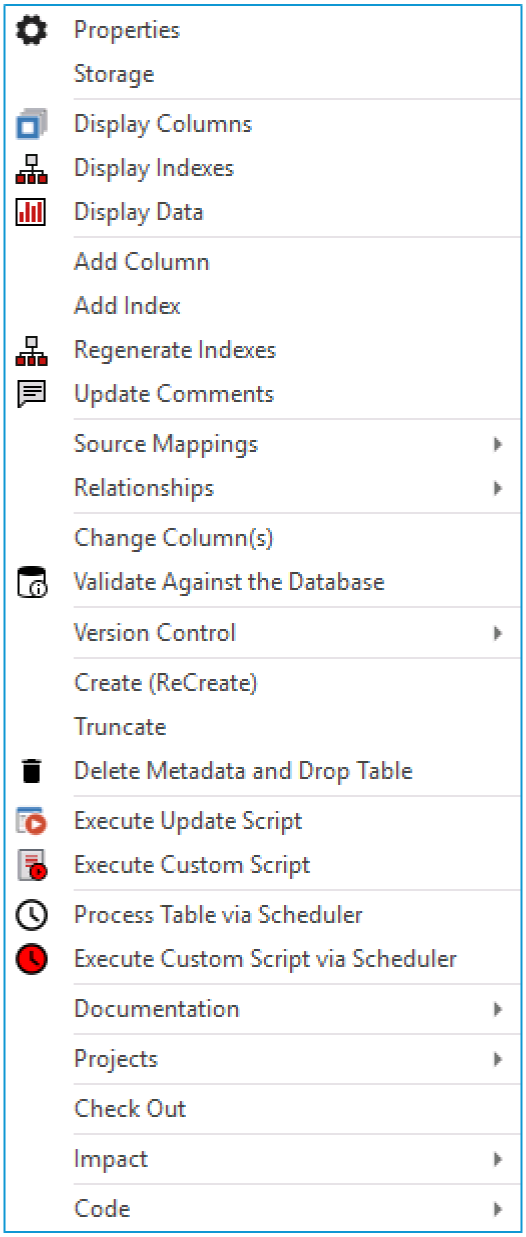
The bulk of the menu options are same as for the other table objects described in this section.

PiT Bridge Reference

Satellite tables are Data Vault objects that contain metadata which provides context for Hub and Link entities at a given time or over a period of time. Each Satellite entity can contain information on one Hub or Link. Satellite tables contain a hash key for the parent Hub or Link, a timestamp for the date of change and relevant descriptive fields.
Satellites are usually created once per source system. Because descriptive attributes can change at different rates, Satellites can also be created based on rate of change.

This table can be created via a Wizard—refer to **Creating the Hub, Link and Satellite Tables** for details.

Following is an example of the menu displayed when you right-click a Satellite table object.



Tip:

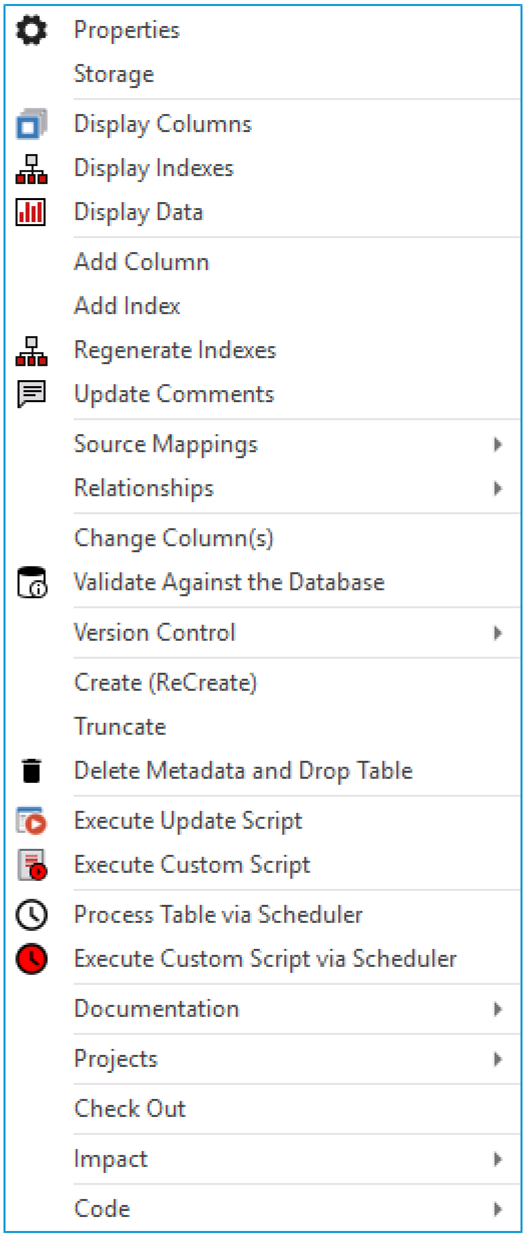
The bulk of the menu options are same as for the other table objects described in this section.

PiT Bridge Reference

Satellite tables are Data Vault objects that contain metadata which provides context for Hub and Link entities at a given time or over a period of time. Each Satellite entity can contain information on one Hub or Link. Satellite tables contain a hash key for the parent Hub or Link, a timestamp for the date of change and relevant descriptive fields.
Satellites are usually created once per source system. Because descriptive attributes can change at different rates, Satellites can also be created based on rate of change.

This table can be created via a Wizard—refer to **Creating the Hub, Link and Satellite Tables** for details.

Following is an example of the menu displayed when you right-click a Satellite table object.

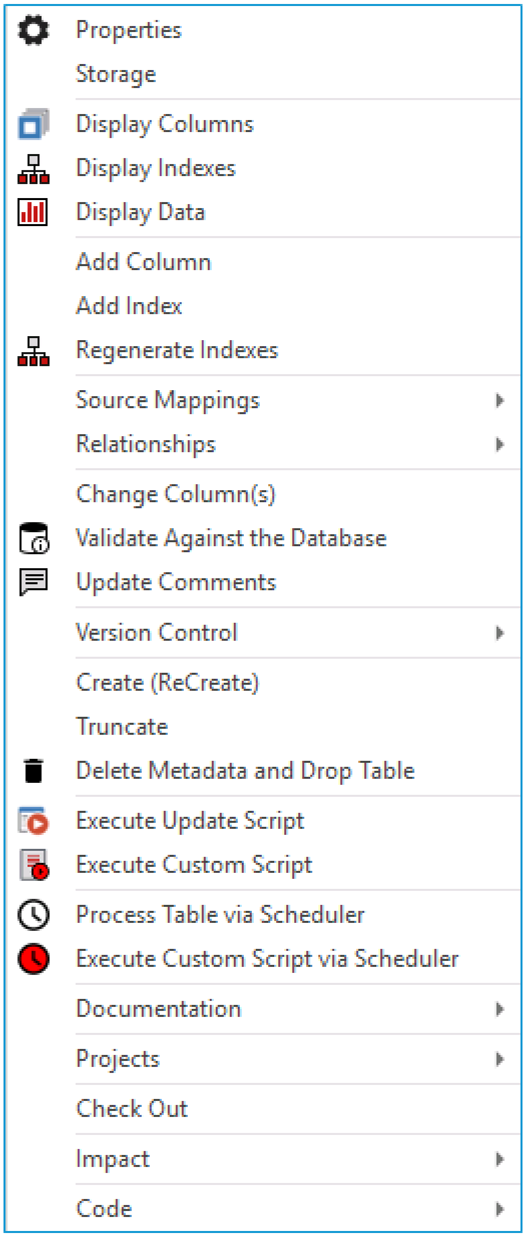


Tip:

The bulk of the menu options are same as for the other table objects described in this section.

Dimension Tables

The context menu for **Dimensions** is shown below:



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays data, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Tip:

The bulk of the menu options are same as for **Load tables**.

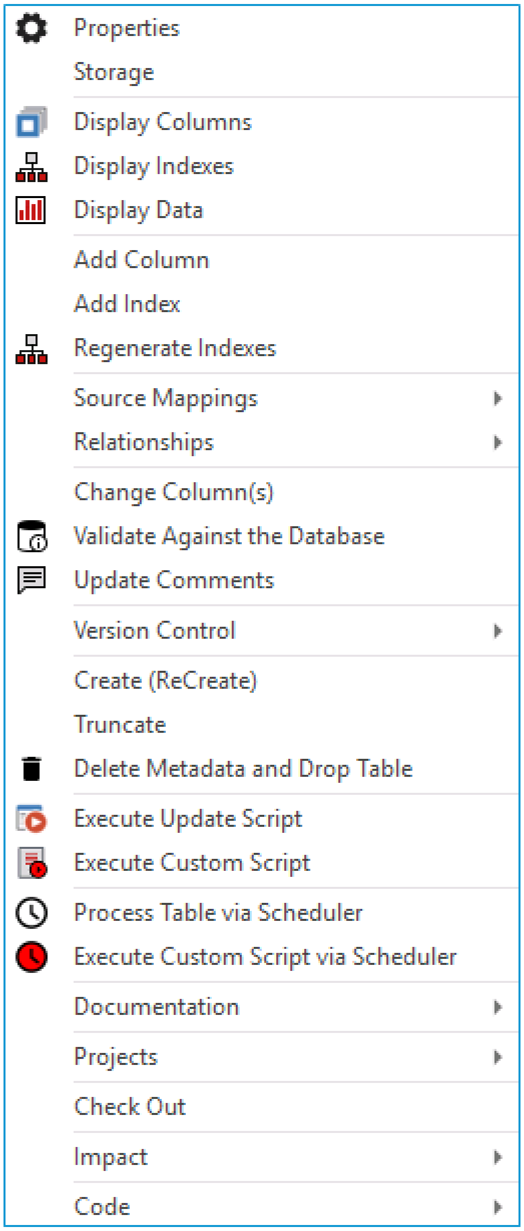
Menu Options	Description
Source Mappings	Source mapping objects are created when there is a requirement to have more than one set of source tables to populate the target table and provides for the generation of an update and custom procedure for each source mapping object. Select this option to

Menu Options	Description
	perform one of the following: - **Add Source Mapping** - Select to add a source mapping. - **List Source Mappings** - Select to list a source mapping. Refer to **Multi Source Processing** for details.
Hierarchies	Select this option to perform one of the following: - **Add Hierarchy** - Select to add hierarchies. - **List Hierarchies** - Select to list hierarchies. - **List Hierarchy Elements** - Select to list hierarchy elements. - **Copy Hierarchies from Source** - Select to copy all hierarchies from the source table to the destination table. Source hierarchies are copied to the destination table automatically during table creation, but this feature is useful if the source table has been updated since destination table was created. Refer to **Dimension Hierarchies** for details.
Relationships	Select this option to perform one of the following: - **Add Relationship** - Select to add relationships. - **List Relationships** - Select to list relationships. - **Generate Relationships** - Select to generate relationships. Refer to **Relationship Maintenance** for details.
Execute Update Procedure	Select this option to execute the procedure defined as the 'update procedure' for the table. The procedure is executed interactively and locks the screen until completed. This menu option is only intended for use when working with small/prototype data volumes, and no index handling is performed.
Execute Custom Procedure	Select this option to execute the procedure defined as the 'custom procedure' for the table. As with the update procedure, the custom procedure is executed interactively.
Process Table via Scheduler	Select this option to process the selected table via the WhereScape RED scheduler. Refer to **Scheduler** for details.
Execute Custom Procedure via Scheduler	Select this option to execute the procedure defined as the 'custom procedure' for the table via the WhereScapeRED scheduler. Refer to **Scheduler** for details.
Code	Select this option to view a procedure attached to a table or to rebuild the table's update procedure; and to view or rebuild the Get Key function on the Dimension/Dimension View. **Tip:** The Get Key function is not available for dimension objects that are stored in custom database targets. Select this option to perform one of the following: - View update_dim_customer - View custom_dim_customer - View get_dim_customer_key - Rebuild update_dim_customer - Regenerate update_dim_customer - Rebuild custom_dim_customer - Regenerate custom_dim_customer - Rebuild get_dim_customer_key **Note:** Only tables with one or more defined procedures have the Code option.
Build	Select this option to build a Scheduler job or an application based on the Dimension

Menu Options	Description
	table(s) selected from the middle pane. Refer to **Building Scheduler Jobs from Object Groups** and **Building Applications from Object Groups** for details. This option is available from the context menu of Dimension tables listed in the middle pane.

Fact Tables/KPI Fact Tables

Following is an example of a standard menu for Fact Tables and KPI Fact tables.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays data, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is

Note:

enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

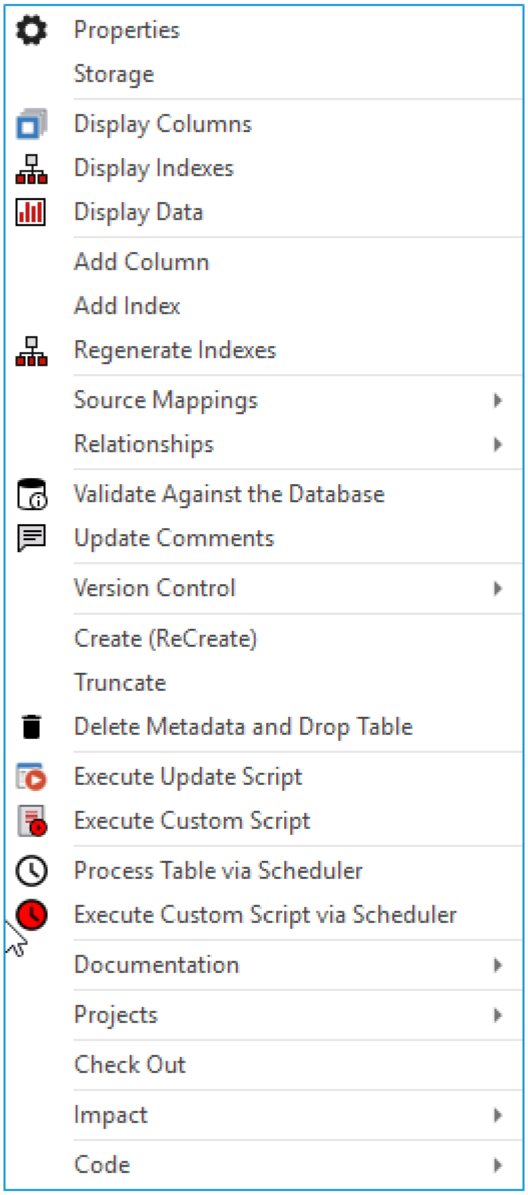
Tip:

The menu options available for Fact Tables are a subset of the options for **Data Store Tables**. KPI Fact Tables have three additional menu options as mentioned below.

Menu Options	Description
Display KPI's	Select this option to display all KPIs that have been set up for the KPI fact table in the Drop Target pane (middle pane).
Add KPI's	Select this option to add a new KPI.
Create Script To Test All KPI's	Select this option to generate an SQL script to test the KPI definitions for a specified time window.

Aggregate Tables

Following is an example of a standard menu for Aggregate tables.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays data, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

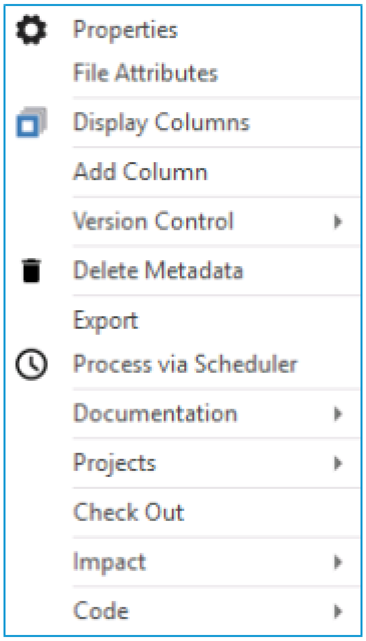
Tip:

The menu options available for Aggregate tables is a subset of the options for **Data Store Tables**.

Exports

Export objects are used in WhereScape RED to produce ASCII files from a single database Table or View for a downstream feed. Some or all of the columns in a Table or View can be exported

Following is an example of the context menu displayed when you right-click an Export object.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **C** creates the index, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Tip:

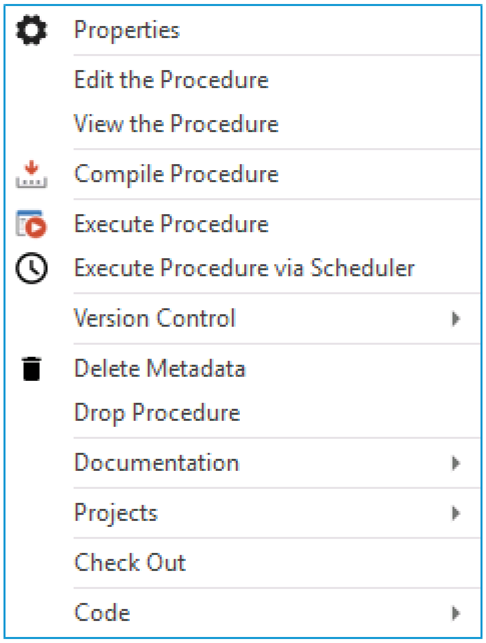
All the other context menu options for Export objects are described under the other object types.

Menu Options	Description
File Attributes	Select this option to display the **Properties** window for the Export table, focusing on the **File Attributes** tab within this window.

Procedures

Procedures are commonly auto built through the **Properties** screen of one of the table types. They can also be created manually. Once created, they can be edited, compiled, etc.

Following is an example of the menu displayed when you right-click a procedure name.



If a procedure has been locked as a result of the WhereScape RED utility being killed or failing or a database failure then it can be unlocked via the **Properties** screen associated with the procedure.

Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **E** displays the procedure for editing, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Menu Options	Description
Edit the Procedure	Select this option to invoke the procedure editor and loads the procedure. A procedure can be compiled and executed within the procedure editor.
View the Procedure	Select this option to display a read only copy of the procedure. If the procedure is locked by another user then viewing the procedure is the only option available.
Compile Procedure	Select this option to compile the procedure from the metadata.
Execute Procedure	Select this option to execute the procedure and displays the results in the results window.
Execute Procedure via Scheduler	Select this option to set up a job to execute the procedure via the scheduler.
Version Control	A version of a Procedure can be created at any time via the Version Control, New Version menu option. The various versions of the procedure can be viewed from within procedure editor, or a new procedure can be created from the version. Select this option to perform one of the following: • **New Version** - Enables you to create a new version of the selected Procedure. • **Build Application** - Enables you to build an application file for the selected Procedure. • **Revert to Version** - Enables you to revert to a previous version of the selected Procedure. Displays a list of the available versions of the selected procedure, from which you can select and revert.
Delete metadata	Select this to option perform one of the following delete options: • **Delete metadata and drop object** - This option deletes the metadata definition for the procedure and drops the procedure object from the database (default). This is a

Menu Options	Description
	permanent delete and no recovery is provided, please use with caution. • **Delete metadata only** - This option deletes the metadata definition for the procedure. A version of the object's metadata is normally auto created (depends on the settings in **Tools > Options > Metadata Versioning**).
Drop Procedure	Select this option to drop the procedure from the metadata and from the database as well. **Note:** Procedures cannot be dropped or deleted if they are in use or being edited with the Edit Lock set in the **Procedure Properties** window.
Documentation	Select this option to generate (or read if already generated) the WhereScape RED HTML documentation for the selected object. Select this option to perform one of the following: • **Display** the documentation available for the selected object. Refer to **Reading the Documentation** for details. • **Create** the documentation for the selected object. Refer to **Creating Documentation** for details.
Projects	Select this option to perform one of the following: • **Remove from Project** - Select to remove the procedure from a project. Refer to **Removing Objects from Projects** for details. • **Add to Project** - Select to add the procedure to a project. Refer to **Adding Objects to Projects** for details. • **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to **List Project Memberships for Objects** for details. **Note:** Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane.
Check Out	Select this option to check out the object for editing to prevent any other users from being able to modify, update or delete any of their associated objects while you are making changes. Refer to **Object Check-Outs and Check-Ins** for details.
Build	Select this option to build a Scheduler job or an application based on the Procedure table(s) selected from the middle pane. Refer to **Building Scheduler Jobs from Object Groups** and **Building Applications from Object Groups** for details. This option is available from the context menu of Procedure tables listed in the middle pane.

Host Scripts

Scripts are very similar in their operations to **Procedures**. Some key differences are as follows:

- If a script is deleted, it is only removed from the metadata. However, because WhereScape RED never stores the script on the host system, this removes the script permanently.
- Unix scripts, Windows CMD scripts, **Windows PowerShell Scripts**, and User Defined Host Script Languages are available on Load objects.
- Unix scripts, Windows Scripts and User Defined Host Script Languages are available on all table object types, where the table is stored in a custom database target and when using a Windows scheduler.
- The table object needs to specify the connection appropriate for the script (typically Windows or Unix).

Referencing other scripts at run-time:

Scripts stored in your RED Metadata can be referenced from other Scripts at run-time using token replacement. This feature is only available on Custom Target Database types on SQL Server Metadata Repositories.

- **Token replacement format**

  $WSL_SCRIPT_<your_script_name_in_RED>_CODE$

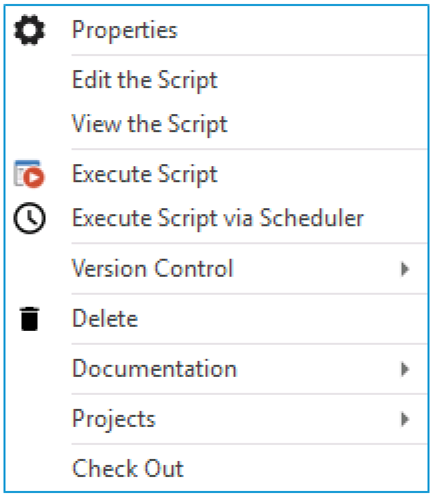
- **Replacement process at run-time**

  RED and RED Windows Scheduler parse the script for script reference tokens and if the script name is found in the metadata then it is written to the work directory and the token is replaced in the script with the full path to the script file before execution.

- **Example PowerShell use case**

  ```
  Import-Module -FullyQualifiedName "$WSL_SCRIPT_WslPowerShellCommon_CODE$" -
  DisableNameChecking
  ```

Following is an example of the menu displayed when you right-click a script object.



Note:

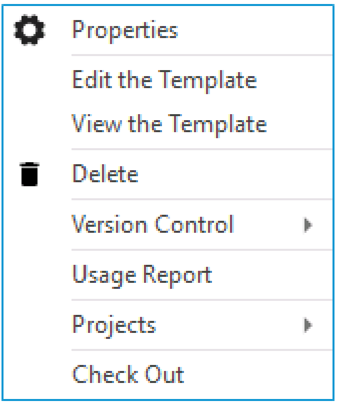
WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** deletes the script metadata, etc. Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Templates

Template objects are used to generate the update routines of table objects in RED, e.g. generation of DDL, update/custom procedures and host scripts. **Templates** are very similar in their operations to **Scripts**.

The additional context menu option is **Usage Report** which enables you identify the table objects that are associated with a selected Template object.

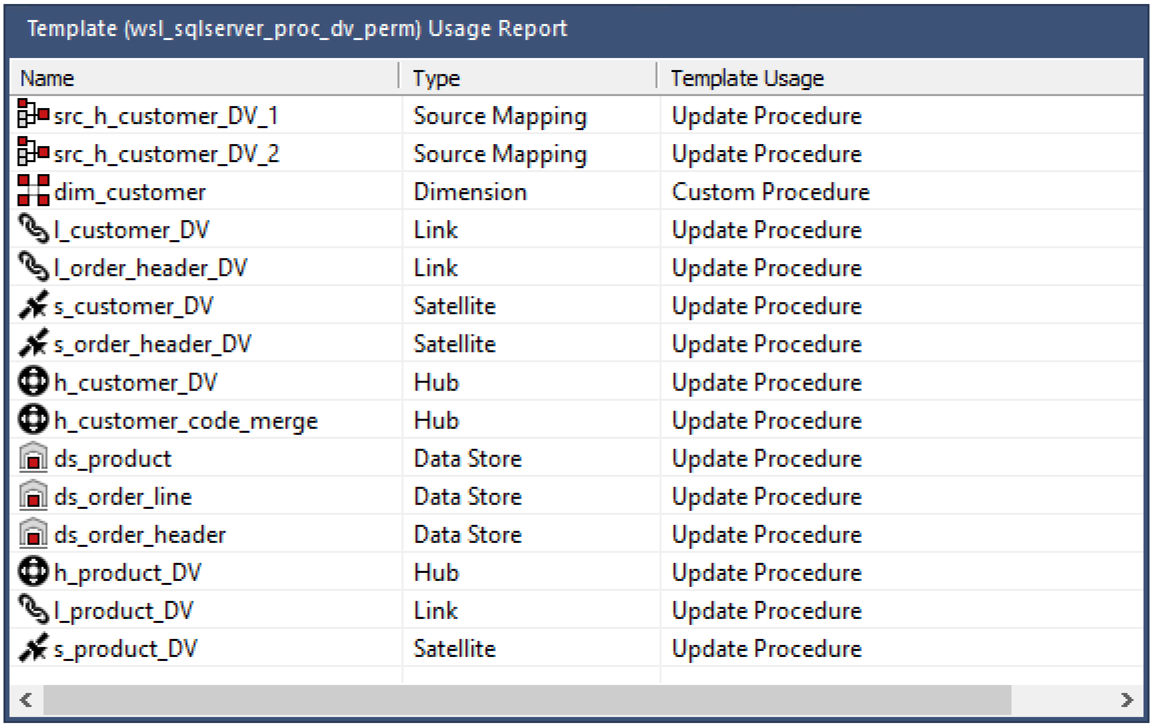
Following is an example of the menu displayed when you right-click a Template object.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected template, pressing **U** displays the RED objects that are using the template, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Selecting **Usage Report** displays a list of objects that are currently using the selected template in the middle pane.



Template (wsl_sqlserver_proc_dv_perm) Usage Report

Name	Type	Template Usage
src_h_customer_DV_1	Source Mapping	Update Procedure
src_h_customer_DV_2	Source Mapping	Update Procedure
dim_customer	Dimension	Custom Procedure
l_customer_DV	Link	Update Procedure
l_order_header_DV	Link	Update Procedure
s_customer_DV	Satellite	Update Procedure
s_order_header_DV	Satellite	Update Procedure
h_customer_DV	Hub	Update Procedure
h_customer_code_merge	Hub	Update Procedure
ds_product	Data Store	Update Procedure
ds_order_line	Data Store	Update Procedure
ds_order_header	Data Store	Update Procedure
h_product_DV	Hub	Update Procedure
l_product_DV	Link	Update Procedure
s_product_DV	Satellite	Update Procedure

You can select multiple objects from the list and then select **Code** from the right-click context menu to perform a bulk update procedure or script of the selected objects. Refer to Templates for details.

You can also define default templates to use for the update routines of table objects in RED by Connection type—refer to Routine Templates for details.

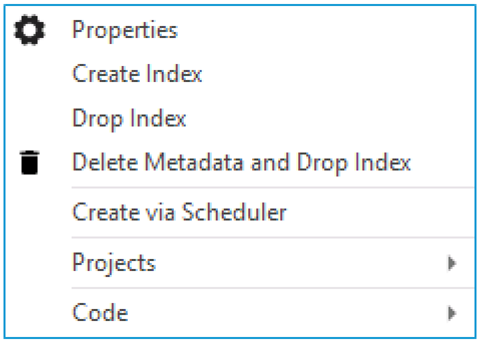
Note:

The **Version Control** sub menu for Template objects provide **New Version**, **Build Application** and **Duplicate Object** options.

Indexes

Indexes are always associated with a table. To define a new index, the menu option associated with the table that is to have the index must be used.

Following is an example of the context menu displayed when you right-click an index.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **C** creates the index, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

If a procedure has been locked as a result of the WhereScape RED utility being killed or failing or a database failure then it can be unlocked via the Properties screen associated with the procedure.

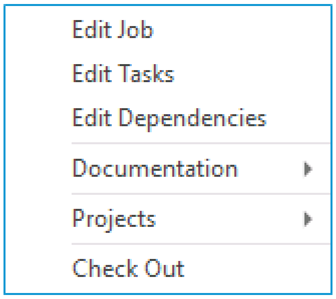
Menu Options	Description
Properties	Select this option to display the **Properties** screen, which contains the entire definition of the index, including the columns in use and the index type etc. The way that the scheduler handles the index is also defined in the Properties. An index can be set so that it is dropped by the scheduler prior to a table update and then rebuilt by the scheduler, once the update has been completed. It can also be defined for rebuild on certain days.
Create Index	Select this option to create an index in the database. **Tip:** It may take some time for large indexes, and in such cases, it is better to schedule a create of the index. Refer to **Scheduler** for details. This menu option is intended for use when working with prototype data volumes.
Drop Index	Select this option to drop the index in the database.
Delete Metadata and Drop Index	Select this option to remove metadata definition of the index and drops it from the database. No recovery is possible once this option is executed. A version of the object's metadata is normally auto created (depends on the settings in **Tools > Options > Metadata Versioning**).
Create via Scheduler	Select this option to submit a create of the index to the scheduler.
Projects	Select this option to: • **Remove from Project** - Select to remove the index from a project. Refer to **Removing Objects from Projects** for details. • **Add to Project** - Select to add the index to a project. Refer to **Adding Objects to**

Menu Options	Description
	Projects for details. - **List Projects -** Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to List Project Memberships for Objects for details. **Note:** Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane.
Build	Select this option to build a Scheduler job or an application based on the Index table(s) selected from the middle pane. Refer to **Building Scheduler Jobs from Object Groups** and **Building Applications from Object Groups** for details. This option is available from the context menu of Index tables listed in the middle pane.

Scheduler Jobs

Jobs, such as data loads and updates can be run in background mode and/or at a pre-determined time, using the RED scheduler. Refer to Scheduler for details.

Following is an example of the menu displayed when you right-click a scheduler job object.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **C** creates the index, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

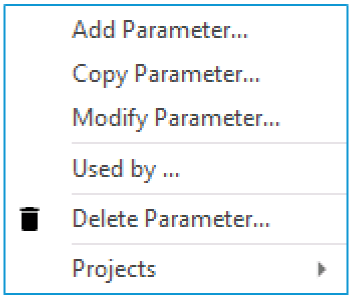
Menu Options	Description
Edit Job	Select this option to launch the **Job Definition** window which enables you to edit the job parameters and settings of the selected scheduler job. Refer to **Editing a Job** for details.
Edit Tasks	Select this option to launch the **Define tasks** window which enables you to edit the tasks included in the selected scheduler job. Refer to **Editing Tasks in a Job** for details.
Edit Dependencies	Select this option to launch the **Dependencies** window which enables you to edit the dependencies between the tasks included in the selected scheduler job. Refer to **Editing Task Dependencies** for details.
Documentation	Select this option to: - **Display** the documentation available for the selected scheduler job. Refer to Reading the Documentation for details. - **Create** the documentation for the selected scheduler job. Refer to Creating

Menu Options	Description
	Documentation for details.
Projects	Select this option to perform one of the following: • **Remove from Project** - Select to remove the Scheduler job from a project. Refer to Removing Objects from Projects for details. • **Add to Project** - Select to add the Scheduler job to a project. Refer to Adding Objects to Projects for details. • **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to List Project Memberships for Objects for details. **Note:** Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane.
Check Out	Select this option to check out the object for editing to prevent any other users from being able to modify, update or delete any of their associated objects while you are making changes. Refer to Object Check-Outs and Check-Ins for details.

Parameters

Parameters are a means of passing information between two or more procedures and between the WhereScape RED environment and procedures. Parameters can be added or removed from a Project.

Following is an example of the menu displayed when you right-click a parameter object.



Note:

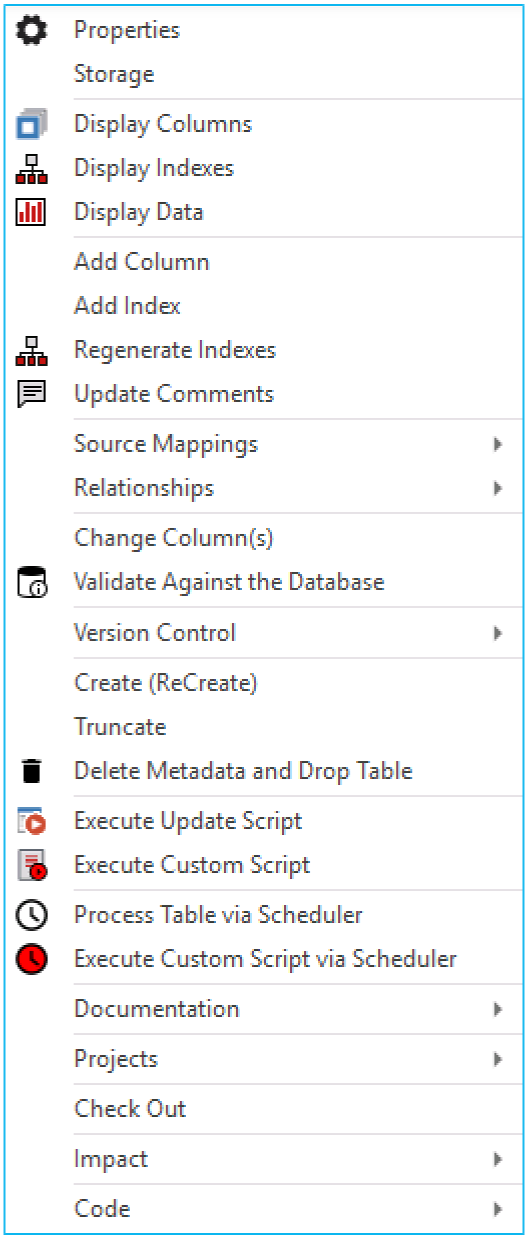
WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **C** creates the index, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

Menu Options	Description
Add Parameter	Select this option to launch the **Parameter Maintenance** window which enables you to define a parameter and its value.
Copy Parameter	Select this option to launch the **Parameter Maintenance** window which enables you to copy the selected parameter which you can use in other objects.
Modify Parameter	Select this option to launch the **Parameter Maintenance** window which enables you to modify the selected parameter and its value.

Menu Options	Description
Used by	Select this option to list the objects that are using the selected parameter in the **Results** pane.
Delete Parameter	Select this option to delete the selected parameter.
Projects	Select this option to perform one of the following: • **Remove from Project** - Select to remove the parameter from a project. Refer to Removing Objects from Projects for details. • **Add to Project** - Select to add the parameter to a project. Refer to Adding Objects to Projects for details. • **List Projects** - Displays a list of projects which contain the current object, results are shown in the bottom pane. Refer to List Project Memberships for Objects for details. **Note:** Multiple objects can be selected by double-clicking the Object Group icon in the left pane and then Ctrl + clicking multiple objects in the Drop Target (middle) pane.

EDW 3NF Tables

Following is an example of a standard menu for EDW 3NF Tables.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **D** displays data, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

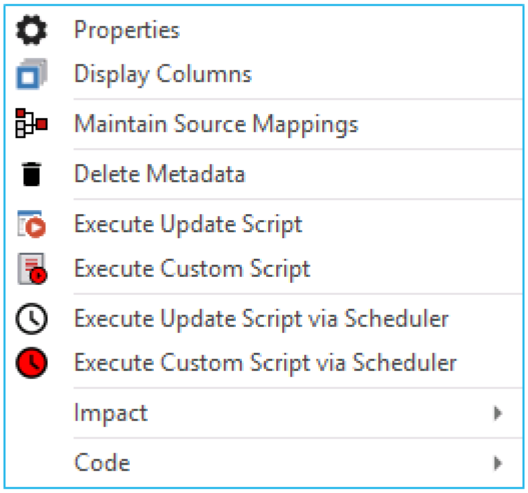
Tip:

The menu options available for Data Store tables is a subset of the options for **Data Store Tables**.

Source Mapping

Source Mapping objects are used in WhereScape RED to define source column mappings, when there is a requirement for the target table to be populated by more than one set of source tables. These objects support the generation of an update procedure for the target table, sourced from each set of the source tables. Refer to Multi Source Processing for details.

Following is an example of the menu displayed when you right-click a source mapping object.



Note:

WhereScape RED supports the use of keyboard shortcuts—the underlined letter of a menu option. For example, pressing **P** opens the **Properties** window of the selected table, pressing **C** display columns, etc.
Ensure that the **Windows > Control Panel > Ease of Access** setting associated with keyboard shortcuts is enabled, to display keyboard shortcuts in RED. Refer to the relevant MS Windows documentation for details.

If a procedure has been locked as a result of the WhereScape RED utility being killed or failing or a database failure then it can be unlocked via the Properties screen associated with the procedure.

Menu Options	Description
Maintain Source Mappings	Select this option to launch the Source Mapping tool which enables you to manage the column mappings of the selected target table.
Execute Update Procedure via Scheduler	Select this option to set up a WhereScape RED scheduler job to execute the source mapping object's update procedure.
Execute Custom Procedure via Scheduler	Select this option to execute the procedure defined as the 'custom procedure' for the table via the WhereScape RED scheduler.
Impact	Select this option to perform one of the following: • **Track Back Report** – Lists the source tables of the selected source mapping object. • **Track Forward Report** – Lists the parent table object of the selected source mapping object. • **Dependent Jobs Report** – Lists jobs that includes the currently selected source mapping object or the parent table object.
Code	Select this option to view a procedure attached to a table or to rebuild or regenerate the table's update procedure.

Tip:

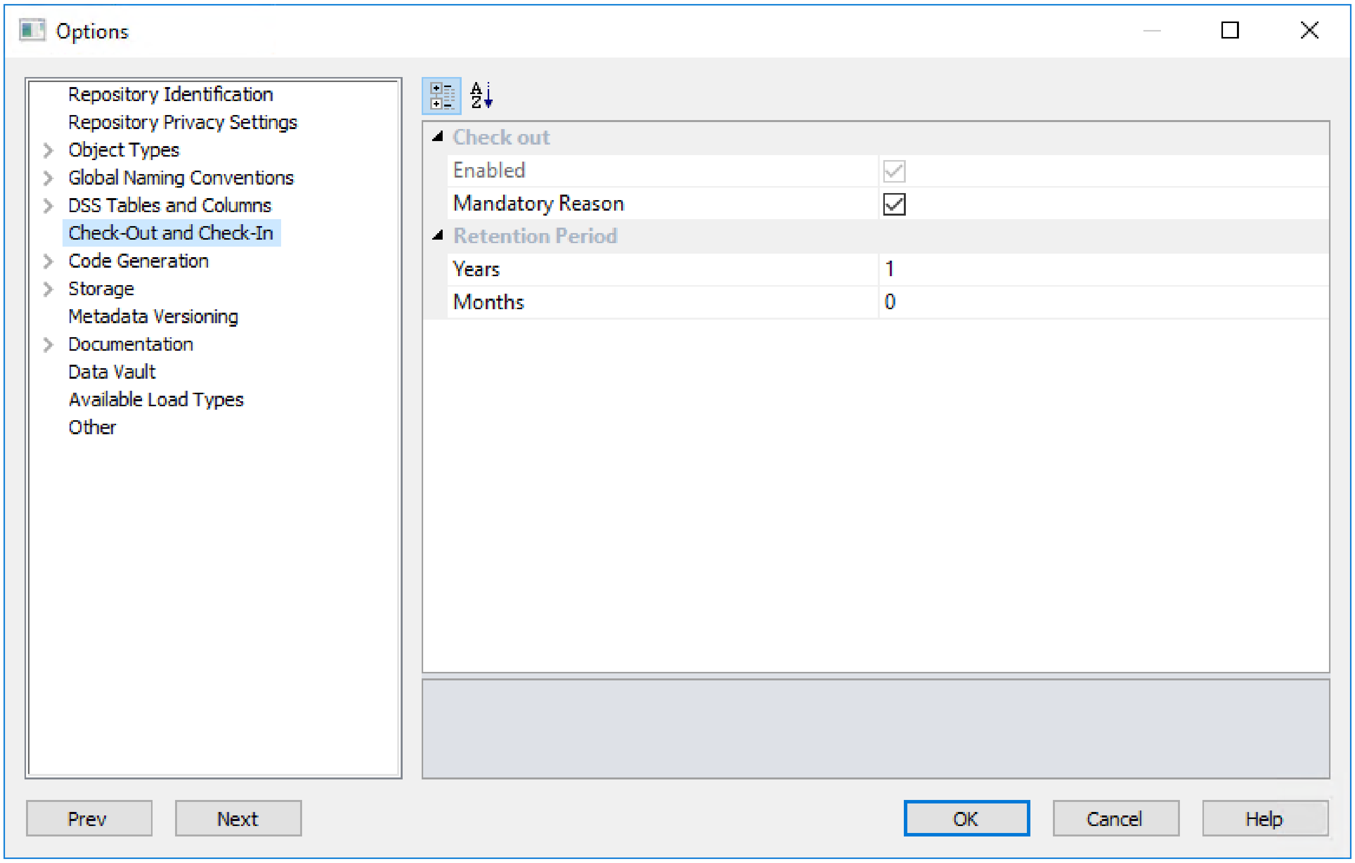
All the other options for source mappings are described under the other object types.

Object Check-Outs and Check-Ins

Objects in WhereScape RED can be checked out for editing to prevent other users from being able to modify, update or delete any of their associated objects, while you are making changes to them.

The Check-Outs and Check-Ins functionality is configured from **Tools > Options > Check-Out and Check-In** window.

The Check-Out option is enabled by default, as shown below:



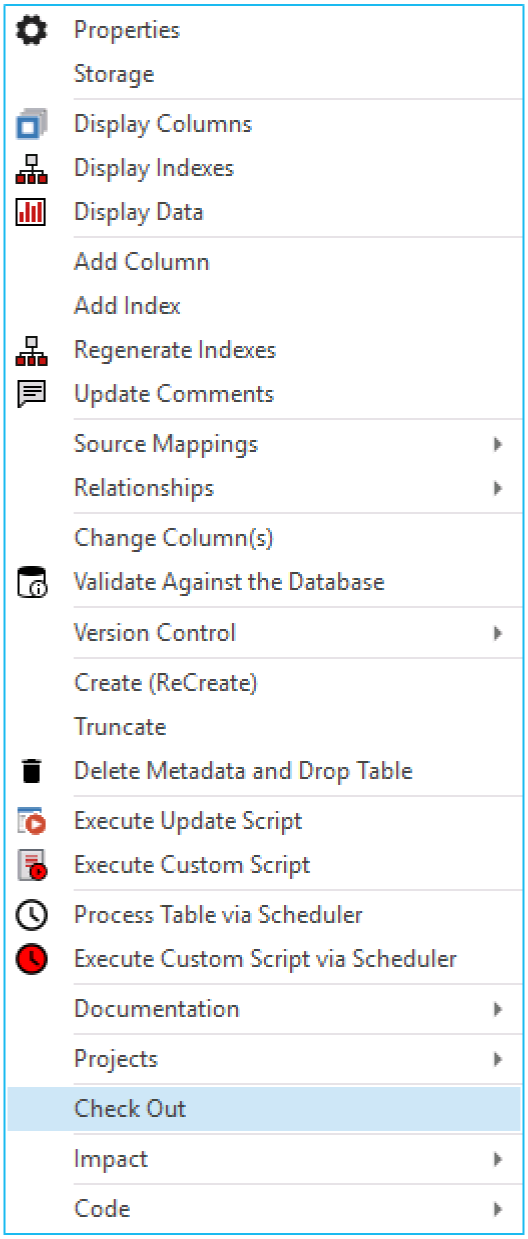
Check out

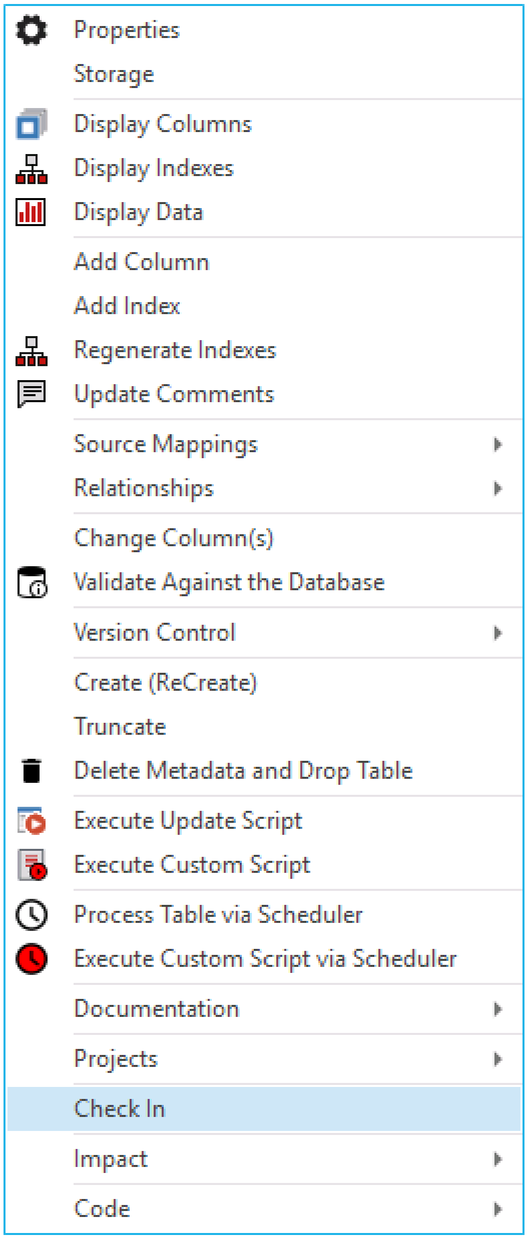
Enabled	☑
Mandatory Reason	☑

Retention Period

Years	1
Months	0

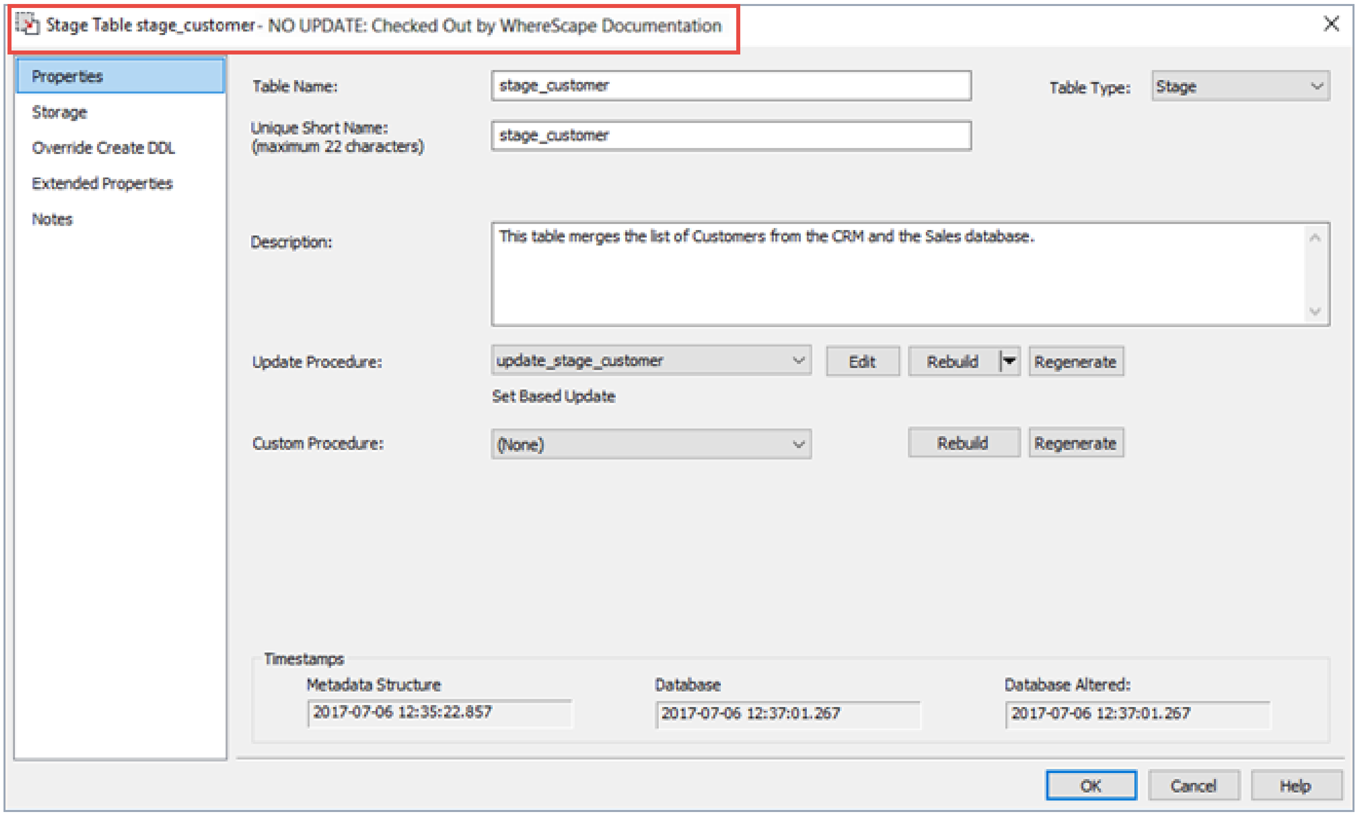
The following screens displays the setting to check-out/check-in objects in WhereScape RED through the object's right-click context menus.





To check-in an object, simply right-click it and select **Check In** from the context menu.

Checked out objects and their associated procedures and scripts cannot be modified, updated or deleted by other users. If a different user tries to access the **Properties** or **Procedure** screens for those objects, the fields will not be available and headed NO UPDATE: Checked Out by (user name).



Stage Table stage_customer- NO UPDATE: Checked Out by WhereScape Documentation

Table Name:	stage_customer
Table Type:	Stage
Unique Short Name: (maximum 22 characters)	stage_customer
Description:	This table merges the list of Customers from the CRM and the Sales database.
Update Procedure:	update_stage_customer Set Based Update
Custom Procedure:	(None)

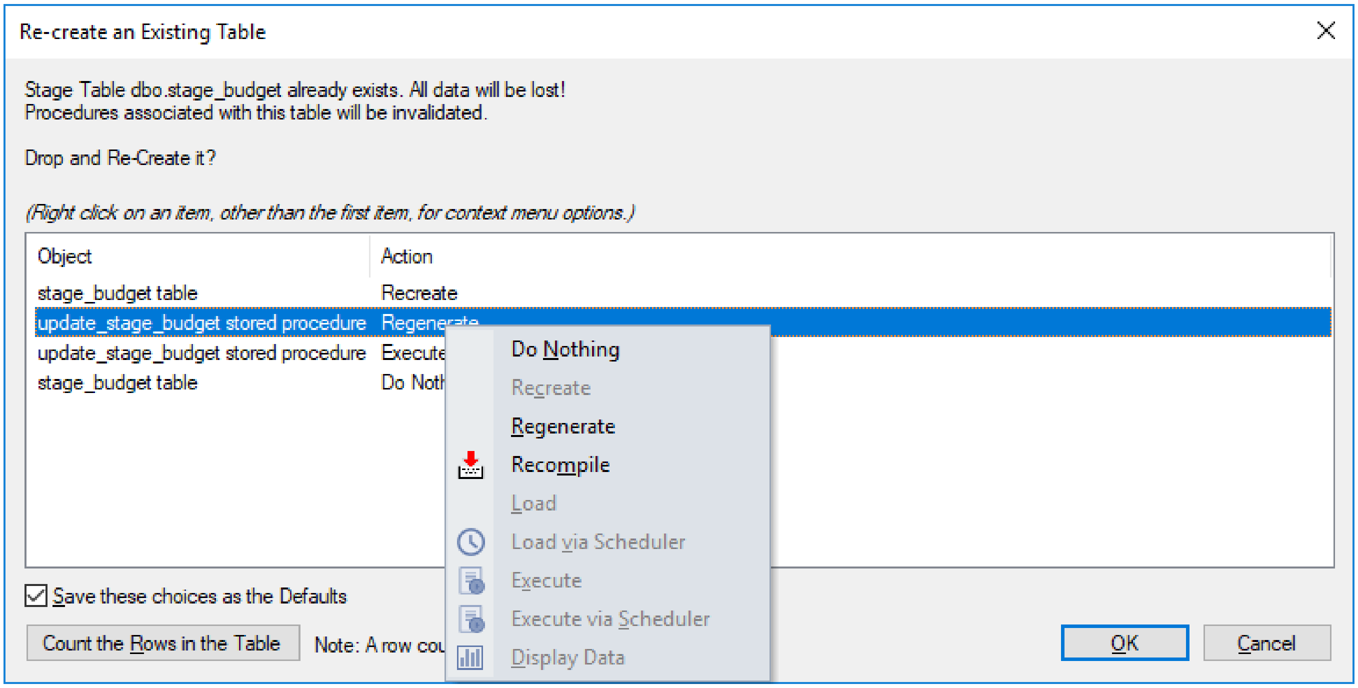
Timestamps

Metadata Structure	Database	Database Altered:
2017-07-06 12:35:22.857	2017-07-06 12:37:01.267	2017-07-06 12:37:01.267

Other users logging into the same metadata repository will have some of the checked-out object's context menu options unavailable.

Re-create an Existing Table

The Re-create window, as shown in the following screen provides available options for dropping and recreating an existing table and associated procedures. Objects and actions available depend on the table being recreated.



Re-create an Existing Table

Stage Table dbo.stage_budget already exists. All data will be lost!
Procedures associated with this table will be invalidated.

Drop and Re-Create it?

(Right click on an item, other than the first item, for context menu options.)

Object	Action
stage_budget table	Recreate
update_stage_budget stored procedure	Regenerate
update_stage_budget stored procedure	Execute
stage_budget table	Do Noth

☑ Save these choices as the Defaults

Note: A row cou

Actions in the table are completed from top to bottom; the action performed on each object can be set as desired:

Object(s)	Action	Description
Load Tables	Do Nothing (Default)	Do nothing for this object, skip to the next action.
	Load	Perform an interactive load of the data into the table. WhereScape recommends performing this via the Scheduler for large tables.
	Load via Scheduler	Add a data load task to the scheduler. Useful for large tables where processing may take some time.
Load Script	Regenerate	Regenerate the table's load script. This option is only available for Load tables with a load type of script and with the source connection setting defined. If the Load table does not have an existing script then a new script is created.
Update procedure	Do Nothing (Default)	Do nothing for this object, skip to the next action.
	Execute	Execute the update procedure. WhereScape recommends performing this via the Scheduler for large tables.
	Execute via Scheduler	Execute the update procedure via the scheduler. Useful for large tables where processing may take some time.
Tables	Do Nothing	Do not display the data.
	Display Data	Display the table data after recreating the table and update procedure (if applicable). • This is the default setting for Views and Dimension Views. • This is the default setting for tables when an update procedure is present and has been executed in the previous action. • Unavailable when an update procedure is present but has not been executed in the Re-create window.

Save these choices as the Defaults

This option can be used to store the current actions as the default choices for all object types. The Re-create window is then automatically populated with the saved actions each time it appears.

Count the Rows in the Table

Displays the total number of rows in the table at the bottom of the Re-create window. Knowing the number of rows in a table can help estimate how long Load and Execute actions may take.

Organizing Objects

As described in the previous section, there are many types of object in WhereScape RED .

The objects in the metadata repository are displayed in the left pane of the Builder window. They are displayed in a tree structure which can be expanded and closed as required.

The tree can be refreshed by using **F5** or the **Ctrl+R** key.

Object Groups

The objects created in WhereScape RED are grouped together into **object groups** based on object type.

For example, we store all the dimension objects in the **Dimension** object group. Optionally, we can choose to display dimension tables as the Dimension object group and dimension views as the Dimension View object group.

Projects

These object groups are in turn stored within **Projects**. When WhereScape RED is first started, the special project called **All Objects** is the only project. This project will always contain all the objects that exist in the metadata repository.

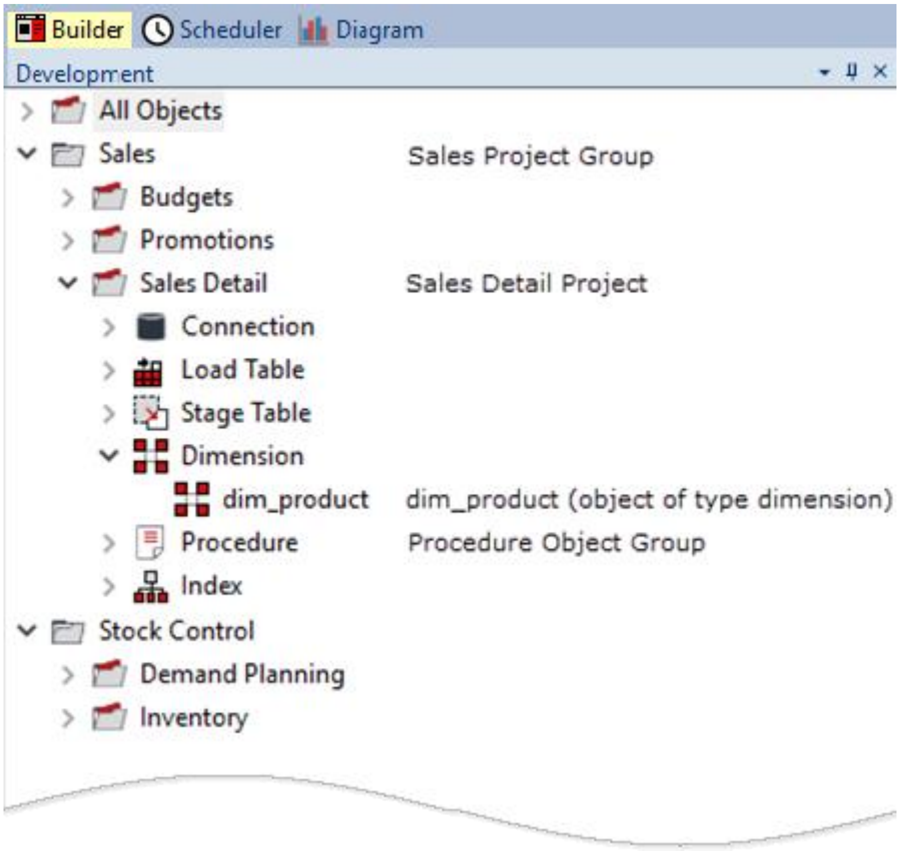
Additional projects can be created if required. These additional projects can hold some or all of the objects, as seen in **All Objects**. An object as such, only exists once in the metadata. Therefore, if we have a Dimension object called dim_product, there is and can only be one copy of the object dim_product within the metadata.

Projects are used to hold a group of objects that relate back to a similar module or analysis area of the data warehouse.

In the example below, we have additional projects called Budgets, Promotions, Sales Detail, Demand Planning and Inventory. In this way, projects enable us to restrict the amount of information (objects) we need to deal with to just those relevant to the area being worked on.

Projects can also be used to group objects for an upcoming code promote.

The example below shows a metadata repository using two project groups (Sales and Stock Control), and five additional projects (Budgets, Promotions, Sales Detail, Demand Planning and Inventory).



Notes:

If you delete an object using the right-click context menu, the object is deleted from the metadata and removed from all Projects. You can remove the object from the project instead of deleting it. Use the object's right-click context menu to see the Projects to which the object is currently included. Refer to **List Project Memberships for Objects** for details.

It is important to understand that these projects are only a means of visualizing the objects. Even though an object may appear in many projects, it only exists once in the metadata.

To create a project, right-click a Project or Group listed in the left pane and select **New Project** from the context menu.

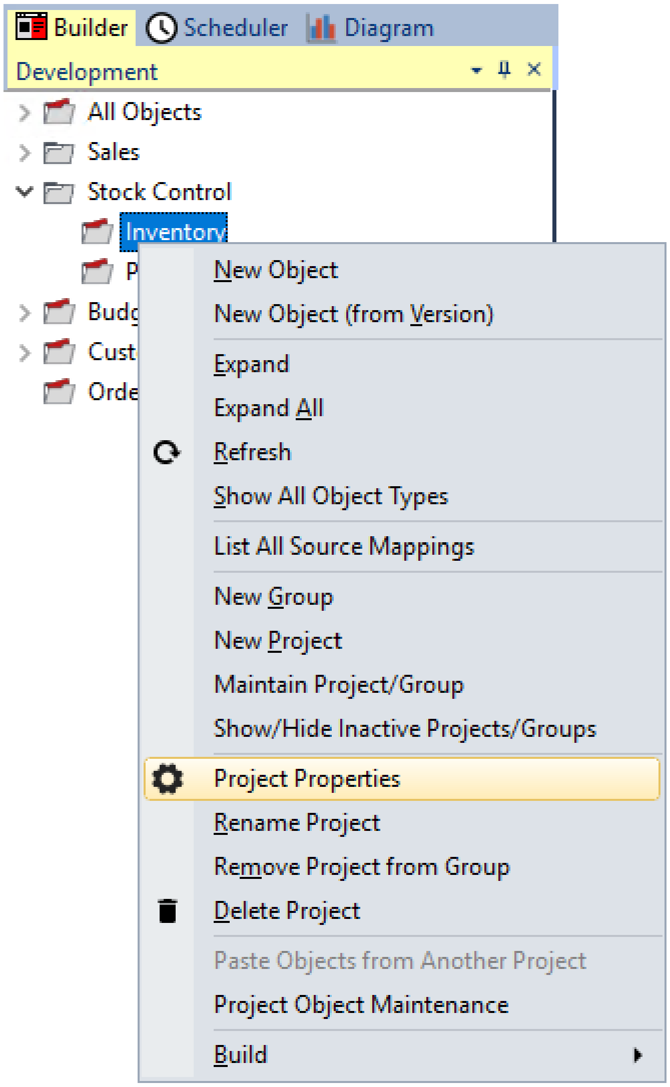
The **File > New Project** menu option can also be used. Projects can be renamed, removed from the project Group or deleted, using the right-click context menu of a selected project.

Tip:

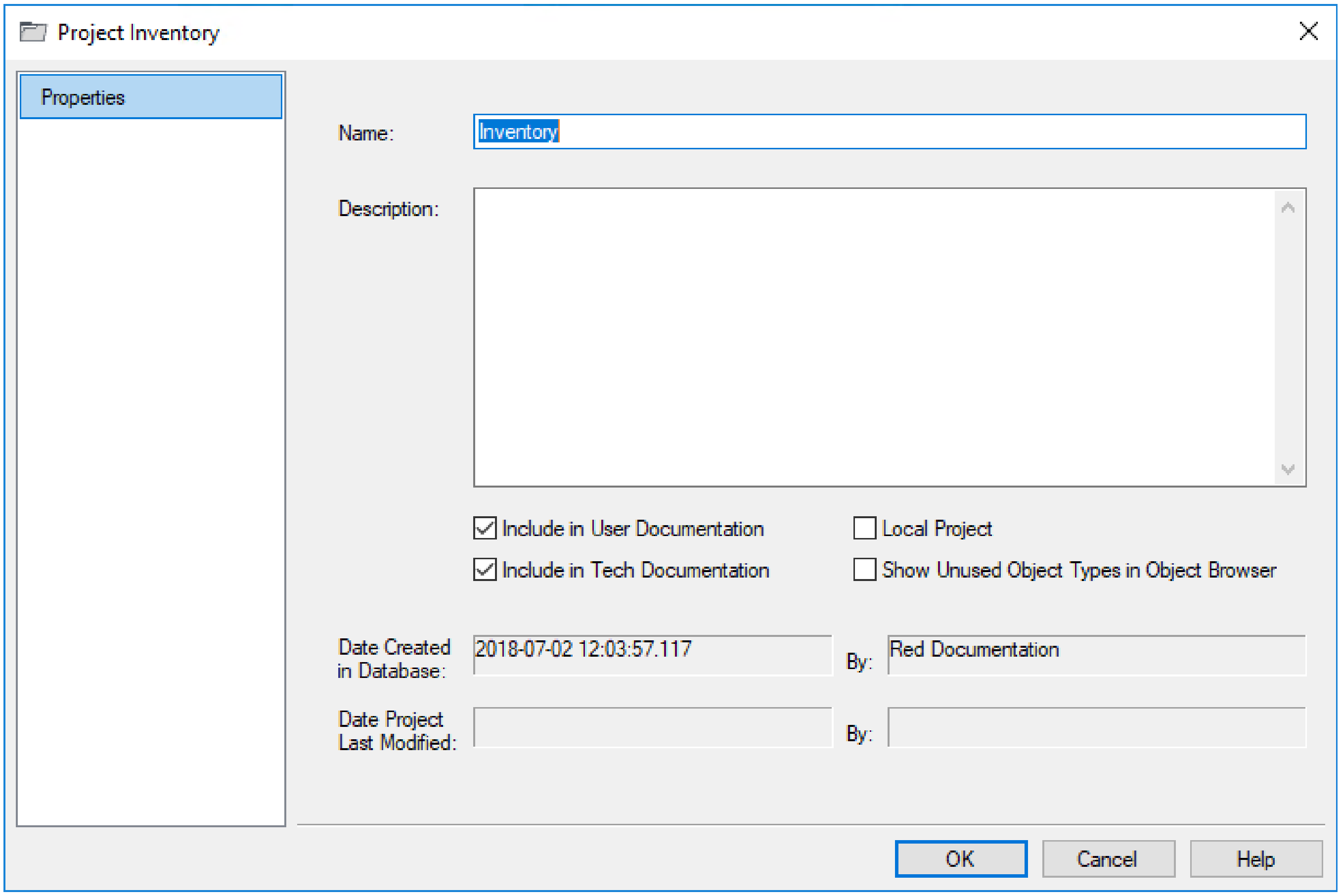
You can automatically create a Project from a Diagram generated from RED—refer to **Creating a Project from a Diagram** for details.

Deleting a Project does not delete the objects associated with it from the metadata; it simply removes their reference in the Project being deleted.

To view or make changes to the project's Properties, right-click the Project name from the left pane and then select **Project Properties**.



The Project Properties window is displayed:



Name: Inventory

Description:

☑ Include in User Documentation
☐ Local Project
☑ Include in Tech Documentation
☐ Show Unused Object Types in Object Browser

Date Created in Database: 2018-07-02 12:03:57.117 By: Red Documentation

Date Project Last Modified: By:

Option	Description
Include in User Documentation	Select to include the project in the User Documentation. This is selected by default.
Include in Tech Documentation	Select to include in the Technical Documentation. This is selected by default.
Local Project	Select to exclude the local project in the Application files.
Show Unused Object Types in Object Browser	Select to display all object types in the Object Browser Pane. If not selected, only object types currently in use are displayed. Refreshing the Object Browser Pane is required after changing this setting.

Project Groups

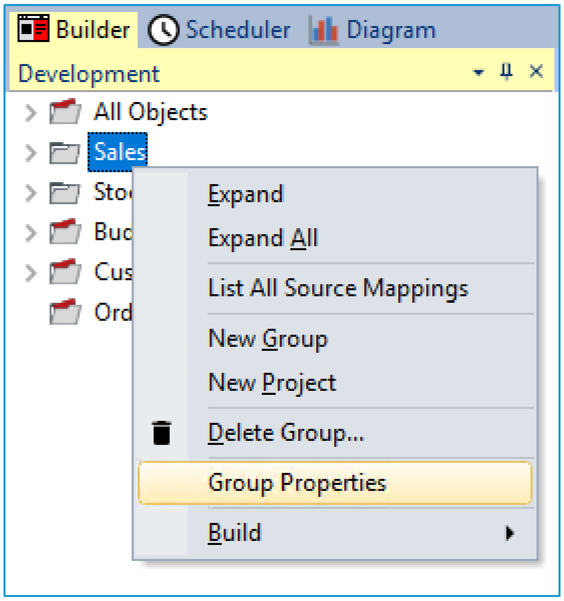
Projects can in turn be organized into **Groups**.

To create a Group, right-click a Project or Group listed in the left pane and select **New Group** from the context menu.

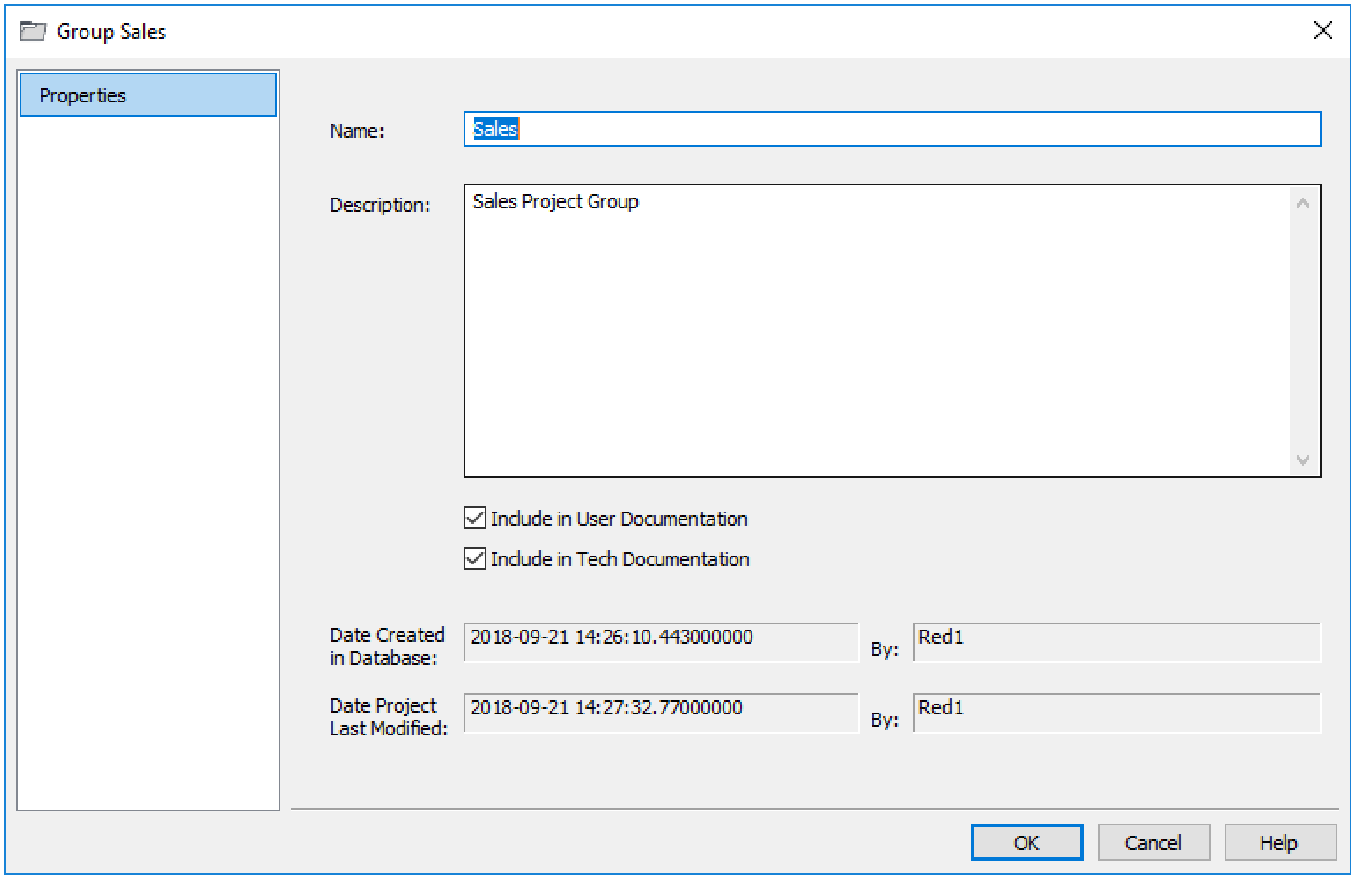
The **File > New Group** menu option can also be used. A project Group can be deleted using the right-click context menu of a selected group.

Deleting a Group does not delete its associated projects or objects from the metadata; it simply removes their reference in the Group being deleted. The projects of a deleted Group remain in the left pane.

To view or make changes to the group's Properties, right-click the Group name from the left pane and select **Group Properties**.



The Group Properties window is displayed:



Name: Sales

Description: Sales Project Group

☑ Include in User Documentation

☑ Include in Tech Documentation

Date Created in Database: 2018-09-21 14:26:10.443000000 By: Red1

Date Project Last Modified: 2018-09-21 14:27:32.77000000 By: Red1

Adding Objects to Projects

There are several ways to add objects to a project:

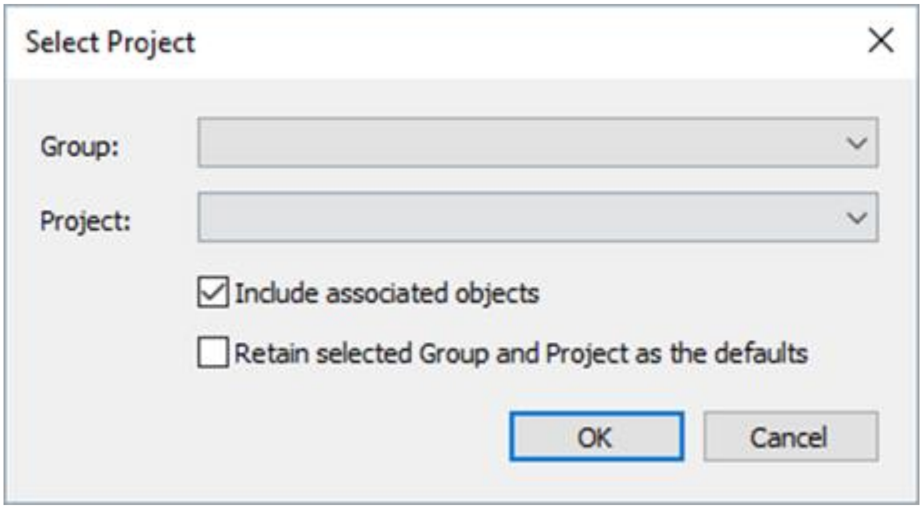
- Click an object in the left pane, press **Ctrl+C**, click the target project (either the project folder or any object group or object within it) and press **Ctrl+V**.

- **Drag** the object to the required project.
- Right-click an object in the left pane and select **Projects > Add to Project**.
- Highlight a number of objects in the middle pane, right-click and select **Projects > Add to Project**.
- **Use the Project Object Maintenance Facility** (refer to Using Project Object Maintenance for details).

Adding an Object to an Independent Project (a project that is not in a group)

To add an object to an independent project:

1. Right-click an object from the **Objects** list pane and select **Projects > Add to Project**.
The **Select Project** window is displayed:

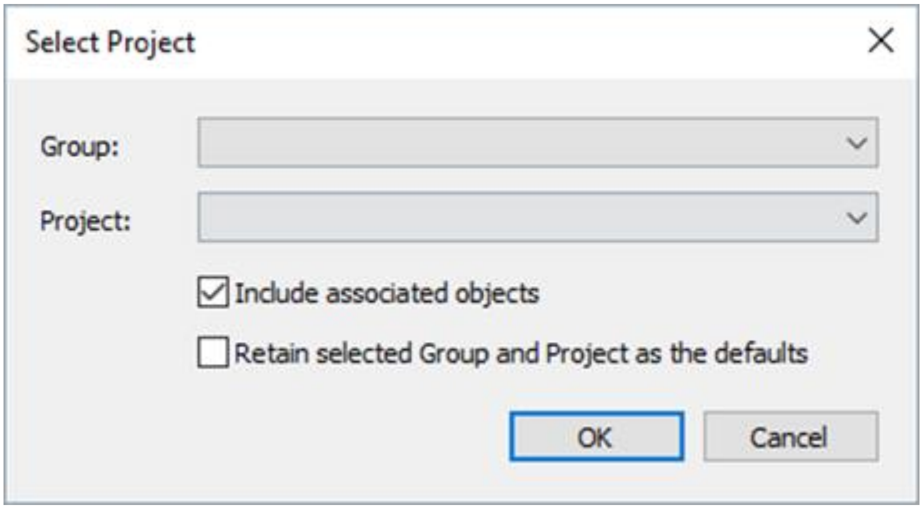

Select Project
Group:
Project:
☑ Include associated objects
□ Retain selected Group and Project as the defaults
OK
Cancel

2. Select a project from the **Project** drop-down list and click **OK**.

Adding an Object to a Project in a Group

To add an object to a project that belongs to a group:

1. Right-click on an object from the **Objects** list pane and select **Projects > Add to Project**.
The **Select Project** window is displayed:


Select Project
Group:
Project:
☑ Include associated objects
□ Retain selected Group and Project as the defaults
OK
Cancel

2. Select a group from the **Group** drop-down list.
3. Select a project from the **Project** drop-down list and click **OK**.

Notes:

- Selecting the **Include associated objects** check box, also adds any indexes, procedures and scripts to the

Notes:

selected project.

- Selecting the **Retain selected Group and Project as the defaults** check box enables you to retain the last values defined in the Group and Project fields and use them by default when an object is added to a Project or Group.

Removing Objects from Projects

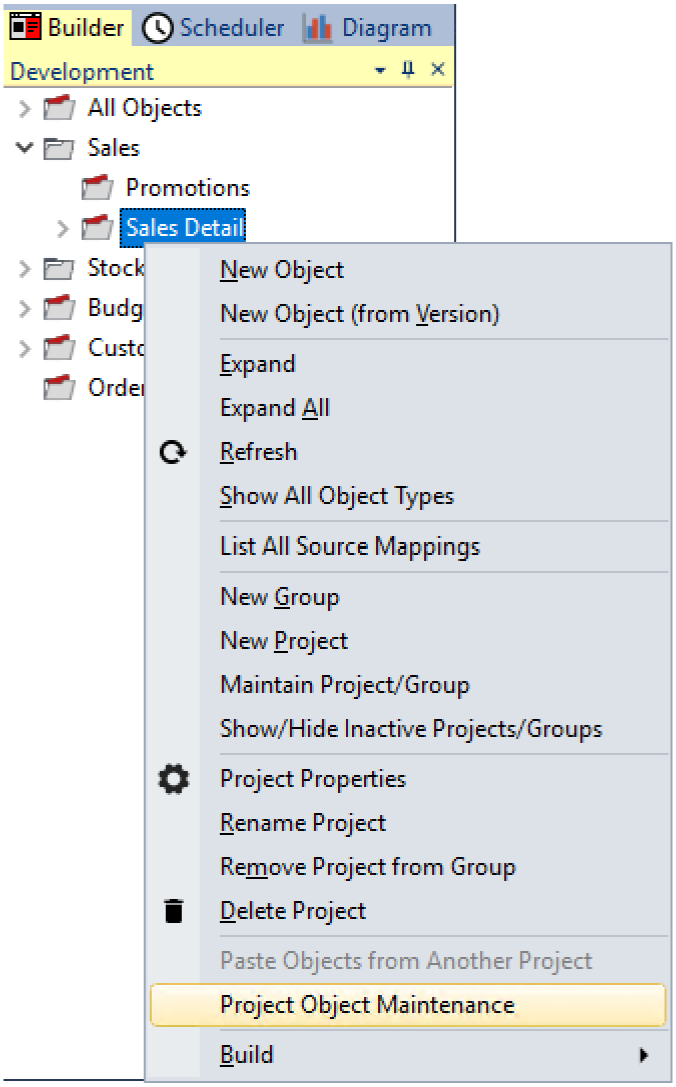
There are several ways to remove objects from a project:

- **Drag** the object to the blank area at the bottom of the pane.
- **Drag** the object into the middle pane.
- **Drag** the object to the **All Objects** project.
- Right-click the object in the left pane and select **Projects > Remove from Project.**
- Highlight several objects in the middle pane, right-click and select **Projects > Remove from Project**.
- **Use the Project Object Maintenance Facility** (Refer to Using Project Object Maintenance for details).

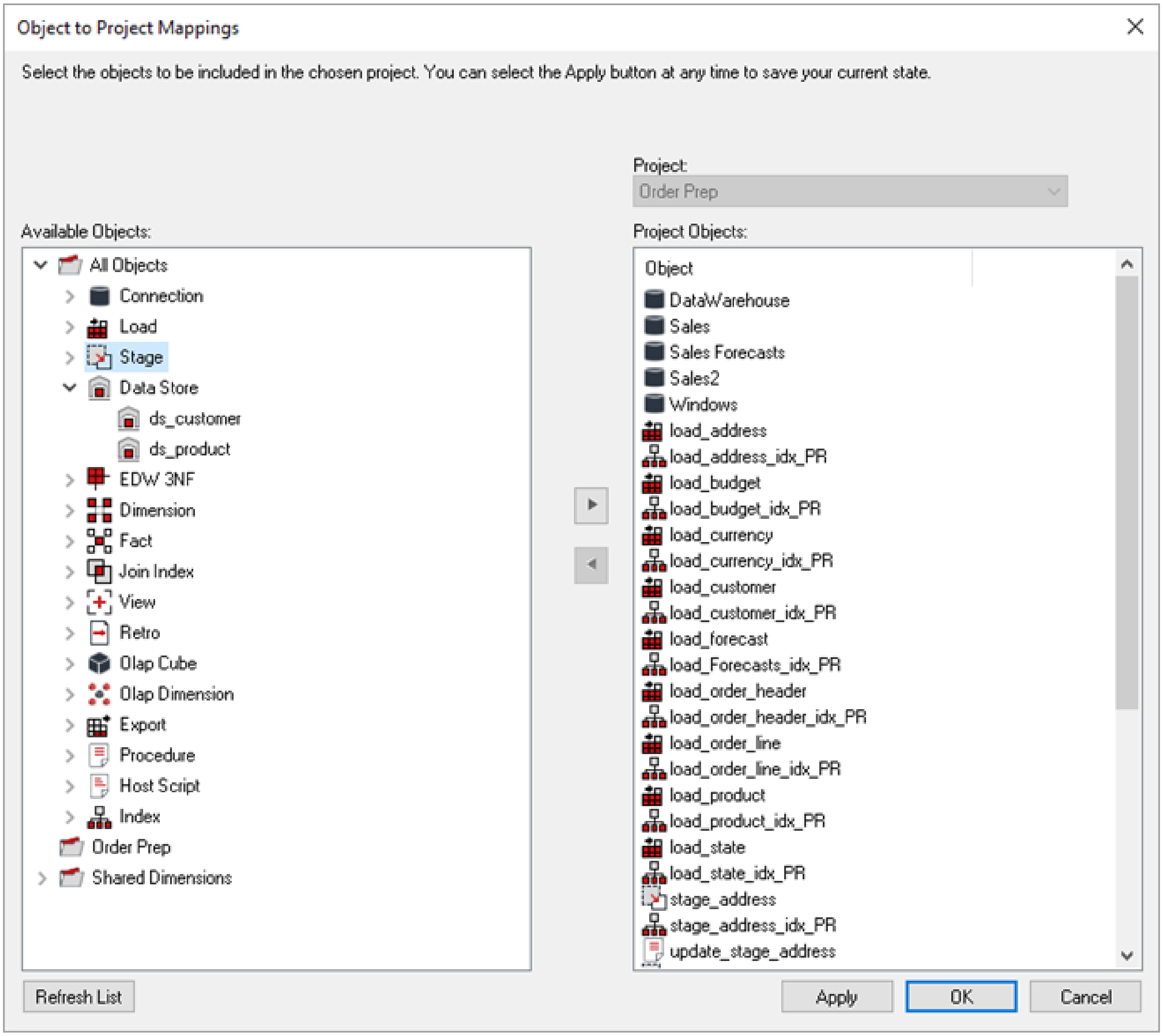
Using Project Object Maintenance

The Project Object Maintenance facility enables you to manage the objects within the different projects defined in your metadata repository.

The **Object to Project Mappings** window is launched by right-clicking a project and selecting **Project Object Maintenance** from the context menu.



The following window is displayed:



Select the objects to be included in the chosen project. You can select the Apply button at any time to save your current state.

Project: Order Prep

Available Objects:

- All Objects
  - Connection
  - Load
  - Stage
  - Data Store
    - ds_customer
    - ds_product
  - EDW 3NF
  - Dimension
  - Fact
  - Join Index
  - View
  - Retro
  - Olap Cube
  - Olap Dimension
  - Export
  - Procedure
  - Host Script
  - Index
- Order Prep
- Shared Dimensions

Project Objects:

Object
DataWarehouse
Sales
Sales Forecasts
Sales2
Windows
load_address
load_address_idx_PR
load_budget
load_budget_idx_PR
load_currency
load_currency_idx_PR
load_customer
load_customer_idx_PR
load_forecast
load_Forecasts_idx_PR
load_order_header
load_order_header_idx_PR
load_order_line
load_order_line_idx_PR
load_product
load_product_idx_PR
load_state
load_state_idx_PR
stage_address
stage_address_idx_PR
update_stage_address

Adding an Object

1. Select and object in the **Available Objects** pane and click ▶.
   The selected object is moved to the **Project Objects** pane.

   Note:

   By default, associated objects (procedures, scripts and indexes) are also moved to the right. These can be manually removed if not required.

2. Click **OK**.

Removing an Object

1. Select and object in the **Project Objects** pane and click ◀.
   The selected object is moved to the **Available Objects** pane.

   Note:

Note:

By default, associated objects (procedures, scripts and indexes) are also removed.

2. Click **OK**.

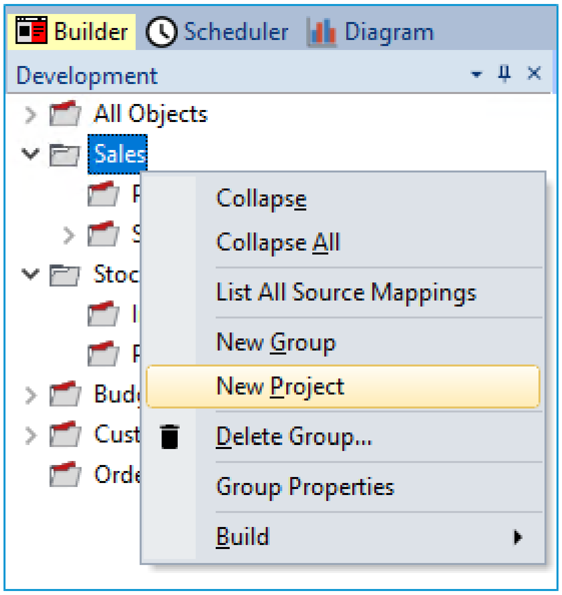
Tips:

- Clicking **Apply** updates object/project relationships without having to exit the Project Object Maintenance facility.
- Clicking **Refresh List** refreshes the object tree in the left pane.

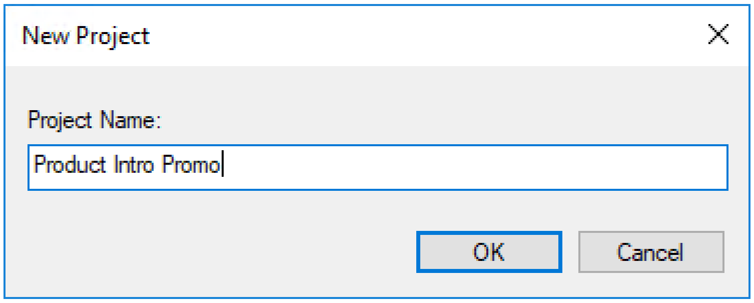
Adding Projects to Groups

You can add projects to a group by creating the project under the group:

1. Right click the target Group and then select **New Project** from the context menu:

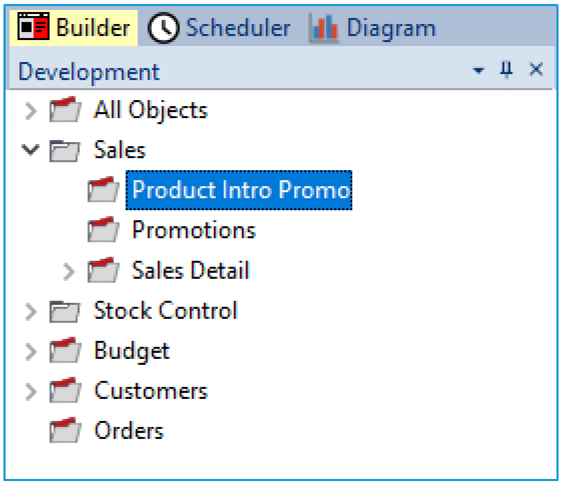


2. Enter a name in the **New Project** window and then click **OK**.



Project Name: Product Intro Promo

The new project is added under the selected Group:



Tip:

You can also move an existing project from another group into this group. Refer to **Moving Projects within Groups** for details.

Removing Projects from Groups

A project can be removed from a group by:

- **Dragging** the project to the blank area at the bottom of the pane.
- **Dragging** the project into the middle pane.
- **Dragging** the project to the **All Objects** project.
- **Removing** the project using right-click **Remove Project from Group**.
- **Deleting** the project using right-click **Delete Project**.

Note:

The first four methods move the project out from the group to become an independent project. The last option removes the project from the metadata repository.

Moving Projects within Groups

Note:

An **object** can be in any number of projects and a **project** can also be included in more than one Project Group.

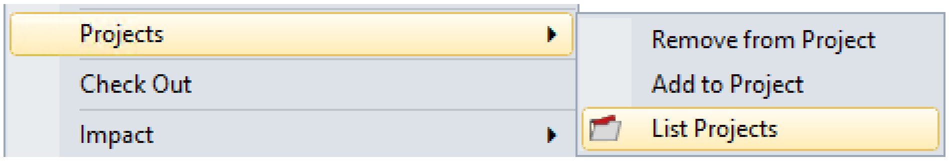
Performing a drag and drop from one group to another group simply creates an additional **Project > Group Mapping**.

To move a project from one group to another group:

1. **Copy** the project by dragging the project from one group to the other group.
2. **Remove** the project from its original group by right-clicking and selecting **Remove Project from Group**.

Listing Project Memberships for Objects

To list projects that contain a specific object, right click the object in the left pane and select **Projects > List Projects**. Results are shown in the bottom pane.



Multiple objects of the same type can be selected by double-clicking the **Object Type** icon in the left pane (e.g. **Connection**, **Load Table**, etc.) and then **Ctrl +** clicking multiple objects displayed in the middle pane.

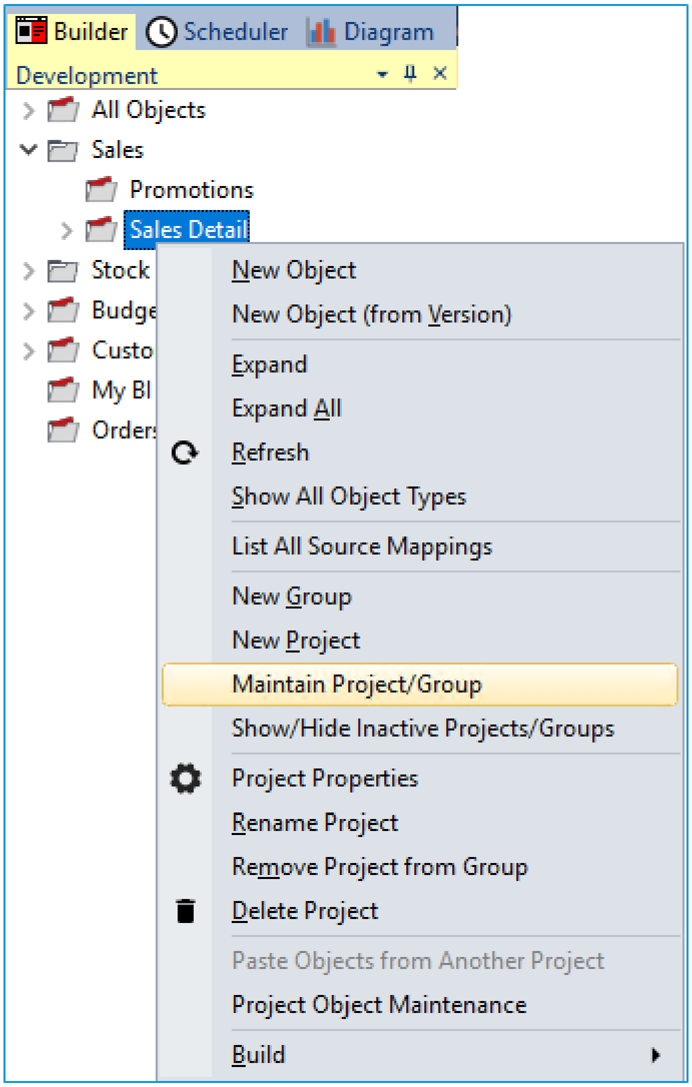
Tip:

You can double click a Project or Project Group listed in the left pane to see the Project membership details of each object in the selected Project or Project Group in your repository.

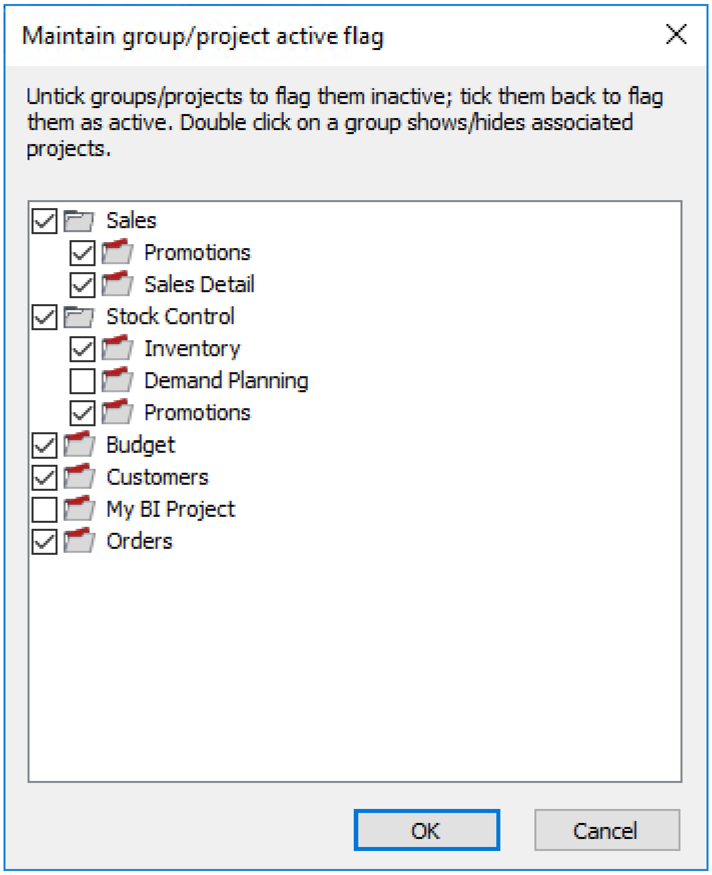
Maintaining Project/Group Active Flag

The Maintain Project/Group facility enables you to flag the active status of Projects and Project Groups defined in your metadata repository. This facility enables you to hide Inactive projects that are no longer required, as an alternative to deleting them to preserve information about the contents of old projects.

The **Maintain group/project active flag** window is launched by right-clicking a project from the left pane and selecting **Maintain Project/Group** from the context menu.



The following window is displayed:



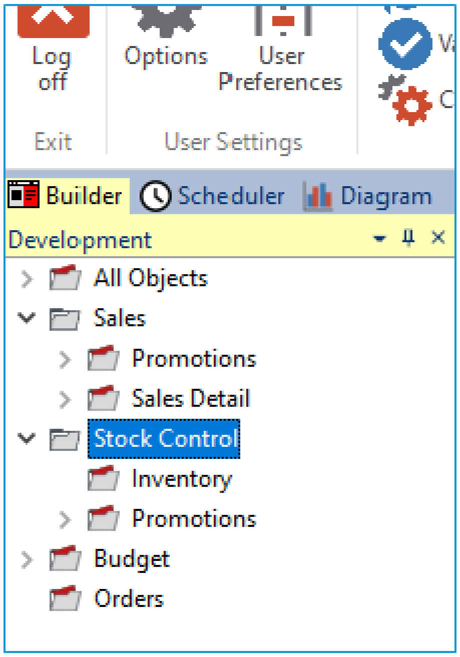
Maintain group/project active flag

Untick groups/projects to flag them inactive; tick them back to flag them as active. Double click on a group shows/hides associated projects.

- ☑ Sales
  - ☑ Promotions
  - ☑ Sales Detail
- ☑ Stock Control
  - ☑ Inventory
  - ☐ Demand Planning
  - ☑ Promotions
- ☑ Budget
- ☑ Customers
- ☐ My BI Project
- ☑ Orders

In the above example, there are two projects that are inactive, the **Demand Planning** project which is grouped under the **Stock Control** project group and the **My BI Project** which is an independent project, e.g. does not belong to a project group.

Because these two projects are flagged as **inactive**, they are not displayed in the left pane of the **Builder** window.



- All Objects
- Sales
  - Promotions
  - Sales Detail
- Stock Control
  - Inventory
  - Promotions
- Budget
- Orders

Notes:

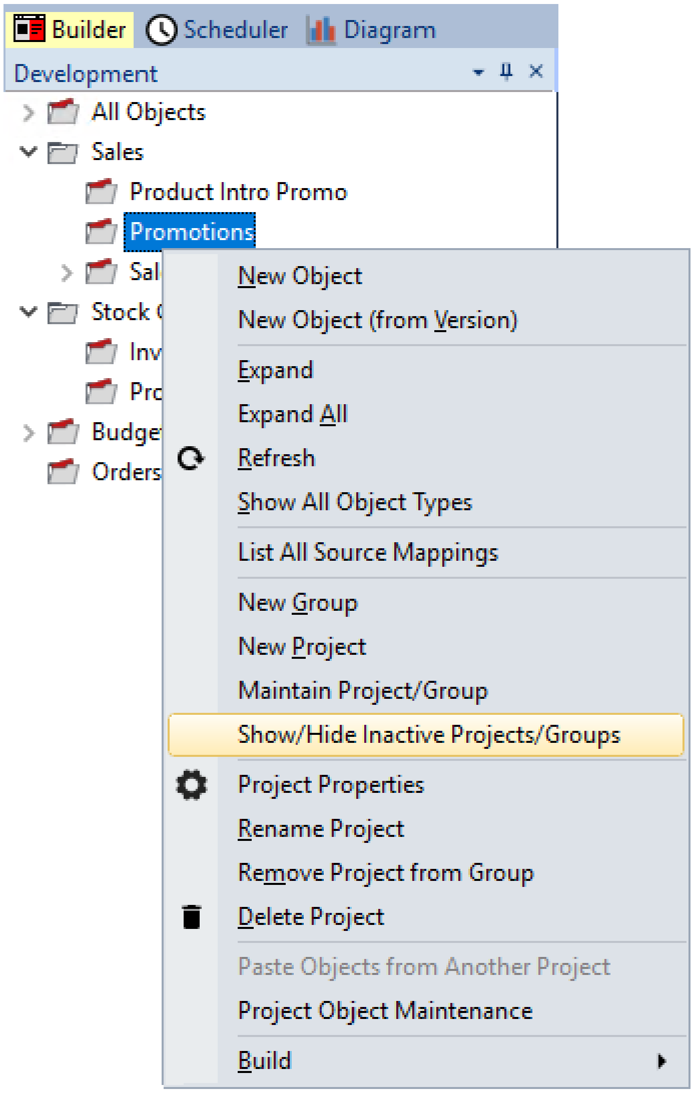
If a Project that belongs to several Groups is flagged as inactive, it becomes inactive on all the Groups it exists.
Projects/Groups that are flagged as inactive are also not displayed/available in the following:

- Generated documentation and diagrams, refer to **Documentation and Diagrams** for more information.
- **Build Deployment Application > Objects to Add/Replace**, refer to **Application Creation** for more information.

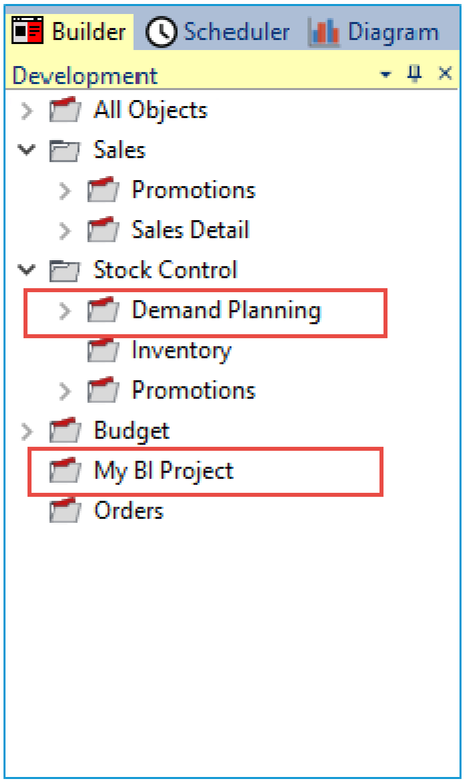
Displaying/Hiding Inactive Projects/Groups

The **Show/Hide Inactive Projects/Groups** function enables you to control the display of the projects or project groups that have been flagged as inactive, using the **Maintain group/object active flag** window.

To display the inactive projects/groups in the left pane of the **Builder** window, right-click a project and then select **Show/Hide Inactive Projects/Groups** from the context menu:



The projects flagged inactive and previously hidden are displayed in the left pane of the **Builder** window:



In the same manner, use **Show/Hide Inactive Projects/Groups** to hide the inactive projects/groups that are displayed in the left pane of the **Builder** window.

Building Scheduler Jobs

Scheduler jobs can be created from Object Groups, Projects and Project Groups defined in RED via the **Build > Build Job** context menu. The detailed steps are described in the following sections.

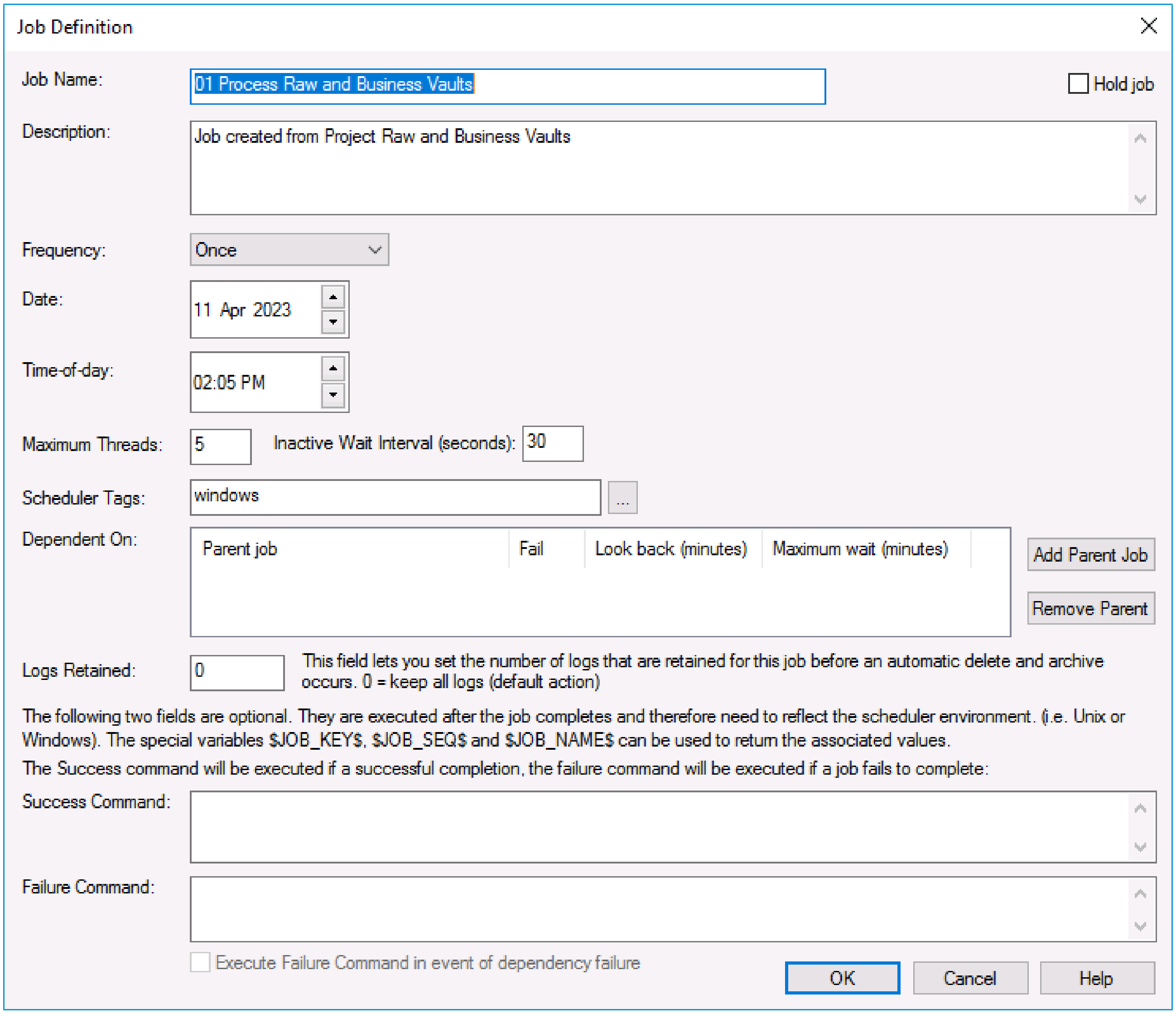
- Building Scheduler Jobs from Projects or Groups
- Building Scheduler Jobs from Object Groups

Building Scheduler Jobs from Projects or Groups

WhereScape RED enables you to automatically create a Scheduler job from the table objects in a Project or Project Group defined in RED. This feature utilizes the track-back information for all the objects included in a Project or Project Group to determine their task dependencies and then creates the Scheduler job.

The steps are outlined below:

1. Right click the **Project** or **Group** from the left pane of the **Builder** window.
2. Select the **Build > Build Job** option from the context menu.
3. Make any changes to the details and settings in the **Job Definition** window as required and then click **OK** to define the tasks in the Scheduler job.



Job Definition

Job Name:	01 Process Raw and Business Vaults
Description:	Job created from Project Raw and Business Vaults
Frequency:	Once
Date:	11 Apr 2023
Time-of-day:	02:05 PM
Maximum Threads:	5
Inactive Wait Interval (seconds):	30
Scheduler Tags:	windows
Logs Retained:	0

□ Hold job

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)

This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

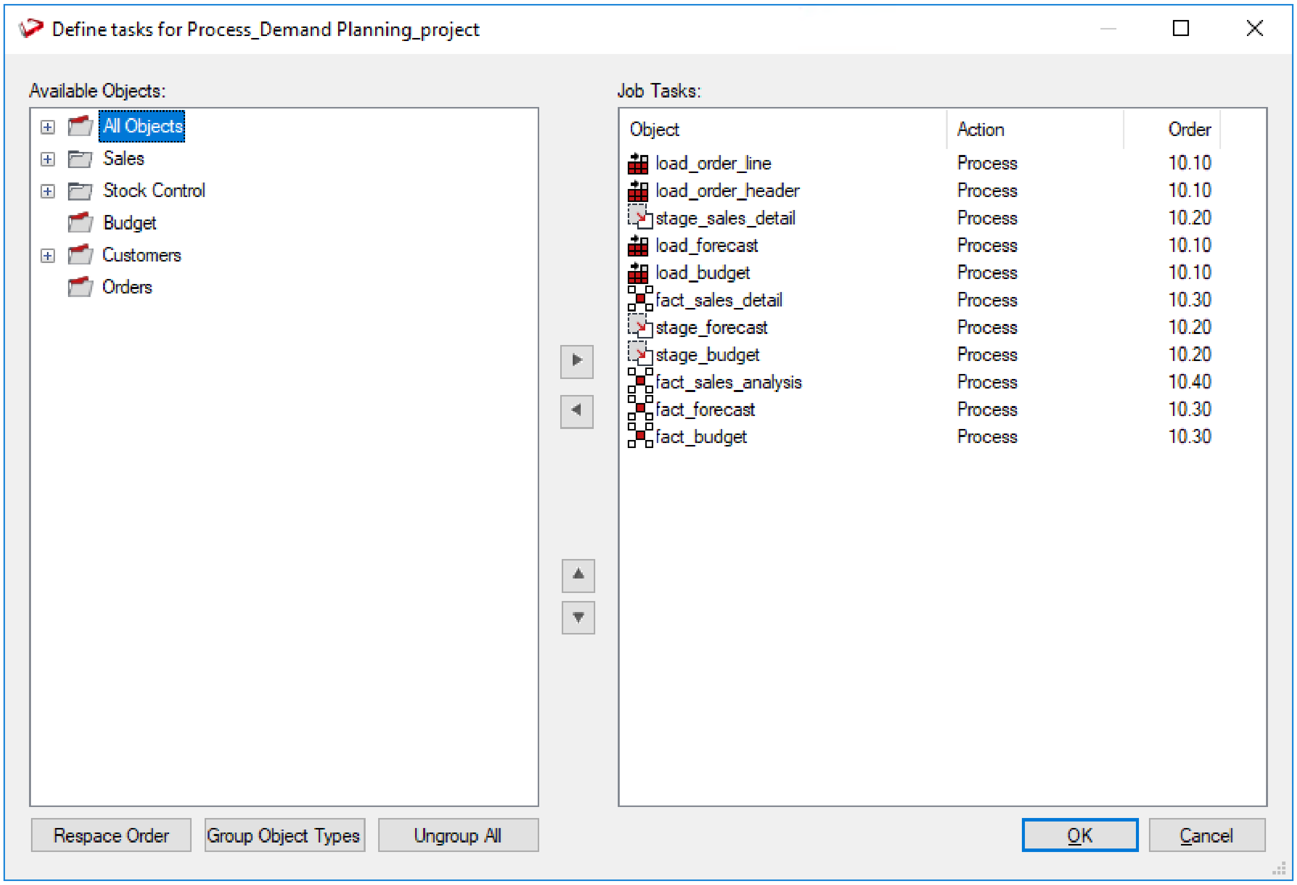
The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

Success Command:

Failure Command:

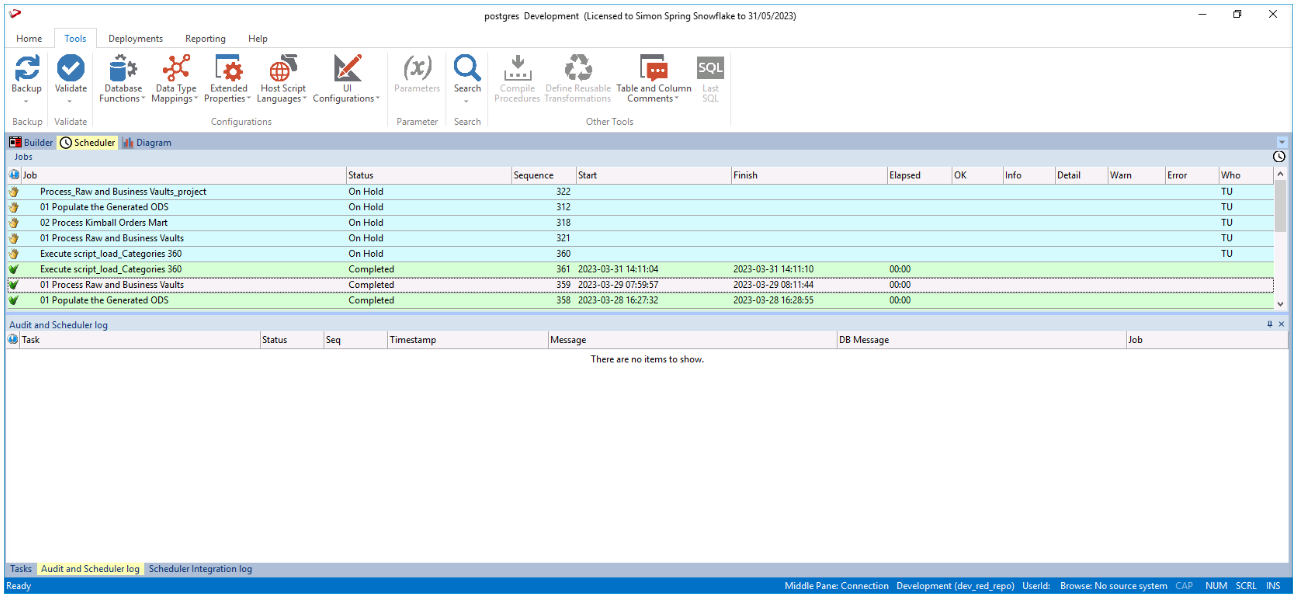
□ Execute Failure Command in event of dependency failure

4. Review the **Job Tasks** that have been defined for the job and remove tasks you want to exclude as required.



Object	Action	Order
load_order_line	Process	10.10
load_order_header	Process	10.10
stage_sales_detail	Process	10.20
load_forecast	Process	10.10
load_budget	Process	10.10
fact_sales_detail	Process	10.30
stage_forecast	Process	10.20
stage_budget	Process	10.20
fact_sales_analysis	Process	10.40
fact_forecast	Process	10.30
fact_budget	Process	10.30

5. Click **OK** to create the job, the **Scheduler** window is displayed.
   Refresh the screen to see the created job, its status and other details.



Job	Status	Sequence	Start	Finish	Elapsed	OK	Info	Detail	Warn	Error	Who
Process_Raw and Business Vaults_project	On Hold	322									TU
01 Populate the Generated ODS	On Hold	312									TU
02 Process Kimball Orders Mart	On Hold	318									TU
01 Process Raw and Business Vaults	On Hold	321									TU
Execute script_load_Categories 360	On Hold	360									TU
Execute script_load_Categories 360	Completed	361	2023-03-31 14:11:04	2023-03-31 14:11:10	00:00						
01 Process Raw and Business Vaults	Completed	359	2023-03-29 07:59:57	2023-03-29 08:11:44	00:00						
01 Populate the Generated ODS	Completed	358	2023-03-28 16:27:32	2023-03-28 16:28:55	00:00						

Task	Status	Seq	Timestamp	Message	DB Message	Job

There are no items to show.

Refer to **Creating a Job** for details about the fields and settings in the **Job Definition** and **Define tasks** windows.

Note:

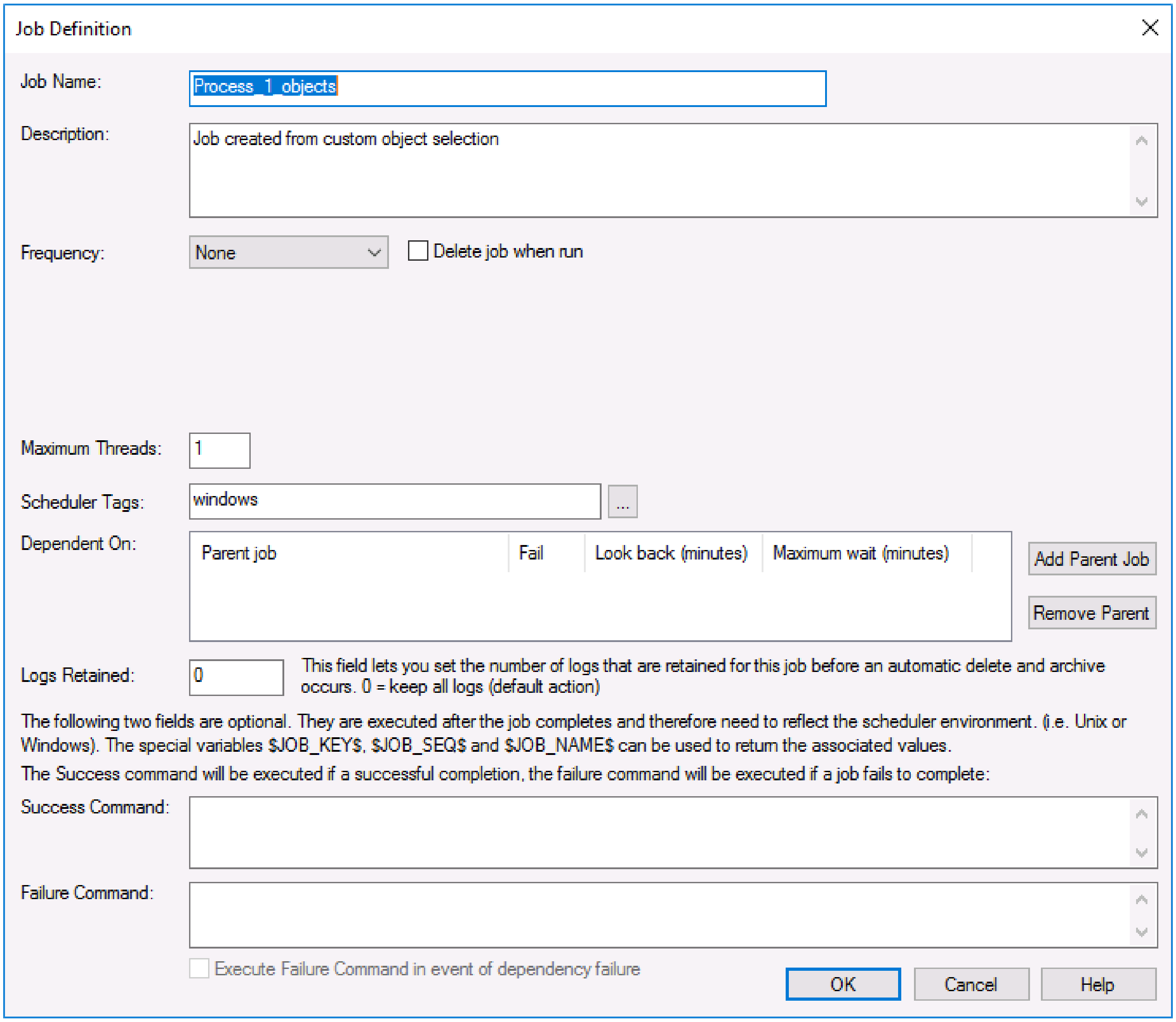
The **Build** context menu is not available for the **All Objects** project, listed in the left pane of the **Builder** window.

Building Scheduler Jobs from Object Groups

WhereScape RED enables you to automatically create a **Scheduler job** from table objects in an Object Group defined in RED. This feature utilizes the track-back information for the objects selected in an Object Group to determine their task dependencies and then creates the Scheduler job.

The steps are outlined below:

1. Double click the Object Group name (e.g. Fact Table, Stage Table, etc.) from the left pane of the **Builder** window to display all the tables in the selected Object Group in the middle pane.
2. Select a table or multiple tables in the middle pane that you want to include in the Scheduler Job.
3. Right-click a highlighted table and then select the **Build > Build Job** option from the context menu.
4. Type in the **Job Name** and make any changes to the details and settings in the **Job Definition** window as required and then click **OK** to define the tasks in the Scheduler job.



Job Definition

Job Name: Process_1_objects

Description: Job created from custom object selection

Frequency: None

□ Delete job when run

Maximum Threads: 1

Scheduler Tags: windows

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)

Logs Retained: 0

This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

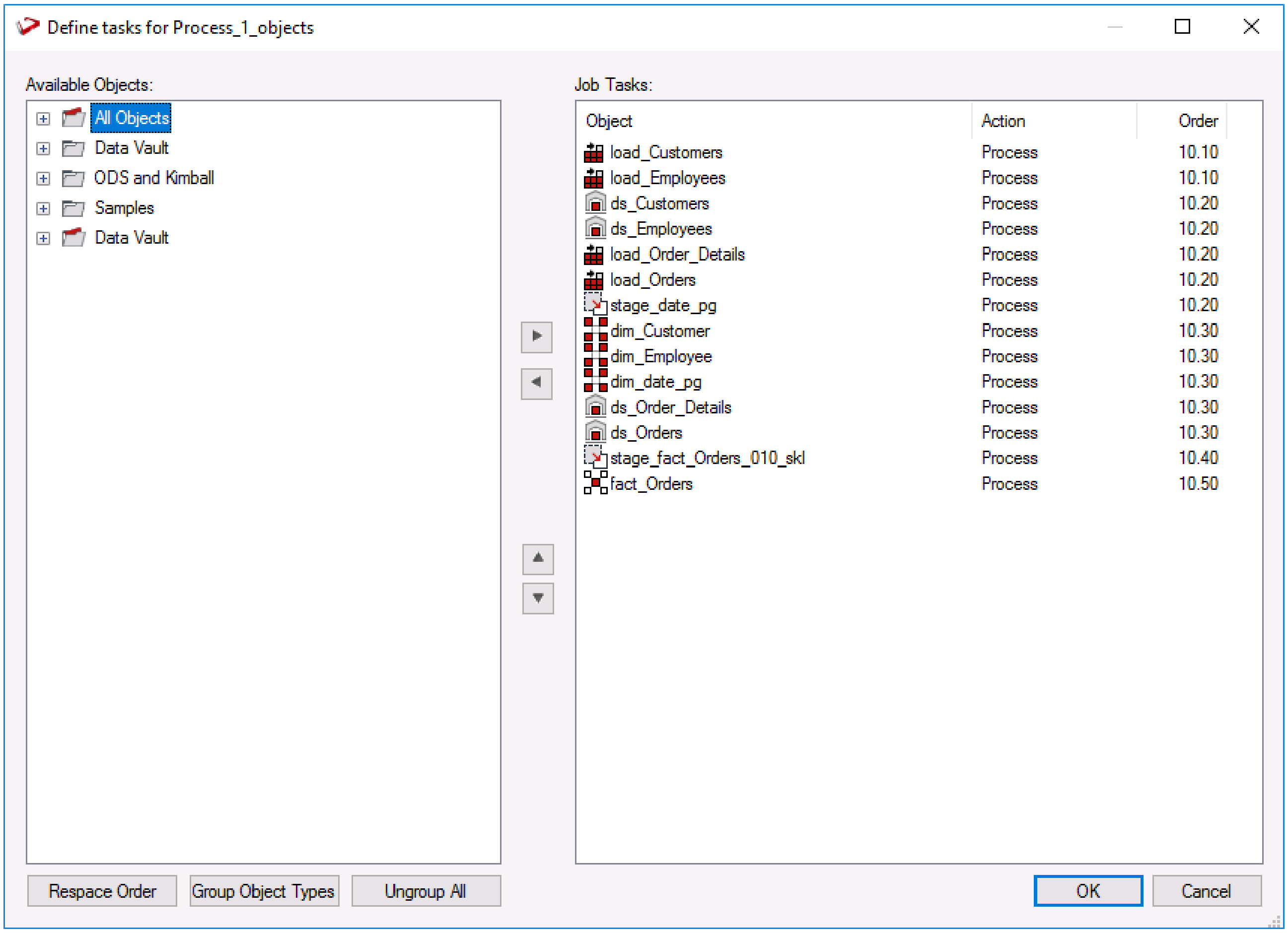
The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

Success Command:

Failure Command:

□ Execute Failure Command in event of dependency failure

5. Review the **Job Tasks** that have been defined for the job and remove tasks you want to exclude as required.



Define tasks for Process_1_objects

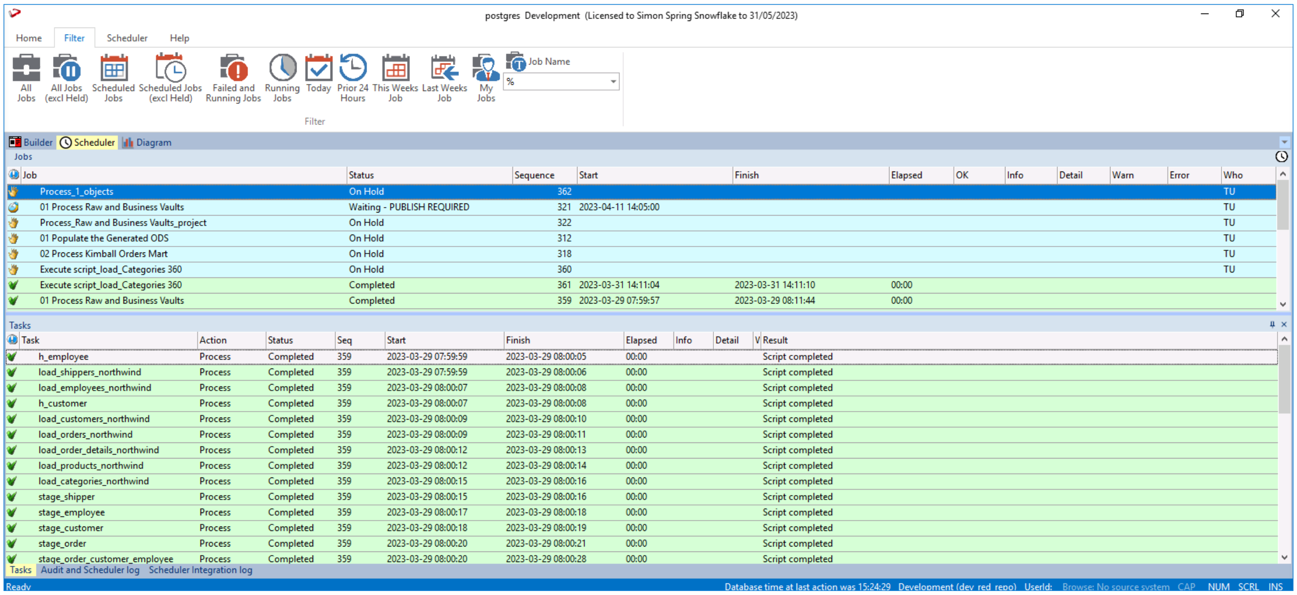
Available Objects:

- All Objects
- Data Vault
- ODS and Kimball
- Samples
- Data Vault

Job Tasks:

Object	Action	Order
load_Customers	Process	10.10
load_Employees	Process	10.10
ds_Customers	Process	10.20
ds_Employees	Process	10.20
load_Order_Details	Process	10.20
load_Orders	Process	10.20
stage_date_pg	Process	10.20
dim_Customer	Process	10.30
dim_Employee	Process	10.30
dim_date_pg	Process	10.30
ds_Order_Details	Process	10.30
ds_Orders	Process	10.30
stage_fact_Orders_010_skl	Process	10.40
fact_Orders	Process	10.50

6. Click **OK** to create the job, the **Scheduler** window is displayed.
   Refresh the screen to see the created job, its status and other details.



postgres Development (Licensed to Simon Spring Snowflake to 31/05/2023)

Jobs

Job	Status	Sequence	Start	Finish	Elapsed	OK	Info	Detail	Warn	Error	Who
Process_1_objects	On Hold	362									TU
01 Process Raw and Business Vaults	Waiting - PUBLISH REQUIRED	321	2023-04-11 14:05:00								TU
Process_Raw and Business Vaults_project	On Hold	322									TU
01 Populate the Generated ODS	On Hold	312									TU
02 Process Kimball Orders Mart	On Hold	318									TU
Execute script_load_Categories 360	On Hold	360									TU
Execute script_load_Categories 360	Completed	361	2023-03-31 14:11:04	2023-03-31 14:11:10	00:00						
01 Process Raw and Business Vaults	Completed	359	2023-03-29 07:59:57	2023-03-29 08:11:44	00:00						

Tasks

Task	Action	Status	Seq	Start	Finish	Elapsed	Info	Detail	V	Result
h_employee	Process	Completed	359	2023-03-29 07:59:59	2023-03-29 08:00:05	00:00				Script completed
load_shippers_northwind	Process	Completed	359	2023-03-29 07:59:59	2023-03-29 08:00:06	00:00				Script completed
load_employees_northwind	Process	Completed	359	2023-03-29 08:00:07	2023-03-29 08:00:08	00:00				Script completed
h_customer	Process	Completed	359	2023-03-29 08:00:07	2023-03-29 08:00:08	00:00				Script completed
load_customers_northwind	Process	Completed	359	2023-03-29 08:00:09	2023-03-29 08:00:10	00:00				Script completed
load_orders_northwind	Process	Completed	359	2023-03-29 08:00:09	2023-03-29 08:00:11	00:00				Script completed
load_order_details_northwind	Process	Completed	359	2023-03-29 08:00:12	2023-03-29 08:00:13	00:00				Script completed
load_products_northwind	Process	Completed	359	2023-03-29 08:00:12	2023-03-29 08:00:14	00:00				Script completed
load_categories_northwind	Process	Completed	359	2023-03-29 08:00:15	2023-03-29 08:00:16	00:00				Script completed
stage_shipper	Process	Completed	359	2023-03-29 08:00:15	2023-03-29 08:00:16	00:00				Script completed
stage_employee	Process	Completed	359	2023-03-29 08:00:17	2023-03-29 08:00:18	00:00				Script completed
stage_customer	Process	Completed	359	2023-03-29 08:00:18	2023-03-29 08:00:19	00:00				Script completed
stage_order	Process	Completed	359	2023-03-29 08:00:20	2023-03-29 08:00:21	00:00				Script completed
stage_order_customer_employee	Process	Completed	359	2023-03-29 08:00:20	2023-03-29 08:00:28	00:00				Script completed

Ready   Database time at last action was 15:24:29 Development (dev_red_repo) UserId: Browse: No source system CAP NUM SCRL INS

Refer to **Creating a Job** for details about the fields and settings in the **Job Definition** and **Define tasks** windows.

Building Applications

Deployment applications can be created from Object Groups, Projects and Project Groups defined in RED via the **Build > Build Job** context menu. The detailed steps are described in the following sections.

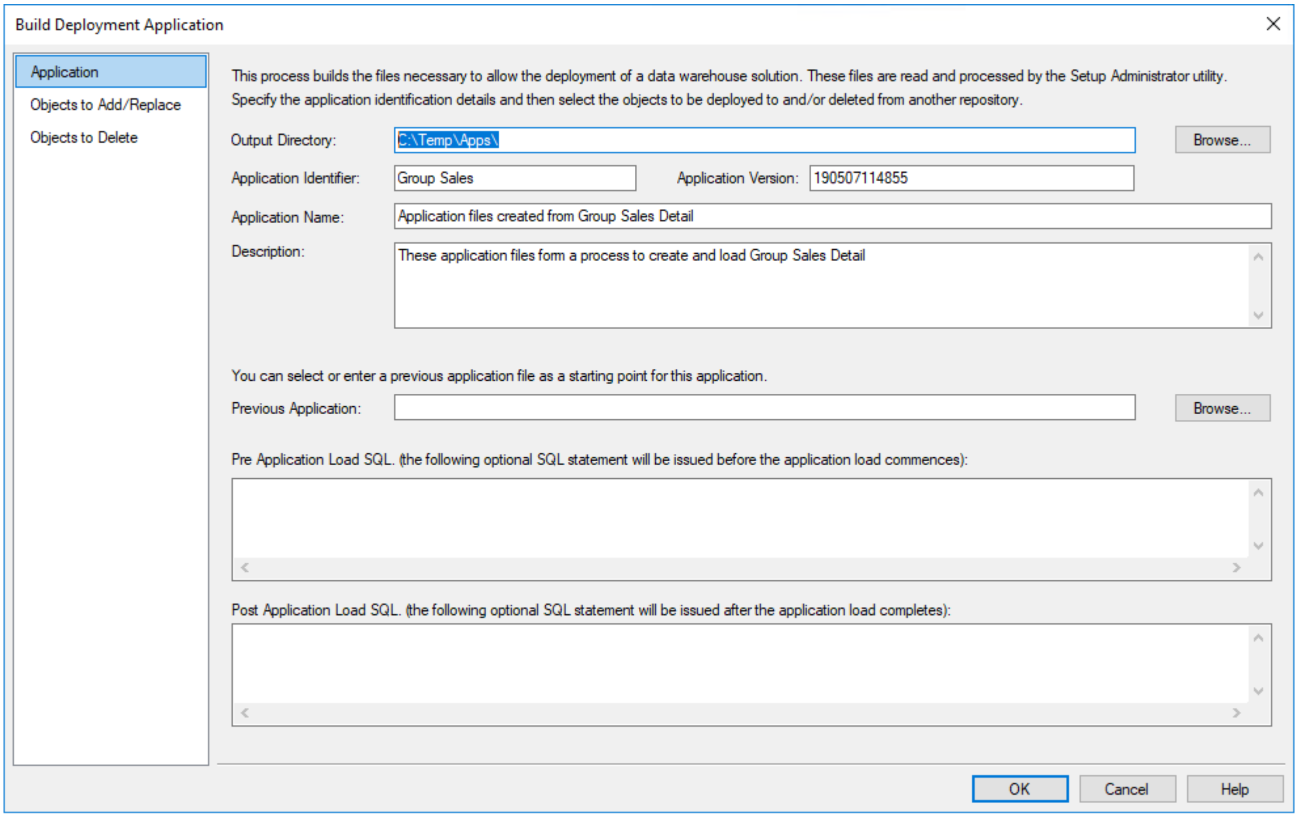
- Building Applications from Projects or Groups
- Building Applications from Object Groups

Building Applications from Projects or Groups

WhereScape RED enables you to automatically create a Deployment Application from all the table objects in a Project or Project Group defined in RED. This feature utilizes the track-back information for all the objects included in a Project or Project Group to determine the objects to be deployed and their task dependencies; and then creates the Application.

The steps are outlined below:

1. Right click the **Project** or **Group** from the left pane of the **Builder** window.
2. Select the **Build > Build Application** option from the context menu.
3. Make any changes to the details and settings in the **Build Deployment Application** window as required.



Build Deployment Application

This process builds the files necessary to allow the deployment of a data warehouse solution. These files are read and processed by the Setup Administrator utility. Specify the application identification details and then select the objects to be deployed to and/or deleted from another repository.

Output Directory: C:\Temp\Apps\

Application Identifier: Group Sales

Application Version: 190507114855

Application Name: Application files created from Group Sales Detail

Description: These application files form a process to create and load Group Sales Detail

You can select or enter a previous application file as a starting point for this application.

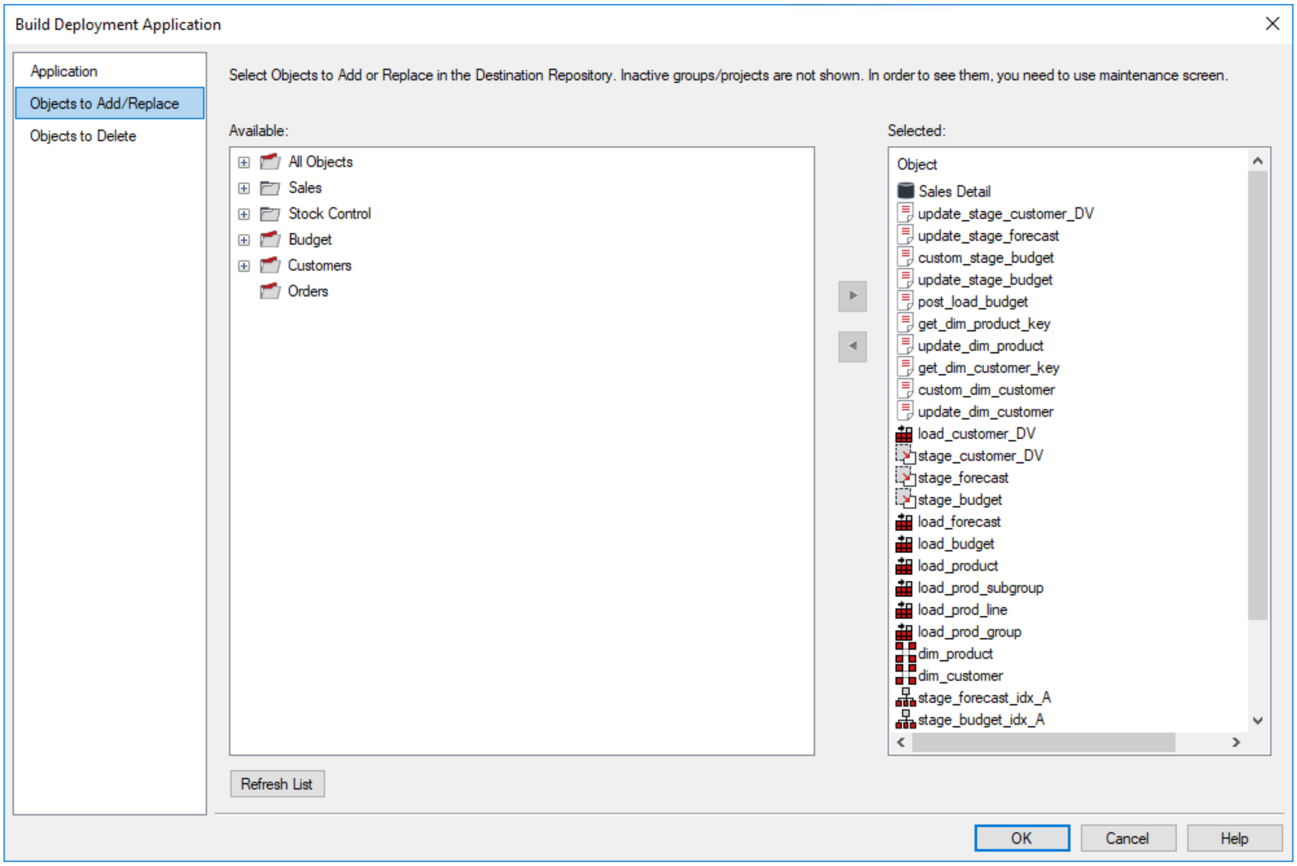
Previous Application:

Pre Application Load SQL. (the following optional SQL statement will be issued before the application load commences):

Post Application Load SQL. (the following optional SQL statement will be issued after the application load completes):

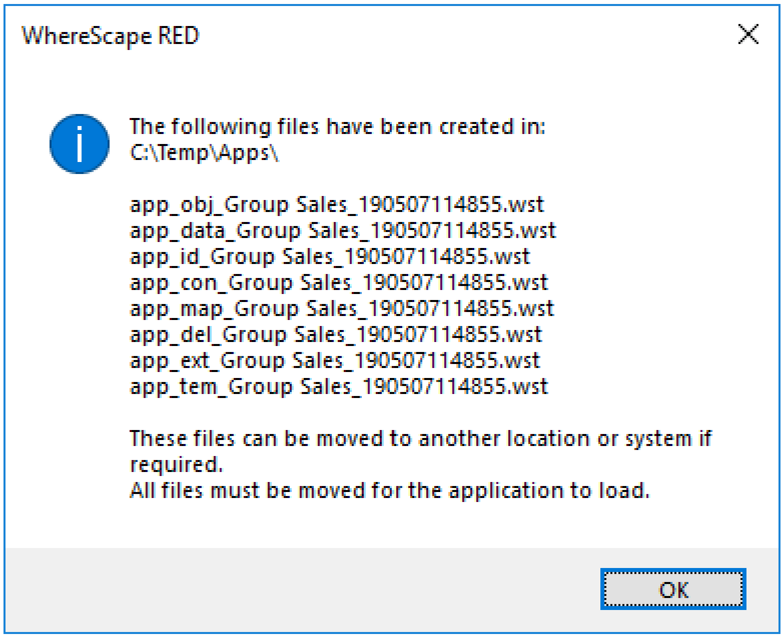
Refer to Application Creation for details about the fields and settings in the **Build Deployment Application** window.

4. Click the other tabs to check or change the selected objects to be deployed in the target metadata repository



Select Objects to Add or Replace in the Destination Repository. Inactive groups/projects are not shown. In order to see them, you need to use maintenance screen.

5. Click **OK** to create the application. A message is displayed to confirm the creation of the application files.



The following files have been created in:
C:\Temp\Apps\

app_obj_Group Sales_190507114855.wst
app_data_Group Sales_190507114855.wst
app_id_Group Sales_190507114855.wst
app_con_Group Sales_190507114855.wst
app_map_Group Sales_190507114855.wst
app_del_Group Sales_190507114855.wst
app_ext_Group Sales_190507114855.wst
app_tem_Group Sales_190507114855.wst

These files can be moved to another location or system if required.
All files must be moved for the application to load.

Note:

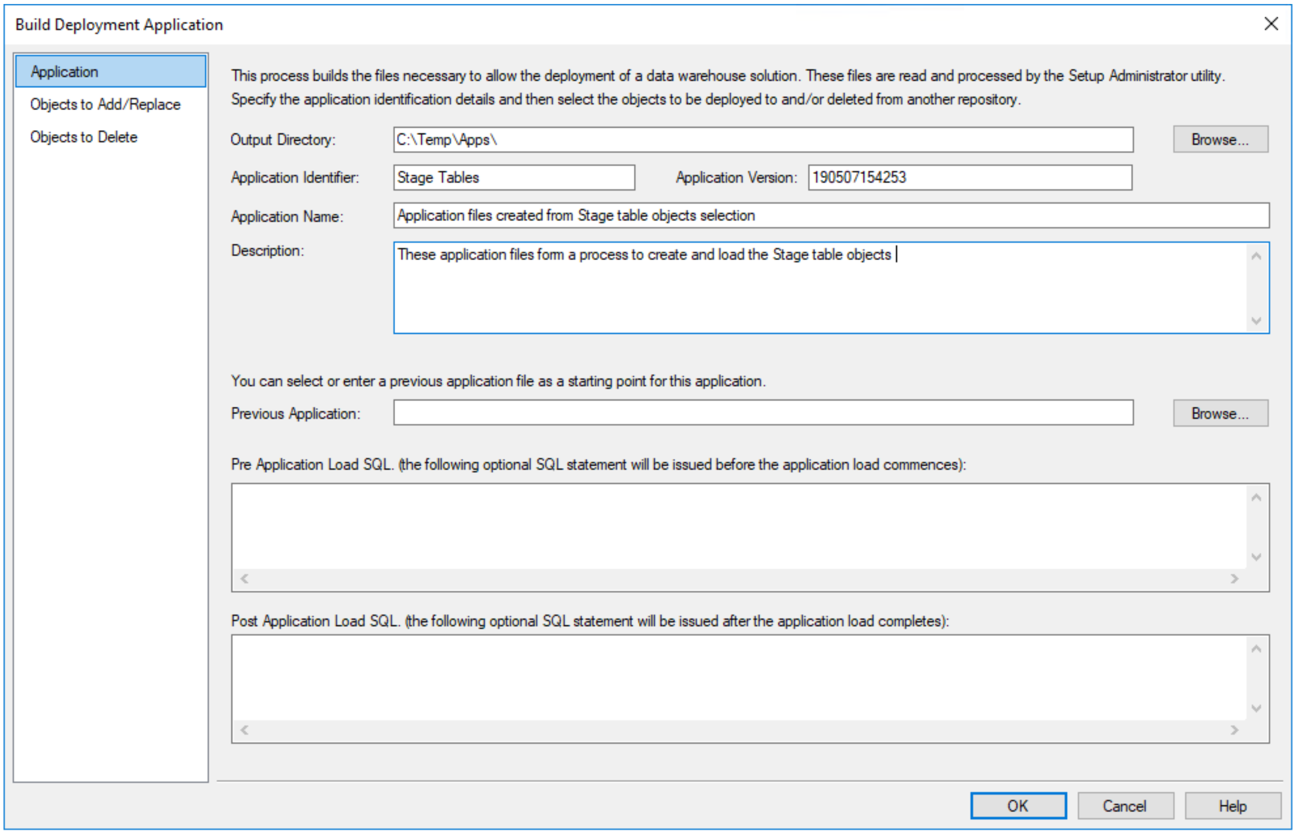
The **Build** context menu is not available for the **All Objects** project, listed in the left pane of the **Builder** window.

Building Applications from Object Groups

WhereScape RED enables you to automatically create a **Deployment Application** from table objects in an Object Group defined in RED. This feature utilizes the track-back information for all the objects included in an Object Group to determine the objects to be deployed and their task dependencies; and then creates the Application.

The steps are outlined below:

1. Double click the Object Group name (e.g. Load Table, Stage Table, etc.) from the left pane of the **Builder** window to display all the tables in the selected Object Group in the middle pane.
2. Select a table or multiple tables in the middle pane that you want to include in the Application.
3. Right-click a highlighted table and then select the **Build > Build Application** option from the context menu.
4. Type in the **Application Identifier** and make any changes to the details and settings in the **Build Deployment Application** window as required.



This process builds the files necessary to allow the deployment of a data warehouse solution. These files are read and processed by the Setup Administrator utility. Specify the application identification details and then select the objects to be deployed to and/or deleted from another repository.

Output Directory: C:\Temp\Apps\

Application Identifier: Stage Tables

Application Version: 190507154253

Application Name: Application files created from Stage table objects selection

Description: These application files form a process to create and load the Stage table objects

You can select or enter a previous application file as a starting point for this application.

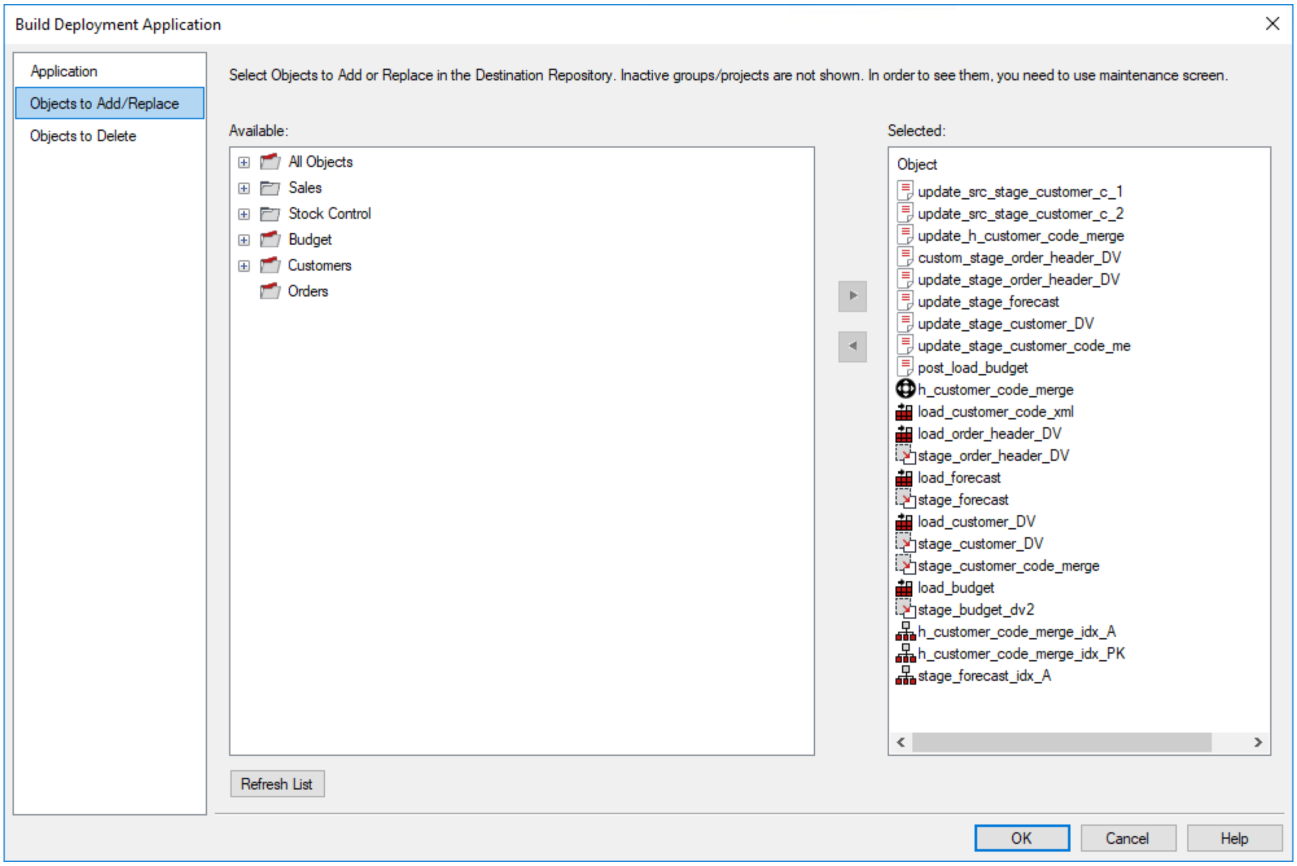
Previous Application:

Pre Application Load SQL. (the following optional SQL statement will be issued before the application load commences):

Post Application Load SQL. (the following optional SQL statement will be issued after the application load completes):

Refer to **Application Creation** for details about the fields and settings in the **Build Deployment Application** window.

5. Click the other tabs to check or change the selected objects to be deployed in the target metadata repository



Build Deployment Application

Select Objects to Add or Replace in the Destination Repository. Inactive groups/projects are not shown. In order to see them, you need to use maintenance screen.

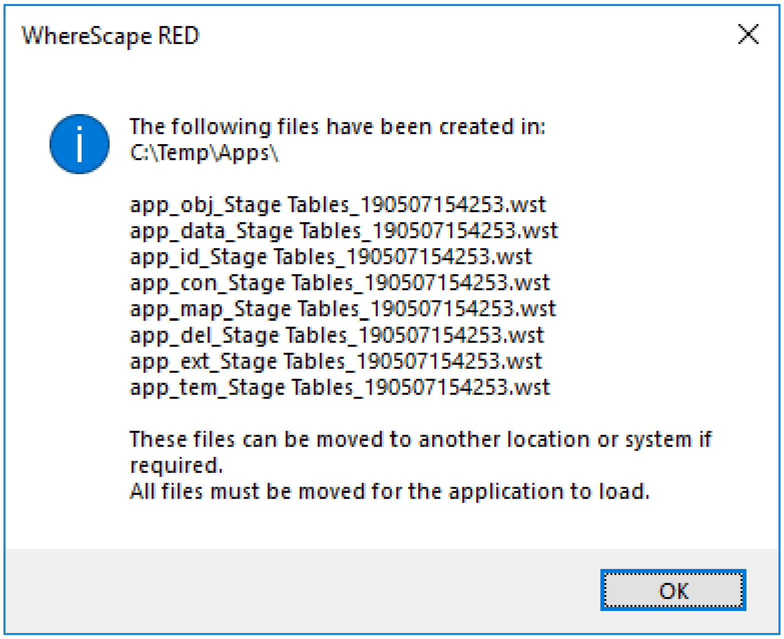
Available:

- All Objects
- Sales
- Stock Control
- Budget
- Customers
- Orders

Selected:

Object
update_src_stage_customer_c_1
update_src_stage_customer_c_2
update_h_customer_code_merge
custom_stage_order_header_DV
update_stage_order_header_DV
update_stage_forecast
update_stage_customer_DV
update_stage_customer_code_me
post_load_budget
h_customer_code_merge
load_customer_code_xml
load_order_header_DV
stage_order_header_DV
load_forecast
stage_forecast
load_customer_DV
stage_customer_DV
stage_customer_code_merge
load_budget
stage_budget_dv2
h_customer_code_merge_idx_A
h_customer_code_merge_idx_PK
stage_forecast_idx_A

6. Click **OK** to create the application. A message is displayed to confirm the creation of the application files.



WhereScape RED

The following files have been created in:
C:\Temp\Apps\

app_obj_Stage Tables_190507154253.wst
app_data_Stage Tables_190507154253.wst
app_id_Stage Tables_190507154253.wst
app_con_Stage Tables_190507154253.wst
app_map_Stage Tables_190507154253.wst
app_del_Stage Tables_190507154253.wst
app_ext_Stage Tables_190507154253.wst
app_tem_Stage Tables_190507154253.wst

These files can be moved to another location or system if required.
All files must be moved for the application to load.

Windows and Panes

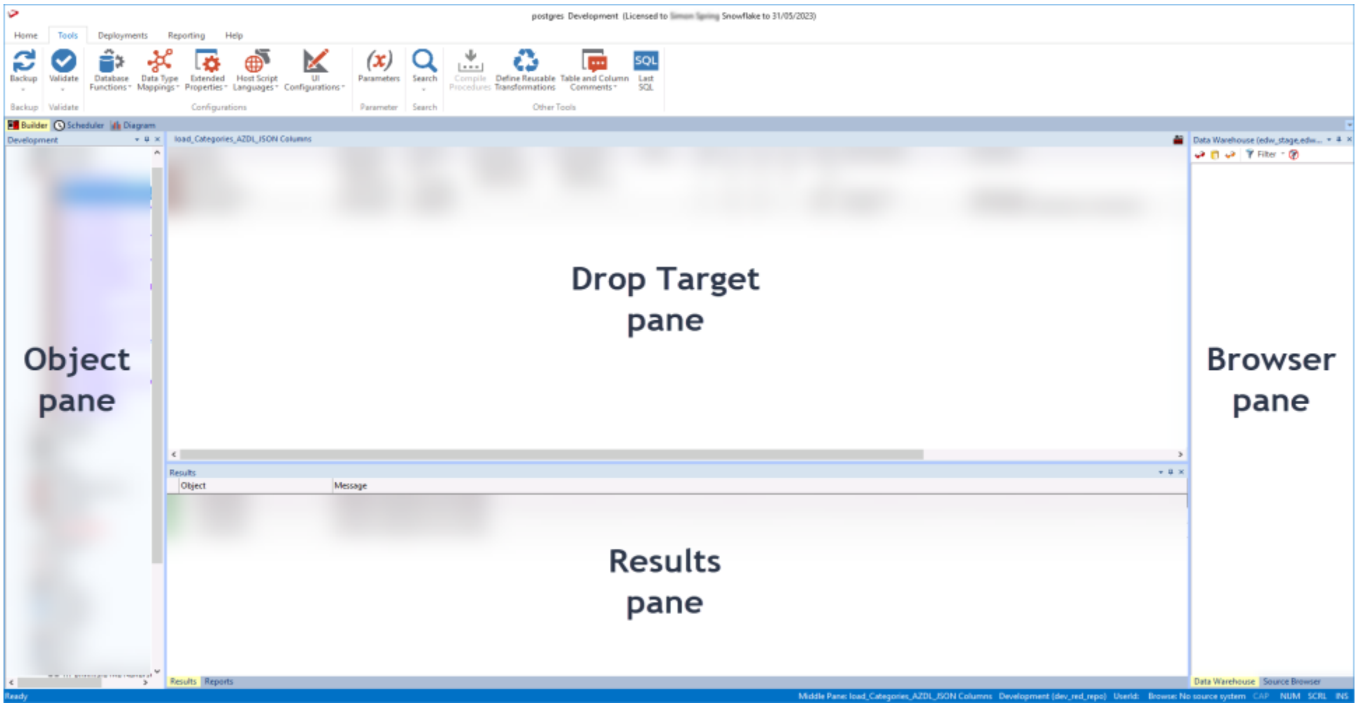
WhereScape RED has a number of different windows that are utilized in the building and maintenance of a data warehouse. Each window, in some cases are divided into panes.

There are four main windows in RED that are used extensively in building a data warehouse:

- The **Builder Window**.
- The **Scheduler Window**.
- The **Diagram Window**.
- The **Procedure Editor Window**.

Builder Window

The Builder window has four panes.



Panes	Description
Object Pane	This pane contains all the objects in the metadata repository. These objects are stored in object groups (e.g. Dimension, Fact, etc.). The object groups in turn can optionally be stored in Projects, and the Projects can optionally be stored within Project Groups.
Drop Target Pane	This pane is used to display the results of various queries on both the metadata and the underlying source and database tables. The middle pane is also used as the drop target in drag and drop operations. The status line at the bottom of the screen displays the current contents of the middle pane. To send the middle pane output to a file or to clipboard, refer to **Export Middle Pane Output** for details.
Results Pane	This pane is used to display the results of various queries on both the metadata and the underlying source and database tables.
Browser Pane	This pane displays both Source and Data Warehouse systems. There are two browser panes available at any one time. The source may be the data warehouse itself. The Browser Pane can be filtered: • To filter by type, click the down arrow next to the **Filter** button. • To filter by name, click the **Filter** button and enter the filter criteria in the **Filter** dialog. Name based filters can be cleared by clicking the down arrow next to the **Filter**

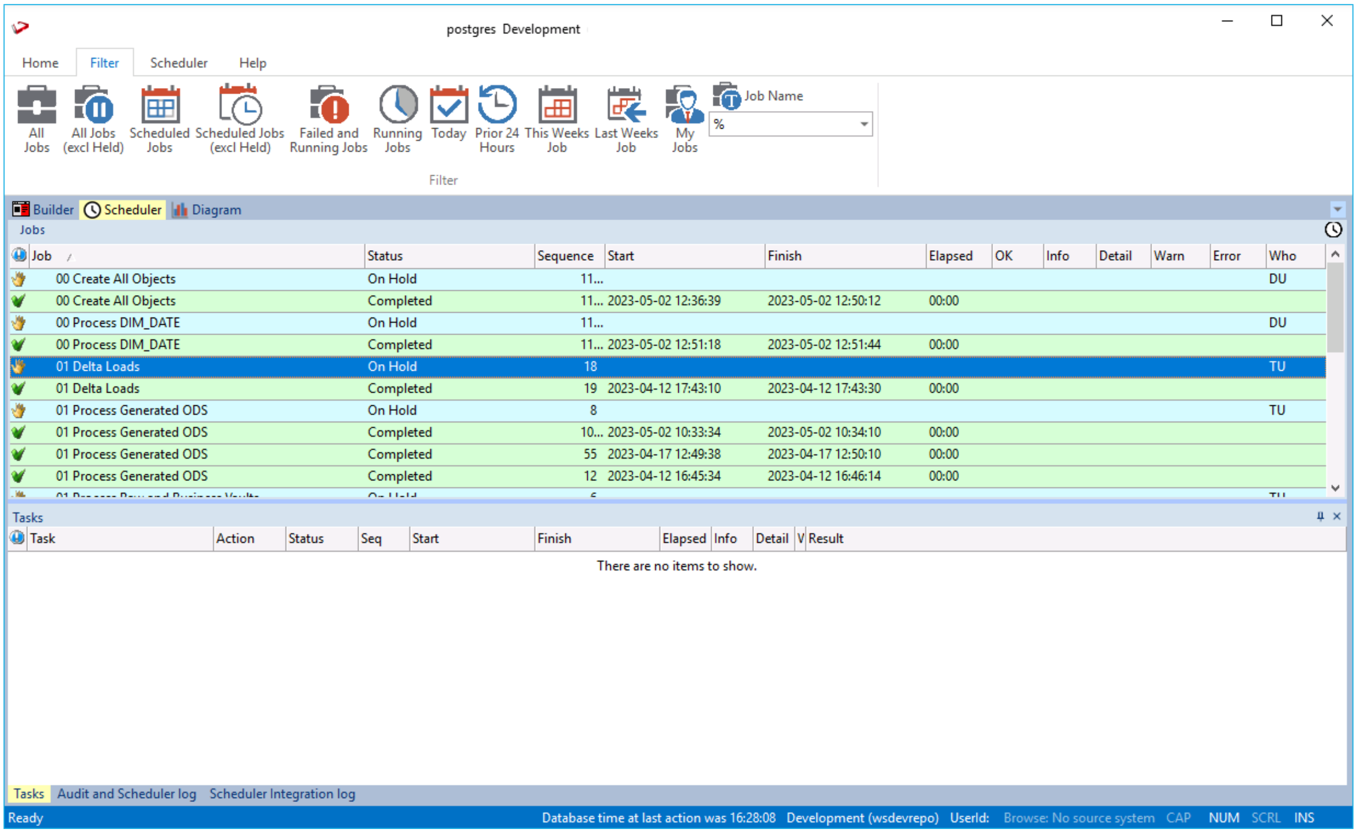
Panes	Description
	button and selecting **Clear Name Based Filter** from the menu.

Tips:

- Pop-up menus are available in all four panes. Use **Ctrl+M** to display the pop-up menu when an item is selected in any of the panes.
- You can select multiple items displayed in the drop target/middle pane.
- **F5** or **Ctrl+R** can be used to refresh the left and right panes, when the cursor is positioned within these panes.
- The left, bottom and right panes can be dragged out and docked elsewhere. Docking handles appear when a pane is dragged which you can use to specify the location for the pane, as shown below:

Scheduler Window

The **Scheduler Window** is used as the main interface to the scheduler. Jobs can be scheduled, monitored, edited and deleted through this window.



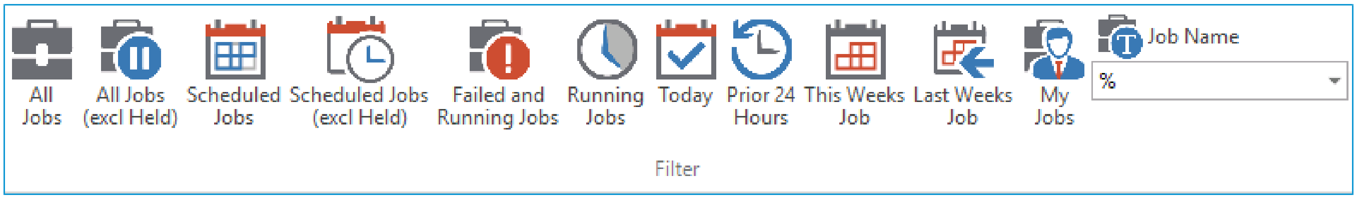
Job	Status	Sequence	Start	Finish	Elapsed	OK	Info	Detail	Warn	Error	Who
00 Create All Objects	On Hold	11...									DU
00 Create All Objects	Completed	11...	2023-05-02 12:36:39	2023-05-02 12:50:12	00:00						
00 Process DIM_DATE	On Hold	11...									DU
00 Process DIM_DATE	Completed	11...	2023-05-02 12:51:18	2023-05-02 12:51:44	00:00						
01 Delta Loads	On Hold	18									TU
01 Delta Loads	Completed	19	2023-04-12 17:43:10	2023-04-12 17:43:30	00:00						
01 Process Generated ODS	On Hold	8									TU
01 Process Generated ODS	Completed	10...	2023-05-02 10:33:34	2023-05-02 10:34:10	00:00						
01 Process Generated ODS	Completed	55	2023-04-17 12:49:38	2023-04-17 12:50:10	00:00						
01 Process Generated ODS	Completed	12	2023-04-12 16:45:34	2023-04-12 16:46:14	00:00						

Task	Action	Status	Seq	Start	Finish	Elapsed	Info	Detail	V	Result

There are no items to show.

Ready   Database time at last action was 16:28:08 Development (wsdevrepo) UserId: Browse: No source system CAP NUM SCRL INS

Toolbar



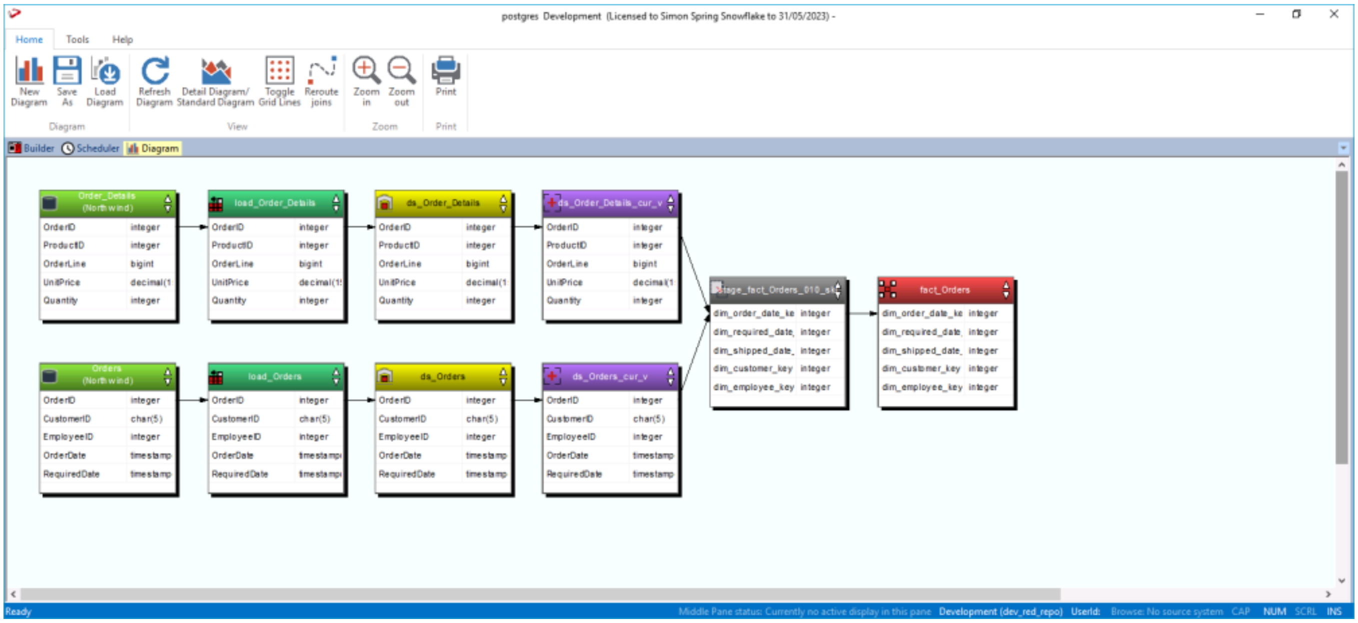
Icon	Description
	All Jobs - Click to view all jobs.
	Scheduled - Click to view scheduled jobs.
	Run/ Fail - Click to view jobs that run/failed.
	Running Jobs - Click to view the current running jobs.
	Today - Click to view jobs scheduled for today.
	Prior 24 Hours- Click to view all the jobs prior the 24 hours.
	This Week Job - Click to view this week jobs.
	Last Weeks Job - Click to view last week jobs.
	My Jobs - Click to view jobs you created.
Job Name %	**Job Name Filter** - Optional Job name criteria to be applied via the core job selection criteria to further filter jobs.

- Double-click the job in the top pane to see the tasks of the selected job in the bottom pane.
- Double-click the task in the bottom pane to see the audit trail displayed in a separate tab in the bottom pane.

Refer to **Scheduler** for more details.

Diagram Window

The **Diagram window** is used to display the tables of the data warehouse in diagrammatic form, showing the various sources or targets of the selected object.



Order_Details (Northwind)	
OrderID	integer
ProductID	integer
OrderLine	bigint
UnitPrice	decimal(1
Quantity	integer

load_Order_Details	
OrderID	integer
ProductID	integer
OrderLine	bigint
UnitPrice	decimal(1!
Quantity	integer

ds_Order_Details	
OrderID	integer
ProductID	integer
OrderLine	bigint
UnitPrice	decimal(1
Quantity	integer

ds_Order_Details_cur_v	
OrderID	integer
ProductID	integer
OrderLine	bigint
UnitPrice	decimal(1
Quantity	integer

Orders (Northwind)	
OrderID	integer
CustomerID	char(5)
EmployeeID	integer
OrderDate	timestamp
RequiredDate	timestamp

load_Orders	
OrderID	integer
CustomerID	char(5)
EmployeeID	integer
OrderDate	timestamp
RequiredDate	timestamp

ds_Orders	
OrderID	integer
CustomerID	char(5)
EmployeeID	integer
OrderDate	timestamp
RequiredDate	timestamp

ds_Orders_cur_v	
OrderID	integer
CustomerID	char(5)
EmployeeID	integer
OrderDate	timestamp
RequiredDate	timestamp

stage_fact_Orders_010_sk	
dim_order_date_ke	integer
dim_required_date_	integer
dim_shipped_date_	integer
dim_customer_key	integer
dim_employee_key	integer

fact_Orders	
dim_order_date_ke	integer
dim_required_date_	integer
dim_shipped_date_	integer
dim_customer_key	integer
dim_employee_key	integer

Ready   Middle Pane status: Currently no active display in this pane   Development (dev_red_repo)   UserId: Browse: No source system   CAP   NUM   SCRL   INS

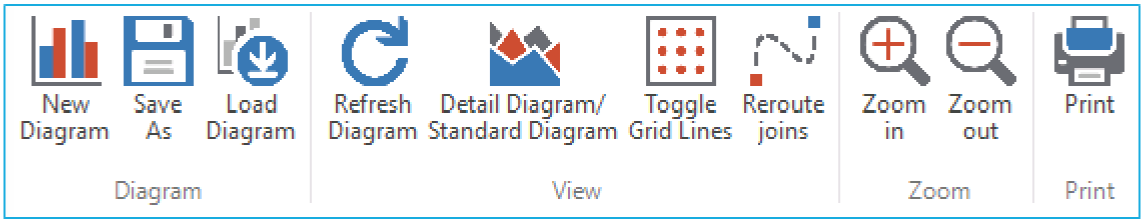
The Diagram Selection is as follows:

- Schema Diagram

- Source Diagram
- Joins Diagram
- Links Diagram
- Spine Diagram
- Impact Diagram
- Dependency Diagram

Refer to **Diagrams** for details.

Toolbar



Icon	Description
📊	**New Diagram** - Click to select a diagram type and table.
💾	**Save As** - Click to Save the diagram.
📥	**Load Diagram**- Click to load a diagram.
↻	**Refresh Diagram** - Click to refresh the diagram.
⛰	**Detail Diagram/Standard Diagram** - Click to expand the diagram for detail view or to toggle the diagram.
⁙	**Toggle grid lines** - Click to toggle printing of grid lines.
∿	**Reroute joins** - Click to reroute the connections.
⊕	**Zoom in** - Click to Zoom In.
⊖	**Zoom out** - Click to Zoom Out.
🖨	**Print** - Click to print the diagram.

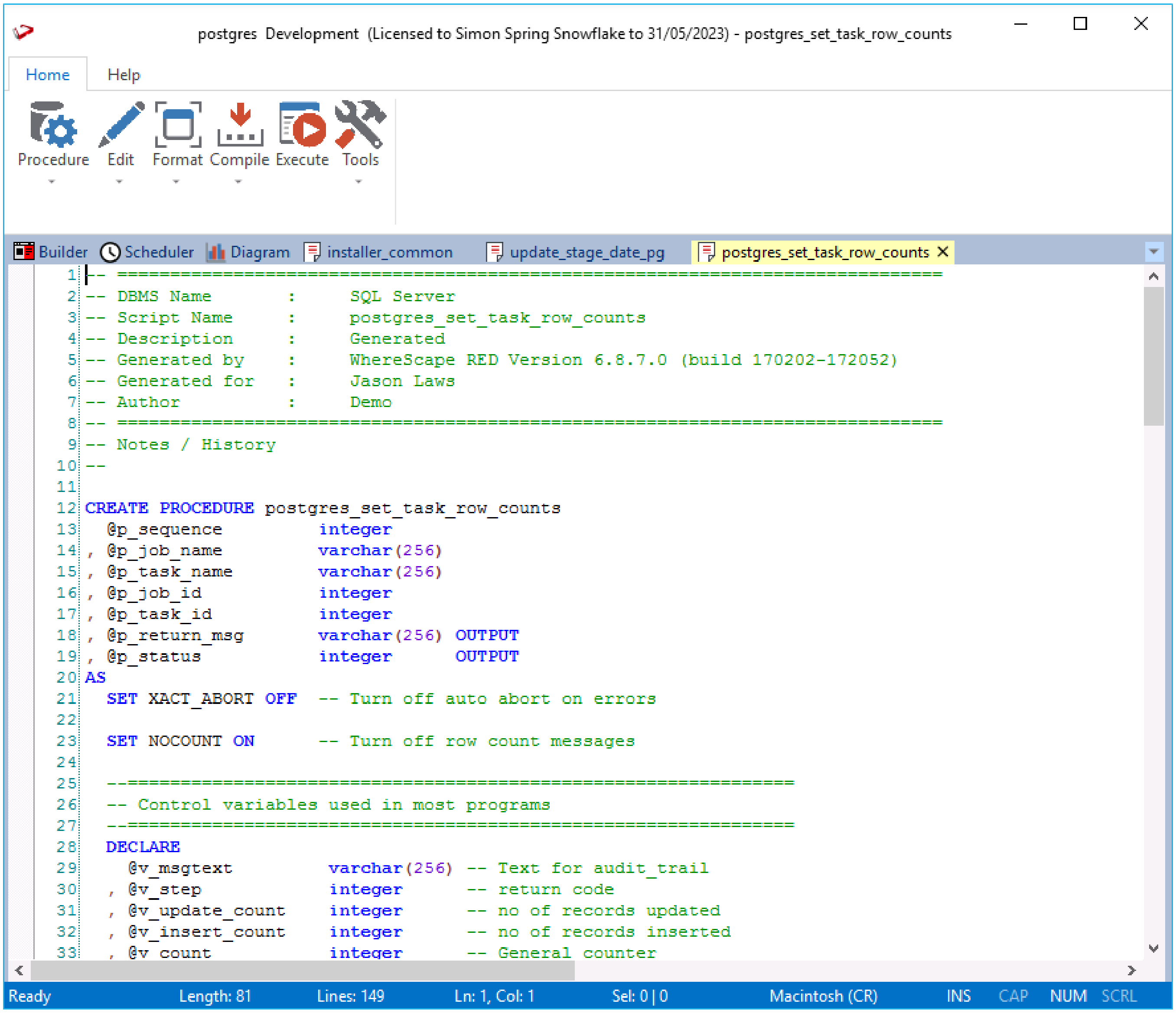
Procedure Editor Window

The **Procedure Editor** window provides a means of viewing, editing, compiling, comparing and running procedures.

The editor includes the following functions and features:

- SQL syntax highlighting
- improved find feature (repeated find, up and down, etc.)
- toggle the display of line numbering
- a status bar that includes line number, line length, selected text length, etc.
- a context sensitive toolbar

- tab to space conversion and vice versa
- uppercase, lowercase, titlecase text conversion command
- increase/decrease test indent command
- bookmark support



```
-- ==============================================================================
-- DBMS Name         :     SQL Server
-- Script Name       :     postgres_set_task_row_counts
-- Description       :     Generated
-- Generated by      :     WhereScape RED Version 6.8.7.0 (build 170202-172052)
-- Generated for     :     Jason Laws
-- Author            :     Demo
-- ==============================================================================
-- Notes / History
--

CREATE PROCEDURE postgres_set_task_row_counts
  @p_sequence          integer
, @p_job_name          varchar(256)
, @p_task_name         varchar(256)
, @p_job_id            integer
, @p_task_id           integer
, @p_return_msg        varchar(256) OUTPUT
, @p_status            integer      OUTPUT
AS
  SET XACT_ABORT OFF   -- Turn off auto abort on errors

  SET NOCOUNT ON       -- Turn off row count messages

  --=================================================================
  -- Control variables used in most programs
  --=================================================================
  DECLARE
    @v_msgtext          varchar(256) -- Text for audit_trail
  , @v_step             integer      -- return code
  , @v_update_count     integer      -- no of records updated
  , @v_insert_count     integer      -- no of records inserted
  , @v_count            integer      -- General counter
```

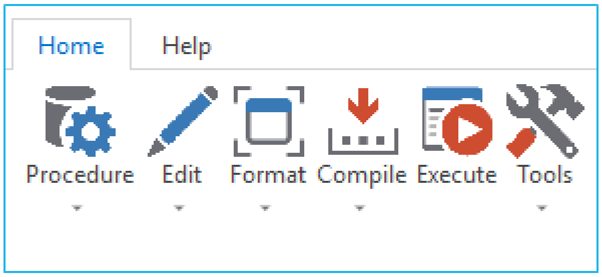
Multiple editor windows can be opened at any one time, each processing a different procedure.

Comments are identified by a leading double dash (--) and displayed in green font in the window, the procedural code in black font.

The font is a fixed pitch font (by default) to make the indentation and alignment of code easier to view. The font, colors and indent size can all be changed as required.

Toolbar

The procedure editor toolbar is shown below:



Refer to **Procedures and Scripts** for details.

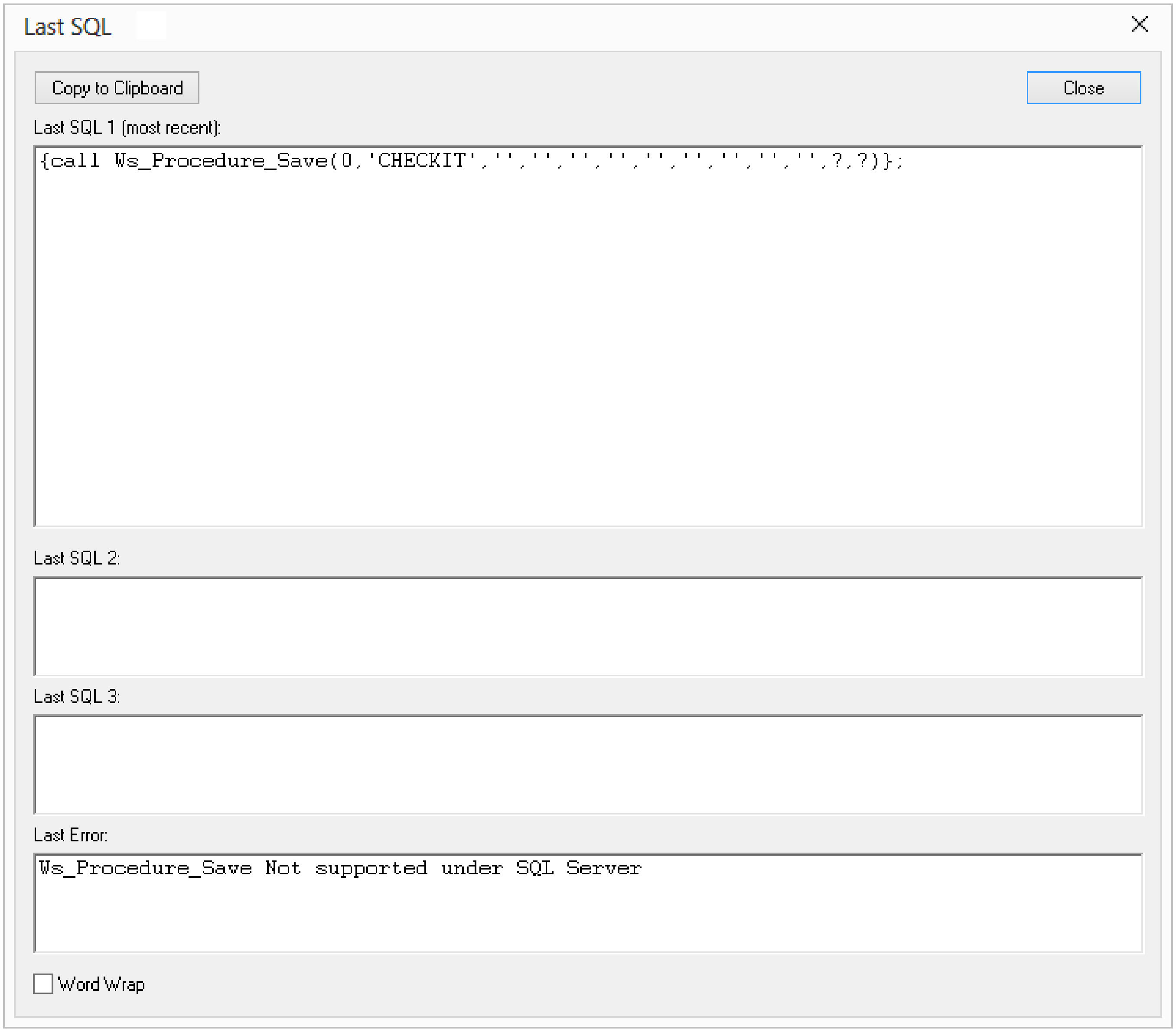
Database Error Messages

From time to time, an SQL error may be raised by WhereScape RED. These are usually caused by the data warehouse database returning an error.

There are many causes of database error messages being returned to RED. The most common causes include permission issues and invalid SQL statements being run.

To find the exact SQL statement run by WhereScape RED, go to the **View** menu and select **Last SQL**.

The following window is displayed:



```
{call Ws_Procedure_Save(0,'CHECKIT','','','','','','','','','',?,?)};
```

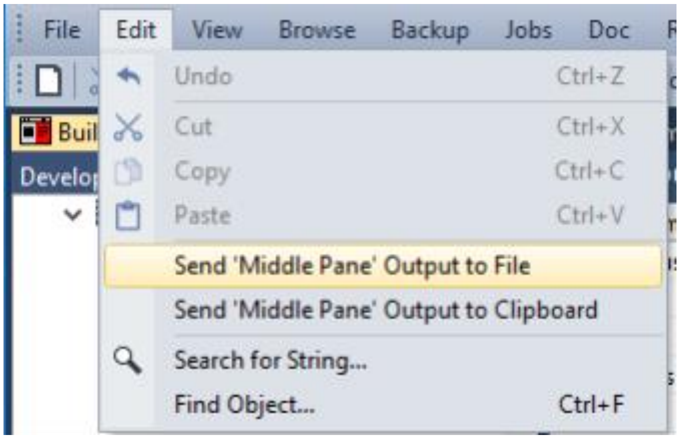
```
Ws_Procedure_Save Not supported under SQL Server
```

The four fields shown are:

- The last SQL statement run
- The second to last SQL statement run
- The third to last SQL statement run
- The error message (if any) from the last SQL statement

Export Middle Pane Output

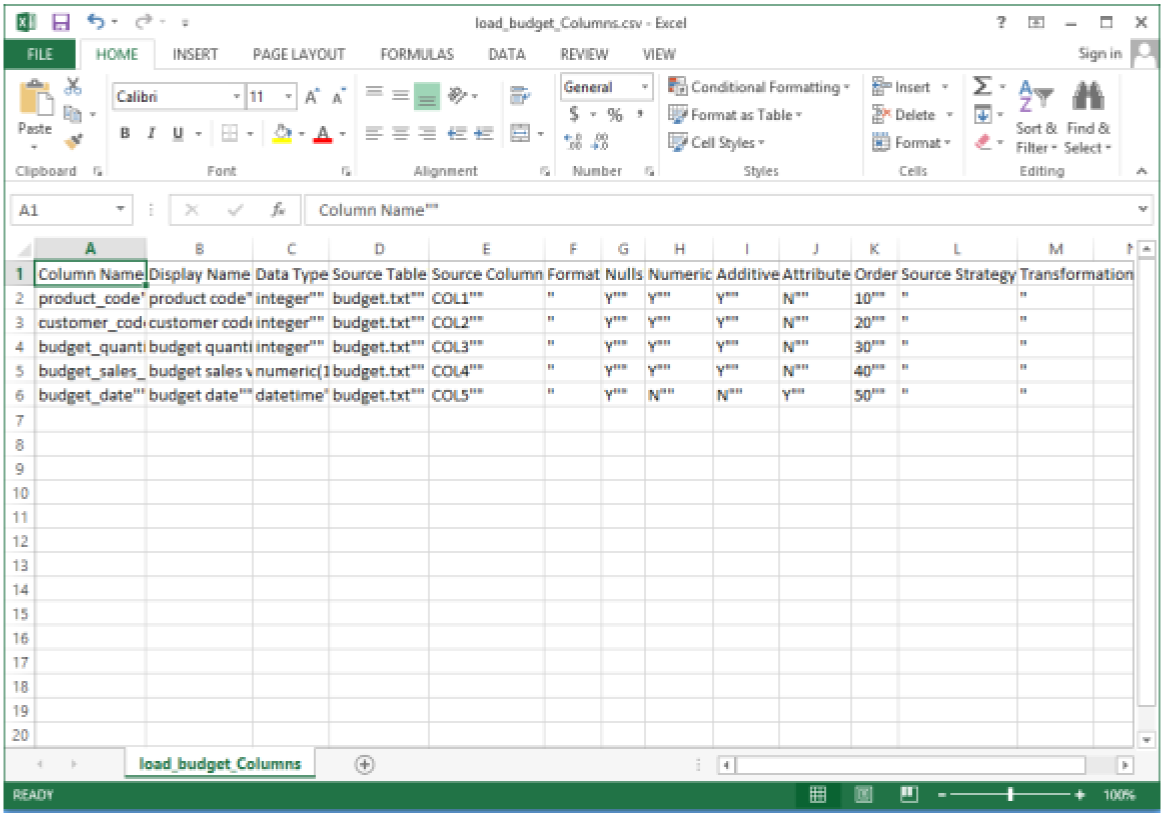
To send the middle pane output to a file, go to the **Edit** menu and select **Send 'Middle Pane' Output to File**.



A file is created in the directory defined in **Tools > User Preferences > Outputs > Output File Directory**.

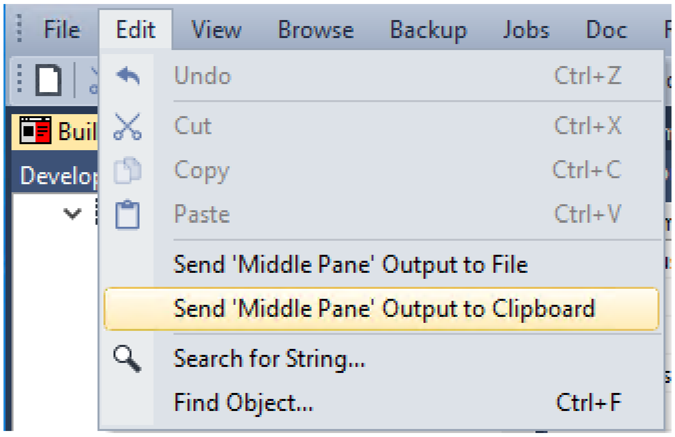
If the Output File Extension is set to .csv in **Tools > User Preferences > Outputs > Output File Extension**, then an Excel file is created.

If the file is set to auto-open in **Tools > User Preferences > Outputs > Output File Auto Open**, then the file opens automatically.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Column Name	Display Name	Data Type	Source Table	Source Column	Format	Nulls	Numeric	Additive	Attribute	Order	Source Strategy	Transformation
2	product_code'	product code"	integer""	budget.txt""	COL1""	"	Y""	Y""	Y""	N""	10""	"	"
3	customer_cod	customer cod	integer""	budget.txt""	COL2""	"	Y""	Y""	Y""	N""	20""	"	"
4	budget_quanti	budget quanti	integer""	budget.txt""	COL3""	"	Y""	Y""	Y""	N""	30""	"	"
5	budget_sales_	budget sales v	numeric(1	budget.txt""	COL4""	"	Y""	Y""	Y""	N""	40""	"	"
6	budget_date"'	budget date""	datetime'	budget.txt""	COL5""	"	Y""	N""	N""	Y""	50""	"	"
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													

- To view the other settings for Middle Pane File Output, refer to Export Middle Pane Output Settings for details.
- To send the middle pane output to clipboard, go to the **Edit** menu and select **Send 'Middle Pane' Output to Clipboard**.

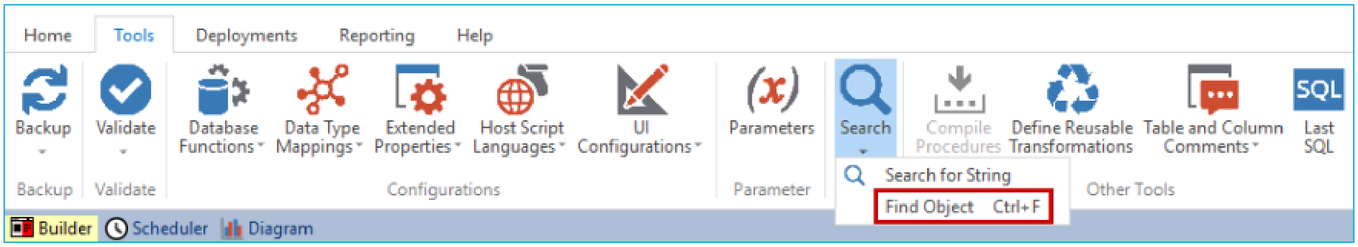


- To view the settings for **Middle Pane Clipboard Output**, refer to [Export Middle Pane Output Settings]() for details.

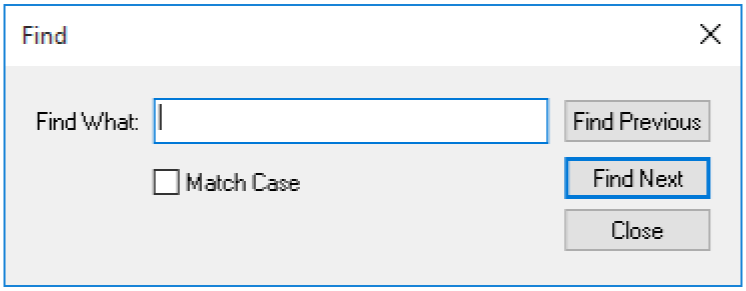
Find Function

The Find function can help users quickly find a table when the list of tables has grown to a large size. The **Find function** can be accessed two ways within WhereScape RED:

- Click anywhere in the Object Pane or Browser Pane and press **Ctrl + F**, or
- select **Tools > Search > Find Object** (this option searches the Object Pane only).



Both of these methods open the following dialog:



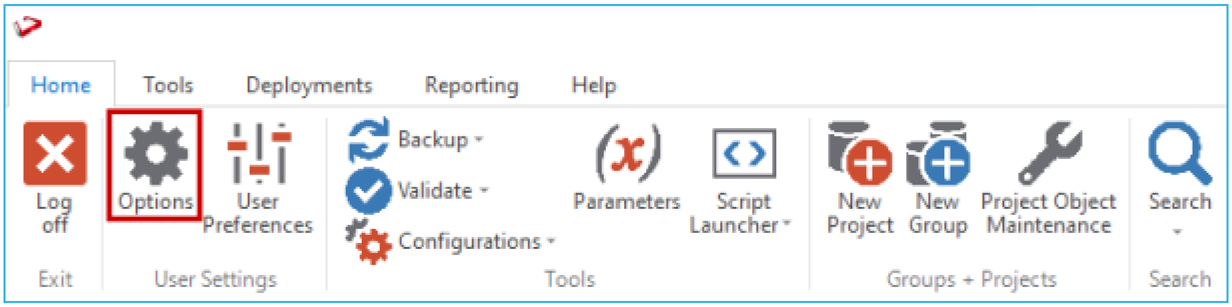
Default Settings

This chapter describes the settings and default values that can be set for a metadata repository.

To access these settings, select the **Tools** menu and then either **Options** or **User Preferences**.

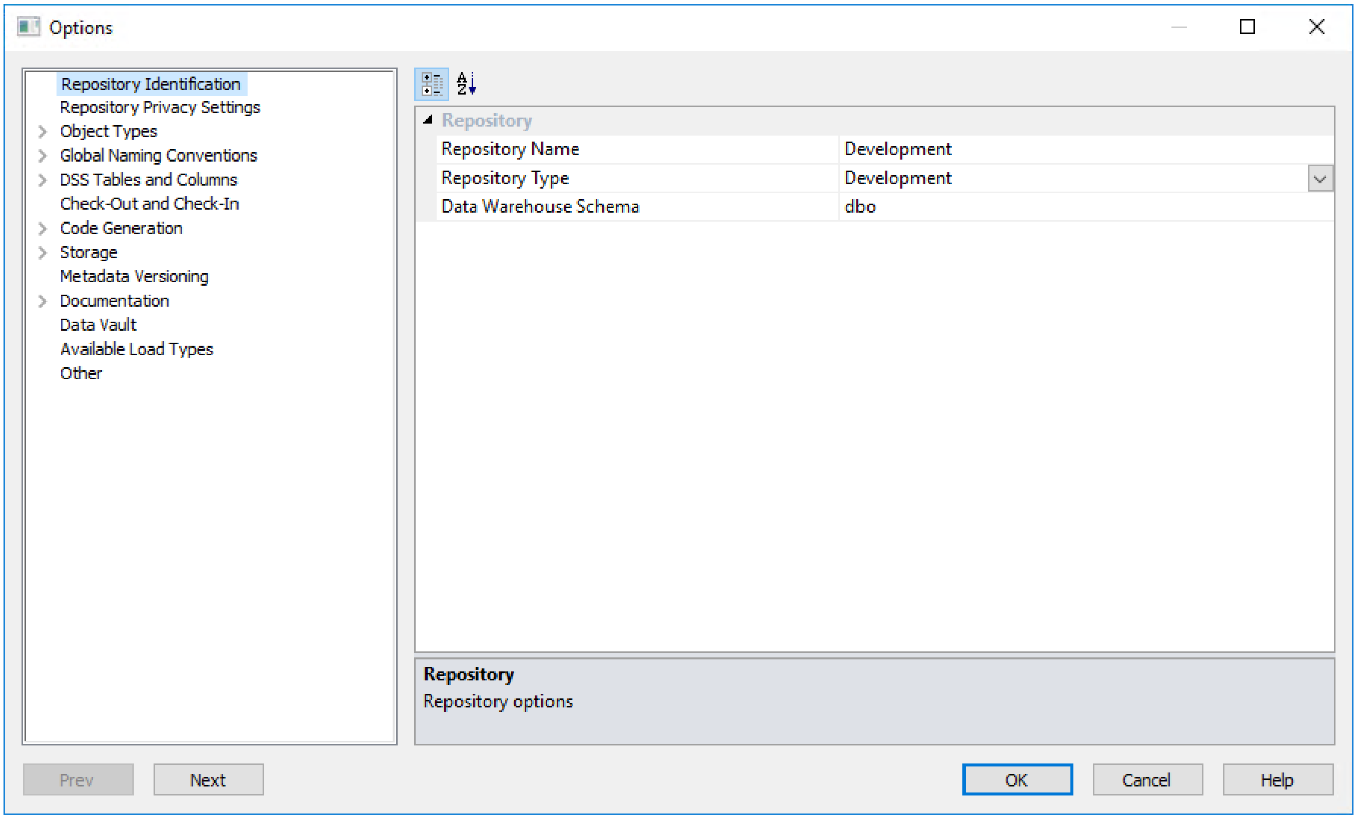
Settings - Options

Select **Options** from the **Tools** menu.



Repository Identification

This option enables users to set the **Repository Identification** settings.



Repository

Repository Name	Development
Repository Type	Development
Data Warehouse Schema	dbo

Repository
Repository options

Fields	Description
Repository Name	Set the **Name** for the repository. This name appears in the title bar in WhereScape RED . Restart WhereScape RED for repository name changes to take effect.
Repository Type	Set the **Type** for the repository. The repository type must reflect the environment. For

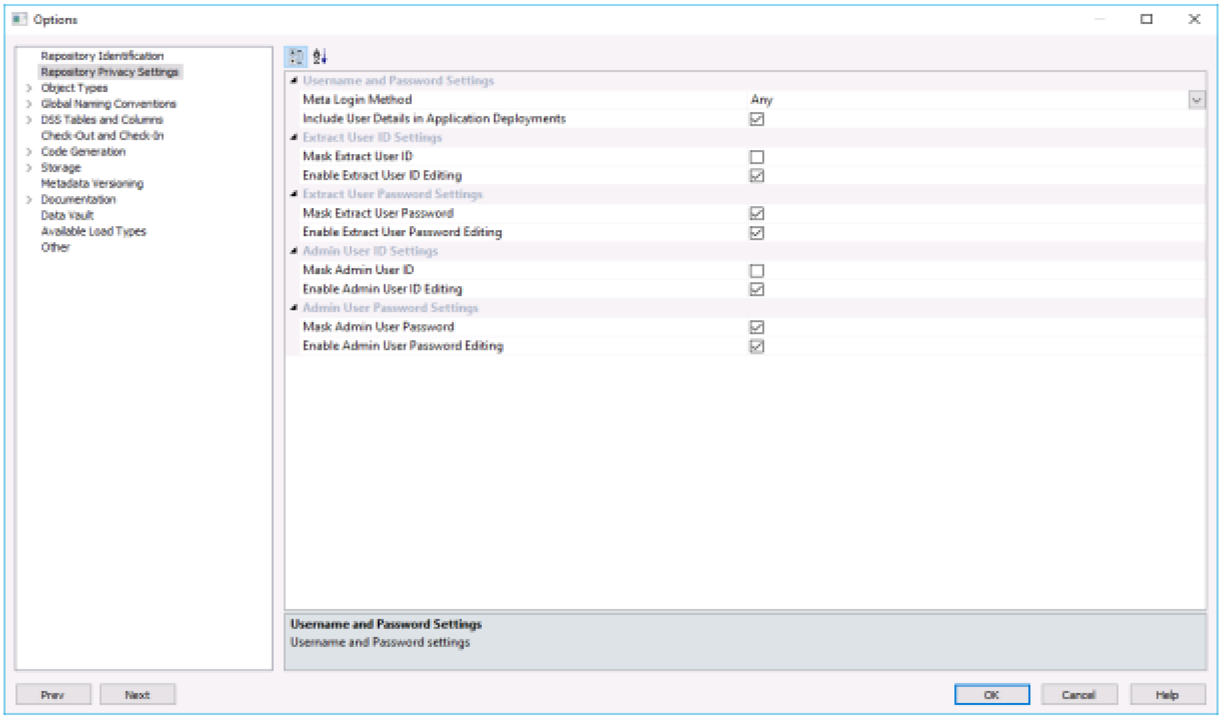
Fields	Description
	example, a 'Production' type must be chosen for the production environment.
Data Warehouse Schema	Set the **Schema** for the repository. The schema of the data warehouse is used by the WhereScape RED scheduler. You should not normally need to change this value.

Repository Privacy Settings

This option enables users to set the **Repository Privacy Settings**.

Warning

For UNIX/Linux scheduler processing, the **Encrypt** User and Password options cannot be used. Encrypt options are only supported when using a Windows scheduler.



Username and Password Settings

Meta Login Method	Any
Include User Details in Application Deployments	☑

Extract User ID Settings

Mask Extract User ID	☐
Enable Extract User ID Editing	☑

Extract User Password Settings

Mask Extract User Password	☑
Enable Extract User Password Editing	☑

Admin User ID Settings

Mask Admin User ID	☐
Enable Admin User ID Editing	☑

Admin User Password Settings

Mask Admin User Password	☑
Enable Admin User Password Editing	☑

Username and Password Settings
Username and Password settings

Changing Repository Settings

Since the repository privacy settings can be configured from the **Tools > Options** menu, for an environment to be secured, **a database administrator has to change the permissions on table ws_meta_admin table to read-only after the appropriate repository privacy change settings** in WhereScape RED have been made.

Note:

Changing this set of permissions to read-only is something which occurs outside of WhereScape RED and is dependent on the specific meta data database.

Username and Password Settings

Fields	Description
Meta Login Method	This option can be set to restrict users from using a particular login method for the meta repository.

Fields	Description
Include User Details in Application Deployments	Includes or excludes User Details in Application Deployment packages.

Extract User ID Settings

Fields	Description
Mask Extract User ID	Masks the input of the "Extract/Unix/Windows User ID" on the connection properties.
Enable Extract User ID Editing	Enables editing the "Extract/Unix/Windows User ID" via the connection properties.

Extract User ID Settings

Fields	Description
Mask Extract User Password	Masks the input of the "Extract/Unix/Windows User Password" on the connection properties.
Enable Extract User Password Editing	Enables editing "Extract/Unix/Windows User Password" via the connection properties.
Encrypt Extract User Password	Encrypts "Extract/Unix/Windows User Password" in the meta repository using WhereScape encryption.

Admin User ID Settings

Fields	Description
Mask Admin User ID	Masks the input of the "Admin/DSS User ID" on the connection properties.
Enable Admin User ID Editing	Enables editing the "Admin/DSS User ID" via the connection properties.

Admin User Password Settings

Fields	Description
Mask Admin User Password	Masks the input of the "Admin/DSS Password" on the connection properties.
Enable Admin Password Editing	Enables editing the "Admin/DSS Password" via the connection properties.

JDBC User ID Settings

Fields	Description
Mask JDBC User ID	Masks the input of the "JDBC User ID" on the connection properties.
Enable JDBC User ID Editing	Enables editing the "JDBC User ID" via the connection properties.

JDBC User Password Settings

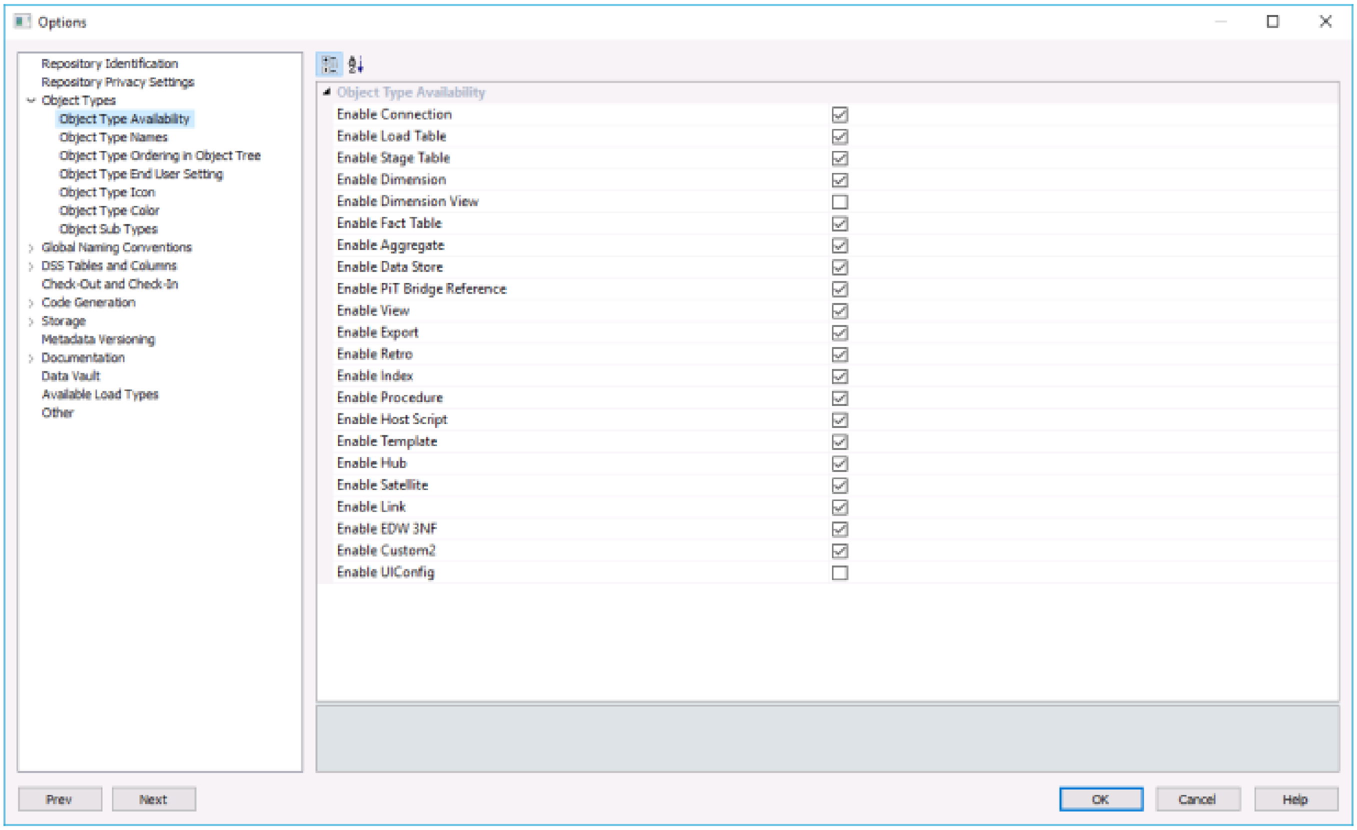
Fields	Description
Mask JDBC User Password	Masks the input of the "JDBC User Password" on the connection properties.
Enable JDBC User	Enables editing the "JDBC User Password" via the connection properties.

Fields	Description
Password Editing	

Object Types

Object Type Availability

This option enables you to activate or deactivate the various object types within the data warehouse repository. **Enable/Disable** object types in the data warehouse by selecting/clearing the availability check-boxes for each object type. All settings are enabled by default, except for the **Enable Cube** and **Enable Virtual Cube** options which are soon to be deprecated.



Object Type Availability

Enable Connection	☑
Enable Load Table	☑
Enable Stage Table	☑
Enable Dimension	☑
Enable Dimension View	□
Enable Fact Table	☑
Enable Aggregate	☑
Enable Data Store	☑
Enable PiT Bridge Reference	☑
Enable View	☑
Enable Export	☑
Enable Retro	☑
Enable Index	☑
Enable Procedure	☑
Enable Host Script	☑
Enable Template	☑
Enable Hub	☑
Enable Satellite	☑
Enable Link	☑
Enable EDW 3NF	☑
Enable Custom2	☑
Enable UIConfig	□

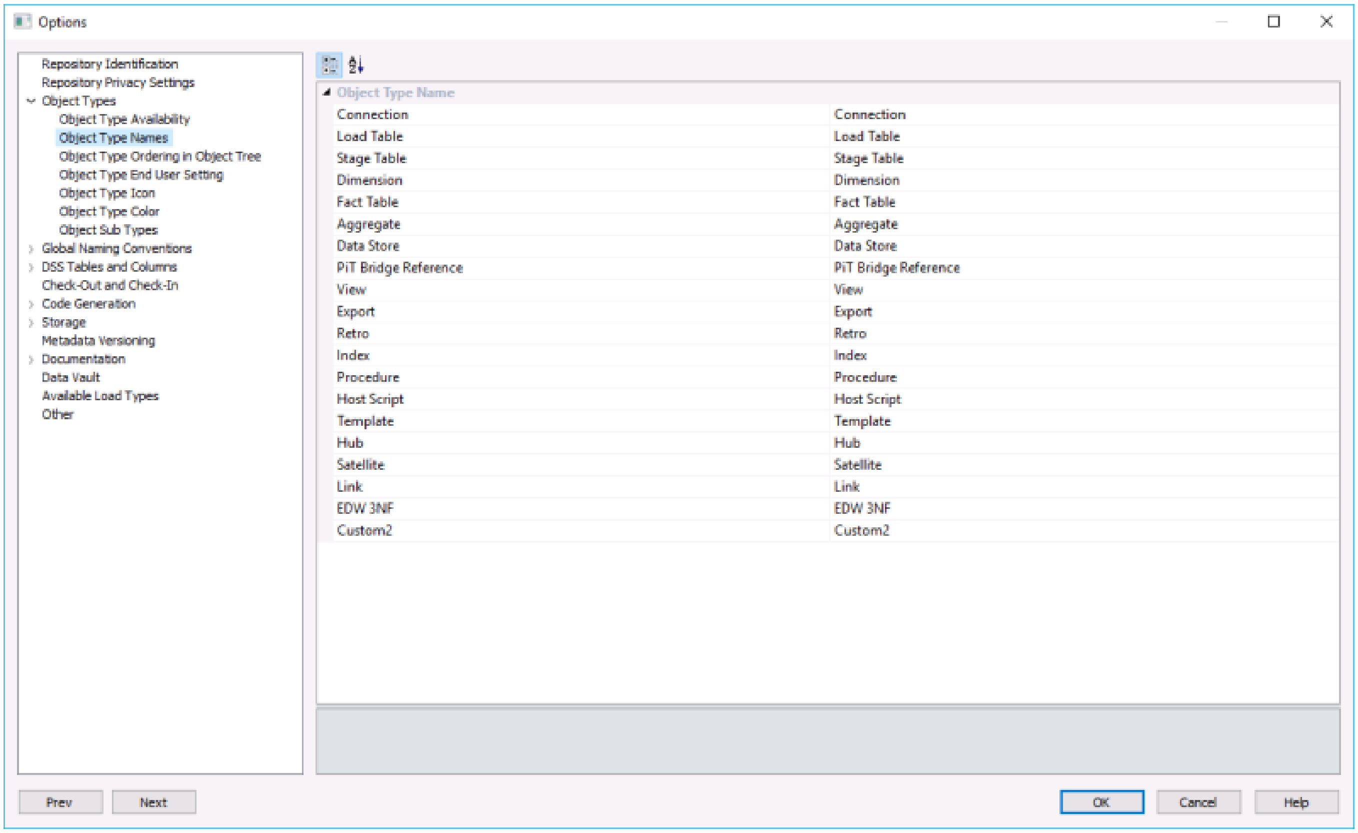
Object Type Names

This option enables you to set the **names** for the various object types.

Set the **Name** for each object type.

Note:

Data Vault Repository Types: Users with Data Vault model type licenses that select a **Data Vault repository type** while creating the RED metadata repository will have appropriate Data Vault repository default settings, such as Object Type Names, Global Naming of Tables, Indexes, Key Columns and Procedures/Scripts; as well as other repository settings/user preferences. These repository types will have their default Object Type Names of Normalized and Data Store objects set to Hub/Link and Satellite.

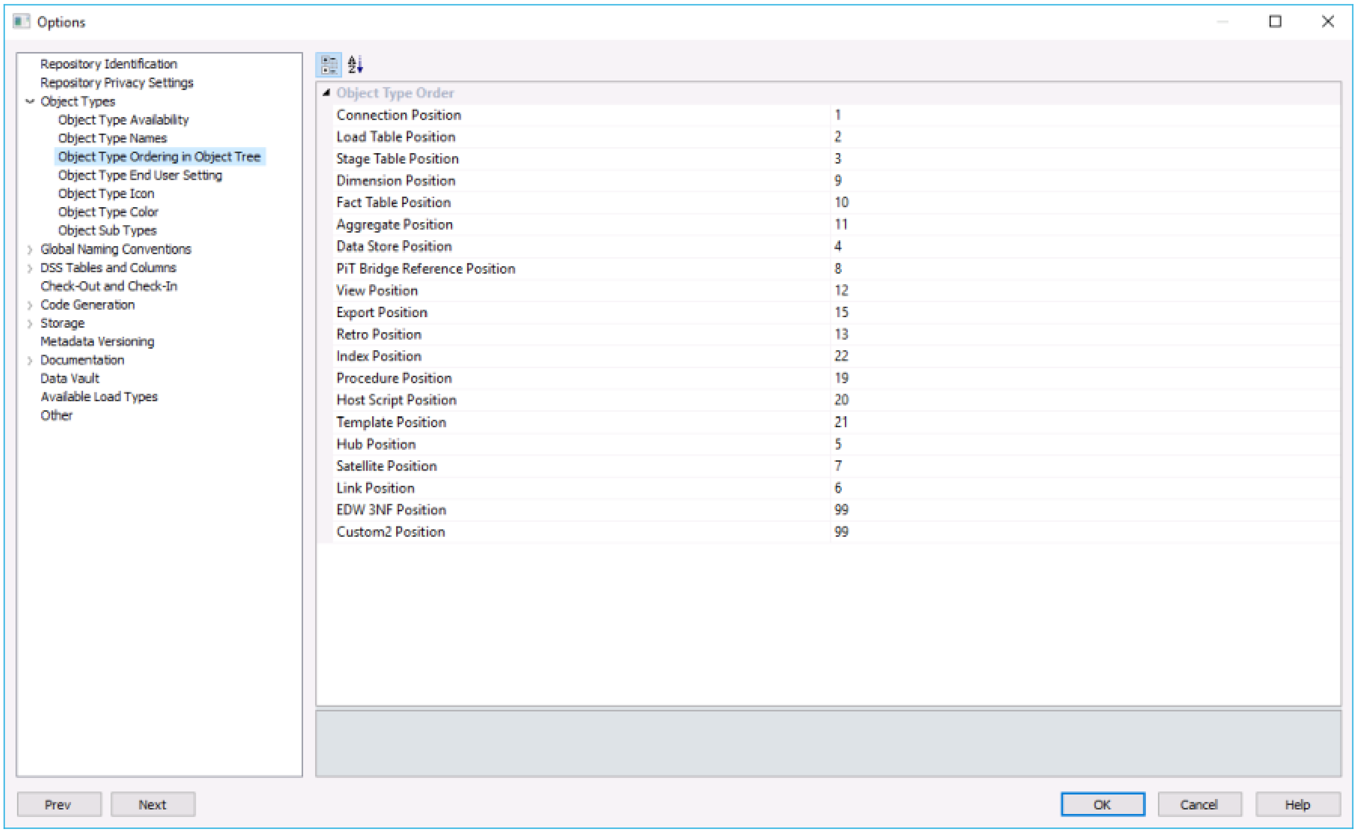


Object Type Name	
Connection	Connection
Load Table	Load Table
Stage Table	Stage Table
Dimension	Dimension
Fact Table	Fact Table
Aggregate	Aggregate
Data Store	Data Store
PiT Bridge Reference	PiT Bridge Reference
View	View
Export	Export
Retro	Retro
Index	Index
Procedure	Procedure
Host Script	Host Script
Template	Template
Hub	Hub
Satellite	Satellite
Link	Link
EDW 3NF	EDW 3NF
Custom2	Custom2

Object Type Ordering

This option enables you to set the **ordering** in which the object types appear in the object tree.

Set the **Ordering** of the object types as displayed in the object tree pane.



Object Type Order

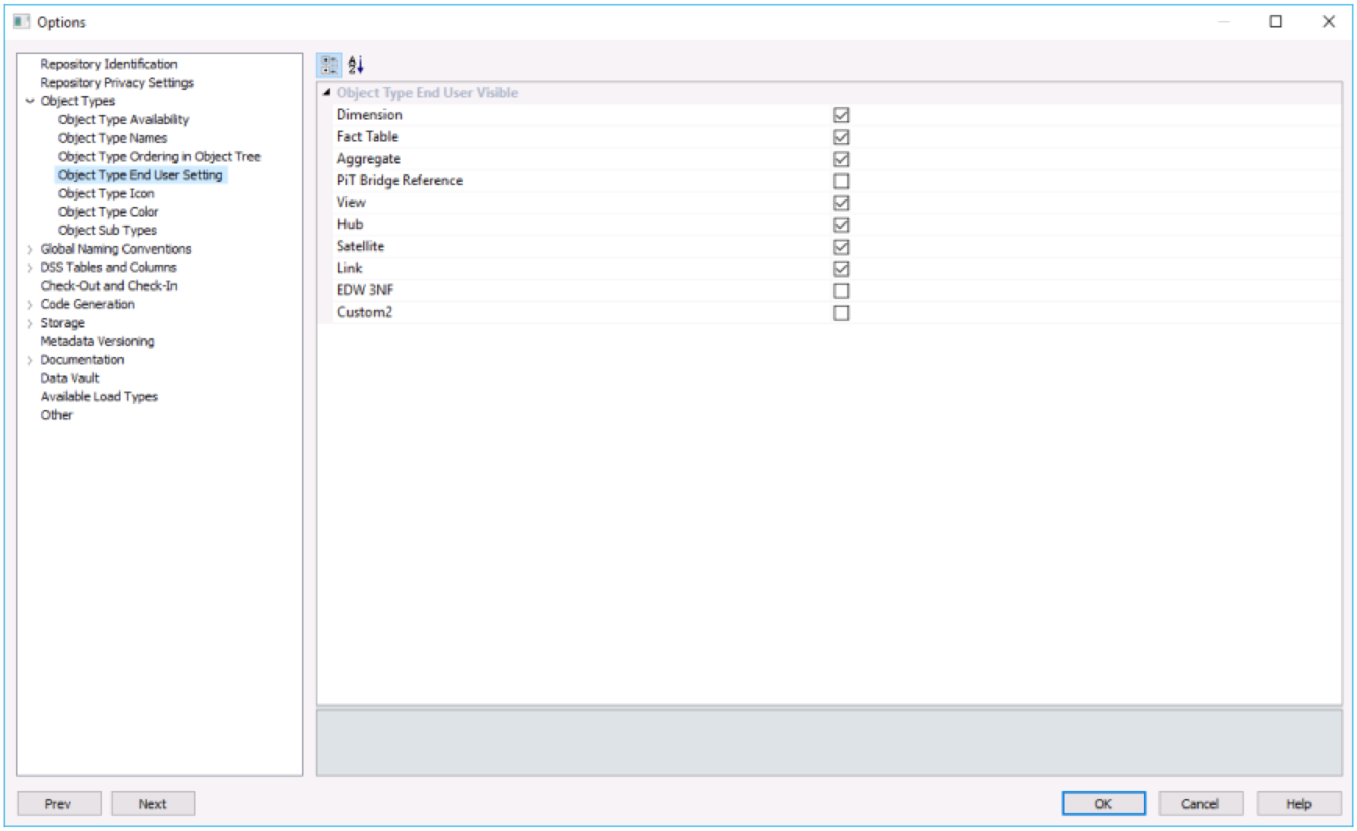
Connection Position	1
Load Table Position	2
Stage Table Position	3
Dimension Position	9
Fact Table Position	10
Aggregate Position	11
Data Store Position	4
PiT Bridge Reference Position	8
View Position	12
Export Position	15
Retro Position	13
Index Position	22
Procedure Position	19
Host Script Position	20
Template Position	21
Hub Position	5
Satellite Position	7
Link Position	6
EDW 3NF Position	99
Custom2 Position	99

Object Type End User Setting

This option enables you to set the Object types as **end user objects**.

All the options in this screen are selected by default.

Clear an object type check box to remove it as an end user object, else leave selected.



Object Type End User Visible

Dimension	☑
Fact Table	☑
Aggregate	☑
PiT Bridge Reference	□
View	☑
Hub	☑
Satellite	☑
Link	☑
EDW 3NF	□
Custom2	□

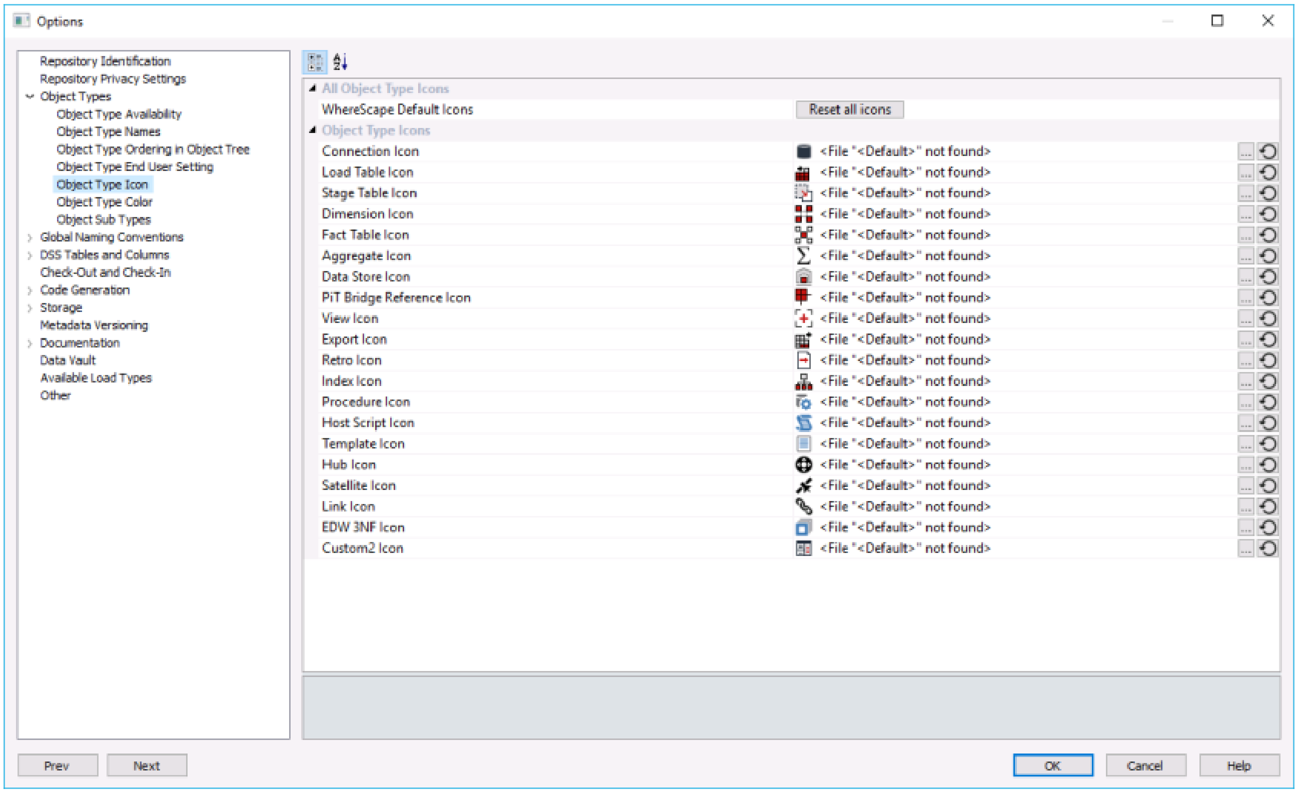
Object Type Icon

This option enables you to configure the **Icons** for all Object Types. To configure custom Object Type icons:

1. Create an 'Icons' folder in the WhereScape RED install directory, if it doesn't exist. For example:

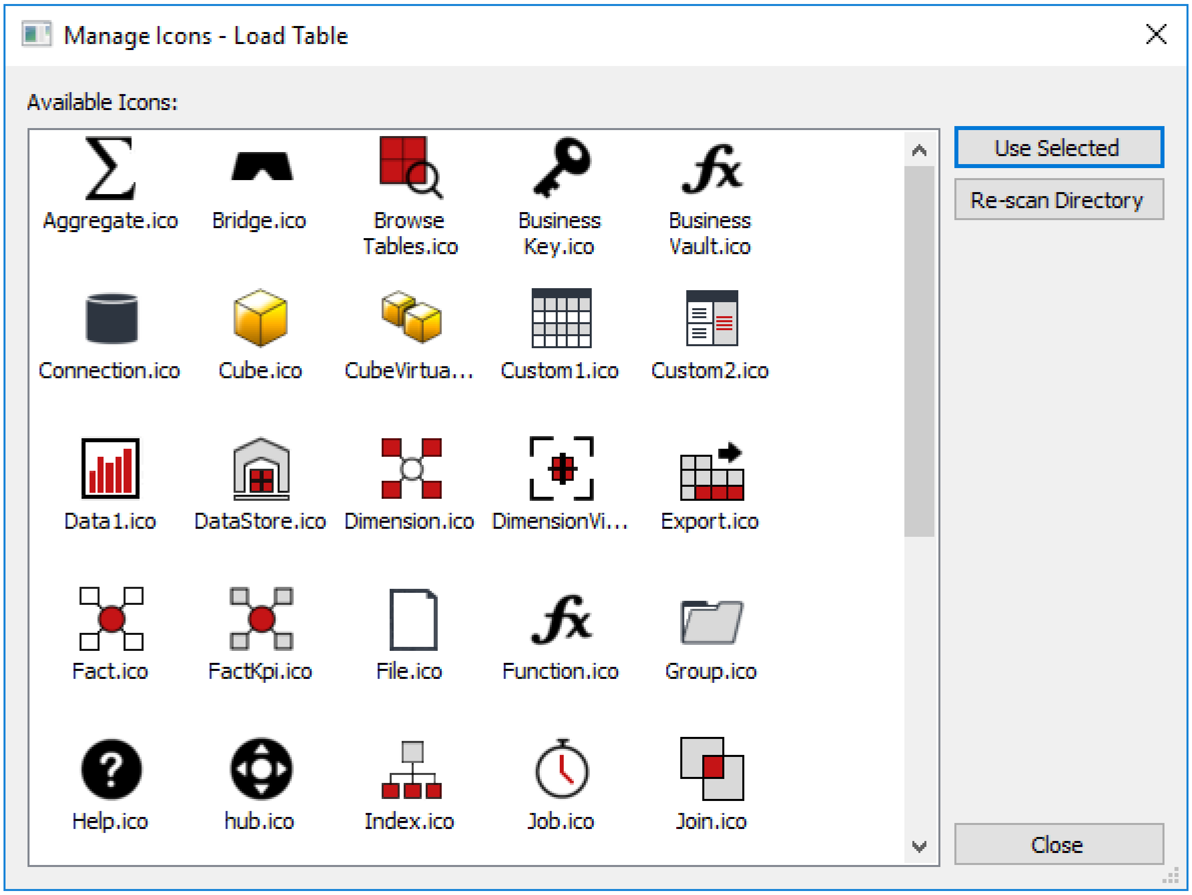
   C:\Program Files (x86)\WhereScape\Icons

2. Place custom '**.ico**' files in the Icons folder.
3. In RED, select **Options > Object Types > Object Type Icon**.

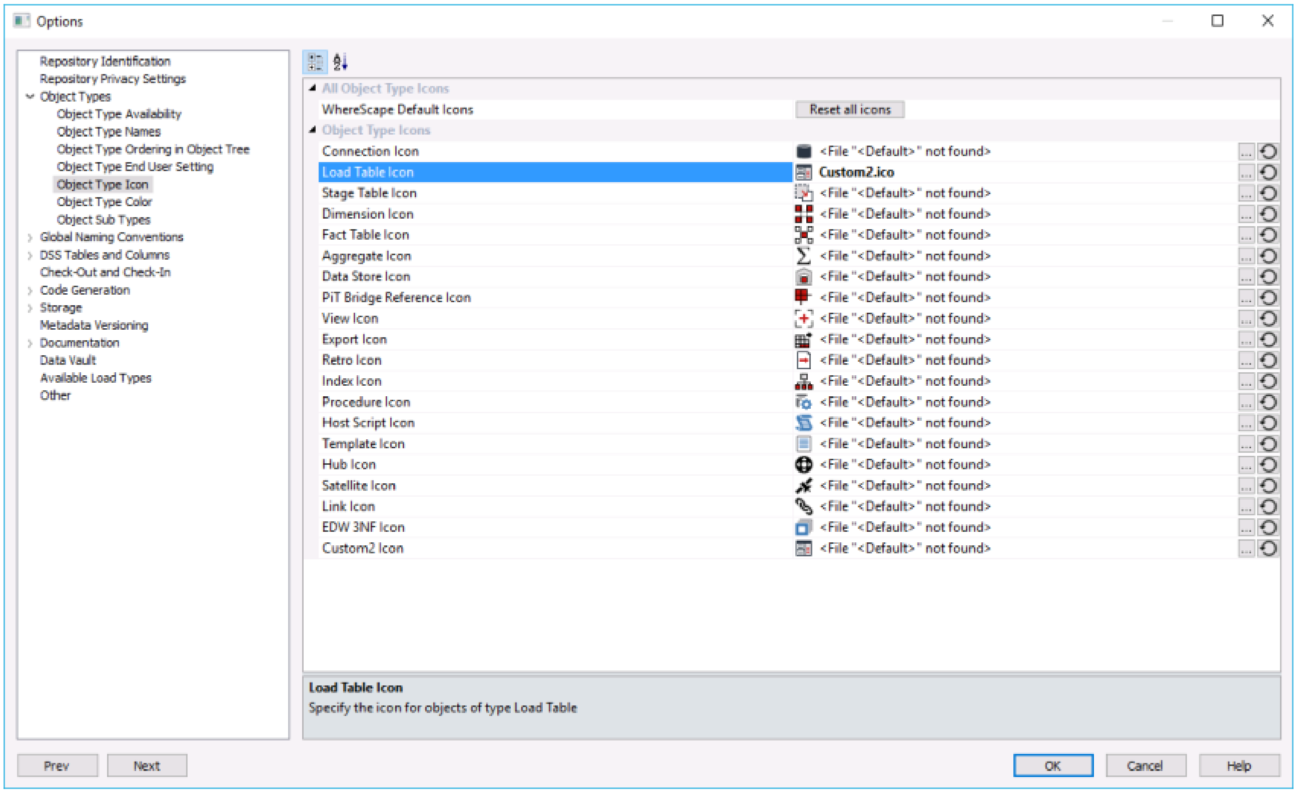


All Object Type Icons	
WhereScape Default Icons	Reset all icons
Object Type Icons	
Connection Icon	<File "<Default>" not found>
Load Table Icon	<File "<Default>" not found>
Stage Table Icon	<File "<Default>" not found>
Dimension Icon	<File "<Default>" not found>
Fact Table Icon	<File "<Default>" not found>
Aggregate Icon	<File "<Default>" not found>
Data Store Icon	<File "<Default>" not found>
PiT Bridge Reference Icon	<File "<Default>" not found>
View Icon	<File "<Default>" not found>
Export Icon	<File "<Default>" not found>
Retro Icon	<File "<Default>" not found>
Index Icon	<File "<Default>" not found>
Procedure Icon	<File "<Default>" not found>
Host Script Icon	<File "<Default>" not found>
Template Icon	<File "<Default>" not found>
Hub Icon	<File "<Default>" not found>
Satellite Icon	<File "<Default>" not found>
Link Icon	<File "<Default>" not found>
EDW 3NF Icon	<File "<Default>" not found>
Custom2 Icon	<File "<Default>" not found>

4. Click the ellipses button **'...'** next to each Object Type and select the desired icon.



5. All configured icons are displayed as file names in the screen. To save changes, click **OK**.



Options

- Repository Identification
- Repository Privacy Settings
- Object Types
  - Object Type Availability
  - Object Type Names
  - Object Type Ordering in Object Tree
  - Object Type End User Setting
  - Object Type Icon
  - Object Type Color
  - Object Sub Types
- Global Naming Conventions
- DSS Tables and Columns
- Check-Out and Check-In
- Code Generation
- Storage
- Metadata Versioning
- Documentation
- Data Vault
- Available Load Types
- Other

All Object Type Icons	
WhereScape Default Icons	Reset all icons
Object Type Icons	
Connection Icon	<File "<Default>" not found>
Load Table Icon	Custom2.ico
Stage Table Icon	<File "<Default>" not found>
Dimension Icon	<File "<Default>" not found>
Fact Table Icon	<File "<Default>" not found>
Aggregate Icon	<File "<Default>" not found>
Data Store Icon	<File "<Default>" not found>
PiT Bridge Reference Icon	<File "<Default>" not found>
View Icon	<File "<Default>" not found>
Export Icon	<File "<Default>" not found>
Retro Icon	<File "<Default>" not found>
Index Icon	<File "<Default>" not found>
Procedure Icon	<File "<Default>" not found>
Host Script Icon	<File "<Default>" not found>
Template Icon	<File "<Default>" not found>
Hub Icon	<File "<Default>" not found>
Satellite Icon	<File "<Default>" not found>
Link Icon	<File "<Default>" not found>
EDW 3NF Icon	<File "<Default>" not found>
Custom2 Icon	<File "<Default>" not found>

Load Table Icon

Specify the icon for objects of type Load Table

To Reset An Icon

To reset an icon to default, click the reset button ⟲ next to the icon.

Reset All Icons

To reset all icons to default, click the **Reset All Icons** button at the top of the dialog.

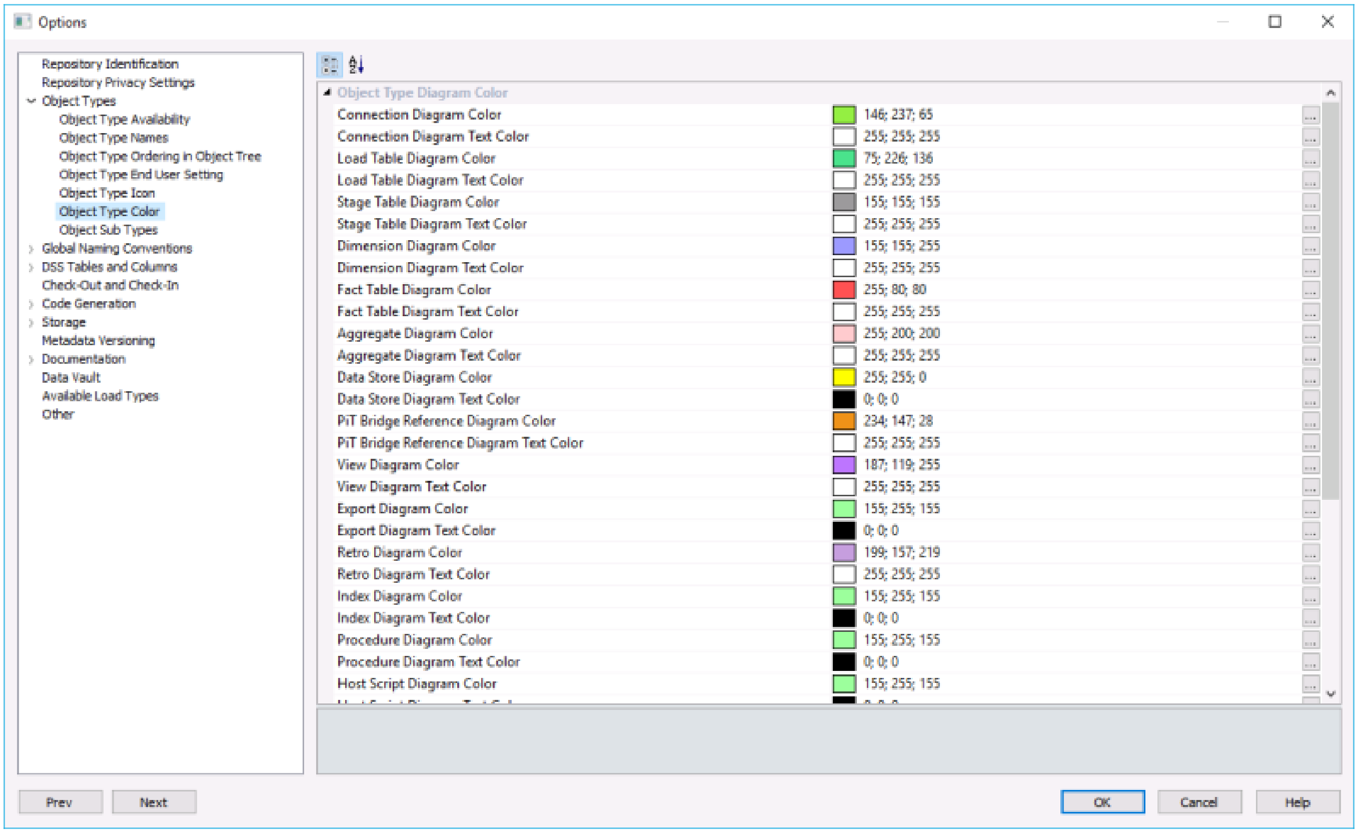
Note:

All installations must have a copy of the icon directory.

Object Type Color

This option enables you to set the **diagram colors** for each object type.

Set the **Diagram Color** for each object type.



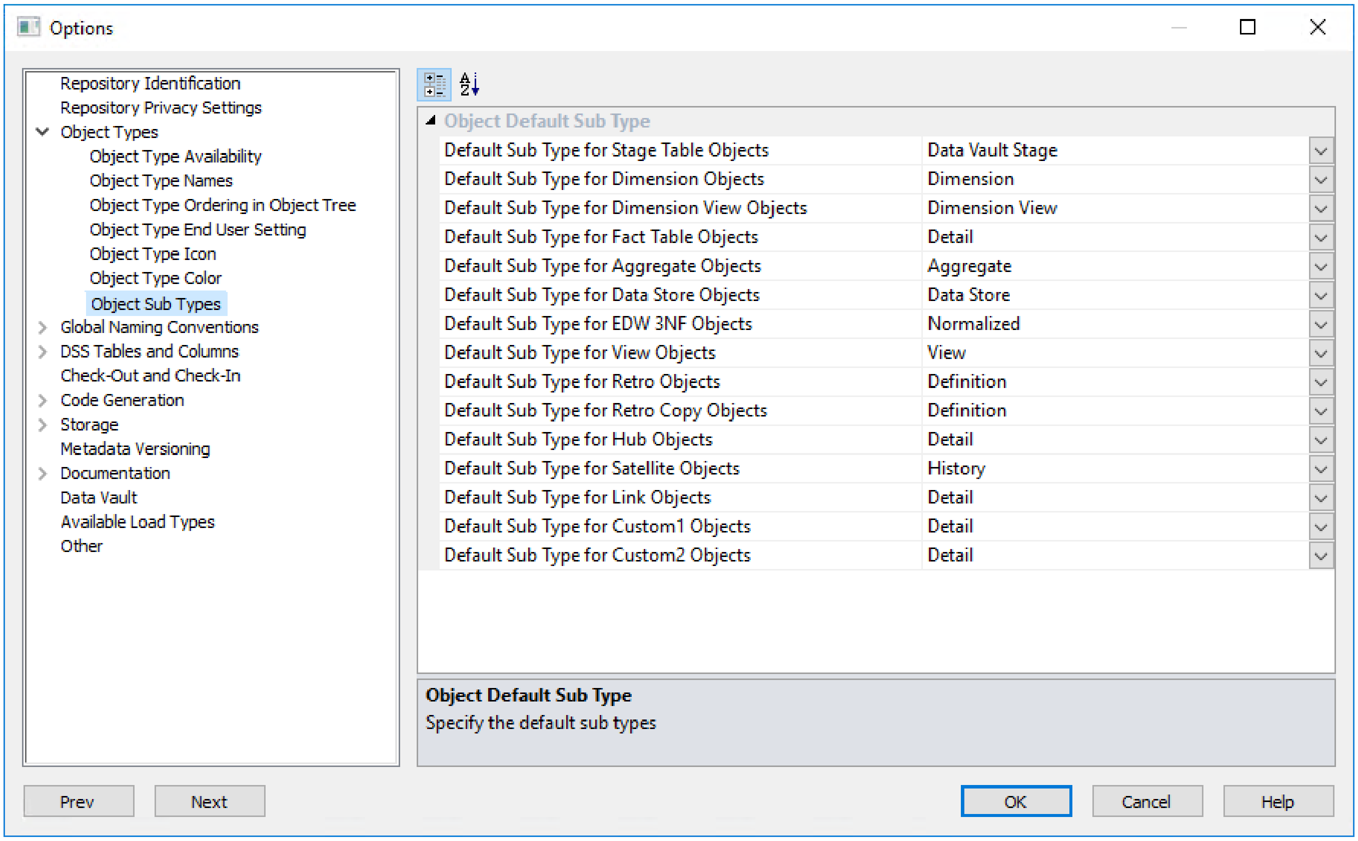
- Repository Identification
- Repository Privacy Settings
- Object Types
  - Object Type Availability
  - Object Type Names
  - Object Type Ordering in Object Tree
  - Object Type End User Setting
  - Object Type Icon
  - Object Type Color
  - Object Sub Types
- Global Naming Conventions
- DSS Tables and Columns
- Check-Out and Check-In
- Code Generation
- Storage
- Metadata Versioning
- Documentation
- Data Vault
- Available Load Types
- Other

Object Type Diagram Color

Connection Diagram Color	146; 237; 65
Connection Diagram Text Color	255; 255; 255
Load Table Diagram Color	75; 226; 136
Load Table Diagram Text Color	255; 255; 255
Stage Table Diagram Color	155; 155; 155
Stage Table Diagram Text Color	255; 255; 255
Dimension Diagram Color	155; 155; 255
Dimension Diagram Text Color	255; 255; 255
Fact Table Diagram Color	255; 80; 80
Fact Table Diagram Text Color	255; 255; 255
Aggregate Diagram Color	255; 200; 200
Aggregate Diagram Text Color	255; 255; 255
Data Store Diagram Color	255; 255; 0
Data Store Diagram Text Color	0; 0; 0
PiT Bridge Reference Diagram Color	234; 147; 28
PiT Bridge Reference Diagram Text Color	255; 255; 255
View Diagram Color	187; 119; 255
View Diagram Text Color	255; 255; 255
Export Diagram Color	155; 255; 155
Export Diagram Text Color	0; 0; 0
Retro Diagram Color	199; 157; 219
Retro Diagram Text Color	255; 255; 255
Index Diagram Color	155; 255; 155
Index Diagram Text Color	0; 0; 0
Procedure Diagram Color	155; 255; 155
Procedure Diagram Text Color	0; 0; 0
Host Script Diagram Color	155; 255; 155

Object Sub Types

This option enables you to set the **Default Sub Type** for enabled object types in RED. Select the desired default sub-types from the Object's drop-down lists.



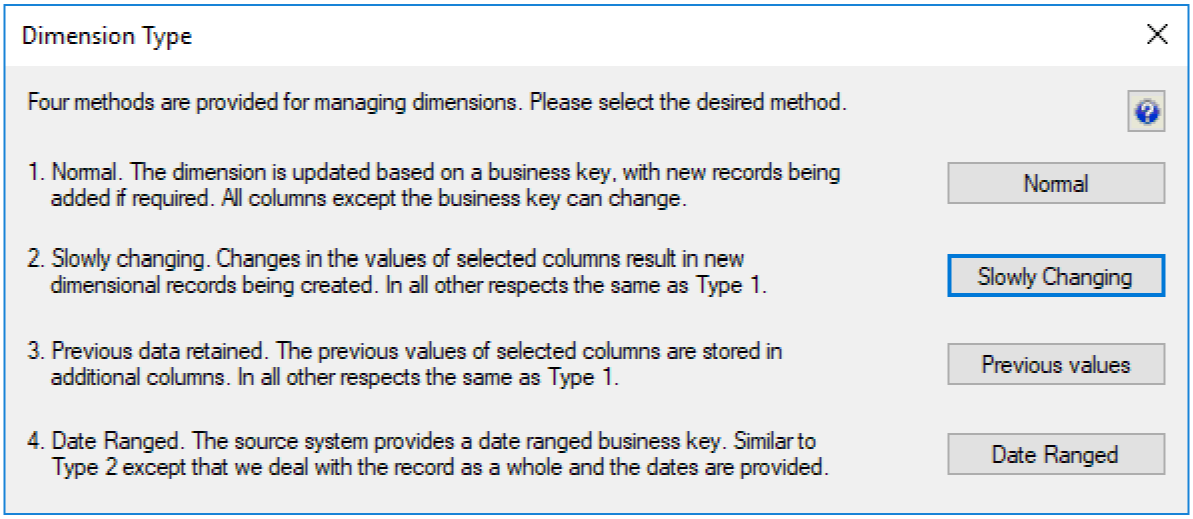
Object Default Sub Type

Default Sub Type for Stage Table Objects	Data Vault Stage
Default Sub Type for Dimension Objects	Dimension
Default Sub Type for Dimension View Objects	Dimension View
Default Sub Type for Fact Table Objects	Detail
Default Sub Type for Aggregate Objects	Aggregate
Default Sub Type for Data Store Objects	Data Store
Default Sub Type for EDW 3NF Objects	Normalized
Default Sub Type for View Objects	View
Default Sub Type for Retro Objects	Definition
Default Sub Type for Retro Copy Objects	Definition
Default Sub Type for Hub Objects	Detail
Default Sub Type for Satellite Objects	History
Default Sub Type for Link Objects	Detail
Default Sub Type for Custom1 Objects	Detail
Default Sub Type for Custom2 Objects	Detail

Object Default Sub Type
Specify the default sub types

For example, to have **Dimension** objects created in RED as **Changing Dimensions** at the time of drag and drop, select the **Changing Dimension** option in the **Default Sub Type for Dimension Objects**.

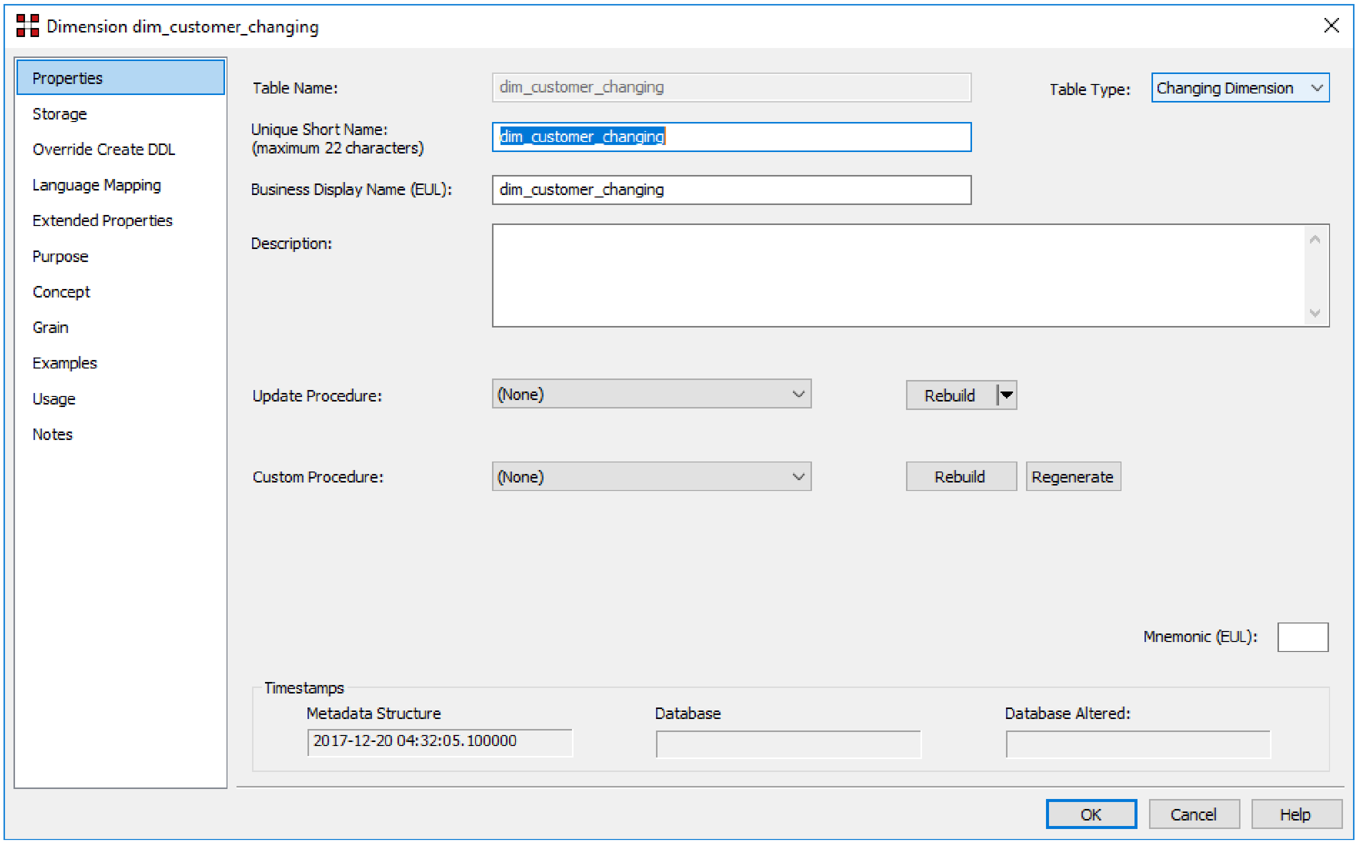
After the table is dragged and dropped, users can simply hit enter to proceed on the Dimension Type where the **Slowly Changing** type is already the default sub type option, as previously selected in **Tool > Options** menu.



Four methods are provided for managing dimensions. Please select the desired method.

1. Normal. The dimension is updated based on a business key, with new records being added if required. All columns except the business key can change.
2. Slowly changing. Changes in the values of selected columns result in new dimensional records being created. In all other respects the same as Type 1.
3. Previous data retained. The previous values of selected columns are stored in additional columns. In all other respects the same as Type 1.
4. Date Ranged. The source system provides a date ranged business key. Similar to Type 2 except that we deal with the record as a whole and the dates are provided.

The **Dimension Properties** screen also reflects the selected table sub type on the **Table Type** drop-down list.



Dimension dim_customer_changing

Table Name: dim_customer_changing

Table Type: Changing Dimension

Unique Short Name: (maximum 22 characters) dim_customer_changing

Business Display Name (EUL): dim_customer_changing

Description:

Update Procedure: (None)

Custom Procedure: (None)

Mnemonic (EUL):

Timestamps

Metadata Structure: 2017-12-20 04:32:05.100000

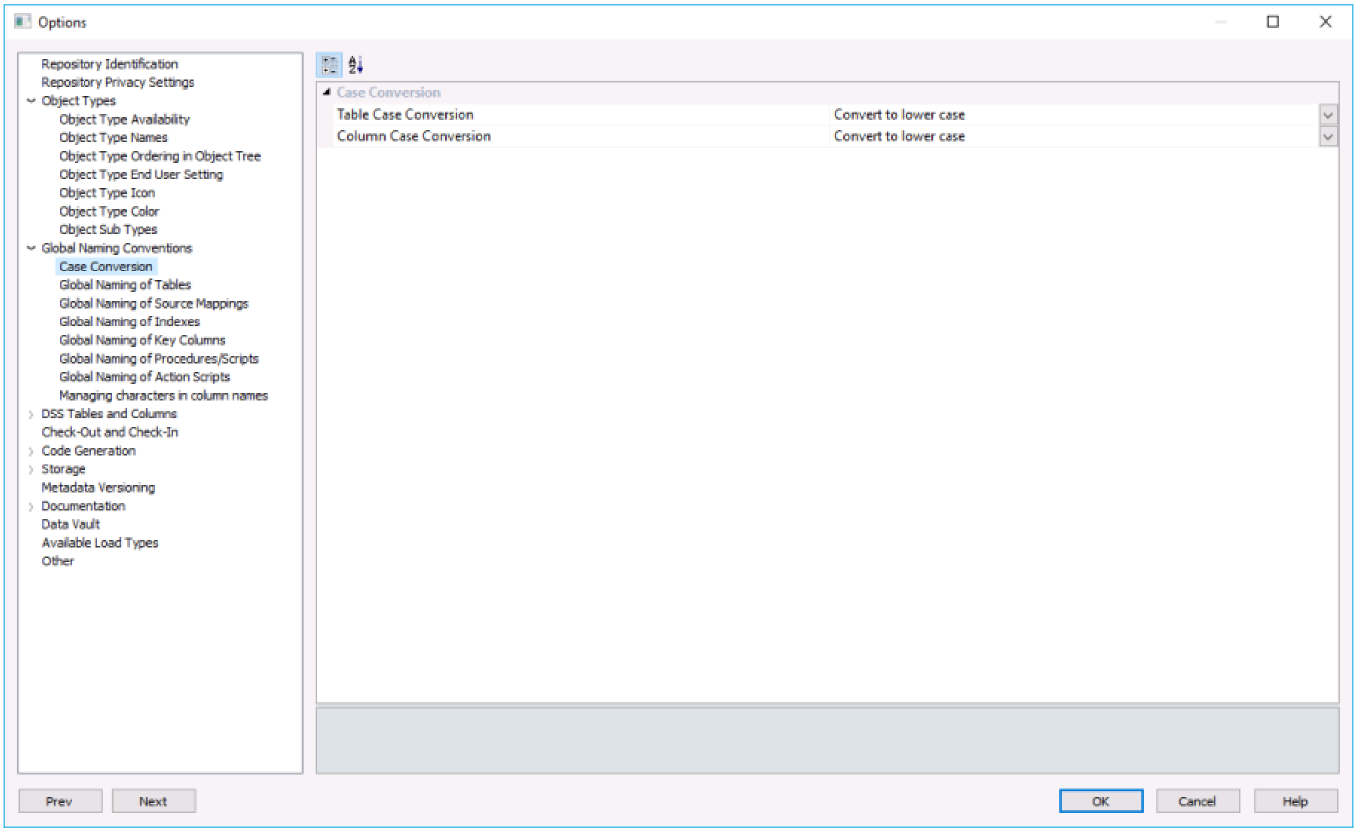
Database:

Database Altered:

Global Naming Conventions

Case Conversion

This option enables you to set the **Case Conversion** methods for the tables and columns in RED.



Case Conversion

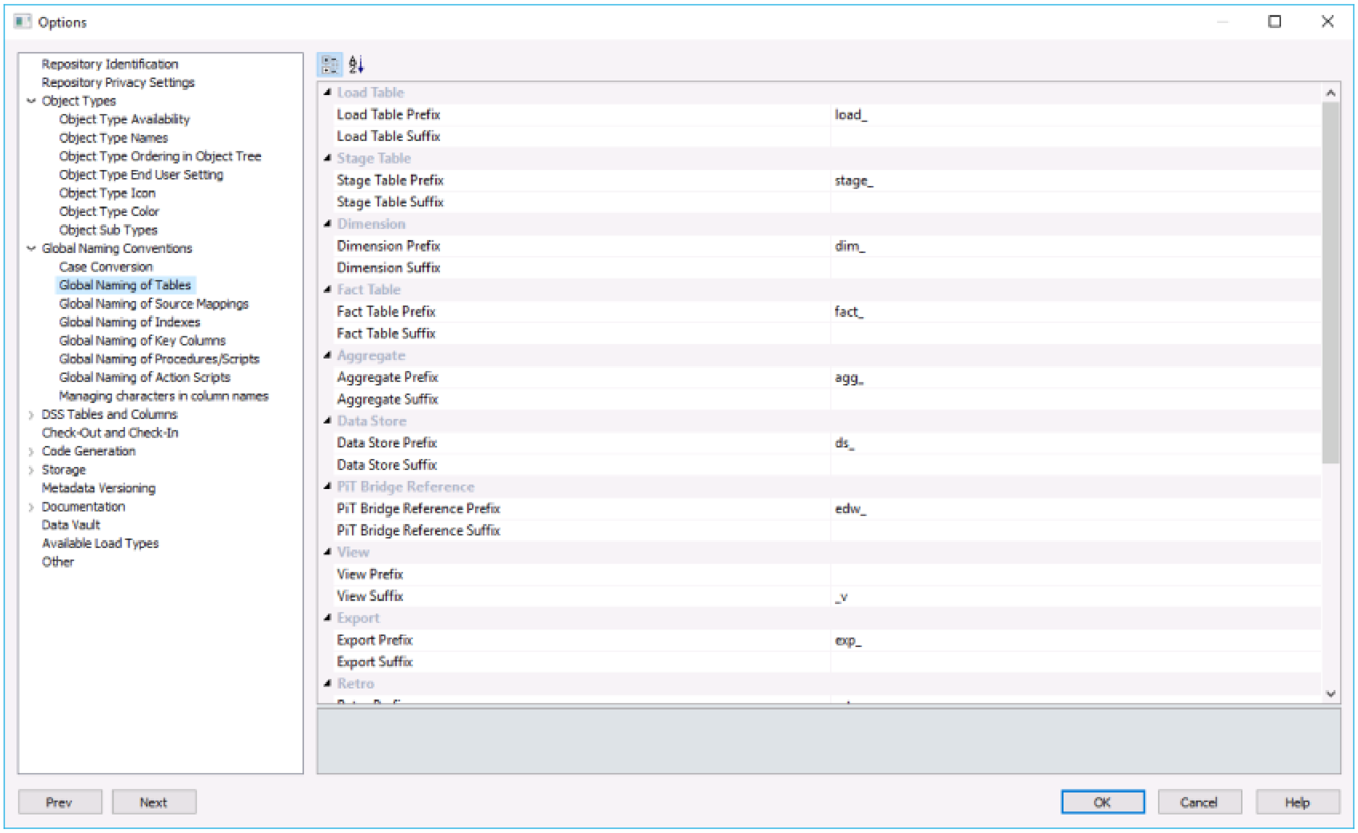
Table Case Conversion	Convert to lower case
Column Case Conversion	Convert to lower case

Case Conversion

Set the **Table Case Conversion** method and the **Column Case Conversion** method from the drop-down lists.

Global Naming of Tables

This option enables you to set the **Global Naming of Tables** options.



Load Table	
Load Table Prefix	load_
Load Table Suffix	
Stage Table	
Stage Table Prefix	stage_
Stage Table Suffix	
Dimension	
Dimension Prefix	dim_
Dimension Suffix	
Fact Table	
Fact Table Prefix	fact_
Fact Table Suffix	
Aggregate	
Aggregate Prefix	agg_
Aggregate Suffix	
Data Store	
Data Store Prefix	ds_
Data Store Suffix	
PiT Bridge Reference	
PiT Bridge Reference Prefix	edw_
PiT Bridge Reference Suffix	
View	
View Prefix	
View Suffix	_v
Export	
Export Prefix	exp_
Export Suffix	
Retro	

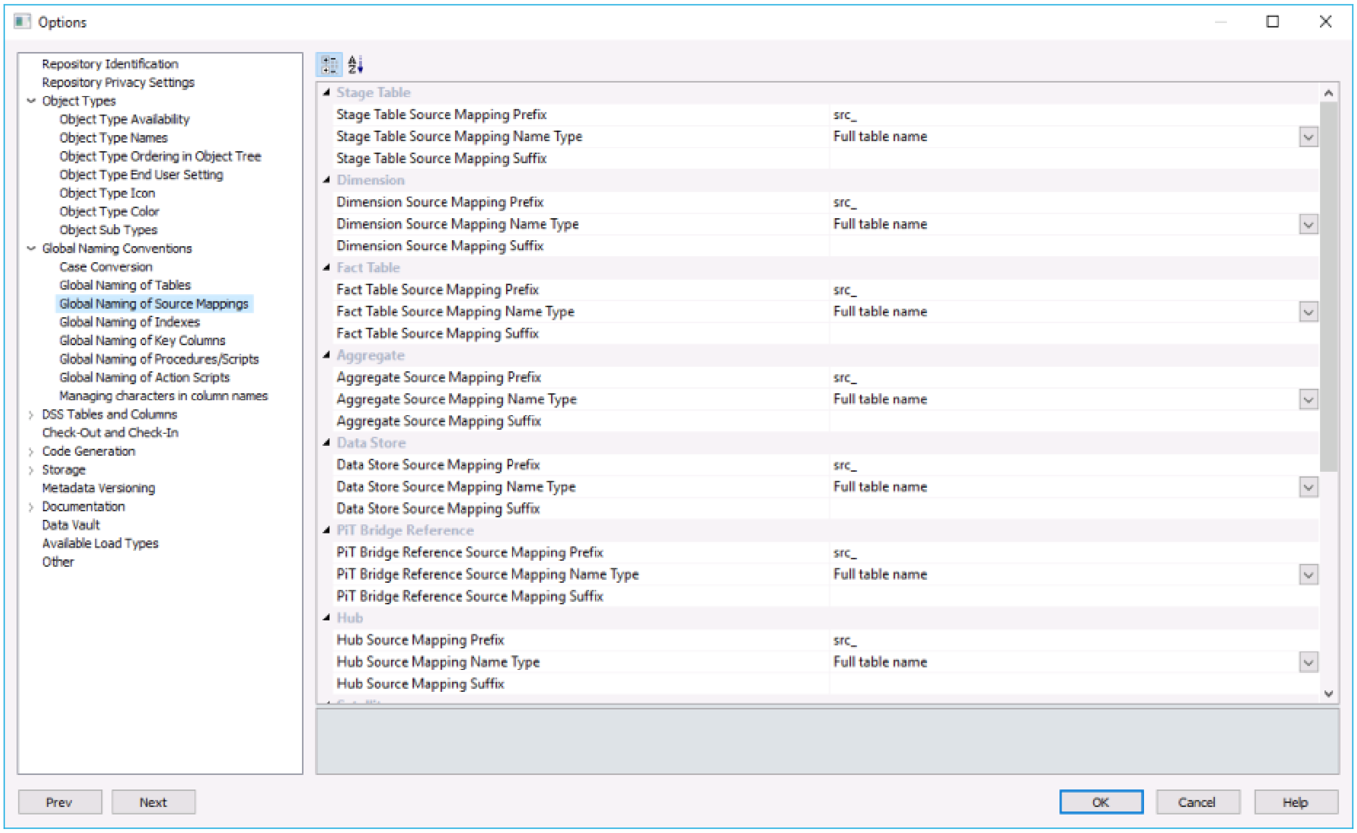
From the example screen above, if a source table called **customer** (with a short name of 'customer') was dragged into a Load table drop target then the default name would be **load_customer**.

The object name defaults shown above are the values that are installed with the base metadata.

They can be changed at any stage; however, the change does not affect any existing objects. Therefore, if a new naming regime is chosen, any existing objects need to be renamed through the **Properties** screen of the object.

Global Naming of Source Mappings

This option enables you to set the **Global Naming of Source Mappings** options.



Stage Table	
Stage Table Source Mapping Prefix	src_
Stage Table Source Mapping Name Type	Full table name
Stage Table Source Mapping Suffix	
Dimension	
Dimension Source Mapping Prefix	src_
Dimension Source Mapping Name Type	Full table name
Dimension Source Mapping Suffix	
Fact Table	
Fact Table Source Mapping Prefix	src_
Fact Table Source Mapping Name Type	Full table name
Fact Table Source Mapping Suffix	
Aggregate	
Aggregate Source Mapping Prefix	src_
Aggregate Source Mapping Name Type	Full table name
Aggregate Source Mapping Suffix	
Data Store	
Data Store Source Mapping Prefix	src_
Data Store Source Mapping Name Type	Full table name
Data Store Source Mapping Suffix	
PiT Bridge Reference	
PiT Bridge Reference Source Mapping Prefix	src_
PiT Bridge Reference Source Mapping Name Type	Full table name
PiT Bridge Reference Source Mapping Suffix	
Hub	
Hub Source Mapping Prefix	src_
Hub Source Mapping Name Type	Full table name
Hub Source Mapping Suffix	

Similar to table objects created in RED, a prefix and/or a suffix string can also be applied to source mapping child objects.

From the example screen above, if a source table was dragged into a table drop target called **stage_customer** (with a full table name of 'customer') then the default name for the first source mapping object created would be **src_stage_customer_1**.

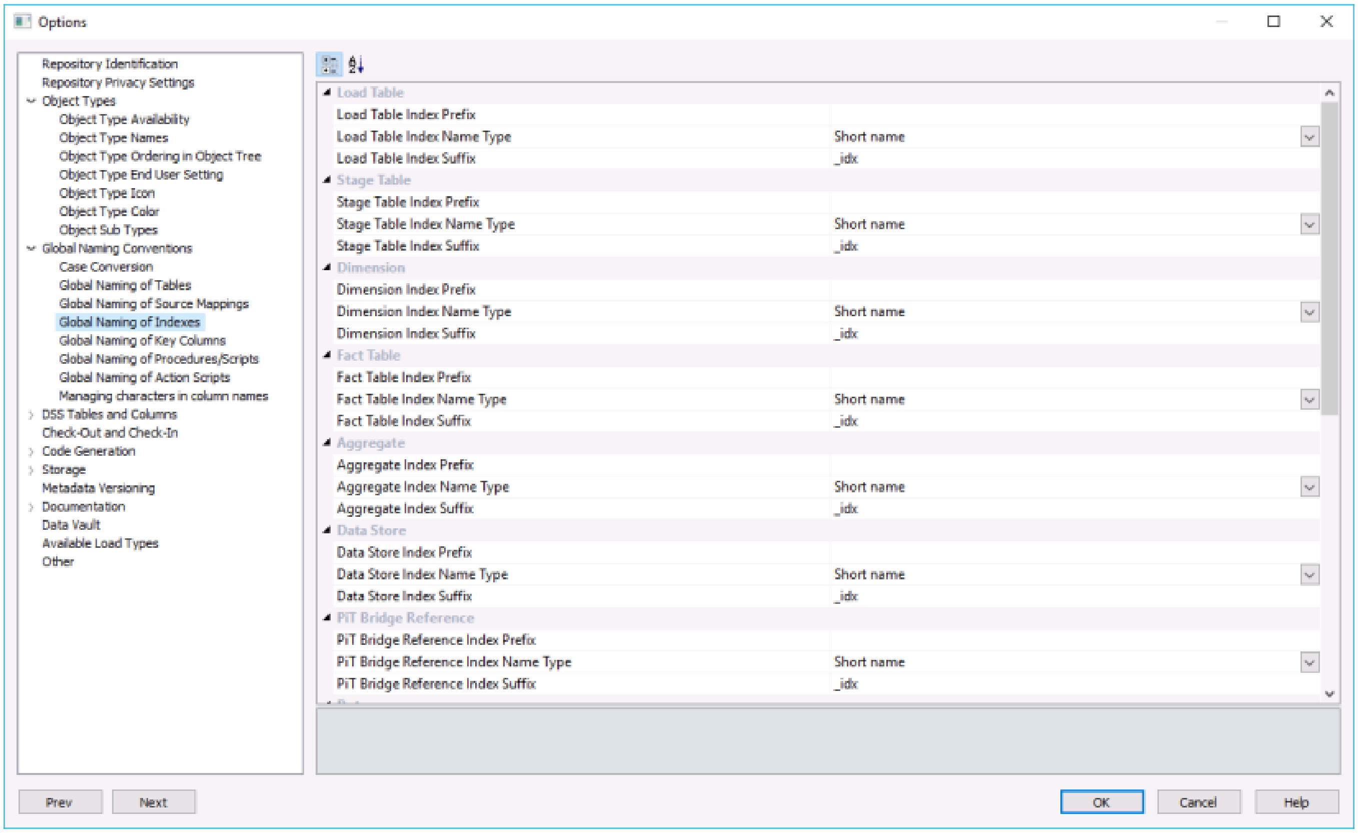
The same applies for the second source mapping object created—the object would be named **src_stage_customer_2**.

The object name defaults shown above are the values that are installed with the base metadata.

They can be changed at any stage; however, the change does not affect any existing objects. Therefore, if a new naming regime is chosen, any existing objects has to be renamed through the **Properties** screen of the object.

Global Naming of Indexes

This option enables you to set the **Global Naming of Indexes** options.



Load Table	
Load Table Index Prefix	
Load Table Index Name Type	Short name
Load Table Index Suffix	_idx
Stage Table	
Stage Table Index Prefix	
Stage Table Index Name Type	Short name
Stage Table Index Suffix	_idx
Dimension	
Dimension Index Prefix	
Dimension Index Name Type	Short name
Dimension Index Suffix	_idx
Fact Table	
Fact Table Index Prefix	
Fact Table Index Name Type	Short name
Fact Table Index Suffix	_idx
Aggregate	
Aggregate Index Prefix	
Aggregate Index Name Type	Short name
Aggregate Index Suffix	_idx
Data Store	
Data Store Index Prefix	
Data Store Index Name Type	Short name
Data Store Index Suffix	_idx
PiT Bridge Reference	
PiT Bridge Reference Index Prefix	
PiT Bridge Reference Index Name Type	Short name
PiT Bridge Reference Index Suffix	_idx

Whenever a new procedure is defined, WhereScape RED builds or rebuilds a standard set of indexes for the table. These indexes are created using the standard defined. As with the key naming, we can set either a pre-fix or a suffix value, or in fact both, as well as choosing the use of either the table name or the short name associated with the table.

In addition to the naming specifications above, WhereScape RED adds up to a further 3 characters to the end of the index name. These additional values will be "_0" through "_99", or "_A" through "_Z", or "_SC".

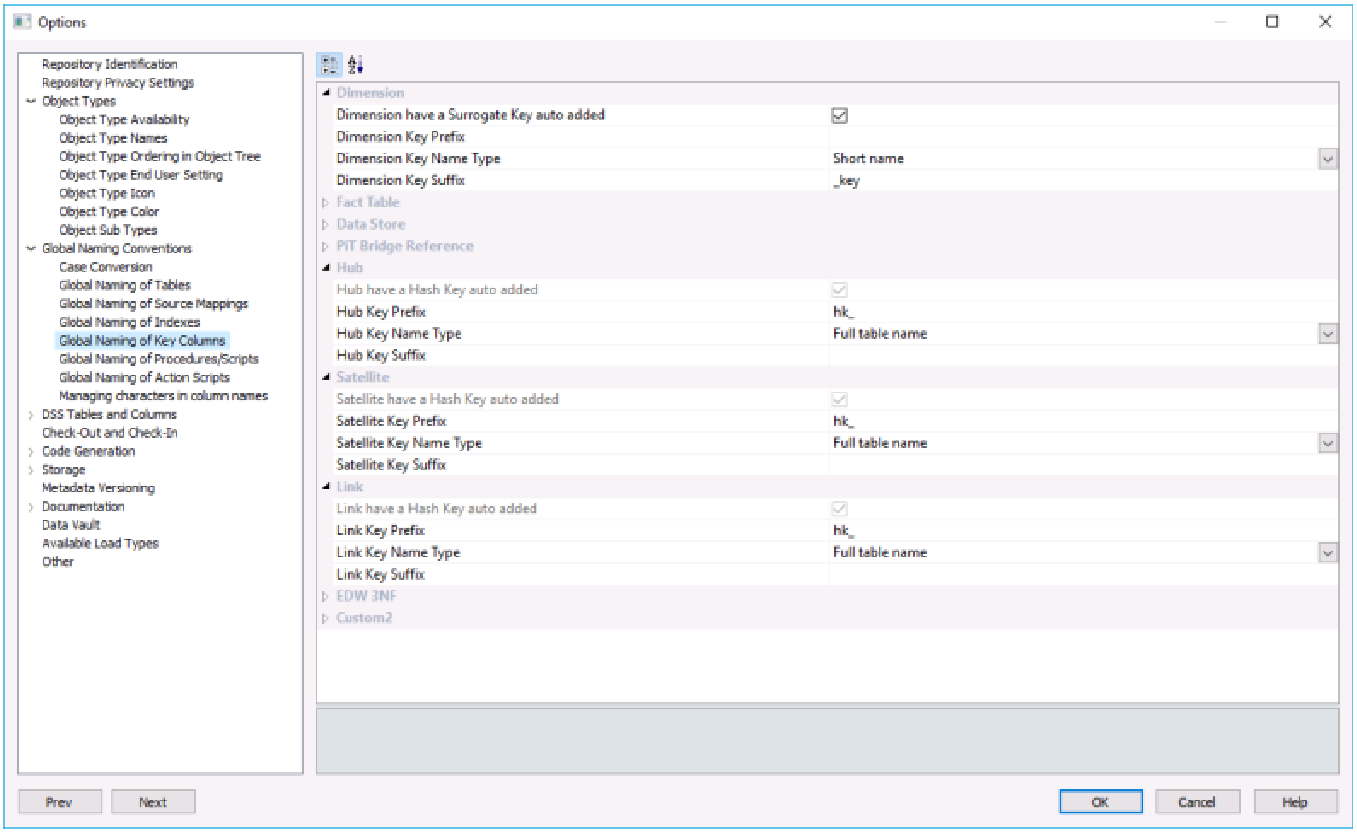
When a new index is manually added it will have the additional value of "_x" by default. This should be changed. The WhereScape RED naming standard for indexes is described below, but any valid name may be used.

From the example screen above, a fact table would have indexes generated using the short name and with a suffix of "_idx". Therefore, a fact_sales fact table would have indexes, such as fact_sales_idx_x.

Ultimate suffix	Meaning
_0	artificial key
_1 thru _99	bitmap key index on dimensional join
_A	primary business key
_B thru _Z	secondary business keys
_SC	key to support slowly changing dimensions

Global Naming of Key Columns

This option enables you to set the **Global Naming of Key Columns**.



Dimension
Dimension have a Surrogate Key auto added ☑
Dimension Key Prefix
Dimension Key Name Type Short name
Dimension Key Suffix _key
Fact Table
Data Store
PiT Bridge Reference
Hub
Hub have a Hash Key auto added ☑
Hub Key Prefix hk_
Hub Key Name Type Full table name
Hub Key Suffix
Satellite
Satellite have a Hash Key auto added ☑
Satellite Key Prefix hk_
Satellite Key Name Type Full table name
Satellite Key Suffix
Link
Link have a Hash Key auto added ☑
Link Key Prefix hk_
Link Key Name Type Full table name
Link Key Suffix
EDW 3NF
Custom2

During the drag and drop generation of new dimension and fact tables, WhereScape RED builds an artificial (surrogate) key for the table.

The naming convention for that key can be set through the same menu option as above. As well as potential prefix and suffix values, you also need to choose between the inclusion of the full table name, or the short name assigned to each table.

In the example screen above (which is the default), a dimension table key would use the table **short name** and have a suffix of "**_key**". Therefore, a load_customer table example would generate a key called **dim_customer_key**, if it was dragged into a dimension drop target.

The example above displays the defaults for Dimension options, to set these fields on Fact, Data Store and EDW 3NF tables, etc. expand the fields below Dimension to view and set your required options.

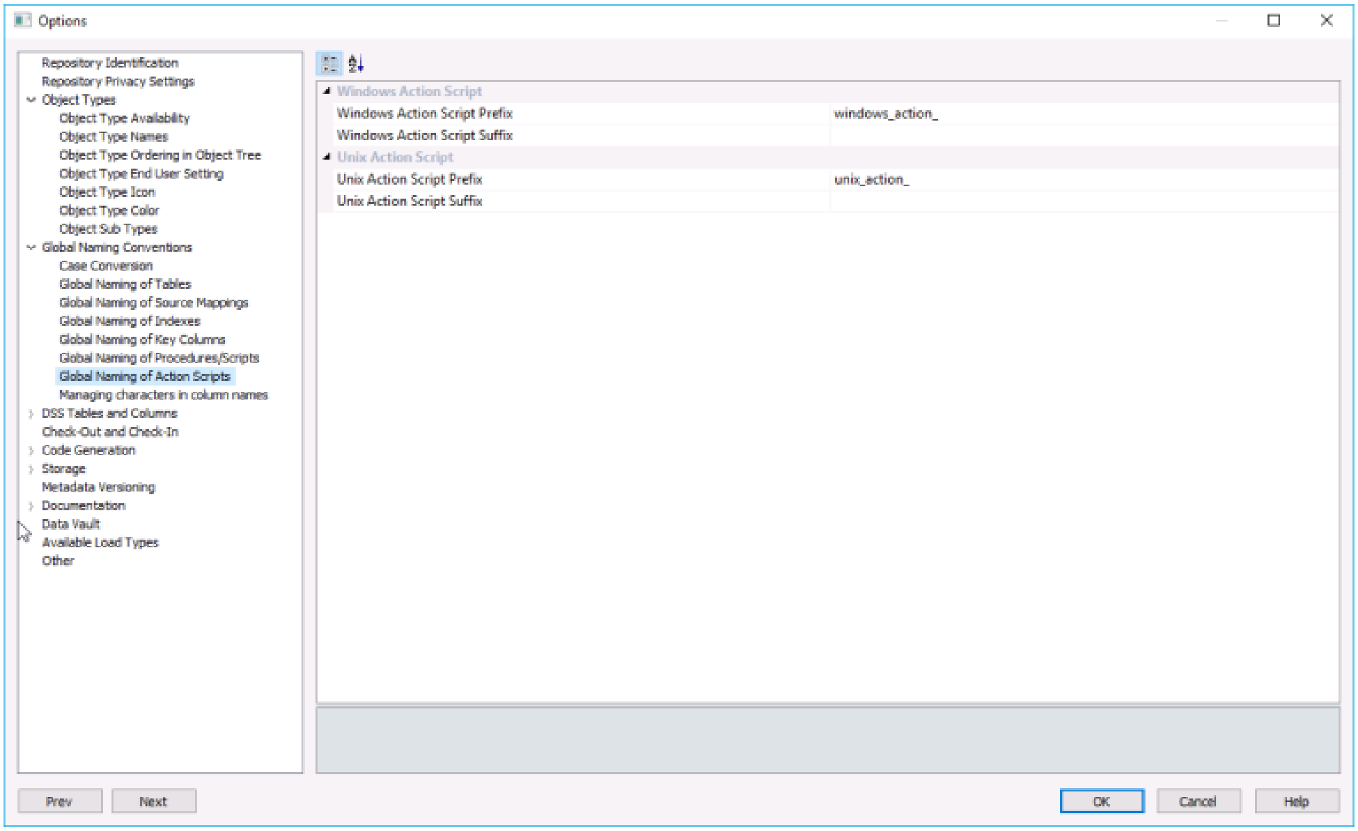
Fields	Description
Dimension have a Surrogate Key auto added	Select this option if a Surrogate key column is to be added automatically to a table. Default for Dimension is selected. Default for Fact, Data Store and EDW 3NF is not selected.
Dimension Key Prefix	Key prefix that can be added to a new Dimension Key.
Dimension Key Name Type	Key name type for new Dimension keys. Select between Short name, Full table name and Base name.
Dimension Key Suffix	Key suffix that can be added to a new Dimension key.

Note:

For Data Vault table objects, hash keys are added automatically by the **Data Vault stage table creation wizard** and are required when using WhereScape data vault code generation templates. Therefore, the **Hash key auto added** option for Hub, Satellite and Link tables are enabled by default and cannot be modified. Refer to **Data Vault Settings** for details.

Global Naming of Action Scripts

The default naming conventions for action scripts can be set through the **Home > Options** menu.



Windows Action Script

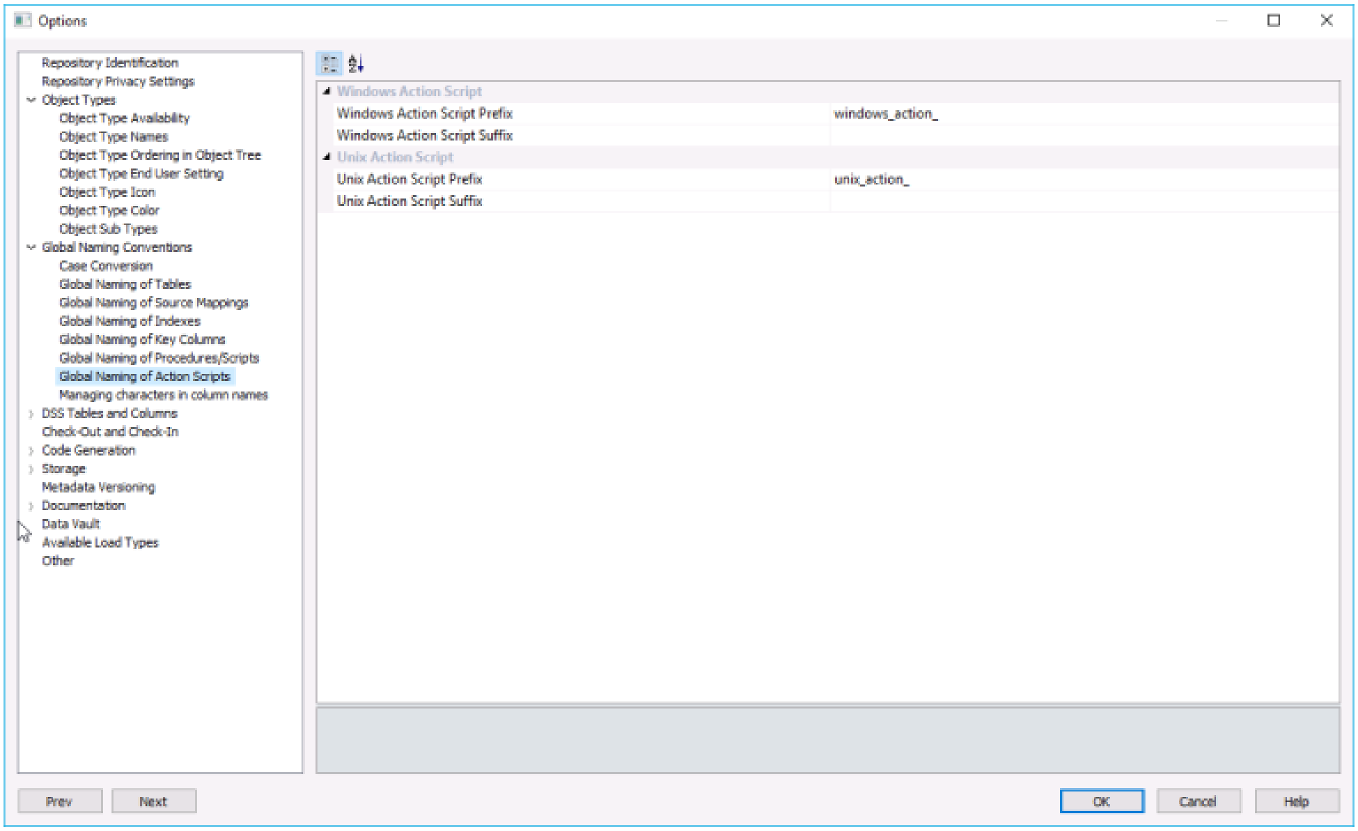
Windows Action Script Prefix	windows_action_
Windows Action Script Suffix	

Unix Action Script

Unix Action Script Prefix	unix_action_
Unix Action Script Suffix	

Global Naming of Action Scripts

The default naming conventions for action scripts can be set through the **Home > Options** menu.



Windows Action Script

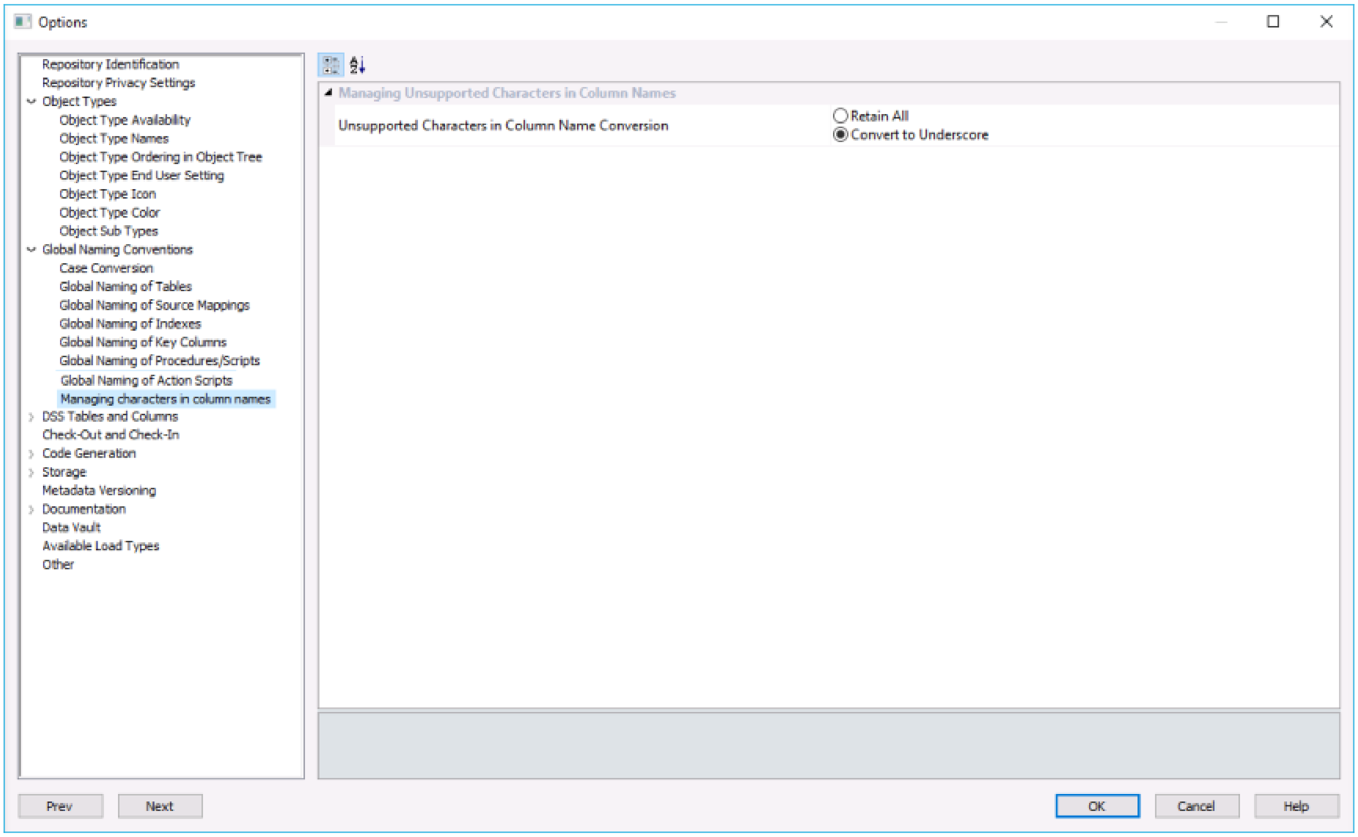
Windows Action Script Prefix	windows_action_
Windows Action Script Suffix	

Unix Action Script

Unix Action Script Prefix	unix_action_
Unix Action Script Suffix	

Managing Characters in Column Names

This option enables you to set the default action for **Managing characters in column names**.



Managing Unsupported Characters in Column Names

Unsupported Characters in Column Name Conversion: ○ Retain All ● Convert to Underscore

This option enables you manage unsupported characters in column names when loading data from a source table, e.g. spaces in the source column names.

Option	Description
Retain All	Set this field if you want to retain all characters in the column name conversion, when you create a **Load** table from a source table with unsupported characters in its column names.
Convert to Underscore	This is the default setting. RED replaces the spaces in the source column names, when you create a **Load** table via drag and drop of a source table from the right pane to the middle pane of the Builder window.

DSS Tables and Columns

When building the data warehouse, WhereScape RED makes use of a number of special tables and columns. The two tables used are called by default **dss_source_system** and **dss_fact_table**. These tables are discussed in detail in the sections below.

The special columns used are defined in the table below.

Column name	Description
dss_batch	Not used at this stage.
dss_source_system_key	Added to support dimensions that cannot be fully conformed, and the inclusion of subsequent source systems. See the section below for more details.
dss_fact_table_key	Used in composite rollup fact tables to identify the source fact table that contributed the particular row.
dss_create_time	Indicates when a record was created.
dss_update_time	Indicates when the record was last updated in the data warehouse. Used in the update of rollup fact tables and aggregate tables.
dss_count	Applied to fact tables. Provides a simple row count variable that can be used by end user tools.
dss_current_flag	Used for slowly changing dimensions. This flag identifies the current record where multiple versions exist.
dss_version	Used for slowly changing dimensions. This column contains the version number of a dimension record. Numbered from 1 upwards with the highest number being the latest or current version. It forms part of the unique constraint for the business key of a slowly changing dimension.
dss_start_date	Used for slowly changing dimensions. This column provides a date time stamp when the dimension record began life. If null then it was the first record. It is used to ascertain which dimension record should be used when multiple are available.
dss_end_date	Used for slowly changing dimensions. This column provides a date time stamp when the dimension record ceased to be the current record. It is used to ascertain which dimension record should be used when multiple are available.
dss_change_hash	Used for a Satellite table. This column identifies the differences in the descriptive columns of a Satellite table which is used for generating the change hash key for creating a Satellite object.

All of these special columns and tables can be renamed through the **Tools > Options > DSS Tables and Columns** menu option. The columns can simply be renamed, but the tables require valid table names that meet certain criteria. Refer to the related sections below.

Note:

Note:

When using table names other than the defaults for dss_source_system and dss_fact_table it is worth considering the fact that by default, the metadata backups will include any table that begins with "dss_". Therefore, if a table is used it is recommended that it have a name starting with "dss_". The advantage is that a working meta repository will be established through a backup and restore, if these tables are included in the backup set.

dss_source_system

This pseudo dimension is designed to identify a data source for a dimension row. Its purpose is to handle non conformed dimensions, or changes in source systems. If its use is not desired (default), then leave this field blank.

For example:

An organization has a number of factories. These factories are referenced by all of the operational systems. The production system has its own code for each factory, and this is the unique means of identifying the factory. The distribution system has a factory short name which it uses for the unique identifier. The raw materials system simply uses the factory name. It is probably not practical or even desirable to force these source systems to utilize a standard factory identification method, so instead we allow the dimension to be non-conformed. We do however, insist on a standard factory name convention, so that our reports and queries will join information when the factory name is used.

In this example, the dss_source_system_key is used to identify the source of the data for the dimension row. It also adds to the unique business key, so that two source systems can utilize the same code to refer to different entities. This key also provides a degree of future proofing in the data warehouse, to assist in the possible changing of an underlying source system.

The generated procedure code always set the key value of this table to 1. Therefore, manual code changes is required to make use of the functionality that this table offers.

When dimensions are built, the dss_source_system_key is added by default. It can be deleted if the need for such a feature is not seen.

If this table is to be given a different name ,then it and all its columns can be renamed, or the following steps can be taken:

1. Create a new table by dragging the column dss_source_system_name from dss_source_system into a dimension target.
2. Change the object type from dimension view to dimension and specify the new table name. (see note above on the use of "dss_").
3. Rename the dss_source_system_name column to match the new table name.
4. Delete the last two columns.
5. Under the table Properties, change the table type to **Mapping table**. This prevents the table from being seen as a dimension in the documentation.
6. Change the dss_source_system table name in the screen above, via the **Tools>Options>DSS Tables and Columns** defaults menu option.

dss_fact_table

This pseudo dimension is used internally by the WhereScape RED generated procedures to assist in the updating of rollup fact tables. It provides a means of identifying which fact table contributed a particular row to a composite rollup fact table. See the section on rollup fact tables for an explanation of how the generated procedures operate.

This table can be renamed, or a new table created. The table identified as the 'dss_fact_table' must, however, conform to a number of standards if the generated code is to compile and work correctly.

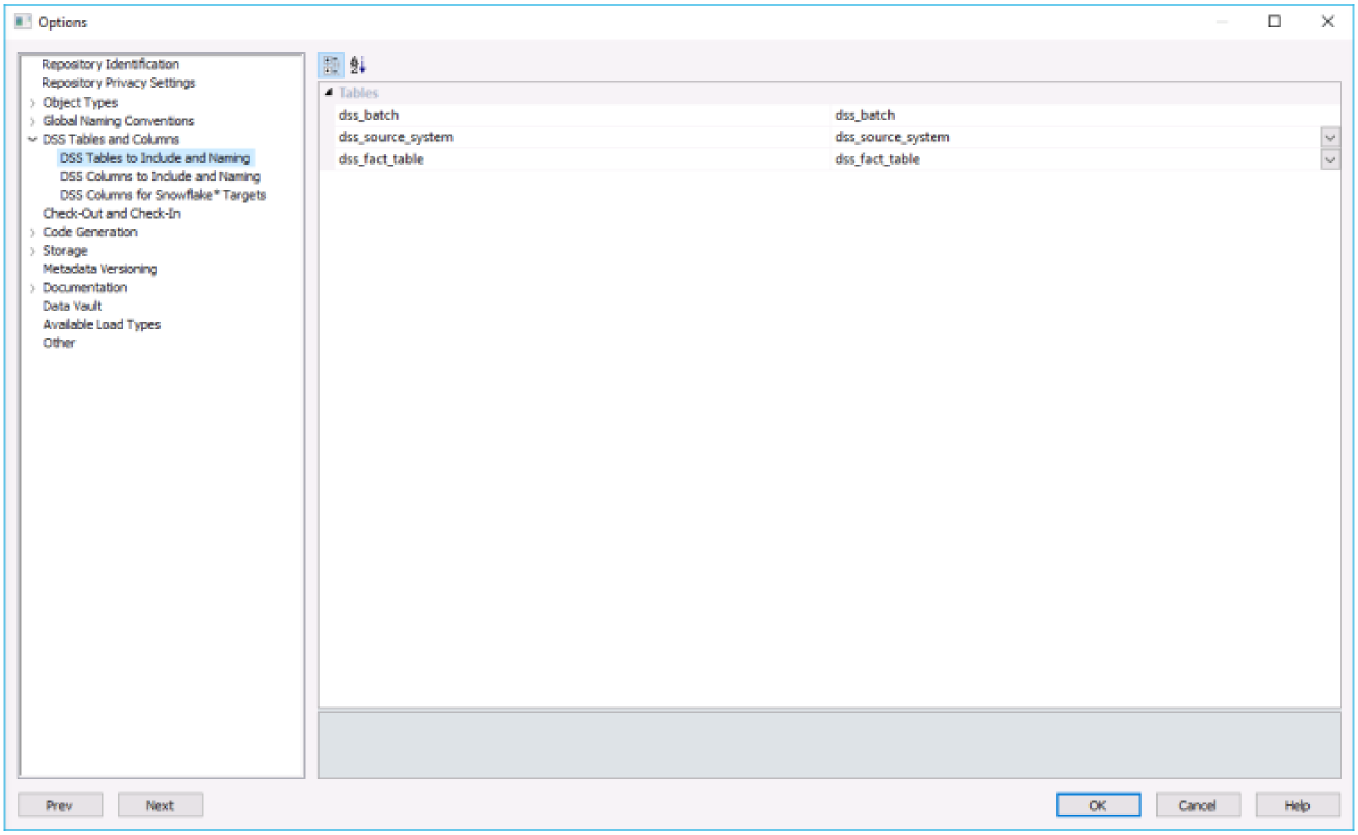
The following columns must exist, where table_name is the new name of the table:

Column name	Purpose
table_name_key	The artificial key for the table. Must be a numeric data type.
table_name_name	The name of the fact table. Must be at least a varchar2(64) data type.
table_name_type	The type of fact table. Must be at least a varchar2(24).

The generated procedures automatically add rows to this table, as required. As with the dss_source_system table, the table type for this table must be set to **Mapping table** to prevent it from being seen as a Dimension in the documentation.

DSS Tables

This option enables you to set the **DSS Tables**.


Options
Repository Identification
Repository Privacy Settings
Object Types
Global Naming Conventions
DSS Tables and Columns
DSS Tables to Include and Naming
DSS Columns to Include and Naming
DSS Columns for Snowflake* Targets
Check-Out and Check-In
Code Generation
Storage
Metadata Versioning
Documentation
Data Vault
Available Load Types
Other
Prev
Next
OK
Cancel
Help

Tables

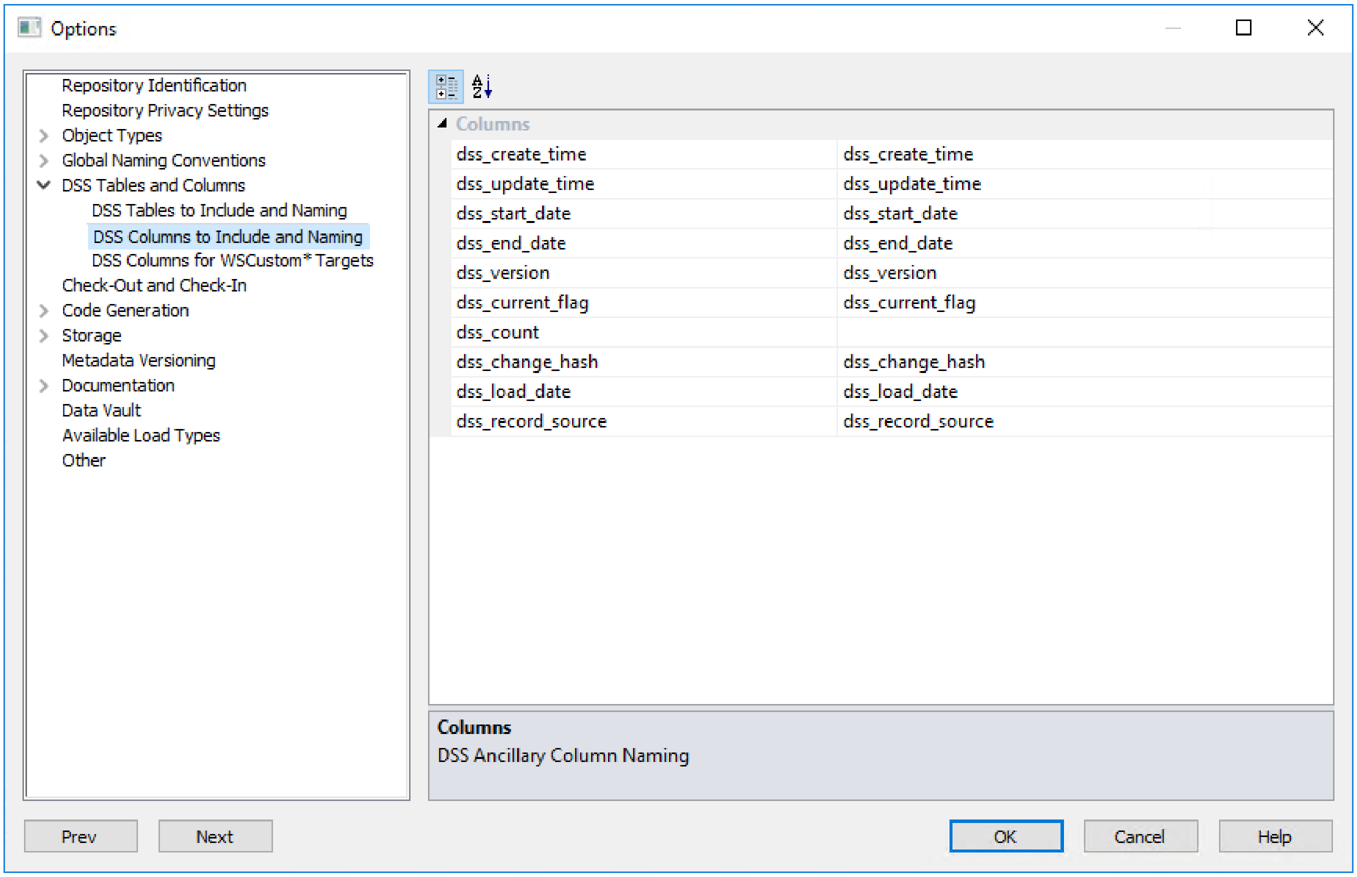
dss_batch	dss_batch
dss_source_system	dss_source_system
dss_fact_table	dss_fact_table

Tables

Set the **DSS Tables**.

DSS Columns

This option enables you to set the **DSS Columns**.


Options
Repository Identification
Repository Privacy Settings
Object Types
Global Naming Conventions
DSS Tables and Columns
DSS Tables to Include and Naming
DSS Columns to Include and Naming
DSS Columns for WSCustom* Targets
Check-Out and Check-In
Code Generation
Storage
Metadata Versioning
Documentation
Data Vault
Available Load Types
Other
Prev
Next
OK
Cancel
Help

Columns

dss_create_time	dss_create_time
dss_update_time	dss_update_time
dss_start_date	dss_start_date
dss_end_date	dss_end_date
dss_version	dss_version
dss_current_flag	dss_current_flag
dss_count	
dss_change_hash	dss_change_hash
dss_load_date	dss_load_date
dss_record_source	dss_record_source

Columns
DSS Ancillary Column Naming

Columns

Set the **DSS Columns**.

Fields	Description
dss_create_time	Column added to all Stage, ODS, EDW 3NF, Dimension, Fact and Aggregate tables for information purposes only. Leave the field blank to deactivate or add a name for the dss_create_time column, default is dss_create_time.
dss_update_time	Column added to all dimension, stage and fact tables. It is required if the generated code for fact and aggregate tables is to be used.
dss_start_date	Column used for slowly changing dimensions. It is used to identify when a dimension row was replaced. This is a required field.
dss_end_date	Column used for slowly changing dimensions. It is used to identify when a dimension row was replaced. This is a required field.
dss_version	Column used for slowly changing dimensions. Is it used to store the version of a dimension row. This is required for unique constraints.
dss_current_flag	Column used for slowly changing dimensions. It is used to identify the current dimension row. This is a required field.
dss_count	Column added to fact tables to provide a simple row counter. This is required but can be deleted after generation. Leave the field blank to deactivate.
dss_change_hash	Column used to identify the differences in the descriptive columns of a Satellite table which is used for generating the change hash key for a Satellite object. Refer to **Creating Data Vault Stage Tables** for details.
dss_load_date	Column used to store the date when the value in this row was loaded into the metadata repository. This column and the **dss_record_source** column below is added to new Load tables that have the option **Add meta data columns to table** selected. Refer to **Data**

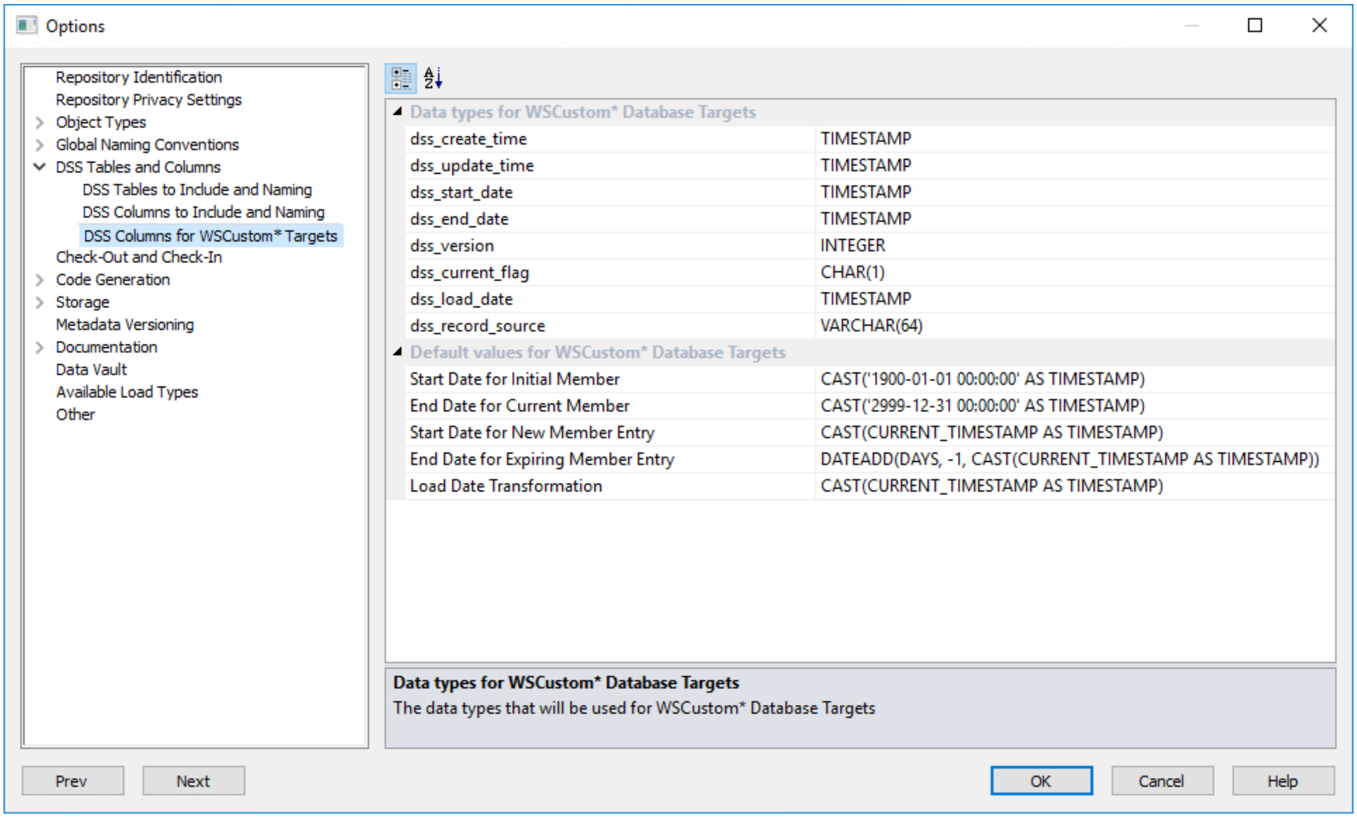
Fields	Description
	Vaults for details.
dss_record_source	Column used to store a descriptive term to identify the source of this record. This column and the **dss_load_date** column above is added to new load tables that have the option **Add meta data columns to table** selected. Refer to **Data Vaults** for details.

DSS Columns for Custom Targets

Note:

These settings are only available/displayed if your WhereScape RED license includes support for custom target databases.

This option enables you to set the **DSS Columns** for custom database targets.


Options
Repository Identification
Repository Privacy Settings
Object Types
Global Naming Conventions
DSS Tables and Columns
DSS Tables to Include and Naming
DSS Columns to Include and Naming
DSS Columns for WSCustom* Targets
Check-Out and Check-In
Code Generation
Storage
Metadata Versioning
Documentation
Data Vault
Available Load Types
Other
Prev
Next
OK
Cancel
Help

Data types for WSCustom* Database Targets	
dss_create_time	TIMESTAMP
dss_update_time	TIMESTAMP
dss_start_date	TIMESTAMP
dss_end_date	TIMESTAMP
dss_version	INTEGER
dss_current_flag	CHAR(1)
dss_load_date	TIMESTAMP
dss_record_source	VARCHAR(64)
Default values for WSCustom* Database Targets	
Start Date for Initial Member	CAST('1900-01-01 00:00:00' AS TIMESTAMP)
End Date for Current Member	CAST('2999-12-31 00:00:00' AS TIMESTAMP)
Start Date for New Member Entry	CAST(CURRENT_TIMESTAMP AS TIMESTAMP)
End Date for Expiring Member Entry	DATEADD(DAYS, -1, CAST(CURRENT_TIMESTAMP AS TIMESTAMP))
Load Date Transformation	CAST(CURRENT_TIMESTAMP AS TIMESTAMP)

Data types for WSCustom* Database Targets
The data types that will be used for WSCustom* Database Targets

Data types for Custom Database Targets

Enables you to set the data types that are used in the DSS columns added to new table objects.

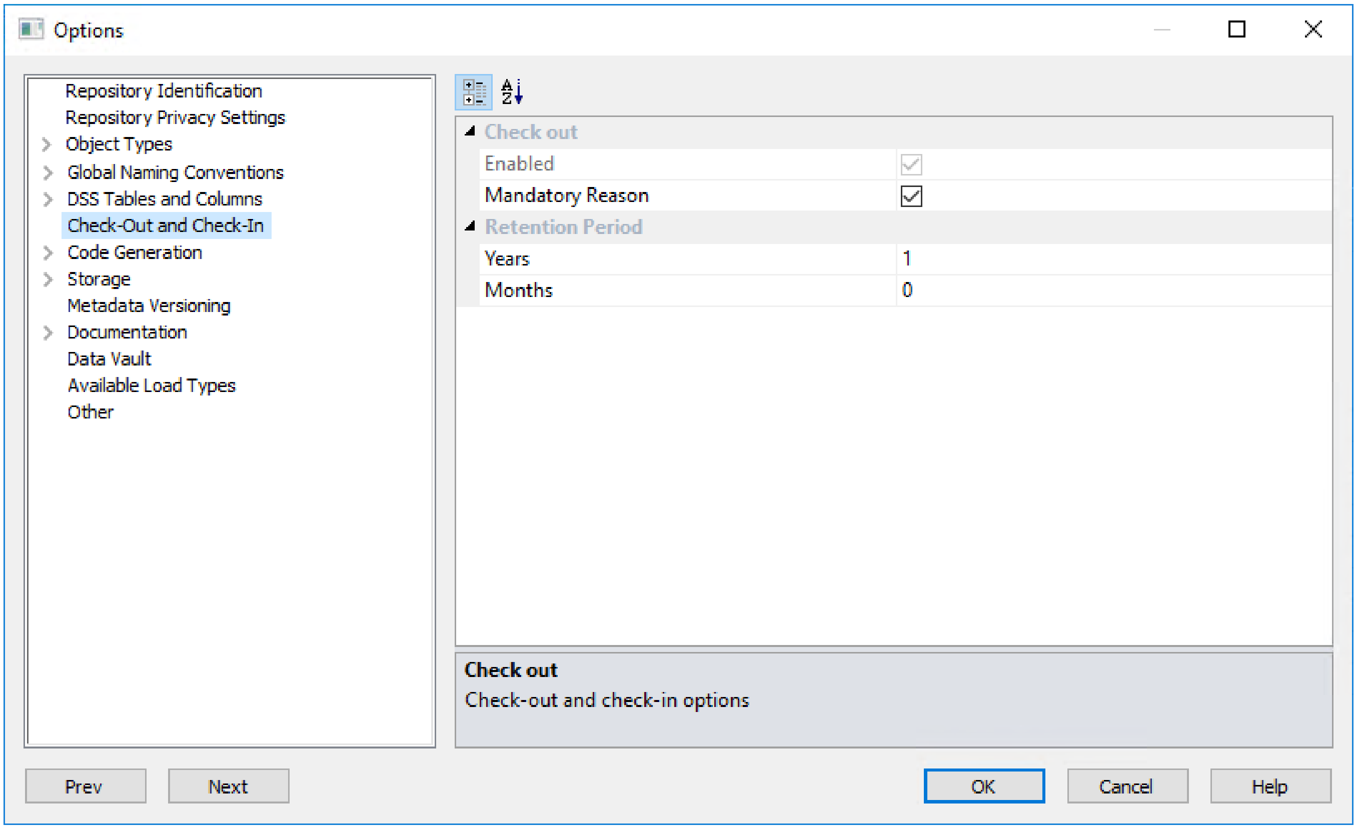
Notes:

1. The **Hash Key Generation Wizard** creates new hub and link hash keys in a Data Vault Stage table with the same data type defined for the **dss_change_hash** column, if the table is in a Custom database target.
2. If the table is **not** in a Custom database target, the data type CHAR(32) is used.
3. Changing this data type has no impact on any existing hash key columns.

Fields	Description
Default values for Custom Database Targets	Enables you to set the default values that are used when generating update procedures for Slowly Changing Dimension tables and Load tables.
Start Date for Initial Member	Default date function to use for the start date of rows added in a Slowly Changing Dimension table.
End Date for Current Member	Default date function to use for the end date of current rows in a Slowly Changing Dimension table.
Start Date for New Member Entry	Default date function to use for the start date of current rows in a Slowly Changing Dimension table.
End Date for Expiring Member Entry	Default date function to use for the end date of expiring rows in a Slowly Changing Dimension table. **Note** Refer to the **Change Detection tab** section under **Generating the Dimension Update Procedure** for details.
Load Date Transformation	Default date function to use for populating the new records inserted in Load tables.

Check-Out and Check-In

This option enables you to set up for the **Checking-out** or **Checking-In** of Procedures.



Check out

Enabled	☑
Mandatory Reason	☑

Retention Period

Years	1
Months	0

Check out
Check-out and check-in options

Check out

- **Enabled**: This option is selected by default to enable procedures to be checked-in or checked-out.

- **Mandatory Reason**: Select this option if a reason is mandatory for checking out or checking in procedures, else deselect.

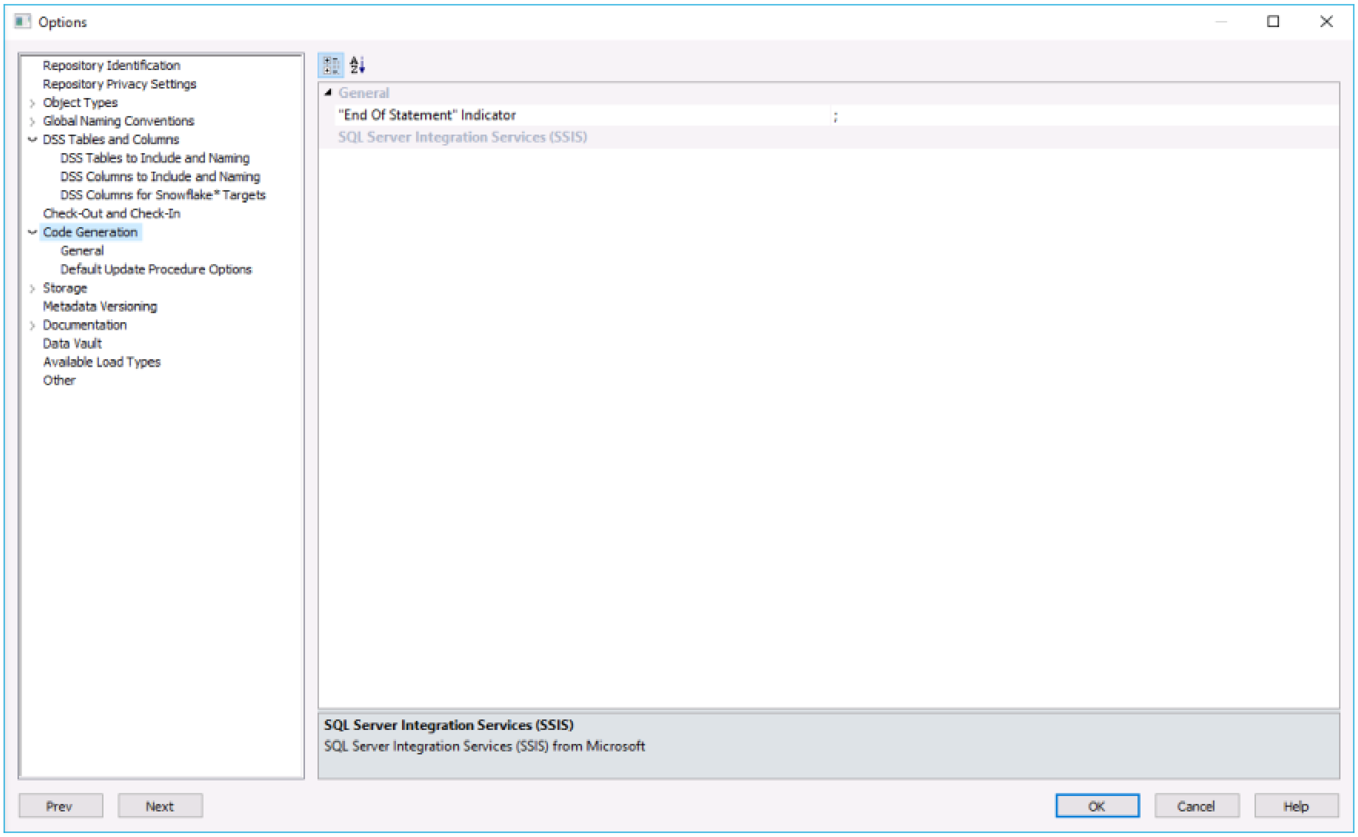
Retention Period

Set the length of time; **Years** and **Months**, for which procedures may be checked-out.

Code Generation

General

This option enables you to set some general **Code Generation** settings.


Options
Repository Identification
Repository Privacy Settings
Object Types
Global Naming Conventions
DSS Tables and Columns
DSS Tables to Include and Naming
DSS Columns to Include and Naming
DSS Columns for Snowflake* Targets
Check-Out and Check-In
Code Generation
General
Default Update Procedure Options
Storage
Metadata Versioning
Documentation
Data Vault
Available Load Types
Other
Prev
Next
OK
Cancel
Help

General
"End Of Statement" Indicator ;
SQL Server Integration Services (SSIS)

SQL Server Integration Services (SSIS)
SQL Server Integration Services (SSIS) from Microsoft

General

- **Include WsWrkTask Procedure**: This option is selected by default and results in a call to the WsWrkTask function being placed at the end of most of the generated update procedures. These calls to WsWrkTask result in counters being set in the meta table ws_wrk_task_log. These counters can be viewed via a query on the view **ws_admin_v_task**.
- **"End of Statement" Indicator**: Set the indicator to separate multiple SQL statements in a SQL block. If left blank, the default value of <EOS> is used.

Default Update Procedure Options

This option enables you to set some **default update procedure** settings for specific object types. Not all options are relevant for all combinations of database, object type and build type.

Fields	Description

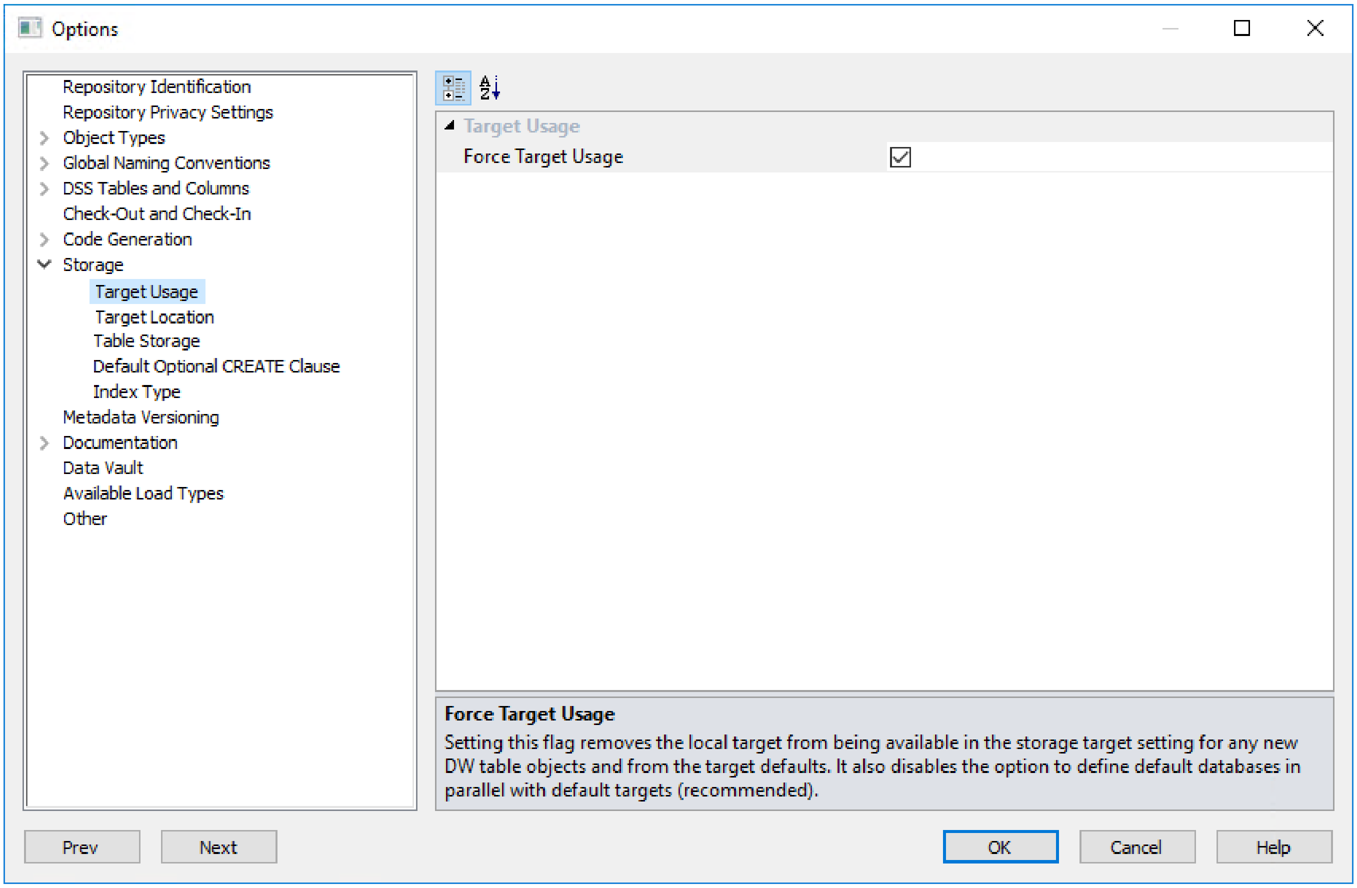
Fields	Description
Process in Batch	Enables users to select a column to drive data processing in a loop based on the distinct ordered values of the selected column.
Include Initial Load Insert	Enables users to add an additional insert statement to the update procedure. If the target table is empty, the new insert statement is run in place of the standard generated code.
Insert Zero KeyRecord	Enables users to add an insert statement for an unknown record with an artificial key of zero. Only applicable to tables with an artificial key.
Distinct Data Select	Enables users to ensure duplicate rows are not added to the table.
Select Hint	Enables users to enter a database-compliant hint to be used in the SELECT statement. Parameters $TABLE$ and $INDEXS is automatically replaced at procedure generation time.
Delete before Insert	Enables users to delete statement to be added to the update procedure before any update or insert statement.
Truncate	Performs a DDL operation to delete all records from a table, rather than a predicate based DML operation to delete individual rows. This option is automatically enabled when **Delete before Insert** is enabled, but can be deactivated separately.
Include Update Statement	Enables users to include an update statement in the procedure to update changing rows in the table.
Update Changed Rows Only	Enables users to use change detection to work out what rows require updating.
Include Insert Statement	Enables users to include an insert statement in the procedure to insert new rows in the table.
Insert New Rows Only	Enables users to use change detection to work out what rows require inserting.
Include Merge Statement	Enables users to include a merge statement in the procedure to merge new/changed rows in the table.
Merge New/Changed Rows Only	Enables users to use change detection to interpret which rows require merging.

Storage

Target Usage

This option enables you to force the use of table target locations (defined in the **Connection Properties > Target Settings**), and prevents the use of legacy local targets when creating new table objects in RED.

This option is enabled by default for new repositories—it is recommended to define targets for the data warehouse objects that is separate from the metadata database.



Target Usage

Force Target Usage ☑

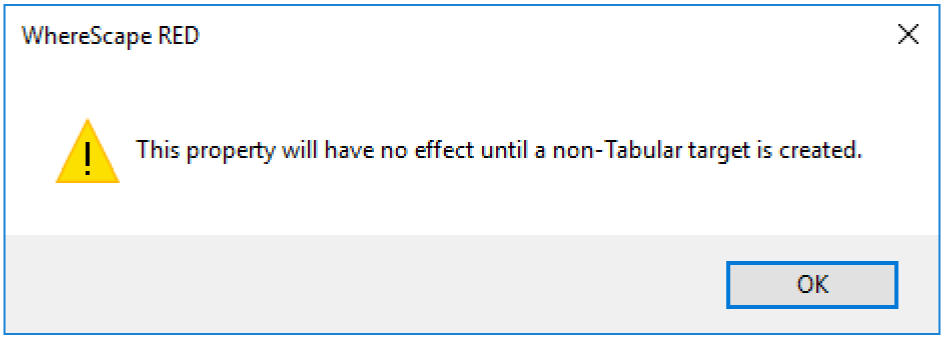
Force Target Usage

Setting this flag removes the local target from being available in the storage target setting for any new DW table objects and from the target defaults. It also disables the option to define default databases in parallel with default targets (recommended).

The **Force Target Usage** setting removes the option to use local targets in the **Default Target** drop-down list of the **Target Location** option and also hides the **Table Storage** option, which is used to set the storage locations for each table object type created in RED.

Notes:

- Existing RED objects or objects loaded into RED with **local** set as their target, retains the local target setting.
- If the target location is changed from local to a specific target, then there is no option to set it back to local without deactivating the **Force Target Usage** setting.
- If table target(s) has not been defined, a message is displayed to warn users that this setting does not take effect, until a target is created.



This property will have no effect until a non-Tabular target is created.

To see more about creating target locations, Refer to **Connection to the Data Warehouse** for details.

Target Location

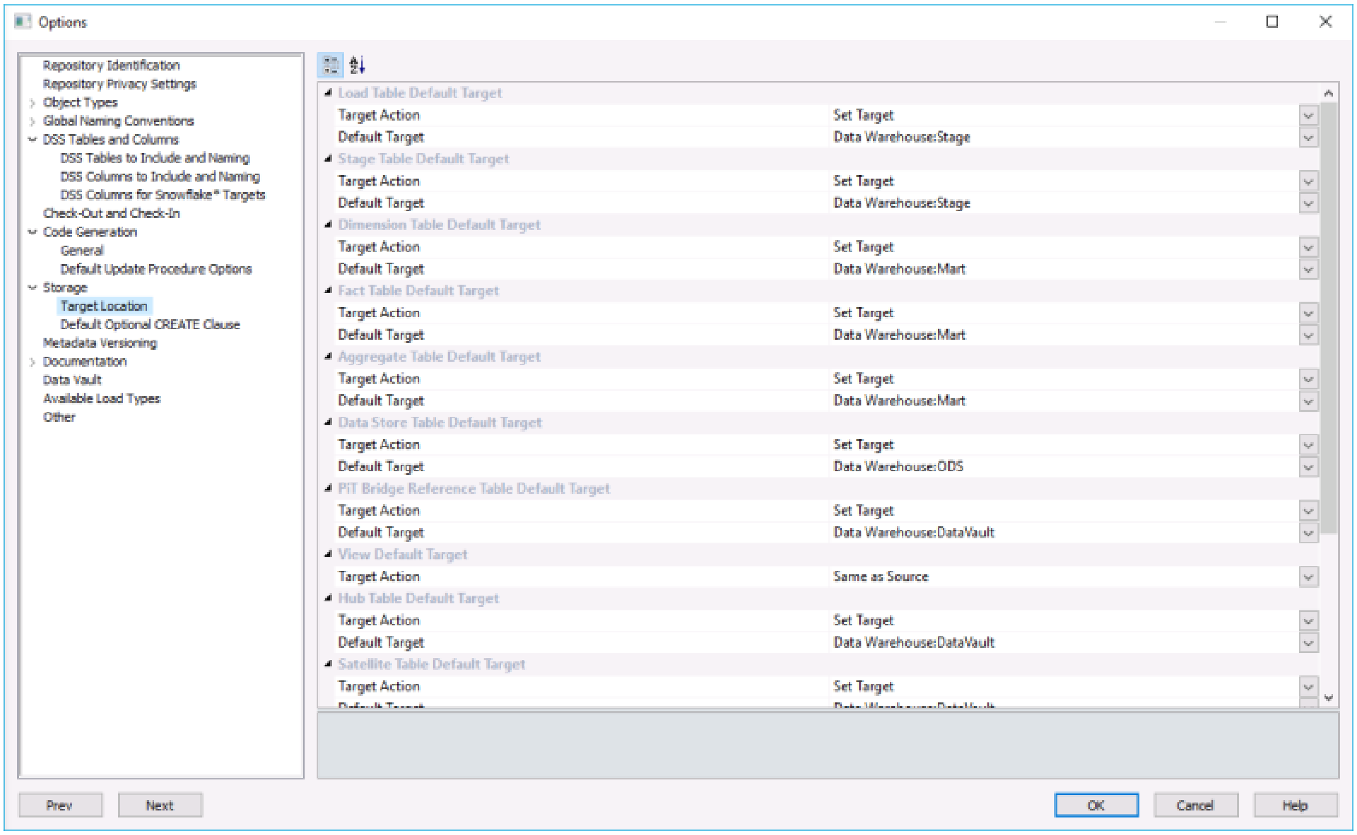
Target Location options enable users who are placing objects across multiple schemas to set default target locations for new tables.

Note:

Refer to **Distributing Tables across Multiple SQL Server Databases** if using multiple databases as target locations for tables in an SQL environment.

Default table target locations can be set for the following objects:

- Load
- Stage
- Dimension
- Fact
- KPI Fact
- Aggregate
- Data Store
- EDW 3NF
- View
- Hub Table
- Satellite
- Link
- Custom


Options
Repository Identification
Repository Privacy Settings
Object Types
Global Naming Conventions
DSS Tables and Columns
DSS Tables to Include and Naming
DSS Columns to Include and Naming
DSS Columns for Snowflake* Targets
Check-Out and Check-In
Code Generation
General
Default Update Procedure Options
Storage
Target Location
Default Optional CREATE Clause
Metadata Versioning
Documentation
Data Vault
Available Load Types
Other
Prev
Next
OK
Cancel
Help

Load Table Default Target	
Target Action	Set Target
Default Target	Data Warehouse:Stage
Stage Table Default Target	
Target Action	Set Target
Default Target	Data Warehouse:Stage
Dimension Table Default Target	
Target Action	Set Target
Default Target	Data Warehouse:Mart
Fact Table Default Target	
Target Action	Set Target
Default Target	Data Warehouse:Mart
Aggregate Table Default Target	
Target Action	Set Target
Default Target	Data Warehouse:Mart
Data Store Table Default Target	
Target Action	Set Target
Default Target	Data Warehouse:ODS
PiT Bridge Reference Table Default Target	
Target Action	Set Target
Default Target	Data Warehouse:DataVault
View Default Target	
Target Action	Same as Source
Hub Table Default Target	
Target Action	Set Target
Default Target	Data Warehouse:DataVault
Satellite Table Default Target	
Target Action	Set Target
Default Target	Data Warehouse:DataVault

Target Action

Set Target

This option enables you to set a default target for new tables to be created. It enables the **Default Target** drop-down list where a specific target for new tables can be defined.

Note:

If the **Force Target Usage** option is enabled in the **Target Usage** setting, selecting **Set Target** automatically sets the **Default Target** field with the first target location value, available from the drop-down list. The option to use **Local** target is also not available.

Same as Source

This option is selected if the table's default storage must be the same as the original source where the table is coming from. This option cannot be selected for Load tables.

Default Target

A default target can only be entered if the **Set Target** action has been selected in the **Target Action** drop-down list.

With this option, users can choose between setting a table's default location:

- (**local**) if the **Force Target Usage** setting is not enabled in the **Target Usage** setting, or
- to any other **target locations** that have been defined in the relevant connection Properties.

To set a default target location on a table by table basis:

1. Select the **Set Target > Same as Source** option from the **Target Action** drop-down list.
2. To have tables located in a specific target, select a **default target** where the new object is stored, when the object is dragged and dropped to the middle work pane.

Even though the default target location can be set in the **Target Location** options, this setting can also be changed after the table has been created via the **Storage** tab of each table object's **Properties** screen.

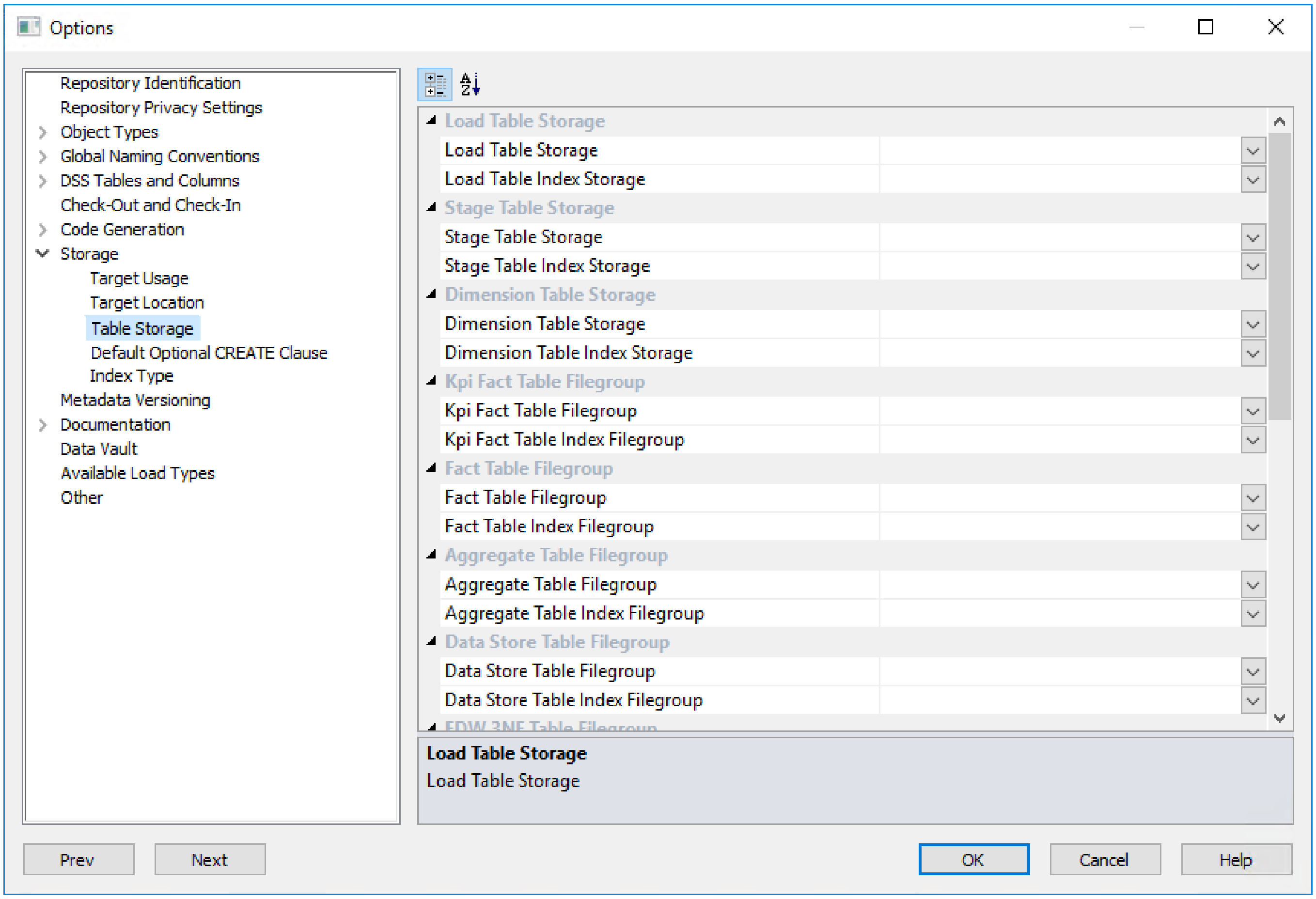
To see more information about changing the schema after a table has been created, refer to **Storage**.

Table Storage

Note

The **Table Storage** option is only available/visible if the **Target Usage > Force Target Usage** setting is not enabled.

This option enables you to set the **Storage** locations for each table object type created in RED.



Set the **Storage** locations for each table type.

These defaults are applied when a table is created. They can be changed by selecting the **Storage** tab on the **Properties** screen of a table.

Default Optional CREATE Clause

This option enables you to define a default value for the "Optional CREATE Clause" property of each object type, which is populated when the object is first created.

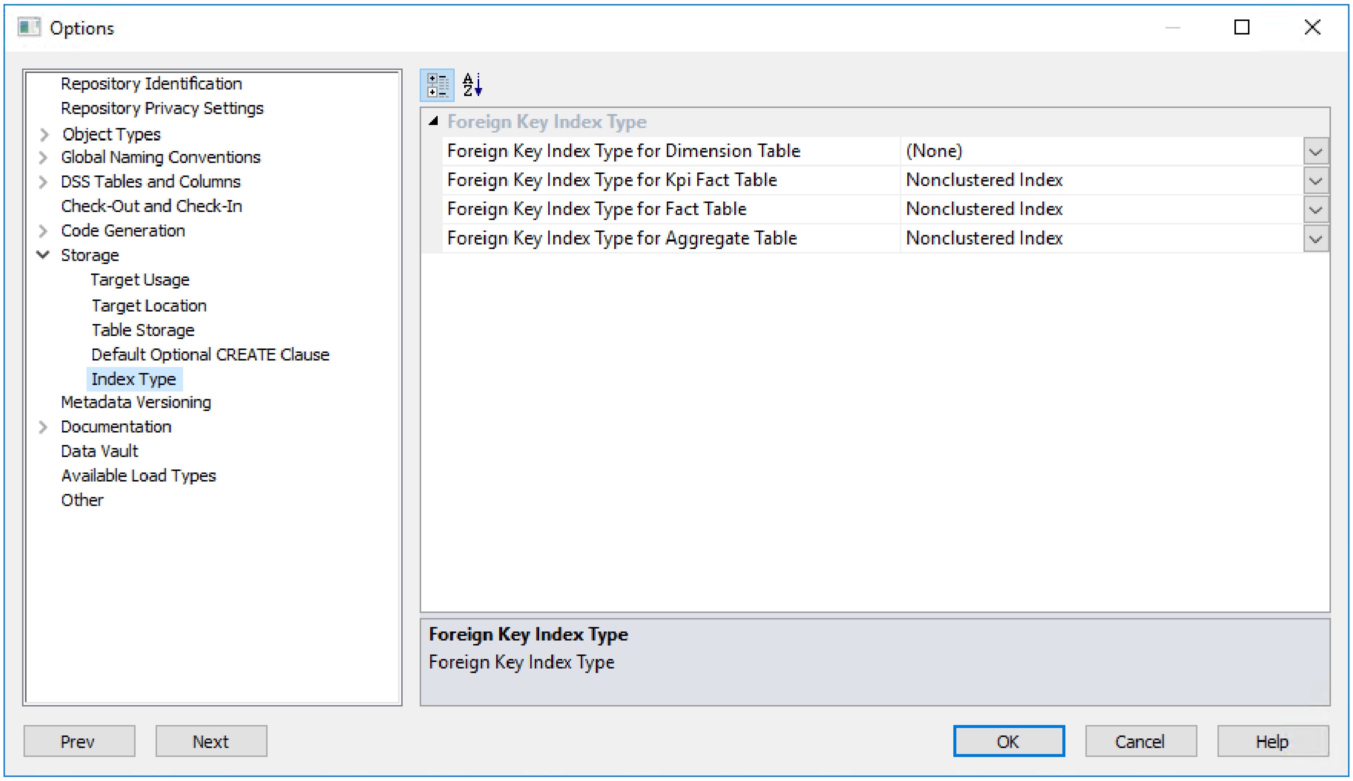
The Optional CREATE Clause text is appended to the DDL CREATE statement when the table is generated.

Tip:

This option is only used to set the default optional create clause for new objects.
To edit the Optional CREATE Clause of an existing object or edit the clause on a table by table basis, go to the object's Properties screen, click the Storage tab and edit the Optional CREATE Clause field.

Index Type

This option enables you to set the default type of **Foreign Key Index Type** for each table type.



Foreign Key Index Type

Foreign Key Index Type for Dimension Table	(None)
Foreign Key Index Type for Kpi Fact Table	Nonclustered Index
Foreign Key Index Type for Fact Table	Nonclustered Index
Foreign Key Index Type for Aggregate Table	Nonclustered Index

Foreign Key Index Type
Foreign Key Index Type

Foreign Key Index Type

Set the default index types built for foreign key columns of Dimension tables, KPI Fact tables, Fact tables and Aggregate tables.

SQL Server options:

- (None) - will not be defined
- Nonclustered Index- this is a standard index per key column
- COVERING - This is a single index with all the key columns in it
- COVERING ColumnStore Index

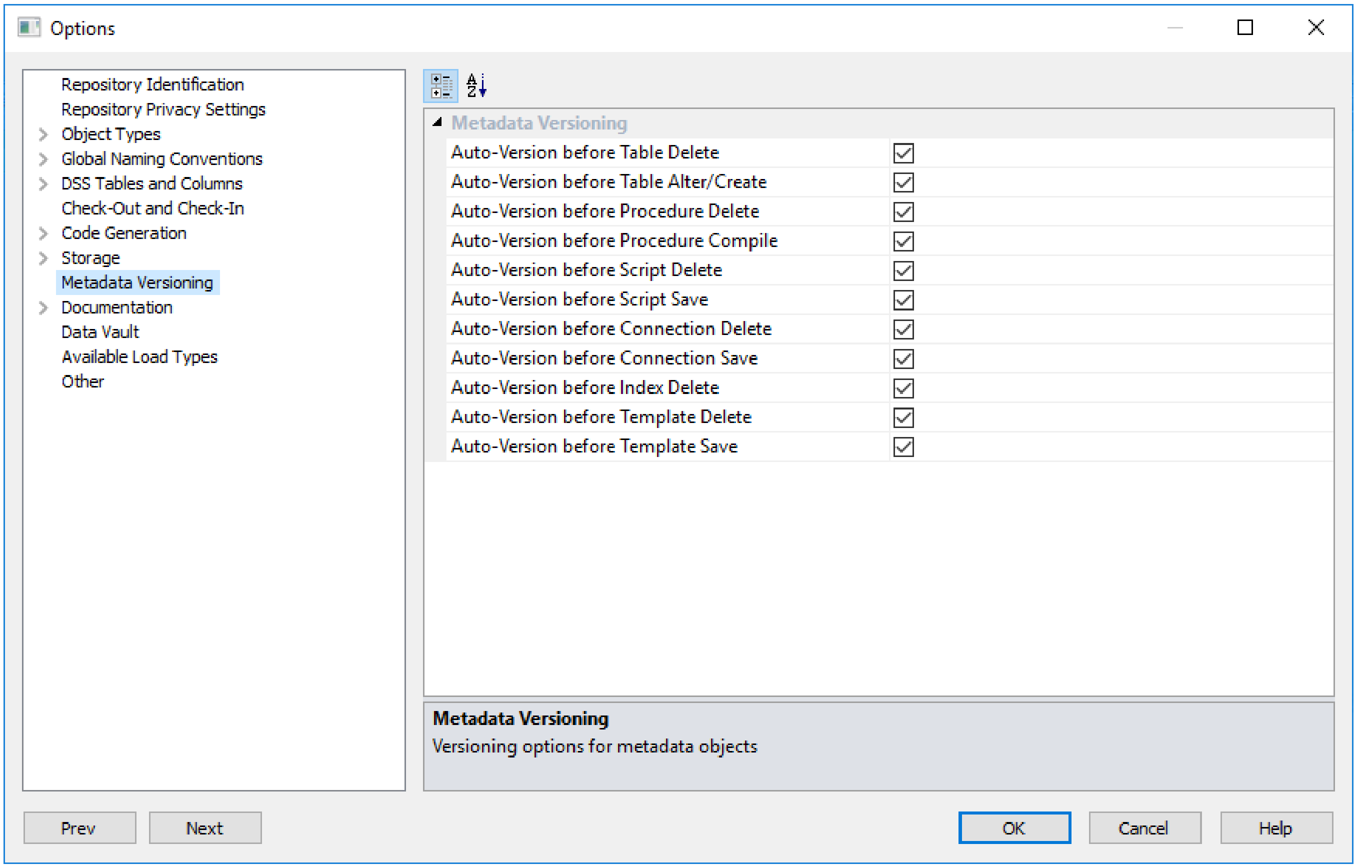
These defaults are applied when an index definition is created.

They can be changed by selecting the **Storage** tab in the **Properties** window of an index.

Metadata Versioning

This option enables you to alter the **Metadata Versioning** settings.

Select or clear the corresponding check box to set when to auto-version the metadata. All options are selected by default.



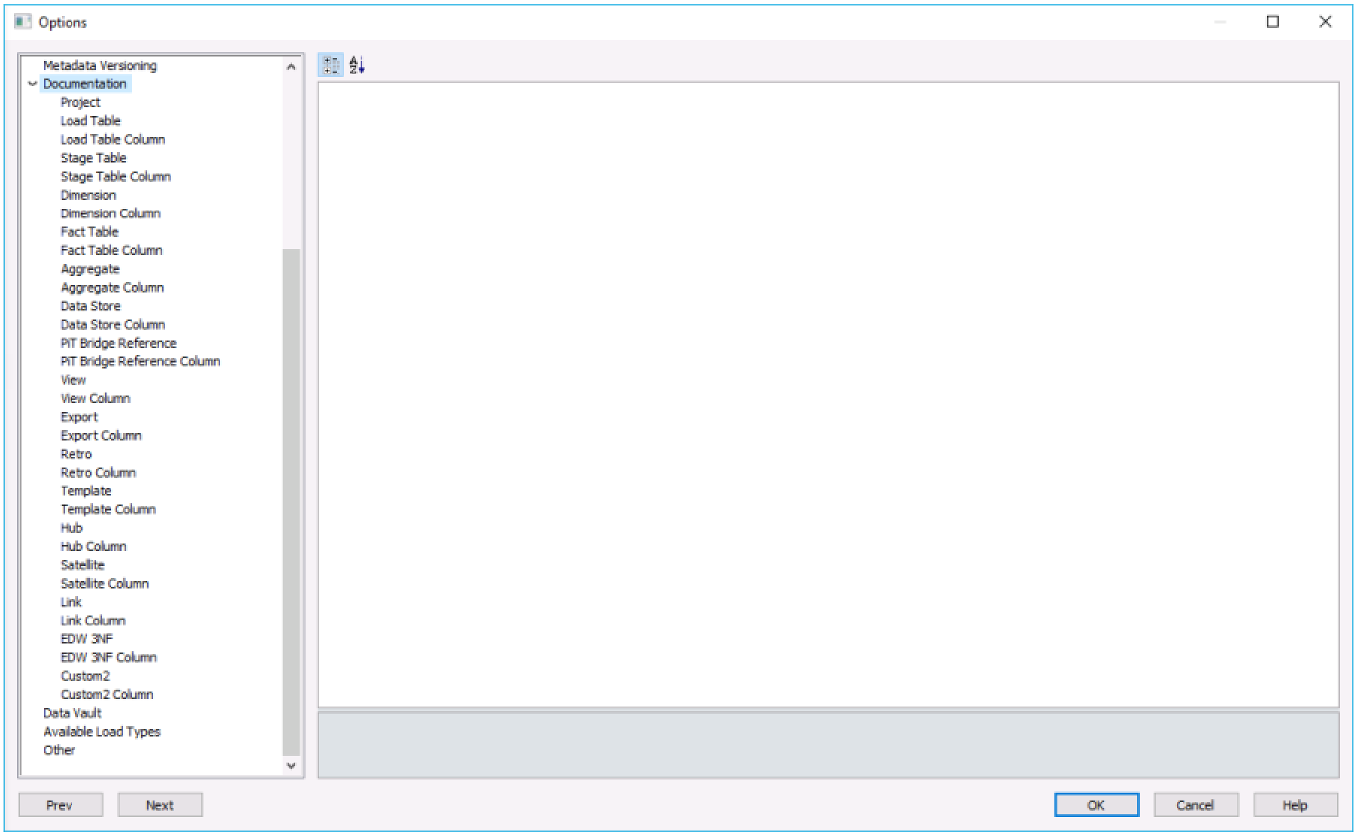
Metadata Versioning

Auto-Version before Table Delete	☑
Auto-Version before Table Alter/Create	☑
Auto-Version before Procedure Delete	☑
Auto-Version before Procedure Compile	☑
Auto-Version before Script Delete	☑
Auto-Version before Script Save	☑
Auto-Version before Connection Delete	☑
Auto-Version before Connection Save	☑
Auto-Version before Index Delete	☑
Auto-Version before Template Delete	☑
Auto-Version before Template Save	☑

Metadata Versioning
Versioning options for metadata objects

Documentation

This options enables you to alter the **Documentation** settings.



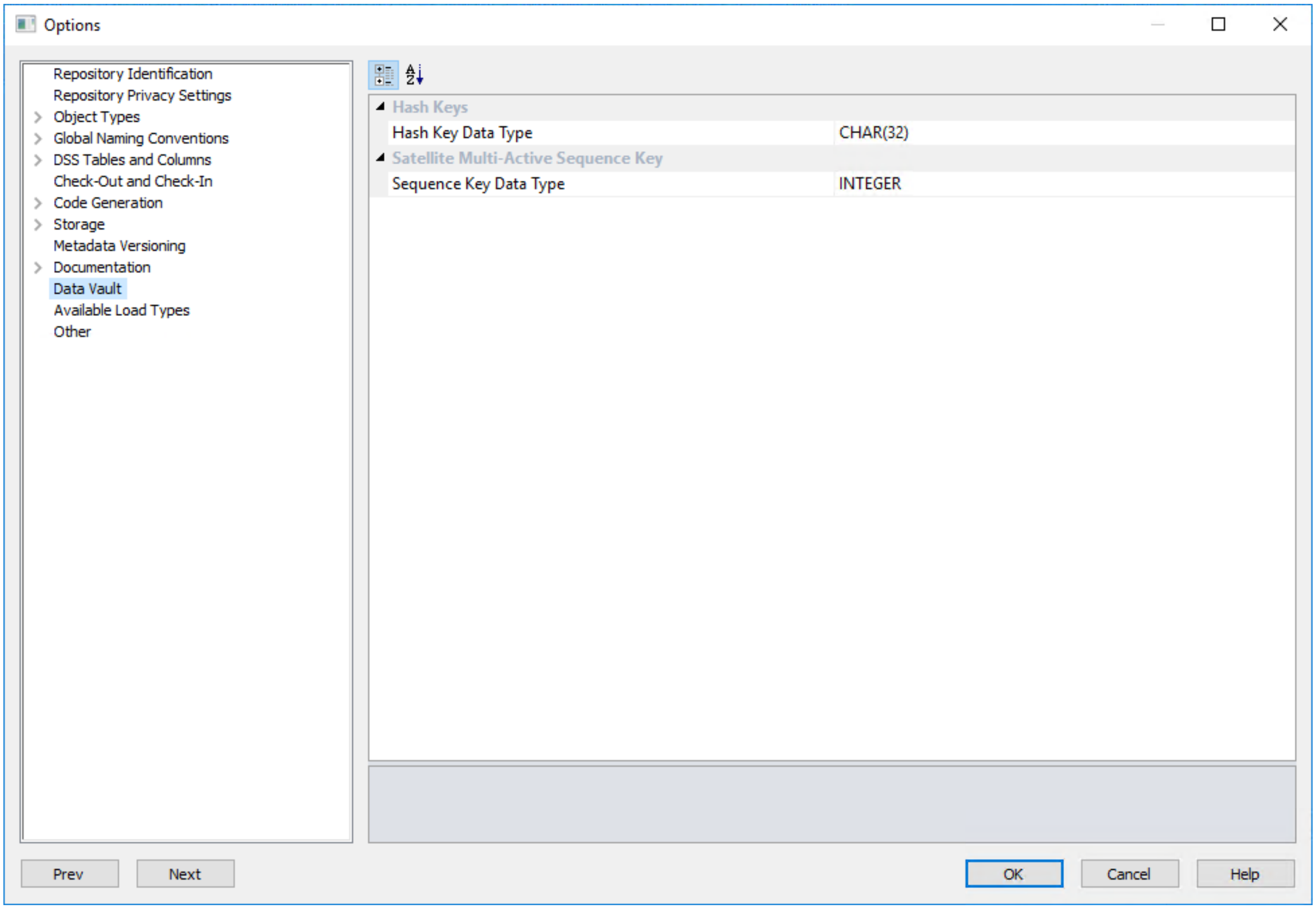
Project Documentation Field

Fields	Description
Documentation Name	Sets the name of the appropriate tab in the **Properties** screen.
Documentation Label	Sets the label or description of the appropriate documentation tab.
Documentation User	Defines if the documentation information is visible to end users and included in end user documentation.
Documentation Before Columns	Defines if the documentation tab information is shown in the documentation before or after the column information.
Documentation Order	Defines the field order on the **Properties** screen tabs.

Data Vault

This option enables you to define the data type to use for the change hash key columns generated in RED.

The generated Hash Keys are used to build the Hub, Link and Satellite objects in WhereScape RED. Refer to **Hash Key Generation Wizard** for details.



Hash Keys

Hash Key Data Type	CHAR(32)

Satellite Multi-Active Sequence Key

Sequence Key Data Type	INTEGER

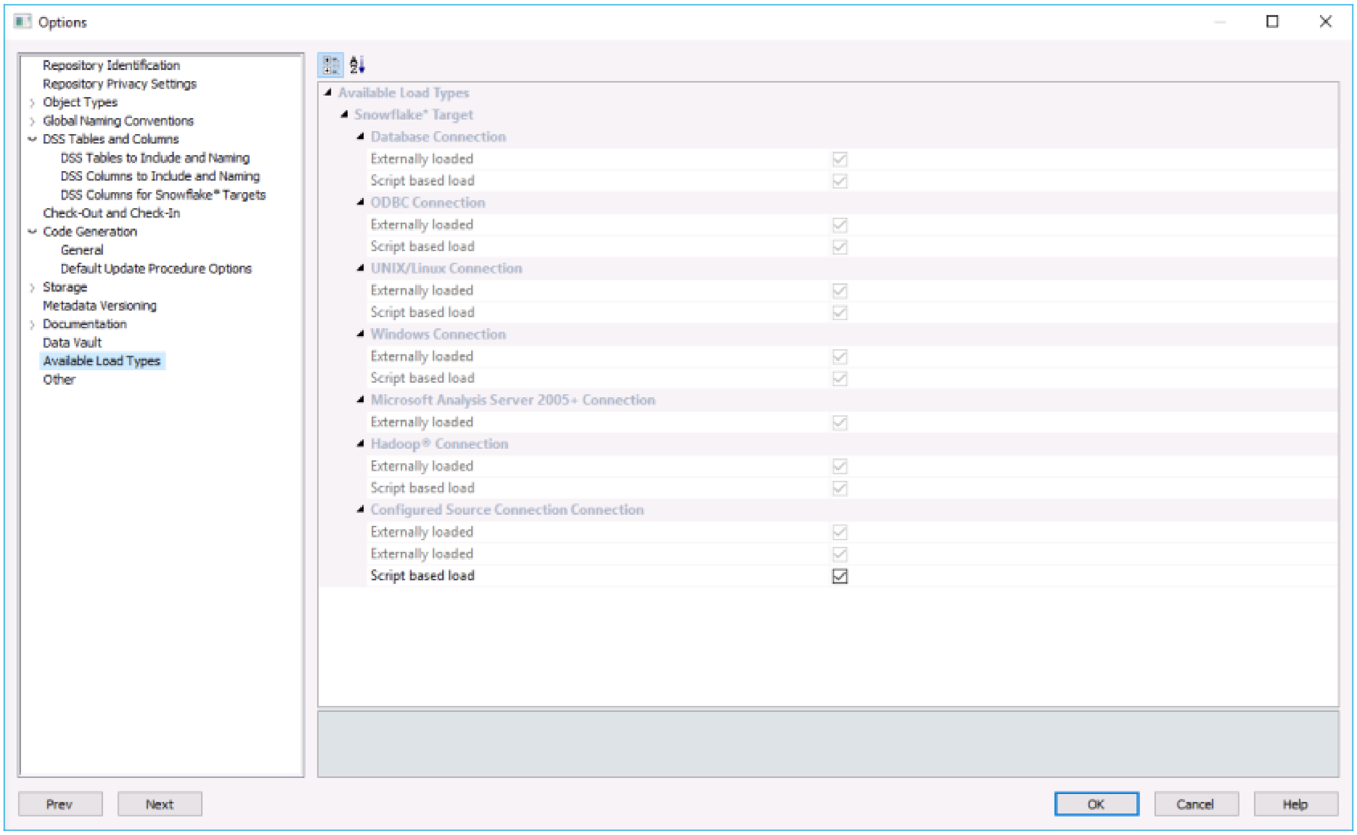
Note:

Changing this data type has no impact on any existing hash key columns.

Available Load Types

This option enables you to activate or deactivate the various load types available in RED, based on Target and Source database.

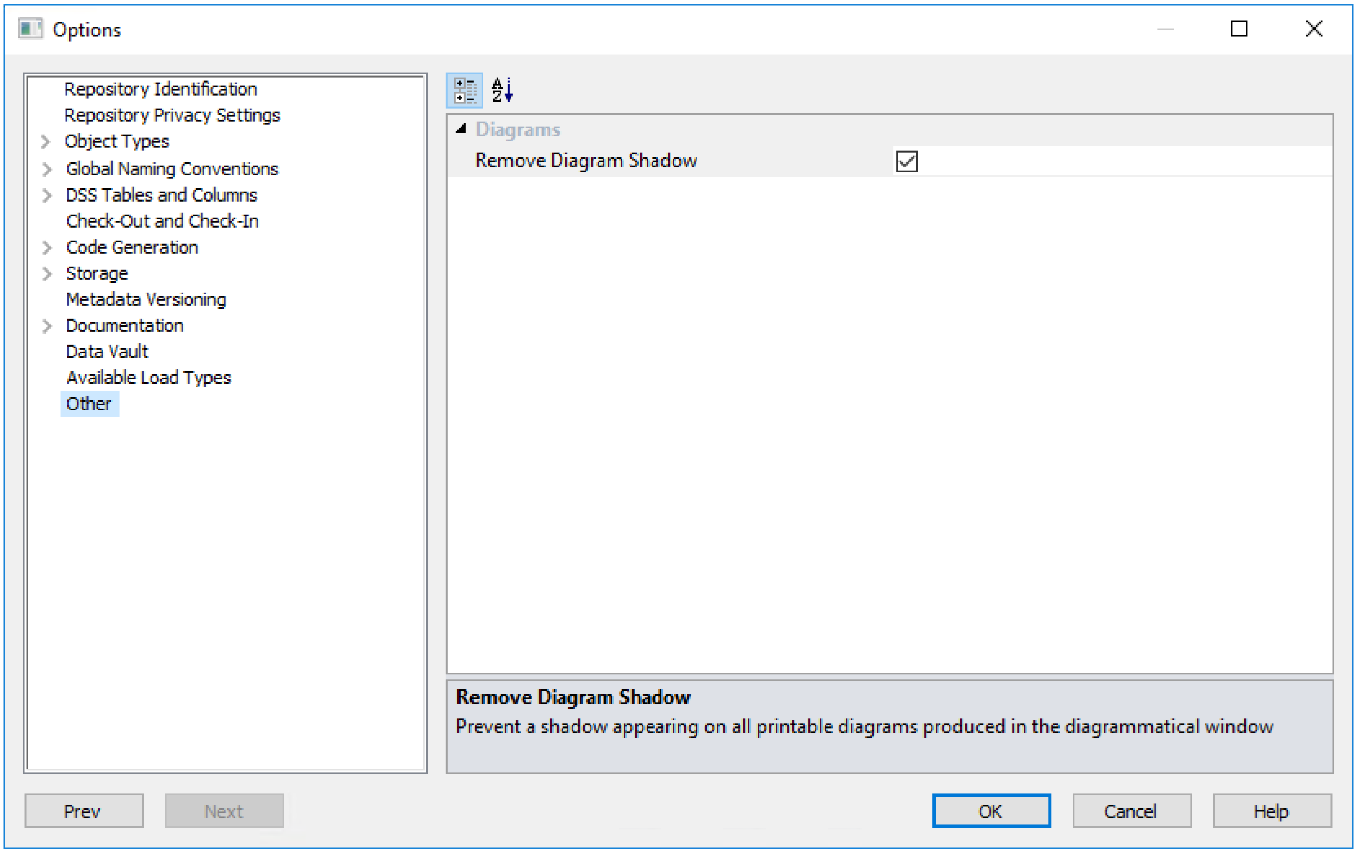
Select from the combination of load types supported by target and source databases, to control the default load type options that can be set by users from the Connection and the Load table **Properties** windows.


Options
Repository Identification
Repository Privacy Settings
Object Types
Global Naming Conventions
DSS Tables and Columns
DSS Tables to Include and Naming
DSS Columns to Include and Naming
DSS Columns for Snowflake® Targets
Check-Out and Check-In
Code Generation
General
Default Update Procedure Options
Storage
Metadata Versioning
Documentation
Data Vault
Available Load Types
Other
Prev
Next
OK
Cancel
Help

- Available Load Types
  - Snowflake® Target
    - Database Connection
      - Externally loaded ☑
      - Script based load ☑
    - ODBC Connection
      - Externally loaded ☑
      - Script based load ☑
    - UNIX/Linux Connection
      - Externally loaded ☑
      - Script based load ☑
    - Windows Connection
      - Externally loaded ☑
      - Script based load ☑
    - Microsoft Analysis Server 2005+ Connection
      - Externally loaded ☑
    - Hadoop® Connection
      - Externally loaded ☑
      - Script based load ☑
    - Configured Source Connection Connection
      - Externally loaded ☑
      - Externally loaded ☑
      - Script based load ☑

Other

This option enables users to add or remove shadows in the diagrams.



Remove Diagram Shadow ☑

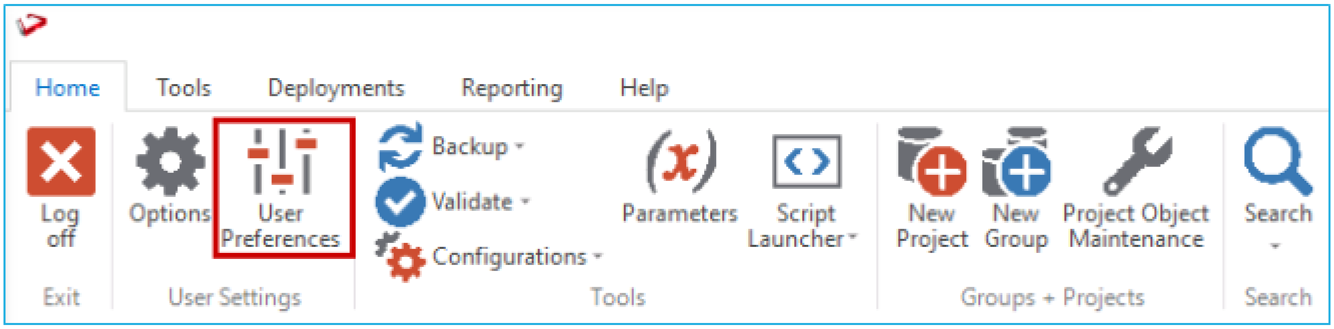
Remove Diagram Shadow
Prevent a shadow appearing on all printable diagrams produced in the diagrammatical window

Remove Diagram Shadow

Selected by default to prevent a shadow appearing on all printable diagrams produced in the **Diagram** window; else clear to change.

User Preferences

Select **User Preferences** from the **Home** toolbar.

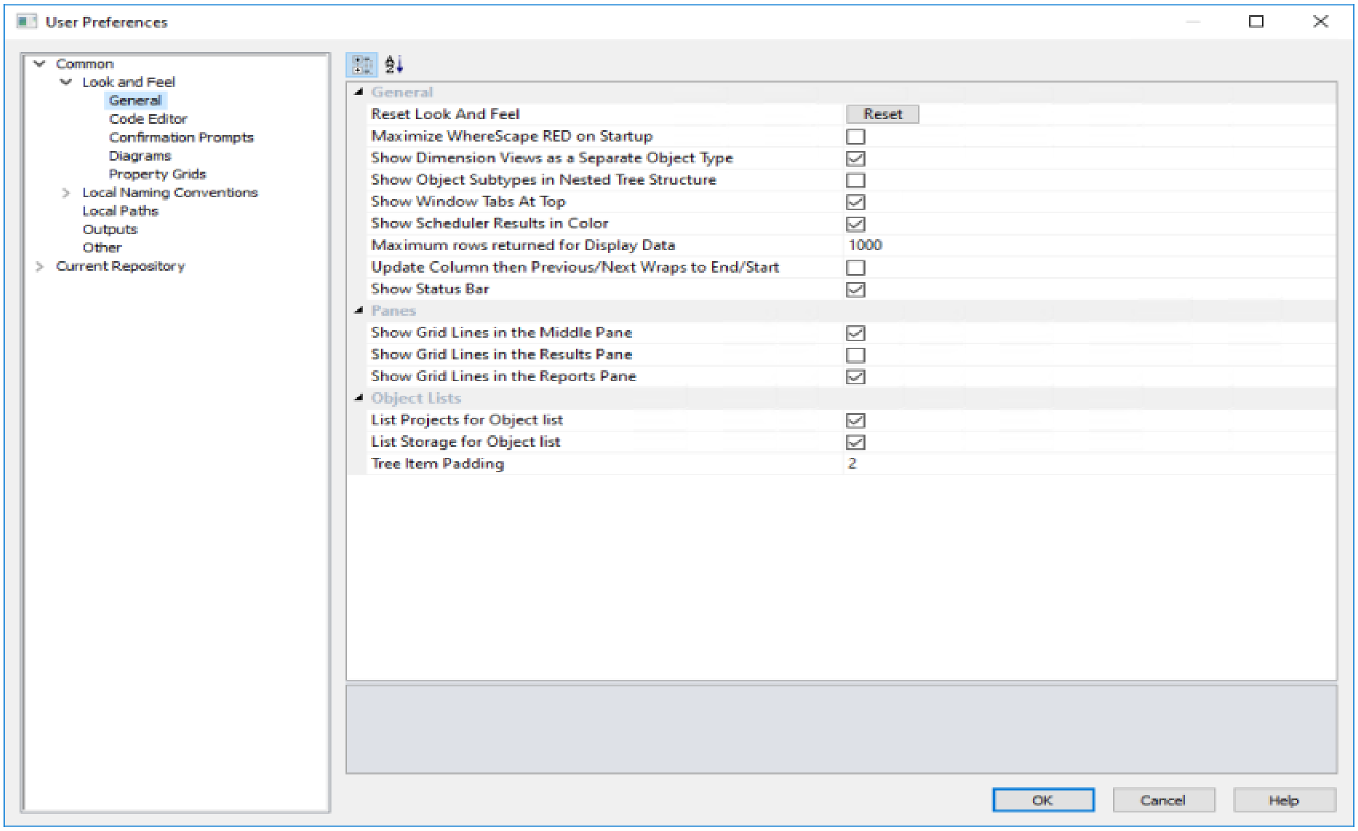


Common

Look and Feel

General

This option enables you to set the **Look and Feel** in **General**.



General

Reset Look And Feel	Reset
Maximize WhereScape RED on Startup	□
Show Dimension Views as a Separate Object Type	☑
Show Object Subtypes in Nested Tree Structure	□
Show Window Tabs At Top	☑
Show Scheduler Results in Color	☑
Maximum rows returned for Display Data	1000
Update Column then Previous/Next Wraps to End/Start	□
Show Status Bar	☑

Panes

Show Grid Lines in the Middle Pane	☑
Show Grid Lines in the Results Pane	□
Show Grid Lines in the Reports Pane	☑

Object Lists

List Projects for Object list	☑
List Storage for Object list	☑
Tree Item Padding	2

General

Options	Description
Reset Look And Feel	Enables you to reset all window tab positions for Builder and Scheduler panes. Resets scheduler and report headings.
Maximize WhereScape RED on Startup	Select the check box to start WhereScape RED in full screen mode.
Show Dimension Views as a Separate Object Type	Select the check box to display a new object group called Dimension Views where all view objects are stored. This option is selected by default.
Show Object Subtypes in Nested Tree Structure	Select the checkbox to separate object subtypes in all repository trees.
Show Window Tabs At Top	Select the check box to display window tabs at the top of the screen.
Show Scheduler Results in Color	Select the check box to turn on job status color coding in the scheduler. This option is selected by default.
Maximum rows returned for Display Data	This option sets the maximum number of rows that are returned when displaying data. This is set to 1000 by default.
Update Column then Previous/Next Wraps to End/Start	Select the check box to control the behavior of the directional **Update** buttons on the Column Properties window. When enabled, the **< Update** button will wrap to the last column when it moves beyond the first column; and the **Update >** button will wrap to the first column when it moves beyond the last column. When deactivated (default), the window closes after an attempt to navigate before the first column or after the last column.

Panes

Options	Description
Show Grid Lines in the Middle Pane	Select the check box to display grid lines in the main work area.
Show Grid Lines in the Results Pane	Select the check box to display grid lines in the results area.
Show Grid Lines in the Reports Pane	Select the check ox to display grid lines in the reports area.

Object Lists

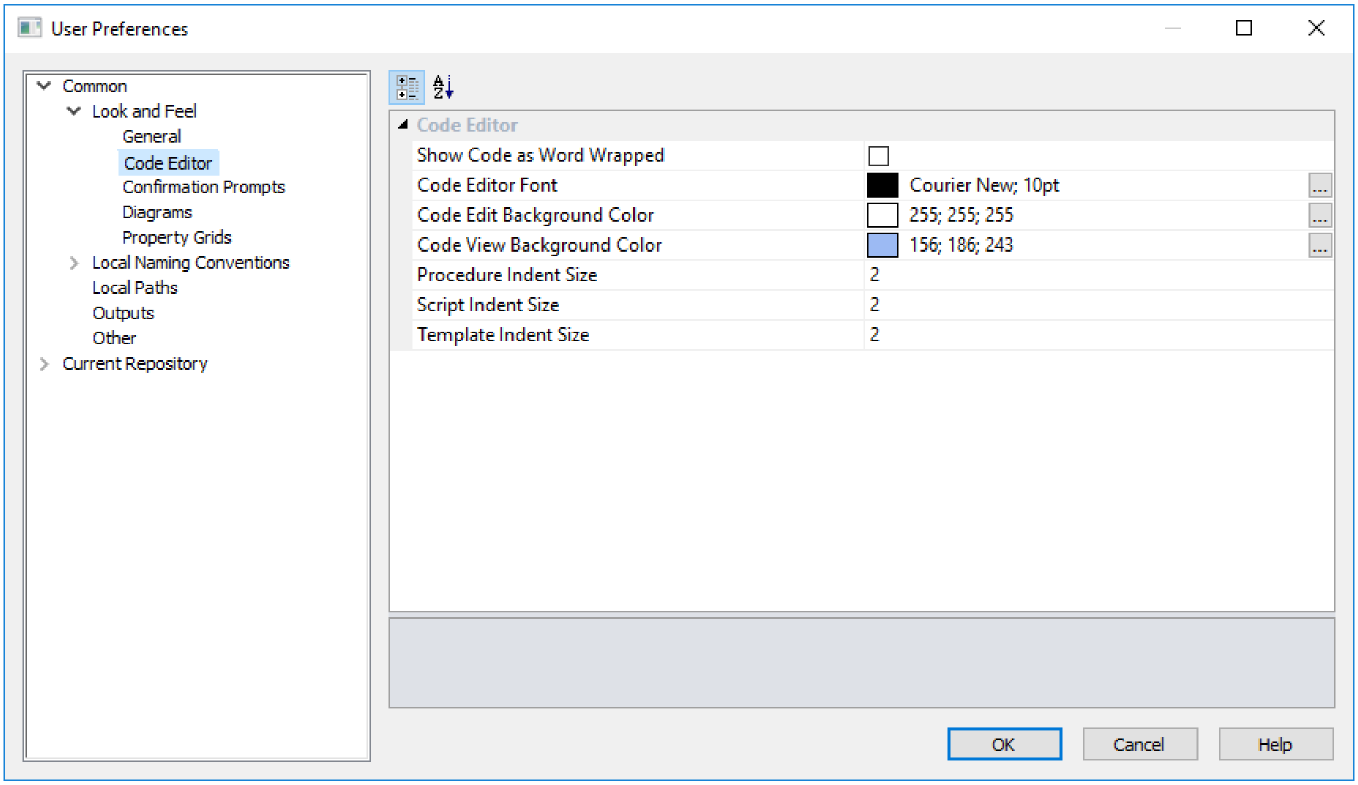
Options	Description
List Projects for Object list	Select the check box to display the projects for each object in the middle pane object list.
List Storage for Object list	Select the check box to display the storage for each object in the middle pane object list.
Tree Item Padding	Select the check box to select the number of pixels used to pad tree items when the tree items represent an Object; for example, in the Object Pane and the Browser Pane. Padding is added to the top and bottom of each tree item. Padding can be set from 0-10 pixels, default value is 2.

Note:

List Storage for Object list and **Tree Item Padding** options impact on the speed lists are generated. Since they are enabled by default, both options can be deactivated to speed up the process or if considered irrelevant according to users preferences.

Code Editor

This option enables you to set the **Look and Feel** in **Code Editor**.



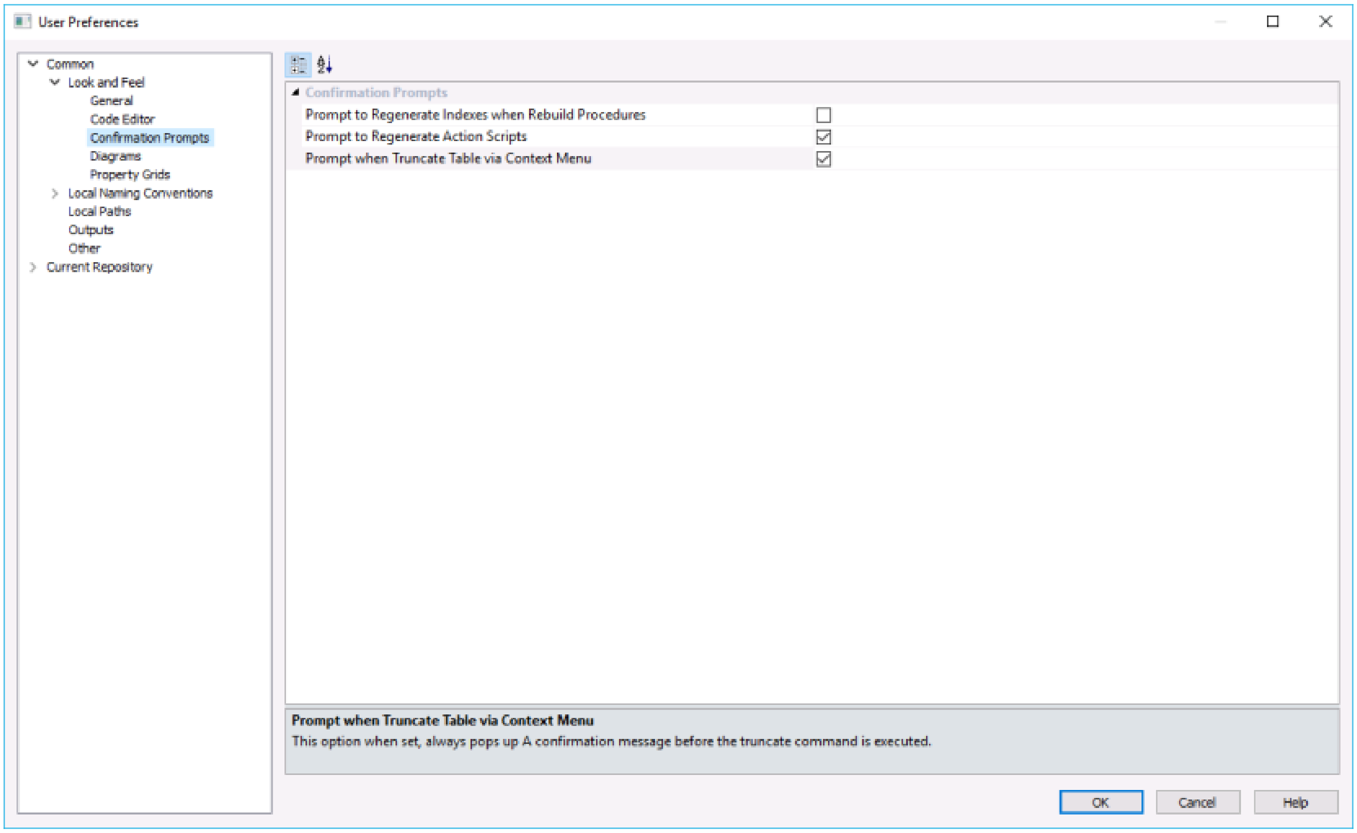
Code Editor

Show Code as Word Wrapped	□
Code Editor Font	Courier New; 10pt
Code Edit Background Color	255; 255; 255
Code View Background Color	156; 186; 243
Procedure Indent Size	2
Script Indent Size	2
Template Indent Size	2

Options	Description
Show Code as Word Wrapped	Enables you to have word wrapping applied to code by default.
Code Editor Font	Enables you to select the font used in the code editors.
Code Editor Background Color	Enables you to select the background color when editing code.
Code View Background Color	Enables you to select the background color when viewing code.
Procedure Indent Size	Enables you to specify the number of spaces that are generated when a TAB character is used within the Procedure editor. Permitted range is 2 through 10.
Script Indent Size	Enables you to specify the number of spaces that are generated when a TAB character is used within the Script editor. Permitted range is 2 through 10.
Template Indent Size	Enables you to specify the number of spaces that are generated when a TAB character is used within the Template editor. Permitted range is 2 through 10.

Confirmation Prompts

This option enables you to set the **Look and Feel** in **Confirmation Prompts**.



Prompt to Regenerate Indexes when Rebuild Procedures □

Prompt to Regenerate Action Scripts ☑

Prompt when Truncate Table via Context Menu ☑

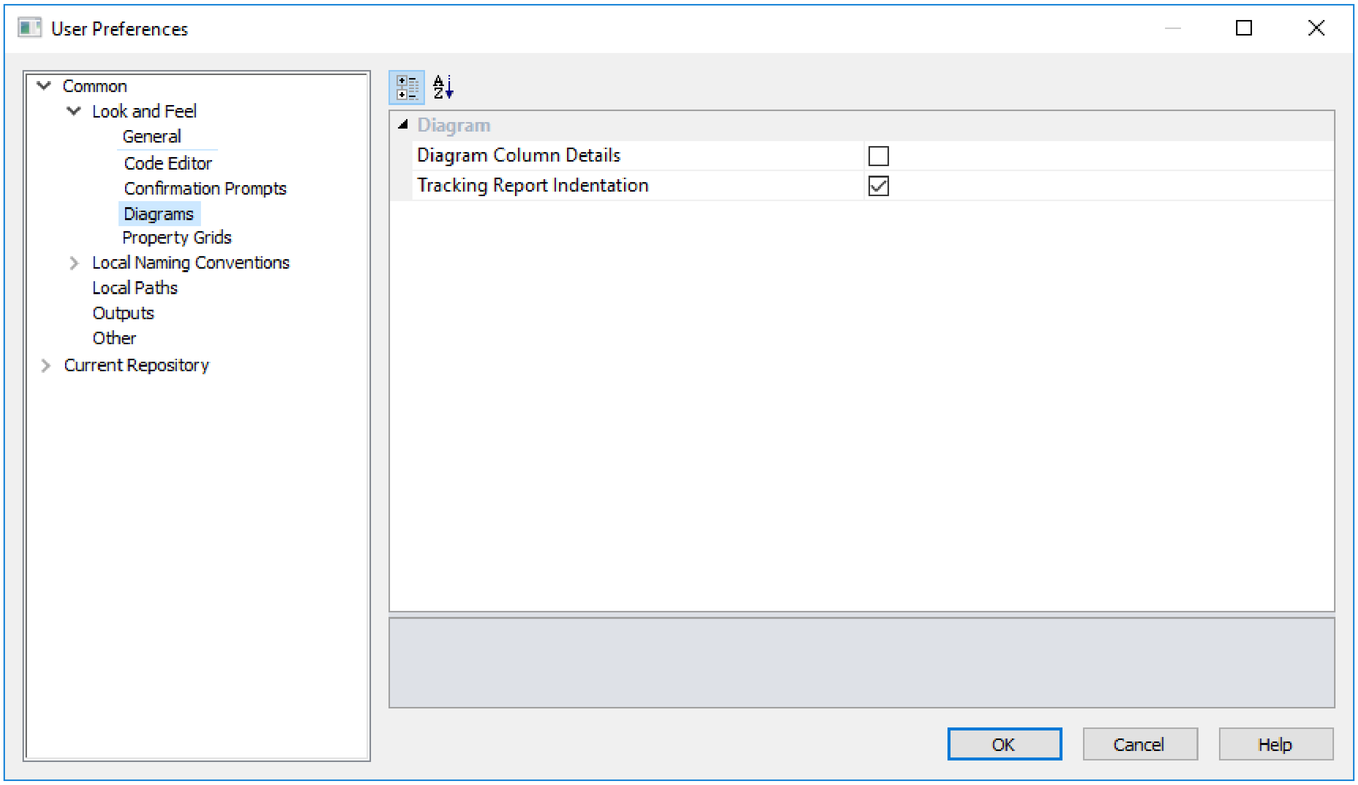
Prompt when Truncate Table via Context Menu

This option when set, always pops up A confirmation message before the truncate command is executed.

Options	Description
Prompt to Regenerate Indexes when Rebuild Procedures	When selected, always prompts for index regeneration whenever an update procedure is rebuilt.
Prompt to Regenerate Action Scripts	When selected, always prompts before Action Script regeneration whenever an Object's metadata has changed such that the Action Script and associated Object's Action Scripts need updating.
Prompt when Truncate Table via Context Menu	When selected, always pops up a confirmation message before the truncate command is executed.

Diagrams

This option enables you to set the **Look and Feel** in the **Diagrams**.



Diagram

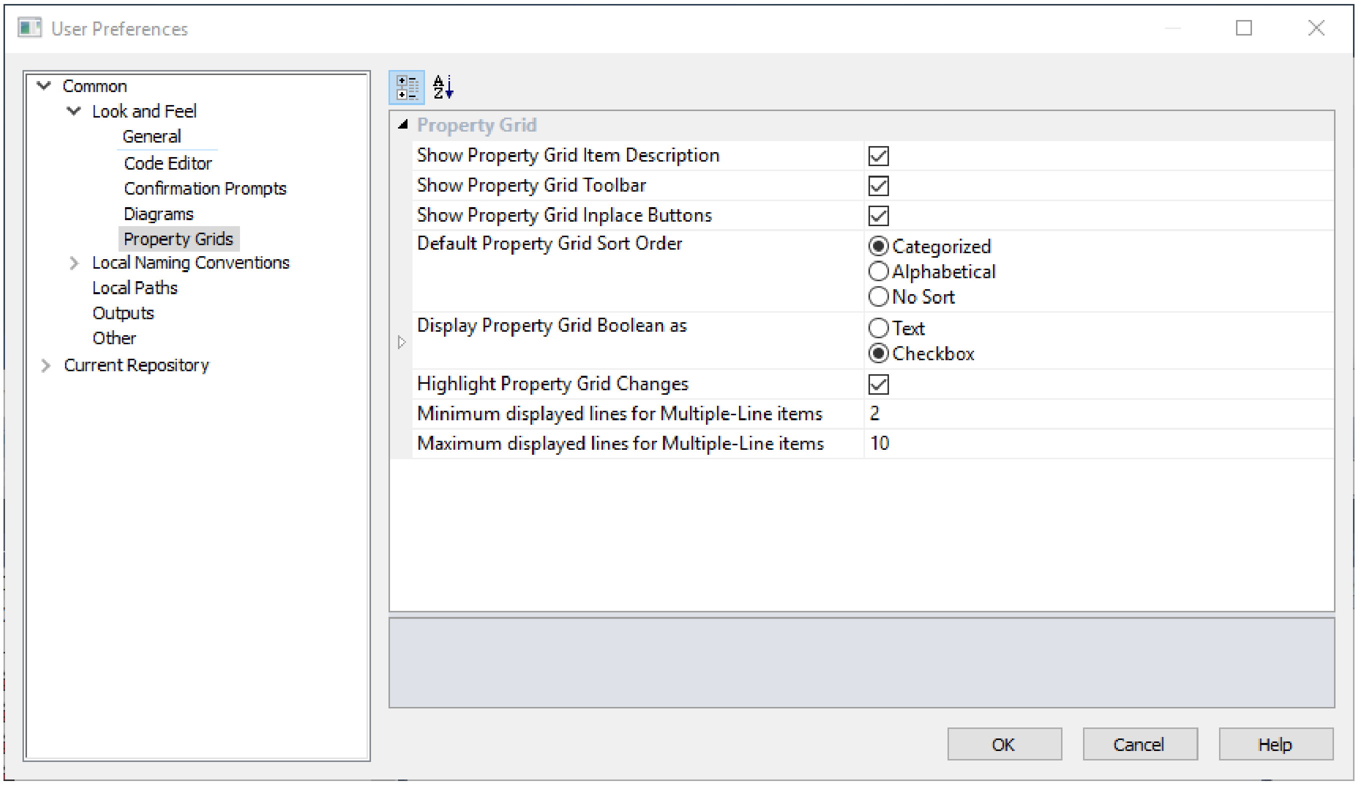
Diagram Column Details □

Tracking Report Indentation ☑

Options	Description
Diagram Column Details	Select the checkbox to display the columns as the initial diagram.
Tracking Report Indentation	Select the checkbox to include tabs in the output to show dependency level.

Property Grids

This option enables you to set the **Look and Feel** in **Property Grids**.



Property Grid

Show Property Grid Item Description	☑
Show Property Grid Toolbar	☑
Show Property Grid Inplace Buttons	☑
Default Property Grid Sort Order	● Categorized ○ Alphabetical ○ No Sort
Display Property Grid Boolean as	○ Text ● Checkbox
Highlight Property Grid Changes	☑
Minimum displayed lines for Multiple-Line items	2
Maximum displayed lines for Multiple-Line items	10

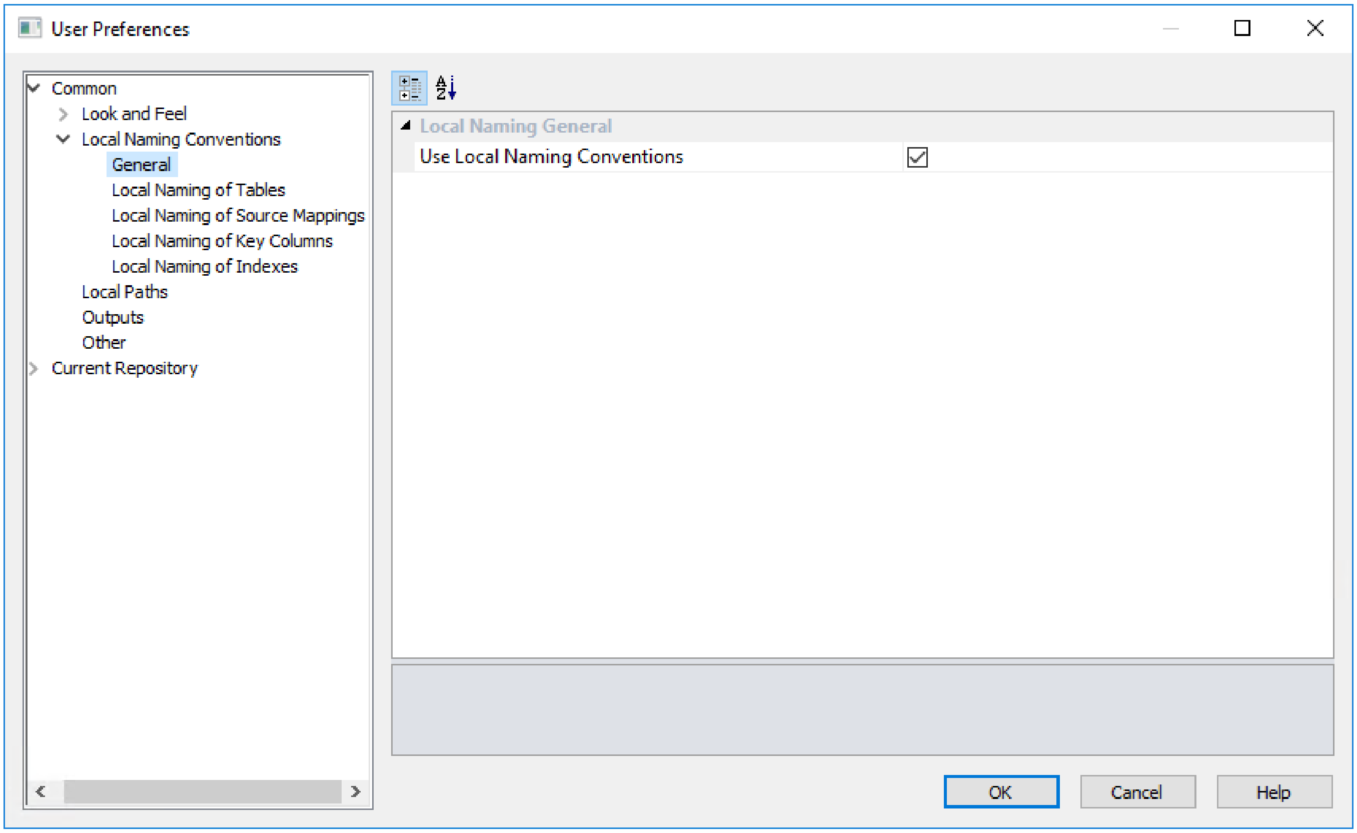
Options	Description
Show Property Grid Item Description	Select to display the property grid item description. The check box is selected by default.
Show Property Grid Toolbar	Select to display the property grid toolbar. The check box is selected by default.
Show Property Grid Inplace Buttons	Select to display property grid buttons for all items. The check box is selected by default.
Default Property Grid Sort Order	Enables you to select the default property grid sort order for items. The options are: • Categorized (default) • Alphabetical • No Sort
Display Property Grid Boolean as	Enables you to select how Boolean items are to be displayed. The options are: • Text (default) • Check box
Text for Boolean True	Enables you to enter the text for the Boolean value False.
Highlight Property Grid Changes	Enables you to highlight changed items in the property grid. The default is True.
Minimum displayed lines for Multiple-Line items	Sets the minimum display lines for multi-line inputs.
Maximum displayed lines for Multiple-Line items	Sets the maximum display lines for multi-line inputs.

Local Naming Conventions

The various options are described below.

General

This option enables you to set the **Local Naming Conventions**.



Use Local Naming Conventions ☑

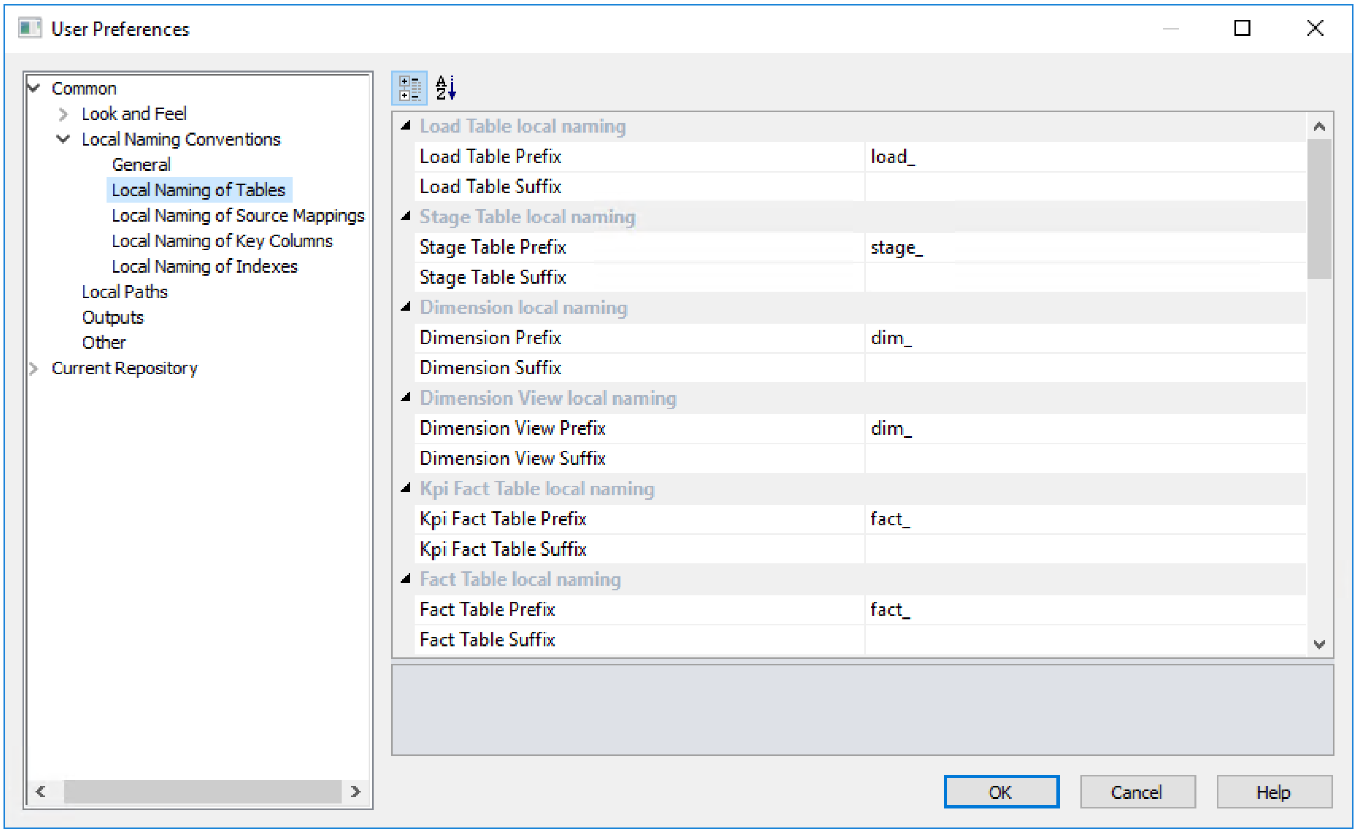
Options	Description
Use Local Naming Conventions	Select the checkbox to enable the **Local Naming of Tables**, **Key Columns** and **Indexes** options in the object tree.

Note

If this option is set, it can overwrite short names and object prefixes.

Local Naming of Tables

This option enables you to set the **Local Naming of Tables**.

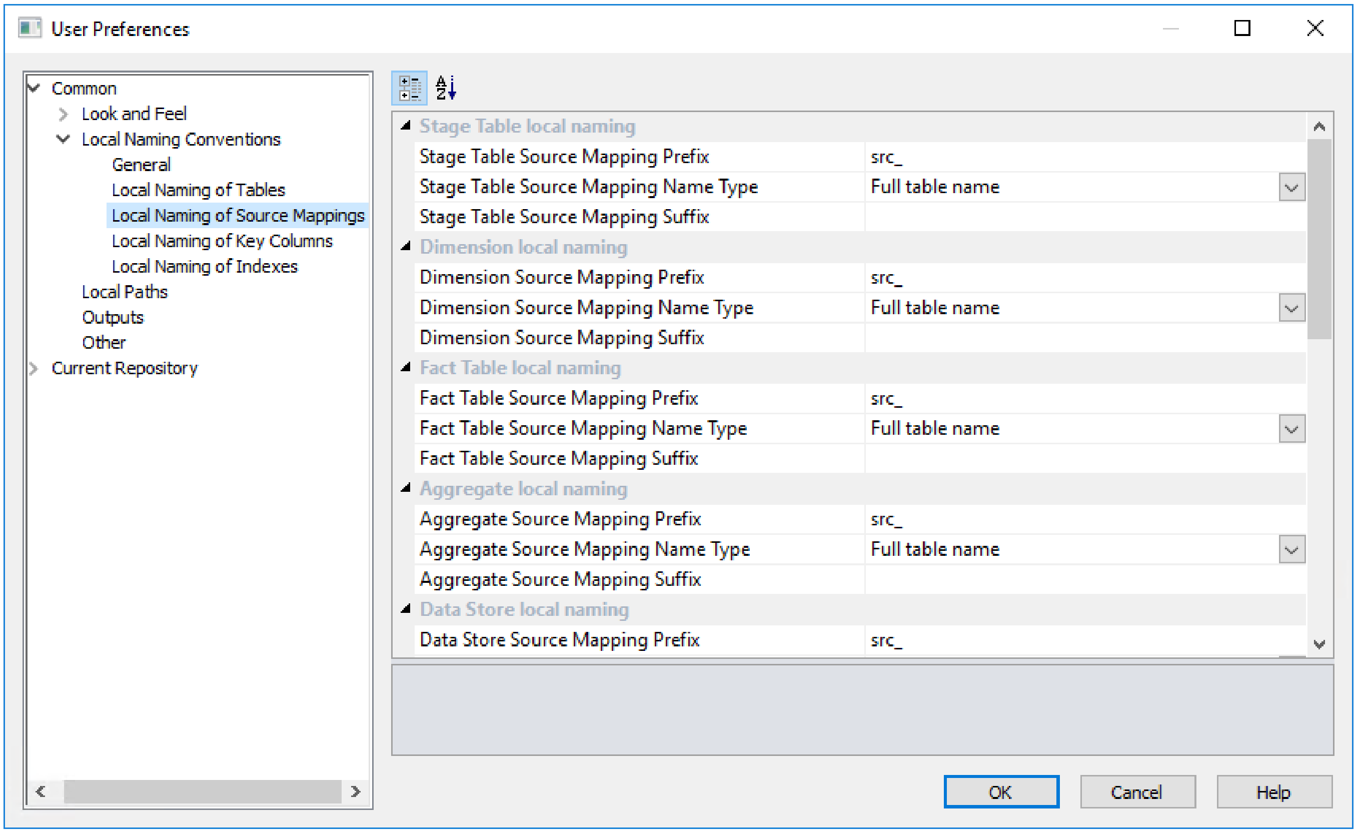


Load Table local naming	
Load Table Prefix	load_
Load Table Suffix	
Stage Table local naming	
Stage Table Prefix	stage_
Stage Table Suffix	
Dimension local naming	
Dimension Prefix	dim_
Dimension Suffix	
Dimension View local naming	
Dimension View Prefix	dim_
Dimension View Suffix	
Kpi Fact Table local naming	
Kpi Fact Table Prefix	fact_
Kpi Fact Table Suffix	
Fact Table local naming	
Fact Table Prefix	fact_
Fact Table Suffix	

Define the **prefix** and **suffix** used in the default naming convention for each table type.

Local Naming of Source Mapping

This option enables users to set the **Local Naming of Source Mappings** options.



Stage Table local naming

Stage Table Source Mapping Prefix	src_
Stage Table Source Mapping Name Type	Full table name
Stage Table Source Mapping Suffix	

Dimension local naming

Dimension Source Mapping Prefix	src_
Dimension Source Mapping Name Type	Full table name
Dimension Source Mapping Suffix	

Fact Table local naming

Fact Table Source Mapping Prefix	src_
Fact Table Source Mapping Name Type	Full table name
Fact Table Source Mapping Suffix	

Aggregate local naming

Aggregate Source Mapping Prefix	src_
Aggregate Source Mapping Name Type	Full table name
Aggregate Source Mapping Suffix	

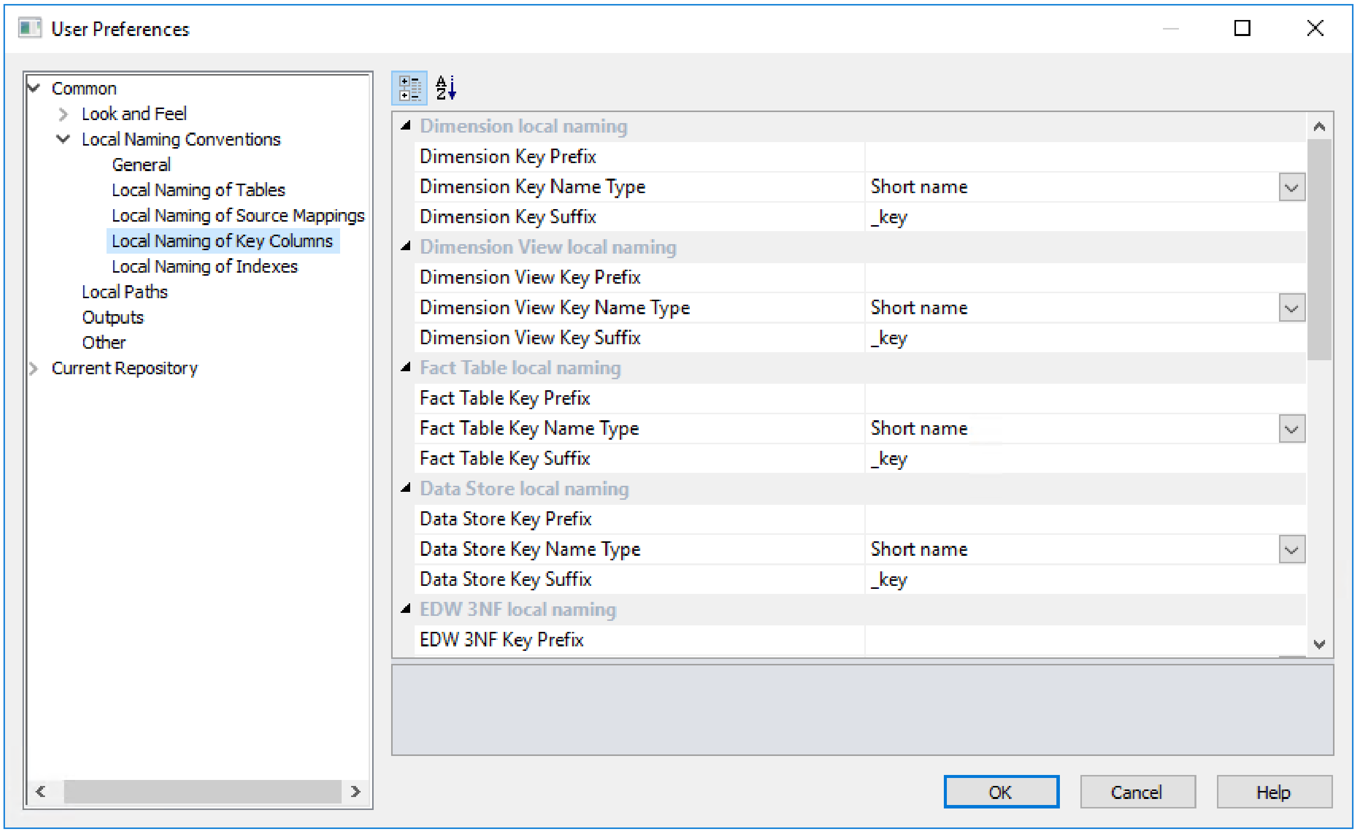
Data Store local naming

Data Store Source Mapping Prefix	src_

Options	Description
Source Mapping Prefix	Enables you to set the prefix used in the default source mapping naming convention.
Source Mapping Name Type	Enables you to set the basis for the source mapping naming.
Source Mapping Suffix	Enables you to set the suffix used in the default source mapping naming convention.

Local Naming of Key Columns

This option enables you to set the **Local Naming of Key Columns**.

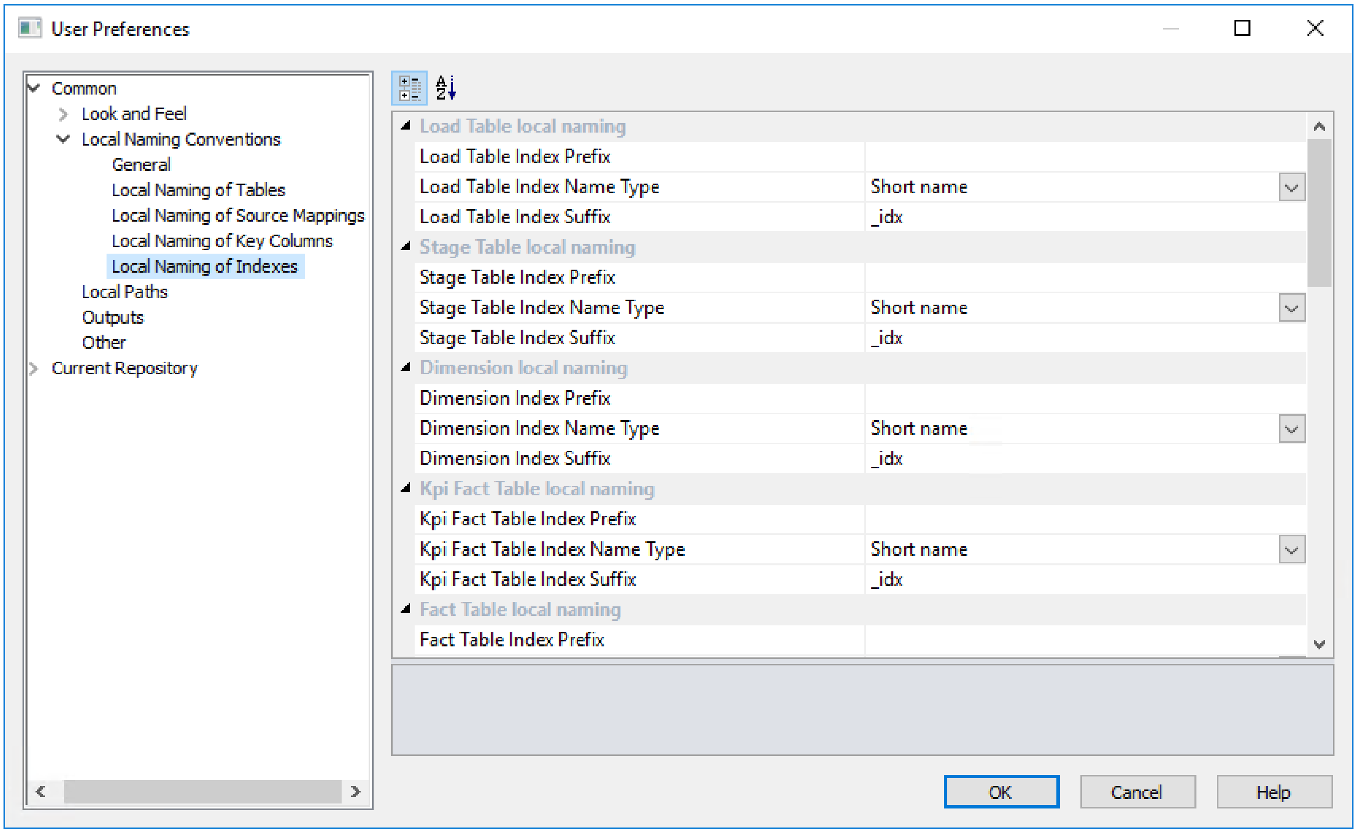


Dimension local naming	
Dimension Key Prefix	
Dimension Key Name Type	Short name
Dimension Key Suffix	_key
Dimension View local naming	
Dimension View Key Prefix	
Dimension View Key Name Type	Short name
Dimension View Key Suffix	_key
Fact Table local naming	
Fact Table Key Prefix	
Fact Table Key Name Type	Short name
Fact Table Key Suffix	_key
Data Store local naming	
Data Store Key Prefix	
Data Store Key Name Type	Short name
Data Store Key Suffix	_key
EDW 3NF local naming	
EDW 3NF Key Prefix	

Options	Description
Key Prefix	Sets the prefix used in the default key naming convention.
Key Name Type	Sets the basis for the key naming.
Key Suffix	Sets the suffix used in the default naming convention.

Local Naming of Indexes

This option enables you to set the **Local Naming of Indexes**.

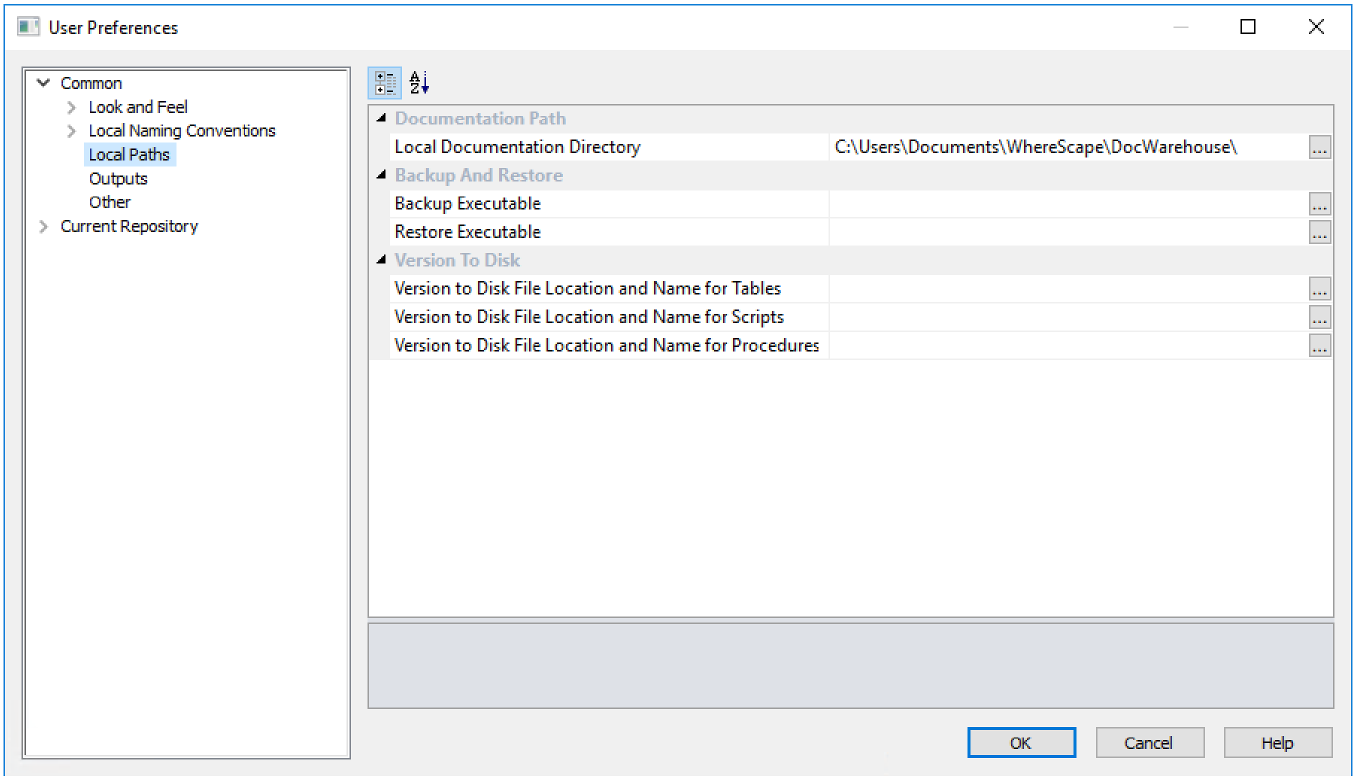


Load Table local naming	
Load Table Index Prefix	
Load Table Index Name Type	Short name
Load Table Index Suffix	_idx
Stage Table local naming	
Stage Table Index Prefix	
Stage Table Index Name Type	Short name
Stage Table Index Suffix	_idx
Dimension local naming	
Dimension Index Prefix	
Dimension Index Name Type	Short name
Dimension Index Suffix	_idx
Kpi Fact Table local naming	
Kpi Fact Table Index Prefix	
Kpi Fact Table Index Name Type	Short name
Kpi Fact Table Index Suffix	_idx
Fact Table local naming	
Fact Table Index Prefix	

Options	Description
Index Prefix	Enables you to set the prefix used in the default index naming convention.
Index Name Type	Enables you to set the basis for the index naming.
Index Suffix	Enables you to set the suffix used in the default index naming convention.

Local Paths

This option enables you to set the **Local Paths** for documentation, backup and restore and for versioning to disk.



Documentation Path

Local Documentation Directory C:\Users\Documents\WhereScape\DocWarehouse\

Backup And Restore

Backup Executable

Restore Executable

Version To Disk

Version to Disk File Location and Name for Tables

Version to Disk File Location and Name for Scripts

Version to Disk File Location and Name for Procedures

Documentation Path

Set the **Local Documentation Directory**.

Options	Description
Local Documentation Directory	Sets the Local Documentation Directory

Backup And Restore

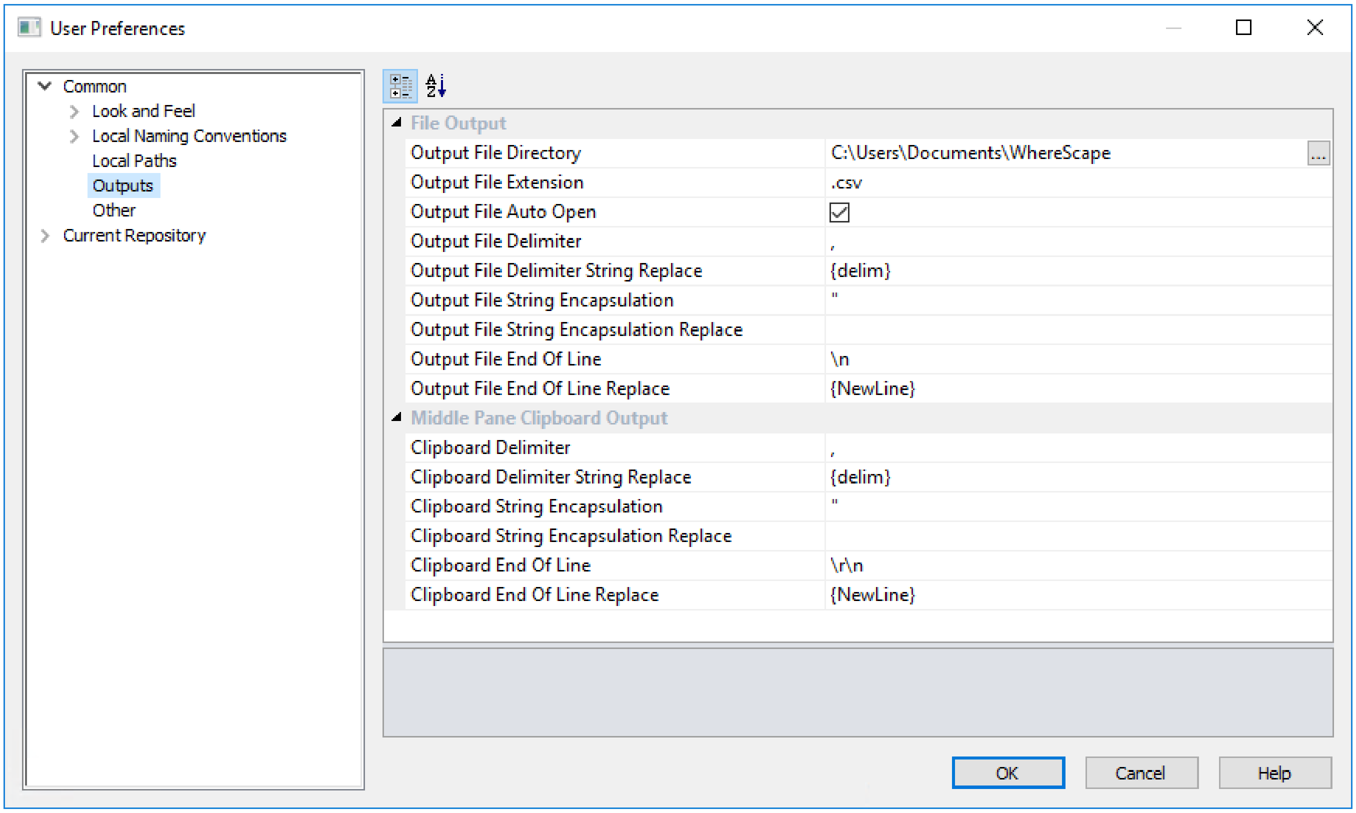
Options	Description
Backup Executable	Sets the override for backup executable. By default WhereScape RED tries to find the path of the backup executable. This edit box provides the ability to specify the exact location and name of the executable. This is useful when WhereScape RED cannot find the program or if there are multiple versions of the program on the PC.
Restore Executable	Sets the override for restore executable. By default, WhereScape RED tries to find the path of the restore executable.This edit box provides the ability to specify the exact location and name of the executable. This is useful when WhereScape RED cannot find the program or if there are multiple versions of the program on the PC.

Version to Disk

Options	Description
Versions to Disk options	Sets the locations and names. If any of the three version to disk paths are set, WhereScape RED automatically creates ASCII files containing the applicable DDL or code, each time an automated version occurs in the entered directory.

Outputs

This option enables you to set the **Output** user preferences.


User Preferences

File Output

Output File Directory	C:\Users\Documents\WhereScape
Output File Extension	.csv
Output File Auto Open	☑
Output File Delimiter	,
Output File Delimiter String Replace	{delim}
Output File String Encapsulation	"
Output File String Encapsulation Replace	
Output File End Of Line	\n
Output File End Of Line Replace	{NewLine}

Middle Pane Clipboard Output

Clipboard Delimiter	,
Clipboard Delimiter String Replace	{delim}
Clipboard String Encapsulation	"
Clipboard String Encapsulation Replace	
Clipboard End Of Line	\r\n
Clipboard End Of Line Replace	{NewLine}

File Output

Options	Description
Output File Directory	Enables you to set the path for output files created from the middle pane.
Output File Extension	Select the checkbox to set the file extension for output files created from the middle pane. This value determines the program that will auto open files.
Output File Auto Open	Select the checkbox to automatically open the results in files created from the middle pane.
Output File Delimiter	Enables you to set the characters that separate each field within each record of output files created from the middle pane. Common values are , and |.
Output File Delimiter String Replace	Enables you to set the characters that will replace the delimiter character if it occurs inside a field.
Output File String Encapsulation	Enables you to set the characters that are used to enclose string values of files created from the middle pane. Common values are " and '.
Output File String Encapsulation Replace	Enables you to set the characters that will replace the encapsulation string if it occurs inside a field.
Output File End Of Line	Enables you to set the characters saved at the end of each record of files created from the middle pane. Common values are \n, \r and \t.
Output File End Of Line Replace	Enables you to set the characters that will replace the end of line string if it occurs inside a field.

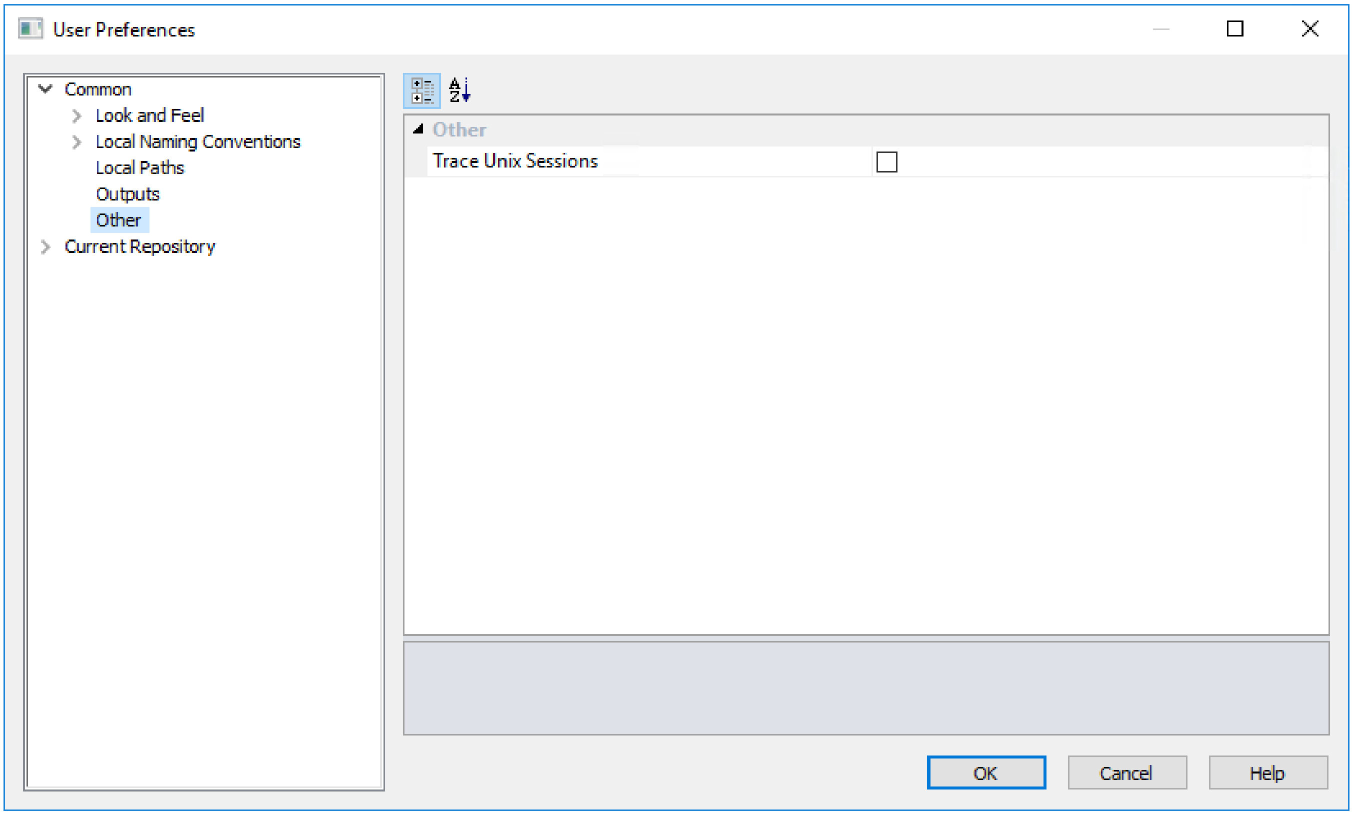
Middle Pane Clipboard Output

Options	Description

Options	Description
Clipboard Delimiter	Enables you to set the characters that separate each field within each record of clipboard output created from the middle pane. Common values are , and |.
Clipboard Delimiter String Replace	Enables you to set the characters that will replace the delimiter character if it occurs inside a field.
Clipboard String Encapsulation	Enables you to set the characters that are used to enclose string values of clipboard output created from the middle pane. Common values are " and '.
Clipboard String Encapsulation Replace	Enables you to set the characters that will replace the encapsulation string if it occurs inside a field.
Clipboard End of Line	Enables you to set the characters saved at the end of each record of clipboard output created from the middle pane. Common values are \n, \r and \t.
Clipboard End of Line Replace	Enables you to set the characters that will replace the end of line string if it occurs inside a field.

Other

This option enables you to set the **Other** user preferences.

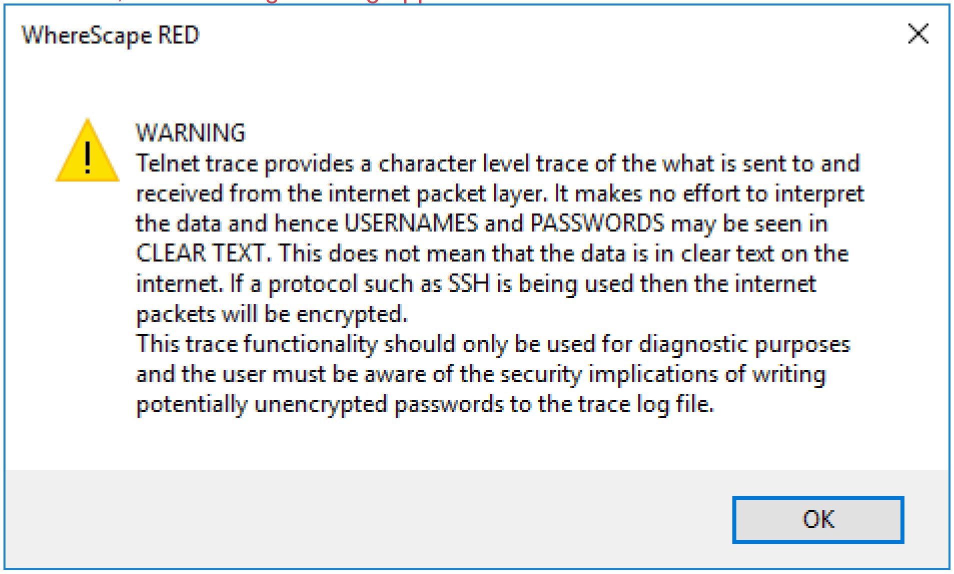


Trace Unix Sessions □

Options	Description
Trace Unix Sessions	Select the checkbox to trace all Unix activity undertaken by the WhereScape RED program until it is terminated. The file WslMedTelnet.txt is created in the program directory for WhereScape RED. **Tip** This option is intended for the debugging of specific Unix problems and the setting of this switch would normally be done at the request of WhereScape when attempting to solve a

<table>
<tr><th>Options</th><th>Description</th></tr>
<tr><td></td><td>Telnet issue. This setting is only relevant for the PC on which the setting is made (e.g. it is not a global setting for the repository).

Note

When set, the following warning appears:

WhereScape RED

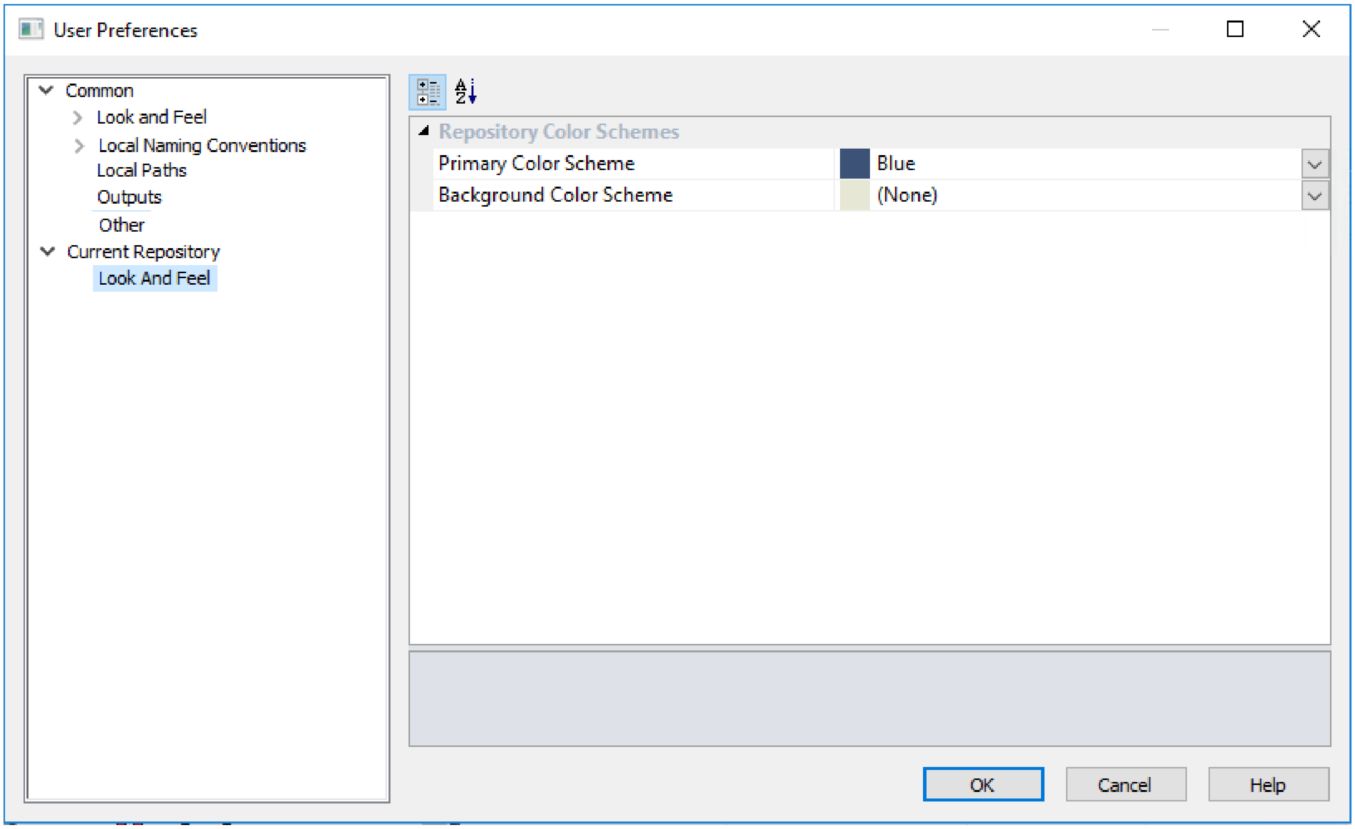
WARNING
Telnet trace provides a character level trace of the what is sent to and received from the internet packet layer. It makes no effort to interpret the data and hence USERNAMES and PASSWORDS may be seen in CLEAR TEXT. This does not mean that the data is in clear text on the internet. If a protocol such as SSH is being used then the internet packets will be encrypted.
This trace functionality should only be used for diagnostic purposes and the user must be aware of the security implications of writing potentially unencrypted passwords to the trace log file.

 </td></tr>
</table>

Current Repository

Look and Feel

This option enables you to set the **Look and Feel** for the **Current Repository**.



Primary Color Scheme	Blue
Background Color Scheme	(None)

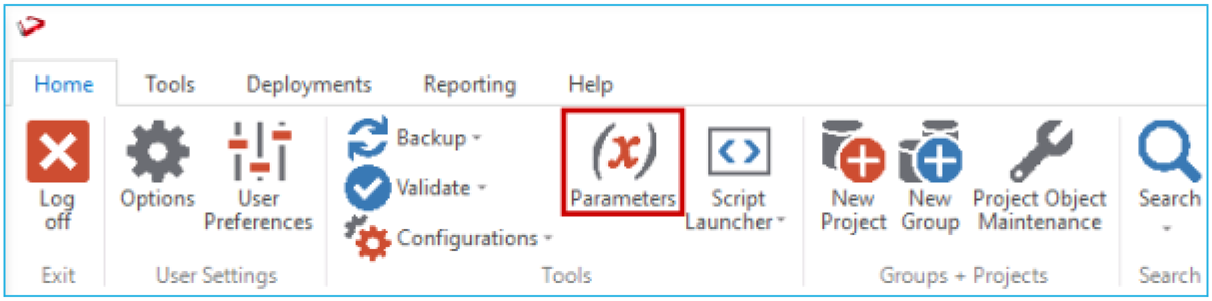
Options	Description
Repository Color Scheme	Enables you to set the primary and background **Color Schemes** for the current repository.

Parameters

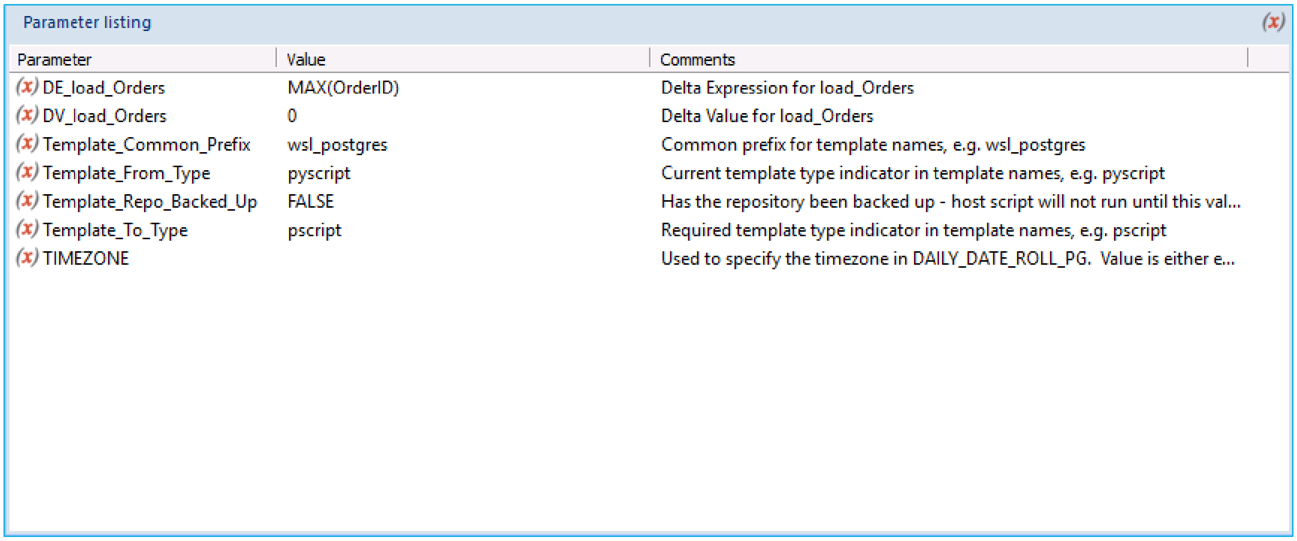
Parameters are a means of passing information between two or more procedures and between the WhereScape RED environment and procedures.

They can be accessed within the WhereScape RED environment in two ways:

1. **Home > Parameters** menu option:



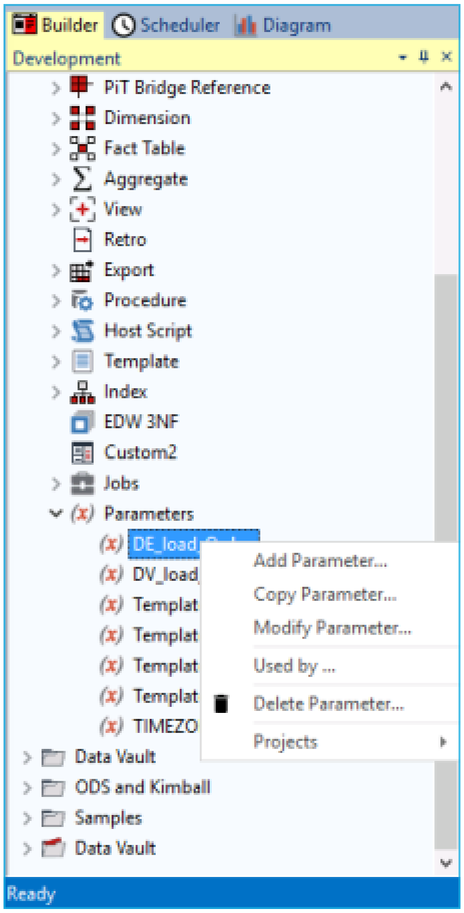
A list of parameters is displayed as per the example below:



Parameter listing

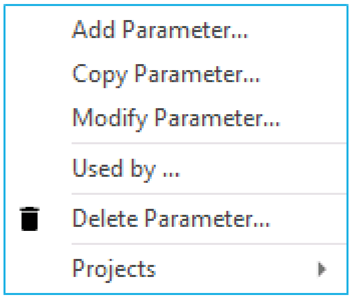
Parameter	Value	Comments
DE_load_Orders	MAX(OrderID)	Delta Expression for load_Orders
DV_load_Orders	0	Delta Value for load_Orders
Template_Common_Prefix	wsl_postgres	Common prefix for template names, e.g. wsl_postgres
Template_From_Type	pyscript	Current template type indicator in template names, e.g. pyscript
Template_Repo_Backed_Up	FALSE	Has the repository been backed up - host script will not run until this val...
Template_To_Type	pscript	Required template type indicator in template names, e.g. pscript
TIMEZONE		Used to specify the timezone in DAILY_DATE_ROLL_PG. Value is either e...

2. **Objects pane > Parameter** objects list:



Managing Parameters

A parameter can be added, edited, copied or deleted using the right-click context menu of the Parameter column:



Parameters can also be added or removed from Projects you have defined in RED using the **Projects** context menu option. Refer to **Organizing Objects** for details .

Parameter Usage

Typical parameter usage may be the global definition of how many days should be looked back for changed data, a month or processing period, etc.

Parameters can be used in **Load** tables to place limits in a 'Where' clause, etc. Refer to **Database Link Load - Source Mapping** for details.

They are also used by **Stage table procedures** as variables. Refer to **Generating the Staging Update Procedure** for details.

- **SQL Serverr**: Two procedures are provided to allow procedures to read and write parameters. These procedures are **WsParameterRead** and **WsParameterWrite**. Using these procedures, a procedure can load and use the contents of a parameter, or modify an existing parameter, or add a new parameter. A function **WsParameterReadF** is also provided. This function will return the value of a parameter.

Global Parameters

A number of global parameters are provided when loading tables, either via WhereScape RED or via the scheduler. These global parameters are accessed by calling the function **WsParameterReadG**.

The Load table name and the source table name can be acquired by using this function.

Refer to **Callable Routines** for more information on **WsParameterRead**, **WsParameterWrite**, **WsParameterReadF** and **WsParameterReadG**.

Connections

Connection objects serve several purposes in WhereScape RED:

1. They are used to browse potential source data in source systems and to acquire metadata. Potential source data includes database tables and flat files.

   For **database tables**, WhereScape RED:

   - Uses the **ODBC Source** set on each connection to browse the source system.
   - Acquires the metadata for new **load tables** built from the source system using drag and drop.

   For **files**, WhereScape RED:

   - Connects directly to Windows, UNIX/Linux and Hadoop to analyze the source file for the new load table and to acquire its metadata.
   - Prompts for user input for any metadata not available in the source file.

     Notes:

     - ODBC connections must be either **User DSN** or **System DSN**. **File DSN** connections are **not** supported.
     - **Windows** and **UNIX** connections do not have an **ODBC Source** property. **UNIX** connections are used for UNIX and Linux systems.

2. Load tables with a connection of **Connection type ODBC** extract data from source systems using ODBC. The **ODBC Source** of the connection is the ODBC DSN used for the extract.

   Note:

   In Teradata environments, if a **Teradata TPT compliant ODBC DSN** is defined in the **Connection Properties**, the **TPT DSN** is used for TPT ODBC Loads.

3. Each data warehouse metadata repository must have a Data Warehouse connection to use drag and drop to create new objects (other than load tables) in the data warehouse. WhereScape RED:
   - Uses the **ODBC Source** set on the Data Warehouse connection to browse the Data Warehouse database.
   - Acquires the metadata for any **tables** built from existing data warehouse tables.

     Note:

     This connection always has a **Connection type** of **Database**.

4. Cube objects require a connection to define the Analysis Services server used to create and load cubes. This is a connection with a **Connection type** of **Microsoft Analysis Server 2005+.**
5. Export objects require a connection to define the target environment where exported data is written. This is a connection with a **Connection type** of **UNIX** or **Windows.**

Connection Types

Connection properties are described in this topic as they apply to the supported database types in WhereScape RED.
Connection types can be set up via the following methods:

- Connection to the Data Warehouse/Metadata repository - Database type connections
- Connections to another Database - Database type connections on the same server or through a linked server
- ODBC Based Connections - ODBC type connections with Native ODBC, ODBC, Externally loaded or Integration Services load types
- Connections to Windows
- Connections to UNIX/Linux

- Extensible Source Connection

The following icon is used to indicate unique properties to a particular database type:

(Custom)

Note

Custom type database connections are assigned a unique name during the licensing process. In this document, the term 'Custom' refers to all custom database connections.

Database - Data Warehouse/Metadata Repository

This topic describes the details of the connection Properties as they apply to **Database** type connections and specifically of the Data Warehouse or Metadata repository connection.
The Data Warehouse connection is the connection that stores the metadata repository and it is the connection that is used in the drag and drop functionality to create the Dimension, Stage, Fact, and Aggregate tables.

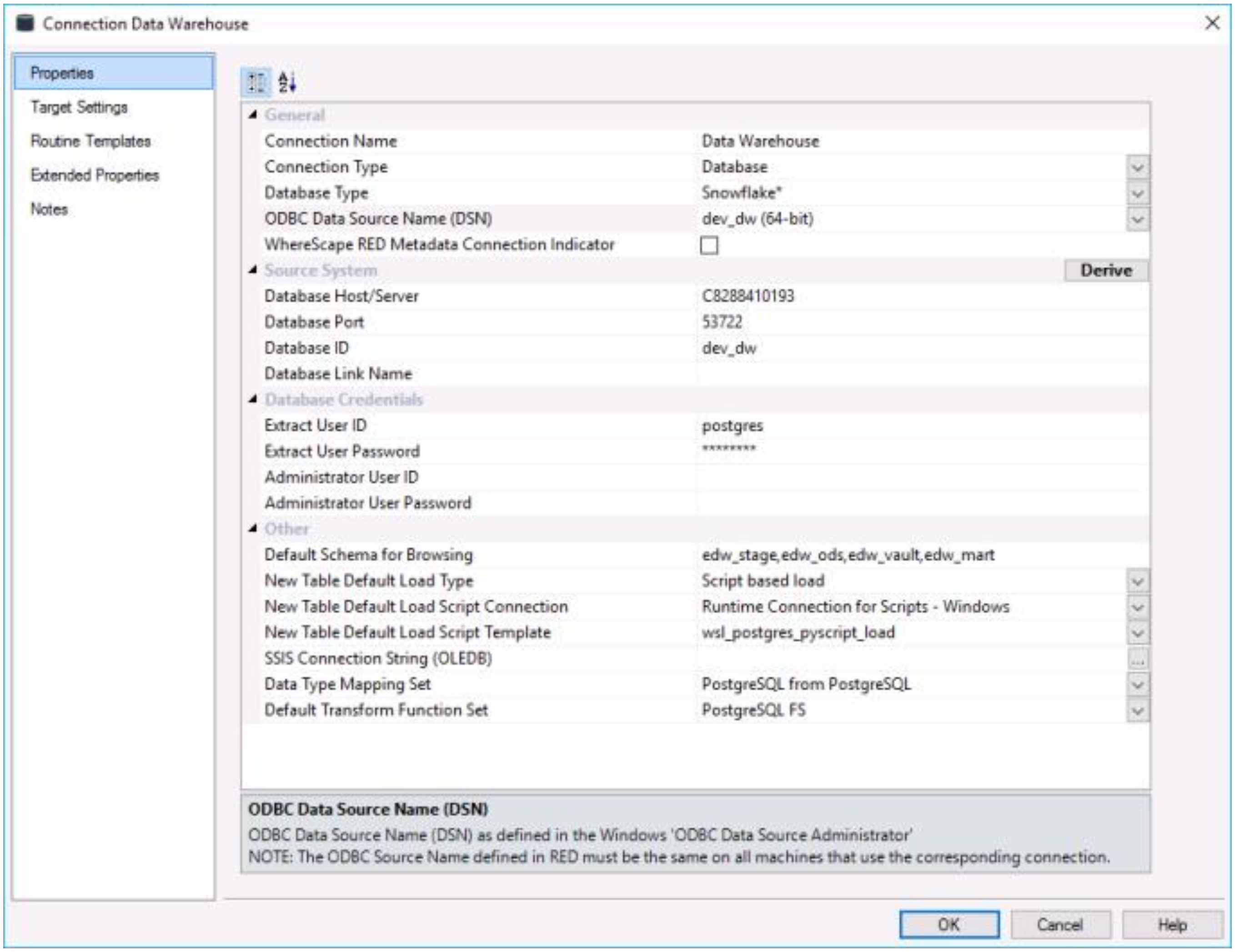
This connection is also used to create Cubes.

Connection types also impact the available load methods.

Note:

The **Data Warehouse connection** must exist if you wish to use drag and drop to create Dimensions, Stage tables, Fact tables, Aggregates, and Cubes.

Apart from the Data Warehouse or Repository connection to SQL Server, there must be at least one other connection to the Target Data warehouse database, such as Greenplum, Netezza or PDW.



General	
Connection Name	Data Warehouse
Connection Type	Database
Database Type	Snowflake*
ODBC Data Source Name (DSN)	dev_dw (64-bit)
WhereScape RED Metadata Connection Indicator	□
Source System	
Database Host/Server	C8288410193
Database Port	53722
Database ID	dev_dw
Database Link Name	
Database Credentials	
Extract User ID	postgres
Extract User Password	********
Administrator User ID	
Administrator User Password	
Other	
Default Schema for Browsing	edw_stage,edw_ods,edw_vault,edw_mart
New Table Default Load Type	Script based load
New Table Default Load Script Connection	Runtime Connection for Scripts - Windows
New Table Default Load Script Template	wsl_postgres_pyscript_load
SSIS Connection String (OLEDB)	
Data Type Mapping Set	PostgreSQL from PostgreSQL
Default Transform Function Set	PostgreSQL FS

ODBC Data Source Name (DSN)
ODBC Data Source Name (DSN) as defined in the Windows 'ODBC Data Source Administrator'
NOTE: The ODBC Source Name defined in RED must be the same on all machines that use the corresponding connection.

General

Options	Description
Connection Name	Name used to label the connection within WhereScape RED. For target databases like Snowflake, Redshift, Greenplum, Netezza or PDW, the Data Warehouse connection can be renamed to **Repository.**
Connection Type	Indicates the connection source type or the connection method, such as Database, ODBC, Windows, Unix. Select the **Database** connection type.
Database Type	The default is **(local)**. • If the database or metadata repository is located on the same server as the RED instance, leave the **(local)** default option selected. • If the database or metadata repository is located on a different server select the relevant database type of SQL Server.
ODBC Data Source Name (DSN)	ODBC Data Source Name (DSN) as defined in the Windows 32-bit **ODBC Data Source Administrator**. • Select the relevant ODBC connection to the data warehouse database or metadata repository. **Note:**

Options	Description
	The ODBC Source Name defined in RED must be the same on all machines that use the corresponding connection.
WhereScape RED Metadata Connection Indicator	Distinguishes the special connection that identifies the WhereScape RED datawarehouse/metadata repository. This option must be enabled for **Data Warehouse/Metadata Repository** type connections. **Note:** There must only be one data warehouse/metadata connection in a WhereScape RED repository.

Source System

Options	Description
Database Host/Server	Used by scripts to pass to native tools.
Database Port	Used by script to pass to native tools.
Database ID	Database Identifier or Database Name.
Database Link Name	Optional name of a Database Link that is used to access the database. Only required for Database type connections where the database is not on the same server as the data warehouse. If the server is on the same database, the link doesn't need to be defined and the field can be left blank. For connections to databases located in different servers where a database link load is required: • If the database link already exists to this source database, then enter that link name in this field. • If the link doesn't exist then enter a link name. SQL Server Enter the defined SQL LINK name that points to the correct server, the link name does not need to be the same name as the server name.
Provider Name	Name of the connection provider/driver used to connect to the database.
Full Database Path Name	This field is used when you enter **Microsoft.Jet.OLEDB.4.0** as the Provider Name, e.g. when the source database is from MS Access or Excel. Enter the full database path name.

Big Data Adapter Settings

Options	Description
JDBC Connection String (JDBC URL)	Connection string used by the WhereScape Big Data Adapter to access this database.
JDBC Driver Class Name	JDBC driver class to be used by the WhereScape Big Data Adapter. This field must be set if the JDBC URL is set. • Select the appropriate JDBC Driver class name from the drop-down list. If this is left empty, this is not specified in generated commands.
Omit Sqoop Driver Option	• If set, the --driver option to Sqoop is omitted. This is required for certain connection types. • If you select the **Omit Sqoop Driver Option** check-box, the driver parameter is not used in Sqoop command line.

Options	Description
Sqoop Connection Manager Class	Custom Sqoop connection manager class. Corresponds to the --connection-manager command line argument. Leave blank if this is not required.
Include Database/Schema Name in Sqoop Table Option	• If set, the --table option to Sqoop includes the database/schema name of the destination table when performing Apache Sqoop loads into this connection. • If this is not set, users must ensure that the database/schema is otherwise communicated to Sqoop, for example by using the $OBJECT_DATABASE$ token in the in the JDBC Connection String.
Include Sqoop Columns Option	• If set, the --columns option to Sqoop is included when performing Apache Sqoop loads into this connection. • If this is not set, users must ensure that the order of columns in the loads match the order in the metadata.

Database Credentials

Options	Description
Extract User ID	Database User that has access to SELECT from the source system tables to extract data. SQL Server For SQL Server, this field can be left blank if using a trusted login, or the server login password.
Extract User Password	The password of the data warehouse user. SQL Server For SQL Server, this field can be left blank if using a trusted login, or the server login password.
Administrator User ID	Database User ID to login, when using WhereScape SQL Admin via the 'Connect using SQL Admin (as SQL Admin User)' context menu item (optional).
Administrator User Password	Optional Database User Password to login, when using WhereScape SQL Admin via the 'Connect using SQL Admin (as SQL Admin User)' context menu item.
JDBC User ID	User ID to login when using JDBC via WhereScape Big Data Adapter (optional).
JDBC Password	Password to login when using JDBC via WhereScape Big Data Adapter (optional).

Other

Options	Description
Default Schema for Browsing	Optional comma-delimited list of schemas for the browser pane filter. Enter the schema(s) you want the connection to browse by default on the right browser pane.
New Table Default Load Type	The default **Load Type** for new Load tables created using this connection as a source. Select the desired default load type from the list, e.g. **Database Link Load, Script based load, Integration Services Load** or **Externally Loaded**. **Note:** The available options in this drop-down list is configured from Home > Options > **Available Load Types**.
New Table Default Load Script Connection	The default **Script Connection** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
Data Type Mapping Set	Mapping Set to use when converting from a source database data type to a destination database data type. Setting this field to **(Default)** makes RED automatically select the relevant mapping set, otherwise you can choose one of the standard mapping sets from the drop-down list or create a new set.

Options	Description
Default Transform Function Set	Function Set that is selected by default in the Transformation dialogs.
When Connection is an OLAP Data Source	This section of fields is only relevant and will only be visible from a **Data warehouse** connection (where the Data warehouse field is enabled). These fields are required so that the data warehouse can be used as a source for the Analysis Services cubes.
Connection Provider/Driver	Name of the Connection Provider/Driver to use to connect to the data warehouse database, when it is used as the data source for OLAP cubes.
Data Warehouse Server	Data Warehouse Server Name.
Data Warehouse Database ID	Data Warehouse Database Identifier or Database Name, which is used when the data warehouse is used as the data source for OLAP cubes.

Tip

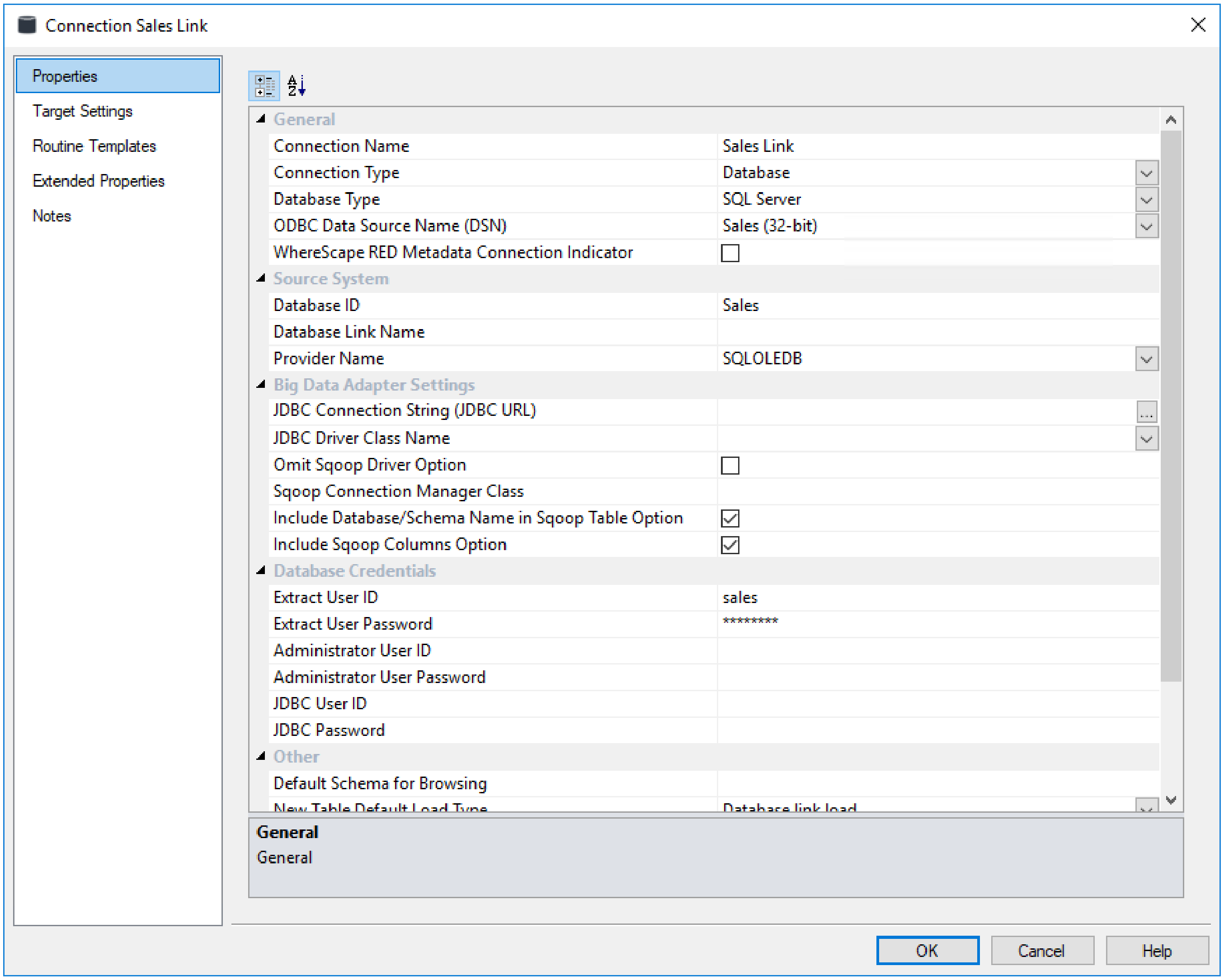
Once the connection has been set up, you can right-click the connection in the middle pane or double click the connection name from the left pane to view or edit the connection's **Properties**.

Database

This topic describes the details of the connection Properties as they apply to the Database type connections.

When coming to the Data Warehouse from an OLTP source system, the connection defines the path to get to those source tables. A connection object must be defined for each source system.

The fields presented in the **Connection Properties** screen below, depends on the connection and database types selected. See below for a description of all possible fields.



Connection Sales Link

General	
Connection Name	Sales Link
Connection Type	Database
Database Type	SQL Server
ODBC Data Source Name (DSN)	Sales (32-bit)
WhereScape RED Metadata Connection Indicator	□
Source System	
Database ID	Sales
Database Link Name	
Provider Name	SQLOLEDB
Big Data Adapter Settings	
JDBC Connection String (JDBC URL)	
JDBC Driver Class Name	
Omit Sqoop Driver Option	□
Sqoop Connection Manager Class	
Include Database/Schema Name in Sqoop Table Option	☑
Include Sqoop Columns Option	☑
Database Credentials	
Extract User ID	sales
Extract User Password	********
Administrator User ID	
Administrator User Password	
JDBC User ID	
JDBC Password	
Other	
Default Schema for Browsing	
New Table Default Load Type	Database link load

General
General

SQL Server **SQL Server:** when running a SQL Server data warehouse it is possible to create database links (linked server) to other SQL Server instances and most of the other major databases, such as DB2, etc.

- Database link load - from a table in the current SQL Server database.
- Database link load - from a table in another database on the same SQL Server instance - the OBDC Data Source Name must be defined in the Windows 32-bit **ODBC Data Source Administrator** and selected on the ODBC DSN field, as well as the Database ID that also needs to be populated in the Database ID field in the connection properties.
- Database link load - through a linked server.
- Integration Services load.

Examples for each of RED's database type connection screens for SQL Server, DB2 Greenplum, Netezza and PDW are provided below the connection properties description.

The connection Properties window has the following fields:

General

Options	Description
Connection Name	Name used to label the connection within WhereScape RED.
Connection Type	Indicates the connection source type or the connection method, such as Database, ODBC, Windows, Unix. Select the **Database** connection type.

Options	Description
Database Type	Type of database, the default is **(local)**.
ODBC Data Source Name (DSN)	ODBC Data Source Name (DSN) as defined in the Windows 32-bit **ODBC Data Source Administrator**. **Note:** The ODBC Source Name defined in RED must be the same on all machines that use the corresponding connection.
WhereScape RED Metadata Connection Indicator	Only required for the Data Warehouse/metadata repository connection. Leave this check box unselected.

Source System

Options	Description
Provider Name	Name of the connection provider/driver used to connect to the database.
Database ID	Database Identifier or Database Name.
Database Host/Server	PDW **PDW Database type connections only:** Used for sqlcmd specification.
Database Port	PDW **PDW Database type connections only:** Used for sqlcmd specification.
Database Link Name	Optional name of a Database Link that is used to access the database. SQL Server If SQL Server and the database are on the same server as the data warehouse then no link needs to be defined and this field can be left blank.
Provider Name	Name of the connection provider/driver used to connect to the database.
Full Database Path Name	This field is used when you enter **Microsoft.Jet.OLEDB.4.0** as the Provider Name, e.g. when the source database is from MS Access or Excel. Enter the full database path name.

Database Credentials

Options	Description
Extract User ID	Database User that has access to SELECT from the source system tables to extract data.
Extract User Password	The password of the data warehouse user. For SQL Server blank if a trusted login, or the server login password.
Administrator User ID	Database User ID to login, when using WhereScape SQL Admin via the 'Connect using SQL Admin (as SQL Admin User)' context menu item (optional).
Administrator User Password	Optional Database User Password to login, when using WhereScape SQL Admin via the 'Connect using SQL Admin (as SQL Admin User)' context menu item.

Other

Options	Description
Default Database for Browsing	(Custom) Optional comma-delimited list of databases for the browser pane filter. Enter the database(s) you want the connection to browse by default on the right browser pane.
Default Schema for Browsing	Optional comma-delimited list of schemas for the browser pane filter. Enter the schema(s) you want the connection to browse by default on the right browser pane.
New Table Default Load Type	The default **Load Type** for new Load tables created using this connection as a source. Select the desired default load type from the list, e.g. **Database Link Load, Script based**

Options	Description
	load, Integration Services Load or **Externally Loaded**. **Note:** The available options in this drop-down list is configured from Home > Options > **Available Load Types**.
New Table Default Load Script Connection	The default **Script Connection** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
Data Type Mapping Set	Mapping Set to use when converting from a source database data type to a destination database data type. Setting this field to **(Default)** makes RED automatically select the relevant mapping set, otherwise you can choose one of the standard mapping sets from the drop-down list or create a new set.

Tip:

Once the connection has been set up, you can right-click the connection in the middle pane or double click the connection name from the left pane to view or edit the connection's **Properties**.

ODBC

This connection type is via an ODBC link. Most data movement is performed using the ODBC connection.

The connection object Properties window has the following fields:

General

Options	Description
Connection Name	Name used to label the connection within WhereScape RED. Typically for SQL Server and DB2 this is **Data Warehouse**. For target databases like Greenplum, Netezza or PDW the Data Warehouse connection can be renamed to **Repository.**
Connection Type	Indicates the connection source type or the connection method, such as Database, ODBC, Windows, Unix. Select the **ODBC** connection type.
Database Type	Type of database, the default is **(local)**.
ODBC Data Source Name (DSN)	ODBC Data Source Name (DSN) as defined in the Windows 32-bit **ODBC Data Source Administrator**. **Note:** The ODBC Source Name defined in RED must be the same on all machines that use the corresponding connection.
WhereScape RED Metadata Connection Indicator	Distinguishes the special connection that identifies the WhereScape RED data warehouse/metadata repository. This option must be enabled for **Data Warehouse/Metadata Repository** type connections. **Note:** There must only be one data warehouse/metadata connection in a WhereScape RED repository.

ODBC

Options	Description
Database ID	Source Database Identifier or Database Name (e.g. as in DB2 or SQL Server).
Work Directory	Windows directory used by WhereScape RED to create temporary files for minimal logged extracts. The directory must exist and allow write access. There must be a different work directory for each WhereScape RED Scheduler running on the same machine to avoid file conflicts. Typically **C:\Temp** or a **sub-directory of C:\Temp** is used.

Credentials

Note:

If the source database does not support windows authentication, a username and password must be specified in the User ID and User Password fields below.

Options	Description
Extract User ID	Database User that has access to SELECT from the source system tables to extract data.
Extract User Password	Database Password to use with the Extract User ID to login to the source system to extract data.
Administrator User ID	Leave blank.
Administrator User Password	Leave blank.

Other

Options	Description
Default Schema for Browsing	Optional comma-delimited list of schemas for the browser pane filter. Enter the schema(s) you want the connection to browse by default on the right browser pane.
New Table Default Load Type	The default **Load Type** for new Load tables created using this connection. Select the desired default load type from **Native ODBC, Integration Services load** or **ODBC load**. **Note:** The available options in this drop-down list is configured from Home > Options > Available Load Types.
New Table Default Load Script Connection	The default **Script Connection** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
Staging database	**PDW Database type connections only:** The staging database to be used by Microsoft SQL Server Integration Services (SSIS) for loading to PDW.
Data Type Mapping Set	Mapping Set to use when converting from a source database data type to a destination database data type. Setting this field to **(Default)** makes RED automatically select the relevant mapping set, otherwise you can choose one of the standard mapping sets from the drop-down list or create a new set.
When Connection is an OLAP Data Source	This section of fields is only relevant and will only be visible from a **Data warehouse** connection (where the Data warehouse field is enabled). These fields are required so that the data warehouse can be used as a source for the Analysis Services cubes.

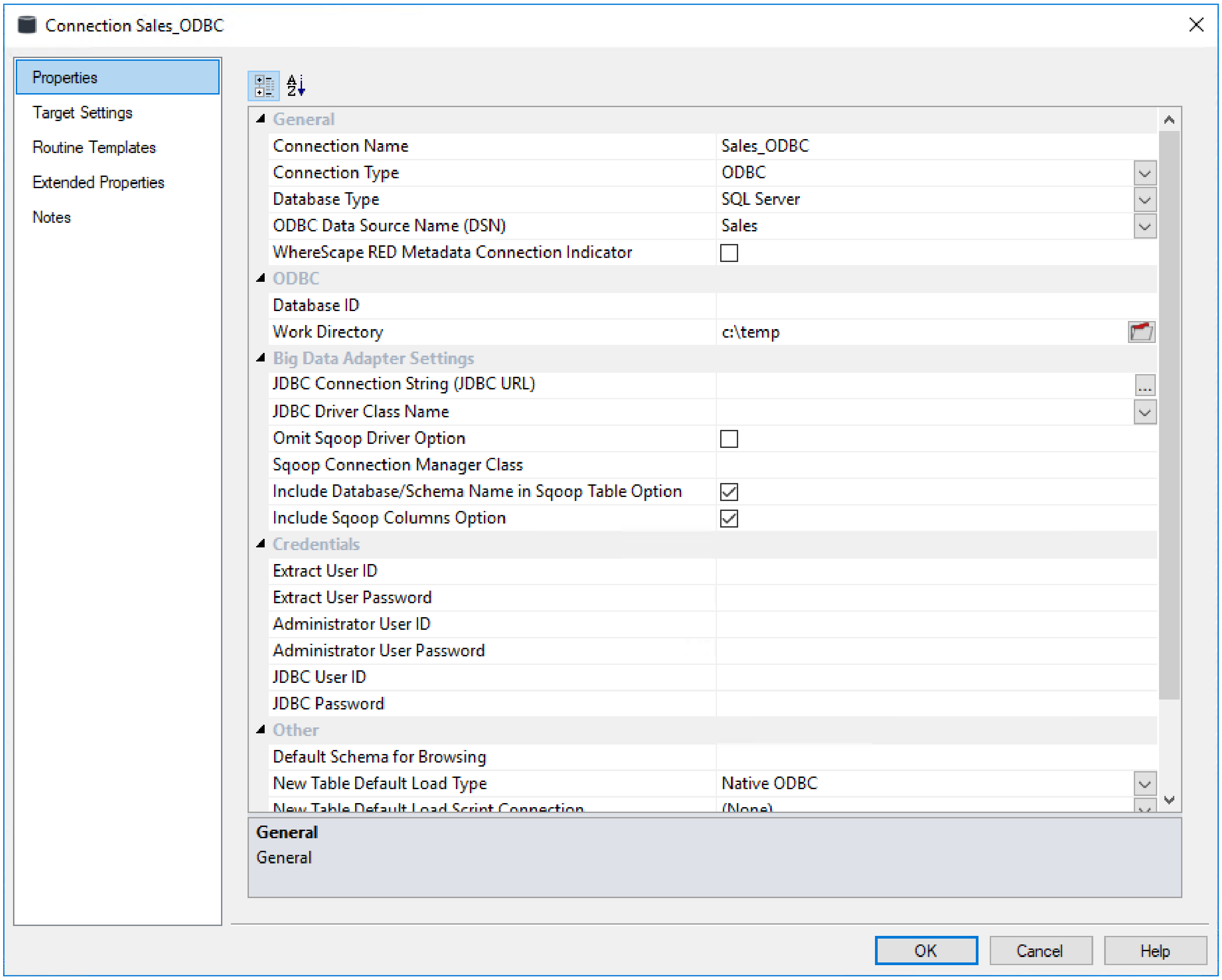
Options	Description
MSAS Connection String	Connection string to be used by Microsoft Analysis Services (MSAS) to connect to the data warehouse. For details on how to use the wizard to build the OLAP connection string, refer to **OLAP Defining Data Source for the OLAP Cube**. **Note:** A connection string is typically composed of multiple property name/value pairs that are semi-colon delimited.
Connection Provider/Driver	Name of the Connection Provider/Driver to use to connect to the data warehouse database, when it is used as the data source for OLAP cubes.
Data Warehouse Server	Data Warehouse Server Name, which is used when the data warehouse is used as the data source for OLAP cubes.
Data Warehouse Database ID	Data Warehouse Database Identifier or Database Name.

Tip:

Once the connection has been set up, you can right-click the connection in the middle pane or double click the connection name from the left pane to view or edit the connection's **Properties**.

Below are two examples of a **Native ODBC** Load.

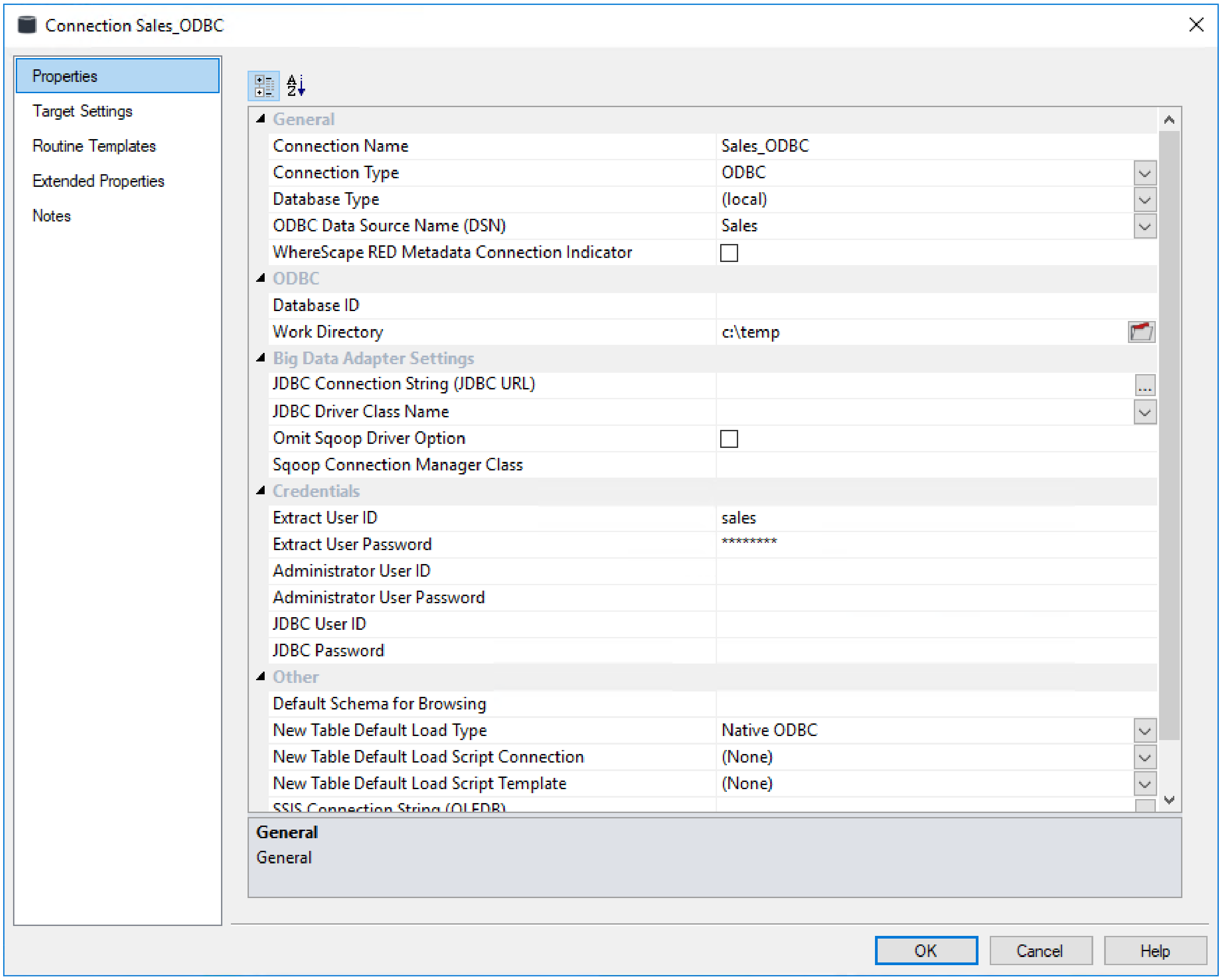
If the source database supports windows authentication, it is not necessary to specify a username and password:



General	
Connection Name	Sales_ODBC
Connection Type	ODBC
Database Type	SQL Server
ODBC Data Source Name (DSN)	Sales
WhereScape RED Metadata Connection Indicator	□
ODBC	
Database ID	
Work Directory	c:\temp
Big Data Adapter Settings	
JDBC Connection String (JDBC URL)	
JDBC Driver Class Name	
Omit Sqoop Driver Option	□
Sqoop Connection Manager Class	
Include Database/Schema Name in Sqoop Table Option	☑
Include Sqoop Columns Option	☑
Credentials	
Extract User ID	
Extract User Password	
Administrator User ID	
Administrator User Password	
JDBC User ID	
JDBC Password	
Other	
Default Schema for Browsing	
New Table Default Load Type	Native ODBC
New Table Default Load Script Connection	(None)

General
General

If the source database does not support windows authentication, a username and password must be specified:



Connection Sales_ODBC

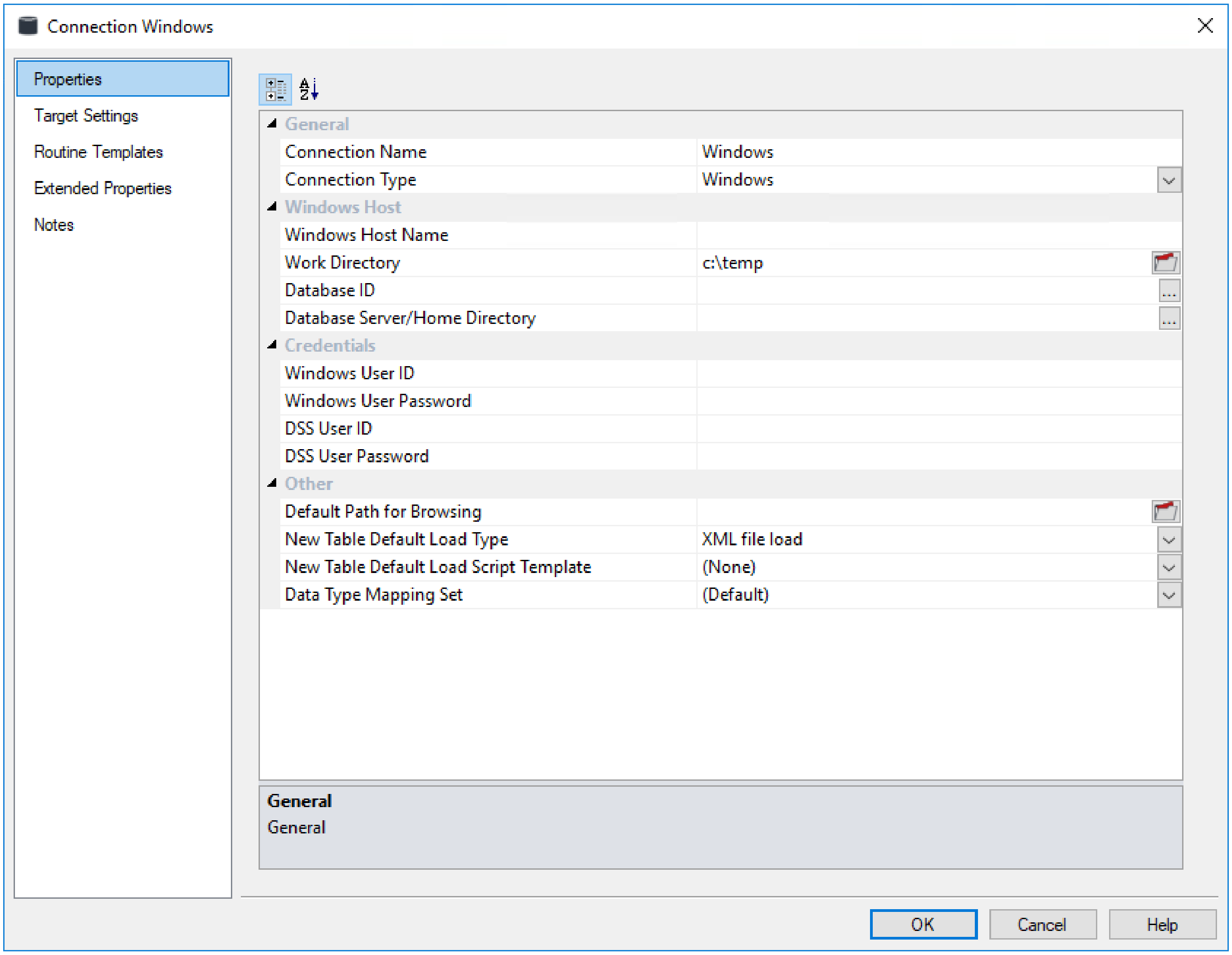
General	
Connection Name	Sales_ODBC
Connection Type	ODBC
Database Type	(local)
ODBC Data Source Name (DSN)	Sales
WhereScape RED Metadata Connection Indicator	□
ODBC	
Database ID	
Work Directory	c:\temp
Big Data Adapter Settings	
JDBC Connection String (JDBC URL)	
JDBC Driver Class Name	
Omit Sqoop Driver Option	□
Sqoop Connection Manager Class	
Credentials	
Extract User ID	sales
Extract User Password	********
Administrator User ID	
Administrator User Password	
JDBC User ID	
JDBC Password	
Other	
Default Schema for Browsing	
New Table Default Load Type	Native ODBC
New Table Default Load Script Connection	(None)
New Table Default Load Script Template	(None)
SSIS Connection String (OLEDB)	

General
General

Windows

This connection is for the PC that you are working on, or to a host Windows PC. From Windows, you can only do flat file loads.

Sample **SQL Server Windows Connection** screen:


Connection Windows

General
- Connection Name: Windows
- Connection Type: Windows

Windows Host
- Windows Host Name:
- Work Directory: c:\temp
- Database ID:
- Database Server/Home Directory:

Credentials
- Windows User ID:
- Windows User Password:
- DSS User ID:
- DSS User Password:

Other
- Default Path for Browsing:
- New Table Default Load Type: XML file load
- New Table Default Load Script Template: (None)
- Data Type Mapping Set: (Default)

General
General

General

Options	Description
Connection Name	Name used to label the connection within WhereScape RED.
Connection Type	Indicates the connection source type or the connection method, such as Database, ODBC, Windows, Unix. Select the **Windows** connection type.

Windows Host

Options	Description
Windows Host Name	IP address or host name that identifies the Windows machine. Leave blank to connect to the local machine.
Work Directory	Windows directory used by WhereScape RED to create temporary files for minimal logged extracts. The directory must exist and allow write access. There must be a different work directory for each WhereScape RED Scheduler running on the same machine to avoid file conflicts. Typically C:\Temp or a sub-directory of C:\Temp is used.
Database ID	The Source Database Identifier.
Database Server/Home Directory	Optional to specify the Database Home Directory if it is different from the standard home directory.

Credentials

Options	Description
Windows User ID and Password	Leave this blank if you are connecting to your own PC. Enter details if you are connecting remotely to another Windows system.
Dss User ID and Password	Enter the relevant details for connecting to the **Data Warehouse**.

Other

Options	Description
Default Path for Browsing	Optional default Path for browser pane filter. When a path has been selected in this field, it becomes the initial point for browsing and it is also expanded on open in the right hand browser pane.
New Table Default Load Type	The default **Load Type** for new Load tables created using this connection as a source. Select the desired default load type from the list, e.g. **Database Link Load, Script based load, Integration Services Load** or **Externally Loaded**. **Note:** The available options in this drop-down list is configured from Home > Options > **Available Load Types**.
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
Data Type Mapping Set	Mapping Set to use when converting from a source database data type to a destination database data type. Setting this field to **(Default)** makes RED automatically select the relevant mapping set, otherwise you can choose one of the standard mapping sets from the drop-down list or create a new set.

Tip:

Once the connection has been set up, you can right-click the connection in the middle pane or double click the connection name from the left pane to view or edit the connection's **Properties**.

To test the connection:

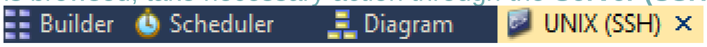
- Select **Browse | Source Tables** from the menu strip
- In the right pane you should be able to drill down to the area required.

UNIX

Tip:

UNIX/Linux connections are only supported from **DB2** and **Greenplum** data warehouses.

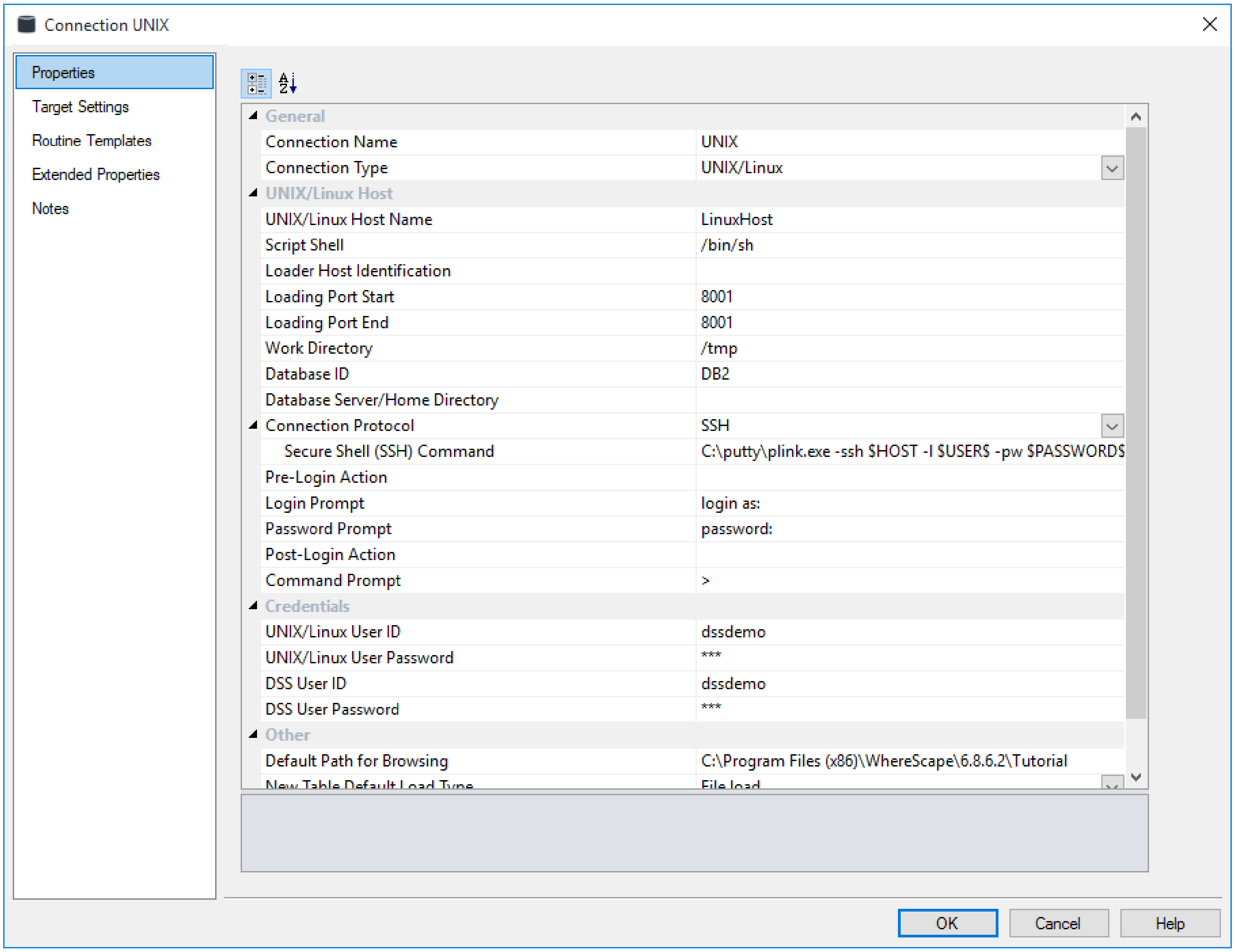
If the **UNIX/Linux** connection returns a blank screen or an error message in the **Results** pane after the connection is browsed, take necessary action through the **Server (SSH)** tab next to the main Builder and Scheduler tabs.



This tab is displayed after browsing the UNIX connection.

This topic describes in greater detail the connection Properties as they apply to **UNIX** connections. From a UNIX connection you can only do flat file loads.

Sample DB2 UNIX connection screen



General	
Connection Name	UNIX
Connection Type	UNIX/Linux
UNIX/Linux Host	
UNIX/Linux Host Name	LinuxHost
Script Shell	/bin/sh
Loader Host Identification	
Loading Port Start	8001
Loading Port End	8001
Work Directory	/tmp
Database ID	DB2
Database Server/Home Directory	
Connection Protocol	SSH
Secure Shell (SSH) Command	C:\putty\plink.exe -ssh $HOST -l $USER$ -pw $PASSWORD$
Pre-Login Action	
Login Prompt	login as:
Password Prompt	password:
Post-Login Action	
Command Prompt	>
Credentials	
UNIX/Linux User ID	dssdemo
UNIX/Linux User Password	***
DSS User ID	dssdemo
DSS User Password	***
Other	
Default Path for Browsing	C:\Program Files (x86)\WhereScape\6.8.6.2\Tutorial
New Table Default Load Type	File load

General

Options	Description
Connection Name	Name used to label the connection within WhereScape RED.
Connection Type	Indicates the connection source type or the connection method, such as Database, ODBC, Windows, Unix. Select the **UNIX** connection type.

Unix/Linux Host

Options	Description
UNIX/Linux Host Name	IP address or host name that identifies the UNIX machine.
Script Shell	Path to the POSIX-compliant UNIX/Linux shell to use for generated scripts. For **UNIX** hosts, set to **/bin/ksh**. For **Linux** hosts set to **/bin/sh**. If this field is left blank, a default will be chosen based on the name of the connection and the type of database used for the WhereScape RED metadata repository.
Loader Host Identification	IP Address or host name(s) that identifies the Loader/ Multiple hosts can be entered with using a comma (,) to delimit.
Work Directory	Windows directory used by WhereScape RED to create temporary files for minimal logged extracts. The directory must exist and allow write access. There must be a different work

Options	Description
	directory for each WhereScape RED Scheduler running on the same machine to avoid file conflicts. Typically C:\Temp or a sub-directory of C:\Temp is used.
Database ID	Database Identifier or Database Name.
Database Server/Home Directory	Optional to specify the Database Home Directory if it is different from the standard home directory.
Connection Protocol	Telnet or Secure Shell (SSH) protocol to use to connect to the UNIX/Linux machine. For SSH, the **Secure Shell (SSH) Command** property is enabled to specify how to connect.
Secure Shell (SSH) Command	Command to execute to connect to a UNIX/Linux machine using the Secure Shell (SSH) protocol such as **C:\Program Files(x86)\PuTTY\plink.exe -ssh $HOST$ -l $USER$ -pw $PASSWORD$**
Pre-Login Action, Login Prompt, Password Prompt, Post-Login Action, and Command Prompt.	These fields are only used to create a Telnet connection to the host machine. WhereScape RED uses the Telnet connection in the drag and drop functionality. They are **not used in the actual production running of the Data Warehouse,** and is only necessary if you wish to use the drag and drop functionality.
Pre-Login Action	Response or command to send BEFORE logging in to the UNIX/Linux machine. Typically this is NOT necessary but it can be used to indicate that the UNIX/Linux Login Prompt is preceded by a line-feed (\n). However it is preferable that the UNIX/Linux login displays the Login Prompt without anything preceding it. [Optional]
Login Prompt	The UNIX login prompt, or the tail end of the login prompt, e.g. **login as:**.
Password Prompt	The UNIX password prompt, or the tail end of the password prompt, e.g. **ssword:**.
Post-Login Action	Not often used but may be necessary to respond to a login question. It is preferable that the UNIX login goes straight to the command prompt.
Command Prompt	Enter the UNIX/Linux command prompt, or the tail end of that prompt, typically **>**. **Note** To ascertain some of the above fields, you have to log in to the UNIX system.

Credentials

Options	Description
UNIX/Linux User ID	User Account to login to the UNIX/Linux Host.
UNIX/Linux User Password	Password to login to the UNIX/Linux Host.
DSS User ID	Database user to connect to the WhereScape RED metadata repository.
DSS User Password	Database password to connect to the WhereScape RED metadata repository.

Other

Options	Description
Default Path for Browsing	Optional default Path for browser pane filter. When a path has been selected in this field, it becomes the initial point for browsing and it is also expanded on open in the right hand browser pane.
New Table Default Load Type	The default **Load Type** for new Load tables created using this connection as a source. Select the desired default load type from the list, e.g. **Database Link Load, Script based load, Integration Services Load** or **Externally Loaded**.

Options	Description
	Note: The available options in this drop-down list is configured from Home > Options > **Available Load Types**.
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
Data Type Mapping Set	XML files have been created to store mappings from one set of data types to another. Setting this field to **(Default)** will cause RED to automatically select the relevant mapping set; otherwise you can choose one of the standard mapping sets from the drop-down list or create a new one.

To validate the fields

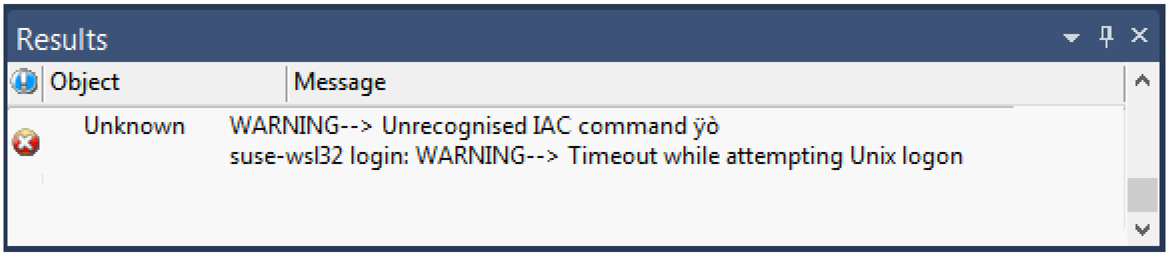
- Right-click on the connection name
- Select **Telnet window**

To test the drag and drop functionality

- From the menu strip select **Browse > Source Tables**
- Drill down to the area required
- Drag an item to the middle pane, (having first selected the object in the left pane).

Connection Failures

In the event that a telnet connection cannot be established to the UNIX host, the following message normally appears in the results pane, after approximately 30 seconds



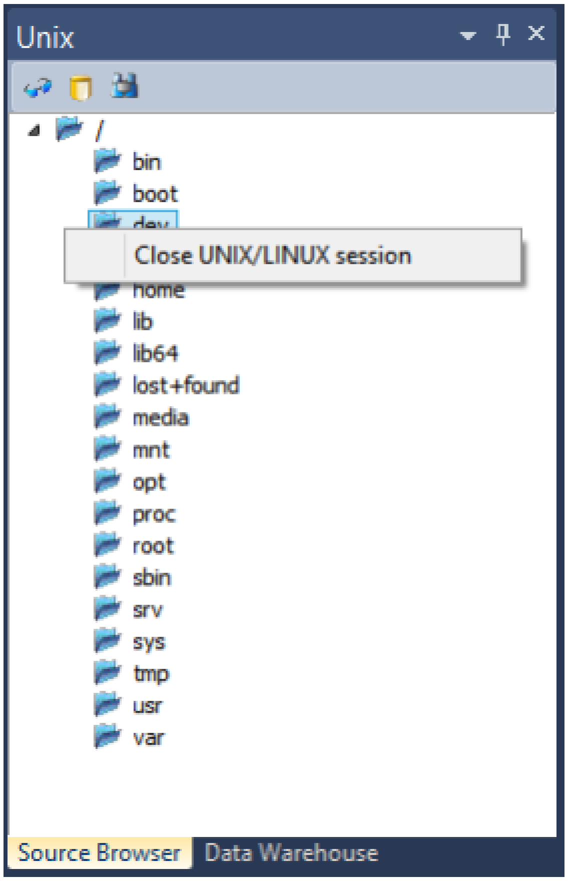
Object	Message
Unknown	WARNING--> Unrecognised IAC command ÿò suse-wsl32 login: WARNING--> Timeout while attempting Unix logon

Attempt the connection again and using the **Window** menu option select the **Telnet** window. This displays the login session and should provide an insight as to why the connection is not being completed.

If the situation cannot be resolved, a telnet trace can be acquired. Select the **Tools > Options** menu option and click on the **Trace all Unix sessions** check box. Then try to do the connection or browse again. A log file called WslMedTelnet.log is created in the WhereScape program directory. Edit the log file and ensure there are no passwords visible and then contact WhereScape support using the WhereScape forum at www.wherescape.com.

Closing the Connection

To close the collection, right-click in the browser pane and select **Close UNIX/LINUX session**:



Extensible Source Connections

This topic describes how to configure and use Extensible Source Connections. This connection type enables any data source type to be browsed and ingested into your data warehouse, as long as your sources have Windows scriptable APIs or command-line tools available or both. Examples of source types made available by this feature are REST/SOAP Services, JDBC, Cloud-Based Storage, JSON/XML/Avro, and many more.

Prerequisites

Before you can create an Extensible Source Connection you must first have setup:

1. A matching set of UI Configurations for Connection Properties and Load Table Properties.
2. Browse Script that utilizes these configurations.
3. Windows Connection to run the Browse Script against.

Additionally you would need a Load Template that works with your Extensible Source Connection and it's configured field metadata to actually load data into your data warehouse from that source. Please see **UI Configurations** section for more information on accessing the configured field metadata through Templates and Scripts.

For detailed information of the prerequisites involved please review the following sections:

- **UI Configurations**
- **Browse Scripts**

RED provides examples for each of the prerequisites above that all form a working example of an SQL ODBC connection utilizing these features. Please follow the instructions **here** to create each of the examples to build the set.

Note

Optionally you can setup a **Column UI Configuration** on your target connection properties. Review the UI Configurations section for more details.

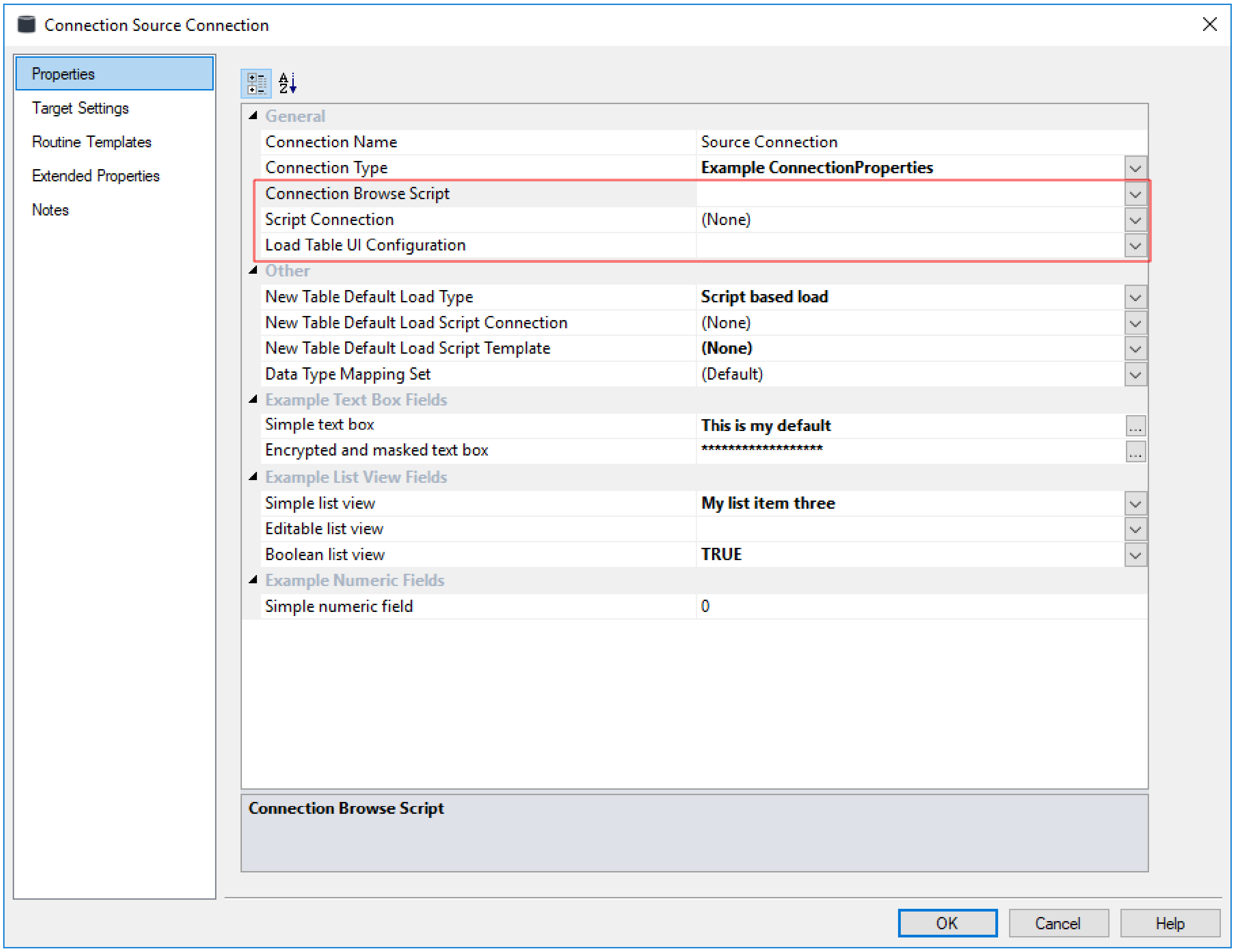
Creating Extensible Source Connections

After you setup the prerequisites described above, add a new Extensible Source Connection by following these steps:

1. To add a new Extensible Source Connection, right-click **Connection** located on the left side panel, and select **New Object**.
2. After adding an object name for the new connection, the **Properties** dialog displays. Under the Connection Type, all the Connection UI Configurations loaded into your repository are displayed in the drop-down list together with the existing connection types.
3. Selecting a Connection UI Configuration name from the Connection Type list will enable three new required connection properties: **Connection Browse Script**, **Script Connection**, and **Load Table UI Configuration**.

- **Connection Browse Script**: The Windows based Host Script that will be executed by the 'Browse' action for this connection.
- **Script Connection**: The Windows connection for the Browse script to execute against.
- **Load Table UI Configuration**: The Load Table UI configuration applied to Load tables sourced from this connection.

The fields presented in the **Connection Properties** screen below, depends on the connection, see below the available options when selecting an UI Configuration connection:



General	
Connection Name	Source Connection
Connection Type	**Example ConnectionProperties**
Connection Browse Script	
Script Connection	(None)
Load Table UI Configuration	
Other	
New Table Default Load Type	**Script based load**
New Table Default Load Script Connection	(None)
New Table Default Load Script Template	**(None)**
Data Type Mapping Set	(Default)
Example Text Box Fields	
Simple text box	**This is my default**
Encrypted and masked text box	******************
Example List View Fields	
Simple list view	**My list item three**
Editable list view	
Boolean list view	**TRUE**
Example Numeric Fields	
Simple numeric field	0

Connection Browse Script

The Configure Source Connection Properties window has the following fields:

General

Options	Description
Connection Name	Name used to label the connections within WhereScape RED.
Connection Type	All available Connection UI Configurations will appear in this list in addition to the existing connection types.
Connection Browse Script	Lists all the Windows Host Scripts in the metadata.
Script Connection	Lists all the Windows connections in the metadata.
Load Table UI Configuration	Lists all the Load UI Configurations in the metadata.

Other

Options	Description
New Table Default Load Type	Only 'Script based load' or 'Externally loaded' are available to Extensible Source Connections.
New Table Default Load	The default **Script Connection** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.

Options	Description
Script Connection	
New Table Default Load Script Template	The default **Script Template** to use when a **Script based load** type is defined for a Load table object that is **sourced** from this connection.
Data Type Mapping Set	Mapping Set to use when converting from a source database data type to a destination database data type. Setting this field to **(Default)** makes RED automatically select the relevant mapping set, otherwise you can choose one of the standard mapping sets from the drop-down list or create a new set.

The **Text Box Fields**, **List View Fields**, and **Numeric Fields** displayed in the screenshot are field configurations taken from the examples found in Field Configuration JSON.

The Column UI Configuration selection is only available on Target Enabled Connections and not Extensible Source Connections. For more information, review Connection Target Settings.

Loading from Extensible Source Connections

To load Load Tables which were created from an Extensible Source Connection you need to develop a load template that has logic in it to work with that connection type.

WhereScape provided Enablement Packs containing Extensible Source Connections will also have the relevant load templates to match. This section briefly explains how to determine the Connection Type from load templates so that you can generate the correct routine to load the table.

The Name of your Connection Properties UI Configuration is used as the Connection Type, so it is important to name your UI Configurations meaningfully.

Here is a pebble template macro to output the Connection Type as a string:

```
{% macro GetConnectionType(con = table.loadInfo.sourceConnection) %}
{%- fetch con -%}
{%- if con.connectionPropertiesConfig is defined -%}
{{ con.connectionPropertiesConfig }}
{%- else -%}
{{ con.connectionType.name }}
{%- endif -%}
{% endmacro %}
```

To use this macro to set a variable in PowerShell syntax:

```
$sourceConnectionType='{{ GetConnectionType() }}'
```

To use this macro in a pebble template conditional block:

```
{% if GetConnectionType() | trim == "ODBC" %} # add your ODBC specific code here {% endif %}
```

Note that there are shortcuts to these items in the table.loadinfo node but the macro provided above is generic and suitable for any routine generation context. The equivalent pebble template shortcuts available to Load Table routine generation are:

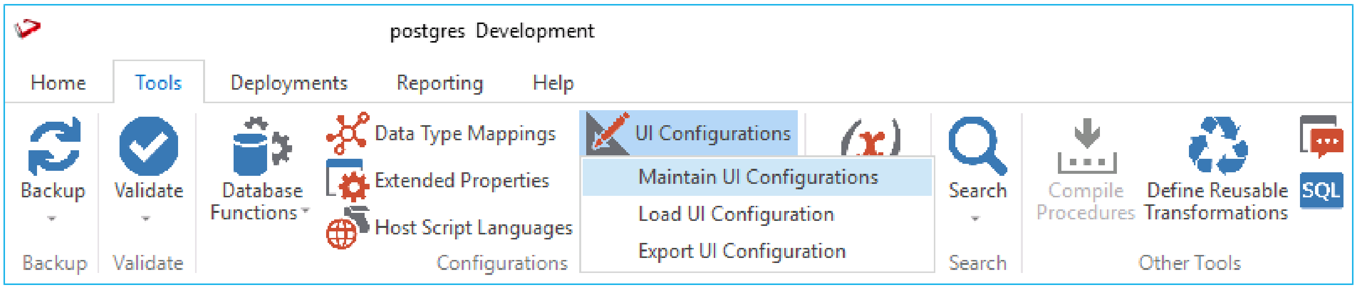
```
{{ table.loadInfo.sourceConnectionType.name }}
{{ table.loadInfo.sourceConnectionPropertiesConfig }}
```

Create an Extensible Source Connection set example

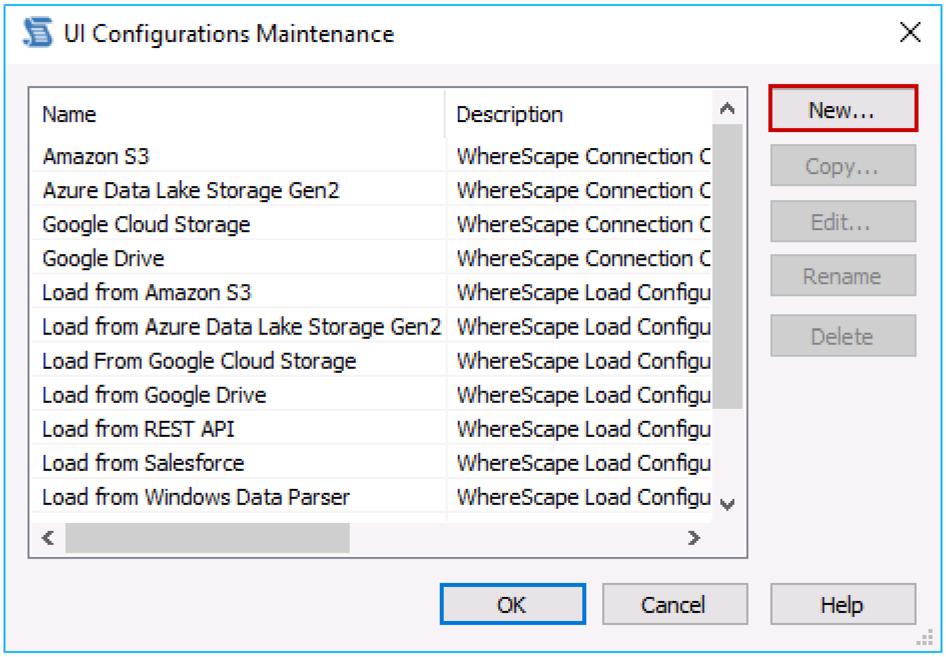
This section steps through creating the example Extensible Source Connection set which provides the prerequisites mentioned earlier. The example set was designed to browse an SQL Server ODBC DSN but can be easily adapted to other ODBC based DSN's also.

Follow these steps to create the example SQL Server ODBC Extensible Source Connection prerequisites:

1. Create a new Connection Properties UI configuration from Maintain UI Configuration by selecting **Tools>UI Configurations>Maintain UI Configurations**

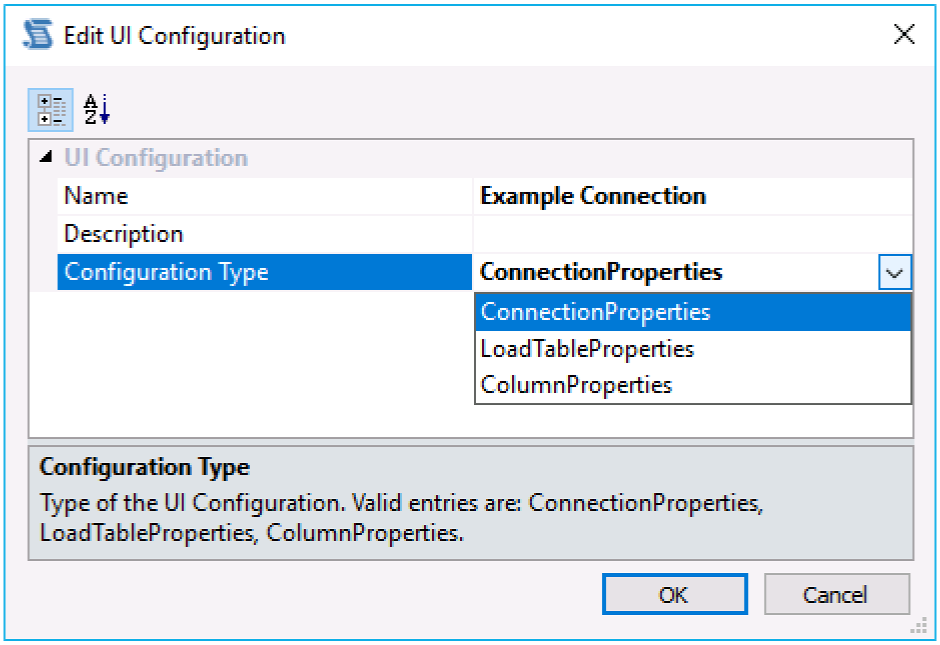


2. In the UI Configurations Maintenance dialog, click **New**.



Name	Description
Amazon S3	WhereScape Connection C
Azure Data Lake Storage Gen2	WhereScape Connection C
Google Cloud Storage	WhereScape Connection C
Google Drive	WhereScape Connection C
Load from Amazon S3	WhereScape Load Configu
Load from Azure Data Lake Storage Gen2	WhereScape Load Configu
Load From Google Cloud Storage	WhereScape Load Configu
Load from Google Drive	WhereScape Load Configu
Load from REST API	WhereScape Load Configu
Load from Salesforce	WhereScape Load Configu
Load from Windows Data Parser	WhereScape Load Configu

3. Under the Edit UI Configuration, add a **Name**, **Description**, select a **Configuration Type** from the drop-down list, and click **OK**.



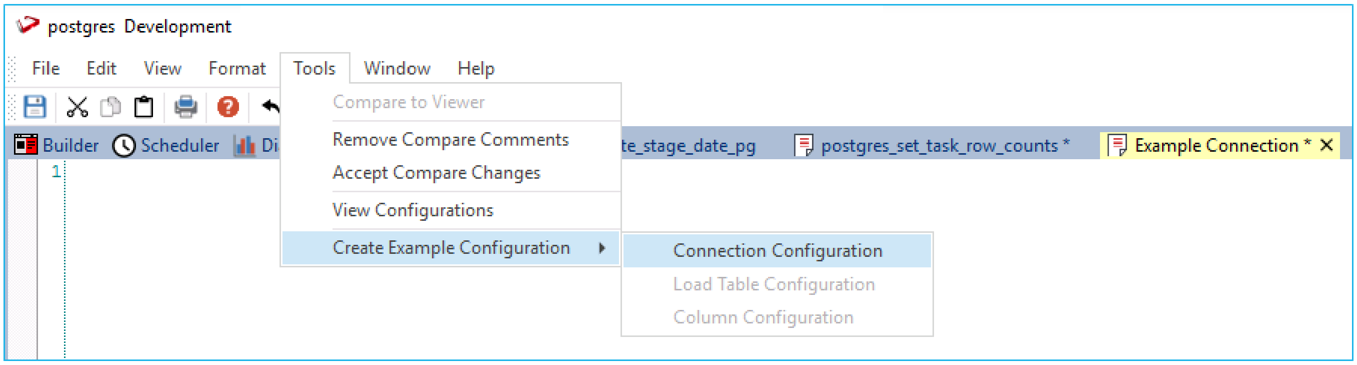
Name: Example Connection

Configuration Type: ConnectionProperties

Configuration Type

Type of the UI Configuration. Valid entries are: ConnectionProperties, LoadTableProperties, ColumnProperties.

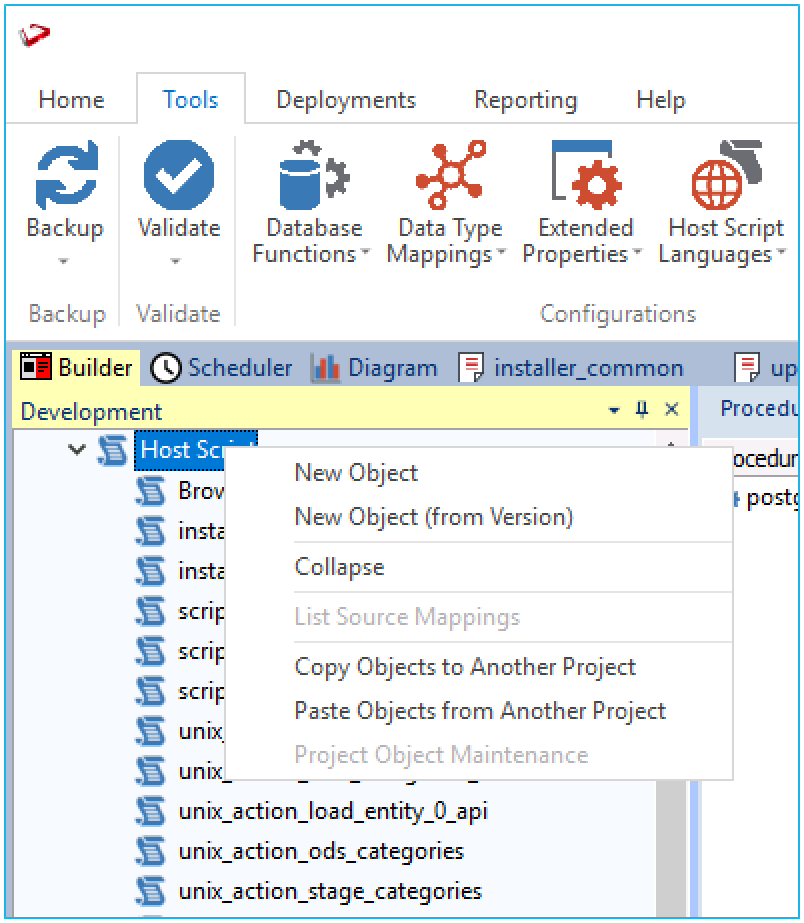
4. Empty the default script completely to enable the examples menu in the editor. Select **Tools>Create Example Configuration>Connection Configuration**



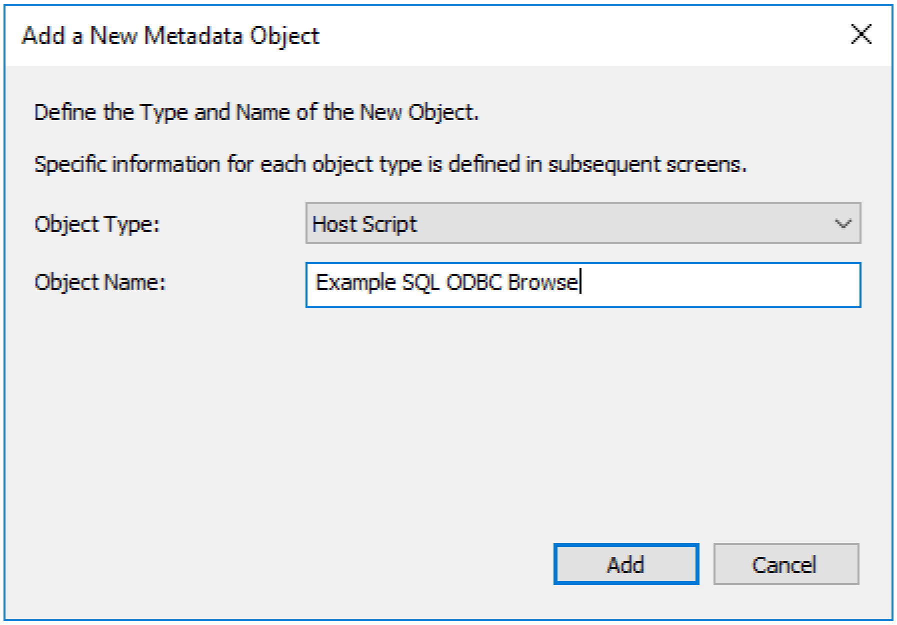
5. Repeat step 1 and step 2 to create the matching Load Table UI and Column UI Configurations.

6. Create a Connection Browse Script by following these steps:

a. Right-click Host Script and select **New Object**.



b. In the Add a New Metadata Object window, add an Object Name for the script, and click Add.



Add a New Metadata Object

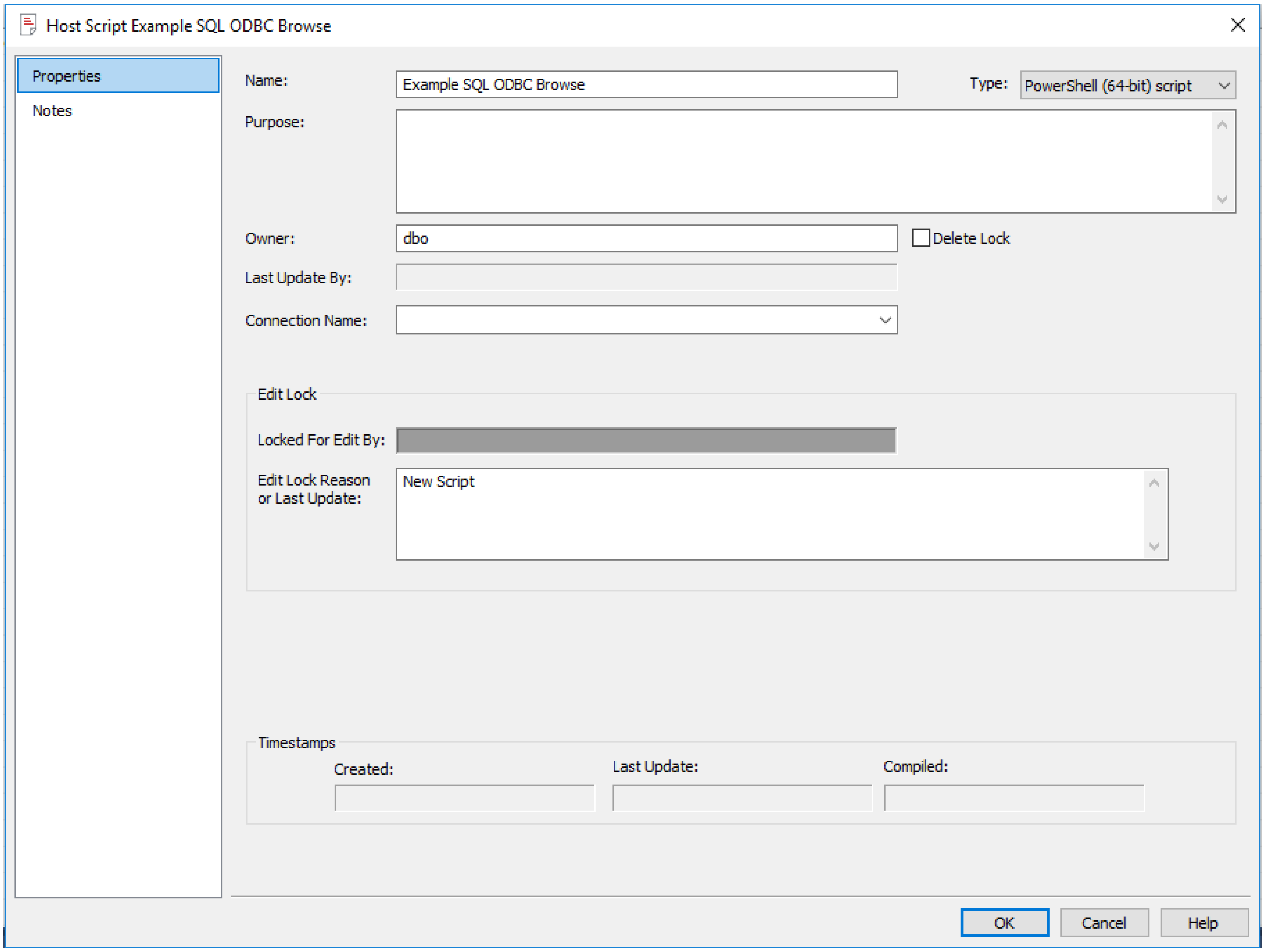
Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: Host Script

Object Name: Example SQL ODBC Browse

c. The Host Script properties window open, edit the options you need, and click **OK**.



Host Script Example SQL ODBC Browse

Name: Example SQL ODBC Browse

Type: PowerShell (64-bit) script

Owner: dbo

Edit Lock Reason or Last Update: New Script

d. Open your Host Script in the script editor and select **Tools>Create Example Browse Script**

e. After the Example Browse Script is created, click **Save**.

Tip

Review **Browse Scripts** for more information on **Output JSON mapping** and the **Scripted Browse Workflow**,

Browse Scripts

For Extensible Source Connections the **Browse Connection** action is performed by a Scripted Browse Workflow. The Browse Script executed for this action is selected through the Extensible Source Connection's Properties under the **Connection Browse Script** setting.

Browse Scripts are created like any other script in the RED metadata with the following restrictions:

- Only Windows based scripts are supported by the Scripted Browse Workflow.
- The Scripted Browse Workflow expects two lines of output:
    - First line is the success code: 1,-1, or -2
    - Second line is the entire minimised JSON output (line feeds must be removed)
    - The output JSON must conform to a predefined structure

The predefined structure for the output JSON is best demonstrated by creating the **example** for Extensible Source Connection set and performing the 'Browse Action'.

The example Browse Script saves the output JSON to the user's temp directory as *ouput.json* for review. This output should be reviewed together with the **Output JSON Mapping** section in order to understand how the structure relates to the UI.

The example Browse Script can be created by adding a new PowerShell (64-bit) script object to RED and selecting **Tools->Create Example Browse Script** from the empty script editor window. This example is intended to be used in conjunction with the matching set of UI Configuration examples.

Browse Script Output JSON Attribute Mappings

This table shows the expected attribute structure for the Output JSON produced by Browse Scripts and the mappings of those attributes to the RED UI.

Node/Attribute	Value Type	Description
.."treeViewLayout"	STRING := "Tabular"	Layout mode of the Browser Tree View
.."treeViewIcons"		Node containing the Browser Tree View icon settings **Icon Configuration for the Source Browser Tree**
..|.."schema"	STRING	Icon file name for the Schema Node in the Browser Tree View
..|.."table"	STRING	Icon file name for the Table Node in the Browser Tree View
.."<Schema Node Label>"	Set <Schema Node Label> as STRING	Displayed in the Browser Tree View as Schema/Folder node containing an array of Objects
.."<Table Node Label>"	Set <Table Node Label> as STRING	Displayed in the Browser Tree View as an Object node containing an Object
.."treeViewCategory"	STRING	[Optional] The category key to match the treeViewIcons configuration for this node **Icon Configuration for the Source Browser Tree**
..|.."name"	STRING	The name of the source object used to create the Load Table Name in RED
..|.."description"	STRING	Maps to Description field in Load Table Properties tab in RED
..|.."rowCount"	NUMERIC	Displayed in the Browser Tree View as the object's row count or size.
..|.."columns"		Node containing the array of one or more columns for the object.
..|..|.."name"	STRING	Maps to Column Name in the Column's properties in RED

<table>
<tr><td><b>..|..|..\"treeViewCategory\"</b></td><td>STRING</td><td>[Optional] The category key name to match to treeViewIcons configuration for this node <b>Icon Configuration for the Source Browser Tree</b></td></tr>
<tr><td><b>..|..|..\"dataType\"</b></td><td>STRING</td><td>Data type used in RED Data Type Mapping</td></tr>
<tr><td><b>..|..|..\"dataTypeLength\"</b></td><td>NUMERIC|NULL</td><td>Data type length used in RED Data Type Mapping</td></tr>
<tr><td><b>..|..|..\"dataTypeScale\"</b></td><td>NUMERIC|NULL</td><td>Data type scale used in RED Data Type Mapping</td></tr>
<tr><td><b>..|..|..\"dataTypePrecision\"</b></td><td>NUMERIC|NULL</td><td>Data type precision used in RED Data Type Mapping</td></tr>
<tr><td><b>..|..|..\"nullAllowed\"</b></td><td>BOOLEAN := TRUE|FALSE</td><td>Maps to the Null Values Allowed field in the Column's properties in RED</td></tr>
<tr><td><b>..|..|..\"defaultValue\"</b></td><td>STRING</td><td>Maps to the Default Value in the Column's properties in RED</td></tr>
<tr><td><b>..|..|..\"description\"</b></td><td>STRING</td><td>Maps to the Column Description in the Column's properties in RED</td></tr>
<tr><td><b>..|..|..\"displayName\"</b></td><td>STRING</td><td>Maps to the Business Display Name in the Column's properties in RED</td></tr>
<tr><td><b>..|..|..\"format\"</b></td><td>STRING</td><td>Maps to the Format field in the Column's properties in RED</td></tr>
<tr><td><b>..|..|..\"additive\"</b></td><td>BOOLEAN := TRUE|FALSE</td><td>Maps to the Additive field in the Column's properties in RED</td></tr>
<tr><td><b>..|..|..\"numeric\"</b></td><td>BOOLEAN := TRUE|FALSE</td><td>Maps to the Format field in the Column's properties in RED</td></tr>
<tr><td><b>..|..|..\"attribute\"</b></td><td>BOOLEAN := TRUE|FALSE</td><td>Maps to the Attribute field in the Column's properties in RED</td></tr>
<tr><td><b>..|..|..\"sourceTable\"</b></td><td>STRING</td><td>Maps to the Source Table field in the Column's properties in RED</td></tr>
<tr><td><b>..|..|..\"sourceColumn\"</b></td><td>STRING</td><td>Maps to the Source Column field in the Column's properties in RED</td></tr>
<tr><td><b>..|..|..\"transform\"</b></td><td>STRING</td><td>Maps to the Transformation field in the Column's properties in RED</td></tr>
<tr><td><b>..|..|..\"transformType\"</b></td><td>STRING := \"A\"|\"D\"</td><td>Maps to the Transformation Type field in the Column's properties in RED. Only 'A' for After-Load or 'D' for During-load are valid entries. Defaults to During-load when no value is given.</td></tr>
<tr><td><b>..|..|..\"uiConfigColumnProperties\"</b></td><td></td><td>Node containing the array (if any) of field value pairs to map to any Configured Column UI fields.</td></tr>
<tr><td><b>..|..|..|.\"a_configured_column_UI_field\"</b></td><td>STRING|NUMERIC</td><td>Maps to \"a_configured_column_UI_field\" field in the Column's Properties in RED</td></tr>
<tr><td><b>..|..\"loadInfo\"</b></td><td></td><td>Node containing the object level mappings to the Load Properties Source tab in RED.</td></tr>
<tr><td><b>..|..|..\"fileLoaderOptions\"</b></td><td>STRING</td><td>Maps to Loader Options field in Load Table Properties Source tab in RED</td></tr>
<tr><td><b>..|..|..\"fileParsed\"</b></td><td>BOOLEAN := TRUE|FALSE</td><td>Maps to an internal metadata flag for the Load Table in RED, not currently displayed in the UI</td></tr>
<tr><td><b>..|..|..\"overrideLoadSQL\"</b></td><td>STRING</td><td>Maps to Loader Options field in Load Table Properties Source tab in RED</td></tr>
<tr><td><b>..|..|..\"overrideSourceColumns\"</b></td><td>STRING</td><td>Maps to Override Source Columns field in Load Table Properties Source tab in RED</td></tr>
</table>

..|..|.."selectDistinctValues"	BOOLEAN := TRUE|FALSE	Maps to Select Distinct Values field in Load Table Properties Source tab in RED
..|..|.."sourceFile"		Node containing the Source File Details
..|..|..|.."charSet"	STRING	Maps to an internal metadata field for the Load Table in RED, not currently displayed in the UI
..|..|..|.."escapeEncoding"	STRING	Maps to an internal metadata field for the Load Table in RED, not currently displayed in the UI
..|..|..|.."fieldDelimiter"	STRING	Maps to Source File Field Delimiter field in Load Table Properties Source tab in RED
..|..|..|.."fieldEnclosure"	STRING	Maps to Source File Field Enclosure Delimiter field in Load Table Properties Source tab in RED
..|..|..|.."headerLine"	STRING	Maps to Source File has Headings/Labels field in Load Table Properties Source tab in RED
..|..|..|.."name"	STRING	Maps to Source File Name field in Load Table Properties Source tab in RED
..|..|..|.."nonStringNullEncoding"	STRING	Maps to Loader Options field in Load Table Properties Source tab in RED
..|..|..|.."nullEncoding"	STRING	Maps to Loader Options field in Load Table Properties Source tab in RED
..|..|..|.."path"	STRING	Maps to Source File path field in Load Table Properties Source tab in RED
..|..|..|.."recordDelimiter"	STRING	Maps to Source File Record Terminator field in Load Table Properties Source tab in RED
..|..|.."sourceSchema"	STRING	Maps to Source Schema field in Load Table Properties Source tab in RED
..|..|.."sourceTables"	STRING	Maps to Source Table(s) field in Load Table Properties Source tab in RED
..|..|.."useOverrideSourceColumns"	BOOLEAN := TRUE|FALSE	Maps to Override Source Column/Transformations field in Load Table Properties Source tab in RED
..|..|.."whereAndGroupByClauses"	STRING	Maps to Where and Group By Clauses field in Load Table Properties Source tab in RED
..|.."uiConfigLoadTableProperties"		Node containing the array (if any) of field value pairs to map to any Configured Load Table UI fields.
..|..|.."a_configured_load_table_UI_field"	STRING|NUMERIC	Maps to "a_configured_load_table_UI_field" field in Load Table Properties Source tab in RED

Scripted Browse Workflow

The Scripted Browse Workflow performs the following steps when browsing an Extensible Source Connection:

1. Prepares the Browse Script and run-time files for execution against the associated Windows Script Connection.
2. Executes the Script and if a success code of 1 is returned:
    1. Reads the second line of output.
    2. Checks if the output is valid JSON.
    3. Checks if the output has the expected attribute structure.
    4. Reads the structure into the Source Browser Tree for the purpose of creating Load Tables from the items displayed.
    5. Deletes all temporary files created during the Scripted Browse Workflow.
3. Or - if the script returns a failure code (-1,-2) then the temporary files are preserved for debugging and the output is not loaded into the Source Browser Tree.

Any object displayed in the Source Browser Tree can be dragged into RED as a Load Table Object as normal and any valid metadata set in the output JSON structure is carried across to RED for that object.

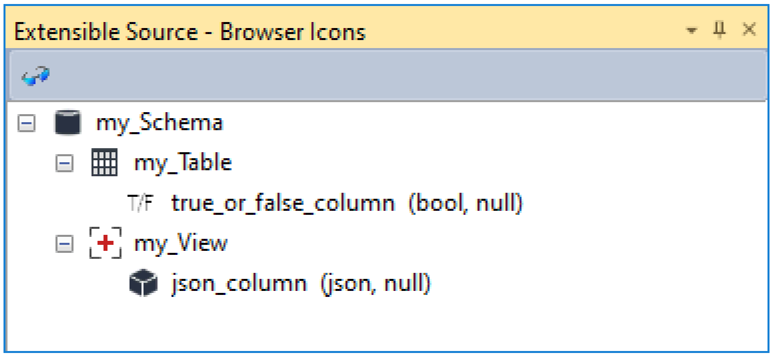
Icon Configuration for the Source Browser Tree

The configuration of the Source Browse Tree icons for Extensible Source Connections helps to distinguish and identify categories easier.

The *treeViewCategory* is an optional attribute that can appear at the Table and Column node level. Each Table or Column can have an association to a *treeViewCategory* that will be matched with the *treeViewIcon* category key and will display the icon you want to use to identify the node.

Define "*category key":"icon name*" pairs in the *treeViewIcons* attributes, where 'category key' is an arbitrary string to act as a key for your category and 'icon name' is either the file name of the icon file to use without path (assumes the icon is located in the installation folder 'Icons' directory) or the full file path to the icon file. If there is an issue loading the icon for a particular node RED will fallback to the default icons without reporting an error.

The image below shows an example of customized tree view icons followed by the output JSON structure that was used to create it:


Extensible Source - Browser Icons
⊟ my_Schema
⊟ my_Table
T/F true_or_false_column (bool, null)
⊟ my_View
json_column (json, null)

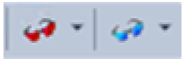
```
$outputJson = @"
{
    "treeViewLayout": "Tabular",
    "treeViewIcons": {
        "schema": "Database.ico",
        "table": "table.ico",
        "view": "view.ico",
        "object": "DataTypeObject.ico",
    "boolean": "DataTypeBoolean.ico"
    },
    "my_Schema": {
        "my_View": {
            "treeViewCategory": "view",
            "name": "myView",
            "description": "description of table: myView",
            "rowCount": 0,
            "columns": [{
                    "treeViewCategory": "object",
                    "name": "json_column",
                    "dataType": "json",
                    "dataTypeLength": null,
                    "dataTypeScale": null,
                    "dataTypePrecision": null,
                    "nullAllowed": true,
                    "defaultValue": "json_column",
                    "description": "description of json_column",
                    "displayName": "display as json_column",
                    "format": "",
                    "additive": false,
                    "numeric": false,
                    "attribute": true,
                    "sourceTable": "Icons",
                    "sourceColumn": "json_column",
                    "transform": "",
                    "transformType": "",
                    "uiConfigColumnProperties": {}
                }
            ],
            "loadInfo": {
                "fileLoaderOptions": "",
                "fileParsed": false,
                "overrideLoadSQL": "",
                "overrideSourceColumns": "",
                "selectDistinctValues": false,
                "sourceFile": {
                    "charSet": "",
                    "escapeEncoding": "",
                    "fieldDelimiter": ",",
                    "fieldEnclosure": "\"",
                    "headerLine": true,
                    "name": "",
                    "nonStringNullEncoding": "",
                    "nullEncoding": "",
                    "path": "",
                    "recordDelimiter": ""
                },
                "sourceSchema": "mySourceSchema",
                "sourceTables": "mySourceTables",
                "useOverrideSourceColumns": false,
                "whereAndGroupByClauses": "Where 1=1"
            },
            "uiConfigLoadTableProperties": {}
        },
        "my_Table": {
            "treeViewCategory": "table",
            "name": "myTable",
            "description": "description of table: myTable",
            "rowCount": 0,
            "columns": [{
                    "treeViewCategory": "boolean",
                    "name": "true_or_false_column",
                    "dataType": "bool",
                    "dataTypeLength": null,
                    "dataTypeScale": null,
                    "dataTypePrecision": null,
                    "nullAllowed": true,
                    "defaultValue": "true_or_false_column",
                    "description": "description of true_or_false_column",
                    "displayName": "display as true_or_false_column",
                    "format": "",
                    "additive": false,
                    "numeric": false,
                    "attribute": true,
                    "sourceTable": "Icons",
                    "sourceColumn": "true_or_false_column.ico",
                    "transform": "",
                    "transformType": "",
                    "uiConfigColumnProperties": {}
                }
            ],
            "loadInfo": {
                "fileLoaderOptions": "",
                "fileParsed": false,
                "overrideLoadSQL": "",
                "overrideSourceColumns": "",
                "selectDistinctValues": false,
                "sourceFile": {
                    "charSet": "",
                    "escapeEncoding": "",
                    "fieldDelimiter": ",",
                    "fieldEnclosure": "\"",
                    "headerLine": true,
                    "name": "",
                    "nonStringNullEncoding": "",
                    "nullEncoding": "",
                    "path": "",
                    "recordDelimiter": ""
                },
                "sourceSchema": "mySourceSchema",
                "sourceTables": "mySourceTables",
                "useOverrideSourceColumns": false,
                "whereAndGroupByClauses": "Where 1=1"
            },
            "uiConfigLoadTableProperties": {}
        }
    }
}
"@
```

```
Write-Output 1
# *IMPORTANT* remove line endings so that the output is printed in a single
line
# This avoids "Invalid JSON" error as the recieving process only consumes
the
# second line of the standard-out as its entire input argument.
Write-Output $($outputJson -replace '\r\n', '')
# For debugging the JSON has been output to the user's temp directory
Out-File -InputObject $outputJson -FilePath $env:Temp"\output_test.json"
```

Browsing a Connection

The tables or files associated with a **Connection** can be displayed in the **Browser** pane on the right side of the **Builder** window by:

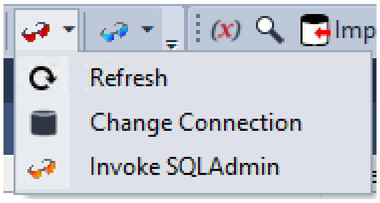
1. Selecting the **Browse > Data Warehouse** menu option to browse for the data warehouse connection,
2. Selecting the **Browse > Source Tables** menu option to browse a source system connection,
3. Right-clicking a Connection in the Objects list pane and selecting **Browse Connection** from the context menu, or
4. Clicking one of the two browser buttons from the toolbar:



The red button is used to browse the data warehouse connection and the blue button is used to browse a source system connection.

Each button remembers the last connection it browsed, so in this way one button can be used for the Data Warehouse and one for a source system.

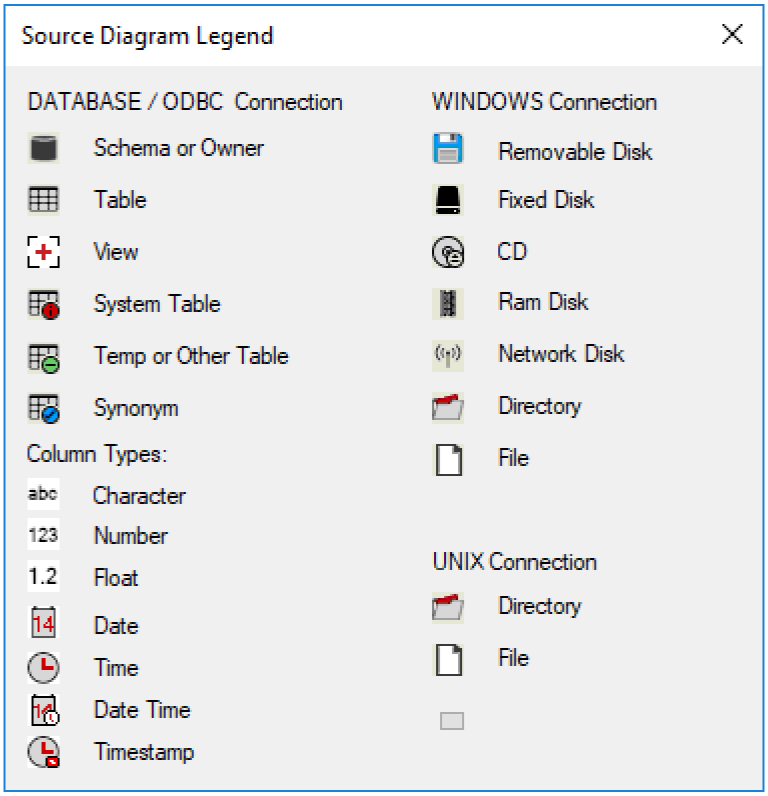
Clicking one of the buttons, displays the source tables without first displaying the source browser window. To change the connection, click the down arrow next to one of the browser buttons and then select **Change Connection**.



The current connection being browsed is shown in the status bar at the bottom right of the screen.

Browser Icons

When browsing a connection, the following legend applies for the source tables and objects.
The legend is displayed via the **Help > Source Legend** menu:



Source Diagram Legend

DATABASE / ODBC Connection	WINDOWS Connection
Schema or Owner	Removable Disk
Table	Fixed Disk
View	CD
System Table	Ram Disk
Temp or Other Table	Network Disk
Synonym	Directory
Column Types:	File
abc Character	
123 Number	
1.2 Float	UNIX Connection
14 Date	Directory
Time	File
Date Time	
Timestamp	

Connection Browse Properties

Note

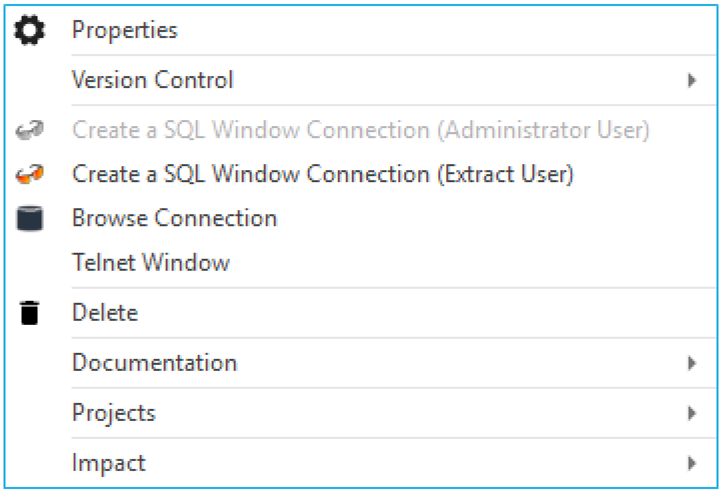
This screen is not applicable to Extensible Source Connections.

Tip

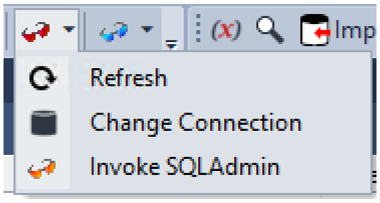
When browsing to a Connection, leave the schema field blank to see all schemas. To make RED browse a specific schema or schemas by default, go to a Connection's properties screen and enter the schema(s) to browse on the **Default Schema for Browsing** field.

To change the properties of the connection in the Browser pane:

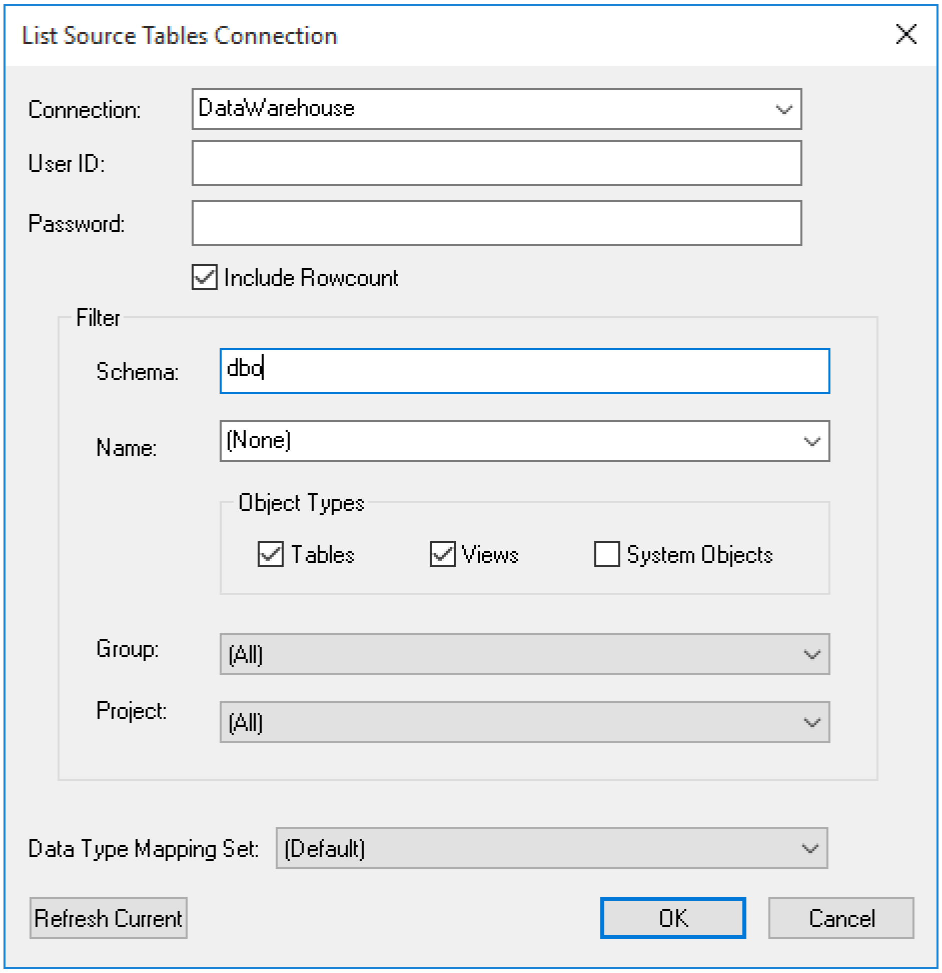
- Right-click a connection in the Objects list pane and select **Browse Connection**, or


Properties
Version Control
Create a SQL Window Connection (Administrator User)
Create a SQL Window Connection (Extract User)
Browse Connection
Telnet Window
Delete
Documentation
Projects
Impact

- Click the drop-down arrow next to one of the Browser buttons on the toolbar and select **Change Connection**.



The **List Source Tables Connection** window is displayed:



List Source Tables Connection

Connection: DataWarehouse

User ID:

Password:

☑ Include Rowcount

Filter

Schema: dbo

Name: (None)

Object Types

☑ Tables ☑ Views □ System Objects

Group: (All)

Project: (All)

Data Type Mapping Set: (Default)

This window enables you to change the properties of the connection you are browsing.

The **User ID** and **Password** fields can be changed in order to browse the connection as a different user.

Selecting the **Include Rowcount** check-box displays a row count in brackets next to each source table in the Browser pane. This is only available for databases which update table statistics.

Note

For **Custom** database connections, the **Include Rowcount** option in the **List Source Tables Connection** screen is only available if the **Bulk Table Row Count SQL** field in the **Connection > Target Settings** tab has been defined. Refer to **Connection Target Settings for Custom Database** for details.

A **Filter** can also be applied when browsing a connection. Filters can be applied to any combination of:

- One or more Schema names (separated by commas),
- a standard SQL table name,
- specific Object Types (Tables, Views or System Objects),
- a Group, or
- a Project

The **Data Type Mapping Set** drop-down can be used to change the data type conversion used during drag and drop operations. If set to **(Default)**, the Data Type Mapping to use for each drag and drop operation is set based on the source and the Target Location selected. However, this can be changed in the **Add New Metadata Object** window, if required.

Changing a Connection's Properties

Whenever a connection's **Properties** are changed, the impact on the objects that use that connection must be considered.

Load tables have information from the connection stored within their **Properties**. This information is stored in the Load objects to minimize the complexity of the scheduled tasks.

The database link and database name are stored locally in each Load table. When either the database link or database name are changed on a connection, WhereScape RED displays the **Update Associated Load Tables** window.

Click **Yes** to automatically update the database link and/or database name on all associated Load tables.

This can also be done manually:

1. Double-click the **Load Tables** object group in the left pane. This displays all Load tables in the middle pane.
2. Select those Load tables that use the connection that has changed. Use the Windows standard keyboard keys (CTRL or SHIFT) for selecting multiple Load tables.
3. Right-click to open a context menu and then select **Change Connect/Schema**.
4. Select a different connection (e.g. Data warehouse) to change all the selected Load tables.
5. Repeat step (3) and now change the tables back to the altered connection. This updates all the Load tables with the new connection information.

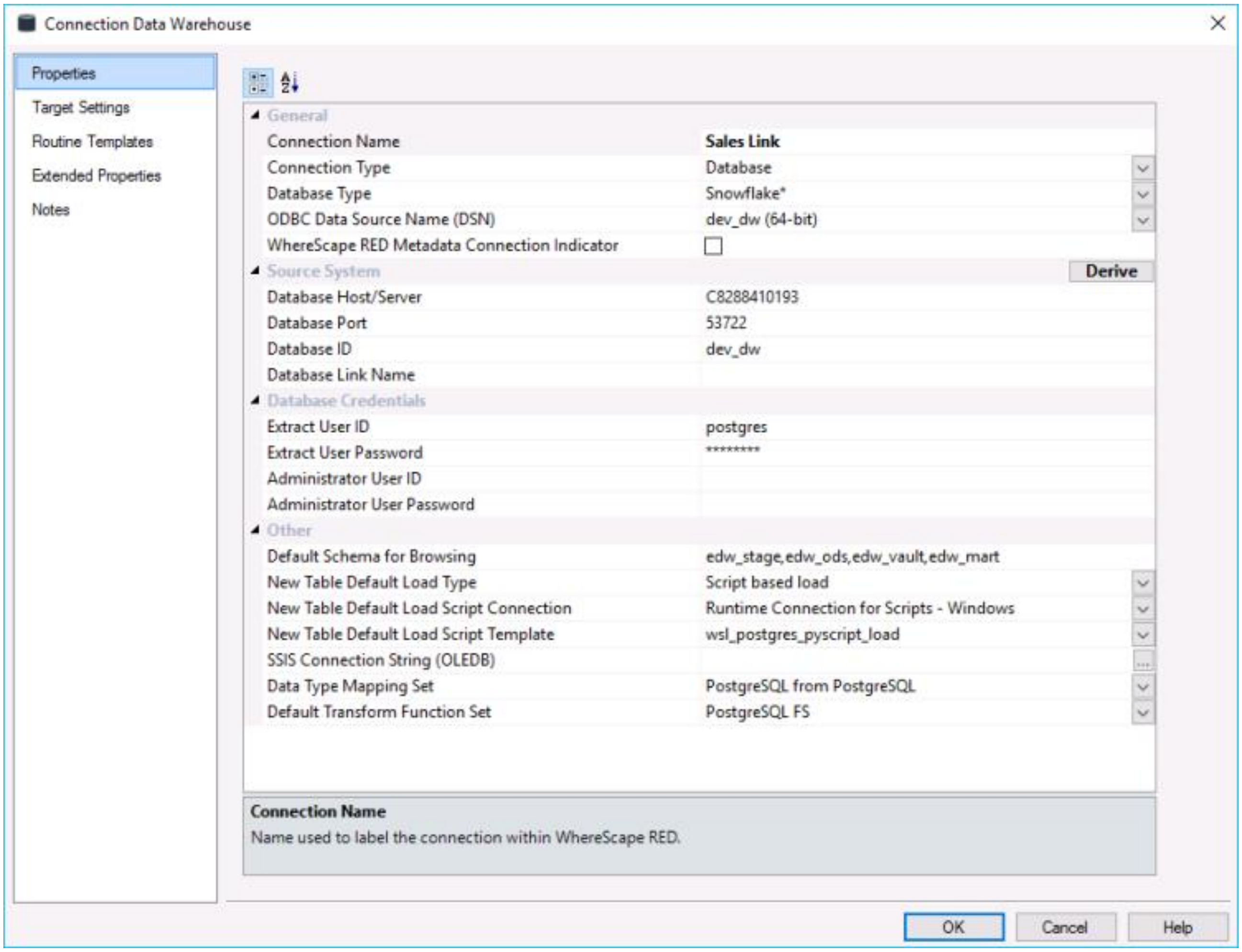
Creating a Database Link

To create a database link connection, the database link must be established by clicking the **Create Database Link** option from the context menu, after the relevant connection has been created in RED.
Any existing connections can be used as a database link connection if:

- the connection type is **Database**;
- the **Database Link Name** field is populated with **a name** for the database link;
- the **Create Database Link** option has been selected from the connection's right click context menu. A confirmation message is displayed when the database link has been created.

To create a **new** Database Link type connection you need to do the following:

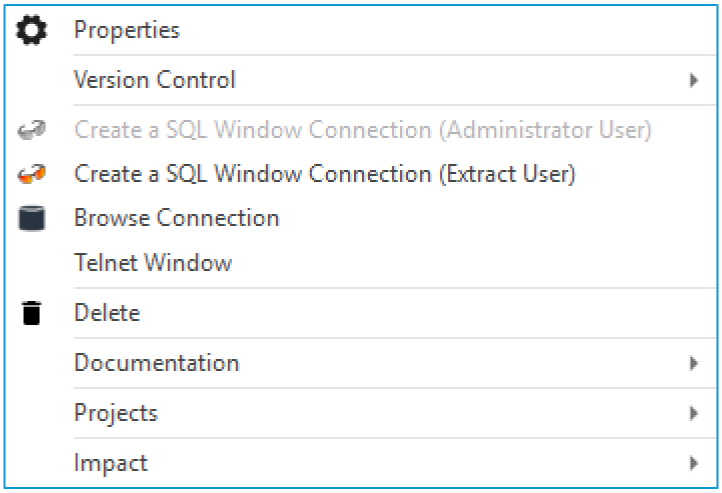
1. Enter a **name** for the new Connection.
2. Set the Connection Type to **Database.**
3. Select an **ODBC Source (DSN)** to extract the source data.
4. Enter a **Database Link Name** for the database link. This is a required field for establishing the **Database Link** connection.
5. Enter the **Extract User ID** and **Extract User Password** for the user that has access to the source database when not using Windows Authentication.



General	
Connection Name	**Sales Link**
Connection Type	Database
Database Type	Snowflake*
ODBC Data Source Name (DSN)	dev_dw (64-bit)
WhereScape RED Metadata Connection Indicator	□
Source System	
Database Host/Server	C8288410193
Database Port	53722
Database ID	dev_dw
Database Link Name	
Database Credentials	
Extract User ID	postgres
Extract User Password	********
Administrator User ID	
Administrator User Password	
Other	
Default Schema for Browsing	edw_stage,edw_ods,edw_vault,edw_mart
New Table Default Load Type	Script based load
New Table Default Load Script Connection	Runtime Connection for Scripts - Windows
New Table Default Load Script Template	wsl_postgres_pyscript_load
SSIS Connection String (OLEDB)	
Data Type Mapping Set	PostgreSQL from PostgreSQL
Default Transform Function Set	PostgreSQL FS

Connection Name
Name used to label the connection within WhereScape RED.

6. Once a connection has been set up, you can right-click the connection name to see the following menu:


Properties
Version Control
Create a SQL Window Connection (Administrator User)
Create a SQL Window Connection (Extract User)
Browse Connection
Telnet Window
Delete
Documentation
Projects
Impact

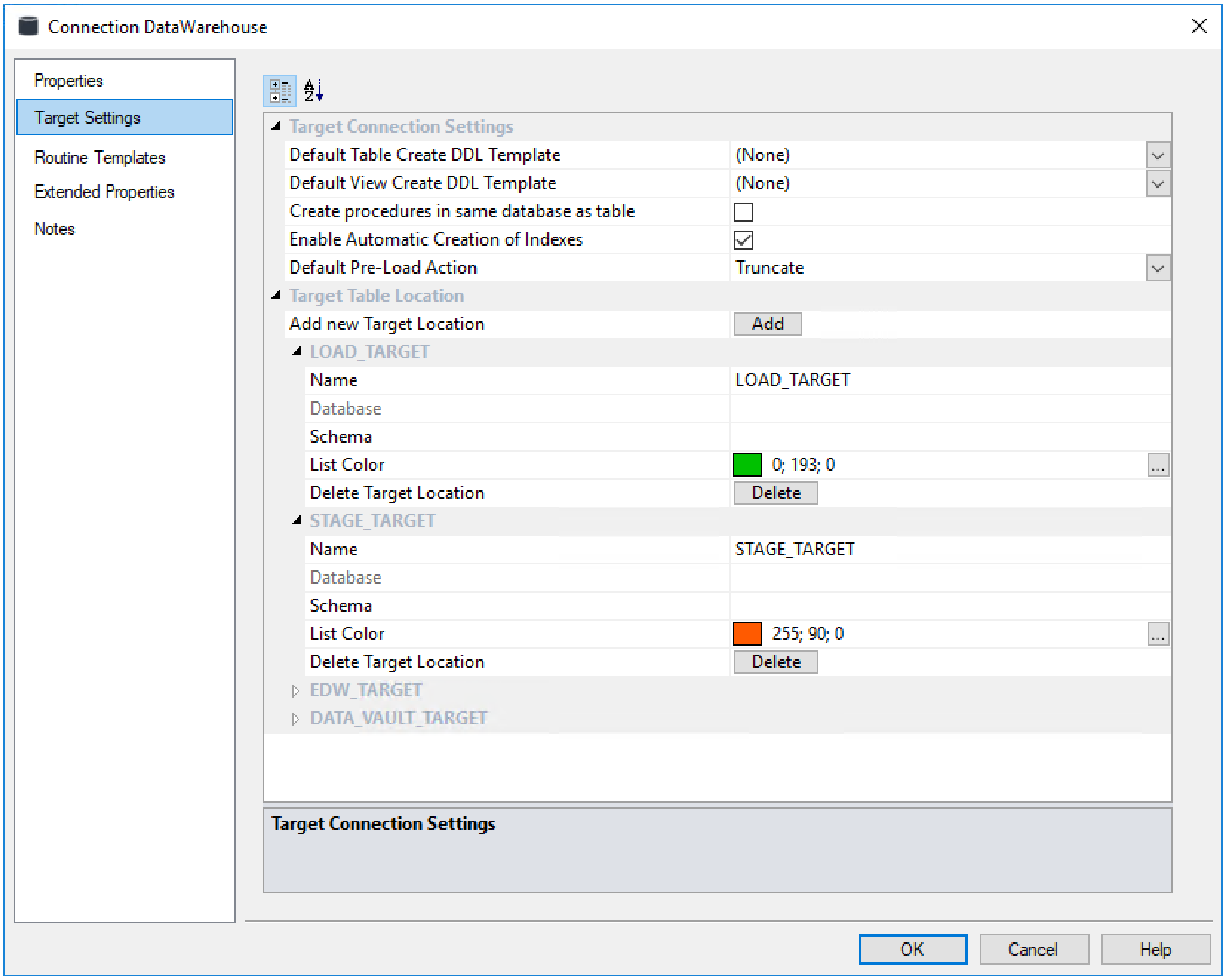
7. Once the **Create Database Link** option is selected, RED attempts to create a user database link to the source database. If the link already exists, a prompt appears to confirm if you want to overwrite the existing link.

Tips:

A number of problems can occur during this action, and your database administrator should be consulted:

Connection Target Settings

These settings enable you to define target connection settings and target table locations.



Target Connection Settings	
Default Table Create DDL Template	(None)
Default View Create DDL Template	(None)
Create procedures in same database as table	□
Enable Automatic Creation of Indexes	☑
Default Pre-Load Action	Truncate
Target Table Location	
Add new Target Location	Add
LOAD_TARGET	
Name	LOAD_TARGET
Database	
Schema	
List Color	🟩 0; 193; 0
Delete Target Location	Delete
STAGE_TARGET	
Name	STAGE_TARGET
Database	
Schema	
List Color	🟧 255; 90; 0
Delete Target Location	Delete
EDW_TARGET	
DATA_VAULT_TARGET	

Target Connection Settings

Target Connection Settings

Options	Description
Default Table Create DDL Template	The default Template used for generating DDL for new Tables created on targets of this connection.
Default View Create DDL Template	The default Template used for generating DDL for new Views created on targets of this connection.
Create procedures in same database as table	This setting enables you to specify if the procedures associated with new Tables created on Targets of this connection is created in the same database as the Table objects, or in the RED repository meta database. This option is not selected (OFF) by default for existing RED repositories, e.g. the Table's stored procedures are created in the RED repository meta database. If the Table is created in a different Target location (e.g. distinct schema/database in the same SQL Server instance), you can turn ON this option if you want the stored procedures to also be located in the same Target database as the Table object. Refer to **Distributing Tables across Multiple SQL Server Databases** for details.

Options	Description
	Notes: • This setting only applies for SQL server Target databases. For new RED repositories, this option is ON by default (e.g. procedures are created in the same location as the Table object). New or regenerated procedures are created in the specified location. • Changing this setting has no impact on existing stored procedures. You must **Rebuild** or **Regenerate** the stored procedures for the changes to take effect. • If the **Lock Storage Location** option in the **Procedure Properties** window is set, then the current location of the procedure is retained, e.g. no changes will occur if the Target location of the procedure is manually changed and a **Rebuild** or **Regenerate** operation is performed via the Table object's **Properties** window or from the Table object's right click **Code** context menu, • If the Target location of a Table object is changed, then both the **Re-creation** of the Table object (as in previous versions of RED) and the **Regeneration** of the Procedure object is required for both objects to be moved. Refer to **Procedure Migration** for details.
Enable Automatic Creation of Indexes	This setting enables you to turn ON/OFF the automatic creation of metadata for indexes when creating new objects. Neither the RED metadata nor the physical index is created, if this setting is not enabled for objects. This option is selected (ON) by default. You can turn OFF this option for databases that do not have indexes, and if you do not require RED to automatically create metadata indexes for objects. **Notes:** • This setting only applies when creating metadata indexes for objects and does not affect existing metadata index records. • Turning this option OFF does not delete existing metadata records or the physical index.
Default Pre-Load Action	The default Pre-Load Action to use for Load tables created on targets of this connection. Options include: • Truncate • Execute Pre-load SQL • Both Truncate and Execute Pre-Load SQL • No action If **Execute Pre-Load SQL** is selected, then the SQL statement to execute before the load takes place must be entered—refer to Pre-Load Action in **Database Link Load Properties Screen** for details.
Column Properties Configuration	Lists all the available Column Properties UI Configurations in the metadata.

Target Table Location [For target enabled licenses]

Options	Description
Add new Target Location	This option enables you to add new database/schema locations for objects in this connection. For Greenplum, Netezza and PDW schema target locations setup, refer to **Database source system - Local/Linked Servers** or **ODBC** connection topics, depending on the connection type chosen for the database source.

Options	Description
	1. Click the **Add** button to add the required target schemas for this connection. 2. Give the new target a name and then enter the target's schema. 3. Default schema location(s) for **New Tables** can also be set from the **Tools > Options** menu. Refer to Settings > Storage > Target Location for details.

Notes:

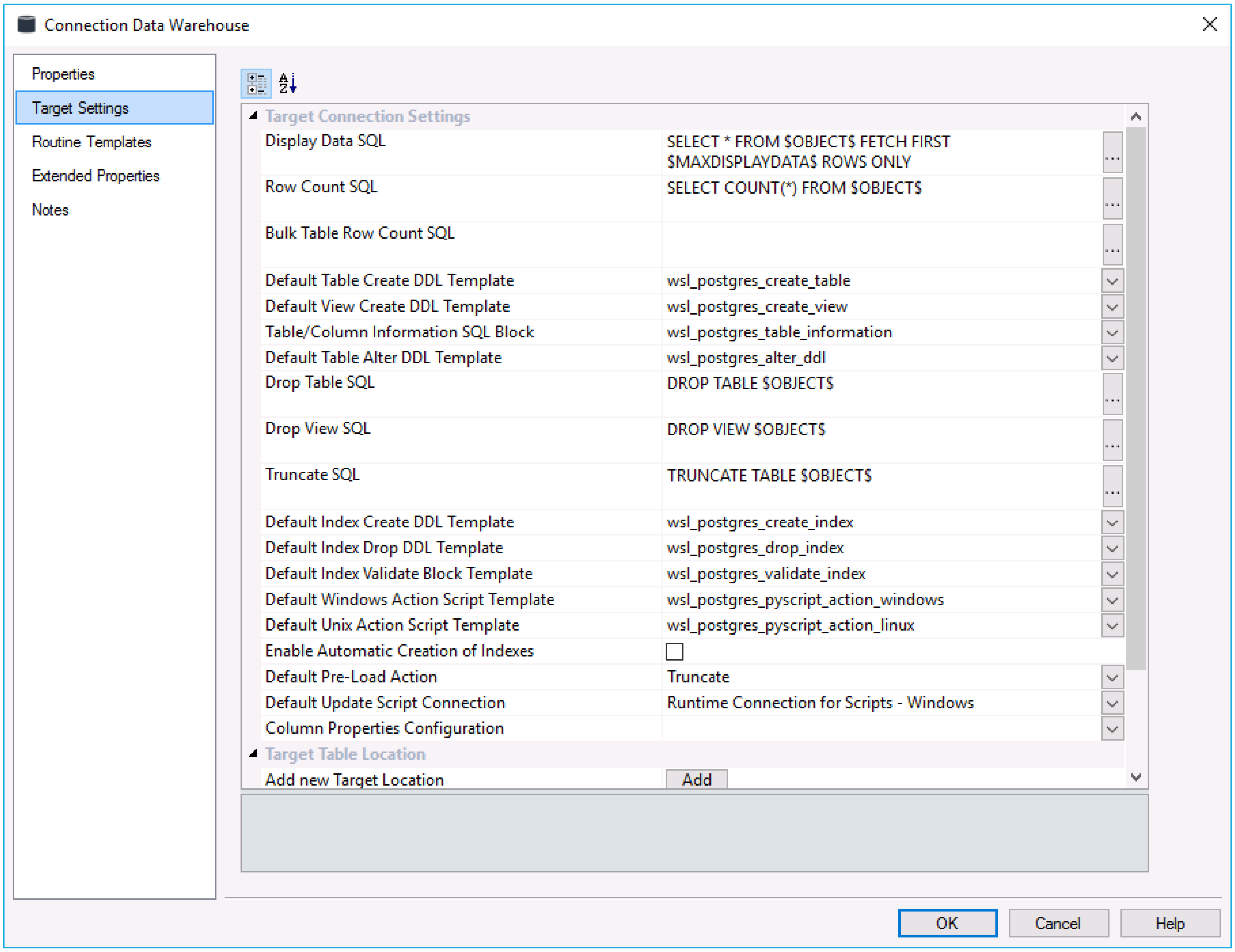
The database and schema names for **Custom** database connections are used as follows:
<database>.<schema>.object name
Leave database name blank if not required. Leave schema name blank to use the default schema.

Connection Target Settings for Custom Database

The **Target Settings** tab for custom database connections provide additional fields for defining default DDL templates to use for the creation of tables, indexes and views in custom database targets.

Note:

Please ensure that you have installed the WhereScape supplied templates or created your own templates, before configuring the settings below.



Target Connection Settings

Display Data SQL	SELECT * FROM $OBJECT$ FETCH FIRST $MAXDISPLAYDATA$ ROWS ONLY
Row Count SQL	SELECT COUNT(*) FROM $OBJECT$
Bulk Table Row Count SQL	
Default Table Create DDL Template	wsl_postgres_create_table
Default View Create DDL Template	wsl_postgres_create_view
Table/Column Information SQL Block	wsl_postgres_table_information
Default Table Alter DDL Template	wsl_postgres_alter_ddl
Drop Table SQL	DROP TABLE $OBJECT$
Drop View SQL	DROP VIEW $OBJECT$
Truncate SQL	TRUNCATE TABLE $OBJECT$
Default Index Create DDL Template	wsl_postgres_create_index
Default Index Drop DDL Template	wsl_postgres_drop_index
Default Index Validate Block Template	wsl_postgres_validate_index
Default Windows Action Script Template	wsl_postgres_pyscript_action_windows
Default Unix Action Script Template	wsl_postgres_pyscript_action_linux
Enable Automatic Creation of Indexes	□
Default Pre-Load Action	Truncate
Default Update Script Connection	Runtime Connection for Scripts - Windows
Column Properties Configuration	

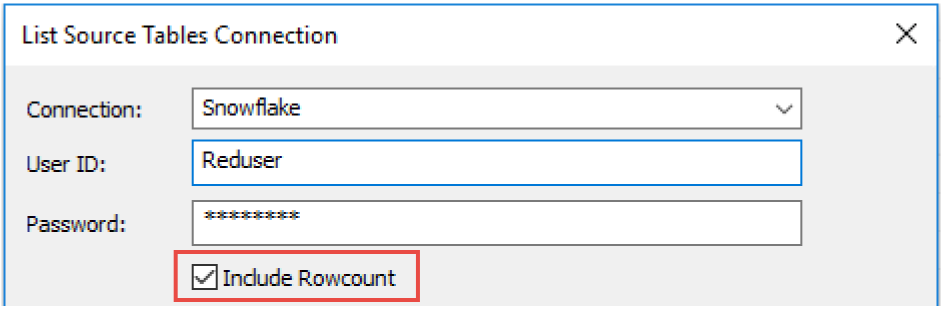
Target Table Location

Add new Target Location	Add

Target Connection Settings

In addition to the standard settings described in the previous section, the following settings are available for custom database connections:

Field	Description
Display Data SQL	The generic syntax of a Display Data SQL statement which should use RED placeholder parameters (e.g. $OBJECT$, $MAXDISPLAYDATA$). **Note:** The $MAXDISPLAYDATA$ parameter is derived from the **User Preferences > Common > Look and Feel > General > Maximum rows returned for Display Data** setting.
Row Count SQL	The generic syntax of a Row Count SQL statement which should use RED placeholder parameters (e.g. $OBJECT$).
Bulk Table Row Count SQL	The SQL statement used to return the table name, schema name and row count of all tables for the connection. For example: **SELECT table_name, schema_name, row_count FROM information_schema.tables**. The variable $DATABASE$ can be used. The SQL statement defined in this field is used in the **Browser Connection screen** if the

Field	Description
	Include Rowcount option is enabled.  List Source Tables Connection Connection: Snowflake User ID: Reduser Password: ********* ☑ Include Rowcount **Notes:** • The **Include Rowcount** option in the Browser Connection screen for Custom database connections is only available, if this field has been defined. • If the **Include Rowcount** option is enabled for a Custom database connection, then the SQL statement defined in this field is used to return the list of tables, schema and row counts as the source of the row counts; instead of the SQL statement defined internally in RED. • The SELECT statement must follow the order "table_name, schema_name, row_count" as shown in the example above.
Default Table Create DDL Template	The default Template used for generating DDL for new Tables created on targets of this connection. **Note:** This template is used in the validation done for custom database targets during Application Load Processing performed in the WhereScape Setup Administrator. Refer to Compile New and Changed Procedures in the RED Installation Guide for details.
Default View Create DDL Template	The default Template used for generating DDL for new Views created on targets of this connection
Table/Column Information SQL Block	The SQL Block Procedure used to query Table/Columns information; should use RED placeholder parameters (e.g. $DATABASE$, $SCHEMA$, $OBJECT$).
Default Table Alter DDL Template	The default Template used for generating DDL to alter existing Tables created on targets of this connection
Drop Table SQL	The generic syntax of a Drop Table SQL statement which should use RED placeholder parameters (e.g. $OBJECT$).
Drop View SQL	The generic syntax of a Drop View SQL statement which should use RED placeholder parameters (e.g. $OBJECT$).
Truncate SQL	The generic syntax of a Truncate SQL statement which should use RED placeholder parameters (e.g. $OBJECT$).
Default Index Create DDL Template	The default Template used for generating DDL for new Indexes created on targets of this connection.
Default Index Drop DDL Template	The default Template used for generating DDL to drop existing Indexes created on targets of this connection.
Default Index Validate Block	The default Template used for generating a block of SQL Statements to validate indexes created on targets on this connection.

Field	Description
Template	
Default Windows Action Script Template	The default Template used for generating a Windows Script to perform actions on objects created on targets of this connection.
Default Unix Action Script Template	The default Template used for generating a Unix Script to perform actions on objects created on targets of this connection.
Enable Automatic Creation of Indexes	RED's default behavior is to create indexes when creating new objects.
Default Pre-Load Action	The default Pre-Load Action to use for Load tables created on targets of this connection. Options include: • Truncate • Execute Pre-load SQL • Both Truncate and Execute Pre-Load SQL • No action If **Execute Pre-Load SQL** is selected, then the SQL statement to execute before the load takes place must be entered—refer to Pre-Load Action in **Database Link Load Properties Screen** for details.
Default Update Script Connection	The default Script Connection for new Update Scripts in tables created using this connection. This setting is only displayed in connections which have a target database type for which Update Script functionality is enabled, e.g. Custom database type.
Column Properties Configuration	List view of the available Column Properties Configs.

Notes:

The following can be used as RED placeholder parameters:
$OBJECT$
$DATABASE$
$SCHEMA$
$TABLE$

The DDL settings defined in the Table Properties > **Storage** tab overrides the DDL settings defined in the **Target Settings** tab for custom database connections above.

Target Table Location [For target enabled licenses]

Options	Description
Add new Target Location	This option enables you to add new database/schema locations for objects in this connection. For Greenplum, Netezza and PDW schema target locations setup, refer to **Database**

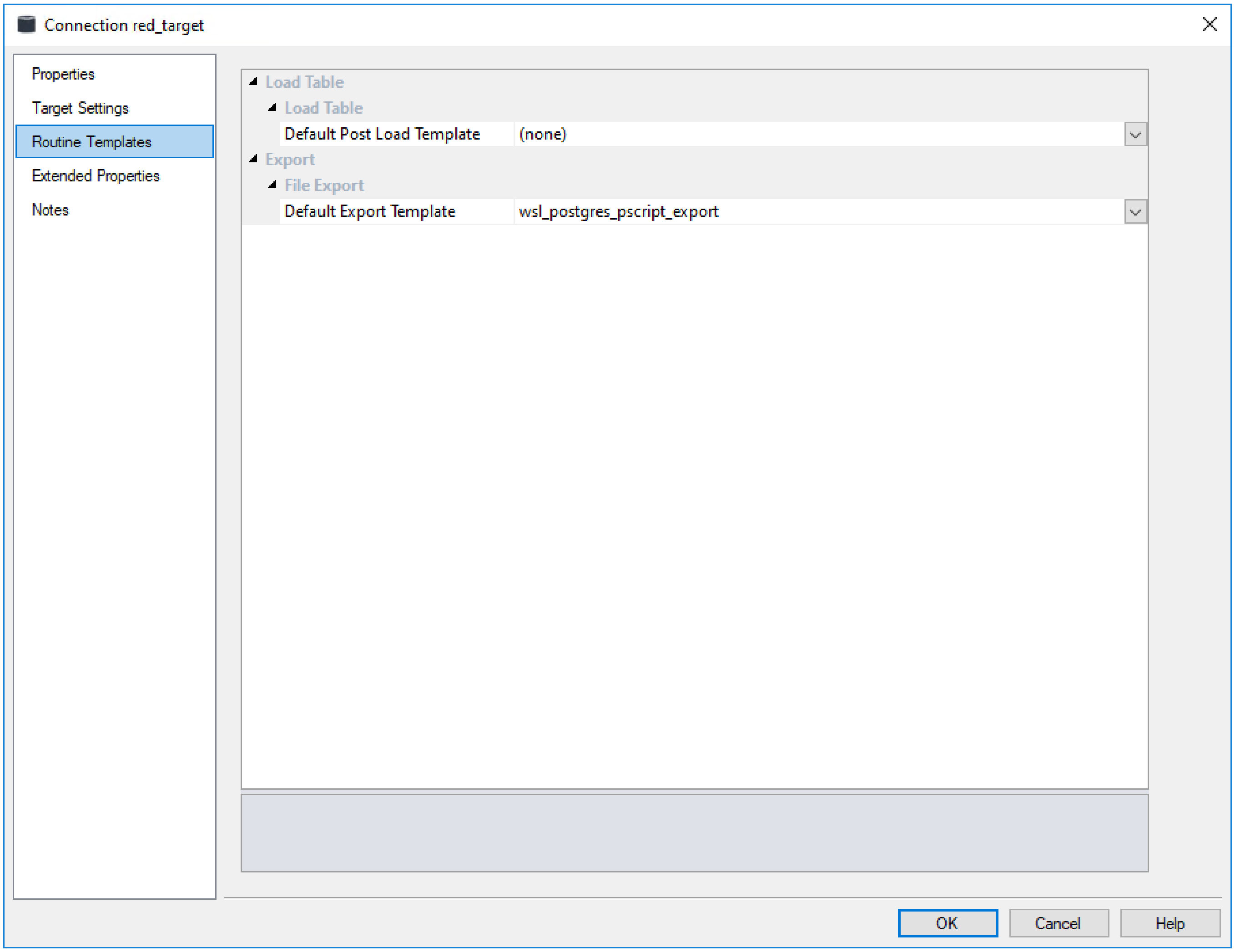
Options	Description
	source system - Local/Linked Servers or **ODBC** connection topics, depending on the connection type chosen for the database source. 1. Click the **Add** button to add the required target schemas for this connection. 2. Give the new target a name and then enter the target's schema. 3. Default schema location(s) for **New Tables** can also be set from the **Tools > Options** menu. Refer to **Settings > Storage > Target Location** for details.

Notes

The database and schema names for **Custom** database connections are used as follows:
<database>.<schema>.object name
Leave database name blank if not required. Leave schema name blank to use the default schema.

Connection Routine Templates

These settings enable you to set the default templates to use for the update routines of load and export objects associated with the selected connection. To set the default routine templates for table types other than Loads and Exports please refer to the **Object Subtypes** section.



Connection red_target

- Load Table
  - Load Table
    - Default Post Load Template: (none)
- Export
  - File Export
    - Default Export Template: wsl_postgres_pscript_export

Notes:

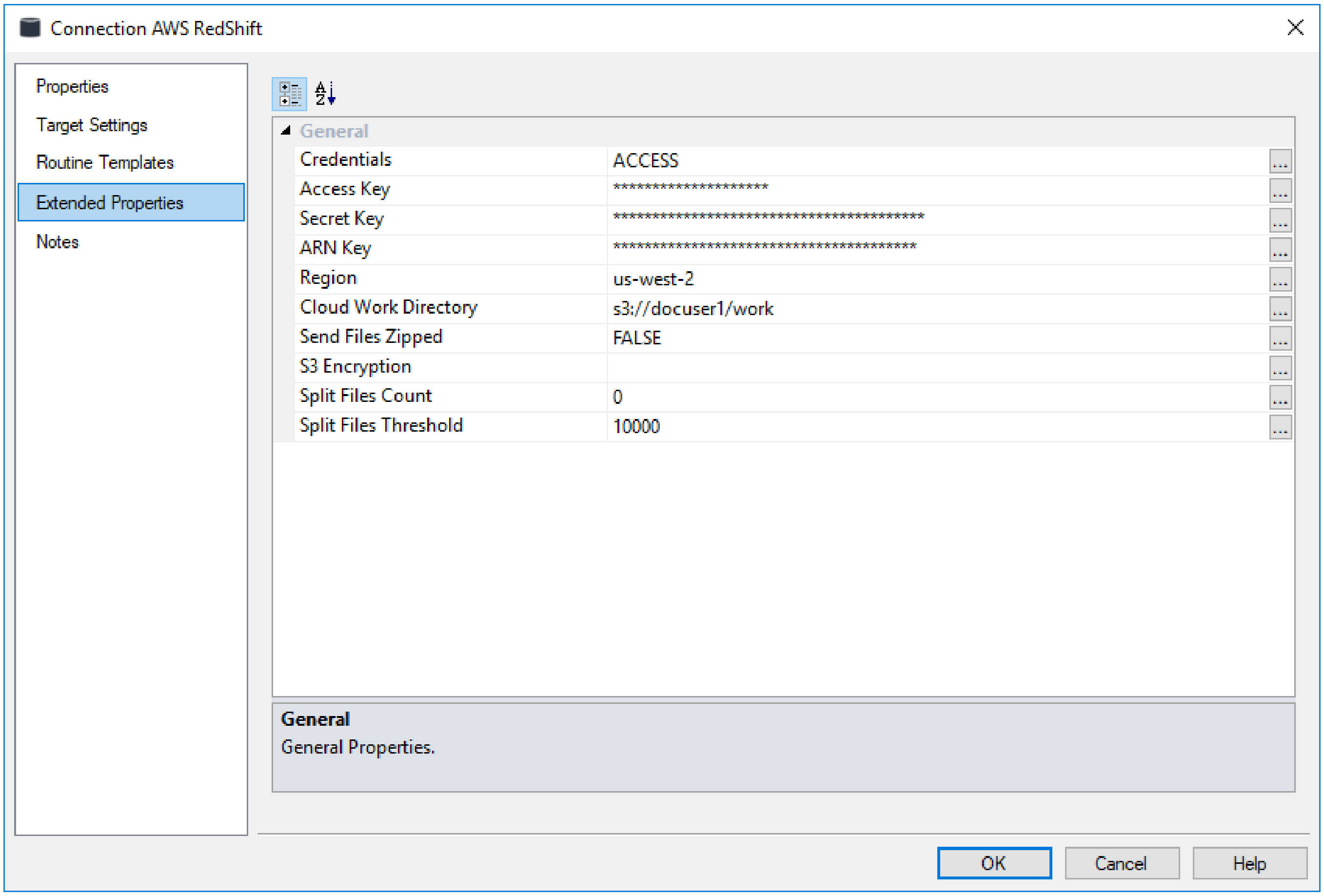
- Default Load Script Templates can be set in the source connection **Properties** tab.
- DDL Templates can be set in the **Target Settings** tab of the target connection.
- The settings in the Routine Templates tab is used in deploying applications into RED via the **REDCLI** command '`object generate-routine`'. Refer to **Dedicated Command Line Interface (REDCLI)** for details.
- The Routine Template defaults can also be set using **REDCLI**, for example:

```
RedCli>connection set-default-template --con-name "MyTargetConnection" --obj-type "Stage" --obj-sub-type "DataVaultStage" --op-type "LoadScript" --tem-name "wsl_pyscript_load"
```

Connection Extended Properties

Note

These settings enable you to assign extended property values for a connection. Refer to **Extended Properties** for details.



General

Credentials	ACCESS
Access Key	********************
Secret Key	**
ARN Key	***************************************
Region	us-west-2
Cloud Work Directory	s3://docuser1/work
Send Files Zipped	FALSE
S3 Encryption	
Split Files Count	0
Split Files Threshold	10000

General
General Properties.

Loading Data

Load tables are the first entry point for all information coming into the data warehouse. There are multiple ways of getting data into Load tables.

Load Type	Description
Database Link Load	The data is loaded from another database
External Load	The Load table is populated by some external process, e.g. an ETL (Extract, Transform and Load) tool or EAI (Enterprise Application Integration) tool, putting the data directly into the Load tables.
ODBC Load	The data is loaded via an ODBC connection, either directly by reading and inserting each row (a Load Type of **ODBC load**), or via a file by reading from the source system using ODBC and writing to a file and then loading the file (a Load Type of **Native ODBC**). ODBC connections require a Windows scheduler.
File Load	A flat file load where most of the processing is managed and controlled by the scheduler.
Script-based load	A flat file load where a host system, e.g. UNIX or Windows script file is executed to perform the load. Script-based loads on Windows supports both DOS Batch and **PowerShell scripts**.
XML File load	This method loads an XML file from a Windows directory.

Notes:

A **Windows scheduler** must be available to handle ODBC based loads.
Refer to **Tools > Options > Available Load Types** for details.

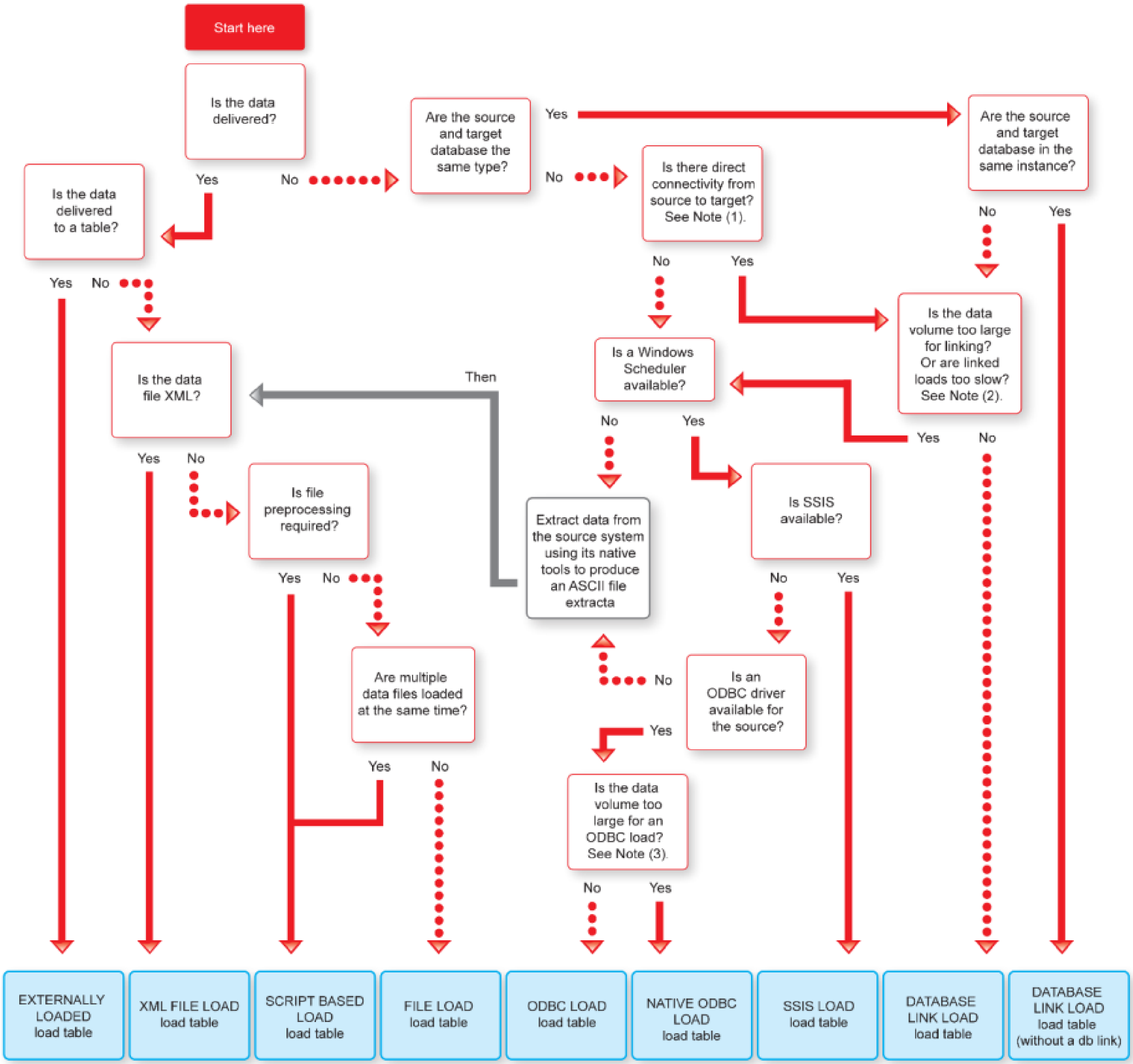
Choosing the Best Load Method

Several different factors need to be considered when choosing the best type of load table to use:

- Source and target database types
- Locations of source and target databases
- Available connectivity options and performance
- Amount of data being loaded
- Is the data delivered or fetched?
- For delivered data, what format is it in and does it require processing to make it loadable?

Load Table Decision Tree

The following decision tree can be used when selecting the best type of load table:


Start here
Is the data delivered?
Yes
No
Is the data delivered to a table?
Yes
No
Are the source and target database the same type?
Yes
No
Are the source and target database in the same instance?
No
Yes
Is there direct connectivity from source to target? See Note (1).
No
Yes
Is the data volume too large for linking? Or are linked loads too slow? See Note (2).
Yes
No
Is a Windows Scheduler available?
No
Yes
Then
Is the data file XML?
Yes
No
Is file preprocessing required?
Yes
No
Extract data from the source system using its native tools to produce an ASCII file extracta
Is SSIS available?
No
Yes
Are multiple data files loaded at the same time?
Yes
No
No
Is an ODBC driver available for the source?
Yes
Is the data volume too large for an ODBC load? See Note (3).
No
Yes
EXTERNALLY LOADED load table
XML FILE LOAD load table
SCRIPT BASED LOAD load table
FILE LOAD load table
ODBC LOAD load table
NATIVE ODBC LOAD load table
SSIS LOAD load table
DATABASE LINK LOAD load table
DATABASE LINK LOAD load table (without a db link)

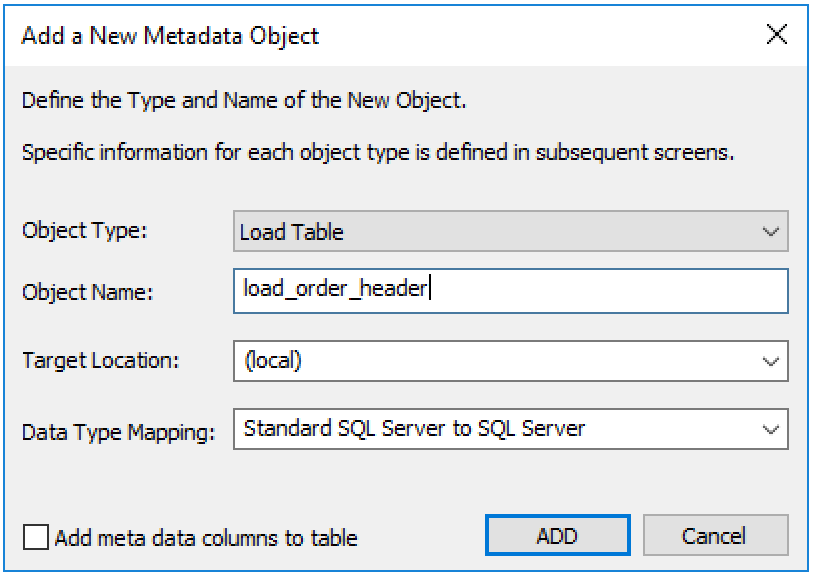
Load Drag and Drop

The simplest way to create a Load table is to use the drag and drop functionality of WhereScape RED.

Drag and drop can be used for all connection types and the process is the same in all cases.

1. Browse to the source system connection (**Browse > Source Tables**).
2. Create a drop target by double-clicking the Load Table object group in the left pane. The middle pane should have a column heading of **Load Table Name** for the leftmost column.
3. Select a table or file in the right pane and drag it to the middle pane. Drop the table or file anywhere in the middle pane.
4. Answer the prompts to create the Load table.

5. When using targets, you can also change the predefined Target Location settings in **Tools > Options > Storage**. At the time of the table object's drag and drop, the **Add a New Metadata Object** window enables you to edit the **Target Location** and **Data Type Mapping**, so the table object's location can be changed on a table by table basis.
   The **Data Type Mapping** field is automatically set, based on the source and the **Target Location** selected, but can be changed if needed.



Add a New Metadata Object

Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: Load Table

Object Name: load_order_header

Target Location: (local)

Data Type Mapping: Standard SQL Server to SQL Server

□ Add meta data columns to table

Note

The option **Add meta data columns to table** is selected when creating Load tables that are used in creating Data Vault objects. If this option is selected, two DSS columns (**dss_record_source** and **dss_load_date**) are included in the meta data for the table and are populated by transformations. These two DSS columns could equally be applied to other Load tables not used in a Data Vault system but are particularly important to comply with the Data Vault standards. Refer to Data Vaults for details.

WhereScape RED supports loading tables of up to 2048 columns. However, this maximum number of column loading restriction can in fact be lower than the target database or tools permit.

The target database will provide users the appropriate warning, if the maximum number of columns loaded is breached at runtime.

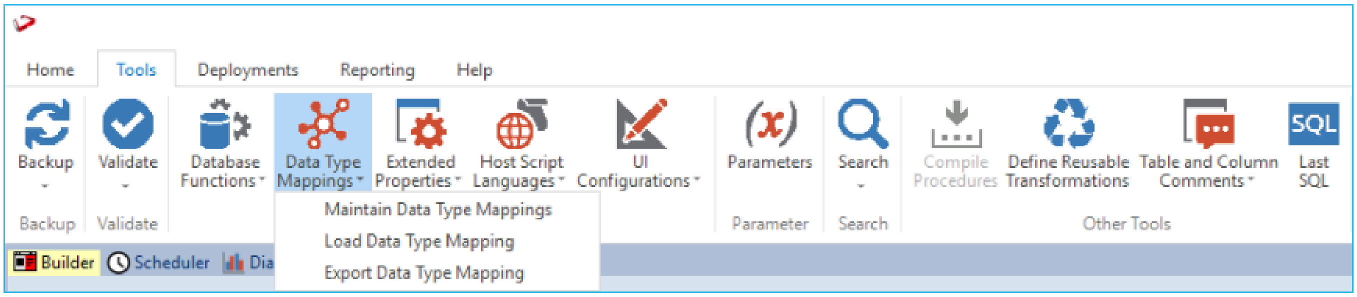
Notes

When creating a Load table in WhereScape RED by dragging and dropping a source table, RED reads the structure of the table in the source system and attempts to construct an equivalent Load table in the data warehouse. There are occasions when the Load table creation will fail due to incompatible data types in the target data warehouse. The remedy is to change the data types of the particular attributes which are causing the load failure. Once corrected, the Load table is created.
It is important that the table load is tested to ensure that data can be INSERTED into the Load table from the source table. If the load fails then the data may need to be explicitly converted to the destination data type, using a column transformation that is executed during the load (refer to Load Table Column Transformations for details).
Incompatible data types that cause Load table creation errors are typically caused by:

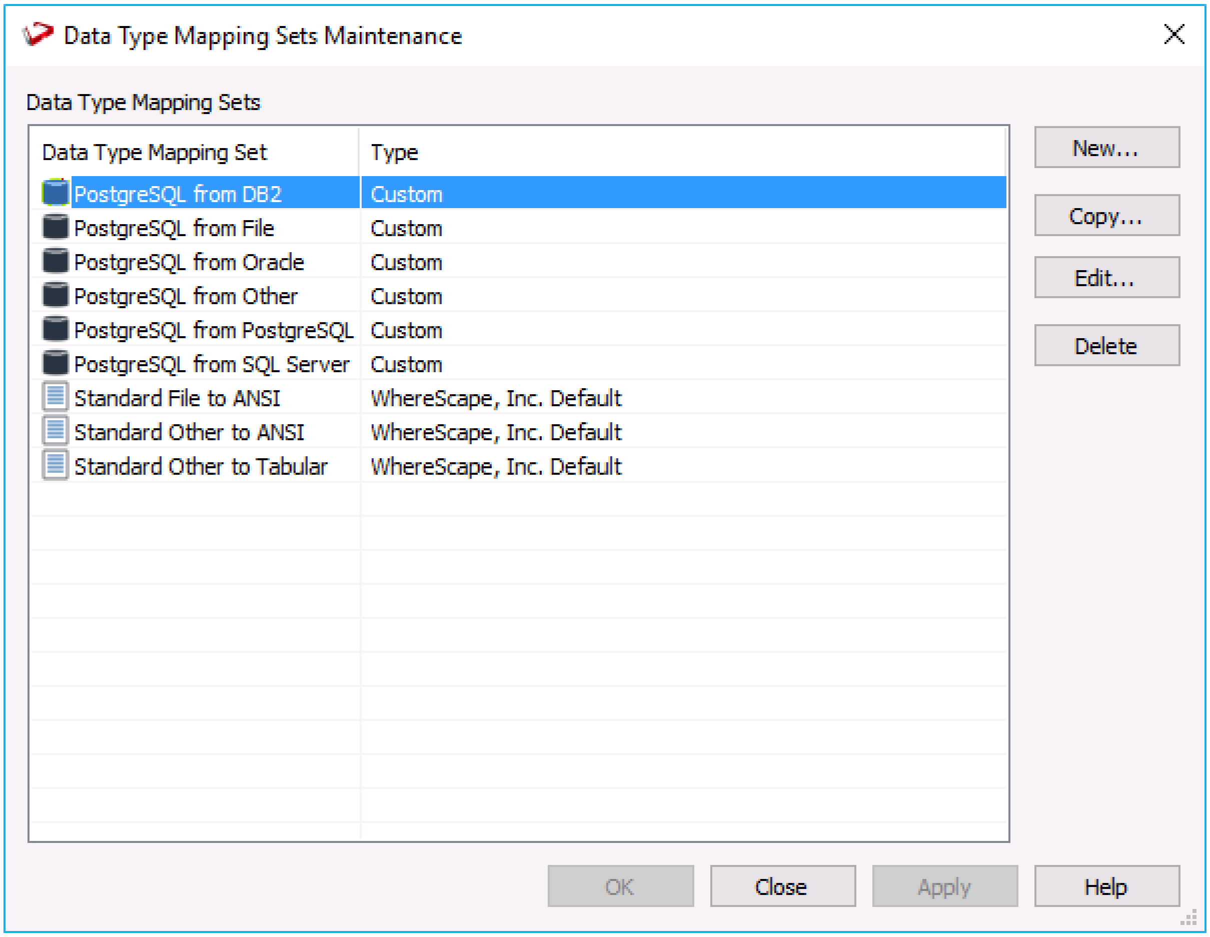
1. User defined data types in the source database.
2. Incorrect data type mapping during Load table definition in WhereScape RED.
3. Data types that cannot be inserted into (e.g. identity columns in SQL Server).

Data Type Mappings

In **Tools > Data Type Mappings**, the first menu option enables you to maintain the data type mappings.



The **Maintain Data Type Mappings** window is as follows:


Data Type Mapping Sets Maintenance

Data Type Mapping Sets

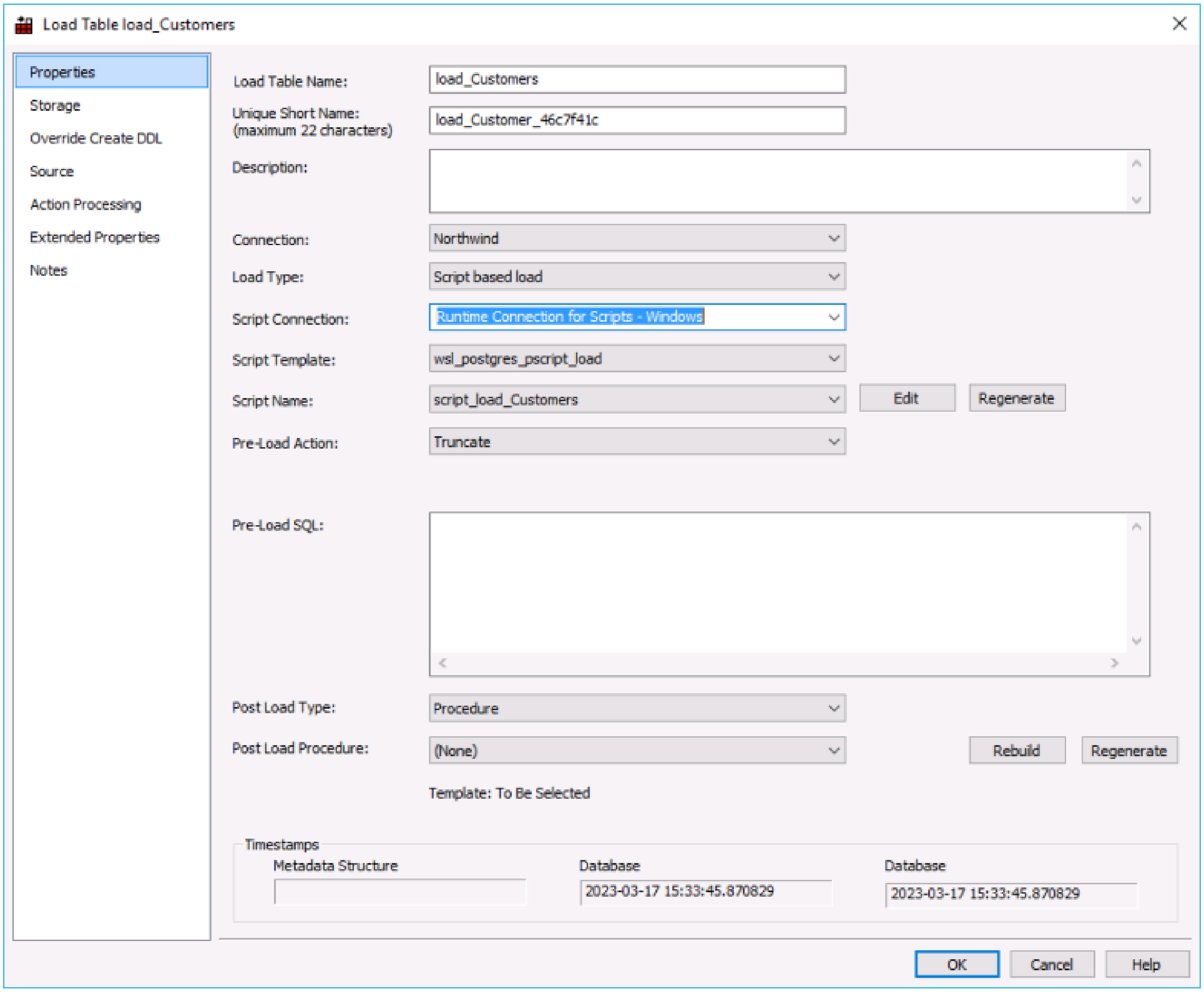
Data Type Mapping Set	Type
PostgreSQL from DB2	Custom
PostgreSQL from File	Custom
PostgreSQL from Oracle	Custom
PostgreSQL from Other	Custom
PostgreSQL from PostgreSQL	Custom
PostgreSQL from SQL Server	Custom
Standard File to ANSI	WhereScape, Inc. Default
Standard Other to ANSI	WhereScape, Inc. Default
Standard Other to Tabular	WhereScape, Inc. Default

Refer to **Data Type Mappings** for details about how to manage data mappings in RED.

Database Link Load

Database Link Load - Properties

The fields in the Database Link Load Properties screen are described below:



Load Table load_Customers

Load Table Name: load_Customers

Unique Short Name: (maximum 22 characters) load_Customer_46c7f41c

Description:

Connection: Northwind

Load Type: Script based load

Script Connection: Runtime Connection for Scripts - Windows

Script Template: wsl_postgres_pscript_load

Script Name: script_load_Customers

Pre-Load Action: Truncate

Pre-Load SQL:

Post Load Type: Procedure

Post Load Procedure: (None)

Template: To Be Selected

Timestamps

Metadata Structure:

Database: 2023-03-17 15:33:45.870829

Database: 2023-03-17 15:33:45.870829

Fields	Description
Load Table Name	The table name is limited by most relational databases to a maximum of 30 characters and must be unique. Table name defaults can be set up in **Tools > Options** to define a prefix or a post fix that can be added in order to identify clearly that this is a load table. Example: **load**_customer. By default, WhereScape RED uses the prefix **load_** for Load tables.
Unique Short Name	It must be unique. The short name is used in the naming of indexes, keys, procedures and scripts.
Description	Enter a description of the table. This description appears in the documentation that can be generated, once the data warehouse is built.
Connection	Enter the connection being used to get the data. The connections for Load tables can be changed in bulk, refer to **Changing load Connection and Schema** for details.
Load Type	The load type is typically defined by the connection, and should not normally be changed. This drop-down shows all valid load types for the connection.
Script Connection	Select the connection for the script.
Script Template	The script template used for a script based load.
Script Name	This field is only active for script based loads. **Tip**

Fields	Description
	When doing a Script based load, use the **Rebuild** button after selecting the relevant script to be rebuilt from the **Script Name** drop-down list.
Pre-load Action	Select an action to be performed on the table before the load occurs. Options are: • Truncate • Execute Pre-Load SQL • Both Truncate and Execute Pre-Load SQL • No action
Pre-load SQL	If a Pre-load Action of **Execute pre-load Sql** was selected, then the SQL statement to execute before the load takes place should be entered in this box.
Table Properties clauses (e.g. Partition)	These are clauses that are added to the end of the table create statement. Typically used for putting partition information on the table.
Post Load Type	**Note:** This field is only available for custom database targets. Enables you to specify the type of post load to execute for the table. Options are **Procedure** or **Script**.
Post Load Procedure	A procedure that is executed immediately following the load. If you execute an externally loaded table, no load occurs, but a post load procedure can still be executed. Post load procedures can either be manually generated from a RED provided procedure outline or generated using a RED template—refer to **Rebuilding Update Procedures** for details.
Post Load Script	**Note:** This field is only available for custom database targets and is only displayed when the **Script** option is selected from the **Post Load Type** field above. A script that is executed immediately following the load. If you execute an externally loaded table, no load occurs but a post load script can still be executed. Post load scripts can be generated using a RED template—refer to **Rebuilding Update Procedures** for details.

Note:

Three fields at the bottom of the Load Table Properties screen display date information:

1. Date table structure last updated.
2. Date created in database.
3. Date last updated in database.

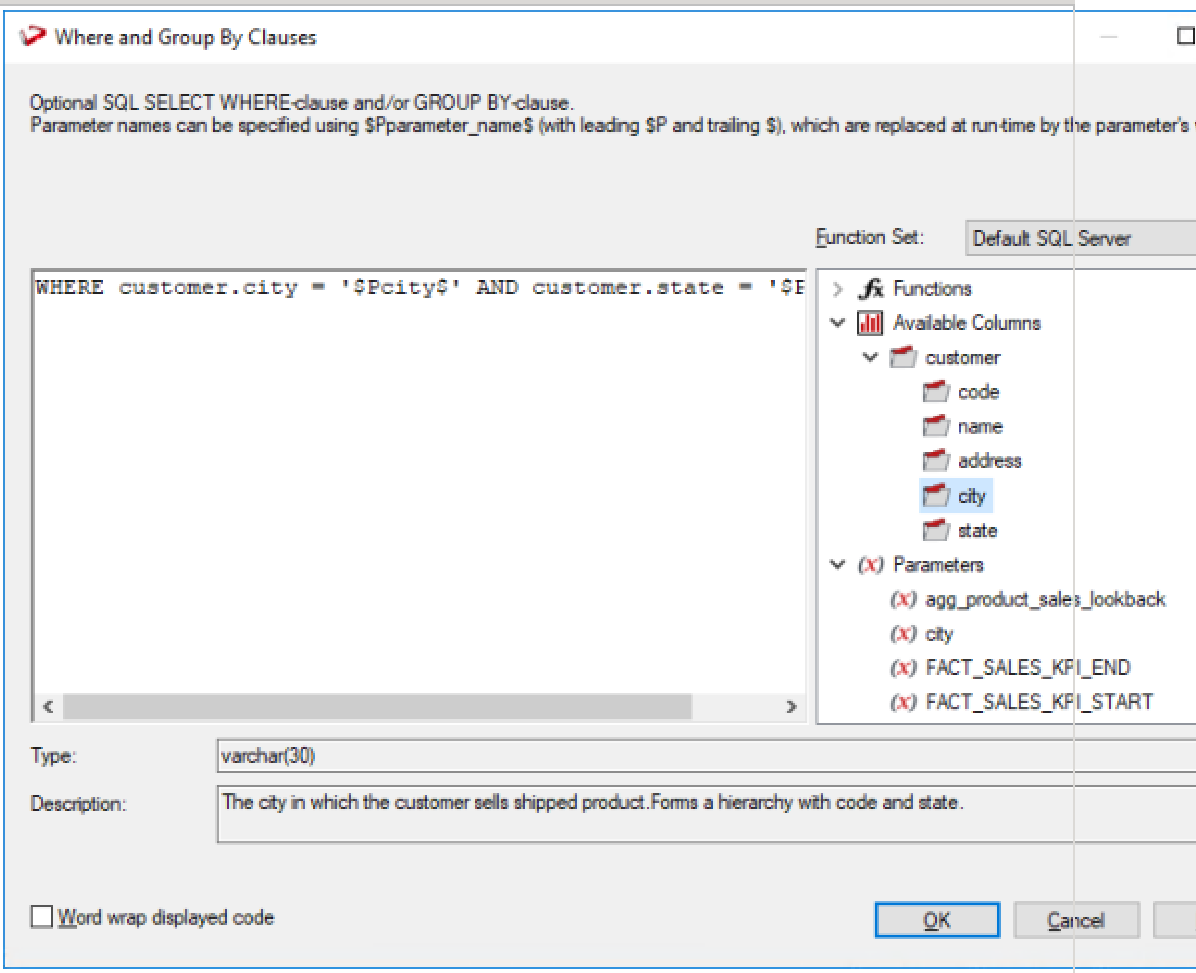
Database Link Load - Source Mapping

The fields in the **Database Link Load Source Screen** are described below:

Fields	Description
Load Type	Method of loading data into the table. The available options are dependent on the **Source Connection**. Defaults to the **Default Load Type** of the **Source Connection**. Can be specified via the table **Properties** window.
Source Connection	The connection that identifies the source database. Can be specified via the table **Properties** window.

General

Fields	Description
Select Distinct Values	Include the DISTINCT keyword in the SQL SELECT statement.
Source Schema	Schema within the source database where the source table resides.
Derive Source Tables(s) and Source Columns	Derive the Source Table(s) and Source Column(s) properties (of this screen) from the source details of this table's columns. **Note** The existing property values will be overwritten.
Source Table(s)	Name of the table(s) where the data is sourced.
Override Source Column/Transformations	Ignore the source and transformation details of this table's columns and instead use the override details specified below. Refer to the section on **Load Table Transformations** below for details.
Exit Status when Missing Source Columns	Load exit status when source columns are missing. Only enabled when **Allow Missing Source Columns** is set.
Where and Group By Clauses	Optional SQL SELECT WHERE-clause and/or GROUP BY-clause. Parameter names can be specified using $Pparameter_name$ (with leading $P and trailing $), which are replaced at run-time by the parameter's value. **Tip** This is where you can build a statement to handle change data. Parameter values can be in-line replaced and included in the 'Where' clause. Prefix the parameter name with a '$P' and add a trailing '$'. For example, if we have a parameter called SALES_LOCATION_CODE we could construct a statement, such as WHERE location_code = '$PSALES_LOCATION_CODE$' AND region_code = 'NY'. When this statement is executed, the value of the parameter will replace the parameter name. For example if the parameter was set to 'New York' then the statement would execute as: WHERE location_code = 'New York' AND region_code = 'NY'. Clicking the SQL button opens the **Where and Group By Clauses** editor which provides control settings to assist you in constructing the SQL statement.

Fields	Description
	 Where and Group By Clauses Optional SQL SELECT WHERE-clause and/or GROUP BY-clause. Parameter names can be specified using $Pparameter_name$ (with leading $P and trailing $), which are replaced at run-time by the parameter's Function Set: Default SQL Server `WHERE customer.city = '$Pcity$' AND customer.state = '$P` Type: varchar(30) Description: The city in which the customer sells shipped product.Forms a hierarchy with code and state. ☐ Word wrap displayed code Functions Available Columns customer code name address city state Parameters agg_product_sales_lookback city FACT_SALES_KPI_END FACT_SALES_KPI_START OK Cancel
Override Load SQL	Optional SQL statement to load data into the table, which overrides all other properties. The specified SQL will be executed instead of generated SQL. For a linked database specify a complete INSERT statement. For an ODBC source, specify only the SELECT statement.
SQL Insert Hint	**SQL Server Only** - Optional hint to include in the SQL insert statement.

ODBC Based Load

An ODBC based load provides an extensive option for acquiring data from many sources. It will be slower than most other forms of loads but may still be a viable option depending on the volume of data being loaded.

An ODBC based 'interactive load' when using the WhereScape RED tool will use the established ODBC connection to pull the data back to the local PC and then push the data to the data warehouse database via a sql ODBC statement.

A scheduler load will perform in the same way except that the data is loaded into the server that is running the scheduler and then pushed to the data warehouse database.

This is normally achieved via a 'During' load transformation such as `TO_CHAR(source_date_column,'YYYY-MM-DD HH24:MI:SS')`

The obvious disadvantage in an ODBC based load is the two network transactions required and the overhead placed on the Scheduler server. The UNIX scheduler does not support ODBC based loads, therefore a Windows scheduler must be available to handle ODBC based loads.

Note

A Windows scheduler must be available to handle ODBC based loads.

The **Properties** screens for an ODBC based load are the same as those of a database link load. Refer to the previous sections for details.

Native ODBC Based Load

A Native ODBC based load is similar to an ODBC load, except it provides a faster method to load data from non-linked databases.

A Native ODBC based **interactive load** when using the WhereScape RED tool will use the established ODBC connection to pull the data back to a delimited file on the local PC and then push the data to the data warehouse database via a Bulk Insert in SQL Server.

A scheduler load will perform in the same way, except that the data is loaded into the server that is running the scheduler and then pushed to the data warehouse database.

All loaders require dates in the default format of the target data warehouse database. Both of these are normally achieved via a **During** load transformation, using the correct casting function for the source database.

The **Properties** and **Storage** screens for a Native ODBC based load are the same as those of a database link load. Refer to the previous sections for details. Details of the source mapping screen follow.

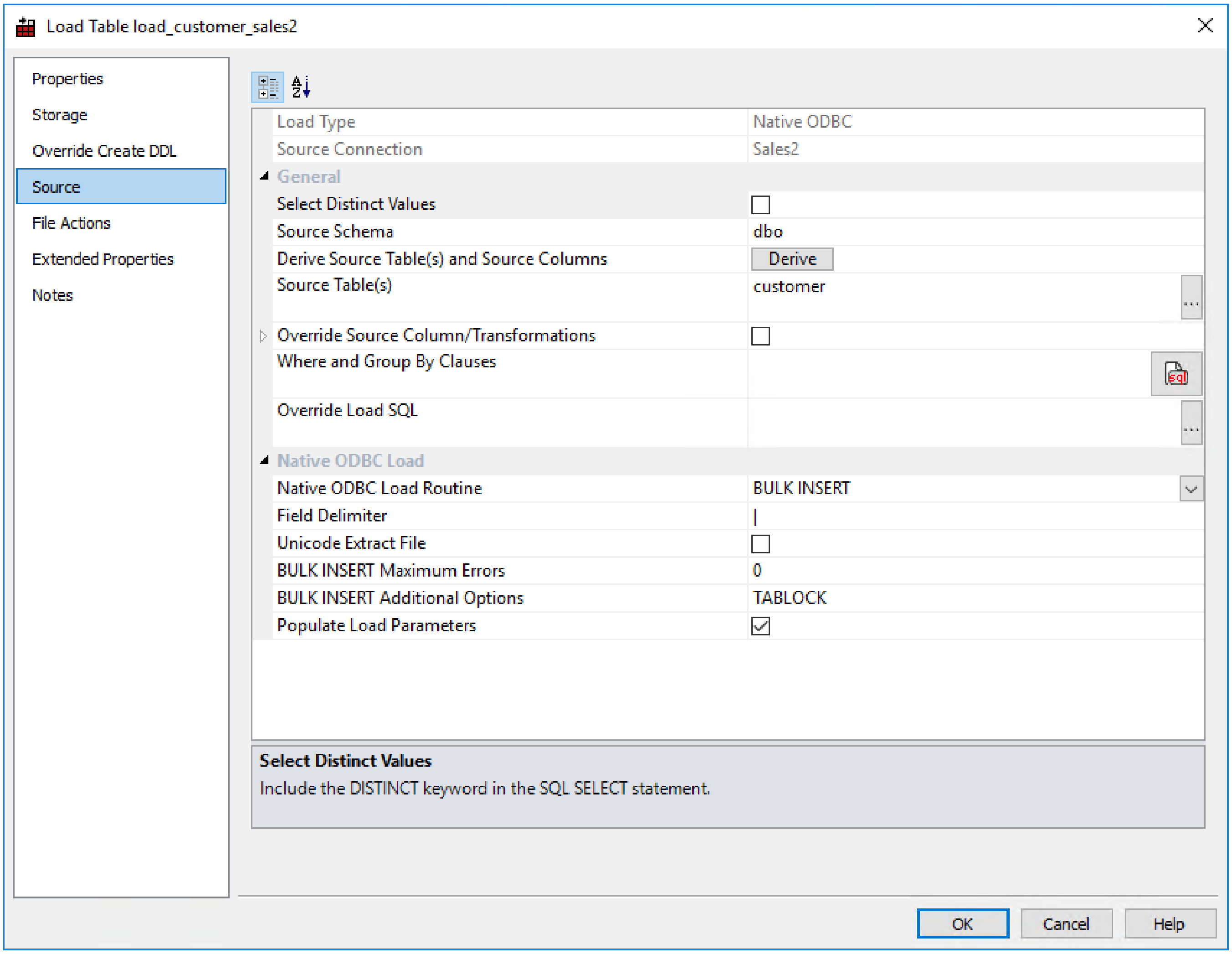
Notes:

- If you are doing Native Loads using UNIX and LINUX, see section Native Loads using UNIX and LINUX for more details about this type of load.

Native ODBC Based Load - Source Screen

The fields on the **Source** tab for Native ODBC loads are described below.

SQL Server Source Screen:



Load Table load_customer_sales2

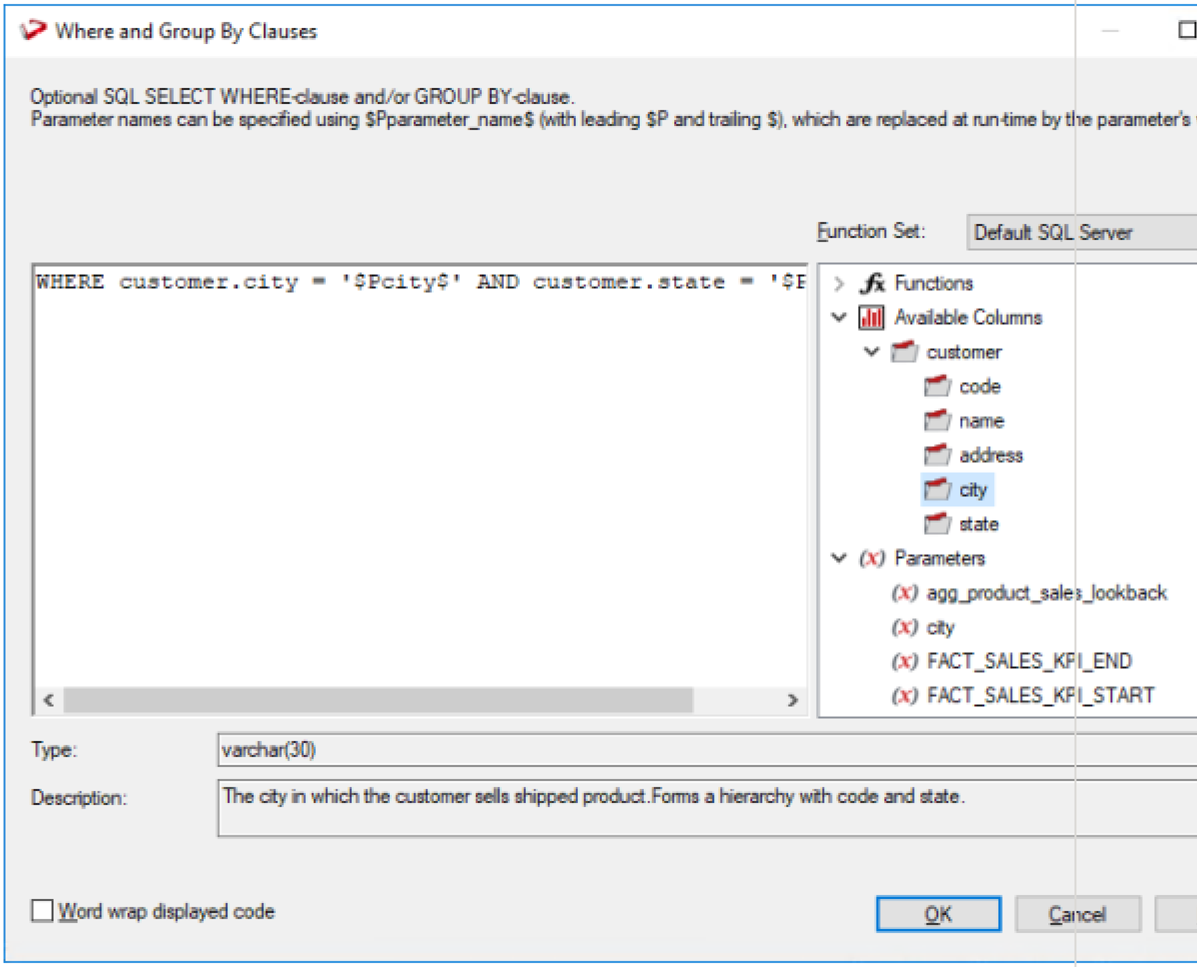
Load Type	Native ODBC
Source Connection	Sales2
General	
Select Distinct Values	□
Source Schema	dbo
Derive Source Table(s) and Source Columns	Derive
Source Table(s)	customer
Override Source Column/Transformations	□
Where and Group By Clauses	
Override Load SQL	
Native ODBC Load	
Native ODBC Load Routine	BULK INSERT
Field Delimiter	|
Unicode Extract File	□
BULK INSERT Maximum Errors	0
BULK INSERT Additional Options	TABLOCK
Populate Load Parameters	☑

Select Distinct Values
Include the DISTINCT keyword in the SQL SELECT statement.

Fields	Description
Load Type	Method of loading data into the table. The available options are dependent on the **Source Connection**. Defaults to the 'Default Load Type' of the **Source Connection**. Can be specified via the Properties screen.
Source Connection	The connection that identifies the source database. Can be specified via the Properties screen.

General

Fields	Description
Select Distinct Values	Include the DISTINCT keyword in the SQL SELECT statement.
Source Schema	Schema within the source database where the source table resides.
Derive Source Tables(s) and Source Columns	Derive the **Source Table(s)** and **Source Column(s)** properties (of this screen) from the source details of this table's columns. **Note** The existing property values will be overwritten.
Source Table(s)	Name of the table or tables that the data is sourced from.
Override Source Column/Transformat	Ignore the source and transformation details of this table's columns and instead use the override details specified below.

Fields	Description
ions	
Where and Group By Clauses	Optional SQL SELECT WHERE-clause and/or GROUP BY-clause. Parameter names can be specified using $Pparameter_name$ (with leading $P and trailing $), which are replaced at run-time by the parameter's value. **Tip** This is where you can build a statement to handle change data. Parameter values can be in-line replaced and included in the 'Where' clause. Prefix the parameter name with a '$P' and add a trailing '$'. For example, if we have a parameter called SALES_LOCATION_CODE we could construct a statement, such as WHERE location_code = '$PSALES_LOCATION_CODE$' AND region_code = 'NY'. When this statement is executed, the value of the parameter will replace the parameter name. For example if the parameter was set to 'New York' then the statement would execute as: WHERE location_code = 'New York' AND region_code = 'NY'. Clicking the SQL button opens the **Where and Group By Clauses** editor which provides control settings to assist you in constructing the SQL statement.  Where and Group By Clauses Optional SQL SELECT WHERE-clause and/or GROUP BY-clause. Parameter names can be specified using $Pparameter_name$ (with leading $P and trailing $), which are replaced at run-time by the parameter's Function Set: Default SQL Server WHERE customer.city = '$Pcity$' AND customer.state = '$P Type: varchar(30) Description: The city in which the customer sells shipped product.Forms a hierarchy with code and state. □ Word wrap displayed code Functions Available Columns customer code name address city state Parameters agg_product_sales_lookback city FACT_SALES_KPI_END FACT_SALES_KPI_START OK Cancel
Override Load SQL	Optional SQL statement to load data into the table, which overrides all other properties. The specified SQL will be executed instead of generated SQL. For a linked database specify a complete INSERT statement. For an ODBC source specify only the SELECT statement.

Native ODBC Load

Fields	Description
Native ODBC Load Routine	File Loader utility/mechanism to use to load the generated extract file.
Field Delimiter	Character that separates the fields within each record of the generated extract file. The default value is a | character (pipe). This should be changed if pipes are possible in the source data.
Unicode Extract File	Data will be loaded via a Unicode format file. The default is unchecked and this field is not available FOR DB2.
BULK INSERT Maximum Errors	**SQL Server Only** - Maximum number of errors accepted before a load failure is triggered. The default for this option is 0.
BULK INSERT Additional Options	**SQL Server Only** - Optional comma-delimited list of options to control the behavior of the SQL Server BULK INSERT command. For example the following options will respectively lock the table; set the batch size to 100,000 rows; and fail the load when the first error occurs: TABLOCK, BATCHSIZE=100000.
Populate Load Parameters	Populate any load-related WhereScape RED parameters, which may be used for validation purposes.

File Actions

A file action defines a copy, move or delete step that happens to **Native ODBC** temporary files after a load is run.

There are six different file action programs:

File Action	Description
copy all files	Copies all temporary files for the Load table (including the temporary data file) to another specified directory.
copy data files	Copies the **Native ODBC** Load table's temporary data file to another specified directory. Also enables a new file name to be specified.
delete all files	Deletes all temporary files for the Load table (including the temporary data file).
delete data files	Deletes the **Native ODBC** Load table's temporary data file.
move all files	Moves all temporary files for the Load table (including the temporary data file) to another specified directory.
move data files	Moves the **Native ODBC** Load table's temporary data file to another specified directory. Also enables a new file name to be specified.

Note

A Load table may have more than one file action. File actions are processed in order, based on the action number.

Creating a File Action

To create a file action:

1. Click the **File Actions** tab of the Load table **Properties** window.
2. Click **Add new action** and then select **After load** from the **Action Type** drop-down list.
3. Enter the **Action Description**.
4. Select the **Action Program** from the drop-down list (see above for the options).
   - For action programs of copy... and move..., enter the target directory name.
   - For copy data files and move data files, optionally enter the file's new name.
5. Click **Save (Update) Action**.
6. Repeat if necessary for additional file actions.

7. Click **OK**.

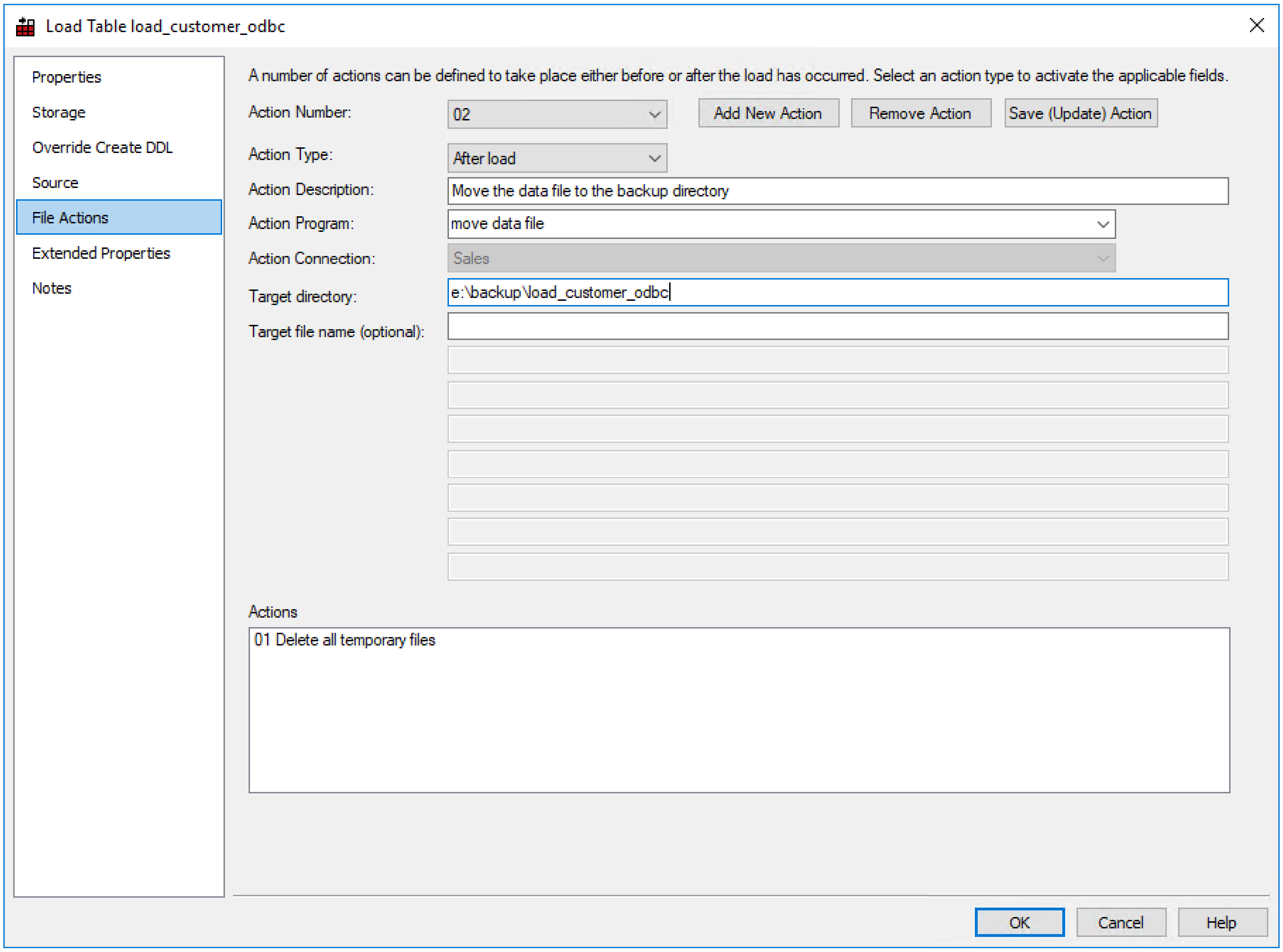
File Action Example

The following example moves the Native ODBC Load data file to another directory then deletes all temporary files.

Note

The data file is not deleted as it has been moved prior to the delete action.

Sample **move data file** file action:



Load Table load_customer_odbc

A number of actions can be defined to take place either before or after the load has occurred. Select an action type to activate the applicable fields.

Action Number: 02

Action Type: After load

Action Description: Move the data file to the backup directory

Action Program: move data file

Action Connection: Sales

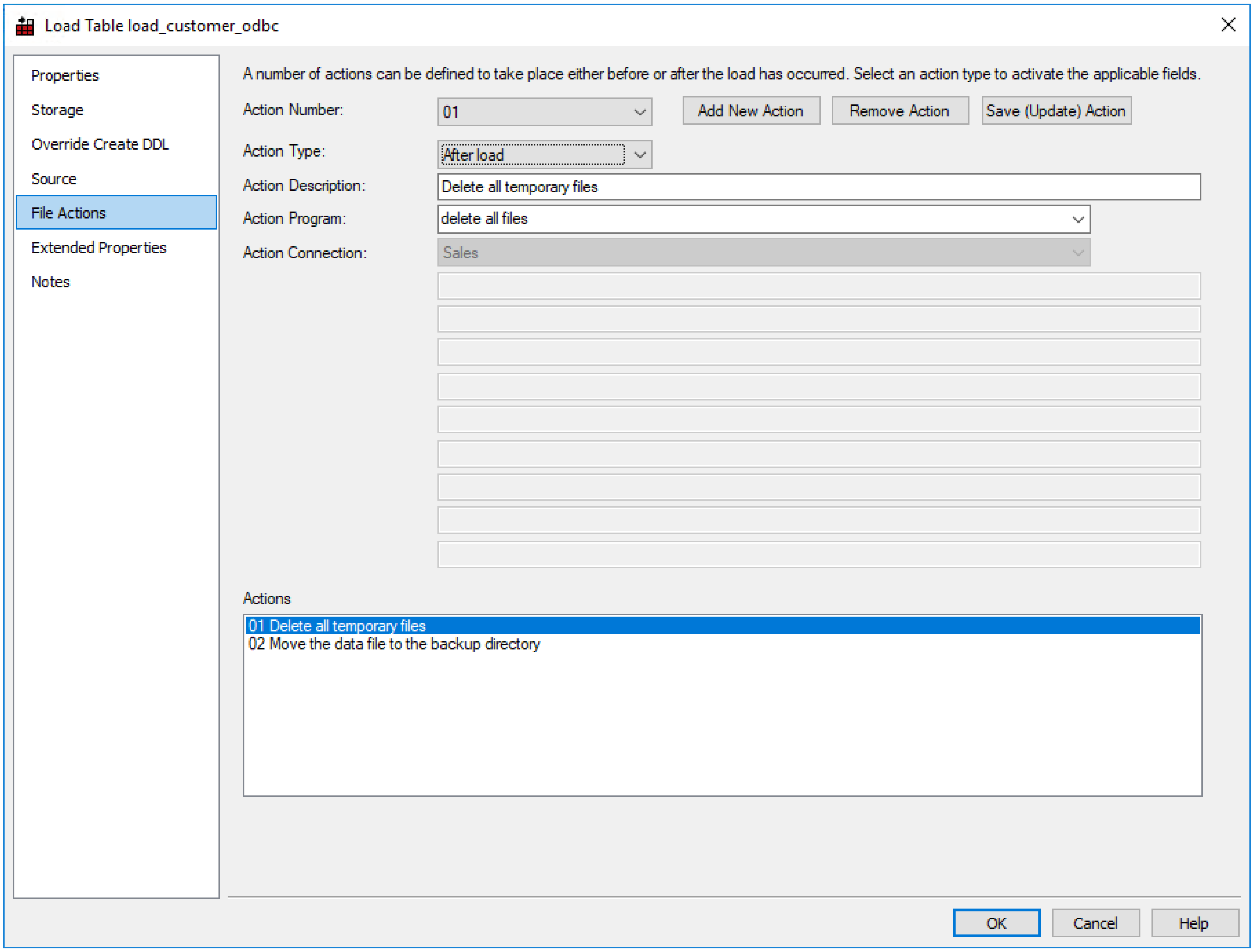
Target directory: e:\backup\load_customer_odbc

Target file name (optional):

Actions

01 Delete all temporary files

Sample **delete all files** file action:



A number of actions can be defined to take place either before or after the load has occurred. Select an action type to activate the applicable fields.

Action Number: 01

Action Type: After load

Action Description: Delete all temporary files

Action Program: delete all files

Action Connection: Sales

Actions

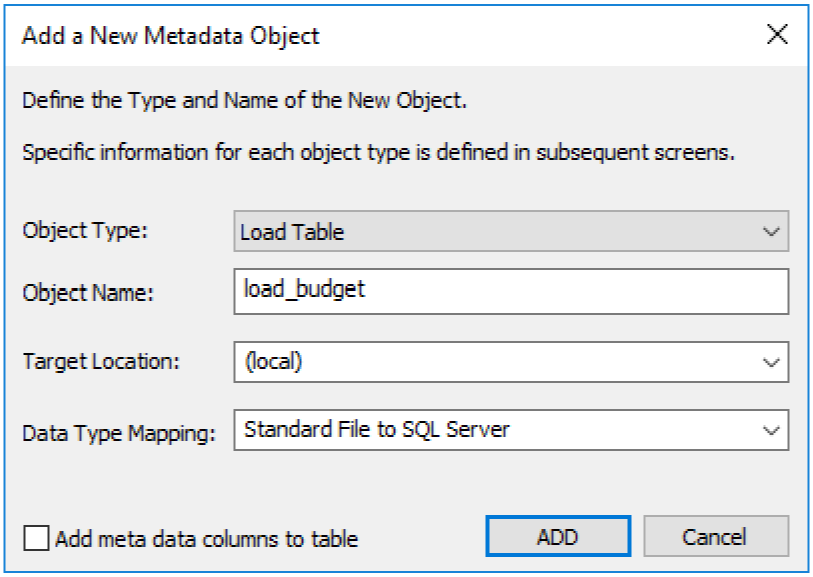
01 Delete all temporary files
02 Move the data file to the backup directory

Flat File Loads

As with all other load types it is easier to use the drag and drop functionality to create load tables.

The **drag and drop** process for flat files is as follows:

1. **Browse** to the directory and file via the appropriate **Connections**.
2. Double-click on the **Load Table** object in the left pane to create a drop target.
3. **Drag** the file from the right pane and **drop** it into the middle pane.
4. The dialog below appears. Rename the file if necessary and then click the **ADD** button.



Add a New Metadata Object

Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: Load Table

Object Name: load_budget

Target Location: (local)

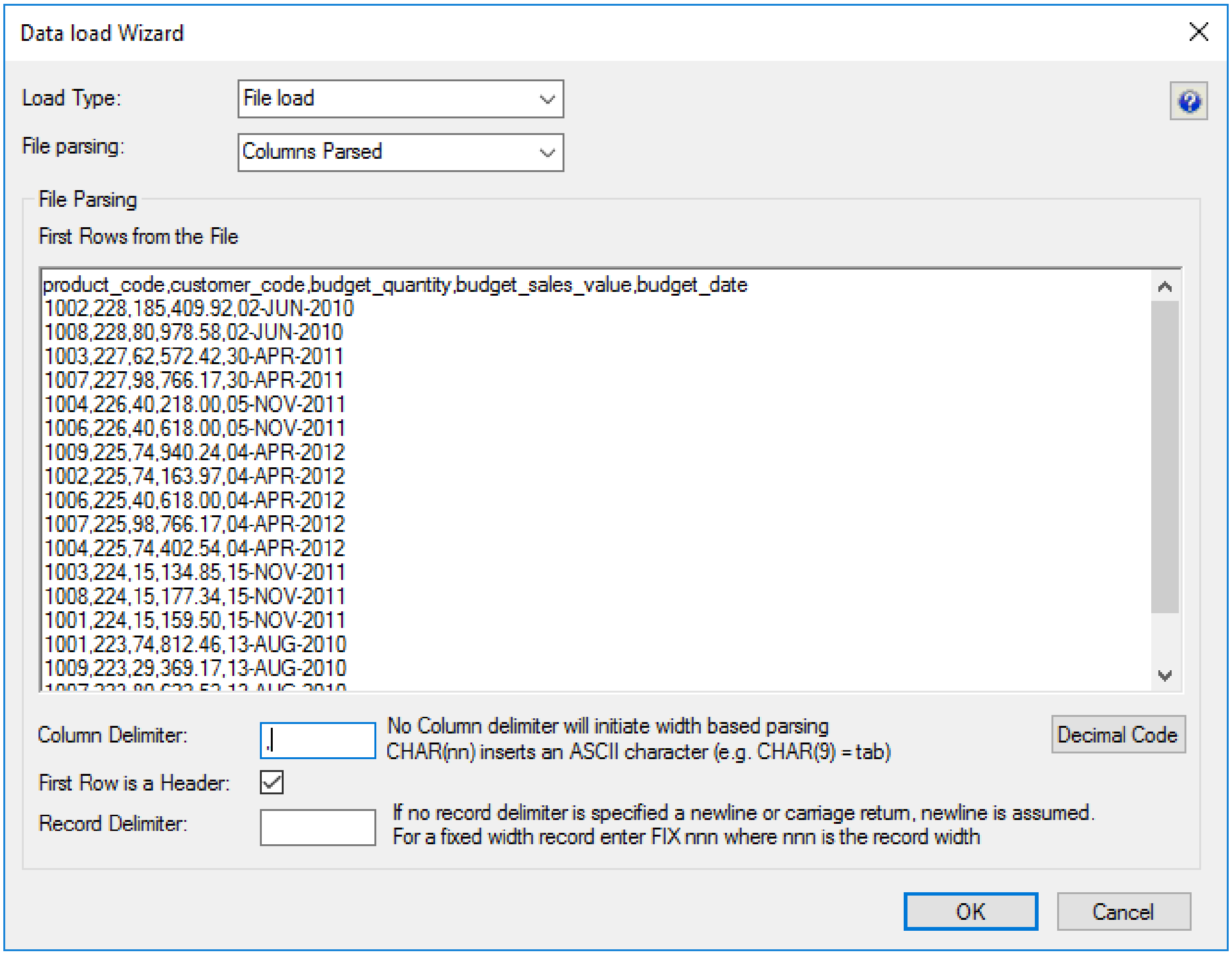
Data Type Mapping: Standard File to SQL Server

□ Add meta data columns to table

Note:

The option **Add meta data columns to table** is used for creating Load tables that are used in creating Data Vault objects. If this option is selected, two DSS columns (**dss_record_source** and d**ss_load_date**) are included in the meta data for the table and are populated by transformations. These two DSS columns could equally be applied to other Load tables not used in a Data Vault system but are particularly important to comply with the Data Vault standards. Refer to **Data Vaults** for details.

5. The following dialog displays for file loads.



Data load Wizard

Load Type: File load

File parsing: Columns Parsed

File Parsing

First Rows from the File

```
product_code,customer_code,budget_quantity,budget_sales_value,budget_date
1002,228,185,409.92,02-JUN-2010
1008,228,80,978.58,02-JUN-2010
1003,227,62,572.42,30-APR-2011
1007,227,98,766.17,30-APR-2011
1004,226,40,218.00,05-NOV-2011
1006,226,40,618.00,05-NOV-2011
1009,225,74,940.24,04-APR-2012
1002,225,74,163.97,04-APR-2012
1006,225,40,618.00,04-APR-2012
1007,225,98,766.17,04-APR-2012
1004,225,74,402.54,04-APR-2012
1003,224,15,134.85,15-NOV-2011
1008,224,15,177.34,15-NOV-2011
1001,224,15,159.50,15-NOV-2011
1001,223,74,812.46,13-AUG-2010
1009,223,29,369.17,13-AUG-2010
```

Column Delimiter: ,

No Column delimiter will initiate width based parsing
CHAR(nn) inserts an ASCII character (e.g. CHAR(9) = tab)

First Row is a Header: ☑

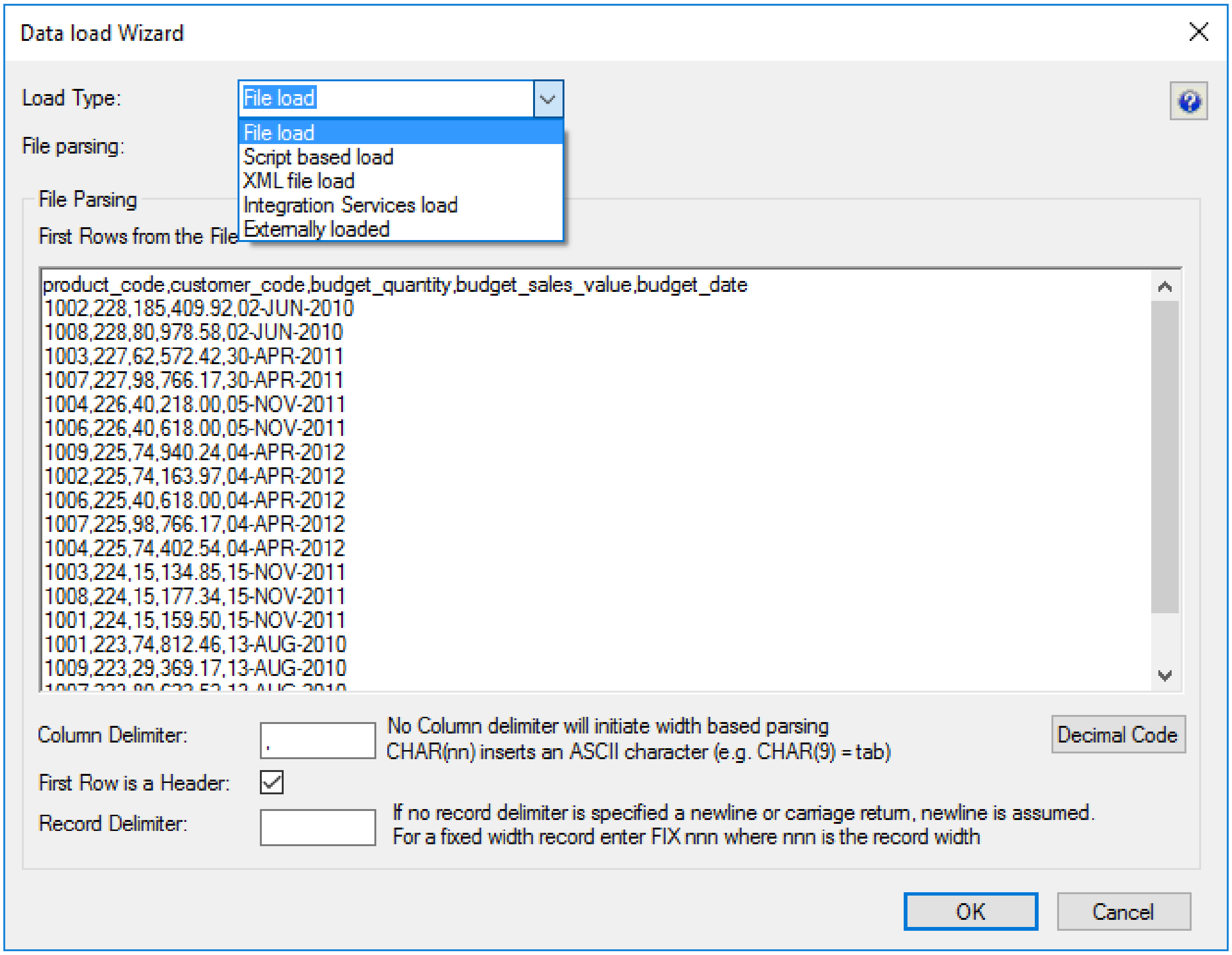
Record Delimiter:

If no record delimiter is specified a newline or carriage return, newline is assumed.
For a fixed width record enter FIX nnn where nnn is the record width

Notes:

- **First Row is a Header**- To make changes in relational database tables after a table has been defined, users can edit the **First Row is a Header** option in the **Source** tab of the relevant table.
- **Hive** - For File loads into Hive tables, the **First Row is a Header** option is a table option. Please ensure this option is selected in the file load wizard if you want to load files with header rows.
  To make any changes after Hive tables have been defined, the **First Row is a Header** option can be found in the **Storage** tab of the relevant Hive table.

6. The load type selected in the **New Table Default Load Type** field in the connection Properties screen is the pre-selected option in the **Load type** drop-down list.
   - To change the desired load type and file parsing, use the **Load type** and **File parsing** drop-down list options.



Load Type: File load

File parsing:

First Rows from the File

```
product_code,customer_code,budget_quantity,budget_sales_value,budget_date
1002,228,185,409.92,02-JUN-2010
1008,228,80,978.58,02-JUN-2010
1003,227,62,572.42,30-APR-2011
1007,227,98,766.17,30-APR-2011
1004,226,40,218.00,05-NOV-2011
1006,226,40,618.00,05-NOV-2011
1009,225,74,940.24,04-APR-2012
1002,225,74,163.97,04-APR-2012
1006,225,40,618.00,04-APR-2012
1007,225,98,766.17,04-APR-2012
1004,225,74,402.54,04-APR-2012
1003,224,15,134.85,15-NOV-2011
1008,224,15,177.34,15-NOV-2011
1001,224,15,159.50,15-NOV-2011
1001,223,74,812.46,13-AUG-2010
1009,223,29,369.17,13-AUG-2010
```

Column Delimiter: ,

No Column delimiter will initiate width based parsing
CHAR(nn) inserts an ASCII character (e.g. CHAR(9) = tab)

First Row is a Header: ☑

Record Delimiter:

If no record delimiter is specified a newline or carriage return, newline is assumed.
For a fixed width record enter FIX nnn where nnn is the record width

Load type options

- The **File based load** options results in a load where the bulk of the load management is handled by the scheduler.
- The **Script based load** option will make WhereScape RED generate a host script and the load table will be a **Script based load.** This host script is executed by the scheduler to perform the load. For further details about Script based loads, refer to **Script based load**.
- The **Externally loaded** option will not execute an actual load into the table but will process the actions specified in the Post Load procedure property. Any **After** transformations recorded against any of the columns in an Externally loaded table will also be processed.

Post load procedures can either be manually generated from a RED provided procedure outline or generated leveraging a RED template—refer to **Rebuilding Update Procedures** for details.

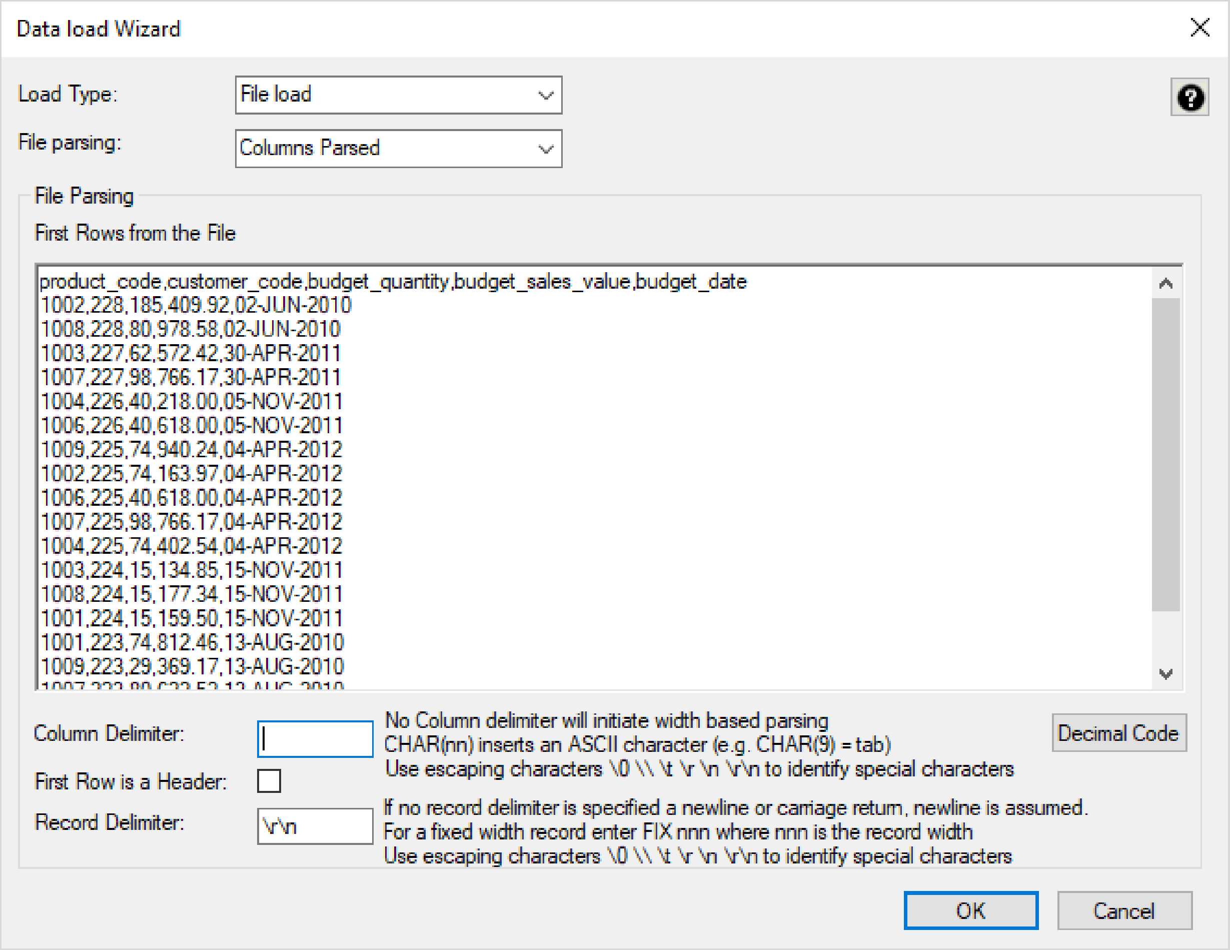
Note:

For custom database targets, the option to use Post Load Scripts is provided as an alternative to post load procedure. Post load scripts can be generated using a RED template—refer to **Rebuilding Update Procedures** for details

File parsing options

- **Single data column** - with this option, the majority of the work in terms of parsing the file must occur in a subsequent procedure within the data warehouse. The data is dumped into a single column. The task of coding a procedure to parse the data must then be undertaken.

- **Columns parsed** - with this option, WhereScape RED will attempt to parse the columns. You will be asked for details and for the column delimiter. You then step through the columns providing names and data types. WhereScape RED attempts to guess the data type, but it needs to be checked and the field length will probably need to be adjusted.



Data load Wizard

Load Type: File load

File parsing: Columns Parsed

File Parsing

First Rows from the File

```
product_code,customer_code,budget_quantity,budget_sales_value,budget_date
1002,228,185,409.92,02-JUN-2010
1008,228,80,978.58,02-JUN-2010
1003,227,62,572.42,30-APR-2011
1007,227,98,766.17,30-APR-2011
1004,226,40,218.00,05-NOV-2011
1006,226,40,618.00,05-NOV-2011
1009,225,74,940.24,04-APR-2012
1002,225,74,163.97,04-APR-2012
1006,225,40,618.00,04-APR-2012
1007,225,98,766.17,04-APR-2012
1004,225,74,402.54,04-APR-2012
1003,224,15,134.85,15-NOV-2011
1008,224,15,177.34,15-NOV-2011
1001,224,15,159.50,15-NOV-2011
1001,223,74,812.46,13-AUG-2010
1009,223,29,369.17,13-AUG-2010
```

Column Delimiter:

No Column delimiter will initiate width based parsing
CHAR(nn) inserts an ASCII character (e.g. CHAR(9) = tab)
Use escaping characters \0 \\ \t \r \n \r\n to identify special characters

First Row is a Header: □

Record Delimiter: \r\n

If no record delimiter is specified a newline or carriage return, newline is assumed.
For a fixed width record enter FIX nnn where nnn is the record width
Use escaping characters \0 \\ \t \r \n \r\n to identify special characters

Notes:

- The **Decimal Code** button will show the decimal value of each character in the lines retrieved from the source file. These decimal codes will be shown below each line and are green.

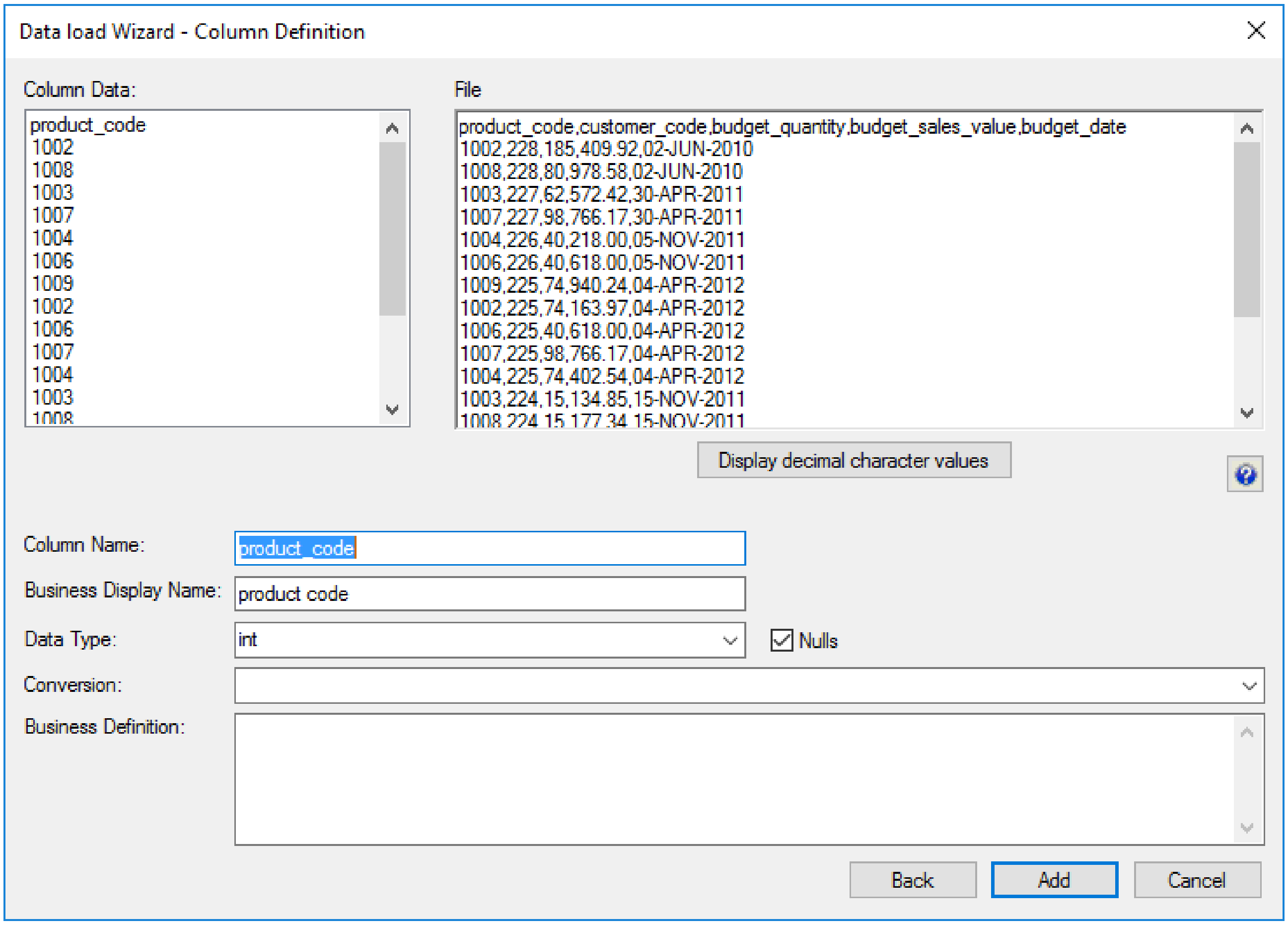
7. Once the selection on the screen above is completed, a screen displays which enable the breakdown of the source data into columns.
   If no delimiter is entered, then width based parsing is assumed and an addition width size is prompted for.

Notes:

- If no delimiter is entered, then **width based parsing** is assumed and an addition width size is prompted.

The following screen is an example of the file parsing technique.

Use the **Back** button to revert to the previous column if an incorrect width or delimiter is entered.


Data load Wizard - Column Definition
Display decimal character values
Back
Add
Cancel

Column Data:

```
product_code
1002
1008
1003
1007
1004
1006
1009
1002
1006
1007
1004
1003
1008
```

File

```
product_code,customer_code,budget_quantity,budget_sales_value,budget_date
1002,228,185,409.92,02-JUN-2010
1008,228,80,978.58,02-JUN-2010
1003,227,62,572.42,30-APR-2011
1007,227,98,766.17,30-APR-2011
1004,226,40,218.00,05-NOV-2011
1006,226,40,618.00,05-NOV-2011
1009,225,74,940.24,04-APR-2012
1002,225,74,163.97,04-APR-2012
1006,225,40,618.00,04-APR-2012
1007,225,98,766.17,04-APR-2012
1004,225,74,402.54,04-APR-2012
1003,224,15,134.85,15-NOV-2011
1008,224,15,177.34,15-NOV-2011
```

Column Name: product_code

Business Display Name: product code

Data Type: int ☑ Nulls

Conversion:

Business Definition:

Conversion

Understanding SQL*Loader

When using either type of loading method **SQL*Loader** is called to perform the actual load. There is therefore a requirement that the executable called **'sqlldr'** be in the path.

Under **UNIX** this means that the UNIX user must be able to execute 'sqlldr' from the command prompt under a normal log on.

Under **Windows** 'sqlldr' must execute when the command is issued from a DOS prompt.

In both the Windows and UNIX environments sqlldr returns a result code indicating if the load worked fully, partially or failed. WhereScape RED makes use of this returned code to report on the results of the load.

Flat File Load - Source Screen

The fields for the **Flat File Source Screen** are described below. See example source screens in **File and Script based load Source screens**.

Tip

If the file has been **dragged and dropped** into the load table (middle pane) then some of the fields on this tab are automatically populated.

Fields	Description
Load Type	Method of loading data into the table. The available options are dependent on the

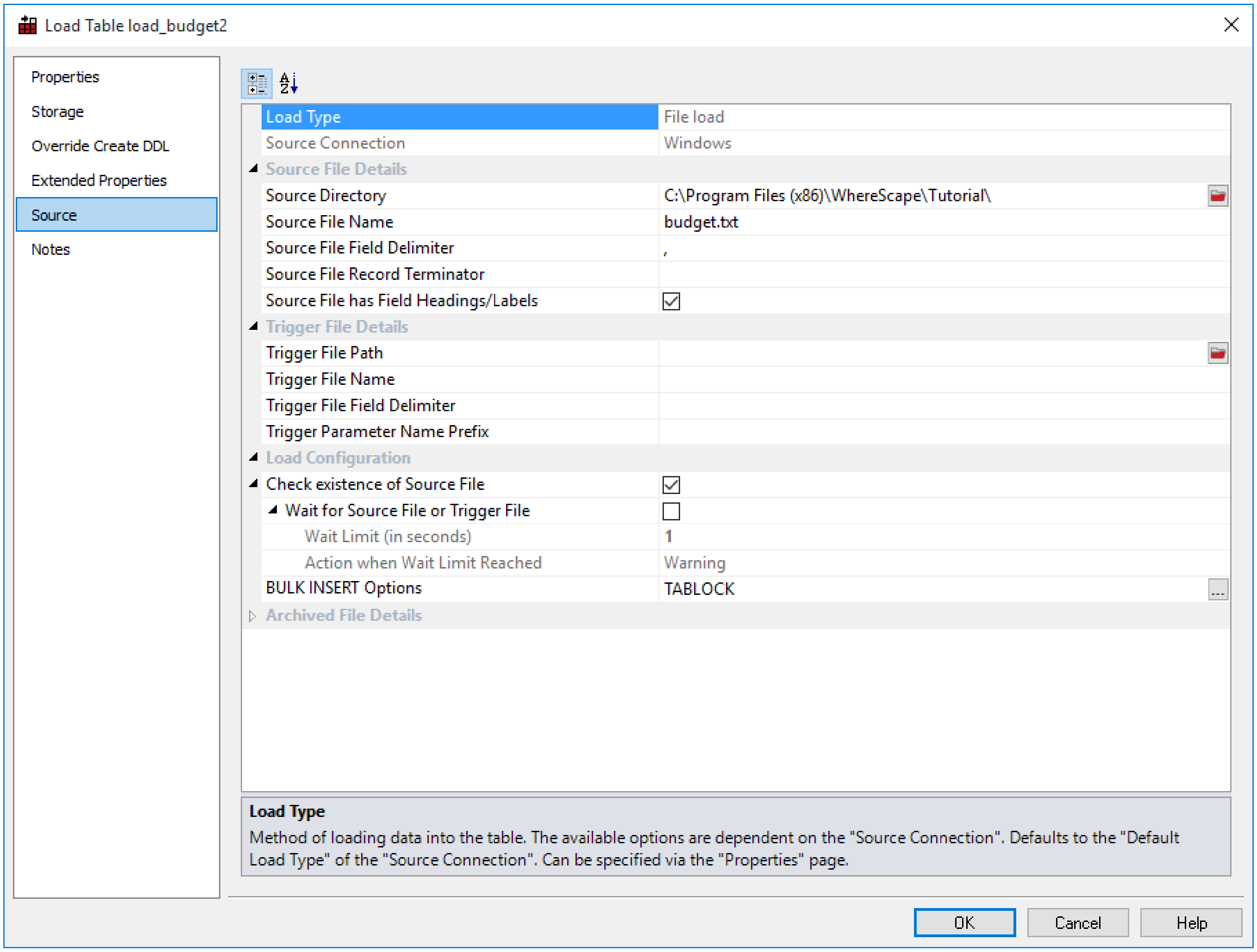
Fields	Description
	Source Connection. Defaults to the 'Default Load Type' of the **Source Connection**. Can be specified via the Properties screen.
Source Connection	Connection to the data source (database or file system). Can be specified via the **Properties** screen.
Load Script Template	Available for script loads only and only if there is a valid template available. Select the template to use when generating a load script, or select (None) to use RED's built-in load script generator. Only templates with the correct Type and Target DB will appear in this drop-down list. Refer to **Templates** for details.
Source File Details	Source File identification and definition information.
Source Directory	The full path (absolute path) of the folder/directory containing the Source File on the Windows or UNIX/Linux system.
Source File Name	The name of the source file containing the data to be loaded.
Source File Character Encoding	**PDW option only**- The character encoding of the input datafile for PDW file loads. Select between **ASCII**, **UTF8**, **UTF16** or **UTF16BE**. The **UTF8** encoding cannot be specified for a fixed width source file (i.e. when the Source File Field Delimiter is blank) because UTF8 is a variable length character encoding.
Source File Field Delimiter	Optional character that separates the fields within each record of the Source File. The delimiter identifies the end of each field. Common field delimiters are tab, comma, colon, semi-colon, pipe, tilde. If no field delimiter is specified the record is regarded as fixed-width. **Note** If an ASCII character value is used this field may show as an unprintable character. To enter a special character enter the uppercase string CHAR with the ASCII value in brackets (e.g. CHAR(9)).
Source File Field Enclosure Delimiter	• **PDW Only** - For **PDW** loads, the delimiter can be specified either as a character or as a hex value (such as 0x22 for a double quote).
Source File Record Terminator	Optional string to identify how each line/record in the Source File is ended/terminated/delineated. The system default is used when not specified. On UNIX/Linux systems, end-of-line is typically line-feed (ASCII 10). On Windows systems, end-of-line is typically carriage-return (ASCII 13) and line-feed (ASCII 10).
Source File has Field Headings/Labels	Indicates whether the first line of the Source File contains a heading/label for each field, which is not regarded as data so it should not be loaded.
Trigger File Details	Optional Trigger File identification and definition information. If a Trigger File is specified, the Source File will not be loaded until the Trigger File is available.
Trigger File Path	The purpose of the trigger file is to indicate that the copying/loading of the main file has completed and that it is now safe to load the file. Secondly the trigger file may contain control sums to validate the contents of the main load file. This field should contain the full path name to the directory in which a trigger file is located on the Windows or UNIX systems. If this field and/or the **Trigger Name** field is populated then the scheduler will look for this file rather than the actual load file.
Trigger File Name	Refers to the name of the file that is used as a trigger to indicate that the main load file is available. A trigger file typically contains check sums. If the trigger file exists then this is the file that is used in the Wait Seconds, and Action on wait expire processing. (see notes under Trigger Path above).
Trigger File Field Delimiter	If the trigger file provides control information then the delimiter identifies the field separation, or **\n** for a return. The data found will be loaded into parameter values whose names will be prefixed by the prefix specified and numbered **0** to **n**.
Trigger Parameter	If a trigger file and delimiter have been specified then the contents of the trigger file are

Fields	Description
Name Prefix	loaded into parameters. The parameters will be prefixed by the contents of this field and suffixed by an underscore and the parameter number. These parameters can be viewed under **Tools > Parameters** from the WhereScape RED menu bar. The checking of these parameters can be achieved by generating a **Post Load procedure**. An example set of parameters may be budget_0, budget_1 and budget_2 where there are 3 values in the trigger file and the prefix is set to 'budget'.
Load Configuration	Configuration settings to control the load processing.
Check existence of Source File	This field controls whether the load process checks if the file exists before performing the load. This check can only be disabled when doing script-based TPT Loads from Windows and UNIX/Linux connections, which can improve the use of built-in TPT functionality to wait for the arrival of the file.
Wait for Source File or Trigger File	Controls whether the load process waits for the file to arrive when it is NOT available to load. Enabling this allows the wait-related properties to be specified. When a Trigger File is specified the load waits for it rather than the Source File.
Wait Limit (in seconds)	Maximum duration to wait when no file is available, which is specified in seconds e.g. 1800 seconds to wait up to 30 minutes. A value of 0 equates to no wait. If the wait time expires and the specified file cannot be found, then the load will exit with the status defined in Action, e.g. **Default Action = Error**
Action when Wait Limit Reached	Action to take when the Wait Limit has been reached and no file is available. The specified action impacts any remaining tasks in the currently running job. When 'Success' or 'Warning' is specified, the WhereScape RED Scheduler will continue to run any dependent tasks in the running job. In contrast, specifying the 'Error' or 'Fatal Error' actions will cause the scheduler to stop/fail the job and hold any remaining tasks when the "Wait Limit" is reached.
BULK INSERT Options	**SQL Server only**- Optional comma-delimited list of options to control the behavior of the SQL Server BULK INSERT command. For example the following options will respectively lock the table; set the batch size to 100,000 rows; and fail the load when the first error occurs: TABLOCK, BATCHSIZE= 100000, MAXERRORS=0.
Archived File Details	Optional Archived file identification and definition information. Once a file has been successfully loaded or processed, it can be optionally archived by moving it and/or renaming it.
Compress Source File when Archive	Optionally compresses the successfully loaded Source File if it is archived.
Archived Source File Path	Optional full path (absolute path) of the folder/directory to MOVE the successfully loaded Source File on the Windows or UNIX/Linux system.
Archived Source File Name	Optional new name to RENAME the successfully loaded Source File to. By default, the original file name is used it is optional to rename it. However, the in-built variable $SEQUENCE$ can be used to include a unique sequence number in the new name. Likewise, the in-built variables $YYYY$, MM, DD, HH, MI and/or SS can be used to include the number of the current year, month, day, hour, minute and/or second respectively. The date/time components can be used separately or can be combined together using one set of enclosing $ such as $YYYYMMDD$.
Archived Trigger File Path	Optional full path (absolute path) of the folder/directory to MOVE the successfully processed Trigger File on the Windows or UNIX/Linux system.
Archived Trigger File Name	Optional new name to RENAME the successfully processed Trigger File to. By default, the original file name is used it is optional to rename it. However, the in-built variable $SEQUENCE$ can be used to include a unique sequence number in the new name. Likewise, the in-built variables $YYYY$, MM, DD, HH, MI and/or SS can be used to include the number of the current year, month, day, hour, minute and/or second respectively. The date/time components can be used separately or can be combined together using one set of enclosing $ such as $YYYYMMDD$.

Fields	Description
PDW File Load Options (Configuration details specific to PDW File Loads)	
Load Mode	Select the loading mode from the list. The available options include: Append/Fast Append/Reload/Upsert. Note that selecting the Upsert Load Mode requires a business key to be defined.
Use Staging database	If this field is set and a staging database is defined in the target connection, then the database will be used for staging, otherwise the target connection's database will be used for staging. To set a Staging Database in the target connection, refer to **Connections - Database** for details.
Additional Command Line Parameters	Optional additional command line parameters (space separated) for PDW file loads which are not configured elsewhere. Such additional configurations include: certificate verification (-N), skipping loads of empty files (-se), specifying the batch load size (-b <batchsize>), white space trimming around string fields (-c), converting empty strings to null (-E), and load failure options (-rt value|percentage, -rv <reject_value>, -rs <reject_sample_size>).

File and Script based load source screens

SQL Server File load Source screen:

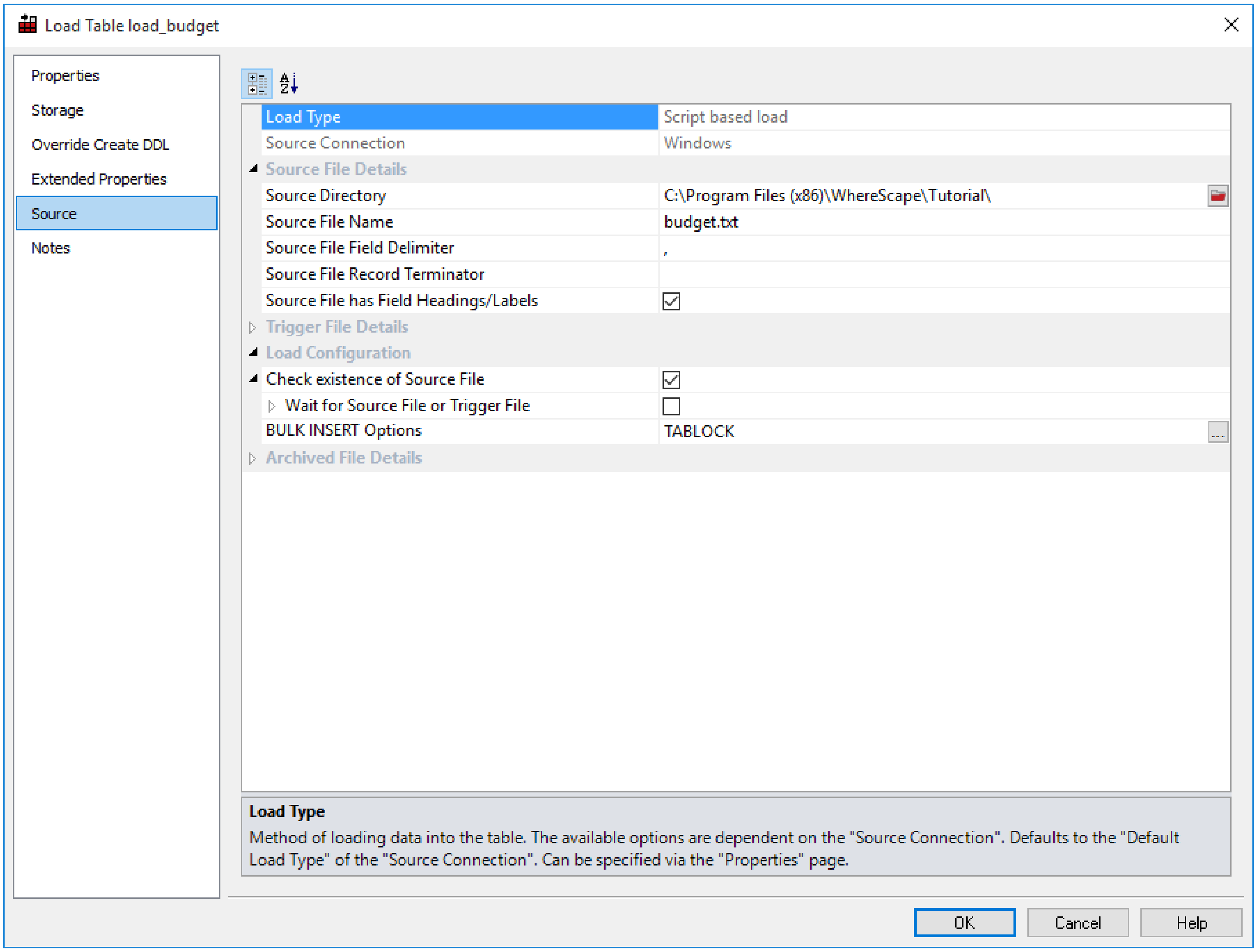


Load Table load_budget2

Load Type	File load
Source Connection	Windows
◢ Source File Details	
Source Directory	C:\Program Files (x86)\WhereScape\Tutorial\
Source File Name	budget.txt
Source File Field Delimiter	,
Source File Record Terminator	
Source File has Field Headings/Labels	☑
◢ Trigger File Details	
Trigger File Path	
Trigger File Name	
Trigger File Field Delimiter	
Trigger Parameter Name Prefix	
◢ Load Configuration	
◢ Check existence of Source File	☑
◢ Wait for Source File or Trigger File	☐
Wait Limit (in seconds)	1
Action when Wait Limit Reached	Warning
BULK INSERT Options	TABLOCK
▷ Archived File Details	

Load Type
Method of loading data into the table. The available options are dependent on the "Source Connection". Defaults to the "Default Load Type" of the "Source Connection". Can be specified via the "Properties" page.

SQL Server Script load Source screen from a Windows connection:

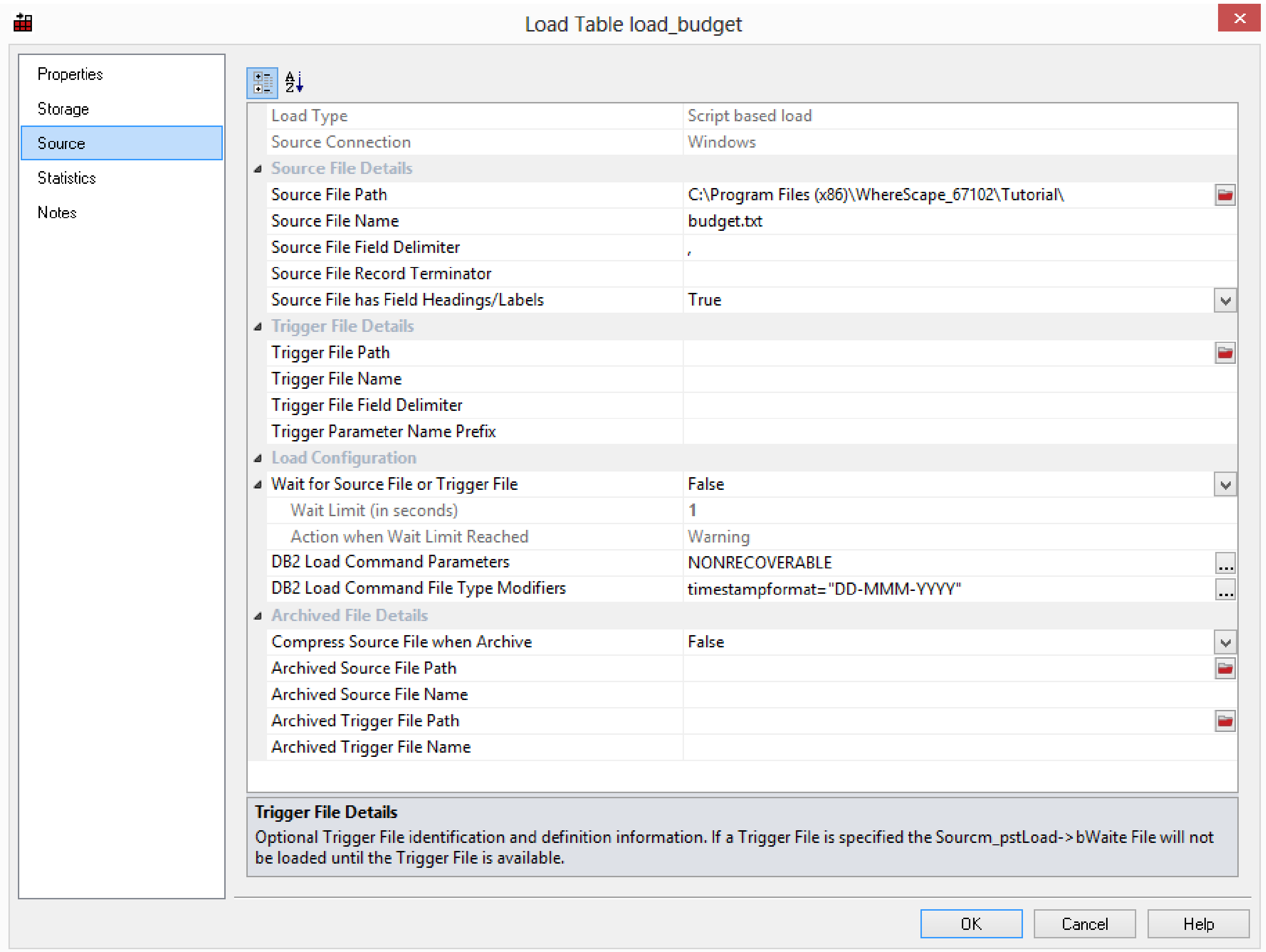


Load Table load_budget

Load Type	Script based load
Source Connection	Windows
Source File Details	
Source Directory	C:\Program Files (x86)\WhereScape\Tutorial\
Source File Name	budget.txt
Source File Field Delimiter	,
Source File Record Terminator	
Source File has Field Headings/Labels	☑
Trigger File Details	
Load Configuration	
Check existence of Source File	☑
Wait for Source File or Trigger File	□
BULK INSERT Options	TABLOCK
Archived File Details	

Load Type
Method of loading data into the table. The available options are dependent on the "Source Connection". Defaults to the "Default Load Type" of the "Source Connection". Can be specified via the "Properties" page.

DB2 File load Source screen:



Load Type	Script based load
Source Connection	Windows
Source File Details	
Source File Path	C:\Program Files (x86)\WhereScape_67102\Tutorial\
Source File Name	budget.txt
Source File Field Delimiter	,
Source File Record Terminator	
Source File has Field Headings/Labels	True
Trigger File Details	
Trigger File Path	
Trigger File Name	
Trigger File Field Delimiter	
Trigger Parameter Name Prefix	
Load Configuration	
Wait for Source File or Trigger File	False
Wait Limit (in seconds)	1
Action when Wait Limit Reached	Warning
DB2 Load Command Parameters	NONRECOVERABLE
DB2 Load Command File Type Modifiers	timestampformat="DD-MMM-YYYY"
Archived File Details	
Compress Source File when Archive	False
Archived Source File Path	
Archived Source File Name	
Archived Trigger File Path	
Archived Trigger File Name	

Trigger File Details

Optional Trigger File identification and definition information. If a Trigger File is specified the Sourcm_pstLoad->bWaite File will not be loaded until the Trigger File is available.

Script Based Load

A script based load table must have a Host Script defined. During the load process this host script is executed and the results returned.

During the 'drag and drop' creation of a load table from a UNIX or Windows file a script can be generated by selecting one of the 'Script based' load options. This script can then be edited to fully meet any requirements.

Note

If the UNIX Scheduler is not being used and a UNIX script is generated and executed interactively, the following two variables will be setup as part of the environment.

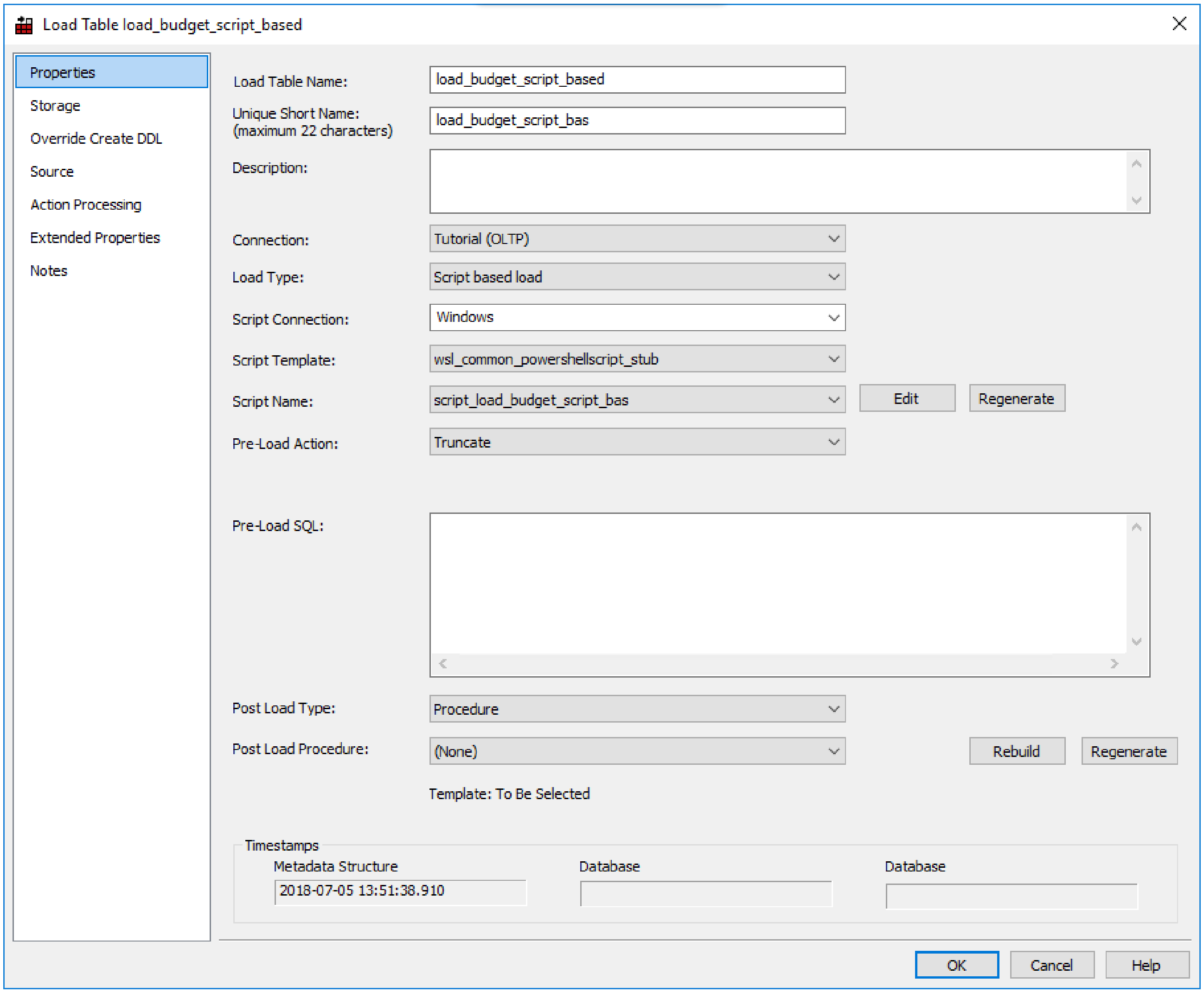
- DSS_USER=username
- DSS_PWD=password

Where 'username' and 'password' are the entries for the data warehouse database.

There are a number of conventions that must be followed if these host scripts are to be used by the WhereScape scheduler.

1. The first line of data in **standard out** must contain the resultant status of the script. Valid values are '1' to indicate success, '-1' to indicate a Warning condition occurred but the result is considered a success, '-2' to indicate a handled Error occurred and subsequent dependent tasks should be held, -3 to indicate an unhandled Failure and that subsequent dependent tasks should be held.
2. The second line of data in **standard out** must contain a resultant message of no more than 256 characters.

3. Any subsequent lines in **standard out** are considered informational and are recorded in the audit trail. The normal practice is to place a minimum of information in the audit trail. All bulk information should be output to **standard error**
4. Any data output to **standard error** will be written to the error/detail log. Both the audit log and detail log can be viewed from the WhereScape RED tool under the **Scheduler** window.
5. When doing **Script based loads**, it is easy to use the **Regenerate** button beside the **Script Name** field to regenerate the scripts. You can then click the **Edit** button to view and edit the generated script, as per your requirements.



Load Table Name: load_budget_script_based
Unique Short Name: (maximum 22 characters) load_budget_script_bas
Description:
Connection: Tutorial (OLTP)
Load Type: Script based load
Script Connection: Windows
Script Template: wsl_common_powershellscript_stub
Script Name: script_load_budget_script_bas
Pre-Load Action: Truncate
Pre-Load SQL:
Post Load Type: Procedure
Post Load Procedure: (None)
Template: To Be Selected

Timestamps

Metadata Structure	Database	Database
2018-07-05 13:51:38.910		

Note

Script-based loads on Windows supports both DOS Batch and **PowerShell scripts**.

File Wild Card Load

The loading of multiple files in one pass is best undertaken as a host script.

A generated host script contains the code to handle wild card loads and the subsequent loading and renaming of multiple files. The process for setting up wild card loads is as follows:

1. Drag one of the files into a **Load Table** target and select one of the **Script** based options.

2. If necessary edit the generated script and replace the load file name with the required wild card version. (For example test_result_234.txt renamed to test_result_*.txt. This load file name is defined as a variable at the start of the script.
3. If renaming or moving to another directory is required, then you may need to uncomment the rename phase and enter the desired directory or rename syntax.
4. If only one file at a time is to be processed, but still based on a wild card file name, then you may need to uncomment the break statement at the bottom of the script. This will result in the loading of only the first file encountered that matches the wild card criteria.

In every event, check the generated script as it will contain instructions on how to handle wild cards for the appropriate script type (Windows or UNIX).

External Load

For an externally loaded table, the only property that is executed is the Post Load procedure.

Any **After** transformations recorded against any of the columns in an Externally loaded table will also be processed.

Post load procedures can either be manually generated from a RED provided procedure outline or generated leveraging a RED template—refer to **Rebuilding Update Procedures** for details.

Note

For custom database targets, the option to use Post Load Scripts is provided, as an alternative to post load procedure. Post load scripts can be generated using a RED template—refer to **Rebuilding Update Procedures** for details.

Load Table Transformations

Each load table column can have a transformation associated with it.

Refer to **Transformations** and **Load Table Column Transformations** for details.

Post-Load Procedures

If a procedure name is entered in the **Post Load Procedure** drop-down field of a Load table's Properties, then this procedure will be executed after the load has completed and after any **after transformations** have occurred.

Click the **Rebuild** button to start the process of generating a new procedure. The **Regenerate** button is used to regenerate the Load table's existing post load procedure.

Refer to **Load Table Column Transformations** for details.

Post load procedures can either be manually generated from a RED provided procedure outline or generated leveraging a RED template—refer to **Rebuilding Update Procedures** for details.

Note

For custom database targets, the option to use Post Load Scripts is provided, as an alternative to post load procedure. Post load scripts can be generated using a RED template—refer to **Rebuilding Update Procedures** for details

Changing Load Connection and Schema

The connection associated with a load table can be changed through the Properties of that table.

Connections can also be changed in bulk by using the following process:

1. Double-click the **Load** table object group in the left pane. This displays all the Load tables in the middle pane.
2. Select the Load tables that you want to change, using standard Windows selection.
3. Right-click to bring up the context menu and select **Change Connect/Schema**.
4. Select the new connection to change all the selected Load tables.

Note

You cannot change the connection **type** but it is possible to change from Database to ODBC connections when the following considerations are taken into account.

Switching Connection from ODBC to Database

This switch should be successful in most cases, although it may not provide the most performant load possible. By default ODBC connections use the source table/column transformation loading method as dates and potentially other data types need to be converted.

When switching to a database link load any transformations will still occur although they may no longer be necessary.

Switching Connection from Database to ODBC

Although it is possible to switch from a Database connection to an ODBC connection, the resultant extract may not function correctly. SQL Server tends to handle the date conversion better in most cases. When an extract occurs over an ODBC connection the date is converted in the extract from the source system into a standard ASCII format. This ASCII formatted date is then converted back when it is loaded into the load table. To resolve this problem place a transformation on any date fields.

Note

If a load table is created via drag and drop from an ODBC based connection WhereScape RED will build all the required date transformations.

There are several supplied APIs for changing schema and connections programmatically. Refer to **Callable Routines** for details.

Dimensions

Dimensions Overview

A dimension table is normally defined, for our purposes, as a table that enables us to constrain queries on the fact table.
A dimension is built from the Data Warehouse connection. Unless you are doing a retro-fit of an existing system, dimensions are typically built from one or more load tables.

The normal steps for creating a dimension are defined below and are covered in this chapter. The steps are:

- Identify the source transactional data that will constitute the dimension. If the data is sourced from multiple tables, ascertain if a join between the source tables is possible, or if a series of lookups would be a better option.
- Using the 'drag and drop' functionality, drag the load table that is the primary source of information for the dimension into a dimension target. Refer to **Building a Dimension** for details.
- If only one table is being sourced and most of the columns are to be used (or if prototyping) you can select the auto create option to build and load the dimension and skip the next 4 steps. Refer to **Building a Dimension** for details.
- Add columns from other load tables if required. Refer to **Building a Dimension** for details.
- Create the dimension table in the database. Refer to **Building a Dimension** for details.
- Build the update procedure. Refer to **Generating the Dimension Update Procedure** for details.
- Run the update procedure and analyze the results. Refer to **Dimension Initial Build** for details.
- Modify the update procedure as required. Refer to **Dimension Initial Build** for details.

Dimension Keys

Dimensions have two types of keys that we will refer to frequently. These are the **Business Key** and the **Artificial Key**.
A definition of these two key types follows:

Business Key

The business key is the column or columns that uniquely identify a record within the dimension. Where the dimension maps back to a single or a main table in the source system, it is usually possible to ascertain the business key by looking at the unique keys for that source table. Some people refer to the business key as the **natural key**. Examples of business keys are:

- The product SKU in a product dimension
- The customer code in a customer dimension
- The calendar date in a date dimension
- The 24 hour time in a time dimension (i.e. HHMM) (e.g.1710)
- The airport short code in an airport dimension.

It is assumed that business keys will never be NULL. If a null value is possible in a business key then the generated code will need to be modified to handle the null value by assigning some default value.

For example the 'Where' clause in a dimension update may become:

SQL Server: Where isnull(business_key,'N/A') = isnull(v_LoadRec.business_key,'N/A')

DB2: Where coalesce(business_key,'N/A') = coalesce(v_LoadRec.business_key,'N/A')

Note

Business keys are assumed to never be Null. If they could be null it is best to transform them to some value prior to dimension or stage table update. If this is not done, an unmodified update will probably fail with a duplicate key

Note

error on the business key index.

Artificial Key

The artificial key is the unique identifier that is used to join a dimension record to a Fact table. When joining dimensions to Fact tables it would be possible to perform the join using the business key. For Fact tables with a large number of records this however would result in slow query times and very large indexes. As query time is one of our key drivers in data warehouse implementations, the best answer is to always use some form of artificial key. A price is paid in the additional processing required to build the Fact table rows, but this is offset by the reduced query times and index sizes.

The artificial key is an integer and is built sequentially from 1 upwards.

Refer to Dimension Artificial Keys for a more detailed explanation. An artificial key is sometimes referred to as a **surrogate key**.

Building a Dimension

Dimensions are often sourced from one table in the base application. In many cases, there are also codes that require description lookups to complete the de-normalization of the dimensional data.

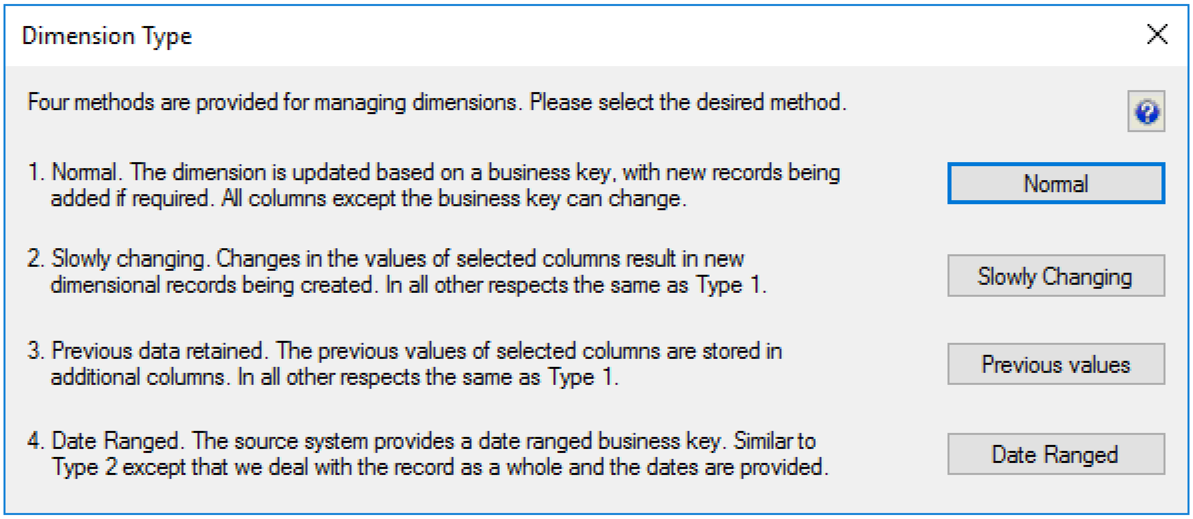
The process for building a dimension is the same for most other tables and begins with the drag and drop of the Load table that contains the bulk of the dimensional information.

Drag and Drop

1. Create a dimension target by double-clicking on the **Dimension group** in the left pane.
2. The middle pane displays a list of all existing dimensions. When this list is displayed in the middle pane, the pane is identified as a target for new dimension tables.
3. Browse to the Data Warehouse via the **Browse > Data Warehouse** menu option.
4. Drag the Load table, that contains the bulk of the dimensional columns, into the middle pane.
5. Drop the table anywhere in the pane.
6. The new object window appears and classifies the new object as a Dimension and provides a default name based on the Load table name. Either accept this name or enter another name and then click **OK** to proceed.

Dimension Type

A dialog appears as shown below. There are four choices for the default generation of the Dimension table and its update procedure.



Dimension Type

Four methods are provided for managing dimensions. Please select the desired method.

1. Normal. The dimension is updated based on a business key, with new records being added if required. All columns except the business key can change.
2. Slowly changing. Changes in the values of selected columns result in new dimensional records being created. In all other respects the same as Type 1.
3. Previous data retained. The previous values of selected columns are stored in additional columns. In all other respects the same as Type 1.
4. Date Ranged. The source system provides a date ranged business key. Similar to Type 2 except that we deal with the record as a whole and the dates are provided.

- The first choice being a **Normal** dimension where a dimensional record is updated and changed whenever any of the non business key information changes.
- The second choice is a **Slowly Changing** dimension where new dimension records are created when certain identified columns in the dimension change.

Note

For custom database targets, you can set the initial default values for the DSS columns that are used in the procedure generation for slowly changing dimensions. Refer to **DSS Columns for Custom Targets** for details.

- The third choice is a **Previous values** dimension, which enables the storing of the last values of selected fields in secondary columns.
- The fourth choice is a **Date Ranged** dimension, which supports source systems that provide start and end dates.

With any dimension, we identify a **Business Key** that uniquely identifies the dimension records.

For example in the case of the product dimension, the product **code** is deemed to be the business key. The code uniquely identifies each product within the dimension. The product may also have a name or description and various other attributes that distinguish it (e.g. size, shape, color, etc.).

A common question when handling dimensions is what to do when the name or description changes:

- Do we want to track our fact table records based only on the product code? or
- Do we also want to track records based on different descriptions?

An example :

code	description	product_group	sub_group
1235	15oz can of brussel sprouts	canned goods	sprouts

This product has been sold for many years and we consequently have a very good history of sales and the performance of the product in the market. The company does a '20% extra for free' promotion for 3 months during which time it increases the size of the can to 18oz. The description is also changed to be '15 + 3oz can of brussel sprouts'. At the end of the promotion, the product is reverted to its original size and the description changed back to its original name.

The question is, do we want to track the sales of the product when it had a different description (slowly changing) , or should the description of the product simply change to reflect its current name (normal). For this scenario a previous value dimension would not provide much advantage, so it is not discussed.

The decision is not a simple one and the advantages and disadvantages of each of the two choices is discussed below.

Slowly Changing

- Allows the most comprehensive analysis capabilities when just using the product dimension.
- Complicates the analysis. Does not allow a continuous analysis of the product called '15oz can of brussel sprouts' when the description is used. This analysis is however still available through the code which has not changed.
- Adds considerable additional processing requirements to the building of the fact tables that utilize this dimension.
- May track data quality improvements rather than real business change.

Normal

- Does not allow specific analysis of the product during its size change. Note, however that this analysis will probably be available through the combination of a 'promotion' dimension.
- Provides a continuous analysis history for the product called '15oz can of brussel sprouts'. An analysis via description and code will produce the same results.
- Simplifies analysis from an end user's perspective.

As mentioned above, the choice is never a simple one. Even among experienced data warehouse practitioners, there will be a variety of opinions. The decision must be based on the business requirements. In many cases keeping the analysis simple is the best choice, at least in the early stages of a data warehouse development. Slowly changing dimensions do have a place, but there is nearly always an alternate method that provides equal or better results. In the example above, a promotion dimension coupled with the product dimension could provide the same analysis results whilst still keeping product only analysis simple and easy to understand.

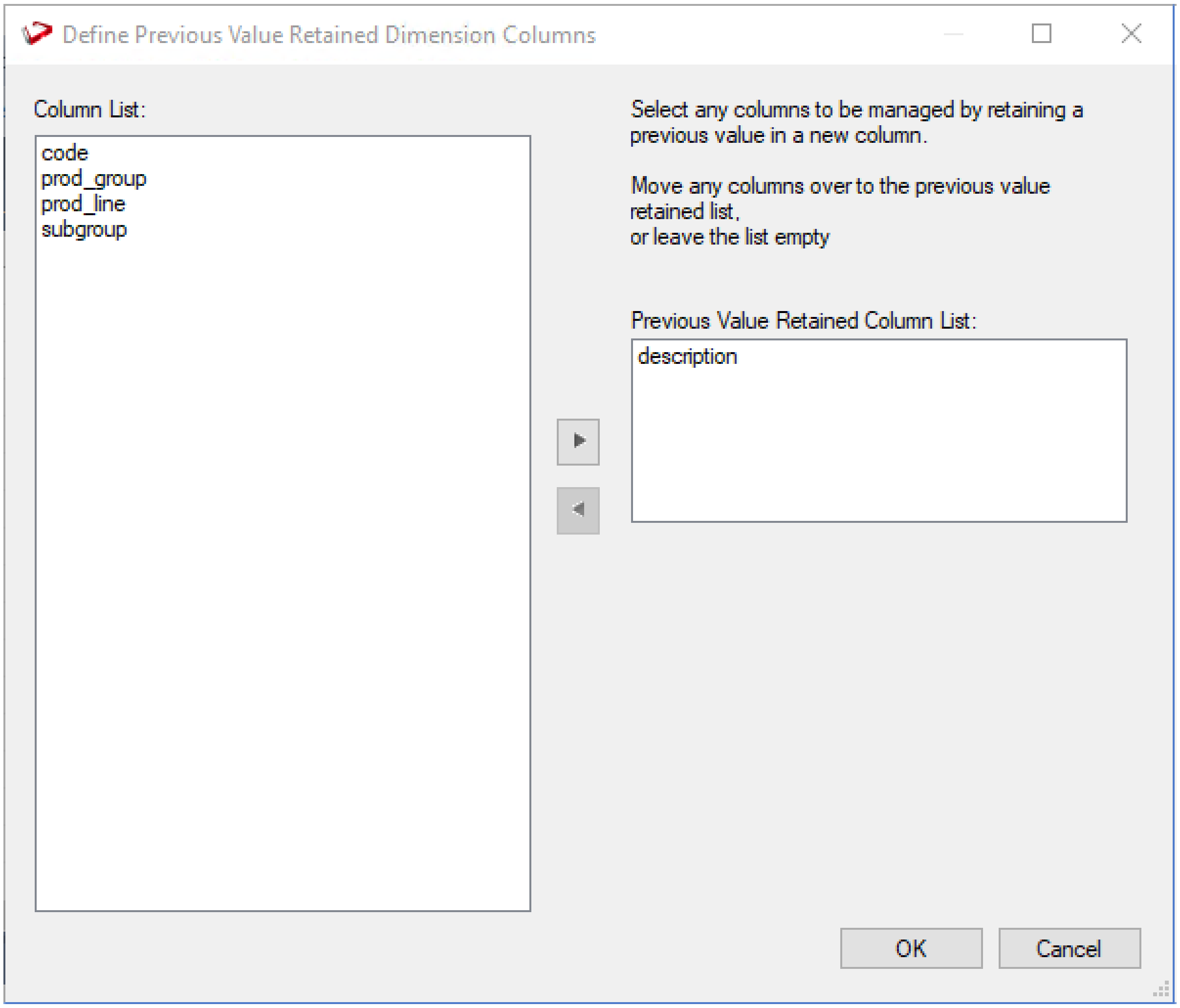
Tip

Do not over complicate the design of an analysis area. Keep it simple and avoid the unnecessary use of slowly changing dimensions.

Previous Values Dimension Type

If you selected a 'Previous values' dimension type then the following questions will be asked. If one of the other two types where chosen then proceed directly to **Dimension Properties**.

The following dialog appears:



Define Previous Value Retained Dimension Columns

Column List:

code
prod_group
prod_line
subgroup

Select any columns to be managed by retaining a previous value in a new column.

Move any columns over to the previous value retained list,
or leave the list empty

Previous Value Retained Column List:

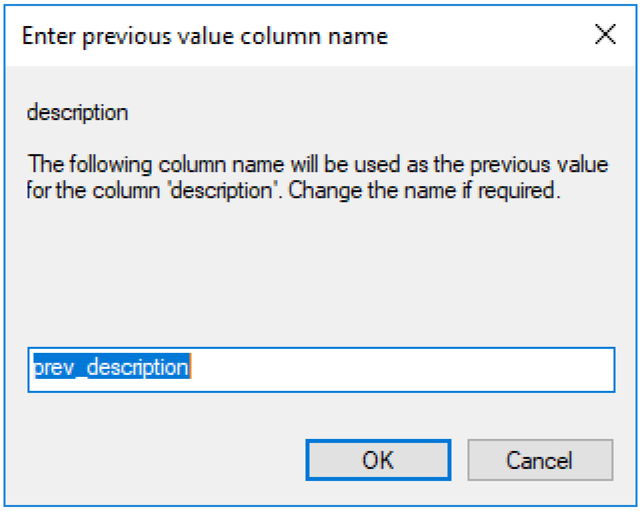
description

It requests the definition of each column that will be managed by storing the previous value in a separate additional column. Note that the business key cannot be used in this manner, as a change to the business key will result in a new record.

Select the columns to be managed and click **OK**.

The following dialog appears for each column that has been selected.

Either accept the name of the additional column or enter a new name.



Enter previous value column name

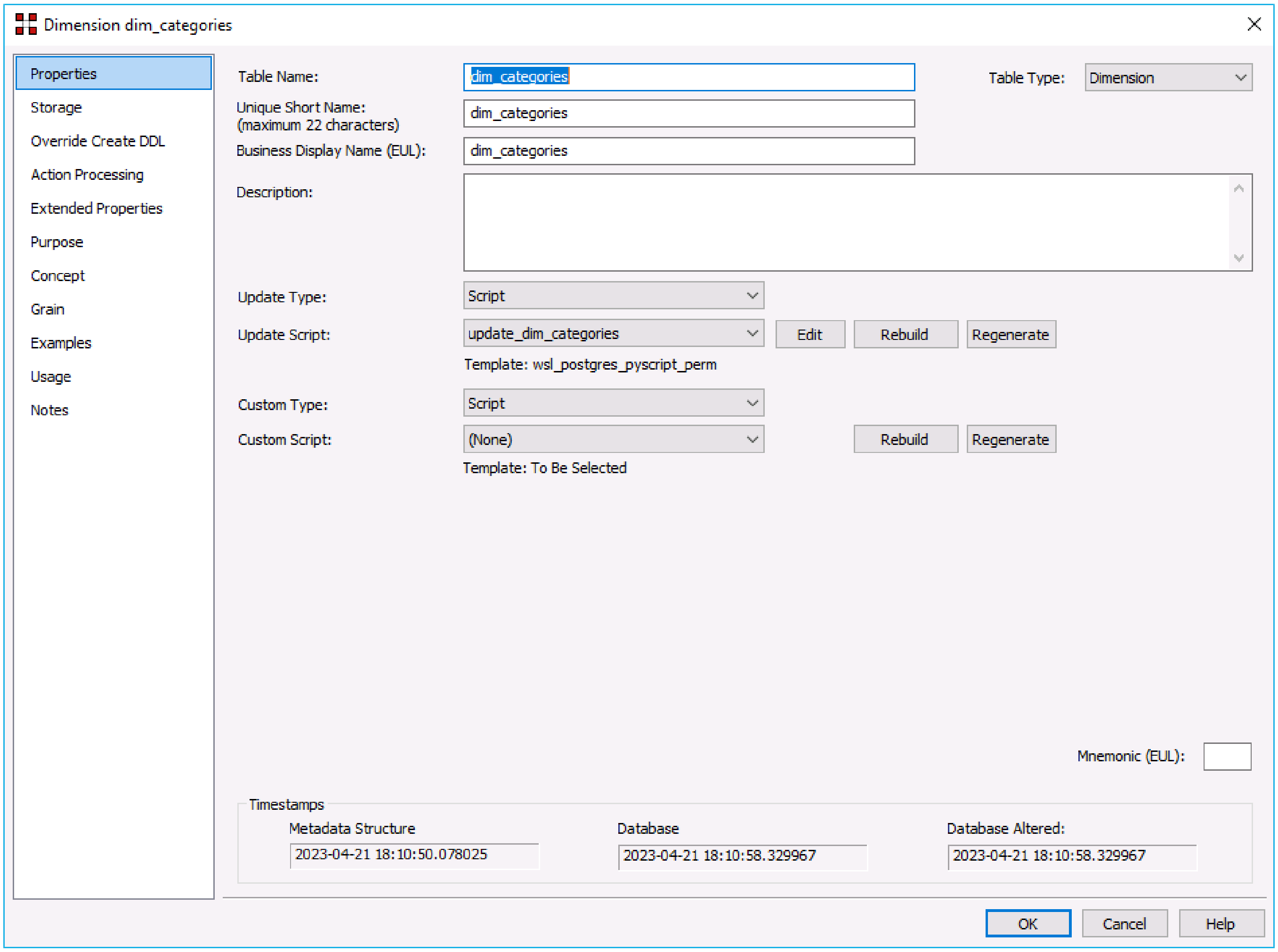
description

The following column name will be used as the previous value for the column 'description'. Change the name if required.

prev_description

Dimension Properties

Once the dimension type is chosen, the **Properties** screen appears.
Change the storage options if required.
If prototyping, and the dimension is simple (i.e. one source table) then it is possible to create, load and update the dimension in a couple of steps.
If you wish to do this select the **(Build Procedure...)** option from the **Update Procedure** drop-down, and answer **Create and Load** to the next question.



Dimension dim_categories

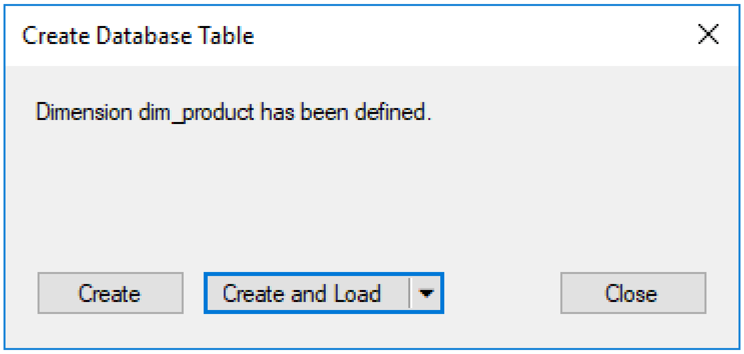
Table Name: dim_categories
Table Type: Dimension
Unique Short Name: (maximum 22 characters) dim_categories
Business Display Name (EUL): dim_categories
Description:
Update Type: Script
Update Script: update_dim_categories
Template: wsl_postgres_pyscript_perm
Custom Type: Script
Custom Script: (None)
Template: To Be Selected
Mnemonic (EUL):

Timestamps

Metadata Structure	Database	Database Altered:
2023-04-21 18:10:50.078025	2023-04-21 18:10:58.329967	2023-04-21 18:10:58.329967

Create and Load

The following window appears after the **Properties** screen and asks if you want to create the dimension table in the database and execute the update procedure.



Create Database Table

Dimension dim_product has been defined.

If you are satisfied with the columns that will be used and do not wish to add additional columns you can click the **Create and Load** button.

In this case it would be normal practice to select the **(Build Procedure...)** option from the 'Update Procedure' drop-down box in the previous **Properties** window.

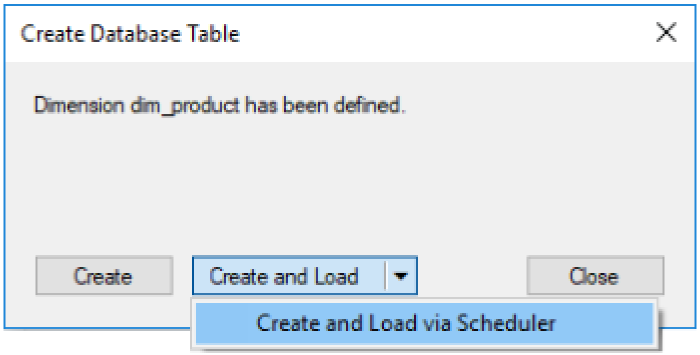
If no procedure has been selected then select the **Create** button as no update will happen in any event.

If **Create** or **Create and Load** is selected and a new procedure creation was chosen, proceed directly to the Generating the Dimension Update Procedure section.

If you have additional columns to add or columns to delete, click **Close** and proceed as follows below.

Note

It is possible to create and load the table via the Scheduler by selecting this option from the drop-down list on the **Create and Load** button:



Create Database Table

Dimension dim_product has been defined.

Deleting and Changing columns

The columns defined for the dimension is displayed in the middle pane.

It is possible to delete any unwanted columns by highlighting a column name or a group of names and choosing the **Delete** key.

The name of a column can also be changed by selecting the column and using the right-click menu to edit its properties. Any new name must conform to the database naming standards. Good practice is to use alphanumerics and the underscore character.

Refer to the Column Properties for a more detailed description of the various fields.

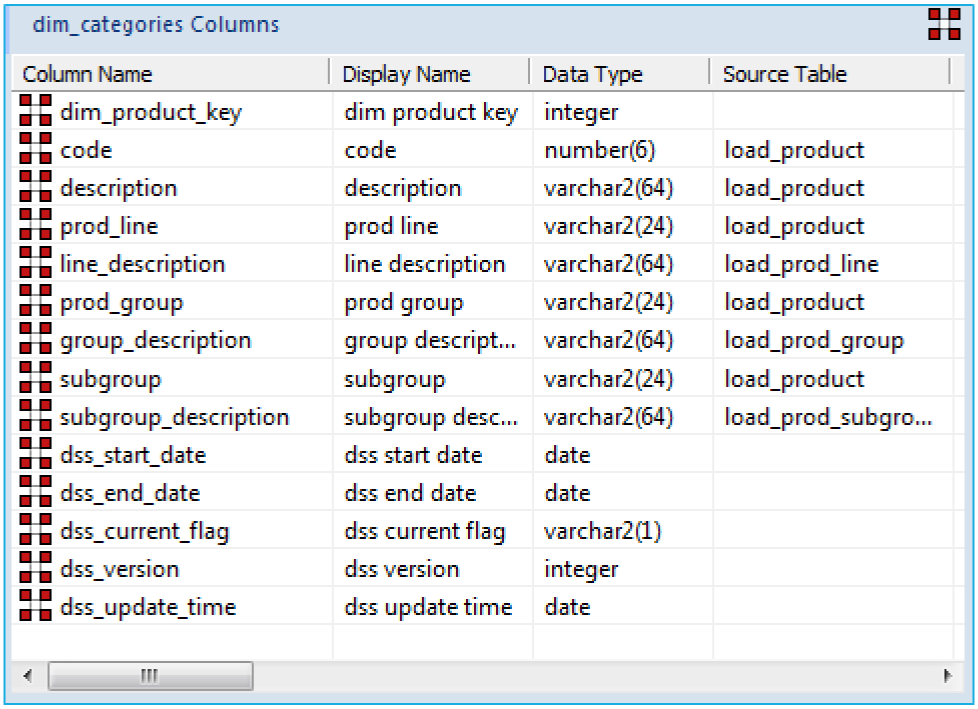
Tip

When prototyping, and in the initial stages of an analysis area build it is best not to remove columns, nor to change their names to any great extent. This type of activity is best left until after end users have used the data and provided feedback.

Adding additional columns

With the columns of the dimension table displayed in the middle pane, this pane is considered a drop target for additional columns.

It is a simple matter therefore to select columns from other tables and to drag these columns into the middle pane.



dim_categories Columns

Column Name	Display Name	Data Type	Source Table
dim_product_key	dim product key	integer	
code	code	number(6)	load_product
description	description	varchar2(64)	load_product
prod_line	prod line	varchar2(24)	load_product
line_description	line description	varchar2(64)	load_prod_line
prod_group	prod group	varchar2(24)	load_product
group_description	group descript...	varchar2(64)	load_prod_group
subgroup	subgroup	varchar2(24)	load_product
subgroup_description	subgroup desc...	varchar2(64)	load_prod_subgro...
dss_start_date	dss start date	date	
dss_end_date	dss end date	date	
dss_current_flag	dss current flag	varchar2(1)	
dss_version	dss version	integer	
dss_update_time	dss update time	date	

The source table shows where each column was dragged from.

In the example above, the description column is acquired from the load_product, load_prod_group and load_prod_subgroup tables. To create the dimension table, these columns must be assigned unique names. For this example the last two columns have been renamed to group_description and subgroup_description.

There are a number of columns that do not have a source table. These columns have been added by WhereScape RED and are added depending on earlier choices.

A description of these columns follows:

Column Name	Description
dim_product_key	The unique identifier (artificial key) for the dimension. This key is used in the joins to the fact table.
dss_start_date	Used for slowly changing dimensions. This column provides a date time stamp when the dimension record came into existence. It is used to ascertain which dimension record should be used when multiple are available.
dss_end_date	Used for slowly changing dimensions. This column provides a date time stamp when the dimension record ceased to be the current record. It is used to ascertain which dimension record should be used when multiple are available.
dss_current_flag	Used for slowly changing dimensions. This flag identifies the current record where multiple versions exist.
dss_source_system_key	Added to support dimensions that cannot be fully conformed, and the inclusion of subsequent source systems. See the ancillary settings section for more details.
dss_version	Used for slowly changing dimensions. This column contains the version number of a dimension record. Numbered from 1 upwards with the highest number being the latest or current version. It forms part of the unique constraint for the business key of a slowly changing dimension.
dss_update_time	Indicates when the record was last updated in the data warehouse.
dss_create_time	Indicates when the record was first created in the data warehouse

Manually adding previous value columns

If a **Previous Value** type of dimension is chosen, or in fact if the dimension is converted to this type, it is possible to manually add any required columns that were not defined as part of the create. The steps are:

1. Add a new column by dragging in the column that is to have a previous value stored.
2. Change the name to a unique name. Typically by adding the prefix 'prev_' to the column name**.**
3. Change the source table, to be that of the dimension we are building.
4. Set the Key Type to 4.
5. Having performed these actions WhereScape RED will detect the column and build the appropriate code during the procedure generation phase.

Creating the table

Once the dimension has been defined in the metadata we need to physically create the table in the database.

1. To do this, right-click on the dimension name and choose **Create (ReCreate)** from the pop-up menu.
2. The **Results** pane show the results of the creation. It displays a message to the effect that the dimension table was created.
3. If the table was not created then ascertain and fix the problem. A common problem is a 'Duplicate column' where a column has the same name in two of the source tables.
4. The best way to find such a column is to double-click on the list heading **Col name**; which will sort the column names into alphabetical order.
   Another double-click on the heading will sort the columns back into their create order.

SQL Server makes use of an identity attribute on the artificial key column.

The next section covers the **Generating the Dimension Update Procedure**.

Generating the Dimension Update Procedure

Once a dimension has been defined in the meta data and created in the database, an update procedure can be generated to handle the joining of any tables and the update of the dimension records.

Notes

- You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.
- If the dimension is created in a custom database target, then an additional processing option is available—PowerShell script-based processing. Refer to **Script Templates for Custom Database Table Objects** for details.

The zero key row

WhereScape RED by default, inserts a record into the dimension with an artificial key value of zero. This record is used to link any fact records that do not have valid dimension joins.

The values of the various columns in this record are acquired from the contents of the field **Zero Key Value** which is set in the Properties screen of each dimension column.

Generating a Procedure

1. To generate a procedure, right-click on the dimension name to edit the properties for the dimension.
2. From the **Update Procedure** drop-down list select **(Build Procedure...)**.
3. Click **OK** to update the properties and start the process of generating the new procedure.
4. A **Procedure Build Type** dialog appears, allowing you to select between a **Cursor** and **Set** procedure build types from the drop-down list.

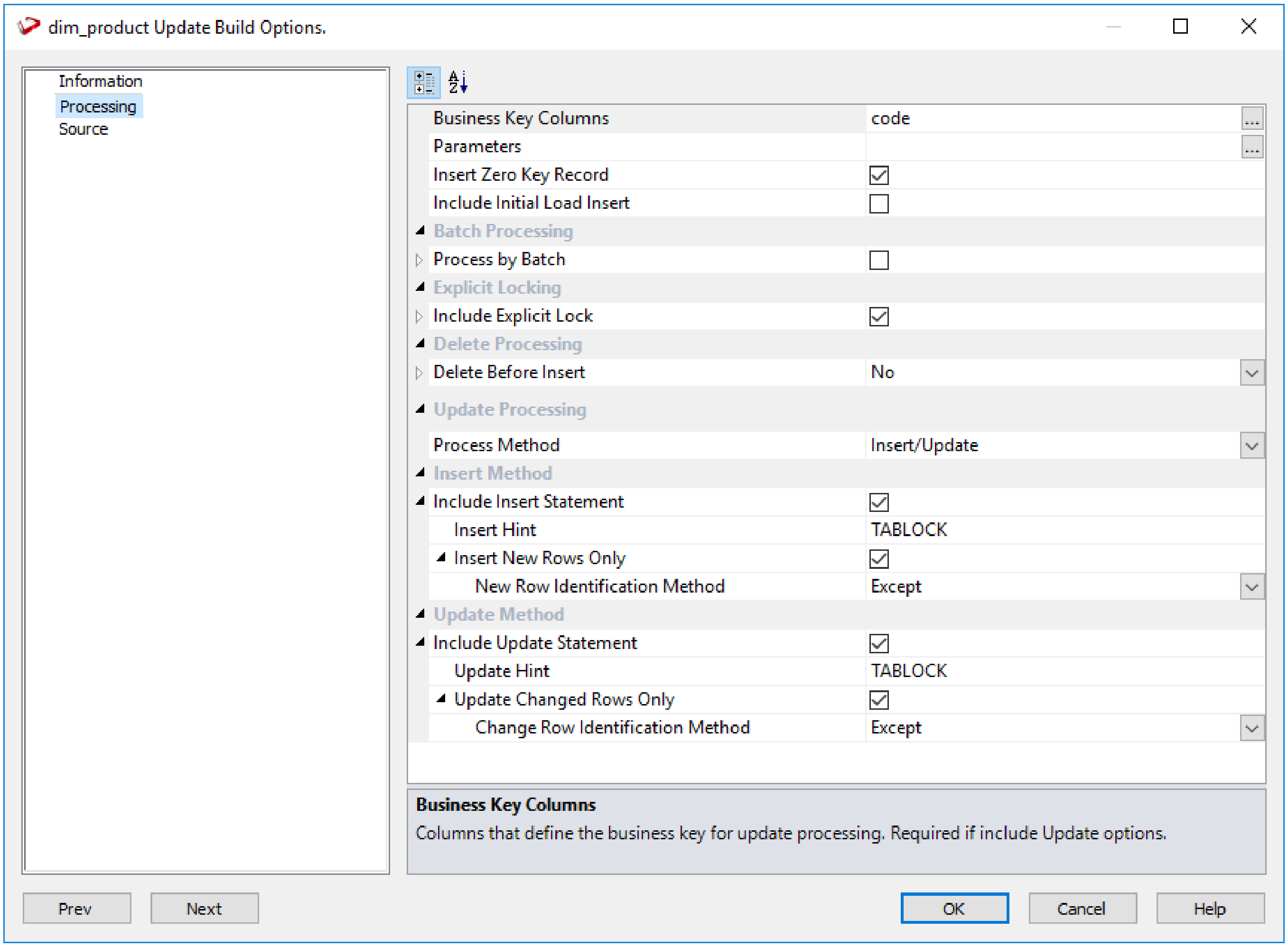
A **Set** based procedure performs one SQL statement to join all the source tables together and then insert this data into the Dimension. This is normally the fastest method of populating a table.

See below in this section for different options using the **Cursor** procedure building type.

Set Based Procedure Building types

1. Select **Set** from the drop-down menu.
2. The following window appears to enter the Update Procedure Build Options.
3. Click on the **ellipsis** button on the top rightmost corner to select the **Business Key Columns** for the dimension.

Processing tab



Business Key Columns	code
Parameters	
Insert Zero Key Record	☑
Include Initial Load Insert	□
Batch Processing	
Process by Batch	□
Explicit Locking	
Include Explicit Lock	☑
Delete Processing	
Delete Before Insert	No
Update Processing	
Process Method	Insert/Update
Insert Method	
Include Insert Statement	☑
Insert Hint	TABLOCK
Insert New Rows Only	☑
New Row Identification Method	Except
Update Method	
Include Update Statement	☑
Update Hint	TABLOCK
Update Changed Rows Only	☑
Change Row Identification Method	Except

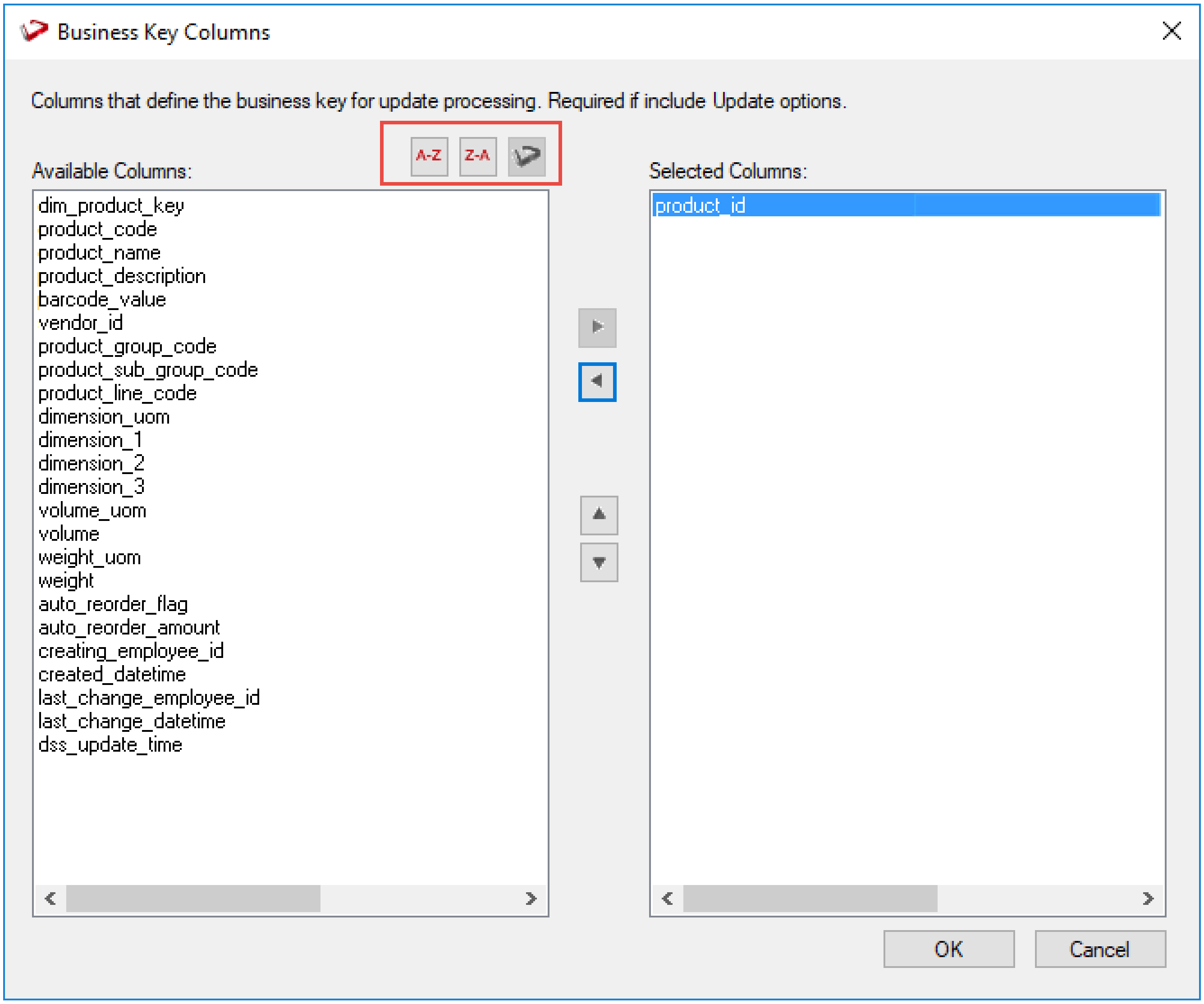
Business Key Columns

Columns that define the business key for update processing. Required if include Update options.

Business key

A business key can be made up of multiple columns, but it must provide a unique identifier. Where multiple columns uniquely and separately identify the dimension, choose one to act as the primary business key.

For example, a source table may have a unique constraint on both a product code and a product description. Therefore, the description as well as the code must be unique. It is of course possible to combine the two columns, but the normal practice would be to choose the code as the business key.



Columns that define the business key for update processing. Required if include Update options.

Available Columns:

dim_product_key
product_code
product_name
product_description
barcode_value
vendor_id
product_group_code
product_sub_group_code
product_line_code
dimension_uom
dimension_1
dimension_2
dimension_3
volume_uom
volume
weight_uom
weight
auto_reorder_flag
auto_reorder_amount
creating_employee_id
created_datetime
last_change_employee_id
last_change_datetime
dss_update_time

Selected Columns:

product_id

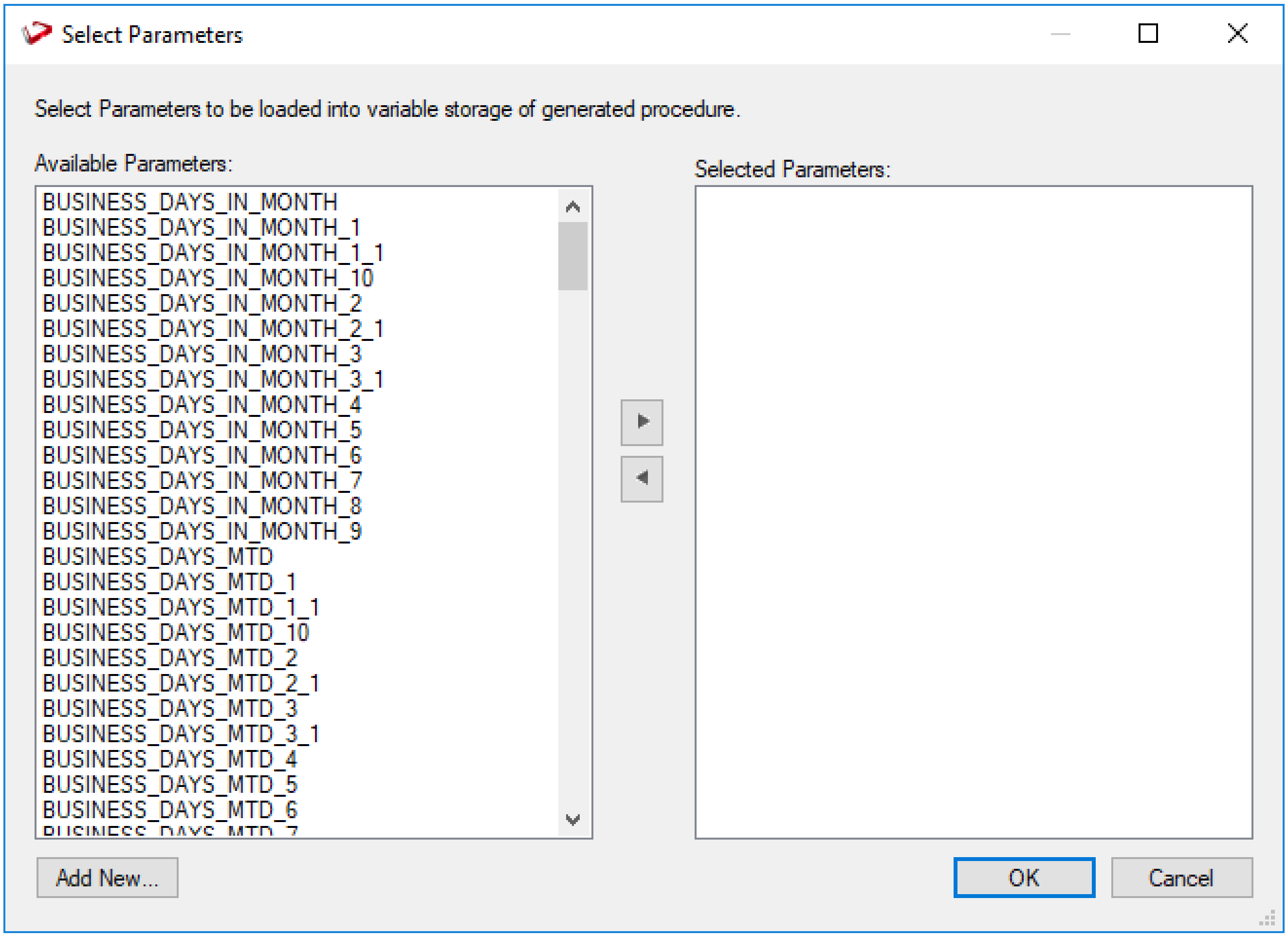
Tips:

- Use the column name ascending/descending buttons to sort column names. To revert to the meta column order, click on the meta column order button.
- **NULL Values** - none of the columns chosen as the business key should ever contain a NULL value. See the note at the start of the Dimensions chapter.

Parameters

If WhereScape RED parameters exist in the metadata, the following window is displayed. Any parameters selected in this window (by moving them to the right pane), are included in the generated update procedure as variables.

The procedure includes code to retrieve the value of the parameter at run time and store it in the declared variable.



Select Parameters

Select Parameters to be loaded into variable storage of generated procedure.

Available Parameters:

BUSINESS_DAYS_IN_MONTH
BUSINESS_DAYS_IN_MONTH_1
BUSINESS_DAYS_IN_MONTH_1_1
BUSINESS_DAYS_IN_MONTH_10
BUSINESS_DAYS_IN_MONTH_2
BUSINESS_DAYS_IN_MONTH_2_1
BUSINESS_DAYS_IN_MONTH_3
BUSINESS_DAYS_IN_MONTH_3_1
BUSINESS_DAYS_IN_MONTH_4
BUSINESS_DAYS_IN_MONTH_5
BUSINESS_DAYS_IN_MONTH_6
BUSINESS_DAYS_IN_MONTH_7
BUSINESS_DAYS_IN_MONTH_8
BUSINESS_DAYS_IN_MONTH_9
BUSINESS_DAYS_MTD
BUSINESS_DAYS_MTD_1
BUSINESS_DAYS_MTD_1_1
BUSINESS_DAYS_MTD_10
BUSINESS_DAYS_MTD_2
BUSINESS_DAYS_MTD_2_1
BUSINESS_DAYS_MTD_3
BUSINESS_DAYS_MTD_3_1
BUSINESS_DAYS_MTD_4
BUSINESS_DAYS_MTD_5
BUSINESS_DAYS_MTD_6
BUSINESS_DAYS_MTD_7

Selected Parameters:

The variables can also be used in column transformations and in the from/where clause for the update procedure. Some databases have a 30 character limit for variable names. WhereScape RED ensures the variables added for any parameters are less than 30 characters long by creating variable names in the form v_ followed by the first 28 characters of the parameter name.

For example, a parameter called MINIMUM_ORDER_NUMBER_SINCE_LAST_SOURCE_LOAD will be available as the variable v_MINIMUM_ORDER_NUMBER_SINCE_L.

Tips

- WhereScape RED parameters should be unique within the first 28 characters to avoid conflicting variables names.
- If the desired parameter doesn't exist in the metadata yet, a new parameter can be added by clicking on the **Add New** button on the bottom leftmost corner of the Select Parameters dialog.

Refer to **Parameters** for more information about WhereScape RED Parameters.

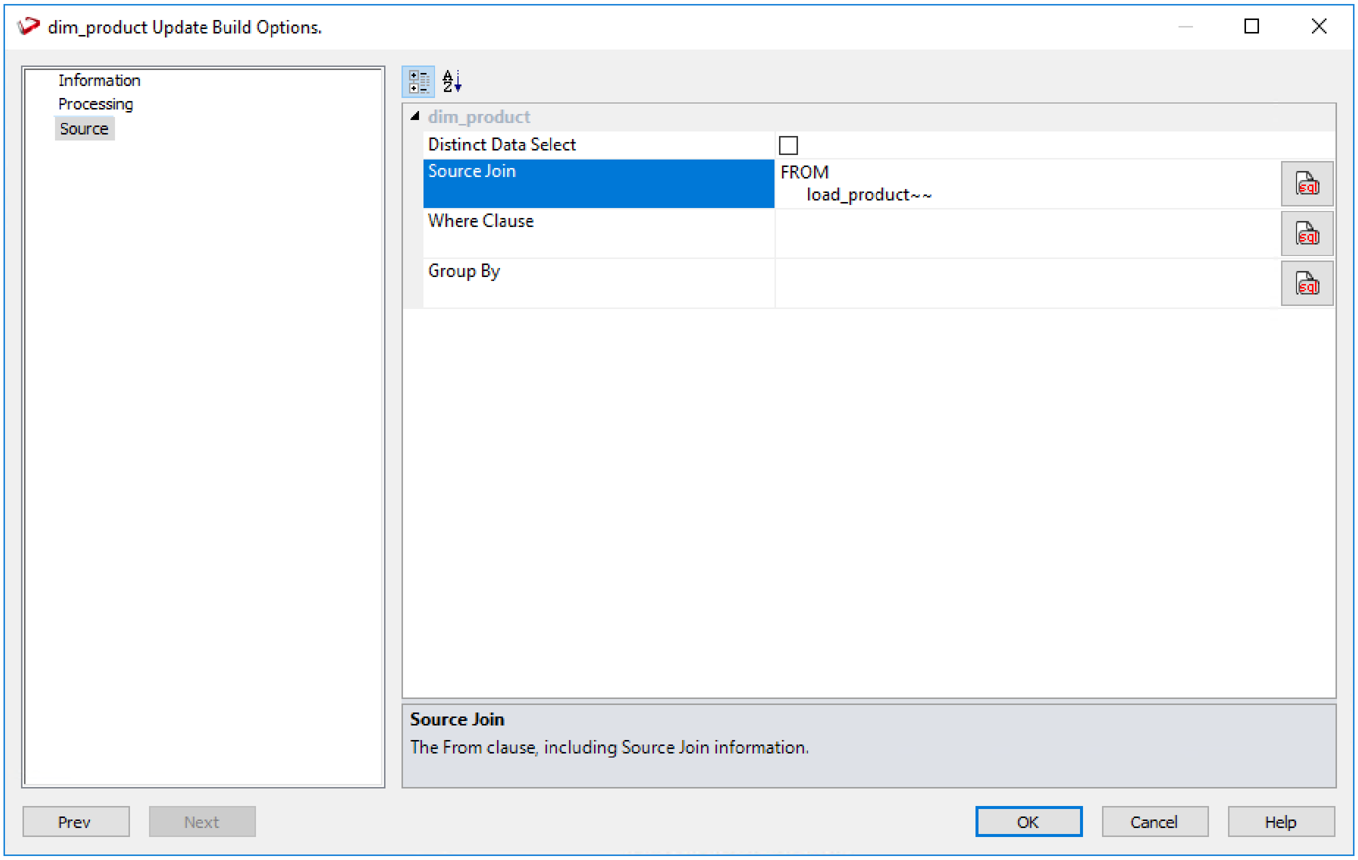
Fields	Description
Insert Zero Key Record	This option enables you to include the zero key (unknown record) for tables with an artificial key. The default for this field is set.
Include Initial Load insert	This option adds an additional insert statement to the update procedure that runs if the target Dimension Object is empty. The benefit of this is improved performance inserting into an empty table without performing any checks to see if rows already exist. The default for this field is FALSE (i.e. an initial insert statement is not added to the procedure).
Process by Batch	This field enables the user to select a column to break up the data being processed in a loop based on the distinct values in the field. The update procedure loops on this field and

Fields	Description
	performs the delete, update and/or insert for each value. If the field chosen is a date datatype (date, datetime or timestamp), then the user is able to specify yearly, monthly, daily or column level looping. The default for this field is False (do not do batch processing).
Batch Processing Field	Enables selecting a field to batch process on. If you select a date field you will have the ability to process by date part. If you select a join field to process by you can choose and attribute of that related table to group by.
Include Explicit Lock	Enables locking of the table to avoid concurrent transactions. Double click the **Lock Clause** field to open a window that enables you to create the lock clause to insert.
Delete before insert	This option enables a delete statement to be added to the update procedure before any update or insert statement. This is a particularly useful option for purging old data and for updates based on a source system batch number. The default for this field is No which automatically grays out the **Issue Warning if a Delete Occurs** and the **Delete Where Clause fields.** • **Issue Warning if a Delete occurs -** sets the procedure to a warning state if deletes occur. • **Delete Where Clause -** The delete where clause is appended to the generated delete statement to constrain the rows deleted.
Process Method	Enables selecting whether the table should be updated using an Insert/Update statement or a Merge statement.
Source Table Locking	Enables a locking request modifier to be specified for each source table. The specified locking request modifier is applied to each source table during generated update procedures. By default this is set to 'ACCESS' which locks each row being accessed, a blank entry will result in no locking clause in the generated procedure. This option may also be presented in a separate dialog
Include Insert Statement	The Include Insert Statement option includes an insert statement in the procedure to insert new rows in the Dimension. If this option is set, the **Insert New Rows Only** option is available. If this option is turned off, the update procedure will not contain an insert statement. The default for this field is set (i.e. an insert statement is included).
Insert New Rows only	The insert new rows only option uses change detection to work out what rows require inserting. • **New Row Identification Method -** Method used to identify that the records in the source are not currently recorded in the target table.
Insert Hint	Enter a database hint to be used in the INSERT statement. This is SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.
Insert New Rows Only	Uses change detection to work out what rows require inserting.
New Row Identification Method	Method used to identify that records in source are not currently recorded in the target table.
Existing Data Selection Hint	Database-compliant hint to be used for the existing data select statement.
Include Update Statement	Includes an update statement in the procedure to update changing rows in the Data Store. If this option is chosen, then the Update Changed rows only option is available.
Include Hint	Enter a database hint to be used in the INCLUDE statement. This is a SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.
Update Changed	Uses change detection to work out what rows require updating. Choosing this option,

Fields	Description
Rows Only	enables the **Change Row identification Method**.
Change Row Identification Method	Method used to identify that records in source have changed from what is currently recorded in the target table.
Existing Data Selection Hint	Database-compliant hint to be used for the existing data select statement.
Merge Hint	Enter a database hint to be used in the MERGE statement. SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.

Source tab

If only one source table was used, adding the **Business Key** and checking the above fields is enough to proceed, otherwise use the **Source** tab to **Join** the relevant tables.



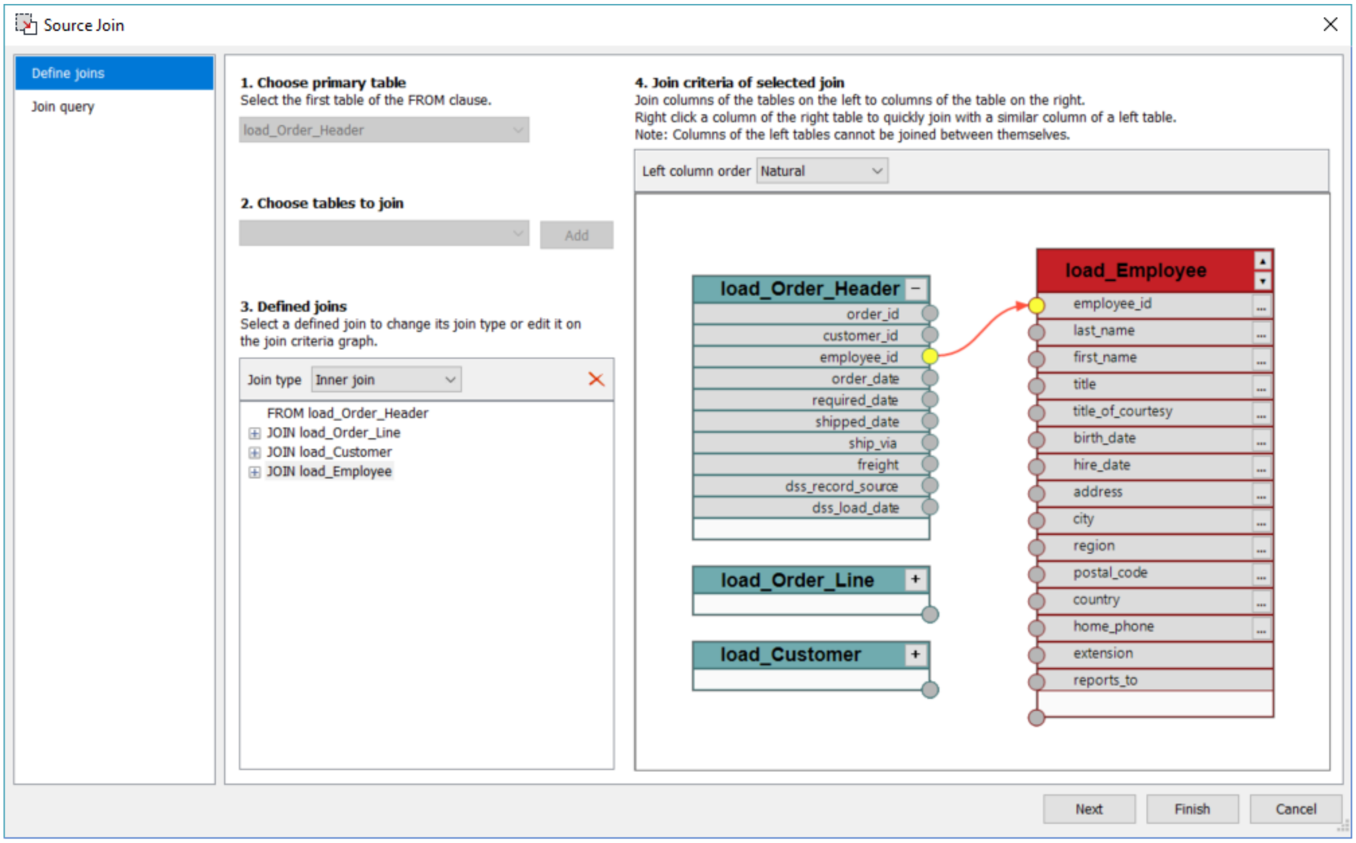
Distinct Data Select ☐
Source Join: FROM
    load_product~~
Where Clause
Group By

Source Join
The From clause, including Source Join information.

Fields	Description
Distinct Data Select	Ensures duplicate rows are not added to the Dimension. This is achieved by the word DISTINCT being added to the source select in the update procedure. The default for this field is not set.
Source Join	The From clause, including Source Join information. See example below for Joining multiple source tables.
Where Clause	The Where Clause. Use as a filter to extract only the necessary records that fulfill a specified criteria.
Group By	The Group by clause. Use in collaboration with the SELECT statement to arrange identical data into groups.

Joining multiple source tables

WhereScape RED provides a wizard that enables you to graphically generate the join clause for an object with multiple source tables. The **Source Join** wizard enables you to specify the primary (driving) table and the source tables to join. A graphical representation of the tables and the join criteria is displayed by the wizard which you can manipulate to change the defined joins.



Source Join

1. Choose primary table
Select the first table of the FROM clause.

load_Order_Header

2. Choose tables to join

3. Defined joins
Select a defined join to change its join type or edit it on the join criteria graph.

Join type: Inner join

```
FROM load_Order_Header
⊞ JOIN load_Order_Line
⊞ JOIN load_Customer
⊞ JOIN load_Employee
```

4. Join criteria of selected join
Join columns of the tables on the left to columns of the table on the right.
Right click a column of the right table to quickly join with a similar column of a left table.
Note: Columns of the left tables cannot be joined between themselves.

Left column order: Natural

The **Source Join** wizard guides you in generating the Join Clause through the following steps:

1. Choose Primary Table

   The first source table encountered is the default primary (driving) table which can be changed before any joins are defined.

2. Choose Tables to Join

   Once the primary table is set, the remaining source tables are listed in this drop down. Select a table to join and then click **Add**. The two tables are added to pane 4 and the basis for the join is displayed in pane 3.

3. Defined Joins

   The join type defaults to **Inner** join but this can be changed from the drop down.

4. Join criteria of selected join

   The column(s) used to join the two tables can be joined manually by clicking the corresponding join column connection point from the left-hand table and then dragging the line that appears to the connection point of the required column in the table on the right.

   Alternatively, if a column with a matching or similar name exists in the left hand table(s) then a context menu is provided on the columns on the right-hand table. You can right click on the column to display the context menu and then select a column from the list.

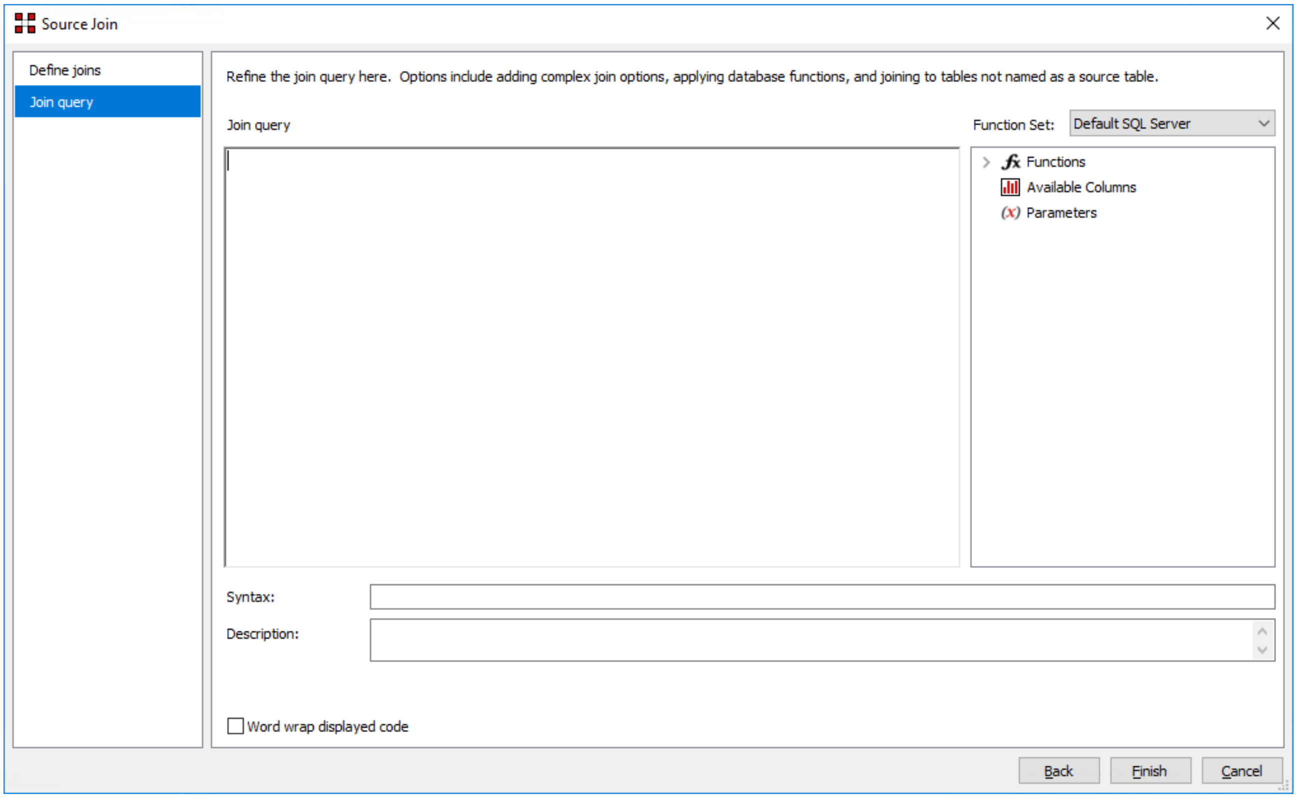
Tip

To aid manual column join, you can expand or collapse columns in the left-hand tables and toggle between

Tip

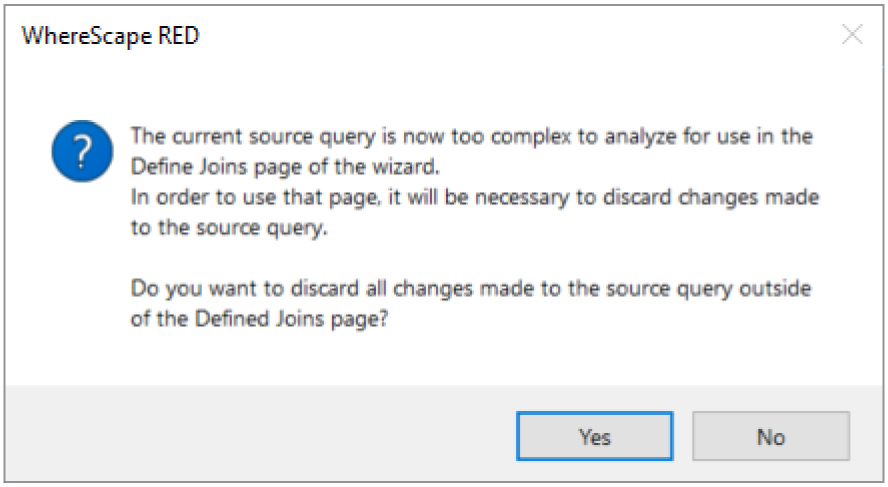
natural and alphabetic column ordering.

5. Once the join column(s) for the table pair are defined, then any additional source tables can be added from the drop-down list on step 2 and the above column mapping process repeat to join this table.
   This process is repeated until all source tables are joined.
6. Clicking the **Next** button or the **Join query** tab progresses to the join editor window, where custom joins, database functions, etc. can be added.


Source Join
Define joins
Join query
Refine the join query here. Options include adding complex join options, applying database functions, and joining to tables not named as a source table.
Join query
Function Set:
Default SQL Server
Functions
Available Columns
Parameters
Syntax:
Description:
□ Word wrap displayed code
Back
Finish
Cancel

Note

If RED is unable to parse a join statement, the user cannot navigate back to the **Define joins** tab and the following warning is displayed:


WhereScape RED

The current source query is now too complex to analyze for use in the Define Joins page of the wizard.
In order to use that page, it will be necessary to discard changes made to the source query.

Do you want to discard all changes made to the source query outside of the Defined Joins page?

Change Detection tab

For Slowly Changing Dimension columns. If the dimension was defined as a Slowly Changing Dimension, the **Change Detection** tab is displayed in the table **Update Build Options** screen.

Note:

For custom database targets, you can set the initial default values for the DSS columns that are used in the procedure generation for slowly changing dimensions.
Please refer to the section **DSS Columns for Custom Targets** for details. The default values set are populated in the corresponding fields of the **Change Detection** tab.

A change detection field(s) must be selected for this Dimension type.

Fields	Description
Change Detection Fields	Click the ellipsis button to select the change detection fields that are required for the dimension
Null Support	If the Add Null support is selected, the change detect column management caters for null values in any change detect columns. If this option is not selected and null values are present, errors may occur running the update procedure. The default for this value is ON (i.e. null values are catered for).
Update Current Records Only	The update current record only option only applies changes to non-change detect columns on the current record. If this option is not selected, all past (non-current) rows will also be updated to reflect changes to non-change detect columns. The default for this value is ON (i.e. only the current record is updated).
Reset Dates to Initial Values	Resets dss_start date and dss_end_date date values to original values.
Start Date for Initial Member	The start date for initial member field contains the start date for the first version of a particular business key. The value should be specified in an appropriate form, taking into account the default date format in the databases. The date may need to be explicitly cast to the current data type. The default value provided is usually cast to the correct database and can be treated as a template. The default for this field is 1 January 1900.
End Date for Current Member	The end date for current member field contains the end date for the current version (the row with a current flag of Y and the maximum version number) of a particular business key. The value must be specified in an appropriate form, taking into account the default date format in the databases. The date may need to be explicitly cast to the current data type. The default value provided is usually cast to the correct database and can be treated as a template. The default for this field is 31 December 2999.
Start Date for New Member Entry	The start date for new member entry field contains the start date for any subsequent rows added to the history table (not the first row for a particular business key i.e. not version 1). The value must be specified in an appropriate form, taking into account the default date format in the databases. The date may need to be explicitly cast to the current data type. The default value provided is usually cast to the correct database and can be treated as a template. The default for this field is the current date and time.
End Date for Expiring Member Entry	The end date for the expiring member entry field contains the end date for any rows updated no longer to no longer be the current row in the history table (i.e. rows that are replaced by a new current row). The value must be specified in an appropriate form, taking into account the default date format in the databases. The date may need to be explicitly cast to the current data type. The default value provided is usually cast to the correct database and can be treated as a template. The default for this field is the current date and time less an arbitrary small amount (for SQL Server this is 0.00000005 of a day, or about 4 thousandth of a second).

Dimension Artificial Keys

The artificial (surrogate) key for a dimension is set via an identity column in SQL Server.

This artificial key normally, and by default, starts at one and progresses as far as is required.

A WhereScape standard for the creation of special rows in the dimension is as follows:

Key value	Usage
1 upwards	The normal dimension artificial keys are numbered from 1 upwards, with a new number assigned for each distinct dimension record.
0	Used as a join to the dimension when no valid join existed. It is the normal convention in the WhereScape generated code that any dimension business key that either does not exist or does not match is assigned to key 0.
-1 through -9	Used for special cases. The most common being where a dimension is not appropriate for the record. For example, we may have a sales system that has a promotion dimension. Not all sales have promotions. In this situation, it is best to create a specific record in the dimension that indicates that a fact table record does not have a promotion. The stage table procedure would be modified to assign such records to this specific key. A new key is used rather than 0 as we want to distinguish between records that are invalid and not appropriate.
-10 backward	Pseudo records. In many cases we have to deal with different granularities in our fact data. For example, we may have a fact table that contains actual sales at a product SKU level and budget information at a product group level. The product dimension only contains SKU based information. To be able to map the budget records to the dimension, we need to create these pseudo keys that relate to product groups. The values -10 and backwards are normally used for such keys. A template called 'Pseudo' is shipped with WhereScape RED to illustrate the generation of these pseudo records in the dimension table.

Dimension Initial Build

The initial population of a dimension with data can be achieved by generating a custom procedure and then use the right-click context menu to select the **Execute Custom Procedure via Scheduler** option.

The dimension should be analyzed once the custom procedure is completed so that the database query optimizer can make use of the indexes.

For smaller dimensions (e.g. less than 500,000 rows) run the normal **Update** procedure against an empty dimension table. There is however a danger in this action in that the query optimizer will not have any information on the table, and hence will do a full table pass when adding each new row. For a very small dimension this will not be an issue, but it will rapidly become a major problem as the row count increases.

The problem with the initial update is that the database does not know to use the index that has been created for the business key, and hence does a full table pass when attempting to update/insert a record.

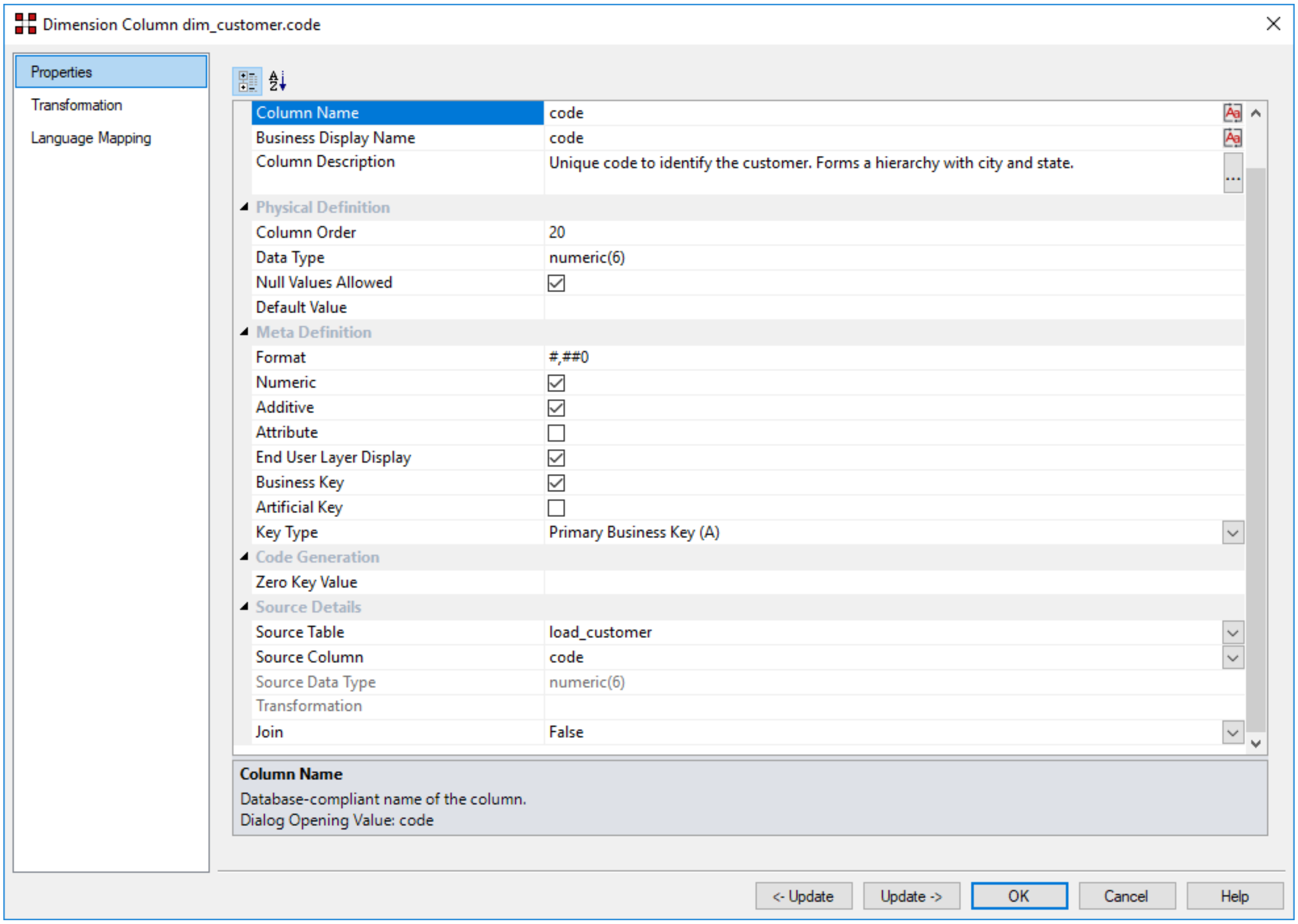
Dimension Column Properties

Each dimension column has a set of associated properties. The definition of each property is described below:

Tip:

If a database table's definition is changed in the metadata, then the table needs to be altered in the database. Use the **Validate > Validate Table Create Status** to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table, through a pop-up menu option from the validated table name. See the example below.

A sample **Properties** screen is as follows:



Column Name	code
Business Display Name	code
Column Description	Unique code to identify the customer. Forms a hierarchy with city and state.
Physical Definition	
Column Order	20
Data Type	numeric(6)
Null Values Allowed	☑
Default Value	
Meta Definition	
Format	#,##0
Numeric	☑
Additive	☑
Attribute	□
End User Layer Display	☑
Business Key	☑
Artificial Key	□
Key Type	Primary Business Key (A)
Code Generation	
Zero Key Value	
Source Details	
Source Table	load_customer
Source Column	code
Source Data Type	numeric(6)
Transformation	
Join	False

Column Name
Database-compliant name of the column.
Dialog Opening Value: code

Tip:

The two special update keys enables you to update the column and step either forward or backward to the next column's properties.
ALT-Left Arrow and **ALT-Right Arrow** can also be used instead of the two special update keys.

General

Options	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** The case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence of each conversion.
Business Display Name / Column Title	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note**

Options	Description
	The case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence of each conversion.
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example, if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col** . This field is also stored as a comment against the column in the database.

Physical Definition

Options	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. See the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.

Meta Definition

Options	Description
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.
Numeric	Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Additive	Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end

<table>
<tr><th>Options</th><th>Description</th></tr>
<tr><td></td><td>user layer. The use of this field is not relevant unless it can be utilized by the end user tools.</td></tr>
<tr><td><b>Attribute</b></td><td>Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or a invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables.
This check box does not affect the physical table definition, but rather provides input to the view <b>ws_admin_v_dim_col</b> which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.</td></tr>
<tr><td><b>End User Layer Display</b></td><td>Indicates whether the table column is available/visible to end users. If set the documentation will include the column in the glossary and in the user documentation.
It is also used to decide what columns appear in the view <b>ws_admin_v_dim_col</b>. Typically columns such as the artificial key would not be enabled for end user display.</td></tr>
<tr><td><b>Business Key</b></td><td>Indicates whether the column is part of the BUSINESS key, which is defined during the update procedure build. [Normally maintained automatically]. Multiple columns can form the primary business key.</td></tr>
<tr><td><b>Artificial Key</b></td><td>Indicates whether the column is the generated artificial/surrogate key (unique identifier) for the table. Only one artificial key per table is supported. [Normally maintained automatically].</td></tr>
<tr><td><b>Key Type</b></td><td>Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested.
The supported values are:
<table>
<tr><th>Key type</th><th>Meaning</th></tr>
<tr><td><b>0</b></td><td>The artificial key. Set when the key is added during drag and drop table generation.</td></tr>
<tr><td><b>1</b></td><td>Component of all business keys. Indicates that this column is used as part of any business key. For example: By default the dss_source_system_key is added to every dimension table. It is considered part of any lookup on that table and has the key type set to 1. Set when the column is added during drag and drop table generation.</td></tr>
<tr><td><b>2</b></td><td>Indicates that this column is a dimensional join. Used on fact tables to indicate the dimension keys. Results in an index being built for the column. Set during the update procedure generation for a fact table, based on information from the staging table.</td></tr>
<tr><td><b>3</b></td><td>Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation.</td></tr>
<tr><td><b>4</b></td><td>Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation.</td></tr>
<tr><td><b>5</b></td><td>Start date of a date ranged dimension. Used on dimension tables to indicate that the column is defined as the starting date for a source system date ranged dimension. Forms part of the business key. Set during the dimension creation.</td></tr>
<tr><td><b>6</b></td><td>End date of a date ranged dimension. Used on dimension tables to indicate that the column is defined as the ending date for a source system date ranged dimension. Forms part of the business key. Set during the dimension creation.</td></tr>
<tr><td><b>A</b></td><td>Indicates that the column is part of the primary business key. Set whenever a</td></tr>
</table>
</td></tr>
</table>

Options	Description
	business key is defined as part of an update procedure generation.
	B-Z Indicates that the column is part of a secondary business key. Only used during index generation and not normally set.

Code Generation

Options	Description
Zero Key Value	Determines the value populated for the column in the **Invalid Join** or **Unknown record**. By default, NULL is used when a value is not specified. All dimensions that use standard WhereScape RED generated procedures have a row with an artificial key of zero. This row is used to link to the fact records when no match on the dimension can be found. For example we may have some sales records that come through without a product defined. WhereScape RED will be default associate these fact records with the zero key product dimension entry. So, we might set the zero key value to 'Unknown product' for the name column in the product dimension.

Source Details

Options	Description
Source Table	Identifies the source table where the column's data comes from. This source table is normally a load table, or a dimension table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the **Source strategy** field. This field is used when generating a procedure to update the stage table. It is also used in the track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a load table column, which in turn may have been a transformation or the combination of multiple columns. This may also be a dimensional key where a dimension is being joined.
Source Data Type	Identifies the source column's data type. [Read-only].
Transformation	Refer to **Dimension Column Transformations** for details. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source table** and **Source column** fields will provide the dimension table's side of the join. The options for this field are: False, True, Manual and Pre Join. • Setting this field to Manual changes the way the Dimension table is looked up during the Stage table update procedure build. It enables you to join the dimension manually in the Cursor mapping dialog (where the 'Where' clause is built). The usual dialog for matching the dimension business key to a column or columns in the Stage table is not displayed, if this option is enabled. • Setting this field to Pre Join activates the **Pre Join Source Table** field and enables you to select a table from the drop-down list.

Changing a Column Name

If the **Column name** or **Data type** is changed for a column then the metadata will differ from the table as recorded in the database.

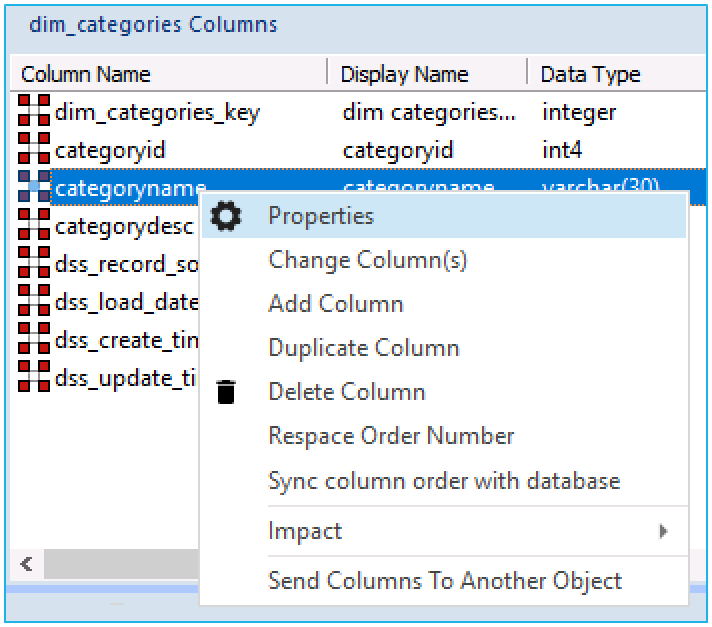
- Use the **Validate > Validate Table Create Status** menu option or the right-click menu to compare the metadata to the table in the database.

- A right-click menu option of **Alter Table** is available when positioned on the table name after the validate has completed. This option will alter the database table to match the metadata definition.

For example: Analysis Services does not like **name** as a column name.

For dim_customer it will therefore be necessary to change the column name from **name** to **cname**.

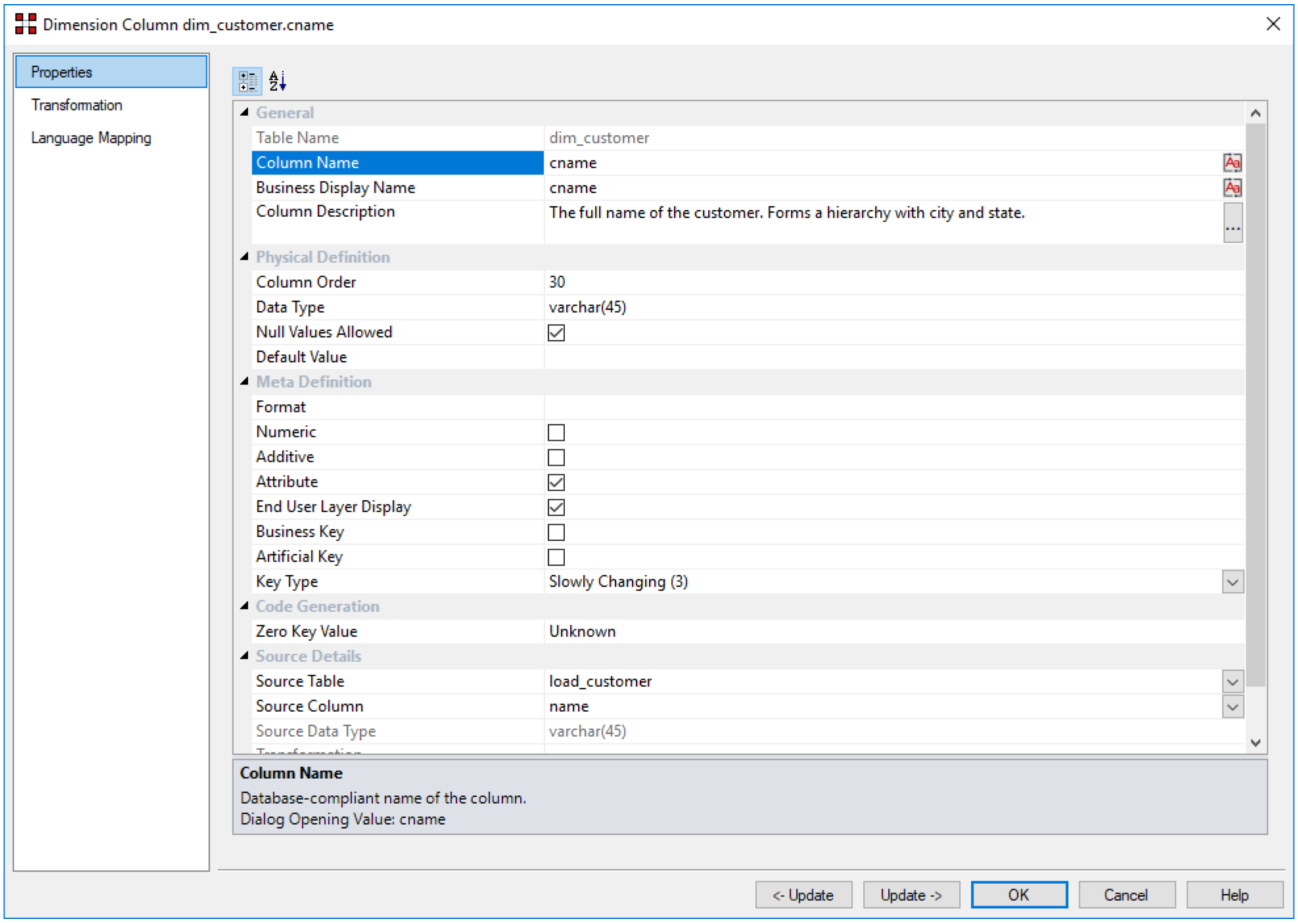
1. Click on the **dim_customer** object in the left pane to display the dim_customer columns in the middle pane.
2. When positioned on the column **name** in the middle pane, right-click and select **Properties** from the drop-down menu.



dim_categories Columns

Column Name	Display Name	Data Type
dim_categories_key	dim categories...	integer
categoryid	categoryid	int4
categoryname	categoryname	varchar(30)
categorydesc		
dss_record_so		
dss_load_date		
dss_create_tin		
dss_update_ti		

3. Change the column name from **name** to **cname** as shown below. Click **OK** to leave the **Properties** window.

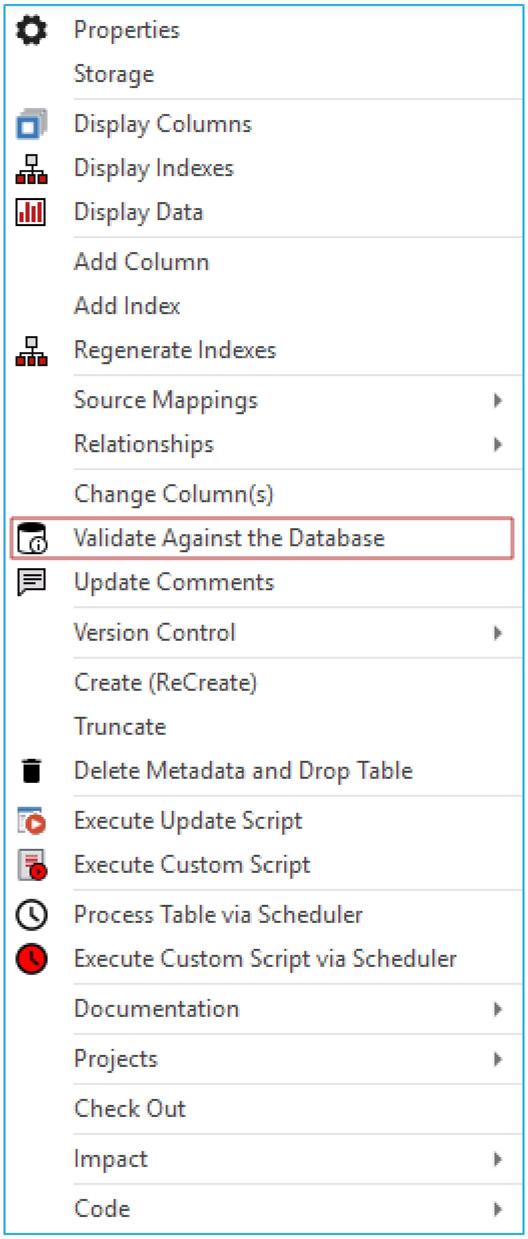


General	
Table Name	dim_customer
Column Name	cname
Business Display Name	cname
Column Description	The full name of the customer. Forms a hierarchy with city and state.
Physical Definition	
Column Order	30
Data Type	varchar(45)
Null Values Allowed	☑
Default Value	
Meta Definition	
Format	
Numeric	☐
Additive	☐
Attribute	☑
End User Layer Display	☑
Business Key	☐
Artificial Key	☐
Key Type	Slowly Changing (3)
Code Generation	
Zero Key Value	Unknown
Source Details	
Source Table	load_customer
Source Column	name
Source Data Type	varchar(45)

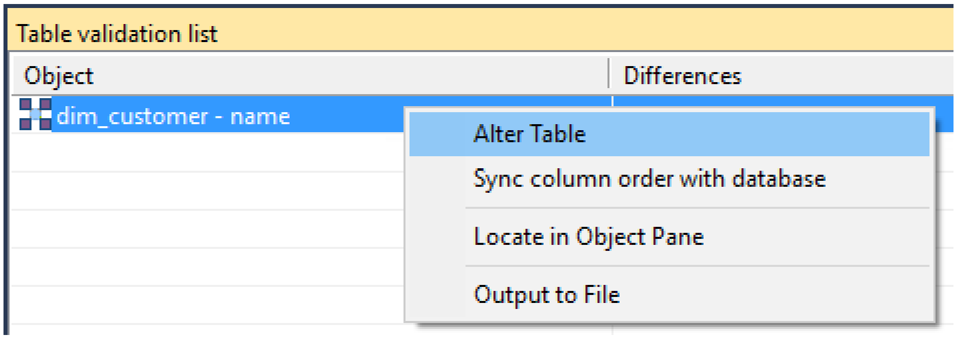
Column Name

Database-compliant name of the column.
Dialog Opening Value: cname

4. Right-click the **dim_customer** object in the left pane and select **Validate against Database**.



5. The results in the middle pane shows that the metadata has been changed to **cname** while the column name in the database is still **name**.
6. Right-click **dim_customer** in the middle pane and select **Alter table** from the context menu.



7. A warning appears, displaying the table and column name to be altered. Select **Alter Table**.

8. A dialog appears confirming that dim_customer has been altered. Click **OK**.

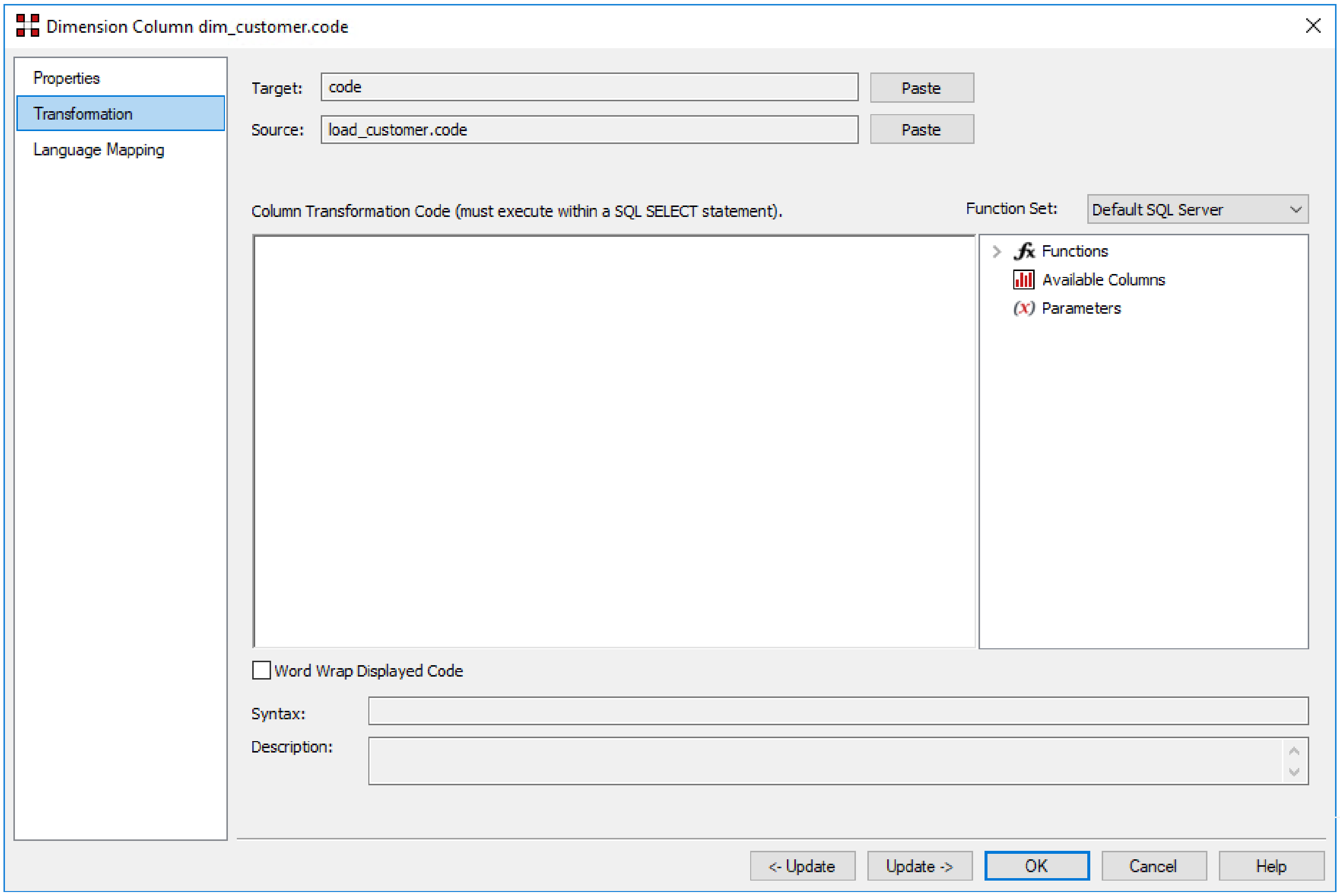
Dimension Column Transformations

Each Dimension table column can have a transformation associated with it. The transformation is included in the generated procedure and is executed as part of the procedure update.

The transformation must therefore be a valid SQL construct that can be included in a **Select** statement. For example, we could have a transformation of 'load_order_line.quantity * 0.125' to calculate a tax column of 12.5%.

Click the **Transformation** tab to enter a transformation code.

The transformation screen is as follows:



Dimension Column dim_customer.code

Target: code

Source: load_customer.code

Column Transformation Code (must execute within a SQL SELECT statement).

Function Set: Default SQL Server

☐ Word Wrap Displayed Code

Syntax:

Description:

Notes:

Transformations only take effect when the procedure is re-generated.

Microsoft Analysis Services 2005+ Tabular Mode Tables: For Tabular Mode table column transformations, **Default DAX** is the only applicable Function Set for **after load** transformations.

Refer to **Transformations** for details.

Snowflake

Snowflake schemas normalize dimensions to eliminate redundancy; that is, the dimension data has been grouped into multiple tables instead of one large table.

For example, a product dimension table in a star schema might be EDW 3NF into a products table, a product_category table, and a product_manufacturer table in a snowflake schema.

While this saves space, it increases the number of dimension tables and requires more foreign key joins.

The result is more complex queries and reduced query performance.

Creating a Snowflake

A snowflake dimensional structure is supported by WhereScape RED.

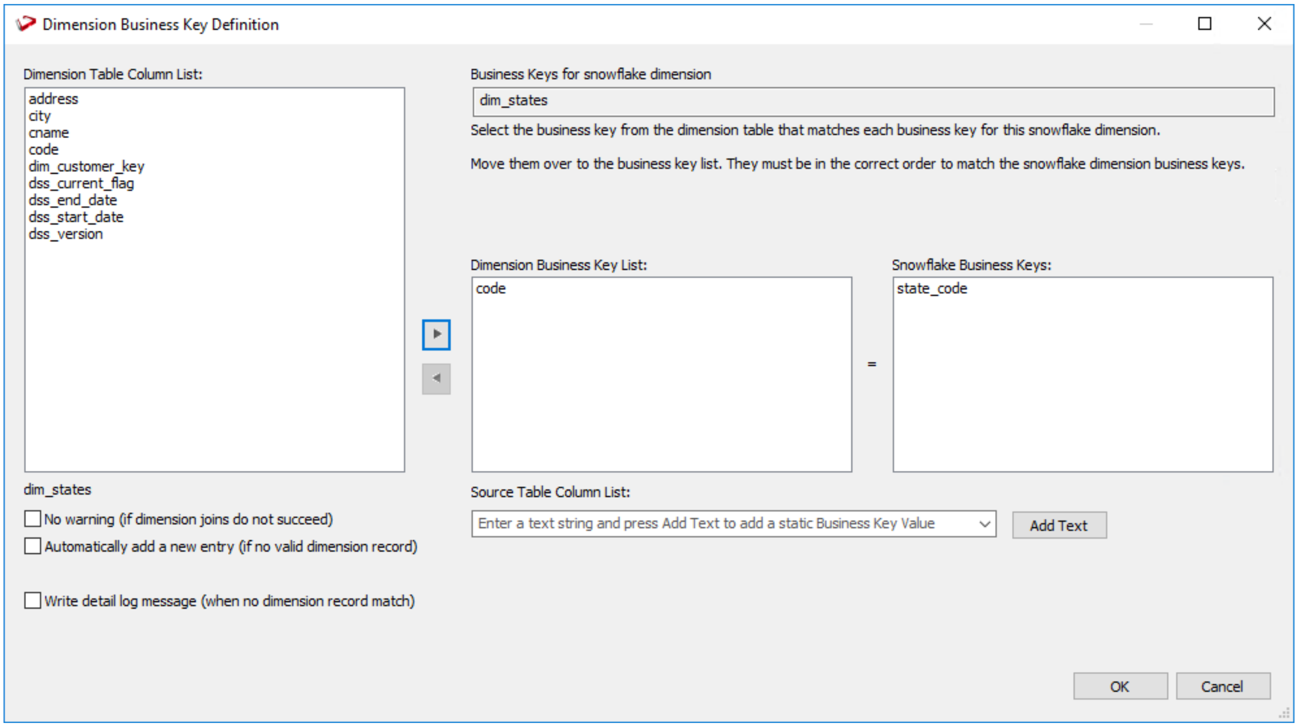
A snowflake can be created for EDW 3NF or partially EDW 3NF dimension tables, by including the surrogate key of the parent dimension in the child dimension.

In the example below, the dim_state table represents the parent dimension.

The column dim_state_key is added to the child dimension dim_customer. Any fact tables that include the dim_customer dimension will inherently have a link to the dim_state dimension.

The process for creating a snowflake is as follows:

1. Build both dimensions (see previous sections).
2. Expand **dimensions** in the left pane.
3. Click on the child dimension table in the left pane to display its columns in the middle pane.
4. Browse the data warehouse connection in the right pane.
5. Expand the parent dimension table in the right pane.
6. Drag the surrogate key of the parent dimension table from the right pane to the child dimension's column list in the middle pane.
7. **Create/Recreate** the child dimension.
8. Rebuild the child dimension's update procedure.
9. A screen appears asking for the business key column(s) in the child dimension that matches the business key for the parent dimension:



Dimension Business Key Definition

Dimension Table Column List:
address
city
cname
code
dim_customer_key
dss_current_flag
dss_end_date
dss_start_date
dss_version

Business Keys for snowflake dimension
dim_states

Select the business key from the dimension table that matches each business key for this snowflake dimension.

Move them over to the business key list. They must be in the correct order to match the snowflake dimension business keys.

Dimension Business Key List: code = Snowflake Business Keys: state_code

dim_states

☐ No warning (if dimension joins do not succeed)

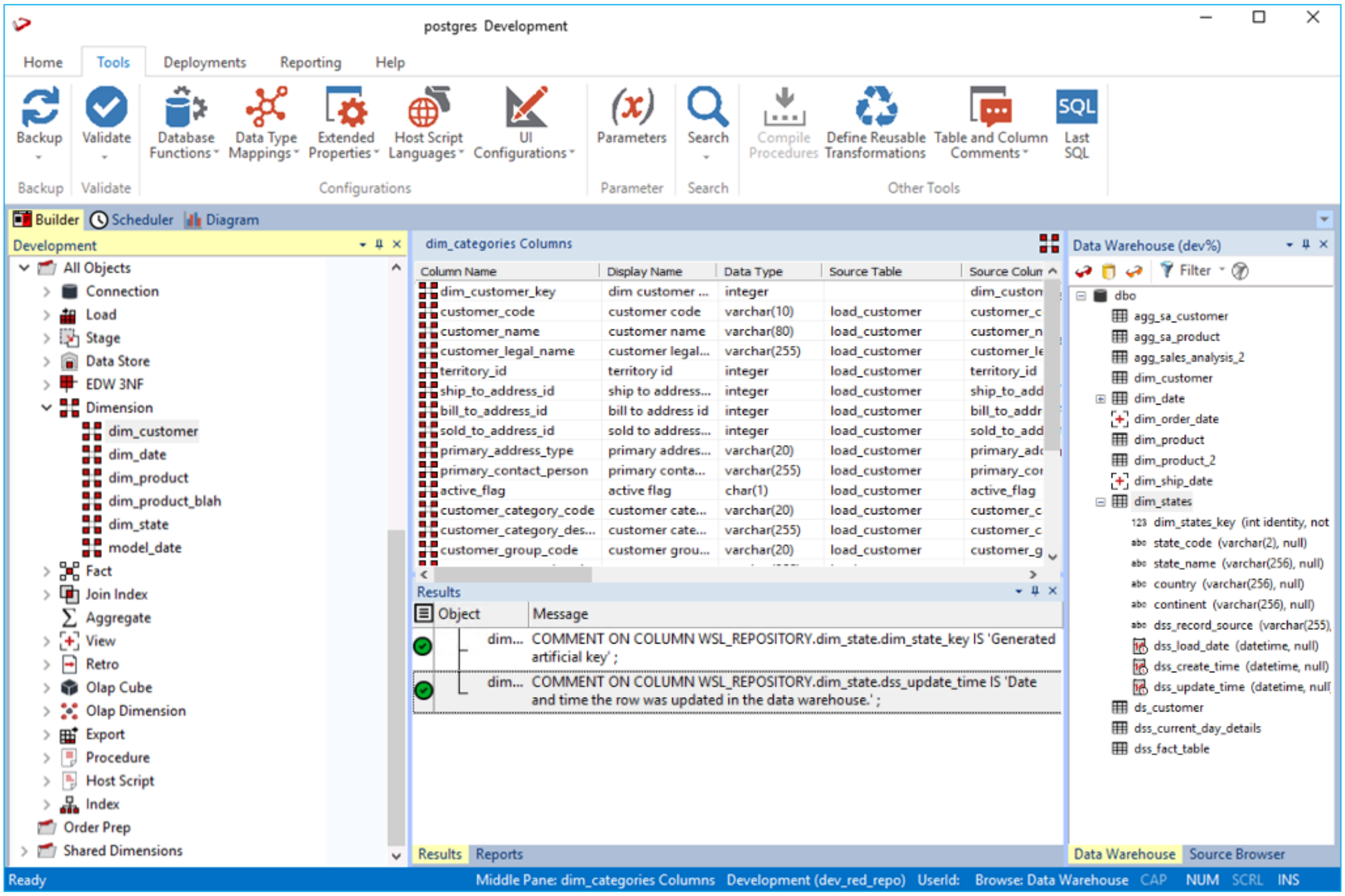
☐ Automatically add a new entry (if no valid dimension record)

☐ Write detail log message (when no dimension record match)

Source Table Column List: Enter a text string and press Add Text to add a static Business Key Value

10. Add the business key column(s) and then click **OK**.

The WhereScape RED screen should look like this:



Column Name	Display Name	Data Type	Source Table	Source Colum
dim_customer_key	dim customer ...	integer		dim_custom
customer_code	customer code	varchar(10)	load_customer	customer_c
customer_name	customer name	varchar(80)	load_customer	customer_n
customer_legal_name	customer legal...	varchar(255)	load_customer	customer_le
territory_id	territory id	integer	load_customer	territory_id
ship_to_address_id	ship to address...	integer	load_customer	ship_to_add
bill_to_address_id	bill to address id	integer	load_customer	bill_to_addr
sold_to_address_id	sold to address...	integer	load_customer	sold_to_add
primary_address_type	primary addres...	varchar(20)	load_customer	primary_adc
primary_contact_person	primary conta...	varchar(255)	load_customer	primary_cor
active_flag	active flag	char(1)	load_customer	active_flag
customer_category_code	customer cate...	varchar(20)	load_customer	customer_c
customer_category_des...	customer cate...	varchar(255)	load_customer	customer_c
customer_group_code	customer grou...	varchar(20)	load_customer	customer_g

Object	Message
dim...	COMMENT ON COLUMN WSL_REPOSITORY.dim_state.dim_state_key IS 'Generated artificial key' ;
dim...	COMMENT ON COLUMN WSL_REPOSITORY.dim_state.dss_update_time IS 'Date and time the row was updated in the data warehouse.' ;

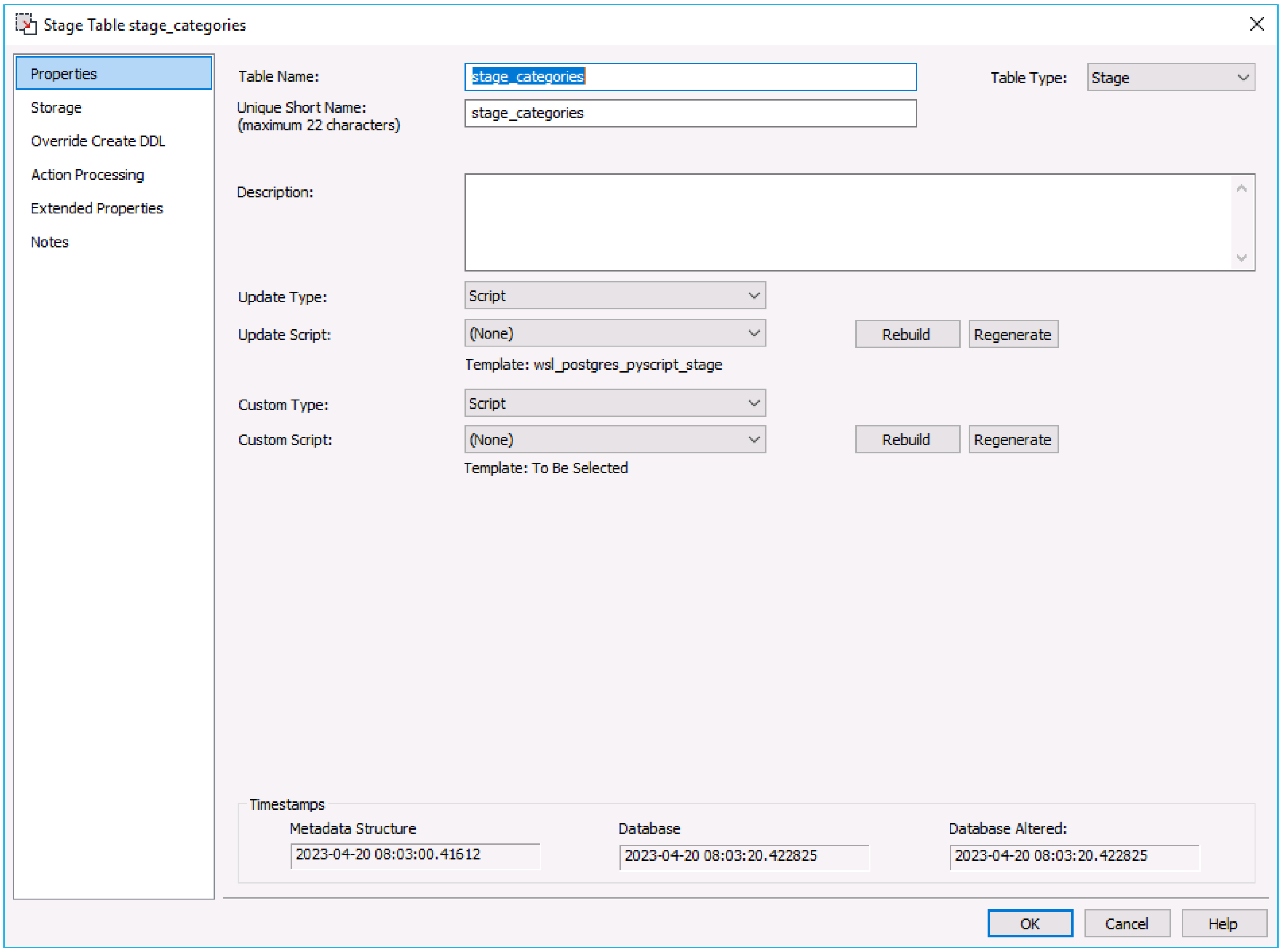
Table Properties

Various properties can be set on all table objects in WhereScape RED. The screens available in the Table Properties window depends on the object type selected and can be a subset of:

- Properties
- Storage
- Override Create DDL
- Source
- Documentation Fields
- Notes

Properties

The fields available in the **Properties** screen depends on the Object Type selected. More specific information is available in the respective sections that describe each RED object type.



Stage Table stage_categories

Table Name: stage_categories

Unique Short Name: (maximum 22 characters) stage_categories

Table Type: Stage

Description:

Update Type: Script

Update Script: (None)

Template: wsl_postgres_pyscript_stage

Custom Type: Script

Custom Script: (None)

Template: To Be Selected

Timestamps

Metadata Structure	Database	Database Altered:
2023-04-20 08:03:00.41612	2023-04-20 08:03:20.422825	2023-04-20 08:03:20.422825

Fields	Description
Table Name	The user-name of the selected table.
Unique Short Name	The short name is derived from the Table Name and is used internally by RED.
Table Type	The drop down provides a list of the available sub types for the selected table type.
Description	Optional, free text.

Fields	Description
Update Type	**Note:** This field is only displayed if the table object is created in a custom database target, an additional processing option is available—script-based processing. Refer to **Script Templates for Custom Database Table Objects** for details. Enables you to select between **Procedure** or **Script** update type. Selecting **Script** displays the **Update Script** drop-down field which enables you to select a script to use from the list of available script templates.
Update Procedure	The name of the procedure to be used when updating the table. **Note:** RED displays the name of the previously used update procedure template below the **Update Procedure** drop-down field by default, if the code generation is via a template.
Custom Type	Same as **Update Type** field above, this field is only displayed for table objects stored in a custom database target and provides option for **Procedure** or **Script** based custom processing.
Custom Procedure	The name of the procedure template to be used for custom update of the table. **Note:** RED displays the name of the previously used custom procedure template below the **Custom Procedure** drop-down field by default.
Edit	Edit the content of the displayed Update or Custom procedure.
Rebuild	Rebuild the defined Update or Custom Procedure. Refer to **Rebuilding Update Procedures** for details.
Regenerate	Regenerate the selected Update or Custom Procedure, using the responses to previously provided information.

Rebuilding tables using Update Procedures and Templates is applicable to the objects listed and described in the summary table, under the **Rebuilding Update Procedures > Generating an Update Procedure via a Template** section.

Rebuilding Update Procedures

The update procedure for a data warehouse object can be generated by RED using parameters provided by the user, or it can be generated using a prepared template. Templates appropriate to the table type must first be created before they can be selected and used to generate update procedures. The option to select a template is only available if RED detects the presence of an appropriate template.

Notes:

- Update procedures for Table objects are stored in the RED repository meta database. However if required, the location of stored procedures can be changed and also be distributed across different Targets. The location of stored procedures can be changed and locked via the **Procedure Properties** window.
  Refer to **Procedure Migration** for details.
- Update procedures for Data Vault and Source Mapping objects can only be generated using a template. Refer to **Data Vault Templates** and **Generating Update Procedures for Source Mapping Objects** respectively for details.

Generating an Update Procedure via a Template

If RED detects the presence of a template, the **Rebuild** button provides additional options, in a drop down list, for rebuilding the selected update procedure. Two additional rebuild options are provided:

- **Rebuild without template** - RED rebuilds the update procedure, but ignores any previously used template. RED prompts you for inputs as required for the rebuild.
- **Rebuild with template** - Prompts you to select from a list, a template that is appropriate to the current object type. Depending on the selected template, the user is prompted to provide responses that will be used by the parameters within the template.

When you click the **Rebuild** button, RED rebuilds the procedure using the last selected option as the default method. If a template has not been previously used, then RED prompts you for inputs as required for the rebuild.

Notes:

1. If a template has been previously used, RED uses this template by default when **Rebuild** is clicked. The name of this template is displayed below the **Update Procedure** field of the table **Properties** screen. The same applies to **Custom Procedures** that were generated via a template.
2. If you are using WhereScape RED on SQL Serve and you are licensed for the Star schema option, a template for Set based processing of Fact tables is available from the RED **Template** objects list pane.
3. During procedure generation, a message prompt is displayed if any source tables are missing in the **Source Join** property of the **Source** tab in the **Update Build Options** screen. Select from the following options:
   - **Yes** to return to the **Source Join** screen and edit the **Source Join** property.
   - **No** to continue with the procedure generation.
   - **Cancel** to cancel the procedure generation.

Storage

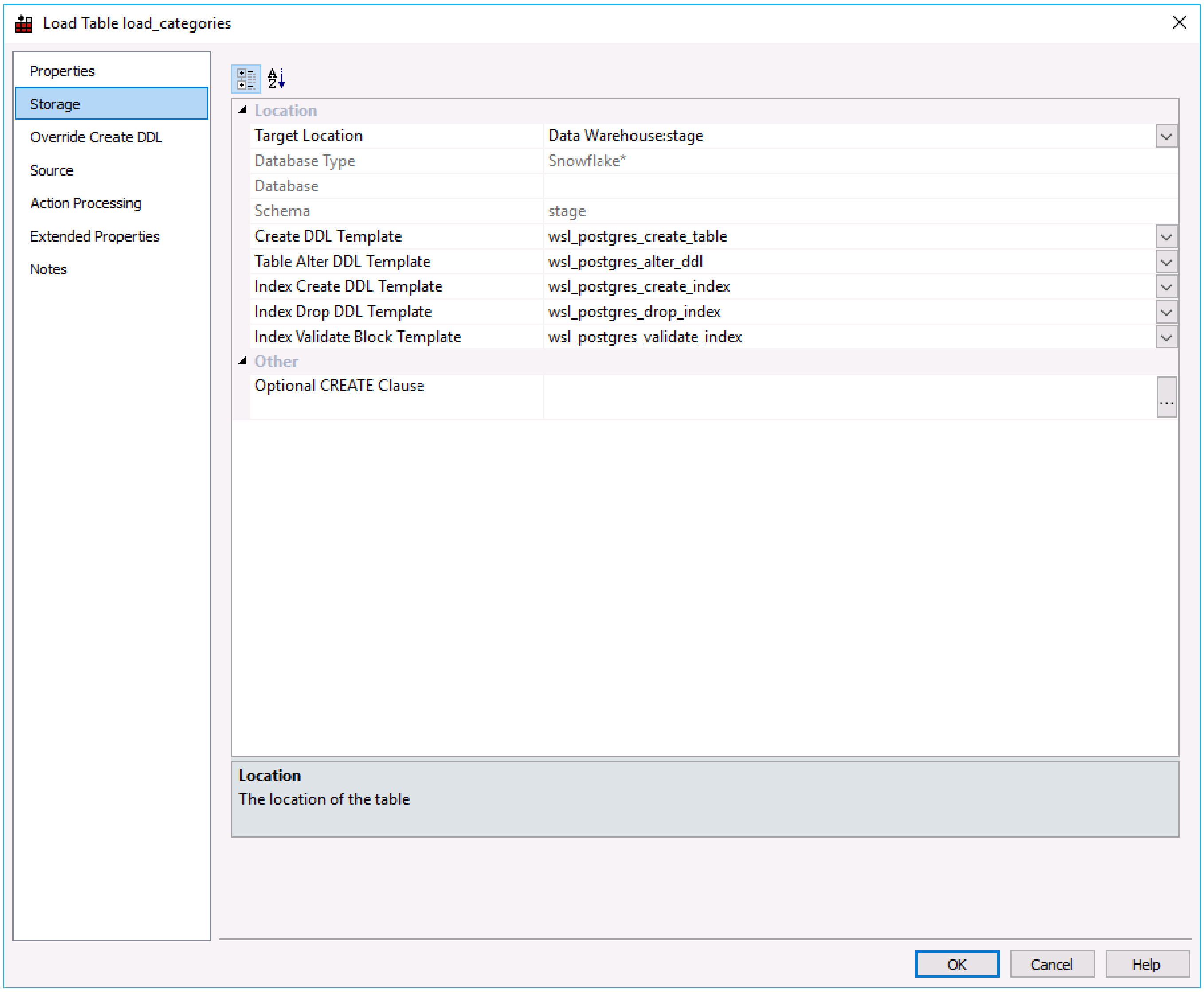
The **Storage** screen of the **Table Properties** window displays the options applicable for storing data in the associated RDBMS.

For a Custom Database example, refer to **Table Storage Screen - Custom Database** for details.

For information on changing storage locations for multiple tables at once, refer to **Bulk Table Storage Change** for details.

Table Storage Screen - Custom Database

Sample **Storage** screen for a **Custom Database** Table:



Location

Target Location	Data Warehouse:stage
Database Type	Snowflake*
Database	
Schema	stage
Create DDL Template	wsl_postgres_create_table
Table Alter DDL Template	wsl_postgres_alter_ddl
Index Create DDL Template	wsl_postgres_create_index
Index Drop DDL Template	wsl_postgres_drop_index
Index Validate Block Template	wsl_postgres_validate_index

Other

Optional CREATE Clause	

Location
The location of the table

Location

Fields	Description
Target Location	The target location that defines the path to the location for the table.
Database Type	The database type for a connection that is used for target data warehouse tables.
Create DDL Template	Optional. Specify the template to use for generating DDL for this table. Set to *None* to use default DDL.
Index Create DDL Template	Optional. Specify the template to use for generating DDL to create Indexes on this table. Set to *None* to use default DDL.
Index Drop DDL Template	Optional. Specify the template to use for generating DDL to drop Indexes on this table. Set to *None* to use default DDL.

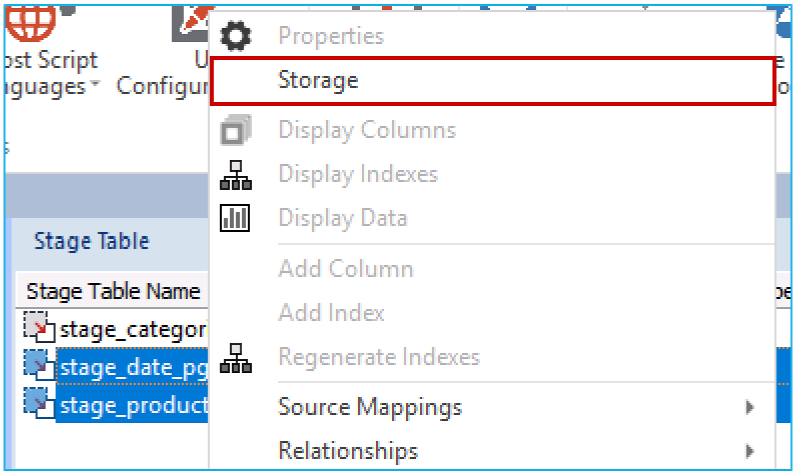
Other

Fields	Description
Optional CREATE Clause	Database-specific-and-compliant DDL to append to the generated CREATE TABLE statement.

Bulk Table Storage Change

Table Storage locations can be changed through the Storage tab on a table's Properties dialog but they can also be changed in bulk by using the following process:

1. Double-click on the desired object group in the left pane. This will display all the tables in that group in the middle pane.
2. Select the tables that you wish to change the storage for using standard Windows selection.
3. Right-click to bring up a menu and select **Storage**.



4. On the Target Location Selection dialog, select the desired **Target Location** for the change.
   - Double-click on the desired object group in the left pane. This will display all the tables in that group in the middle pane.
   - Double-click on the desired object group in the left pane. This will display all the tables in that group in the middle pane.
5. Follow the next dialogs to complete the bulk storage change.
6. Please note that **all procedures** from the affected tables will need to be **manually changed or regenerated** after a bulk storage change.

Warning

Please note that changing the Storage for Dimension and Fact tables need to be handled very carefully, because artificial key relationships between Dimension and Fact could become out of sync.
Recreating Fact tables and large Dimension tables might take a considerable amount of time.

Override Create DDL

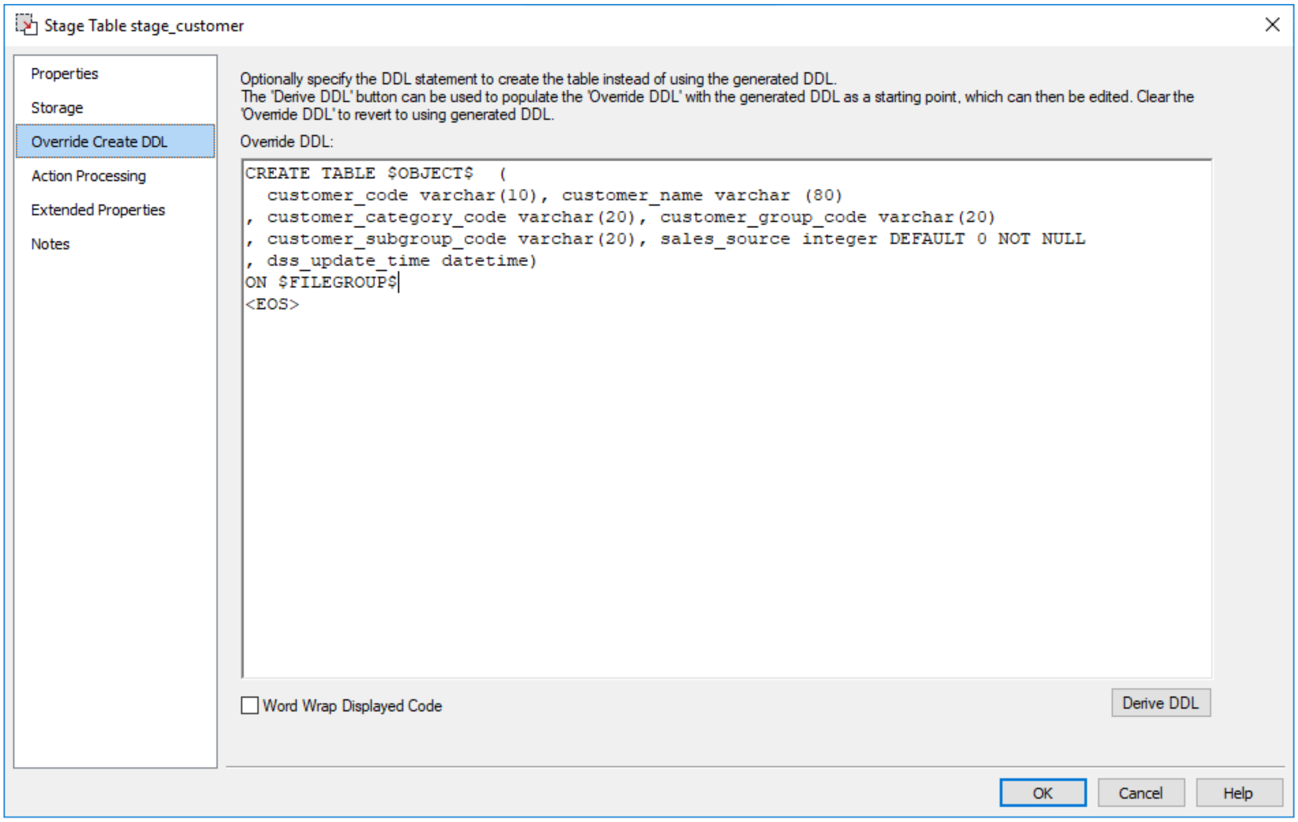
WhereScape RED enables you to create tables, using a specific DDL statement instead of the RED generated DDL.

You can override and edit the created DDL on load, stage, data store, EDW 3NF, dimension, fact, aggregate, user defined view and retro table's **Override Create DDL** tab.

Creating a table using a specific DDL statement:

1. If you are creating a **new** table, drag and drop the table from the right pane to the middle pane.
   - On the Properties dialog, click on the **Override Create DDL tab**.
   - Click on the **Derive DDL** button at the bottom of the dialog to get the generated DDL as your starting point.
   - Edit or clear the generated DDL and enter the desired DDL.
2. If you are **editing** an existing table, double click on the table object on the left pane to open the Properties dialog.

- Click on the **Override Create DDL** tab.
- Click on the **Derive DDL** button at the bottom of the dialog to get the generated DDL as your starting point.
- Edit or clear the generated DDL and enter the desired DDL.



Optionally specify the DDL statement to create the table instead of using the generated DDL.
The 'Derive DDL' button can be used to populate the 'Override DDL' with the generated DDL as a starting point, which can then be edited. Clear the 'Override DDL' to revert to using generated DDL.

Override DDL:

```
CREATE TABLE $OBJECT$  (
  customer_code varchar(10), customer_name varchar (80)
, customer_category_code varchar(20), customer_group_code varchar(20)
, customer_subgroup_code varchar(20), sales_source integer DEFAULT 0 NOT NULL
, dss_update_time datetime)
ON $FILEGROUP$
<EOS>
```

☐ Word Wrap Displayed Code

Note

The $OBJECT$ reference is auto translated to the fully qualified (schema/db.table name) at runtime.

- For DB2, $TABLESPACE$ and $INDEXSPACE$ get replaced at runtime with the storage defined values.
- For Greenplum, OWNER gets replaced at runtime with the storage defined values.

3. The "end of statement" indicator is <EOS> by default but can be configured in **Tools > Options > Code Generation>General**.

Tip

To revert to RED deriving the original generated DDL at runtime, leave the Override DDL box blank or clear out the contents.
Clicking the **Derive DDL** button pops-up a warning message asking if you want to replace the current

Source

The **Source** tab is only available for Load tables. More information is available in the Loading Data chapter, or more specifically:

- **Database Link Load - Source Screen**
- **Native ODBC Based Load - Source Screen**
- **Flat File Load - Source Screen**

Documentation Fields

Each table has a set of associated documentation fields. These are used to store descriptive metadata about the tables and how they are used.

The contents of each field appears in the generated documentation. Their order and placement can be customized in the generated documentation using the **Tools > Options** menu.

The following fields are available to store additional informational metadata that will appear in the generated WhereScape RED documentation:

- Purpose
- Concept
- Grain
- Examples
- Usage

Note

The names of these fields are completely arbitrary and can be changed in the generated documentation in the **Tools > Options > Documentation** menu.
By default, these fields are not enabled for load and stage tables, but can be enabled using the **Tools > Options > Documentation** menu.

Up to 4000 characters of information can be stored in each field.

Some or all of these fields can be removed from the documentation via the **Tools > Options > Documentation** menu.

Documentation Fields Screen

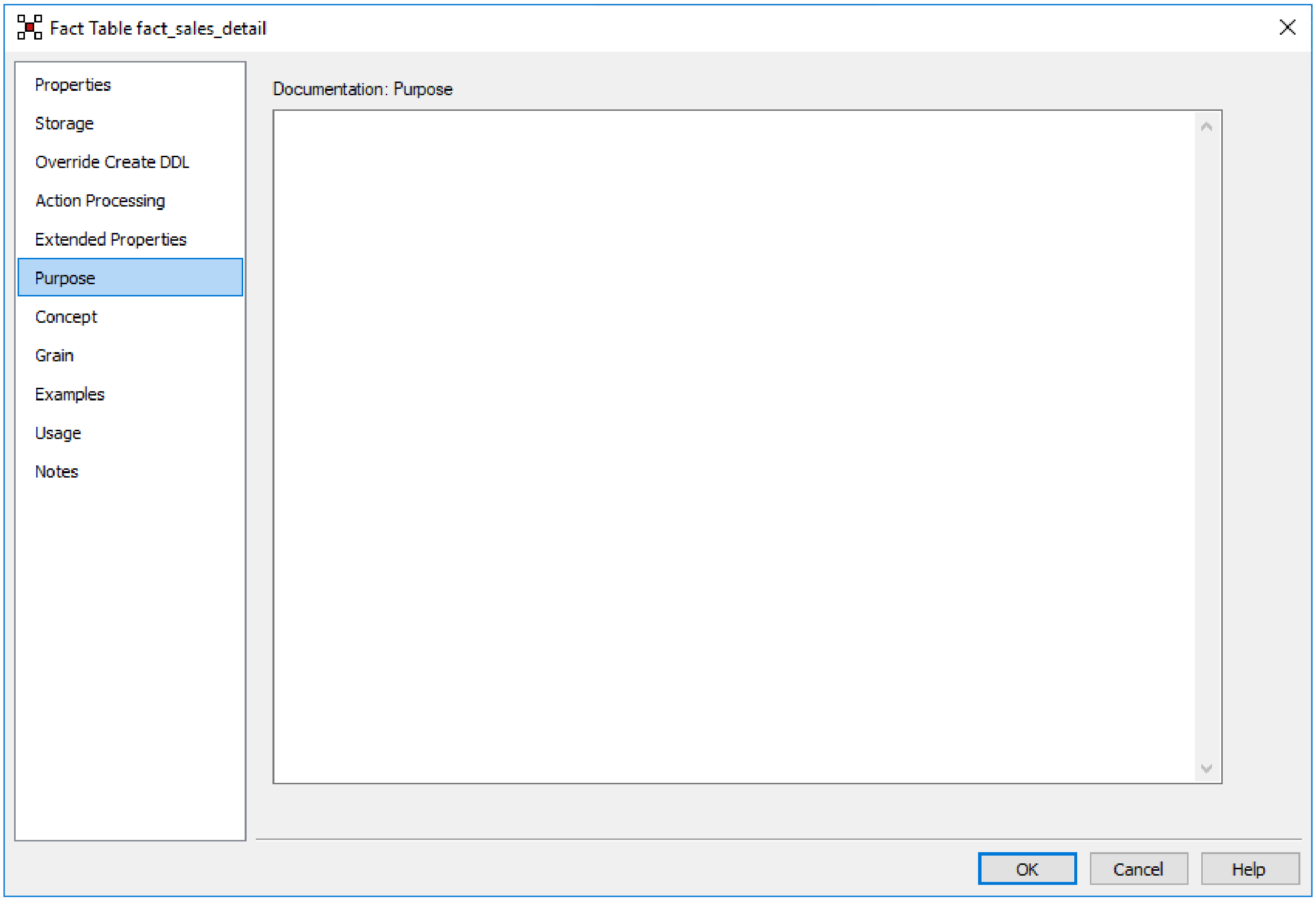
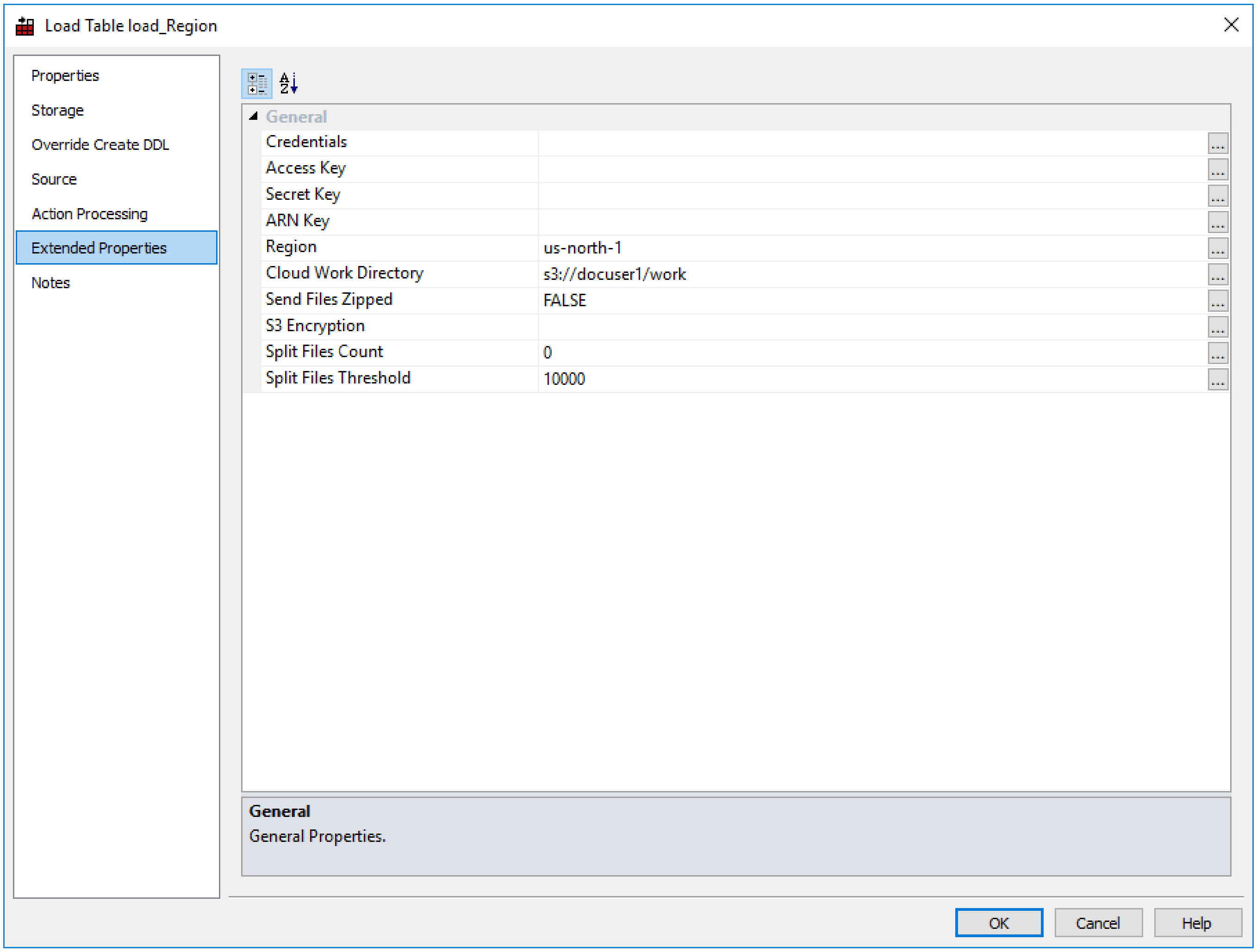


Table Extended Properties

Note

These settings enable you to assign extended property values for a table object. Refer to **Extended Properties** for details.



General

Credentials	
Access Key	
Secret Key	
ARN Key	
Region	us-north-1
Cloud Work Directory	s3://docuser1/work
Send Files Zipped	FALSE
S3 Encryption	
Split Files Count	0
Split Files Threshold	10000

General
General Properties.

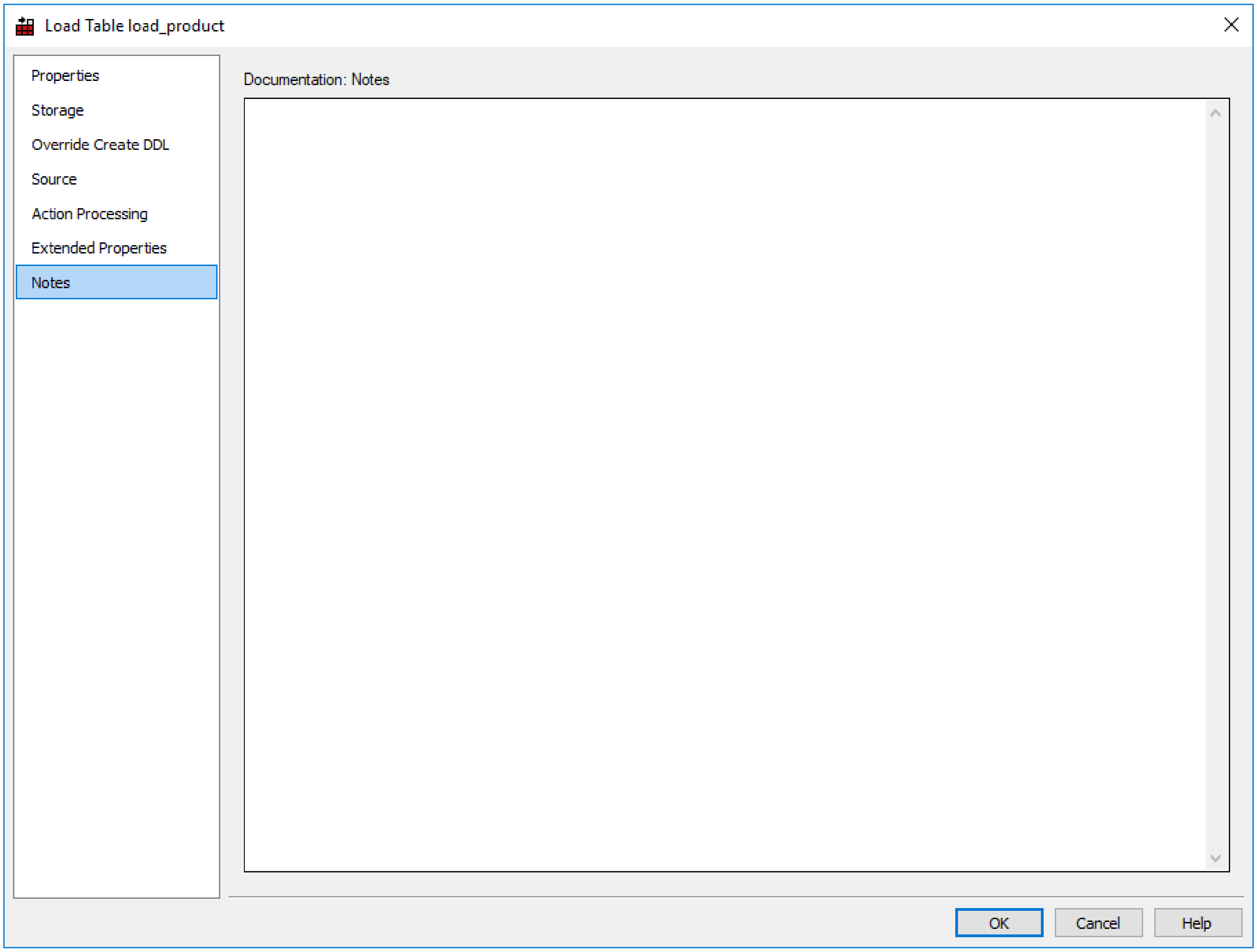
Notes

The **Notes** screen is used to enter notes for a particular WhereScape RED object.

The notes on an object are included in the WhereScape RED documentation. Up to 4000 characters of information can be stored in the Notes field.

By default, the Notes field is enabled for all object types.

The Notes field can be removed from the documentation via the **Tools > Options > Documentation** menu.


Load Table load_product
Properties
Storage
Override Create DDL
Source
Action Processing
Extended Properties
Notes
Documentation: Notes
OK
Cancel
Help

Staging

Stage tables are used to transform the data to a star schema or third normal form model. A stage table can be a fact or an EDW 3NF table that only contains change data or a work table. In star schema data warehouses, the stage table brings all the dimensional joins together in preparation for publishing into the fact table.

A stage table is built from the Data Warehouse connection. Unless you are retrofitting an existing system, stage tables are typically built from one or more load or stage tables. They can utilize the surrogate keys from a number of dimension tables.

The normal steps for creating a stage table are defined below and are covered in this chapter. The steps are:

1. Identify the source transactional data that will ultimately constitute the fact or EDW 3NF table. If the data is sourced from multiple tables ascertain if a join between the source tables is possible, or if a series of passes will be required to populate the stage table. If the latter option is chosen, then bespoke code is needed.
2. Using the 'drag and drop' functionality drag the table with the lowest granular data into a stage target. Refer to **Building the Stage Table** for details.
3. Add columns from other source tables. Refer to **Building the Stage Table** for details.
4. Add any relevant dimension table or EDW 3NF table keys. Refer to **Building the Stage Table** for details.
5. Create the stage table in the database. Refer to **Building the Stage Table** for details.
6. Build the update procedure. Refer to **Generating the Staging Update Procedure** for details.
7. Test the update procedure and analyze the results. Refer to **Tuning the Staging Update Process** for details.
8. Modify the update procedure as required. Refer to **Tuning the Staging Update Process** for details.

Note

If you are building a Data Vault system, a Stage table with sub type of Data Vault Stage can be created to generate hash keys that are used in building Data Vault objects (Hub, Link or Satellite tables). Refer to **Data Vaults** for details.

Building the Stage Table

Building the stage table is potentially the most challenging part of the overall task of building a data warehouse analysis area.

Most of the effort required is in the design phase, in terms of knowing what data needs to come into the fact table that will ultimately be built.

This section assumes that the decision as to what to include has been made.

Multiple Data Sources

A Stage table typically contains the change data for a detail fact table. As such, it normally maps to a specific function within the business and in many cases relates back to one main OLTP table. In many cases however, it may be necessary to combine information from a number of tables. One of the decisions required is whether or not it is practical or even possible to join the data from the different source tables.

We could however, also include two additional source tables being invoice_header and invoice_line which contain specific information relating to what was on the invoice. We may want our fact table to contain information from these tables as well. Although these two tables may contain the order_number and hence potentially allow a join with the order tables we may choose not to perform such a join for performance reasons. In this case we have three obvious choices in terms of how we ultimately update our fact table.

The choices are:

1. Join all four tables using one large join in our staging table.

2. Update the staging table in two passes; one pass updating the order information, the other pass updating the invoice information.
3. Generate two Stage tables, one for order and one for invoice. Use these two staging tables to update the one sales_detail fact table.

Although all three options are viable and a normal situation in the WhereScape RED environment, options (2) and (3) will require specific coding and modifications to the generated procedures from the outset.

Given the example provided, option (2) would be the normal approach; although in some cases option (3) would be valid.

Drag and Drop

The best approach in creating a Stage table is to choose the source table that contains the most fields that we will be using and drag this table into the stage target.

We can then drag specific columns from the other source tables until we have all the source data that is required.

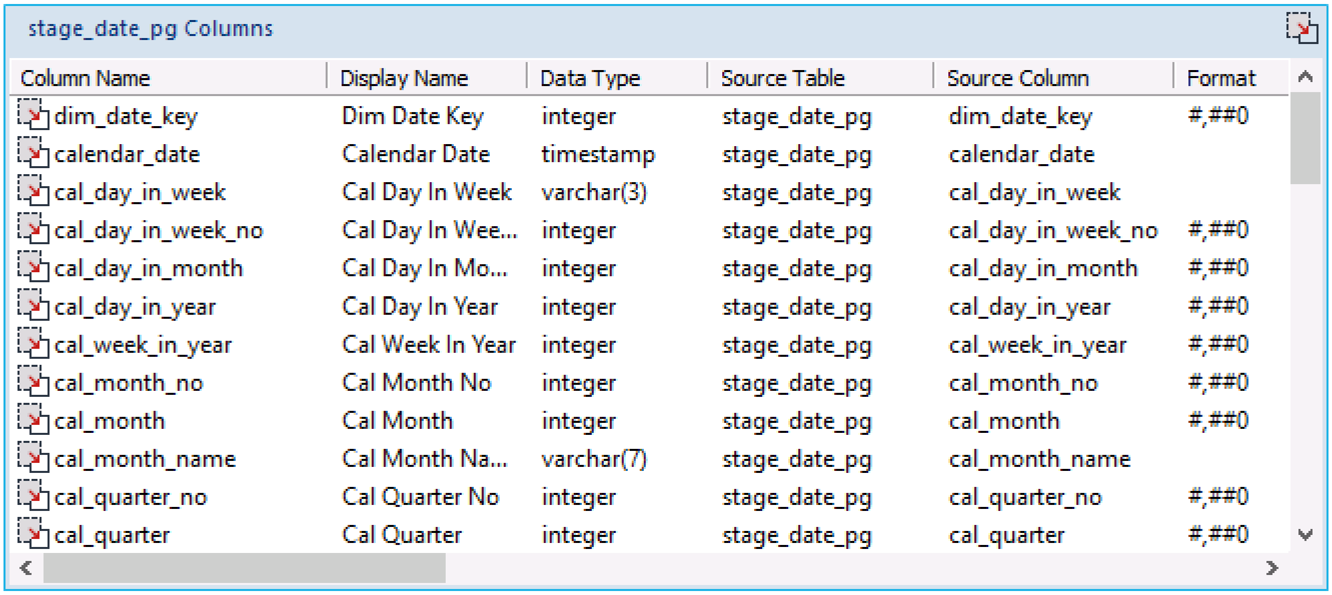
The process for defining the metadata is as follows:

1. Double-click on the **Stage Table** object group in the left pane. This will result in all existing stage tables being displayed in the middle pane. This also sets the middle pane as a **stage drop target**.
2. Browse the Data warehouse connection to display your Load tables in the right pane. This is done via the **Browse > Data Warehouse** menu option.
3. Drag the **primary load table** (i.e. the one with the most columns, or the lowest data granularity) from the right pane and drop it into the middle pane. A dialog appears to create the new staging object.
4. Leave the object type as 'Stage Table' and change the name to reflect what is being done.
5. Once a valid name is entered, the properties for the new Stage table are displayed. Normally these would be left unchanged except perhaps for storage settings.
6. Once the **Properties** window is closed, the columns for the new Stage table are displayed in the middle pane. This middle pane is now considered a **drop target for this specific stage table**. Any additional columns or tables dropped into the middle pane are considered additions to this stage table definition. Any columns that are not required can be deleted.
7. Drag and drop **additional columns** from other source tables if appropriate.
8. Drag in the **dimension artificial key** from each Dimension that is to be joined to the Stage/Fact table.
9. You can only join a Dimension, if a business key exists amongst the stage table columns or if it is possible to derive that business key in some way from the columns or other dimensions.

Note

If a column is being used to join information from two or more source tables, that column must only appear once in the Stage table. It is irrelevant which table is used to create the column in the new Stage table.

Once completed, your list of columns for the Stage table should look something like the list below. Notice the **Source Table** for each column.



stage_date_pg Columns

Column Name	Display Name	Data Type	Source Table	Source Column	Format
dim_date_key	Dim Date Key	integer	stage_date_pg	dim_date_key	#,##0
calendar_date	Calendar Date	timestamp	stage_date_pg	calendar_date	
cal_day_in_week	Cal Day In Week	varchar(3)	stage_date_pg	cal_day_in_week	
cal_day_in_week_no	Cal Day In Wee...	integer	stage_date_pg	cal_day_in_week_no	#,##0
cal_day_in_month	Cal Day In Mo...	integer	stage_date_pg	cal_day_in_month	#,##0
cal_day_in_year	Cal Day In Year	integer	stage_date_pg	cal_day_in_year	#,##0
cal_week_in_year	Cal Week In Year	integer	stage_date_pg	cal_week_in_year	#,##0
cal_month_no	Cal Month No	integer	stage_date_pg	cal_month_no	#,##0
cal_month	Cal Month	integer	stage_date_pg	cal_month	#,##0
cal_month_name	Cal Month Na...	varchar(7)	stage_date_pg	cal_month_name	
cal_quarter_no	Cal Quarter No	integer	stage_date_pg	cal_quarter_no	#,##0
cal_quarter	Cal Quarter	integer	stage_date_pg	cal_quarter	#,##0

The source table (src table) reflects where each column was dragged from.

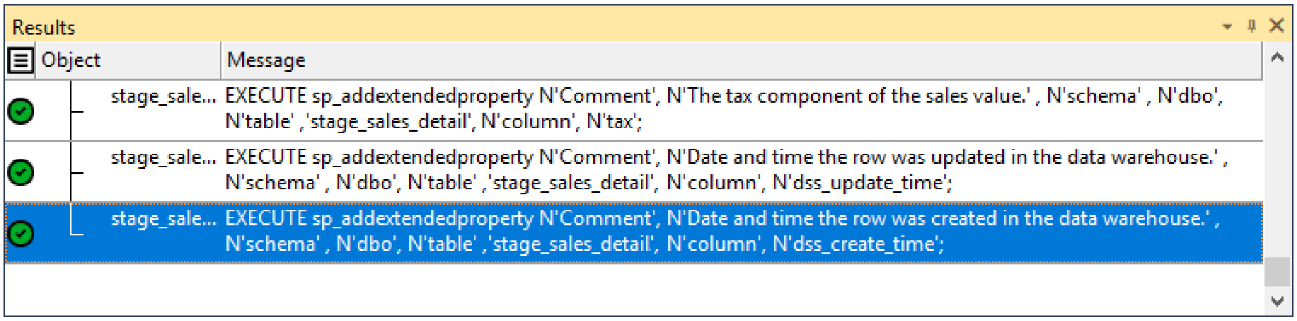
In the example above, the bulk of the columns came from the load_order_line table, and the customer_code, order_date and ship_date came from the load_order_header columns.
These two Load tables will be joined via the order_number column. This order_number column appears in both load tables but is sourced, in this example, from the load_order_line table.

Each dimension artificial key was dragged from its appropriate table. The final column 'dss_update_time' was generated by WhereScape RED and has no source.

Creating the Table

Once the Stage table has been defined in the metadata, you need to physically create the table in the database.

1. Right-click the Stage table from the left pane and select **Create (ReCreate)** from the pop-up menu.
2. The results pane displays the results of the creation. The following example shows a successful creation.



Results

	Object	Message
✓	stage_sale...	EXECUTE sp_addextendedproperty N'Comment', N'The tax component of the sales value.' , N'schema' , N'dbo', N'table' ,'stage_sales_detail', N'column', N'tax';
✓	stage_sale...	EXECUTE sp_addextendedproperty N'Comment', N'Date and time the row was updated in the data warehouse.' , N'schema' , N'dbo', N'table' ,'stage_sales_detail', N'column', N'dss_update_time';
✓	stage_sale...	EXECUTE sp_addextendedproperty N'Comment', N'Date and time the row was created in the data warehouse.' , N'schema' , N'dbo', N'table' ,'stage_sales_detail', N'column', N'dss_create_time';

The contents of this pane is a message to the effect that the table was created followed by a copy of the actual database create statement, and if defined the results of any index creates. For the initial create no indexes will be defined.

If the table was not created then ascertain and fix the problem. A common problem is a 'Duplicate column' where a column has been accidentally added twice.

The best way to find such a column is to double-click the list heading **Column Name**, to sort the column names into alphabetical order.

Another double-click on the heading sorts the columns back into their create order. Column ordering can be changed by altering the column order value against a column's Properties.

Tip

Tip

Double-clicking the heading of a column in a list, sorts the list into alphabetical order based on the column chosen.

Generating the Staging Update Procedure

Once a stage table has been defined in the metadata and created in the database, an update procedure can be generated to handle the joining of any tables and the lookup of the dimension table artificial keys.

Note

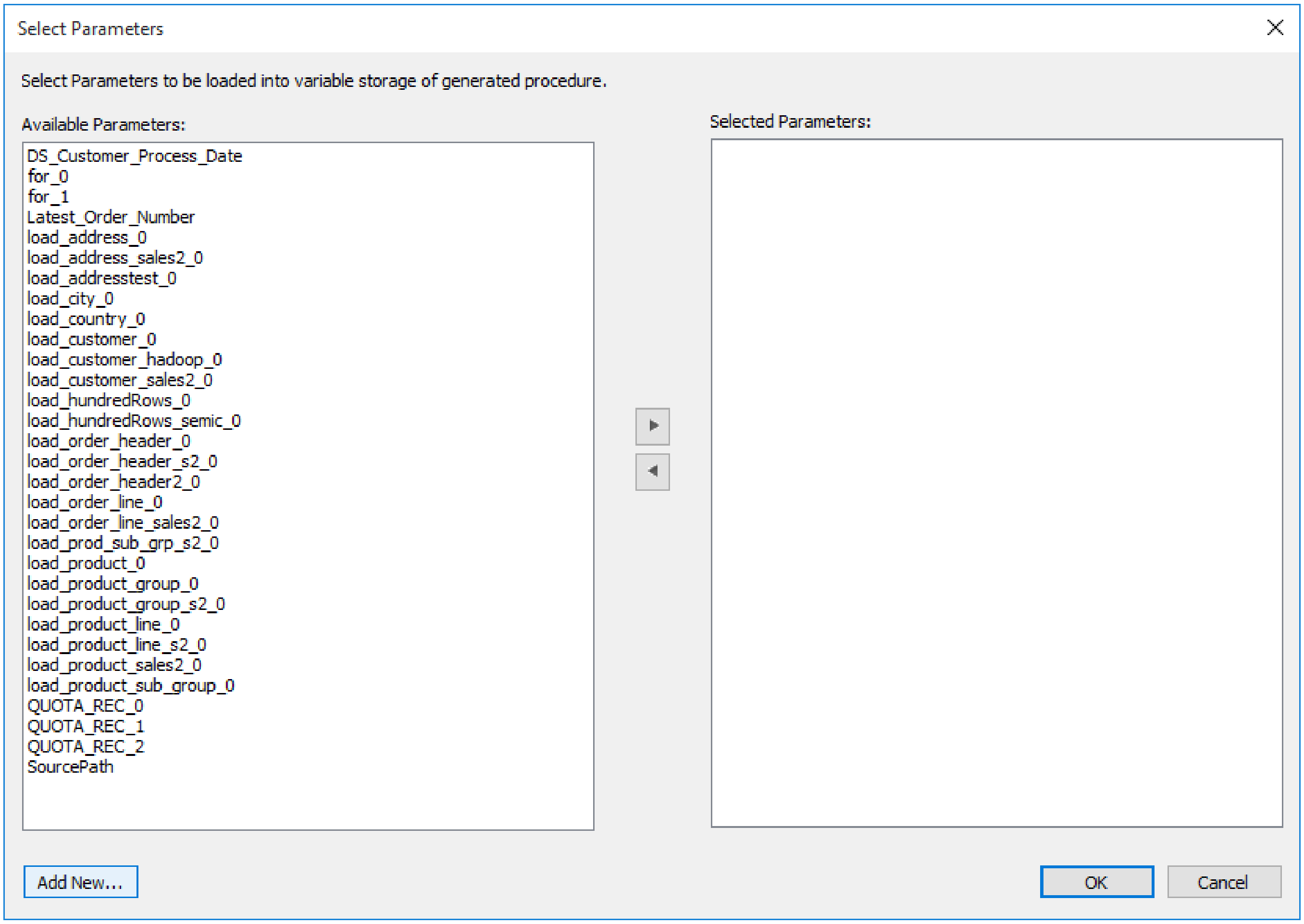
You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.

Generating a Procedure

1. To generate a procedure, right-click on the stage table in the left pane and select **Properties**.
2. From the **Update Procedure** drop-down list, select **(Build Procedure...)**.
3. Click **OK** to update the properties and start the process of generating the new procedure.
4. A series of prompts are displayed during the procedure generation to join the tables and link the dimensions.

Parameters

If WhereScape RED parameters exist in the metadata, the following window is displayed. Any parameters selected in this dialog (by moving them to the right pane), are included in the generated update procedure as variables.


Select Parameters
Select Parameters to be loaded into variable storage of generated procedure.
Available Parameters:
Selected Parameters:
Add New...
OK
Cancel

DS_Customer_Process_Date
for_0
for_1
Latest_Order_Number
load_address_0
load_address_sales2_0
load_addresstest_0
load_city_0
load_country_0
load_customer_0
load_customer_hadoop_0
load_customer_sales2_0
load_hundredRows_0
load_hundredRows_semic_0
load_order_header_0
load_order_header_s2_0
load_order_header2_0
load_order_line_0
load_order_line_sales2_0
load_prod_sub_grp_s2_0
load_product_0
load_product_group_0
load_product_group_s2_0
load_product_line_0
load_product_line_s2_0
load_product_sales2_0
load_product_sub_group_0
QUOTA_REC_0
QUOTA_REC_1
QUOTA_REC_2
SourcePath

The variables can also be used in column transformations and in the From/Where clause for the update procedure.

Some databases have a 30 character limit for variable names. WhereScape RED ensures the variables added for any parameters are less than 30 characters long by creating variable names in the form **v_** followed by the first 28 characters of the parameter name.

For example, a parameter called MINIMUM_ORDER_NUMBER_SINCE_LAST_SOURCE_LOAD will be available as the variable v_MINIMUM_ORDER_NUMBER_SINCE_L.

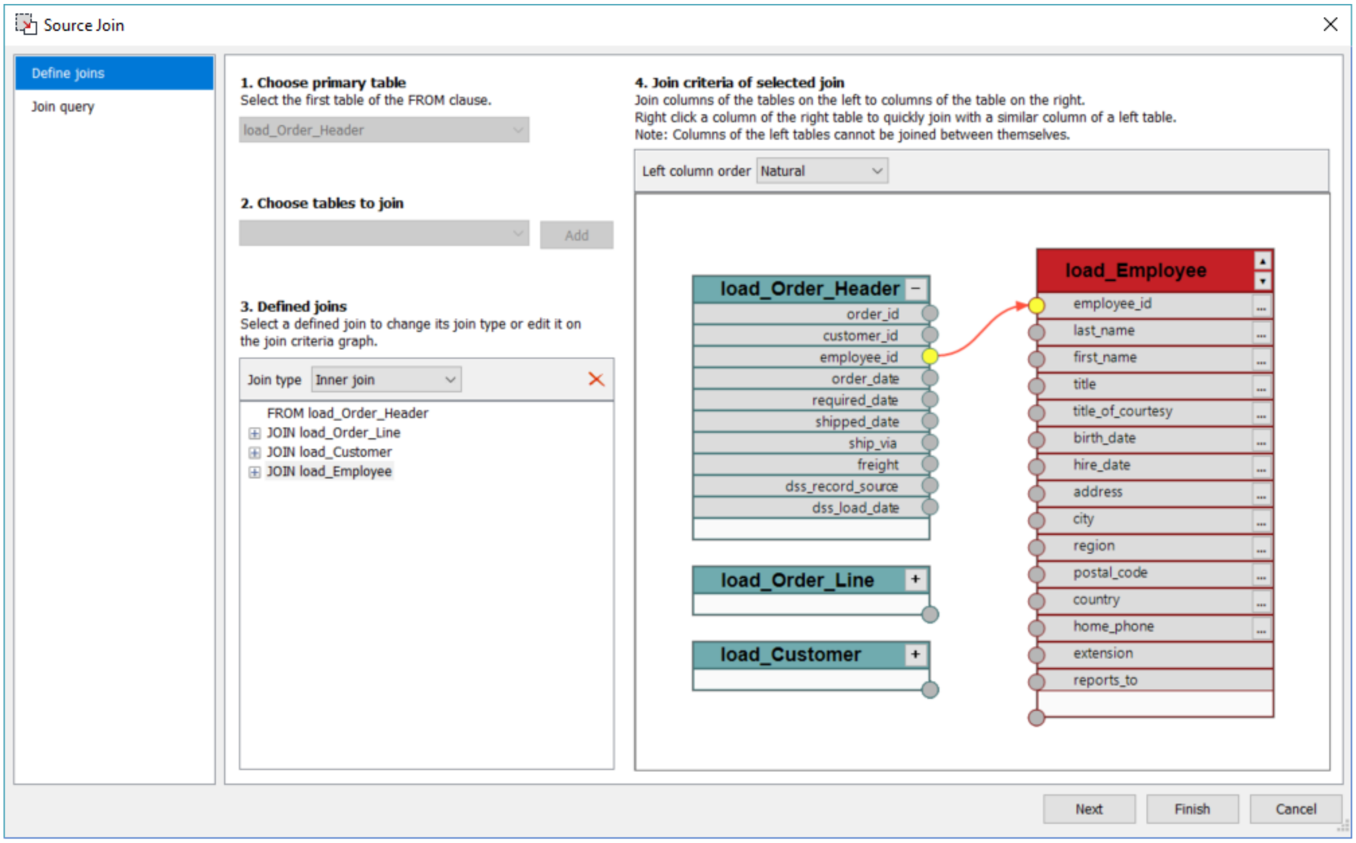
Tips

- WhereScape RED parameters should be unique within the first 28 characters to avoid conflicting variables names.
- If the desired parameter doesn't exist in the metadata yet, a new parameter can be added by clicking on the **Add New** button on the bottom leftmost corner of the Select Parameters dialog.

Refer to **Parameters** for more information on WhereScape RED Parameters.

Multiple Source Tables

WhereScape RED provides a wizard that enables you to graphically generate the join clause for an object with multiple source tables. The **Source Join** wizard enables you to specify the primary (driving) table and the source tables to join. A graphical representation of the tables and the join criteria is displayed by the wizard which you can manipulate to change the defined joins.



1. Choose primary table
Select the first table of the FROM clause.
load_Order_Header

2. Choose tables to join

3. Defined joins
Select a defined join to change its join type or edit it on the join criteria graph.

Join type: Inner join

FROM load_Order_Header
JOIN load_Order_Line
JOIN load_Customer
JOIN load_Employee

4. Join criteria of selected join
Join columns of the tables on the left to columns of the table on the right.
Right click a column of the right table to quickly join with a similar column of a left table.
Note: Columns of the left tables cannot be joined between themselves.

Left column order: Natural

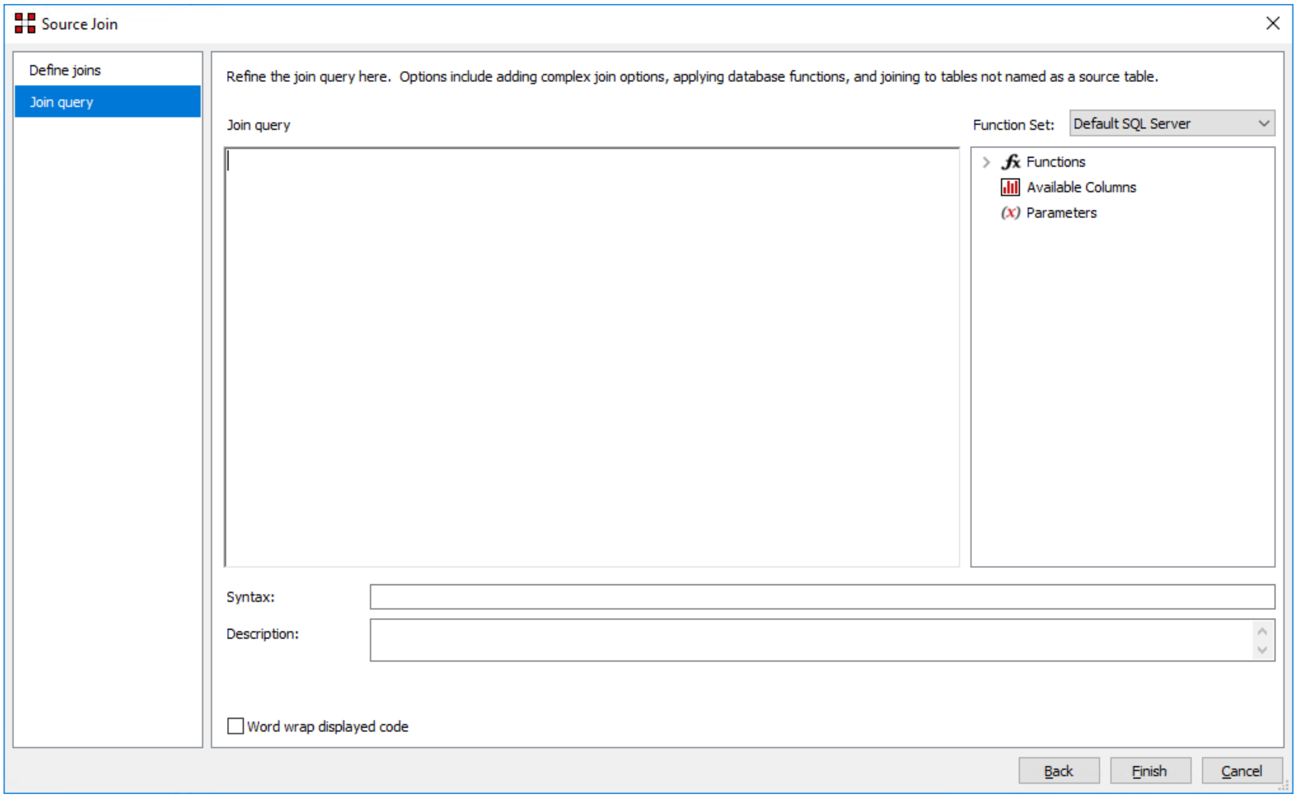
The **Source Join** wizard guides you in generating the join clause through the following steps:

1. **Choose Primary Table**
   The first source table encountered is the default primary (driving) table which can be changed before any joins are defined.
2. **Choose Tables to Join**
   Once the primary table is set, the remaining source tables are listed in this drop down. Select a table to join and then click **Add**. The two tables are graphically added in pane 4, which is the basis for the join displayed in pane 3.
3. **Defined Joins**
   The join type defaults to **Inner** join but this can be changed from the drop down.
4. **Join criteria of selected join**
   The column(s) used to join the two tables can be joined manually by clicking the corresponding join column connection point from the table on the left and then dragging the line that appears to the connection point of the required column in the table on the right.
   Alternatively, if a column with a matching or similar name exists in the left hand table(s) then a context menu is provided on the columns on the right-hand table. You can right click the column to display the context menu and then select a column from the list.

   Tip

   To aid manual column join, you can expand or collapse columns in the tables on the left and toggle between natural and alphabetic column ordering.
5. Once the join column(s) for the table pair are defined, then any additional source tables can be added from the drop-down list in step 2 and the above column mapping process repeat to join this table.

   This process is repeated until all source tables are joined.
6. Clicking the **Next** button or the **Join Query** tab progresses to the join editor window, where custom joins, database functions, etc. can be added.



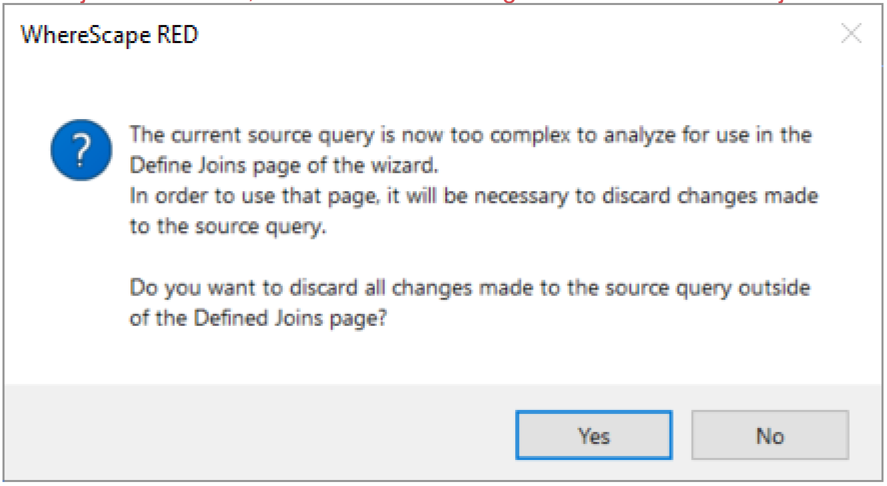
Refine the join query here. Options include adding complex join options, applying database functions, and joining to tables not named as a source table.

Join query

☐ Word wrap displayed code

Note

If RED is unable to parse a join statement, the user cannot navigate back to the define join tab and the following warning is displayed:



The current source query is now too complex to analyze for use in the Define Joins page of the wizard.
In order to use that page, it will be necessary to discard changes made to the source query.

Do you want to discard all changes made to the source query outside of the Defined Joins page?

Dimension Joins

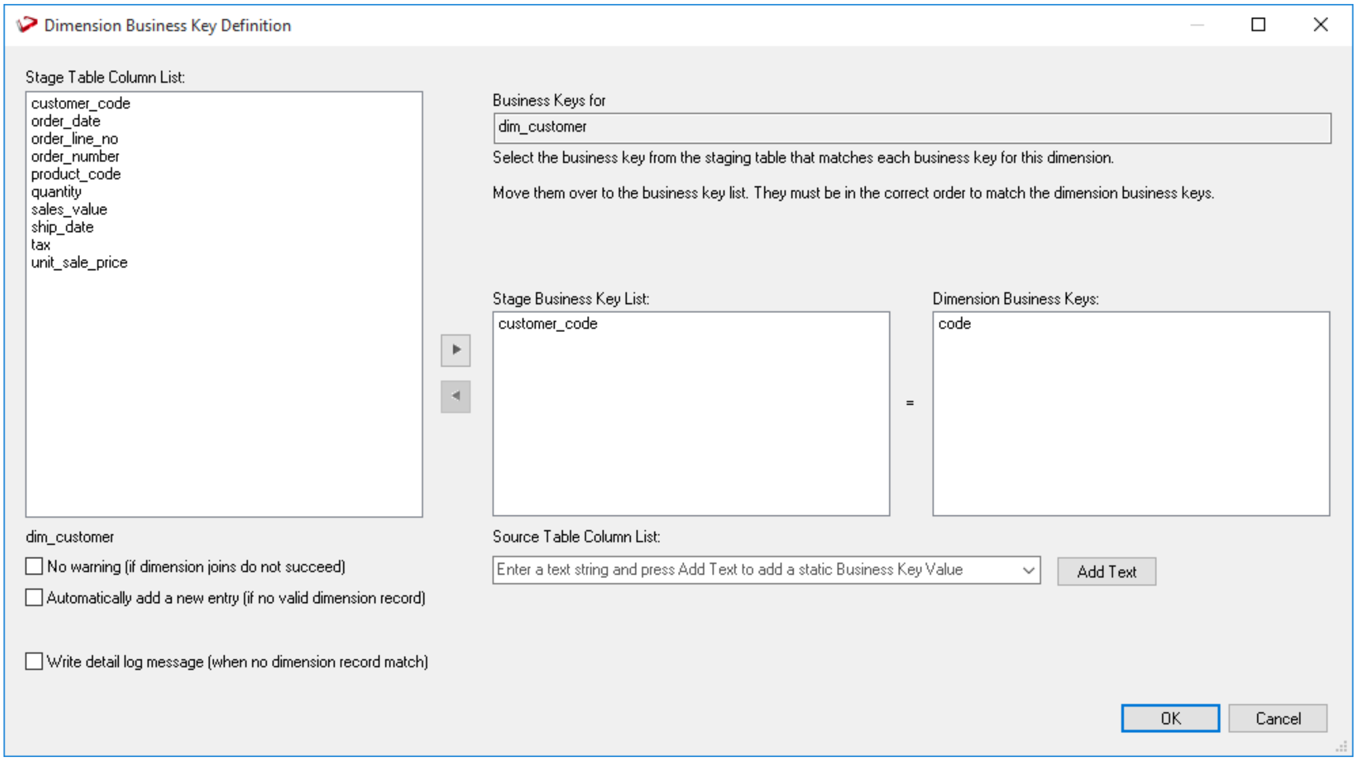
For each dimension key, a dialog appears asking for the business key from the stage table that matches the business key for the dimension.

In the example below, we are asked for the stage table business key for the customer dimension. The dimension name is shown both on the first prompt line and at the lower left side of the window.

The customer dimension has a unique business key named **code**.

We must provide the corresponding business key from the staging table. In the case of our example, this is the customer_code column.

1. Click **OK** after the correct business key has been entered.
2. If the business key does not exist and will be derived from another dimension or from some form of lookup, then enter any column and edit the procedure once produced.


Dimension Business Key Definition

Stage Table Column List:

customer_code
order_date
order_line_no
order_number
product_code
quantity
sales_value
ship_date
tax
unit_sale_price

Business Keys for: dim_customer

Select the business key from the staging table that matches each business key for this dimension.

Move them over to the business key list. They must be in the correct order to match the dimension business keys.

Stage Business Key List: customer_code

Dimension Business Keys: code

dim_customer

□ No warning (if dimension joins do not succeed)

□ Automatically add a new entry (if no valid dimension record)

□ Write detail log message (when no dimension record match)

Source Table Column List: Enter a text string and press Add Text to add a static Business Key Value

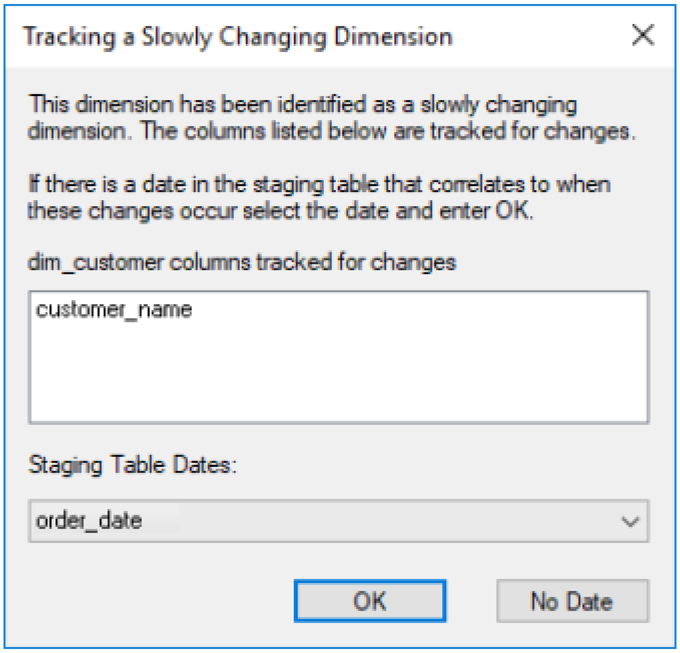
Note:

The **Add Text** button and the associated message and edit box are only shown if the user possess a full license, thus enabling the advanced procedure build options. When the **Add Text** button is clicked, any data in its edit box is placed in the Stage table column list. In this way, a number or string can be assigned as part or all of a Dimension join.

Slowly Changing Dimension information

If the dimension being joined was defined as a slowly changing dimension, then an additional window appears, as shown below. This window asks for a date field in the Stage table that enables RED to determine which version of the slowly changing dimension (the **customer_name** field, below) to use based on the specified date range in the Customer Dimension.

Select the appropriate date field for your business needs and click **OK**. If you wish to take the last (or current) version for the dimension, select **No Date**.



Tracking a Slowly Changing Dimension

This dimension has been identified as a slowly changing dimension. The columns listed below are tracked for changes.

If there is a date in the staging table that correlates to when these changes occur select the date and enter OK.

dim_customer columns tracked for changes

customer_name

Staging Table Dates:

order_date

For Example:

As shown in the screen above, we have defined the **customer_name** as an item in the dimension that we expect to have versions for over time, e.g. each time the data warehouse processing sees a new **customer_name** value, the dimension will record the date range for that version's validity even though the business key (customer_code in this example) remains the same. This implies we want to create a new dimensional record whenever a customer name is changed even though the customer_code remains the same.

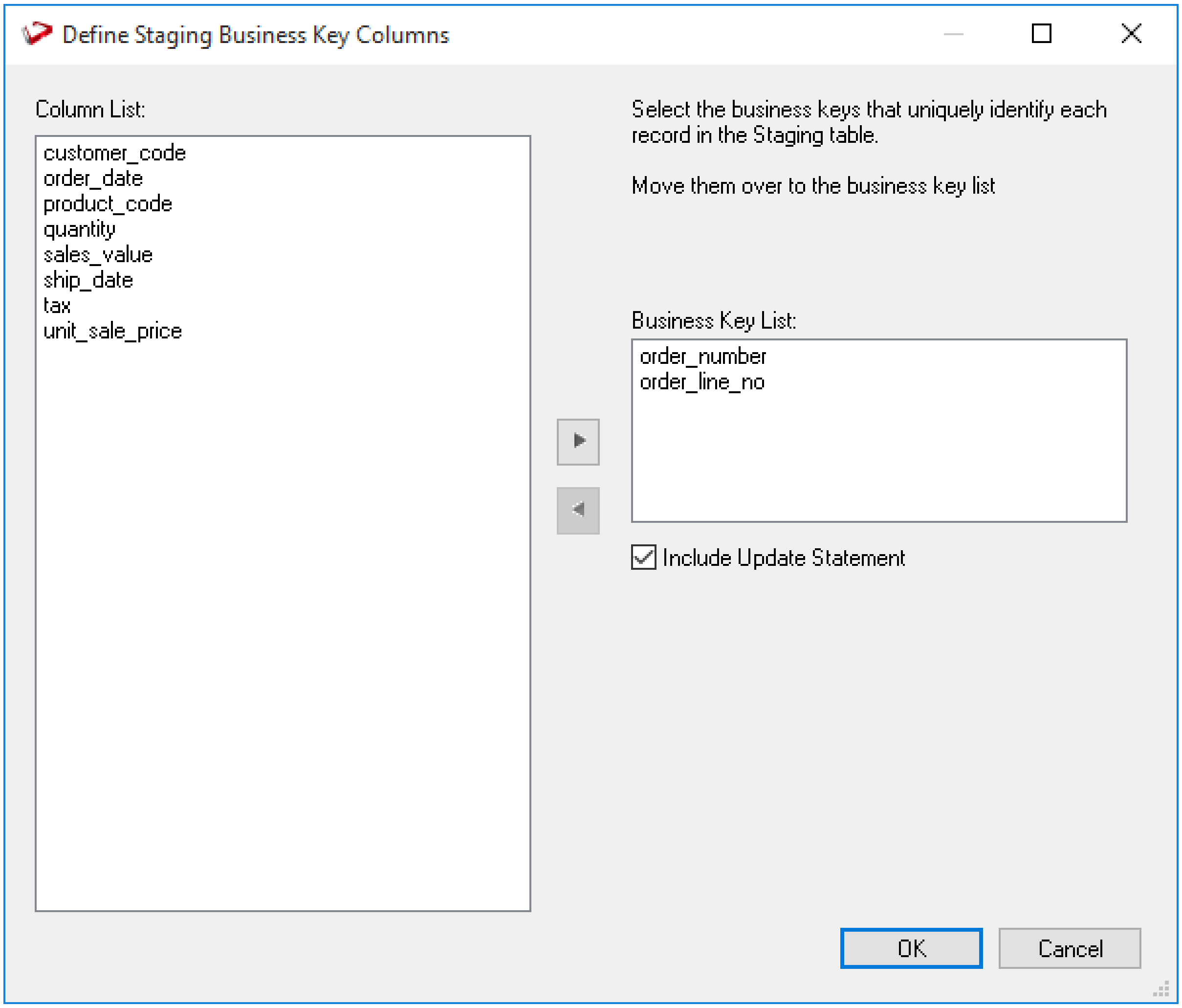
Let's say a customer changes their name on the 5th of the month. If the **Staging Table Dates** field is set to **order_date**, any order received before the 5th of the month is identified under the old customer name and any order received on or after the 5th has the new customer name.

Alternatively, by setting the **Staging Table Dates** to **ship_date**, we can specify that any order **shipped** on or after the 5th of the month is shipped with the new name.

Staging Business Key

Once all the dimensional joins have been defined, we will be asked to define the unique business key for the staging table. This is the column or columns that allow us to uniquely identify each record in the staging table.

In the example below, the unique business key is a combination of the **order_number** and the **order_line_no**.



Define Staging Business Key Columns

Column List:

customer_code
order_date
product_code
quantity
sales_value
ship_date
tax
unit_sale_price

Select the business keys that uniquely identify each record in the Staging table.

Move them over to the business key list

Business Key List:

order_number
order_line_no

☑ Include Update Statement

Notes

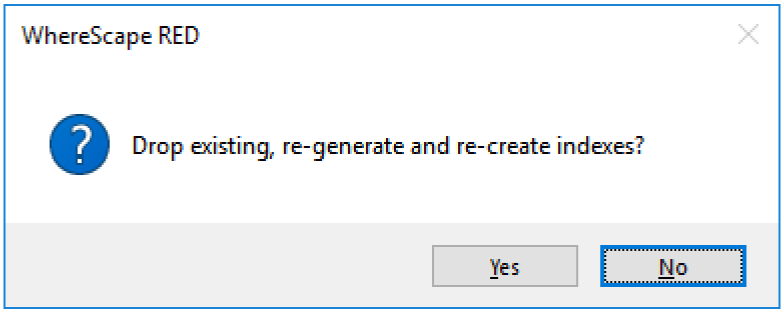
- The order of the columns in the business key list is set to the order that the columns appear in the table. This is done to optimize performance for any indexes.
- **NULL VALUES:** None of the columns chosen as the business key should ever contain a NULL value. See the note in the start of the Dimension chapter.

The **Include Update Statement** check-box provides two options in the generated procedure. If selected, then an update/insert combination will be included. If clear, then only an Insert statement will be included in the procedure. If cleared, you must be sure that the business key for the table is unique, otherwise either a duplicate key error will occur or records will be lost, depending on the type of procedure.

Index re-creation

Finally, you are asked if you want to drop and re-create any indexes defined on the table:

- Click **Yes** to drop and re-create.
- **No** to leave existing indexes in place.


WhereScape RED

Drop existing, re-generate and re-create indexes?

Stage Table Column Properties

Each Stage table column has a set of associated properties. The definition of each property is defined below:

If the **Column name** or **Data type** is changed for a column, then the metadata will differ from the table as recorded in the database.

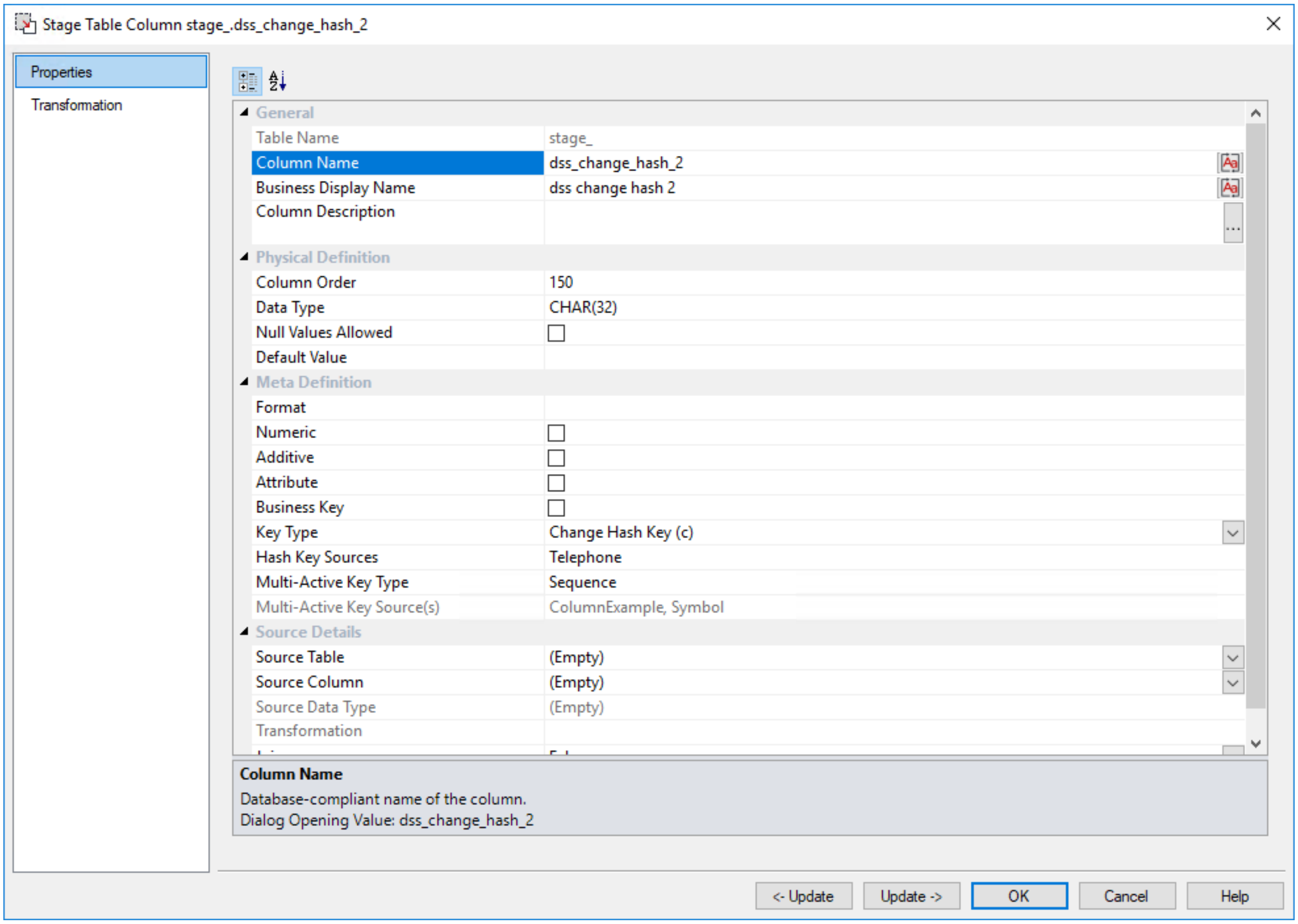
Use the **Validate > Validate Table Create Status** menu option to compare the metadata to the table in the database.

A context menu option **Alter Table** is available when you right click the table name, after the validate has been completed. This option alters the database table to match the metadata definition.

Tip

If a database table's definition is changed in the metadata, then the table need to be altered in the database. Use the **Validate > Validate Table Create Status** menu to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table, through a pop-up menu option from the validated table name.

A sample **Properties** screen is as follows:



Stage Table Column stage_.dss_change_hash_2

General	
Table Name	stage_
Column Name	dss_change_hash_2
Business Display Name	dss change hash 2
Column Description	
Physical Definition	
Column Order	150
Data Type	CHAR(32)
Null Values Allowed	□
Default Value	
Meta Definition	
Format	
Numeric	□
Additive	□
Attribute	□
Business Key	□
Key Type	Change Hash Key (c)
Hash Key Sources	Telephone
Multi-Active Key Type	Sequence
Multi-Active Key Source(s)	ColumnExample, Symbol
Source Details	
Source Table	(Empty)
Source Column	(Empty)
Source Data Type	(Empty)
Transformation	

Column Name
Database-compliant name of the column.
Dialog Opening Value: dss_change_hash_2

Tip

The two special update keys enables you to update the column and step either forward or backward to the next column's properties.
ALT-Left Arrow and **ALT-Right Arrow** can also be used instead of the two special update keys.

General

Options	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** The case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence of each conversion.
Business Display Name	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** The case conversion button on the right converts the text between different cases:

Options	Description
	UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence of each conversion.
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col .** This field is also stored as a comment against the column in the database.

Physical Definition

Options	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. For SQL Server, common types are integer, numeric, varchar() and datetime. See the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.

Meta Definition

Options	Description
Numeric	Indicates whether the table column holds values that are numeric. This is normally only relevant for Fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Additive	Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.

<table>
<tr><th>Options</th><th>Description</th></tr>
<tr><td><b>Attribute</b></td><td>Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example, we may have an order number, or a invoice number stored in the Fact table. These columns are considered attributes, rather than facts.
This check-box is therefore normally only relevant for fact tables. This check-box does not affect the physical table definition, but rather provides input to the view <b>ws_admin_v_dim_col</b> which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.</td></tr>
<tr><td><b>Business Key</b></td><td>Indicates whether the column is part of the BUSINESS key, which is defined during the update procedure build. [Normally maintained automatically]. Multiple columns can form the primary business key.</td></tr>
<tr><td><b>Key Type</b></td><td>Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested.
The supported values are:
<table>
<tr><th>Key type</th><th>Meaning</th></tr>
<tr><td><b>0</b></td><td>The artificial key. Set when the key is added during drag and drop table generation.</td></tr>
<tr><td><b>1</b></td><td>Component of all business keys. Indicates that this column is used as part of any business key. For example: By default the dss_source_system_key is added to every dimension table. It is considered part of any lookup on that table and has the key type set to 1. Set when the column is added during drag and drop table generation.</td></tr>
<tr><td><b>2</b></td><td>Indicates that this column is a dimensional join. Used on fact tables to indicate the dimension keys. Results in bitmap indexes being built for the columns. Set during the update procedure generation for a fact table, based on information from the staging table.</td></tr>
<tr><td><b>3</b></td><td>Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation.</td></tr>
<tr><td><b>4</b></td><td>Previous value column indicator. Used on dimension tables to indicate that the column is being managed as a previous value column. The source column identifies the parent column. Set during the dimension creation.</td></tr>
<tr><td><b>5</b></td><td>Indicates a column is a start date column.</td></tr>
<tr><td><b>6</b></td><td>Indicates a column is a end date column.</td></tr>
<tr><td><b>7</b></td><td>History column indicator. Used in model history tables to indicate that the column is being managed as a history column within the context of a model history table. Set when a column is identified during the model history update procedure generation.</td></tr>
<tr><td><b>c</b></td><td>Change Hash Key column indicator. Used in Data Vault tables to indicate the differences in the descriptive columns of a Satellite table. Refer to <b>Data Vaults</b> for details.</td></tr>
<tr><td><b>h</b></td><td>Hub Hash Key column indicator. Used in Data Vault tables to indicate the hash key column of a Hub Table. Refer to <b>Data Vaults</b> for details.</td></tr>
<tr><td><b>l</b></td><td>Link Hash Key column indicator. Used in Data Vault tables to indicate the hash key column of a Link Table. Refer to <b>Data Vaults</b> for details.</td></tr>
<tr><td><b>m</b></td><td>Multi-Active Natural key indicator. Used in Data Vault tables to indicate the Multi-Active key column.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Options</th><th>Description</th></tr>
<tr><td></td><td><table>
<tr><td><b>s</b></td><td>Multi-Active Sequence key indicator. Used in Data Vault tables to indicate the Multi-Active key column for a generated sequence key</td></tr>
<tr><td><b>A</b></td><td>Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation.</td></tr>
<tr><td><b>B-Z</b></td><td>Indicates that the column is part of a secondary business key. Only used during index generation and not normally set.</td></tr>
</table></td></tr>
<tr><td><b>Hash Key Sources</b></td><td>This field is only displayed for Hash key types. Displays the hash source columns that are used to generate the selected Hub, Link or Change hash key.</td></tr>
<tr><td><b>Hash Key Source For</b></td><td>This field is only displayed for Hash key types. Displays the hash keys columns that use the displayed hash key sources.</td></tr>
<tr><td><b>Multi-Active Key Type</b></td><td>The value is only 'Natural' or 'Sequence' based on the Multi-Active key type.</td></tr>
<tr><td><b>Multi-Active Key Source(s)</b></td><td>Lists the key column for Multi-Active Natural Key or the sort columns for Multi-Active Sequence Key.</td></tr>
</table>

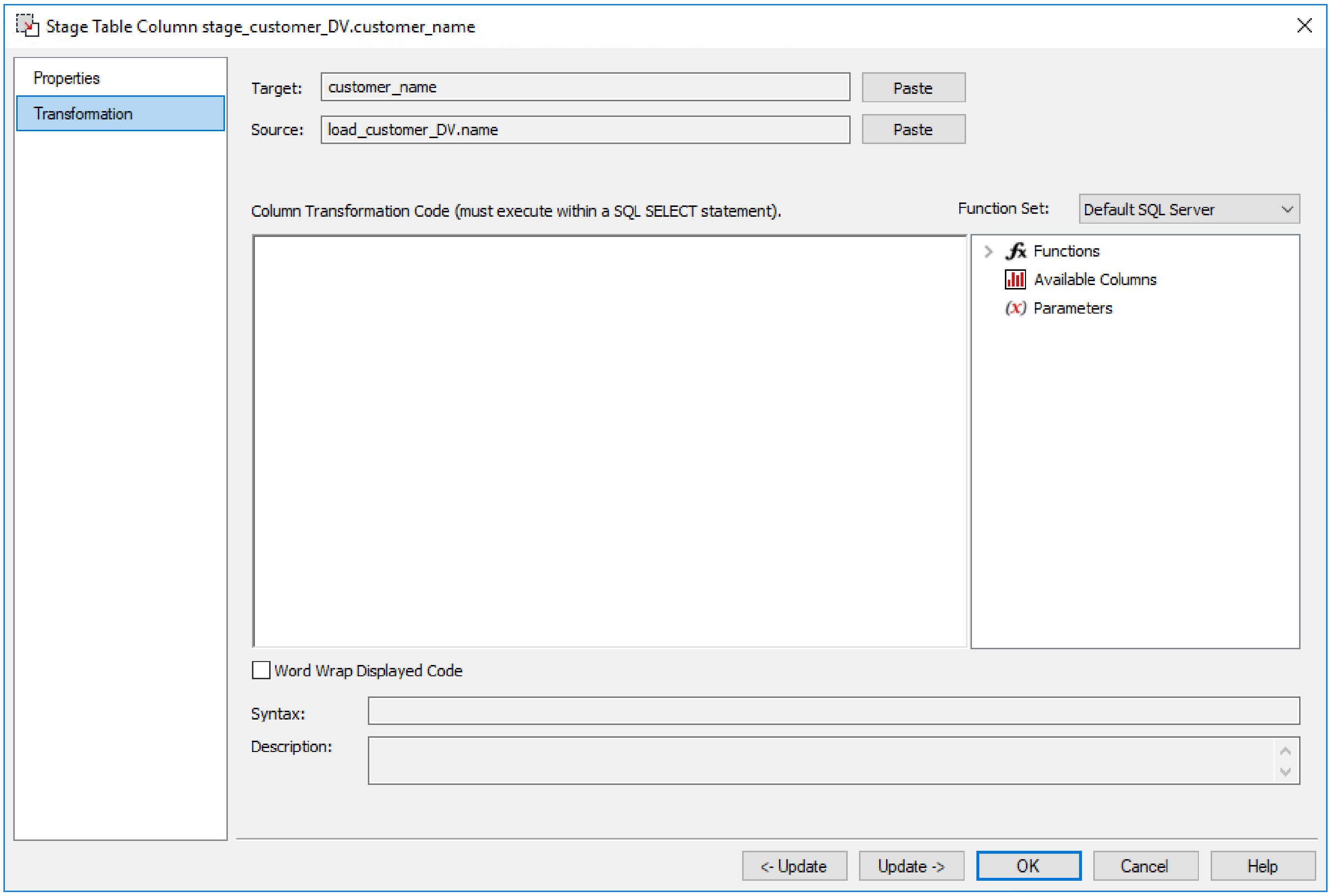
Source Details

Options	Description
Source Table	Identifies the source table where the column's data comes from. This source table is normally a Load table, or a Dimension table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the **Source strategy** field. This field is used when generating a procedure to update the Stage table. It is also used in the track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a load table column, which in turn may have been a transformation or the combination of multiple columns. This may also be a dimensional key where a dimension is being joined.
Source Data Type	Identifies the source column's data type. [Read-only].
Transformation	Refer to **Stage Table Column Transformations** for details. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source table** and **Source column** fields will provide the Dimension table's side of the join. The options for this field are: False, True, Manual and Pre Join. Setting this field to Manual, changes the way the Dimension table is looked up during the Stage table update procedure build. It enables you to join the dimension manually in the **Source Join** wizard (used to build the 'Where' clause). The usual dialog for matching the dimension business key to a column or columns in the Stage table is not displayed, if this option is enabled. Setting this field to Pre Join, activates the **Pre Join Source Table** field and enables you to select a table from the drop-down list.
Pre Join Source Table	Indicates the table from which the pre joined column was sourced. When the Join option is set to False, this field becomes inactive. When the Join option is set to True or Manual, this field is set to the current table name. When the Join option is set to Pre Join, then you can select the required table from the drop-down list.

Stage Table Column Transformations

Each Stage table column can have a transformation associated with it. The transformation will be included in the generated procedure and will be executed as part of the procedure update.
The transformation must therefore be a valid SQL construct that can be included in a **Select** statement.

For example we could have a transformation of 'load_order_line.quantity * 0.125' to calculate a tax column of 12.5%. Click the **Transformation** tab to enter a transformation.



Stage Table Column stage_customer_DV.customer_name

Target: customer_name

Source: load_customer_DV.name

Function Set: Default SQL Server

□ Word Wrap Displayed Code

Note

Transformations only take effect when the procedure is re-generated. Refer to **Transformations** for details.

Data Store Objects

Overview

A Data Store Object is a data warehouse object used to store any type of data for later processing.
In WhereScape RED, Data Store objects have many of the code generating attributes of Stage, Dimension and Fact tables. Data Store objects can be thought of as a source system for the data warehouse.
Alternatively, they may be reported off directly by users and reporting tools. Data Store Objects can be considered either reference or transactional in nature.

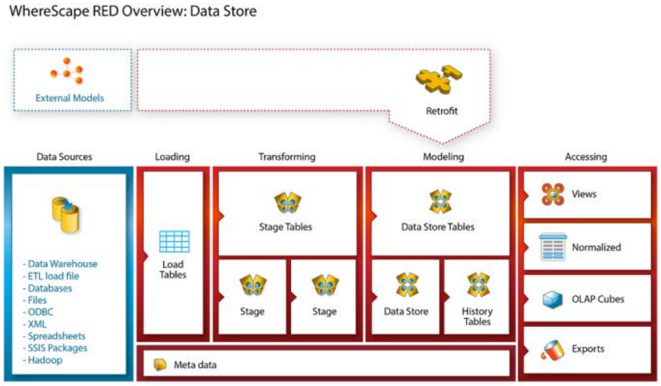
A Data Store Object is built from the Data Warehouse connection. Unless you are retrofitting an existing system, Data Store Objects are typically built from one or more Load or Stage tables.
The Data Store model may be retrofitted from an enterprise modeling tool. Refer to **Importing a Data Model** for details.

The usual steps for creating a Data Store model are defined below and are covered in this chapter.

The steps are:

1. Identify the source reference or transactional data that will constitute the Data Store Object. If the data is sourced from multiple tables ascertain if a join between the source tables is possible, or if one or more intermediate stage (work) tables would be a better option.
2. Using the 'drag and drop' functionality drag the load or stage table that is the primary source of information for the Data Store Object into a Data Store target. Refer to **Building a Data Store Object** for details.
3. If there's only one source table and all of the columns from it are being used, you can select the auto create option to build and load the table. This automatically completes the next four steps. Refer to **Building a Data Store Object** for details..
4. Add columns from other Load and/or Stage tables if required. Refer to **Building a Data Store Object** for details.
5. Create the Data Store Object in the database. Refer to **Building a Data Store Object** for details.
6. Build the update procedure. Refer to **Generating the Data Store Update Procedure**.
7. Run the update procedure and analyze the results.

If necessary, modify the update procedure or create a custom procedure.



Data Store Object Keys

Data Store Objects have Business Keys, they do not usually have Artificial Keys.

Business Key

The business key is the column or columns that uniquely identify a record within a Data Store Object.

If the Data Store Object maps back to a single or a main table in the source system, it is usually possible to ascertain the business key by looking at the unique keys for that source table.

The business key is sometimes referred to as the 'natural' key. Examples of business keys are:

- The product SKU in a product table
- The customer code in a customer table
- The IATA airport code in an airport table.

It is assumed that business keys will never be NULL. If a NULL value is possible in a business key then the generated code needs to be modified to handle the NULL value by assigning some default value.
In the following examples, the business key column is modified by using a database function and default value:

- **DB2:** COALESCE(business_key,'N/A')
- **SQL Server:** ISNULL(business_key,'N/A')

Note

Business keys are assumed to never be NULL. If they can be null it is best to transform them to some value prior to the Data Store or Stage table update.
If this is not done, an unmodified update procedure will probably fail with a duplicate key error on the business key index.

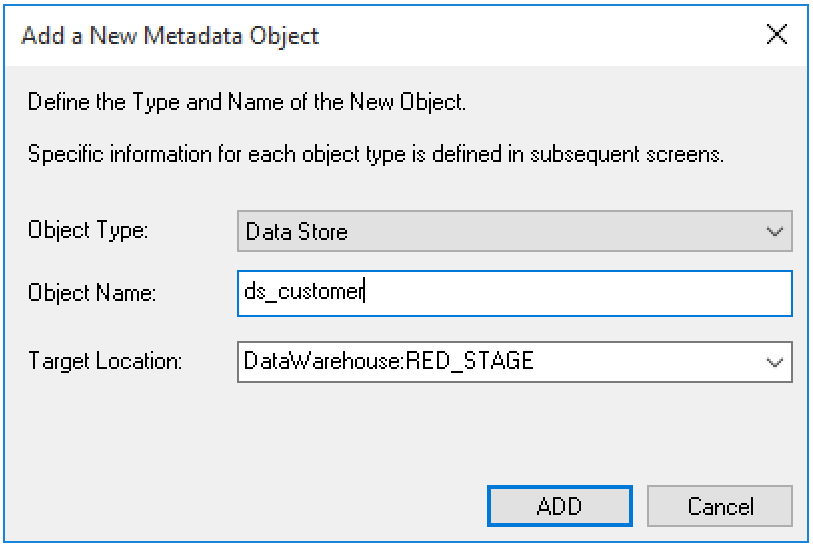
Building a Data Store Object

Data Store Objects are often sourced from one table in the base application.

The process for building a Data Store Object begins with the drag and drop of the load or stage table that contains the bulk of the Data Store Object's information.

Drag and Drop

1. Create a Data Store Object target by double-clicking on the **Data Store** group in the left pane. The middle pane will display a list of all existing Data Store Objects in the current project. When such a list is displayed in the middle pane, the pane is identified as a Drop Target for new Data Store Objects.
2. Browse to the Data Warehouse via the **Browse > Data Warehouse** menu option.
3. Drag the Load or Stage table that contains the bulk of the Data Store Object columns into the middle pane. Drop the table anywhere in the pane.
4. The new object window appears and identifies the new object as a Data Store Object and provides a default name, based on the name of the selected Load or Stage table.
5. Either accept this name or type in the name of the Data Store Object and click **ADD** to proceed:



Add a New Metadata Object

Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: Data Store

Object Name: ds_customer

Target Location: DataWarehouse:RED_STAGE

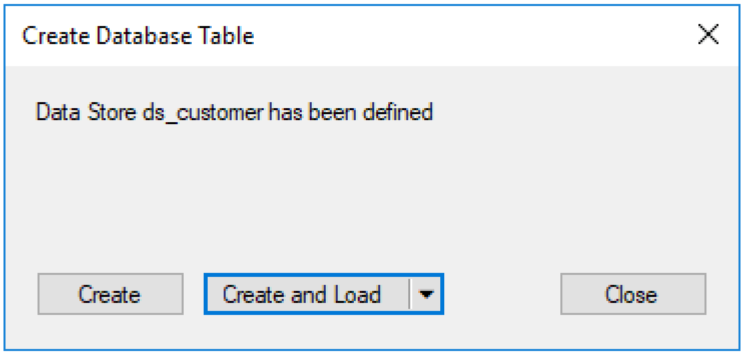
Data Store Object Properties

The table Properties window for the new table is displayed.

- If required, the Data Store Object can be changed to be a history table by choosing History from the table type drop-down list on the right side of the dialog. History tables are like slowly changing dimensions in dimensional data warehouses. Refer to **Building a Dimension** for details. Change the storage options if desired.
- If prototyping, and the Data Store Object is simple (e.g. one source table) then it is possible to create, load and update the Data Store Object in a couple of steps.
- If you want to do this, select the **(Build Procedure...)** option from the **Update Procedure** drop-down, and answer **Create and Load** to the next question.

Create and Load

If you chose to build the update procedure the following window appears, after clicking **OK** on the Properties window. It asks if you want to create the Data Store table in the database and execute the update procedure.

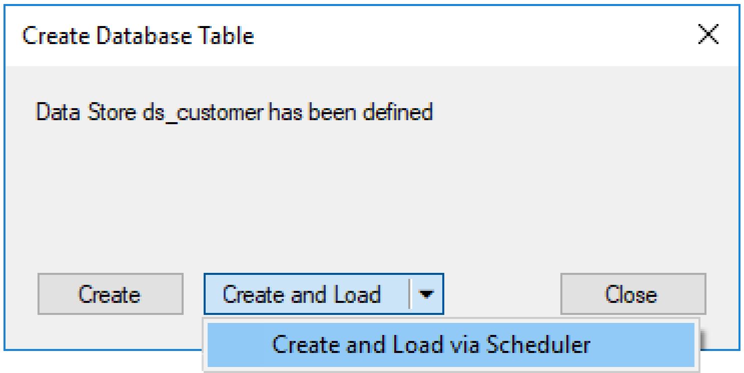


Data Store ds_customer has been defined

If **Create** or **Create and Load** is selected and a new procedure creation was chosen, proceed directly to the **Generating the Data Store Update Procedure**.
If you have additional columns to add or columns to delete then select **Close** and proceed as follows below.

Note

It is possible to create and load the table via the Scheduler; by selecting this option from the drop-down list on the **Create and Load** button:



Data Store ds_customer has been defined

Deleting and Changing columns

The columns defined for the Data Store Object is displayed in the middle pane.

- It is possible to delete any unwanted columns by highlighting a column name or a group of names and clicking **Delete**.
- The name of a column can also be changed by by right-clicking the column and choosing **Properties** to edit its properties.

- Any new name must conform to the database naming standards. Good practice is to use alphanumerics and the underscore character.
  Refer to **Data Store Column Properties** for a more details.

Tip:

When prototyping and in the initial stages of an analysis area build, it is best not to remove columns, nor to change their names to any great extent. This type of activity is best left, and done after end users have used the data and provided feedback.

Adding columns

With the columns of the Data Store Object displayed in the middle pane, this pane is considered a drop target for additional columns.

- It is a simple matter to select columns from other load and/or stage tables and drag these columns into the middle pane.
- The source table column in the middle pane shows where each column was dragged from.
- The column **description** could be acquired from three different tables. Best practice is to rename at least two of the columns, perhaps also adding context to the column name. For example, description could become group_description, and so forth.
- There are a number of WhereScape RED ancillary columns that do not have a source table.
  These columns have been added by WhereScape RED, and are added depending on earlier choices.

A description of these columns follows:

Column name	Description
dss_start_date	Used for history tables. This column provides a date time stamp when the Data Store Object record came into existence. It is used to ascertain which Data Store Object record should be used when multiple are available.
dss_end_date	Used for history tables. This column provides a date time stamp when the Data Store Object record ceased to be the current record. It is used to ascertain which Data Store Object record should be used when multiple are available.
dss_current_flag	Used for Data Store history tables. This flag identifies the current record where multiple versions exist.
dss_source_system_key	Added to support history tables that cannot be fully conformed, and the inclusion of subsequent source systems. See the ancillary settings section for more details.
dss_version	Used for Data Store history tables. This column contains the version number of a Data Store history tables record. Numbered from 1 upwards with the highest number being the latest or current version. It forms part of the unique constraint for the business key of a Data Store history tables.
dss_update_time	Indicates when the record was last updated in the data warehouse.
dss_create_time	Indicates when the record was first created in the data warehouse

Creating the table

Once the Data Store Object has been defined in the metadata, we need to physically create the table in the database.

1. This is done by right-clicking on the Data Store Object in the left pane and selecting **Create (ReCreate)** from the pop-up menu.

2. The Results pane displays the results of the creation. A message confirms that the Data Store Object was created. A copy of the actual database create statement and if defined, the results of any index create statements is listed. For the initial create, no indexes will be defined.
3. If the table was not created, then ascertain and fix the problem. A common problem is a 'Duplicate column' where a column has the same name in two of the source tables. The best way to find such a column is to double-click the list heading **Column Name**, which sorts the column names into alphabetical order.
4. Another double-click on the heading sorts the columns back into their create order.

The next section covers **Generating the Data Store Update Procedure**.

Generating the Data Store Update Procedure

Once a Data Store object has been defined in the metadata and created in the database, an update procedure can be generated to handle the joining of any tables and the update of the Data Store object.

Notes

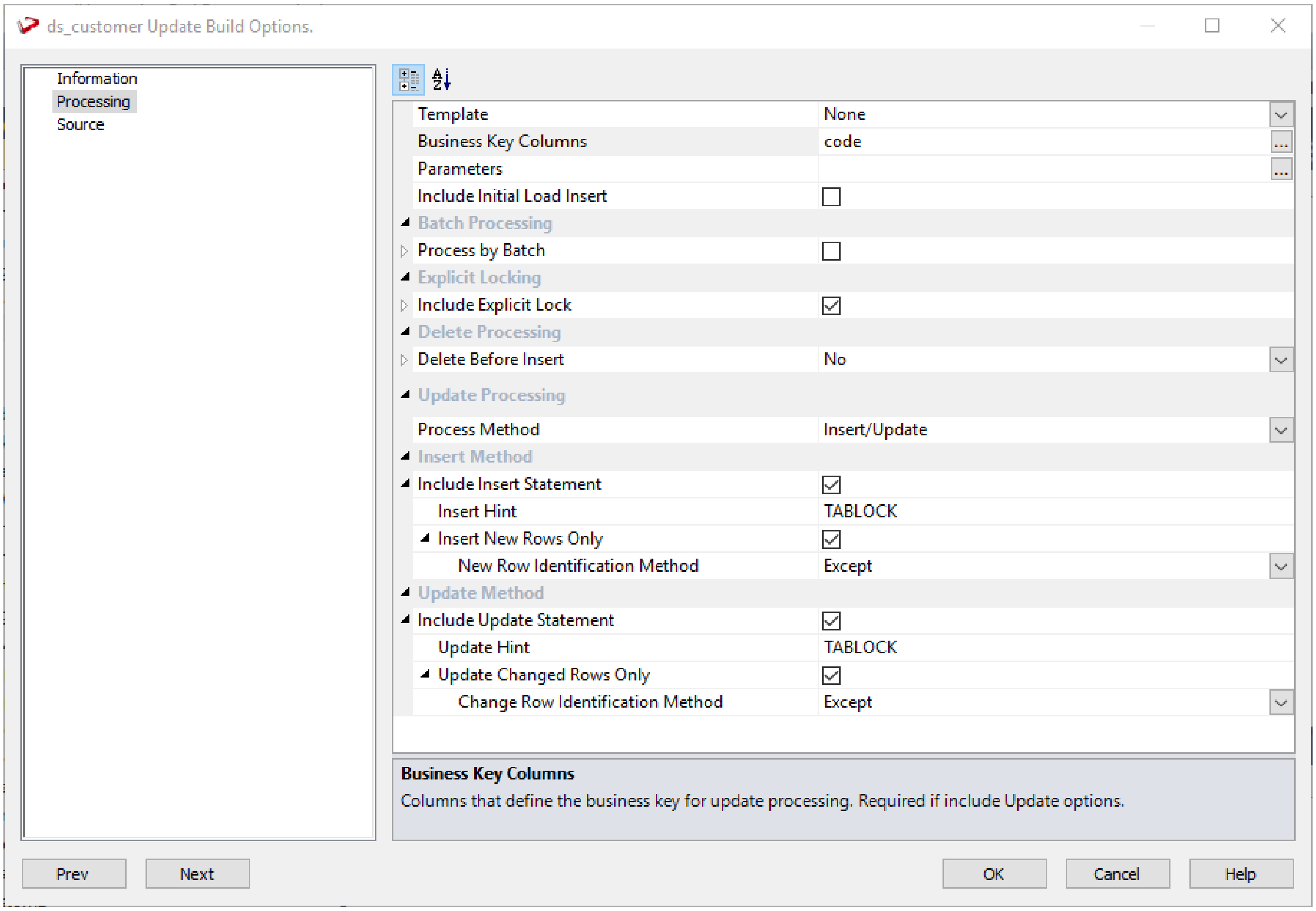
- You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.
- If the Data Store object is created in a custom database target, then an additional processing option is available—PowerShell script-based processing. Refer to **Script Templates for Custom Database Table Objects** for details.

Generating a Procedure

To generate a procedure:

1. Right-click the Data Store object in the left pane and select **Properties**.
2. Click the **Rebuild** button to start the process of generating the new procedure.
3. A series of options are presented.

Processing Tab



Template	None
Business Key Columns	code
Parameters	
Include Initial Load Insert	□
Batch Processing	
Process by Batch	□
Explicit Locking	
Include Explicit Lock	☑
Delete Processing	
Delete Before Insert	No
Update Processing	
Process Method	Insert/Update
Insert Method	
Include Insert Statement	☑
Insert Hint	TABLOCK
Insert New Rows Only	☑
New Row Identification Method	Except
Update Method	
Include Update Statement	☑
Update Hint	TABLOCK
Update Changed Rows Only	☑
Change Row Identification Method	Except

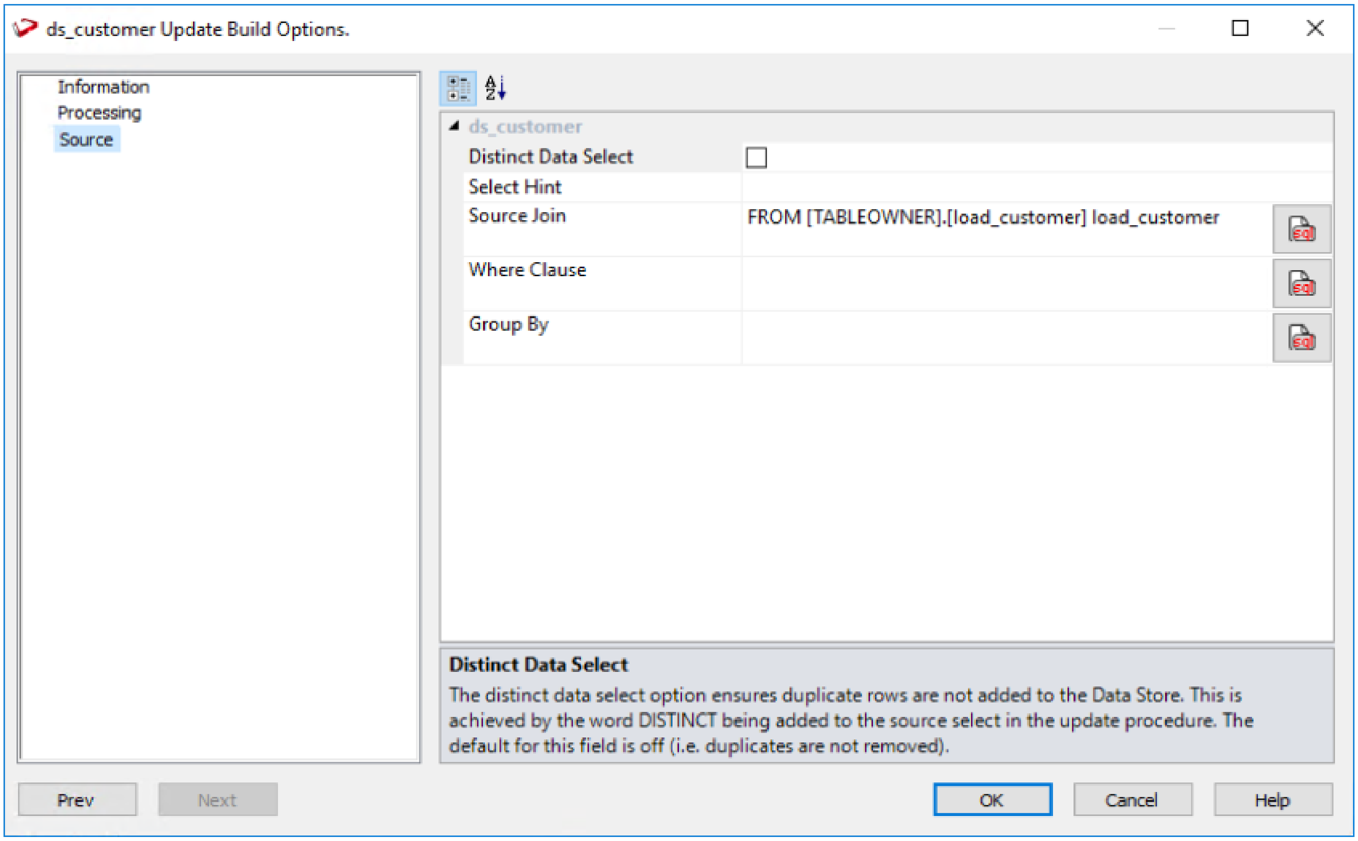
Business Key Columns
Columns that define the business key for update processing. Required if include Update options.

Fields	Description
Template	Enables you to generate update procedures via a **template**.
Business Key Columns	Columns that define the business key for update processing. This is required for include Update options. • Clicking on the ellipsis button will bring up the **Business Key** selection screen. • A business key can be made up of multiple columns, but it must provide a unique identifier. Where multiple columns separately uniquely identify rows in the Data Store object, choose one to act as the primary business key. For example, a source table may have a unique constraint on both a product code and a product description. Therefore, the description as well as the code must be unique. It is of course possible to combine the two columns, but the normal practice would be to choose the code as the business key. **Tip** • Use the column name ascending/descending buttons to sort column names. To revert to the meta column order, click on the meta column order button. • **NULL Values** - none of the columns chosen as the business key should ever contain a NULL value. See the note at the start of the Dimensions chapter.
Parameters	Any parameters selected are included in the generated update procedure as variables. The procedure will include code to retrieve the value of the parameter at run time and store it in the declared variable.

Fields	Description
	Clicking on the ellipsis button opens the **Parameters** selection screen. The variables can also be used in column transformations and in the from/where clause for the update procedure. Some databases have a 30 character limit for variable names. WhereScape RED ensures the variables added for any parameters are less than 30 characters long by creating variable names in the form v_ followed by the first 28 characters of the parameter name. For example, a parameter called MINIMUM_ORDER_NUMBER_SINCE_LAST_SOURCE_LOAD will be available as the variable v_MINIMUM_ORDER_NUMBER_SINCE_L. **Tip** • WhereScape RED parameters should be unique within the first 28 characters to avoid conflicting variables names. • If the desired parameter doesn't exist in the metadata yet, a new parameter can be added by clicking on the **Add New** button on the bottom leftmost corner of the Select Parameters dialog. • Refer to **Parameters** for more information.
Include Initial Load Insert	Adds an additional insert statement to the update procedure that runs if the target Data Store is empty. The benefit of this is improved performance inserting into an empty table without performing any checks to see if rows already exist. The default for this field is off (i.e. an initial insert statement is not added to the procedure).
Process by Batch	Allows users to select a column to drive data processing in a loop based on the distinct ordered values of the selected columns. The update procedure loops on this column and performs the delete, update and/or insert for each value. If the column chosen is a date datatype (date, datetime or timestamp), then the user is able to specify yearly, monthly, daily or column level looping. The default for this field is off (do not do batch processing).
Delete Before Insert	Allows selection of how to process deletes. It enables a delete statement to be added to the update procedure before any update or insert statement. This is a particularly useful option for purging old data and for updates based on a source system batch number.
Issue Warning if a Delete occurs	This option sets the procedure to a warning state if any deletes occur.
Delete Where Clause	The delete where clause is appended to the generated delete statement to constrain the rows deleted.
Process Method	Select between Insert/Update and Merge which enables you to use the Merge statement instead of two separate Insert and update statements.
Include Insert Statement	Includes the insert statement in the procedure to insert new rows in the Data Store.
Insert Hint	Enter a database hint to be used in the INSERT statement. This is a SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.
Insert New Rows Only	Uses change detection to work out what rows require inserting.
New Row Identification Method	Method used to identify that records in source are not currently recorded in the target table.
Existing Data Selection Hint	Database-compliant hint to be used for the existing data select statement.
Include Update Statement	Includes an update statement in the procedure to update changing rows in the Data Store. If this option is chosen, then the Update Changed rows only option is available.

Fields	Description
Include Hint	Enter a database hint to be used in the INCLUDE statement. This is a SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.
Update Changed Rows Only	Uses change detection to work out what rows require updating. Choosing this option, enables the **Change Row identification Method**.
Change Row Identification Method	Method used to identify that records in source have changed from what is currently recorded in the target table.
Existing Data Selection Hint	Database-compliant hint to be used for the existing data select statement.
Merge Hint	Enter a database hint to be used in the MERGE statement. This is a SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.

Source Tab



Distinct Data Select	□
Select Hint	
Source Join	FROM [TABLEOWNER].[load_customer] load_customer
Where Clause	
Group By	

Distinct Data Select

The distinct data select option ensures duplicate rows are not added to the Data Store. This is achieved by the word DISTINCT being added to the source select in the update procedure. The default for this field is off (i.e. duplicates are not removed).

Fields	Description
Distinct Data Select	Ensures duplicate rows are not added to the Data Store. This is achieved by the word DISTINCT being added to the source select in the update procedure. The default for this field is not set.
Source Join	The From clause, including Source Join information.
Where Clause	The Where clause.
Group By	The Group By clause.

Simple Join

A simple join only returns rows where data is matched in both tables. So for example if table A has 100 rows and table B has a subset of 24 rows. If all the rows in table B can be joined to table A then 24 rows will be returned. The other 76 rows from table A will not be returned.

Outer Join

The outer join returns all rows in the master table regardless of whether or not they are found in the second table. So if the example above was executed with table A as the master table then 100 rows would be returned. 76 of those rows would have null values for the table B columns.

Notes

- When WhereScape RED builds up an outer join, it needs to know which table is the master table, and which is subordinate. Select the join column from the master table first. In the example screen above the table 'load_order_header' has had its column chosen and the column for the table 'load_order_line' is currently being chosen. This will result in the 'load_order_header' table being defined as the master, as per the example statement above. The results of this example select are that a row will be added containing order information regardless of whether or not a corresponding load_order_line entry exists.
- When upgrading from a RED version previous to 6.8.2.0 and moving existing objects to a target location, all procedures that reference those objects will need to be rebuilt.
  Any **FROM** clauses will also need to be manually regenerated in order for the table references to be updated to the new [TABLEOWNER] form.

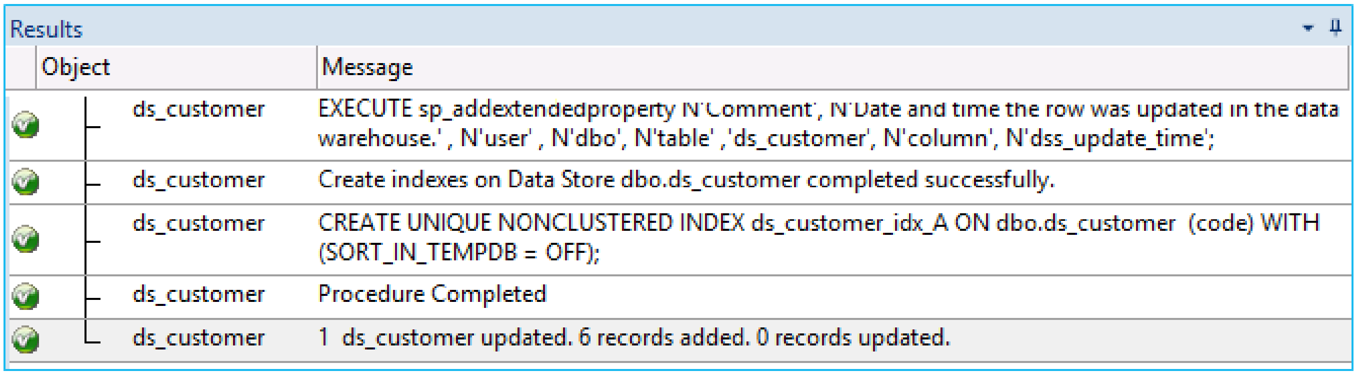
Building and Compiling the Procedure

- Once the relevant options are completed, click **OK**. The procedure is built and compiled.
- If the compile fails, an error is displayed along with the first few lines of error messages.
- Compile fails typically occur when the physical creation of the table was not done. If the compile fails for some other reason, the best approach is to use the procedure editor to edit and compile the procedure.
- The procedure editor highlights all the errors within the context of the procedure.
- Once the procedure has been successfully compiled, it can either be executed interactively or passed to the scheduler.

Indexes

By default, a number of indexes is created to support each Data Store object.

These indexes are added, once the procedure has been built. An example of the type of indexes created is as follows:



Results

	Object	Message
✓	ds_customer	EXECUTE sp_addextendedproperty N'Comment', N'Date and time the row was updated in the data warehouse.' , N'user' , N'dbo', N'table' ,'ds_customer', N'column', N'dss_update_time';
✓	ds_customer	Create indexes on Data Store dbo.ds_customer completed successfully.
✓	ds_customer	CREATE UNIQUE NONCLUSTERED INDEX ds_customer_idx_A ON dbo.ds_customer (code) WITH (SORT_IN_TEMPDB = OFF);
✓	ds_customer	Procedure Completed
✓	ds_customer	1 ds_customer updated. 6 records added. 0 records updated.

Additional indexes can be added, or these indexes changed. Refer to **Indexes** for details.

Data Store Artificial Keys

By default, Data Store Objects in WhereScape RED do not have an artificial (surrogate) key.

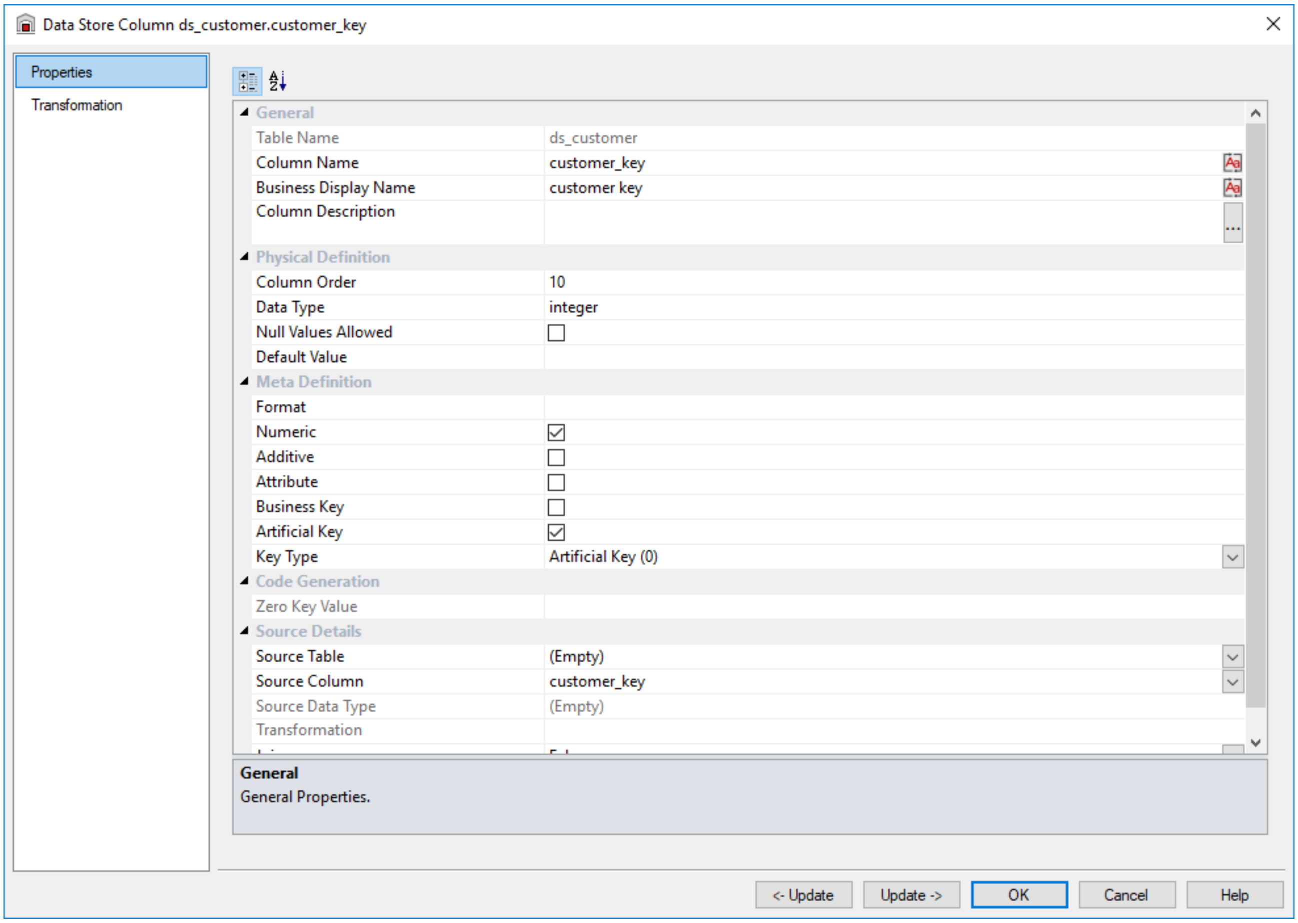
Artificial keys can be added manually but if needed could indicate Data Store Objects are not the correct WhereScape RED object for this table (perhaps an EDW 3NF Table would be more appropriate).

Edit the properties of the new column to have the correct name and order, source table and column, data type, key type and flags.

Specifically:

1. Right click a column in the middle pane and select either **Add Column** or **Duplicate Column** from the context menu.
2. Edit the properties of the new column to have the correct name and order, source table and column, data type, key type and flags as below:
   - The **Column Name** and **Source Column** must be the same.
   - The **Source Table** must be empty.
   - The **Data Type** should be:
     - o **DB2:** integer generated by default as identity (start with 1, increment by 1)
     - o **SQL Server:** integer identity(0,1)
   - The **Key Type** must be 0.
3. Only the **Numeric** and **Artificial Key** options should be enabled.

   The following example shows a manually added artificial key column:



Data Store Column ds_customer.customer_key

General	
Table Name	ds_customer
Column Name	customer_key
Business Display Name	customer key
Column Description	
Physical Definition	
Column Order	10
Data Type	integer
Null Values Allowed	□
Default Value	
Meta Definition	
Format	
Numeric	☑
Additive	□
Attribute	□
Business Key	□
Artificial Key	☑
Key Type	Artificial Key (0)
Code Generation	
Zero Key Value	
Source Details	
Source Table	(Empty)
Source Column	customer_key
Source Data Type	(Empty)
Transformation	

General
General Properties.

A WhereScape standard for the creation of special rows in the Data Store tables is as follows:

Key value	Usage
1 upwards	The standard artificial keys are numbered from 1 upwards, with a new number assigned

Key value	Usage
	for each distinct Data Store Object record.
0	Used as a join to the Data Store Object when no valid join existed. It is the convention in the WhereScape generated code that any EDW 3NF table business key that either does not exist or does not match is assigned to key 0.
-1 through -9	Used for special cases. The most common being where an EDW 3NF table is not appropriate for the record. A new key is used rather than 0 as we want to distinguish between records that are invalid and not appropriate.
-10 backward	Pseudo records. In many cases we have to deal with different granularities in our data. For example, we may have a table that contains actual sales at a product SKU level and budget information at a product group level. The product table only contains SKU based information. To be able to map the budget records to the same table, we need to create these pseudo keys that relate to product groups. The values -10 and backwards are normally used for such keys.

Note

To have a surrogate key auto added for Data Store tables, refer to **Global Naming of Key Columns**.

Data Store Column Properties

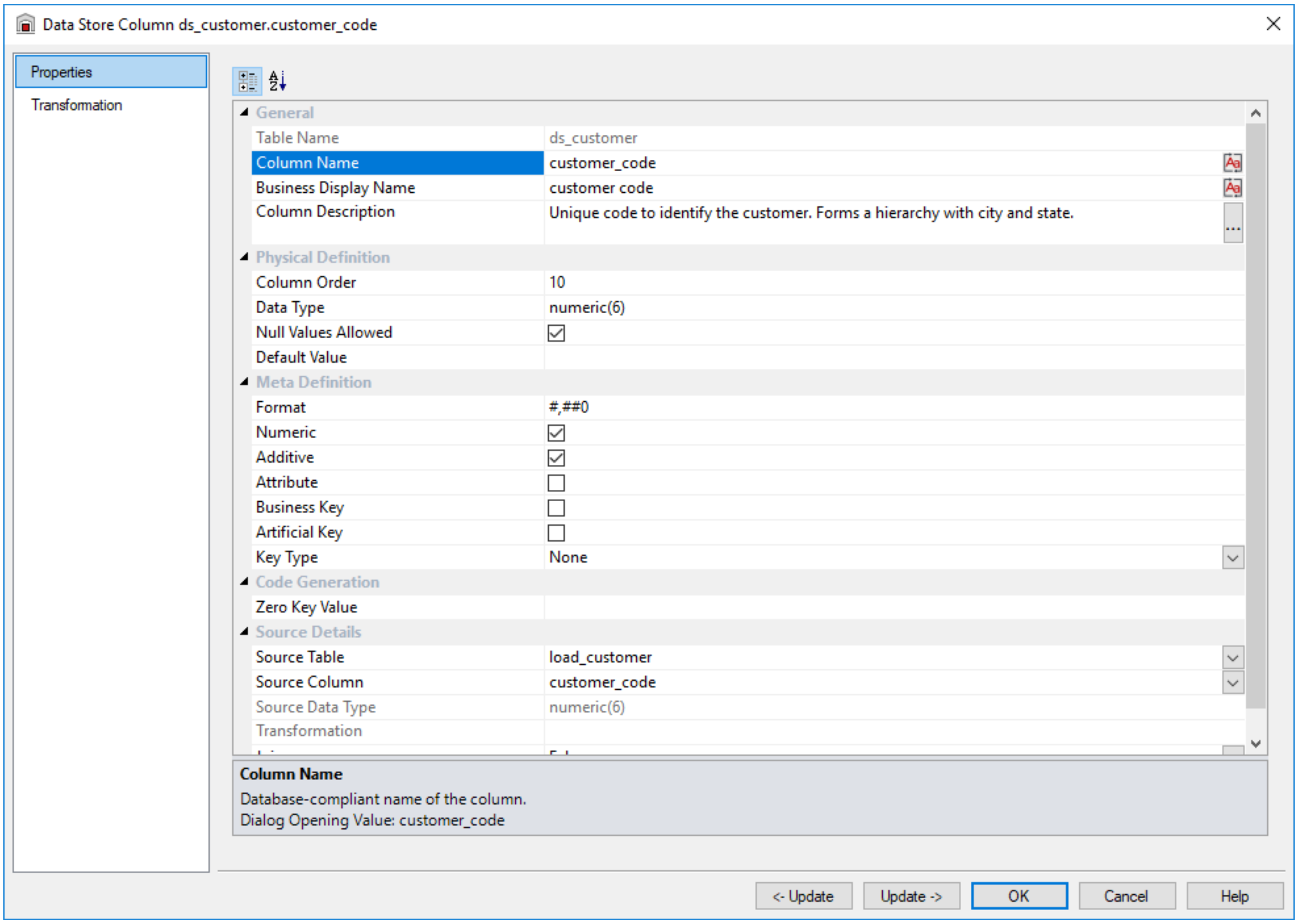
Each Data Store Object column has a set of associated properties. The definition of each property is described below:

- If the **Column name** or **Data type** is changed for a column then the metadata will differ from the table as recorded in the database.
- Use the **Validate/Validate Table Create Status** menu option or the right mouse menu to compare the metadata to the table in the database.
- A right mouse menu option of **Alter table** is available when positioned on the table name after the validate has completed. This option will alter the database table to match the metadata definition.

Tip

- If a database table's definition is changed in the metadata then the table will need to be altered in the database.
- Use the **Validate/Validate Table Create Status** to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table, through a pop-up menu option from the validated table name.

A sample **Properties** screen is as follows:



Data Store Column ds_customer.customer_code

General	
Table Name	ds_customer
Column Name	customer_code
Business Display Name	customer code
Column Description	Unique code to identify the customer. Forms a hierarchy with city and state.
Physical Definition	
Column Order	10
Data Type	numeric(6)
Null Values Allowed	☑
Default Value	
Meta Definition	
Format	#,##0
Numeric	☑
Additive	☑
Attribute	☐
Business Key	☐
Artificial Key	☐
Key Type	None
Code Generation	
Zero Key Value	
Source Details	
Source Table	load_customer
Source Column	customer_code
Source Data Type	numeric(6)
Transformation	

Column Name
Database-compliant name of the column.
Dialog Opening Value: customer_code

Tip

- The two special update keys allow you to update the column and step either forward or backward to the next column's properties.
- **ALT-Left Arrow** and **ALT-Right Arrow** can also be used instead of the two special update keys.

General

Options	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** The case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence of each conversion.
Business Display Name / Column Title	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note**

Options	Description
	The case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence of each conversion.
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col .** This field is also stored as a comment against the column in the database.

Physical Definition

Options	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. For SQL Server, common types are integer, numeric, varchar() and datetime. See the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.

Meta Definition

Options	Description
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.
Numeric	Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Additive	Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end

<table>
<tr><th>Options</th><th>Description</th></tr>
<tr><td></td><td>user layer. The use of this field is not relevant unless it can be utilized by the end user tools.</td></tr>
<tr><td><b>Attribute</b></td><td>Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or a invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables. This checkbox does not affect the physical table definition, but rather provides input to the view <b>ws_admin_v_dim_col</b> which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.</td></tr>
<tr><td><b>End User Layer Display</b></td><td>Indicates whether the table column is available/visible to end users. If set the documentation will include the column in the glossary and in the user documentation. It is also used to decide what columns appear in the view <b>ws_admin_v_dim_col</b>. Typically columns such as the artificial key would not be enabled for end user display.</td></tr>
<tr><td><b>Business Key</b></td><td>Indicates whether the column is part of the BUSINESS key, which is defined during the update procedure build. [Normally maintained automatically]. Multiple columns can form the primary business key.</td></tr>
<tr><td><b>Artificial Key</b></td><td>Indicates whether the column is the generated artificial/surrogate key (unique identifier) for the table. Only one artificial key per table is supported. [Normally maintained automatically].</td></tr>
<tr><td><b>Key Type</b></td><td>Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested.
The supported values are:
<table>
<tr><th>Key type</th><th>Meaning</th></tr>
<tr><td><b>0</b></td><td>The artificial key. Set when the key is added during drag and drop table generation.</td></tr>
<tr><td><b>1</b></td><td>Component of all business keys. Indicates that this column is used as part of any business key. For example: By default the dss_source_system_key is added to every dimension table. It is considered part of any lookup on that table and has the key type set to 1. Set when the column is added during drag and drop table generation.</td></tr>
<tr><td><b>2</b></td><td>Indicates that this column is a dimensional join. Used on fact tables to indicate the dimension keys.. Set during the update procedure generation for a fact table, based on information from the staging table.</td></tr>
<tr><td><b>3</b></td><td>Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation.</td></tr>
<tr><td><b>4</b></td><td>Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation.</td></tr>
<tr><td><b>5</b></td><td>Start date of a date ranged dimension. Used on dimension tables to indicate that the column is defined as the starting date for a source system date ranged dimension. Forms part of the business key. Set during the dimension creation.</td></tr>
<tr><td><b>6</b></td><td>End date of a date ranged dimension. Used on dimension tables to indicate that the column is defined as the ending date for a source system date ranged dimension. Forms part of the business key. Set during the dimension creation.</td></tr>
<tr><td><b>A</b></td><td>Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation.</td></tr>
<tr><td><b>B-Z</b></td><td>Indicates that the column is part of a secondary business key. Only used during</td></tr>
</table></td></tr>
</table>

Options	Description
	index generation and not normally set.

Code Generation

Options	Description
Zero Key Value	Determines the value populated for the column in the **Invalid Join** or **Unknown record**. By default, NULL is used when a value is not specified. All dimensions that use standard WhereScape RED generated procedures have a row with an artificial key of zero. This row is used to link to the fact records when no match on the dimension can be found. For example we may have some sales records that come through without a product defined. WhereScape RED by default associates these fact records with the zero key product dimension entry. Therefore, we might set the zero key value to 'Unknown product' for the name column in the product dimension.

Source Details

Options	Description
Source Table	Identifies the source table where the column's data comes from. This source table is normally a load table, or a dimension table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the **Source strategy** field. This field is used when generating a procedure to update the stage table. It is also used in the track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a load table column, which in turn may have been a transformation or the combination of multiple columns. This may also be a dimensional key where a dimension is being joined.
Source Data Type	Identifies the source column's data type. [Read-only].
Transformation	Refer to **Dimension Column Transformations** for details. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source table** and **Source column** fields will provide the dimension table's side of the join. The options for this field are: False, True, Manual and Pre Join. • Setting this field to Manual changes the way the Dimension table is looked up during the Stage table update procedure build. It enables you to join the dimension manually in the **Source Join** wizard (used to build the 'Where' clause). The usual dialog for matching the dimension business key to a column or columns in the Stage table is not displayed, if this option is enabled. • Setting this field to Pre Join activates the **Pre Join Source Table** field and enables you to select a table from the drop-down list.

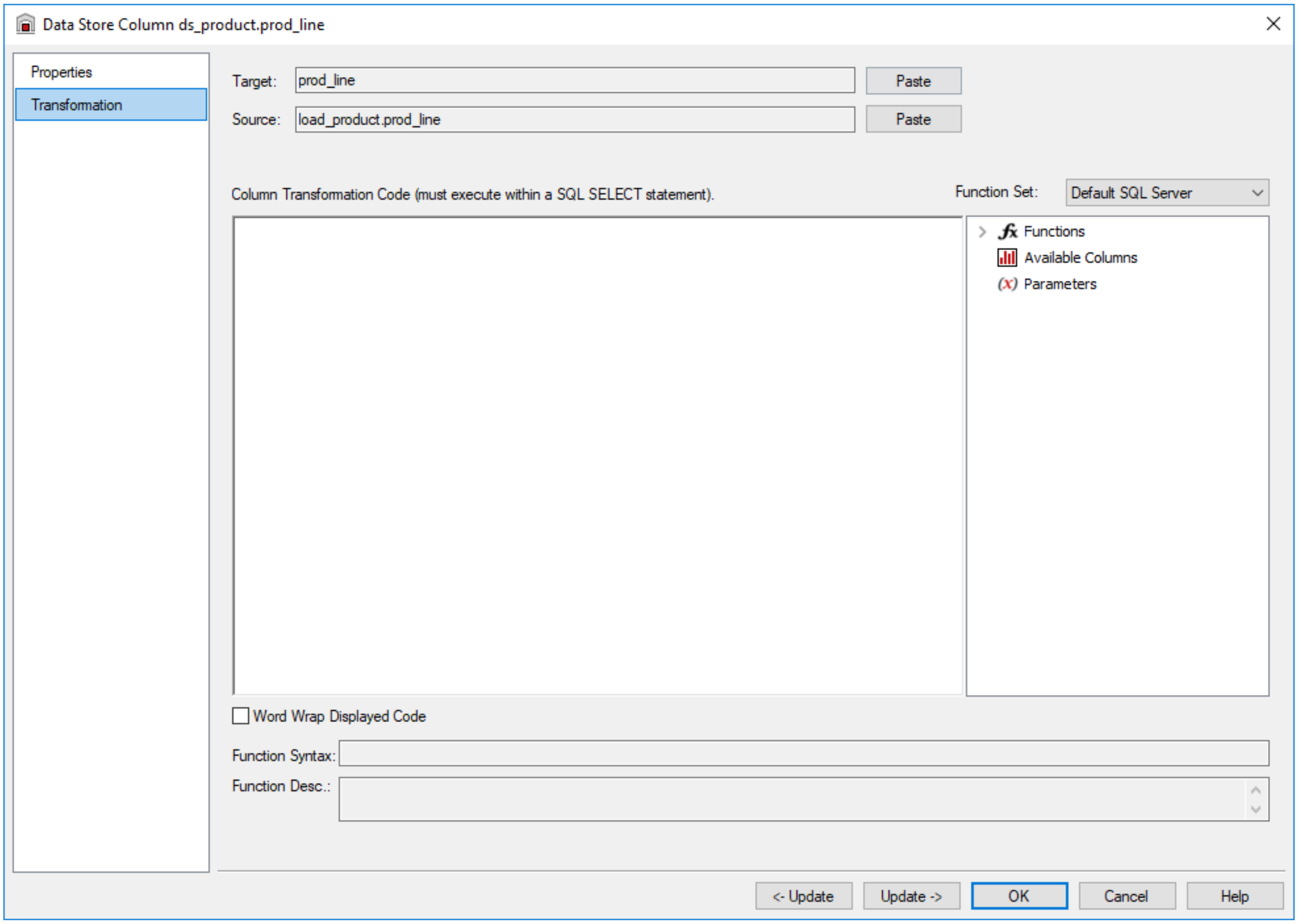
Data Store Object Column Transformations

Each Data Store Object column can have a transformation associated with it. The transformation is included in the generated procedure and is executed as part of the procedure update.
The transformation must therefore be a valid SQL construct that can be included in a **Select** statement.

For example, we could have a transformation of 'load_order_line.qty * 0.125' to calculate a tax column of 12.5%.

Click the **Transformation** tab to enter a transformation.

The transformation screen is as follows:


Data Store Column ds_product.prod_line

Target: prod_line

Source: load_product.prod_line

Column Transformation Code (must execute within a SQL SELECT statement).

Function Set: Default SQL Server

□ Word Wrap Displayed Code

Function Syntax:

Function Desc.:

Note

Transformations are only put into effect when the procedure is re-generated.

Refer to **Transformations** for details.

EDW 3NF Tables

Note

EDW 3NF/Normalized Table rename

- Former WhereScape RED Normalized Tables have been renamed to EDW 3NF from RED version 6.8.4.3. However, this change applies only for new metadata repositories, existing metadata repositories will not be affected and will not have its table's naming modified.
  All references to Normalized tables have been updated in the RED documentation from version 6.8.4.3 onwards. However, some screenshots of the RED left pane browser might still show instances of the Normalized object type instead of the new EDW 3NF type.
- To modify table naming from Normalized to EDW 3NF in existing repositories see **Object Type Names** and **Global Naming Conventions**.
  Please note that short name and table prefixes can be overwritten by the Local Naming conventions setting in the User Preferences. If this is the case, you can disable this option here: **Local Naming conventions**.

EDW 3NF Table Overview

An EDW 3NF Table is a data warehouse object used to build third normal form enterprise data warehouses.

In WhereScape RED, EDW 3NF objects have many of the code generating attributes of Stage, Dimension and Fact tables.

Third normal form enterprise data warehouses can be thought of as a source system for star schema data marts. Alternatively, they may be reported off directly by users and reporting tools.

EDW 3NF tables can be considered either reference or transactional in nature.

An EDW 3NF table is built from the Data Warehouse connection. Unless you are retrofitting an existing system, EDW 3NF Tables are typically built from one or more load or stage tables.

The EDW 3NF model may be retrofitted from an enterprise modeling tool. Refer to **Importing a Data Model** for details.

The usual steps for creating an EDW 3NF model are defined below and are covered in this chapter. The steps are:

1. Identify the source reference or transactional data that will constitute the EDW 3NF Table. If the data is sourced from multiple tables ascertain if a join between the source tables is possible, or if one or more intermediate stage (work) tables would be a better option.
2. Using the 'drag and drop' functionality drag the load or stage table that is the primary source of information for the EDW 3NF Table into an EDW 3NF target. Refer to **Building an EDW 3NF Table** for details.
3. If there's only one source table and all the columns from it are being used, you can select the auto create option to build and load the table. This automatically completes the next four steps. Refer to Building an EDW 3NF Table for details.
4. Add columns from other load and/or stage tables if required. Refer to **Building an EDW 3NF Table** for details.
5. Create the EDW 3NF Table in the database. Refer to **Building an EDW 3NF Table** for details.
6. Build the update procedure. Refer to **Generating the EDW 3NF Update Procedure** for details.
7. Run the update procedure and analyze the results.

If necessary, modify the update procedure or create a custom procedure.

EDW 3NF Table Keys

EDW 3NF Tables have two types of keys that we will refer to frequently. These are the Business Key and the Artificial Key. A definition of these two key types follows:

Business Key

The business key is the column or columns that uniquely identify a record within an EDW 3NF Table. Where the EDW 3NF Table maps back to a single or a main table in the source system, it is usually possible to ascertain the business key by looking at the unique keys for that source table. The business key is sometimes referred to as the 'natural' key. Examples of business keys are:

- The product SKU in a product table
- The customer code in a customer table
- The IATA airport code in an airport table.

It is assumed that business keys will never be NULL. If a null value is possible in a business key then the generated code will need to be modified to handle the null value by assigning some default value.

Note

Business keys are assumed to never be Null. If they can be null it is best to transform them to some value prior to the EDW 3NF or Stage table update.
If this is not done, an un-modified update procedure will probably fail with a duplicate key error on the business key index.

Artificial Key

By default, EDW 3NF Tables in WhereScape RED do not have an artificial key (artificial keys can be added manually, refer to **EDW 3NF Table Artificial Keys** for details.

An artificial key is the unique identifier that can be used to join an EDW 3NF Table record to other EDW 3NF Tables.
When joining EDW 3NF Tables it would be possible to perform the join using the business key. For EDW 3NF Tables that satisfy one of more of the following conditions, joining with business keys could result in slow query times and excessive use of database storage:

- Multiple column business keys (excessive storage and multiple column joins)
- One or more large character business key columns (excessive storage)
- Very large tables (excessive storage - integer artificial keys often use less space than one small character field)
- History EDW 3NF Tables (complex joins involving a between dates construct)

As query time is one of our key drivers in data warehouse implementations the best answer is often to use some form of artificial key.
A price is paid in the additional processing required doing key lookups, but this is offset by the reduced query times and reduced complexity.

The artificial key is an integer and is built sequentially from 1 upwards.

Refer to the section on artificial keys for a more detailed explanation. An artificial key is sometimes referred to as a "surrogate" key.

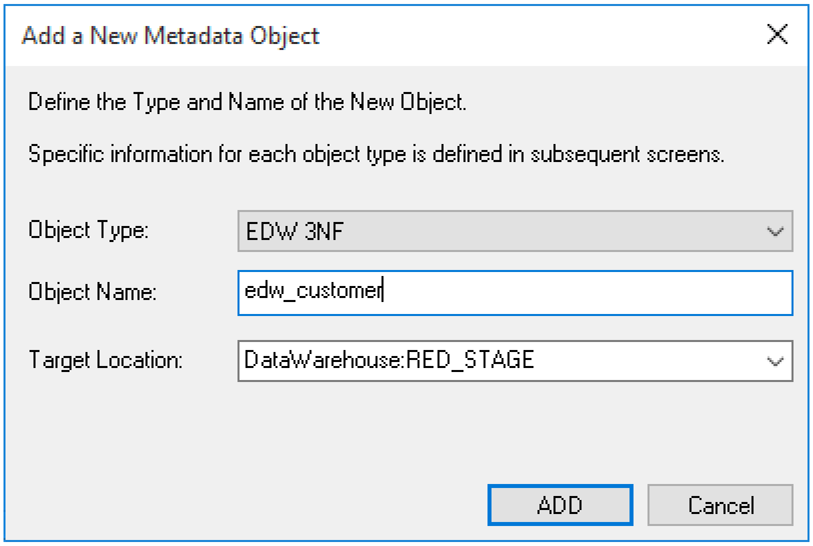
Building an EDW 3NF Table

EDW 3NF tables are often sourced from one table in the base application. The process for building an EDW 3NF table begins with the drag and drop of the Load or Stage table that contains the bulk of the EDW 3NF table's information.

Drag and Drop

1. Create an EDW 3NF table target by double-clicking the **EDW 3NF** object group in the left pane. The middle pane displays a list of all existing EDW 3NF tables in the current project. When this list is displayed in the middle pane, the pane is identified as a Drop Target for new EDW 3NF tables.

2. Browse to the Data Warehouse via the **Browse > Data Warehouse** menu option.
3. Drag the Load or Stage table, that contains the bulk of the EDW 3NF table columns, into the middle pane. Drop the table anywhere in the pane.
4. The new object window appears and identifies the new object as a EDW 3NF table and provides a default name, based on the name of the selected Load or Stage table.
5. Either accept this name or type in the name of the EDW 3NF table and click **ADD** to proceed:


Add a New Metadata Object

Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: EDW 3NF

Object Name: edw_customer

Target Location: DataWarehouse:RED_STAGE

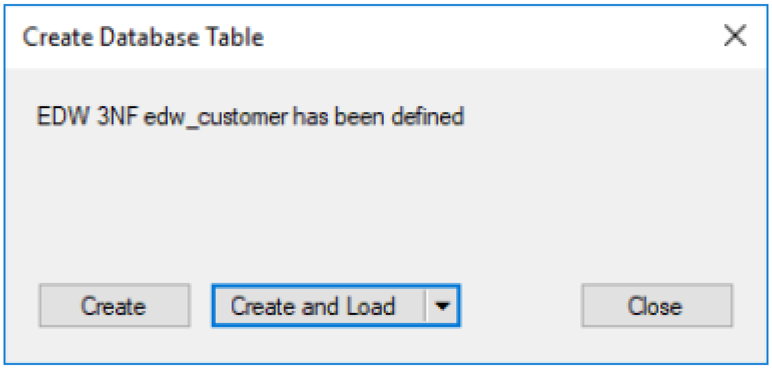
EDW 3NF Table Properties

The table Properties window for the new table is displayed.

- If required, the EDW 3NF table can be changed to be a history table by choosing History from the table type drop-down list on the right side of the dialog. History tables are like slowly changing dimensions in dimensional data warehouses. Refer to **Building a Dimension** for details. Change the storage options if desired.
- If prototyping, and the EDW 3NF table is simple (e.g. one source table) then it is possible to create, load and update the EDW 3NF table in a couple of steps.
- If you wish to do this, select the **(Build Procedure...)** option from the **Update Procedure** drop-down, and answer **Create and Load** to the next question.

Create and Load

If you chose to build the update procedure, the following window appears after clicking **OK** in the **Properties** window. It asks if you want to create the EDW 3NF table in the database and execute the update procedure.


Create Database Table

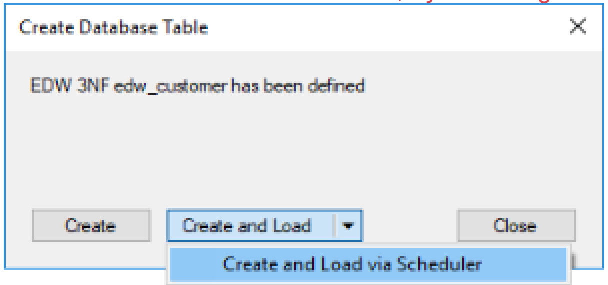
EDW 3NF edw_customer has been defined

If you are satisfied with the columns that will be used and do not wish to add any additional columns you can select the **Create and Load** button. Alternatively, the **Create** button creates the table in the repository but does not execute an update, enabling you to change columns before loading data into the table.

If **Create** or **Create and Load** is selected and a new procedure creation was chosen, proceed directly to the **Generating the EDW 3NF Update Procedure**.

Note

It is possible to create and load the table via the Scheduler; by selecting this option from the drop-down list on the **Create and Load** button:


Create Database Table

EDW 3NF edw_customer has been defined

If you have additional columns to add or columns to delete then select **Close** and proceed as follows below.

Deleting and Changing columns

The columns defined for the EDW 3NF table is displayed in the middle pane.

- It is possible to delete any unwanted columns by highlighting a column name or a group of names and clicking **Delete**.
- The name of a column can also be changed by right-clicking the column and choosing **Properties** to edit its properties.
- Any new name must conform to the database naming standards. Good practice is to use alphanumerics and the underscore character.
  Refer to **EDW 3NF Column Properties** for more details.

Tip:

When prototyping, and in the initial stages of an analysis area build, it is best not to remove columns, nor to change their names to any great extent. This type of activity is best left, and done after end users have used the data and provided feedback.

Adding columns

With the columns of the EDW 3NF table displayed in the middle pane, this pane is considered a Drop Target for additional columns.

- Select the columns from other Load and/or Stage tables and drag these columns into the middle pane. The source table column in the middle pane shows where each column originated.
- The column **description** could be acquired from three different tables. Best practice is to rename at least two of the columns, perhaps also adding context to the column name. For example, description could become group_description, and so forth.
- There are a number of WhereScapeRED ancillary columns that do not have a source table. These columns have been added by WhereScape RED and are added depending on earlier choices.
  A description of these columns follows:

Column name	Description
dss_start_date	Used for history tables. This column provides a date time stamp when the EDW 3NF table record came into existence. It is used to ascertain which EDW 3NF table record should be used when multiple tables are available.
dss_end_date	Used for history tables. This column provides a date time stamp when the EDW 3NF table record ceased to be the current record. It is used to ascertain which EDW 3NF table record

Column name	Description
	should be used when multiple tables are available.
dss_current_flag	Used for EDW 3NF history tables. This flag identifies the current record where multiple versions exist.
dss_source_system_key	Added to support history tables that cannot be fully conformed, and the inclusion of subsequent source systems. See the ancillary settings section for more details.
dss_version	Used for EDW 3NF history tables. This column contains the version number of an EDW 3NF history tables record. Numbered from 1 upwards with the highest number being the latest or current version. It forms part of the unique constraint for the business key of an EDW 3NF history tables.
dss_update_time	Indicates when the record was last updated in the data warehouse.
dss_create_time	Indicates when the record was first created in the data warehouse

Creating the table

Once the EDW 3NF table has been defined in the metadata, we need to physically create the table in the database.

1. This is done by right-clicking on the EDW 3NF table in the left pane and selecting **Create (ReCreate).**
2. The Results pane appears to show the results of the creation. The message confirms that the EDW 3NF table was created. A copy of the actual database create statement and if defined, the results of any index create statements is listed. For the initial create, no indexes will be defined.
3. If the table was not created, then ascertain and fix the problem. A common problem is a 'Duplicate column' where a column has the same name in two of the source tables. The best way to find such a column is to double-click the list heading **Col name**; which sorts the column names into alphabetical order.
4. Another double-click on the heading sorts the columns back into their create order.

Generating the EDW 3NF Update Procedure

Once an EDW 3NF object has been defined in the metadata and created in the database, an update procedure can be generated to handle the joining of any tables and the update of the EDW 3NF object.

Notes

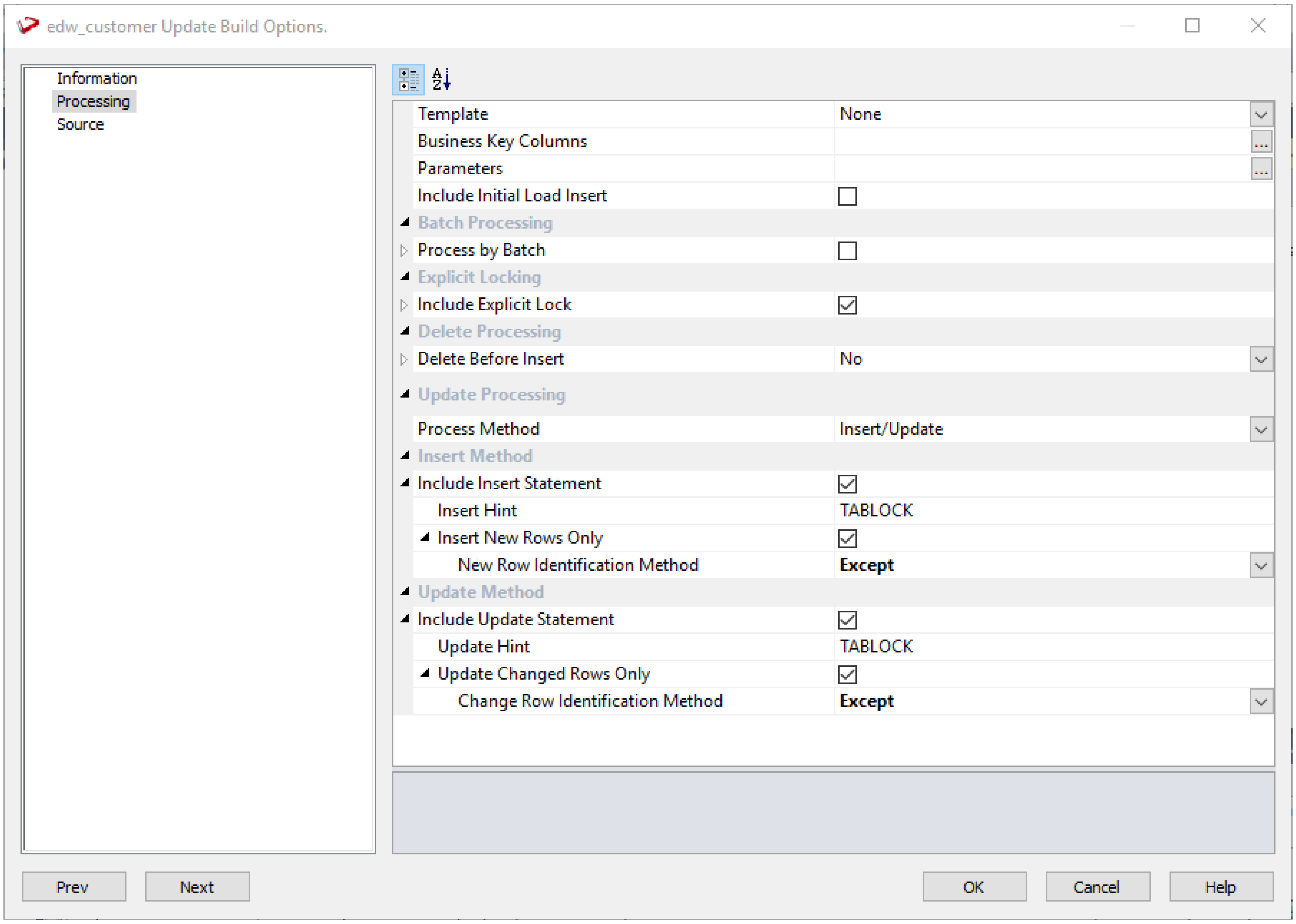
- You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.
- If the EDW 3NF object is created in a custom database target, then an additional processing option is available—PowerShell script-based processing. Refer to **Script Templates for Custom Database Table Objects** for details.

Generating a Procedure

To generate a procedure:

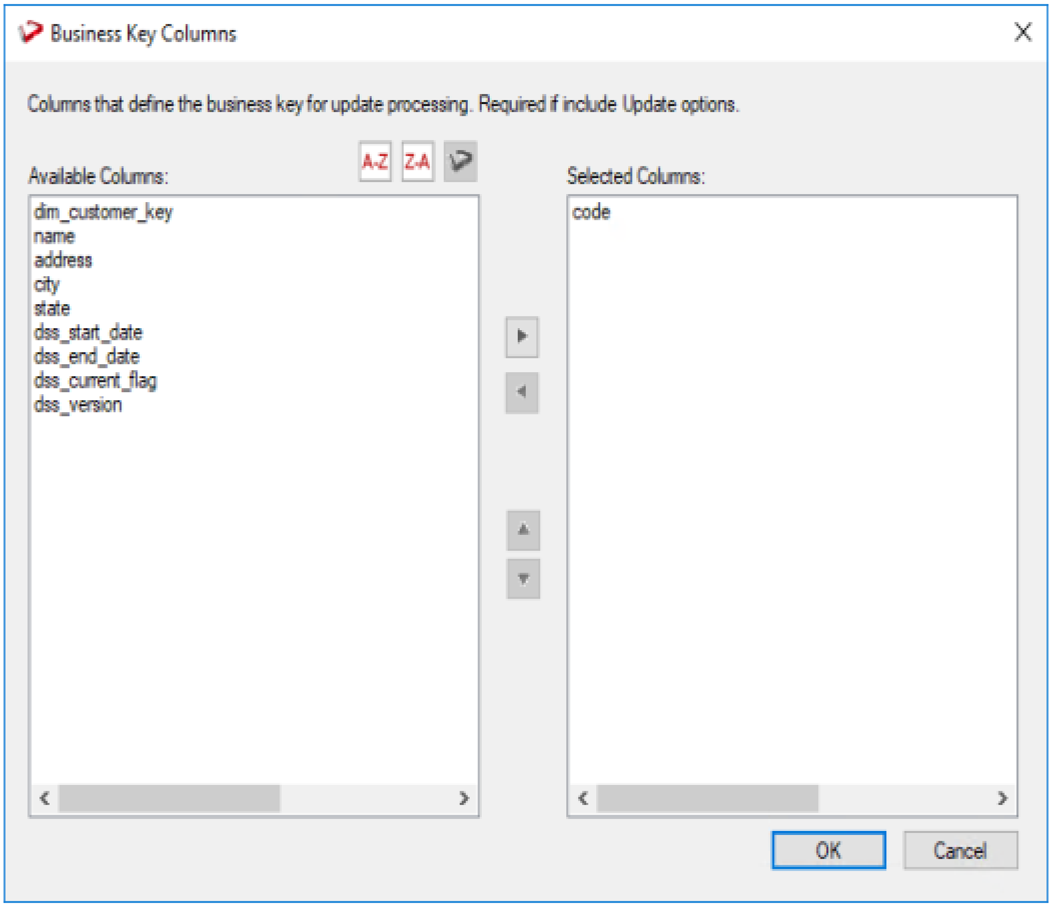
1. Right-click the EDW 3NF object in the left pane and select **Properties.**
2. Click the **Rebuild** button to start the process of generating the new procedure.
3. A series of options are available.

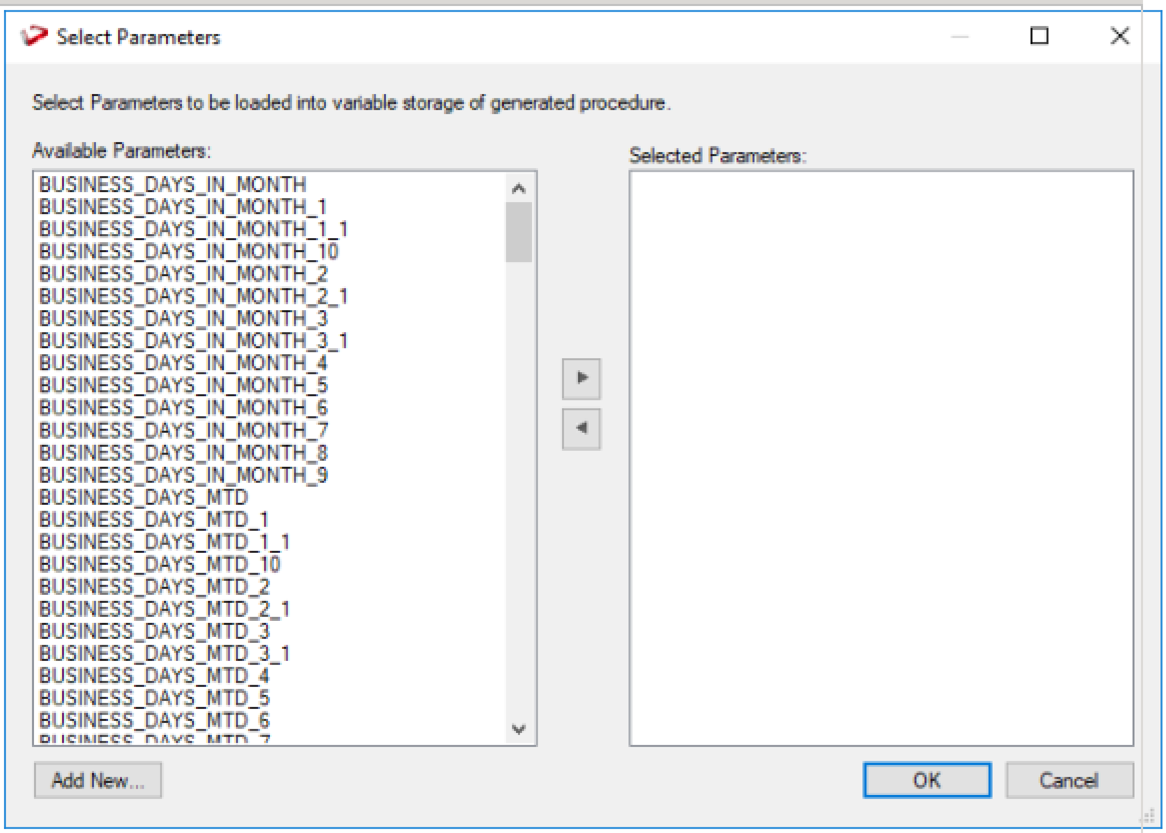
Processing Tab



Template	None
Business Key Columns	...
Parameters	...
Include Initial Load Insert	□
Batch Processing	
Process by Batch	□
Explicit Locking	
Include Explicit Lock	☑
Delete Processing	
Delete Before Insert	No
Update Processing	
Process Method	Insert/Update
Insert Method	
Include Insert Statement	☑
Insert Hint	TABLOCK
Insert New Rows Only	☑
New Row Identification Method	**Except**
Update Method	
Include Update Statement	☑
Update Hint	TABLOCK
Update Changed Rows Only	☑
Change Row Identification Method	**Except**

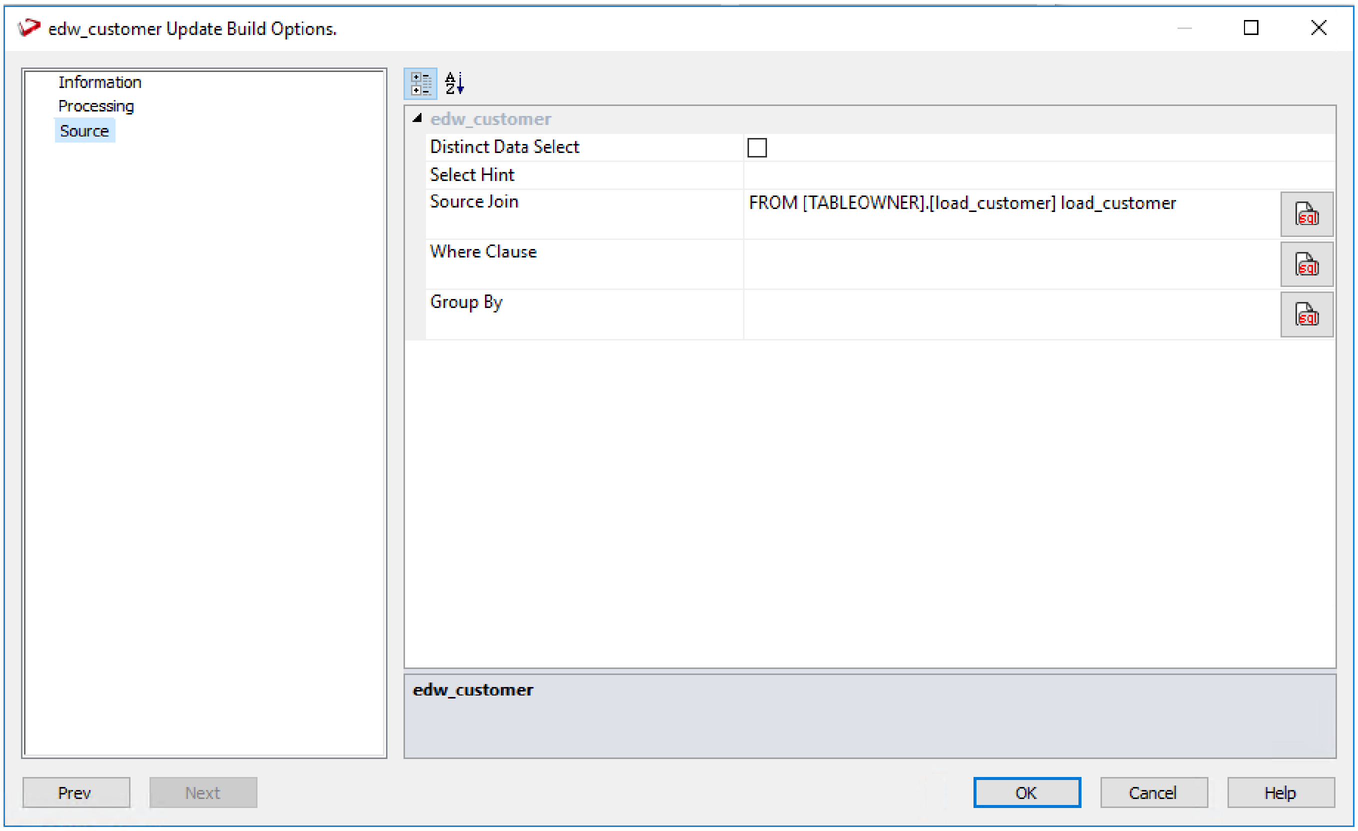
Options	Description
Template	Enables you to generate update procedures via a **template**.
Business Key Columns	Columns that define the business key for update processing. Required for include Update options. Clicking the ellipsis button brings up the **Business Key** selection screen.

Options	Description
	 Business Key Columns Columns that define the business key for update processing. Required if include Update options. Available Columns: dim_customer_key name address city state dss_start_date dss_end_date dss_current_flag dss_version Selected Columns: code A business key can be made up of multiple columns, but it must provide a unique identifier. Where multiple columns separately uniquely identify rows in the EDW 3NF Object, choose one to act as the primary business key. For example, a source table may have a unique constraint on both a product code and a product description. Therefore the description as well as the code must be unique. It is possible to combine the two columns, but the normal practice would be to choose the code as the business key. **Tips** • Use the column name ascending/descending buttons to sort column names. To revert to the meta column order, click on the meta column order button. • **NULL Values** - none of the columns chosen as the business key should ever contain a NULL value.
Parameters	Any parameters selected are included in the generated update procedure as variables. The procedure includes code to retrieve the value of the parameter at run time and store it in the declared variable. Clicking the ellipsis button brings up the **Parameters** selection screen.

Options	Description
	 Select Parameters Select Parameters to be loaded into variable storage of generated procedure. Available Parameters: BUSINESS_DAYS_IN_MONTH BUSINESS_DAYS_IN_MONTH_1 BUSINESS_DAYS_IN_MONTH_1_1 BUSINESS_DAYS_IN_MONTH_10 BUSINESS_DAYS_IN_MONTH_2 BUSINESS_DAYS_IN_MONTH_2_1 BUSINESS_DAYS_IN_MONTH_3 BUSINESS_DAYS_IN_MONTH_3_1 BUSINESS_DAYS_IN_MONTH_4 BUSINESS_DAYS_IN_MONTH_5 BUSINESS_DAYS_IN_MONTH_6 BUSINESS_DAYS_IN_MONTH_7 BUSINESS_DAYS_IN_MONTH_8 BUSINESS_DAYS_IN_MONTH_9 BUSINESS_DAYS_MTD BUSINESS_DAYS_MTD_1 BUSINESS_DAYS_MTD_1_1 BUSINESS_DAYS_MTD_10 BUSINESS_DAYS_MTD_2 BUSINESS_DAYS_MTD_2_1 BUSINESS_DAYS_MTD_3 BUSINESS_DAYS_MTD_3_1 BUSINESS_DAYS_MTD_4 BUSINESS_DAYS_MTD_5 BUSINESS_DAYS_MTD_6 BUSINESS_DAYS_MTD_7 Selected Parameters: Add New... OK Cancel The variables can also be used in column transformations and in the from/where clause for the update procedure. Some databases have a 30 character limit for variable names. WhereScape RED ensures the variables added for any parameters are less than 30 characters long by creating variable names in the form v_ followed by the first 28 characters of the parameter name. For example, a parameter called MINIMUM_ORDER_NUMBER_SINCE_LAST_SOURCE_LOAD will be available as the variable v_MINIMUM_ORDER_NUMBER_SINCE_L. **Tips** • WhereScape RED parameters should be unique within the first 28 characters to avoid conflicting variables names. • If the desired parameter doesn't exist in the metadata yet, a new parameter can be added by clicking on the Add New button on the bottom leftmost corner of the Select Parameters dialog. • Refer to **Parameters** for details.
Include Initial Load Insert	Adds an additional insert statement to the update procedure that runs if the target EDW 3NF object is empty. The benefit of this is improved performance inserting into an empty table without performing any checks to see if rows already exist. The default for this field is off (i.e. an initial insert statement is not added to the procedure).
Process by Batch	Enables you to select a column to drive data processing in a loop based on the distinct ordered values of the selected columns. The update procedure loops on this column and performs the delete, update and/or insert for each value. If the column chosen is a date data type (date, datetime or timestamp), then the user is able to specify yearly, monthly, daily or column level looping. The default for this field is off (do not do batch processing).
Delete Before Insert	Enables you to select how to process deletes. It enables a delete statement to be added to the update procedure before any update or insert statement. This is a particularly useful option for

Options	Description
	purging old data and for updates based on a source system batch number. If this option is selected, the following options are also available: • **Issue Warning if a Delete occurs:** this option sets the procedure to a warning state if any deletes occur. • **Delete Where Clause:** the delete where clause is appended to the generated delete statement to constrain the rows deleted.
Process Method	Select between Insert/Update and Merge which allows you to use the Merge statement instead of two separate Insert and update statements.
Include Insert Statement	Includes the insert statement in the procedure to insert new rows in the EDW 3NF Object. **Tip: Insert** Enter a database compliant hint to be used in the INSERT statement. This is a SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.
Insert New Rows Only	Uses change detection to work out what rows require inserting.
New Row Identification Method	Method used to identify that records in source are not currently recorded in the target table.
Existing Data Selection Hint	Database-compliant hint to be used for the existing data select statement.
Include Update Statement	Includes an update statement in the procedure to update changing rows in the EDW 3NF Object. If this option is chosen, then the **Update Changed rows** only option is available. **Tip: Update** Enter a database compliant hint to be used in the UPDATE statement. This is a SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.
Update Changed Rows Only	Uses change detection to work out what rows require updating. Choosing this option, enables the **Change Row identification Method**.
Change Row Identification Method	Method used to identify that records in source have changed from what is currently recorded in the target table.
Existing Data Selection Hint	Database-compliant hint to be used for the existing data select statement. **Tip: Merge** Enter a database hint to be used in the MERGE statement. This is a SQL Server only option. Defaults can be configured in **Tools > Options > Default Update Procedure Options**.

Source Tab



edw_customer

Distinct Data Select	□
Select Hint	
Source Join	FROM [TABLEOWNER].[load_customer] load_customer
Where Clause	
Group By	

Options	Description
Distinct Data Select	Ensures duplicate rows are not added to the EDW 3NF Object. This is achieved by the word DISTINCT being added to the source select in the update procedure. This not selected by default. Select Hint: A database compliant hint to be used in the source Select statement.
Source Join	The From clause, including Source Join information.
Where Clause	The Where clause.
Group By	The Group By clause.

Simple Join

A simple join only returns rows where data is matched in both tables. So for example if table A has 100 rows and table B has a subset of 24 rows. If all the rows in table B can be joined to table A then 24 rows will be returned. The other 76 rows from table A will not be returned.

Outer Join

The outer join returns all rows in the master table regardless of whether or not they are found in the second table. So if the example above was executed with table A as the master table then 100 rows would be returned. 76 of those rows would have null values for the table B columns.

Notes

- When WhereScape RED builds up an outer join, it needs to know which table is the master table and which is subordinate. Select the join column from the master table first. In the example screen above the table 'load_order_header' has had its column chosen and the column for the table 'load_order_line' is currently being chosen. This will result in the 'load_order_header' table being defined as the master, as per the example statement above. The results of this example select are that a row is added containing order information, regardless of whether or not a corresponding load_order_line entry exists.

Notes

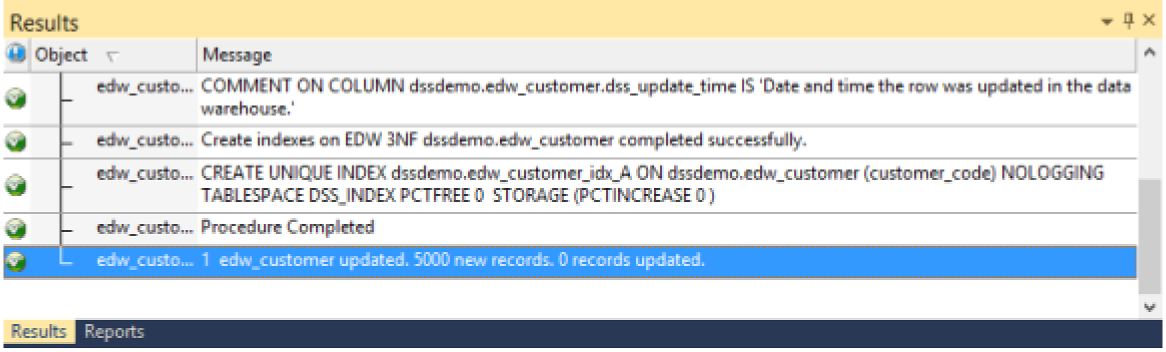
- When upgrading from a RED version previous to 6.8.2.0 and moving existing objects to a target location, all procedures that reference those objects will need to be rebuilt.
Any **FROM** clauses will also need to be manually regenerated in order for the table references to be updated to the new [TABLEOWNER] form.

Building and Compiling the Procedure

- Once the relevant options are completed, click **OK**. The procedure is built and compiled.
- If the compile fails an error is displayed along with the first few lines of error messages. Compile fails typically occur when the physical creation of the table was not done.
- If the compile fails for some other reason the best approach is to use the procedure editor to edit and compile the procedure. The procedure editor highlights all the errors within the context of the procedure.
- Once the procedure has been successfully compiled, it can either be executed interactively or passed to the scheduler.

Indexes

By default, a number of indexes will be created to support each EDW 3NF table. These indexes are added once the procedure has been built.
An example of the type of indexes created is as follows:



Object	Message
edw_custo...	COMMENT ON COLUMN dssdemo.edw_customer.dss_update_time IS 'Date and time the row was updated in the data warehouse.'
edw_custo...	Create indexes on EDW 3NF dssdemo.edw_customer completed successfully.
edw_custo...	CREATE UNIQUE INDEX dssdemo.edw_customer_idx_A ON dssdemo.edw_customer (customer_code) NOLOGGING TABLESPACE DSS_INDEX PCTFREE 0 STORAGE (PCTINCREASE 0)
edw_custo...	Procedure Completed
edw_custo...	1 edw_customer updated. 5000 new records. 0 records updated.

Additional indexes can be added, or these indexes changed. Refer to Indexes for more details.

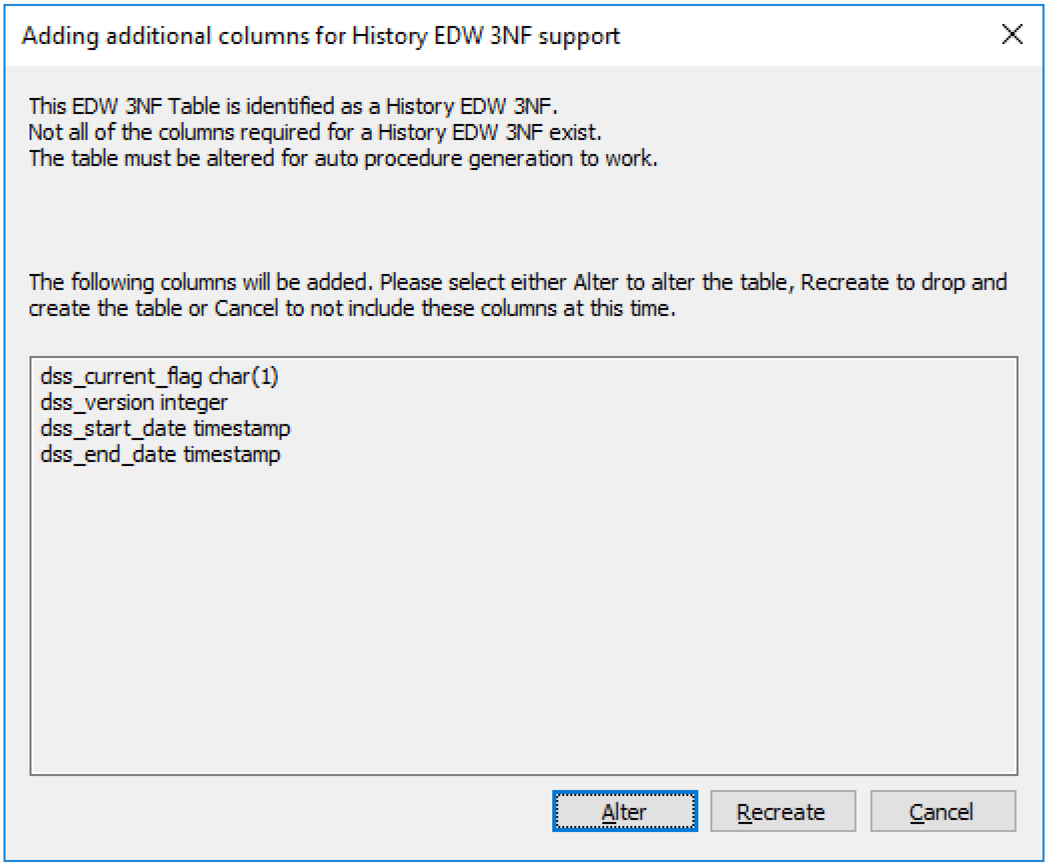
Converting an existing EDW 3NF Table to an EDW 3NF History Table

To convert an EDW 3NF table to an EDW 3NF history table:

- Change the table type to **History** in the Properties screen.
- Select **(Build Procedure...)** from the **Update Procedure** drop-down list and click **OK**.

If the existing EDW 3NF table is **NOT** to be dropped and recreated, then the following process must be followed:

1. Click **OK** in the **Procedure Type** window.
2. Click **OK** in the **Business Key** window.
3. Click **Alter** in the **Adding Additional Columns** window.

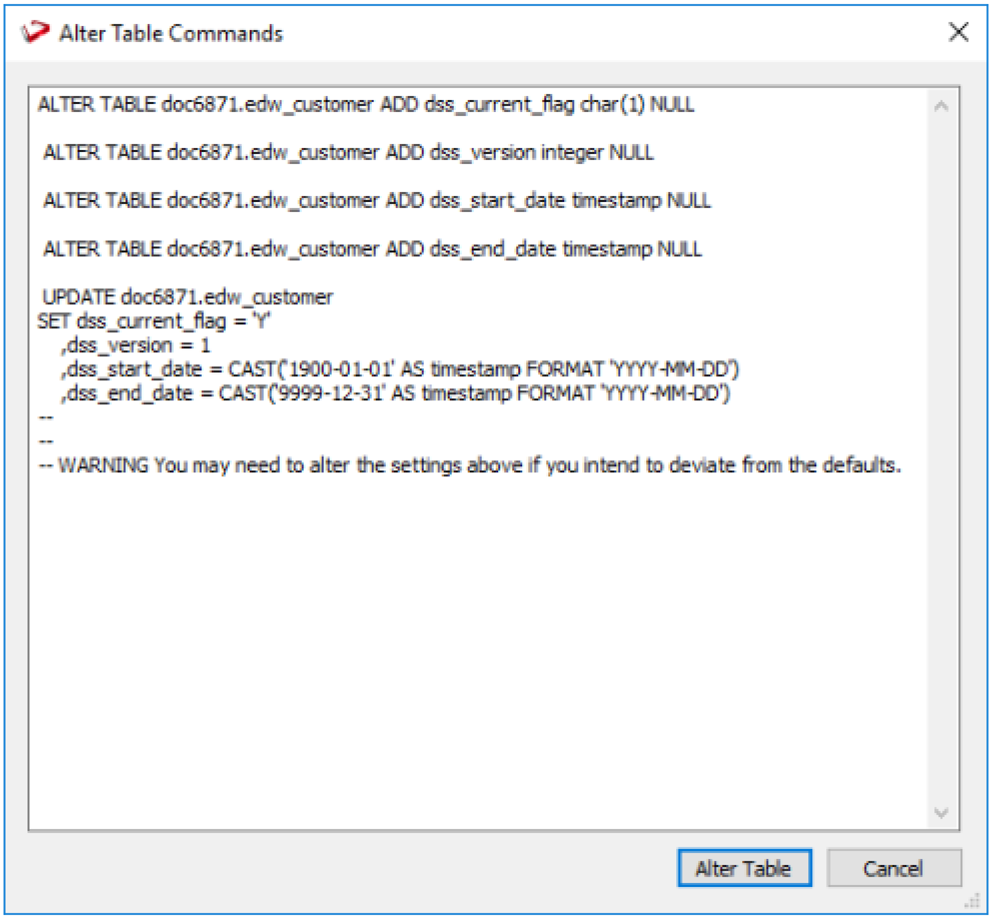


This EDW 3NF Table is identified as a History EDW 3NF.
Not all of the columns required for a History EDW 3NF exist.
The table must be altered for auto procedure generation to work.

The following columns will be added. Please select either Alter to alter the table, Recreate to drop and create the table or Cancel to not include these columns at this time.

```
dss_current_flag char(1)
dss_version integer
dss_start_date timestamp
dss_end_date timestamp
```

4. Click **Alter Table** on the **Alter Table Commands** window:



```
ALTER TABLE doc6871.edw_customer ADD dss_current_flag char(1) NULL

 ALTER TABLE doc6871.edw_customer ADD dss_version integer NULL

 ALTER TABLE doc6871.edw_customer ADD dss_start_date timestamp NULL

 ALTER TABLE doc6871.edw_customer ADD dss_end_date timestamp NULL

 UPDATE doc6871.edw_customer
SET dss_current_flag = 'Y'
   ,dss_version = 1
   ,dss_start_date = CAST('1900-01-01' AS timestamp FORMAT 'YYYY-MM-DD')
   ,dss_end_date = CAST('9999-12-31' AS timestamp FORMAT 'YYYY-MM-DD')
--
--
-- WARNING You may need to alter the settings above if you intend to deviate from the defaults.
```

Note

The SQL statement in this window can be edited if required.

5. Click **OK** on all remaining windows.

EDW 3NF Table Artificial Keys

By default, EDW 3NF tables in WhereScape RED do not have an artificial (surrogate) key. Artificial keys can be added manually.

The quickest way to do this is to add an extra column to the EDW 3NF table by using either **Add Column** or **Copy Column**.

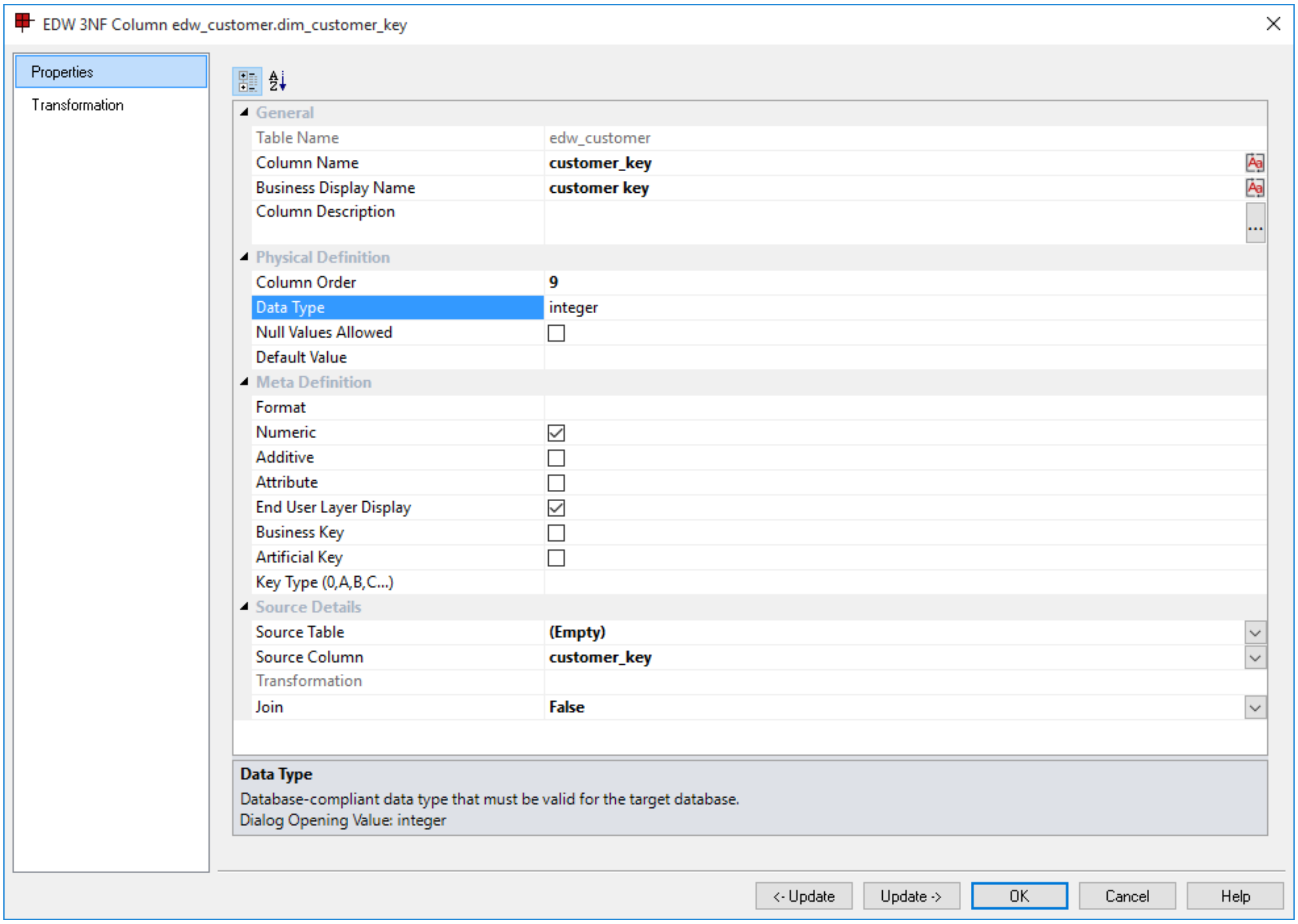
Edit the properties of the new column to have the correct name and order, source table and column, datatype, key type and flags.

Specifically:

- The **Column Name** and **Source Column** should be the same.
- The **Source Table** should be empty.
- The **Data Type** should be:
- DB2: integer generated by default as identity (start with 1, increment by 1)
- SQL Server: integer identity(0,1)
- The **Key Type** should be 0.

- Only the **Numeric** and **Artificial Key** flags must be set.

The following example shows a manually added artificial key column:



EDW 3NF Column edw_customer.dim_customer_key

General

Table Name	edw_customer
Column Name	**customer_key**
Business Display Name	**customer key**
Column Description	

Physical Definition

Column Order	**9**
Data Type	integer
Null Values Allowed	□
Default Value	

Meta Definition

Format	
Numeric	☑
Additive	□
Attribute	□
End User Layer Display	☑
Business Key	□
Artificial Key	□
Key Type (0,A,B,C...)	

Source Details

Source Table	**(Empty)**
Source Column	**customer_key**
Transformation	
Join	**False**

Data Type
Database-compliant data type that must be valid for the target database.
Dialog Opening Value: integer

This artificial key normally, and by default, starts at one and progresses as far as is required.

A WhereScape standard for the creation of special rows in the EDW 3NF tables is as follows:

Key value	Usage
1 upwards	The standard artificial keys are numbered from 1 upwards, with a new number assigned for each distinct EDW 3NF table record.
0	Used as a join to the EDW 3NF table when no valid join existed. It is the convention in the WhereScape generated code that any EDW 3NF table business key that either does not exist or does not match is assigned to key 0.
-1 through -9	Used for special cases. The most common being where an EDW 3NF table is not appropriate for the record. A new key is used rather than 0 as we want to distinguish between records that are invalid and not appropriate.
-10 backward	Pseudo records. In many cases, we have to deal with different granularities in our data. For example, we may have a table that contains actual sales at a product SKU level and budget information at a product group level. The product table only contains SKU based information. To be able to map the budget records to the same table, we need to create these pseudo keys that relate to product groups. The values -10 and backwards are normally used for such keys.

Note

Note

To have a surrogate key auto added for EDW 3NF tables, refer to **Global Naming of Key Columns** for details.

EDW 3NF Table Column Properties

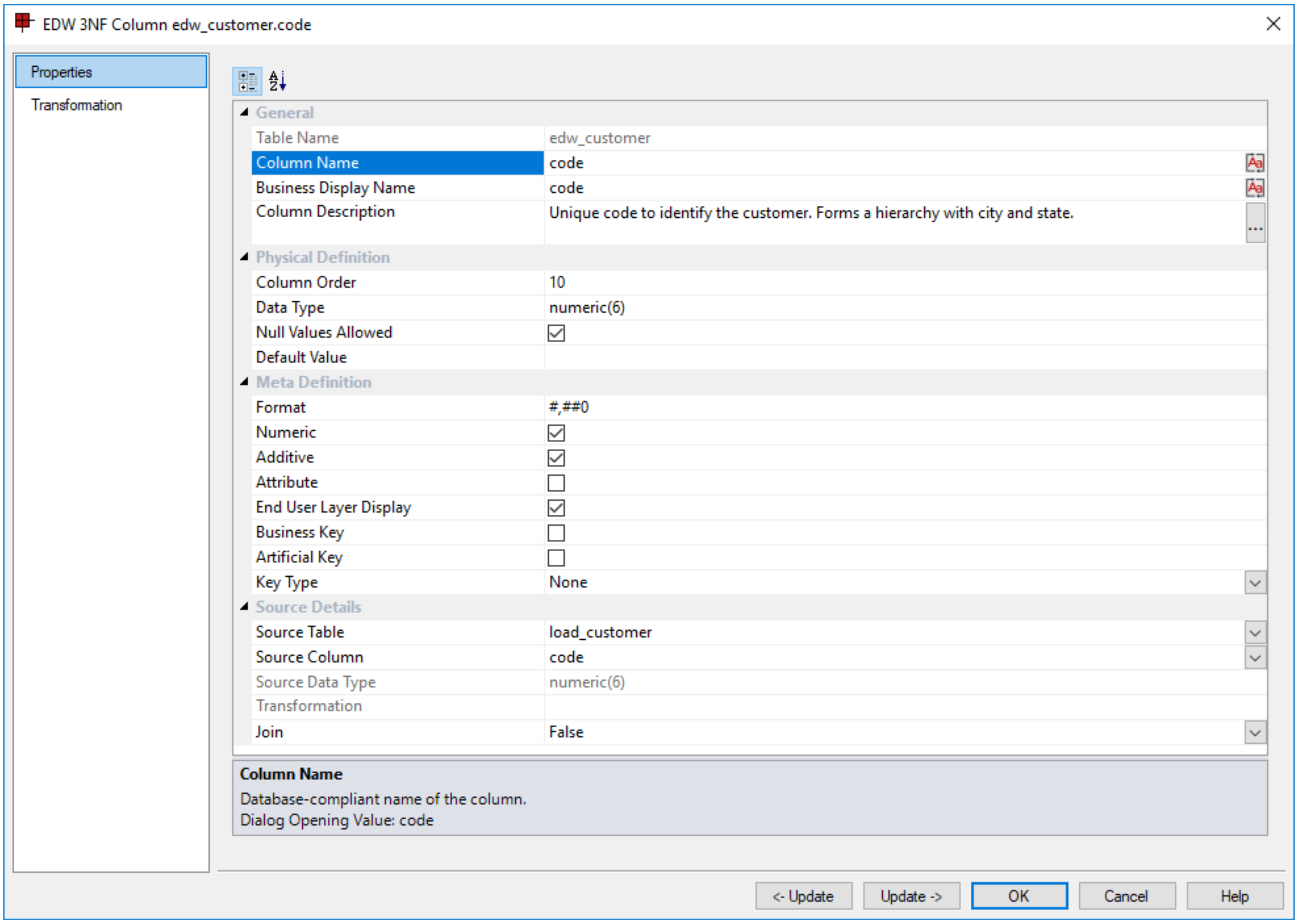
Each EDW 3NF table column has a set of associated properties. The definition of each property is described below:

- If the **Column name** or **Data type** is changed for a column then the metadata will differ from the table as recorded in the database.
- Use the **Validate > Validate Table Create Status** menu option or the right-click menu to compare the metadata to the table in the database.
- A right-click menu option of **Alter table** is available when positioned on the table name after the validate has completed. This option will alter the database table to match the metadata definition.

Tip:

If a database table's definition is changed in the metadata then the table will need to be altered in the database. Use the **Validate > Validate Table Create Status** to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table, through a pop-up menu option from the validated table name.

Sample EDW 3NF **Properties** screen:



EDW 3NF Column edw_customer.code

General

Table Name	edw_customer
Column Name	code
Business Display Name	code
Column Description	Unique code to identify the customer. Forms a hierarchy with city and state.

Physical Definition

Column Order	10
Data Type	numeric(6)
Null Values Allowed	☑
Default Value	

Meta Definition

Format	#,##0
Numeric	☑
Additive	☑
Attribute	□
End User Layer Display	☑
Business Key	□
Artificial Key	□
Key Type	None

Source Details

Source Table	load_customer
Source Column	code
Source Data Type	numeric(6)
Transformation	
Join	False

Column Name
Database-compliant name of the column.
Dialog Opening Value: code

Tip:

The two special update keys allow you to update the column and step either forward or backward to the next column's properties.
ALT-Left Arrow and **ALT-Right Arrow** can also be used instead of the two special update keys.

General

Field	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note:** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Business Display Name / Column Title	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col**. This field is also stored as a comment against the column in the database.

Physical Definition

Field	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required
Data Type	Database-compliant data type that must be valid for the target database. For SQL Server common types are integer, numeric, varchar() and datetime. See the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always

Field	Description
	mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.

Meta Definition

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td><b>Format</b></td><td>Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.</td></tr>
<tr><td><b>Numeric</b></td><td>Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view <b>ws_admin_v_dim_col</b> which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.</td></tr>
<tr><td><b>Additive</b></td><td>Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view <b>ws_admin_v_dim_col</b> which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.</td></tr>
<tr><td><b>Attribute</b></td><td>Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or an invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables. This checkbox does not affect the physical table definition, but rather provides input to the view <b>ws_admin_v_dim_col</b> which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.</td></tr>
<tr><td><b>End User Layer Display</b></td><td>Indicates whether the table column is available/visible to end users. If set the documentation will include the column in the glossary and in the user documentation. It is also used to decide what columns appear in the view <b>ws_admin_v_dim_col.</b> Typically columns such as the artificial key would not be enabled for end user display</td></tr>
<tr><td><b>Business Key</b></td><td>Indicates whether the column is part of the BUSINESS key, which is defined during the update procedure build. [Normally maintained automatically]. Multiple columns can form the primary business key.</td></tr>
<tr><td><b>Artificial Key</b></td><td>Indicates whether the column is the artificial key. Only one artificial key is supported. This indicator is set by WhereScape RED during the initial drag and drop creation of a table, and should not normally be altered.</td></tr>
<tr><td><b>Key Type</b></td><td>Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested.
The supported values are:
<table>
<tr><th>Key type</th><th>Meaning</th></tr>
<tr><td><b>0</b></td><td>The artificial key. Set when the key is added during drag and drop table generation.</td></tr>
<tr><td><b>1</b></td><td>Component of all business keys. Indicates that this column is used as part of any business key.</td></tr>
<tr><td><b>A</b></td><td>Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation.</td></tr>
</table>
</td></tr>
</table>

Field	Description	
	B-Z	Indicates that the column is part of a secondary business key. Only used during index generation and not normally set.

Source Details

Field	Description
Source Table	Identifies the source table where the column's data comes from. This source table is normally a load table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the **Source Strategy** field. This field is used when generating a procedure to update the EDW 3NF table. It is also used in the track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a load table column, which in turn may have been a transformation or the combination of multiple columns. For previous value managed columns the source column is the column in the table whose previous value is being recorded.
Source Data Type	Identifies the source column's data type. [Read-only].
Transformation	Refer to **EDW 3NF Table Column Transformations**. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source Table** and **Source Column** fields will provide the other EDW 3NF table's side of the join. The options for this field are: False, True, Manual and Pre Join. • Setting this field to Manual changes the way the dimension table is looked up during the update procedure build. It allows you to join the dimension manually in the **Source Join** wizard (used to build the 'Where' clause). • Setting this field to Pre Join activates the **Pre Join Source Table** field and allows you to select a table from the drop-down list.
Pre Join Source Table	Indicates the table from which the pre joined column was sourced. When the Join option is set to False, this field becomes inactive. When the Join option is set to True or Manual, this field is set to the current table name. When the Join option is set to Pre Join, then you can select the required table from the drop-down list.

EDW 3NF Table Column Transformations

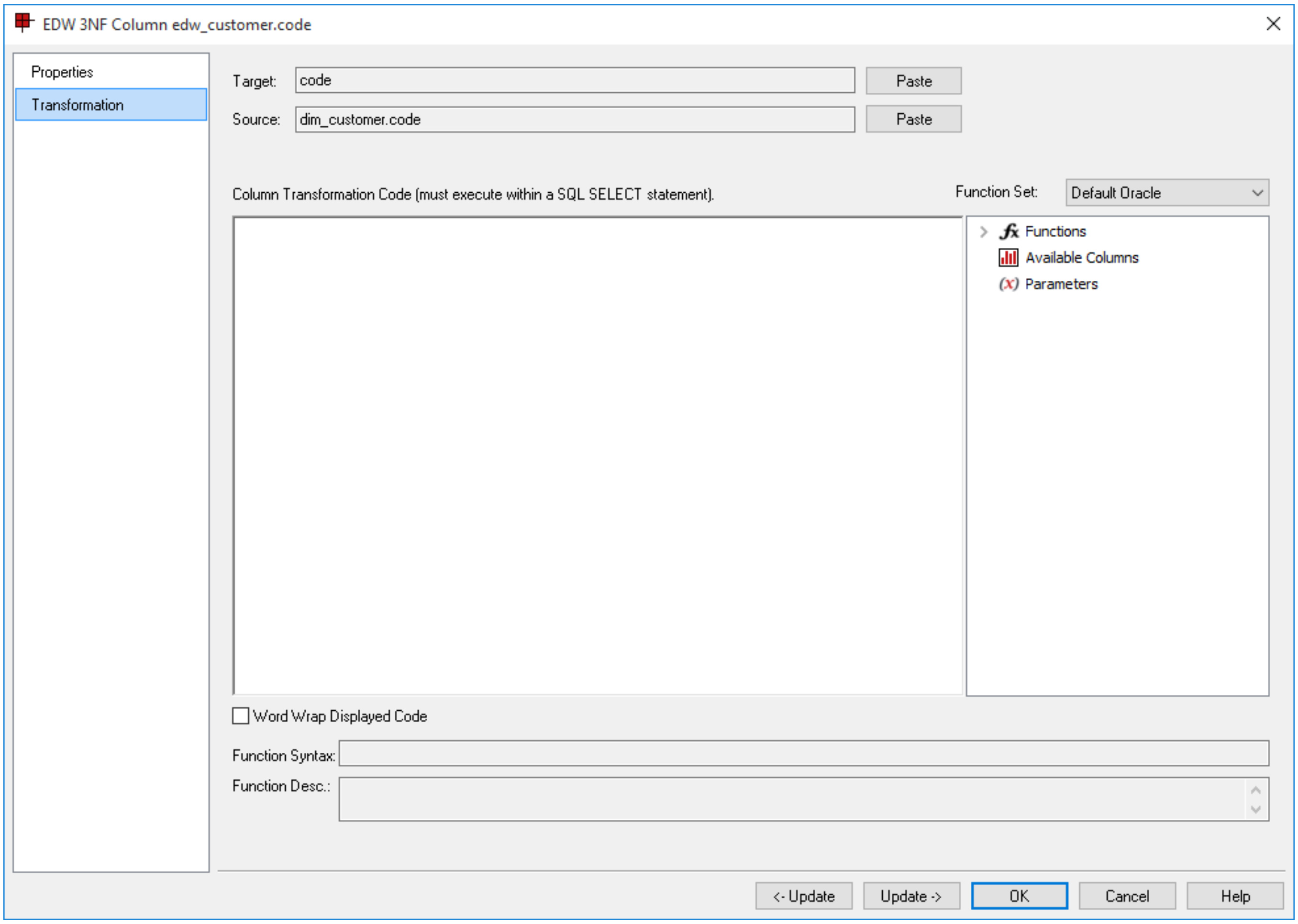
Each EDW 3NF table column can have a transformation associated with it. The transformation will be included in the generated procedure and will be executed as part of the procedure update.

The transformation must therefore be a valid SQL construct that can be included in a **Select** statement.

For example we could have a transformation of 'load_order_line.quantity * 0.125' to calculate a tax column of 12.5%.

Click the **Transformation** tab to enter a transformation.

The transformation screen is as follows:



EDW 3NF Column edw_customer.code

Target: code

Source: dim_customer.code

Column Transformation Code (must execute within a SQL SELECT statement).

Function Set: Default Oracle

☐ Word Wrap Displayed Code

Function Syntax:

Function Desc.:

Note:

Transformations are only put into effect when the procedure is re-generated.

Tip:

Microsoft Analysis Services 2005+ Tabular Mode Tables: For Tabular Mode table column transformations, **Default DAX** is the only applicable Function Set for **after load** transformations.

Refer to **Transformations** for details.

Data Vaults

The Data Vault system is an alternative approach to modeling an enterprise data warehouse that has been gaining popularity among organizations.

The Data Vault data warehouse architecture was invented by Dan Linstedt to provide an alternative to the traditional data warehouse modeling approach that includes developing 3rd Normal Form (3NF) type models or dimensional star schema models. The data vault methodology seeks to improve the efficiency of data ingestion and the flexibility of structure changes. For more information about Data Vaults, please refer to Dan Linstedt's website (**http://danlinstedt.com/solutions-2/data-vault-basics/**).

WhereScape RED has been enhanced to expand its current Data Vault functionality and provide improved automation for creating and managing Data Vault objects in WhereScape RED managed Data Warehouses. This includes the following:

- DSS columns for Load tables
- Wizard for Hash key generation
- Wizard for building Hub, Link and Satellite tables
- Templates for Procedure generation

All these enhancements are designed to be compliant with Data Vault 2.0 standards and are described in the succeeding sections below.

Note that previous releases of WhereScape RED used a workflow for Data Vault objects that is similar to the workflow for creating Data Store Objects. The Hub, Link and Satellite tables are based on standard Load or Stage tables (that do not include the hash key column type flags), WhereScape RED reverts to this behavior and the resulting procedures are generated by internal WhereScape RED automation and not via templates.

If the above legacy method of creating Data Vault Objects is required, please refer to the Data Store Objects chapter of this User Guide. You can also refer to the older versions of this User Guide, for details on the previous processes used to import Data Vault objects into WhereScape RED.

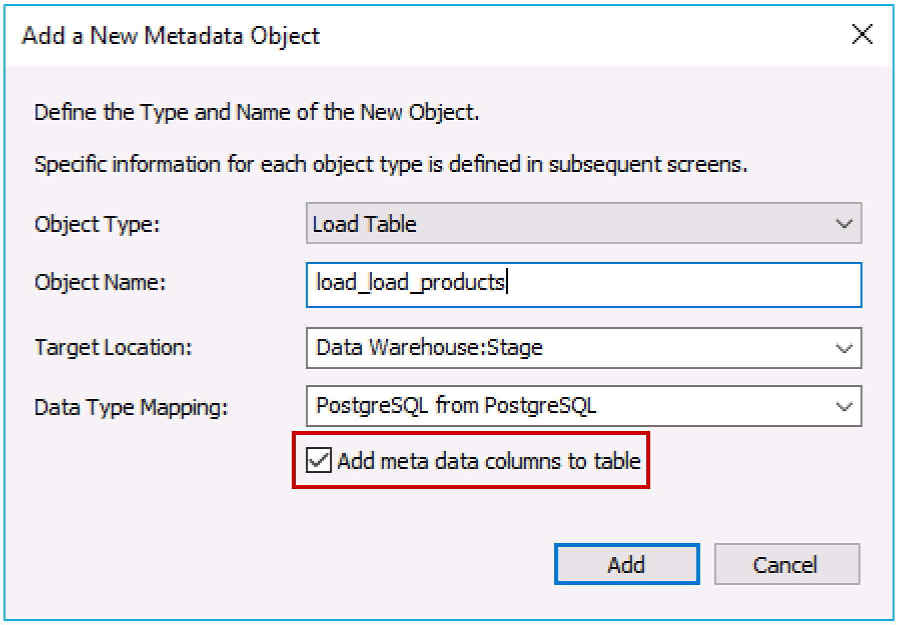
For more information on Data Vault design, refer to *Building a Scalable Data Warehouse With Data Vault 2.0* by *Daniel Linstedt and Michael Olschimke*.

Data Vault Functions and Features

The following describe the WhereScape RED functions and features that are used for building Data Vault objects (Hub, Link and Satellite) to create a Data Vault model.

Load Table Meta Data Columns

The option to add default meta data columns to a new Load table object in WhereScape RED.



Add a New Metadata Object

Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: Load Table

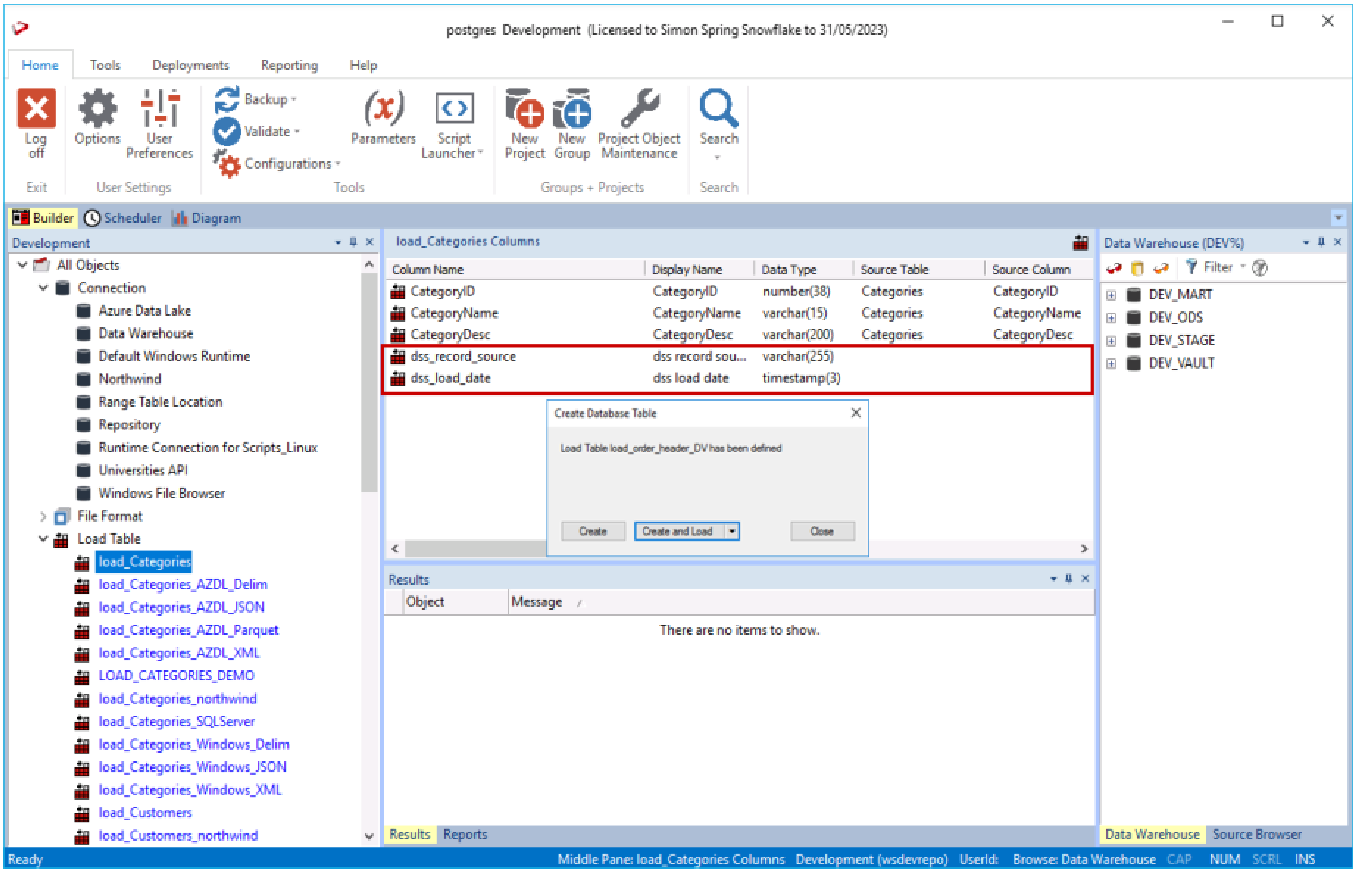
Object Name: load_load_products

Target Location: Data Warehouse:Stage

Data Type Mapping: PostgreSQL from PostgreSQL

☑ Add meta data columns to table

If the option **Add meta data columns to table** is selected, two DSS columns (**dss_record_source** and **dss_load_date**) are included in the meta data for the table and are populated by transformations. Note that these two DSS columns could equally be applied to other Load tables not used in a Data Vault model but are particularly important to comply with the Data Vault standards.



postgres Development (Licensed to Simon Spring Snowflake to 31/05/2023)

load_Categories Columns

Column Name	Display Name	Data Type	Source Table	Source Column
CategoryID	CategoryID	number(38)	Categories	CategoryID
CategoryName	CategoryName	varchar(15)	Categories	CategoryName
CategoryDesc	CategoryDesc	varchar(200)	Categories	CategoryDesc
dss_record_source	dss record sou...	varchar(255)		
dss_load_date	dss load date	timestamp(3)		

Create Database Table

Load Table load_order_header_DV has been defined

Results

There are no items to show.

Ready Middle Pane: load_Categories Columns Development (wsdevrepo) UserId: Browse: Data Warehouse CAP NUM SCRL INS

Note

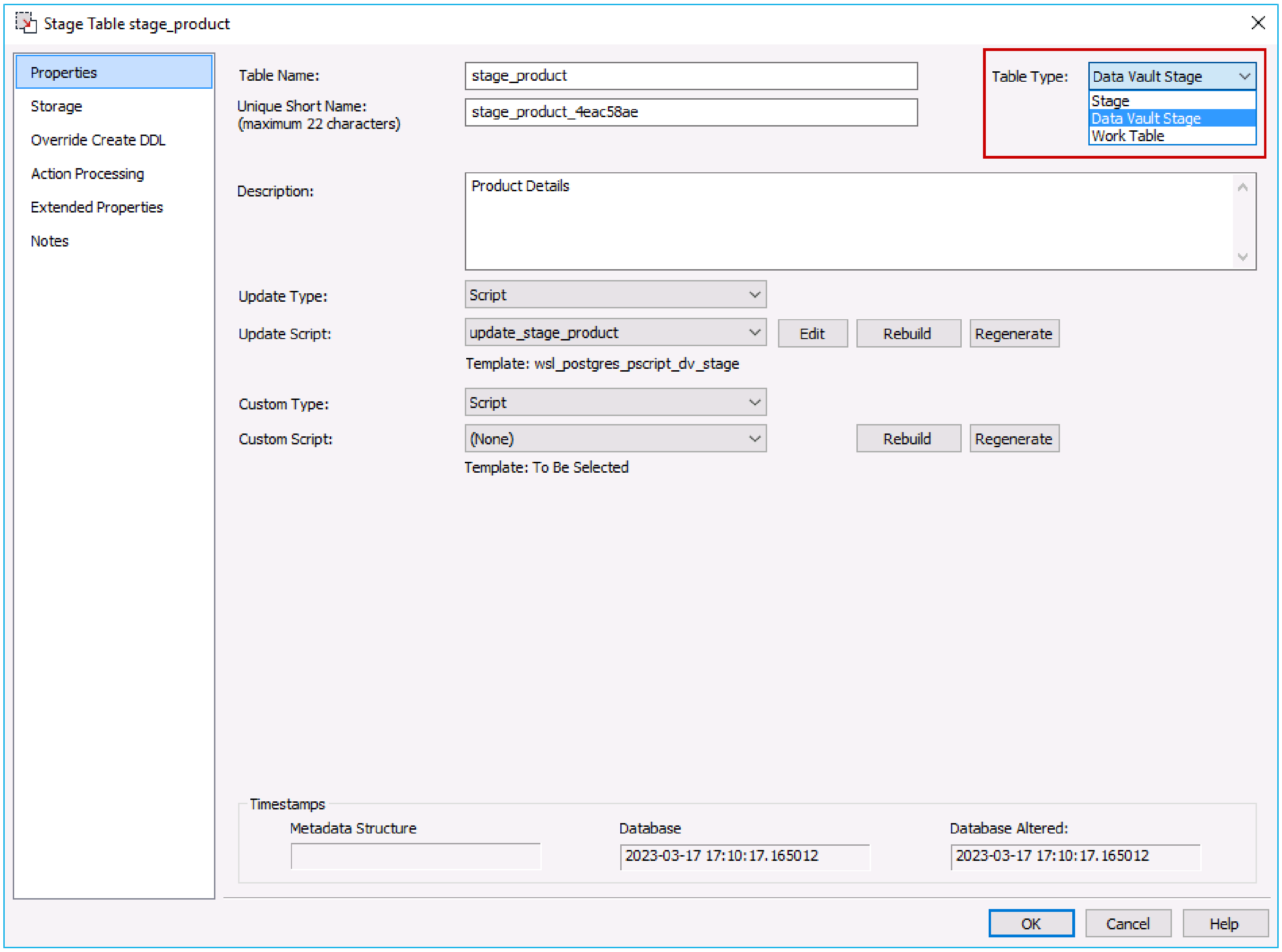
Custom aliases for the **dss_load_date** and **dss_record_source** columns can be defined in the **Home> Options > DSS Tables and Columns > DSS Columns to Include and Naming** window. If custom DSS aliases are defined then the aliases are used for the column names that are added to the new load tables. Refer to DSS

Note

Columns for details.

Data Vault Stage Table

WhereScape RED provides the option to create **Data Vault Stage** objects that are used to define the source columns for the Hub, Link and Change hash key columns.



Stage Table stage_product

Table Name: stage_product

Unique Short Name: (maximum 22 characters) stage_product_4eac58ae

Table Type: Data Vault Stage

Description: Product Details

Update Type: Script

Update Script: update_stage_product

Template: wsl_postgres_pscript_dv_stage

Custom Type: Script

Custom Script: (None)

Template: To Be Selected

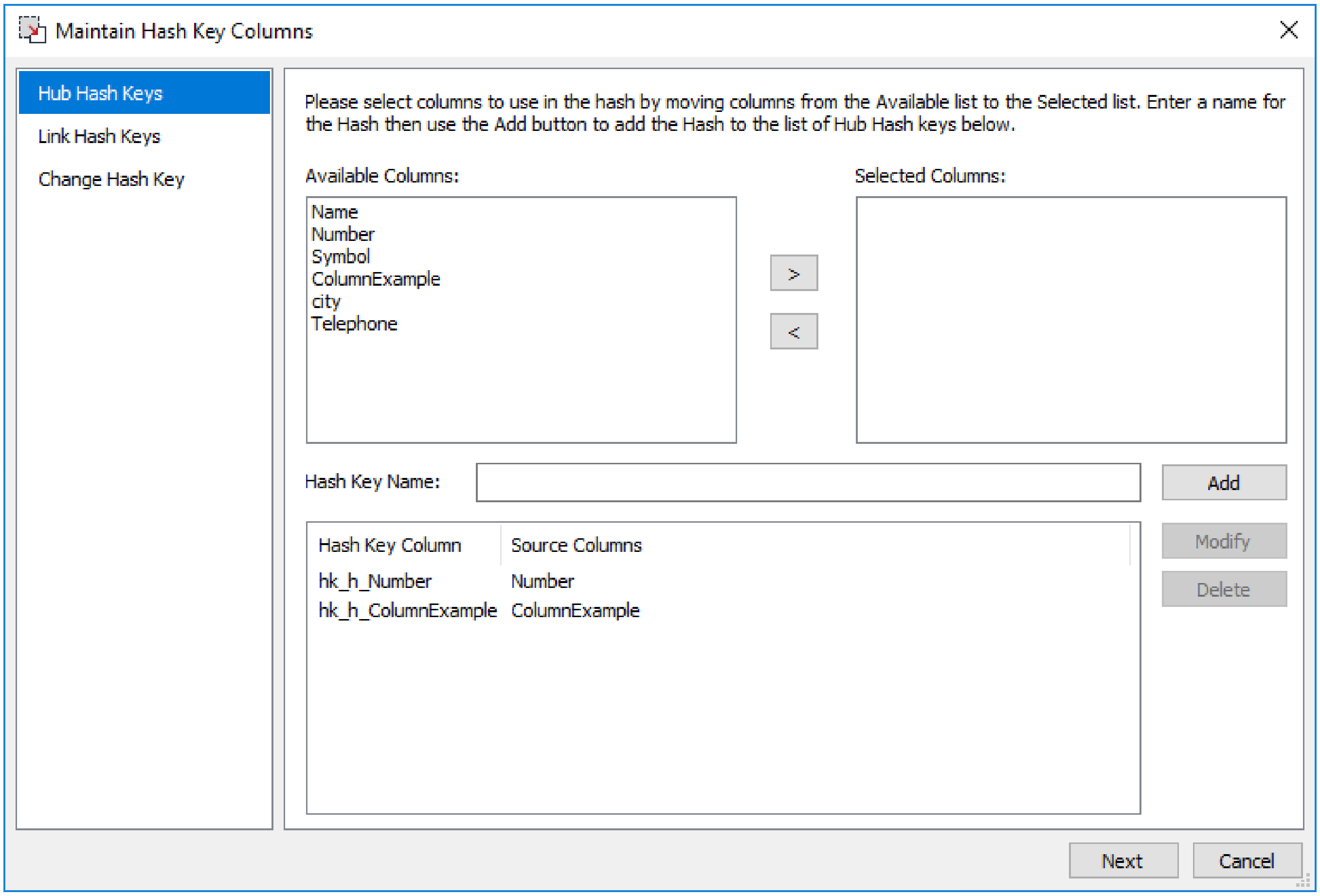
Timestamps

Metadata Structure	Database	Database Altered:
	2023-03-17 17:10:17.165012	2023-03-17 17:10:17.165012

The **Data Vault Stage** option can be selected from **Table Type** drop-down of the **Stage Table Properties** screen. This object type is created via a Wizard, described in **Creating Data Vault Stage Tables**.

Hash Key Generation Wizard

This Wizard is launched when building a **Stage** table with **Table Type** of **Data Vault Stage**. It enables you to specify the source columns to be used in defining the **Hub**, **Link** and **Change** hash key columns.



Please select columns to use in the hash by moving columns from the Available list to the Selected list. Enter a name for the Hash then use the Add button to add the Hash to the list of Hub Hash keys below.

Available Columns:

Name
Number
Symbol
ColumnExample
city
Telephone

Selected Columns:

Hash Key Name:

Hash Key Column	Source Columns
hk_h_Number	Number
hk_h_ColumnExample	ColumnExample

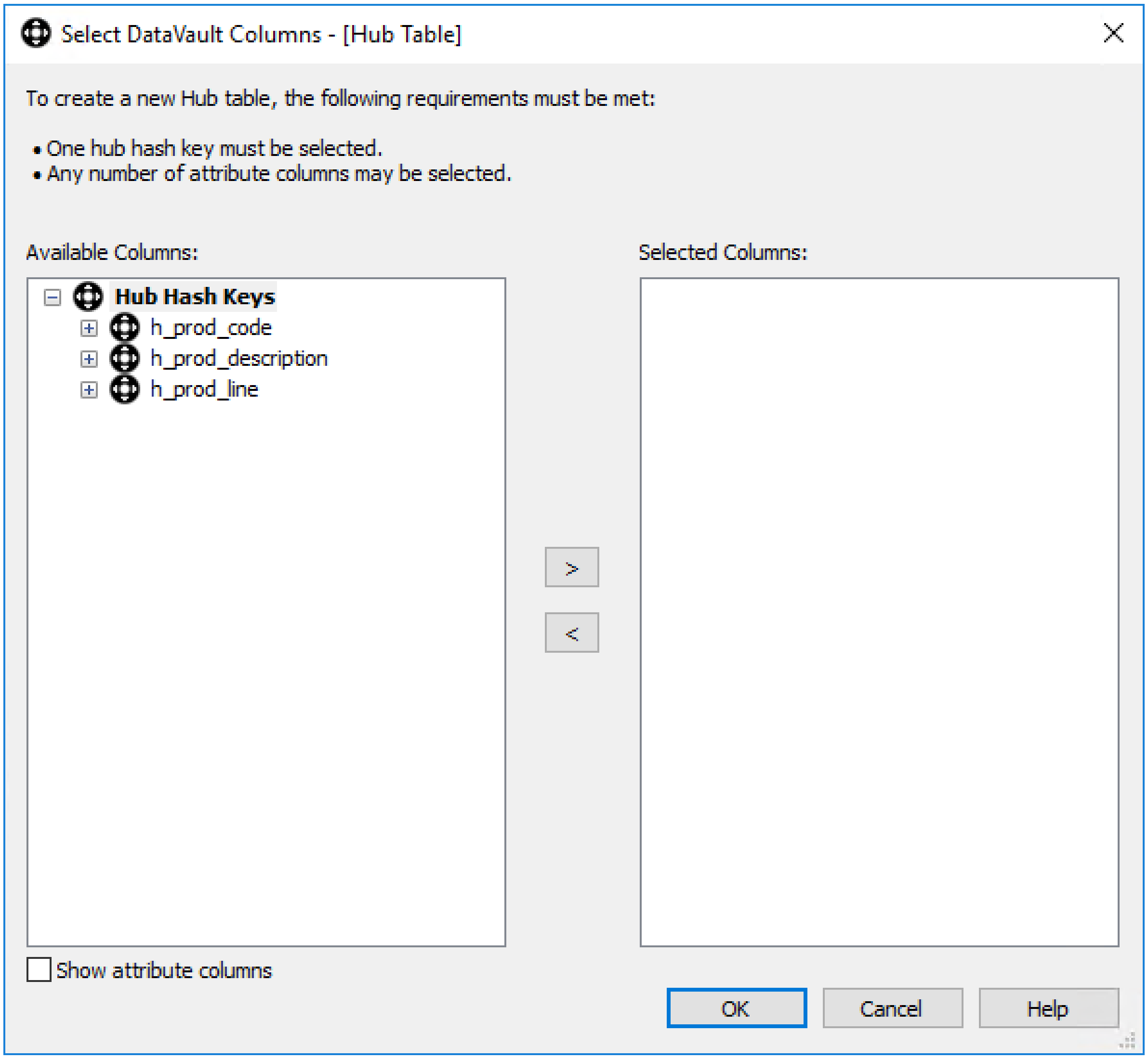
The generated Hash Keys are used to build the **Hub**, **Link** and **Satellite** objects in WhereScape RED. The detailed steps for using this Wizard is described in **Creating Data Vault Stage Tables**.

Notes

1. Custom aliases for the **dss_change_hash** column can be defined in the **Tools > Options > DSS Tables and Columns > DSS Columns to Include and Naming** window. If custom DSS aliases are defined then the aliases are used for the column names that are added to the new Load tables. Refer to **DSS Columns** for details.
2. The data type defined for the **dss_change_hash** column in **Tools > Options > Data Vault** window is applied when creating a satellite change hash key. Refer to **Data Vault Hash Keys setting** for details.
3. Refer to **Changing the Data Vault Hash Key Function in WhereScape RED 6.9.1.0** and above, if you need to amend the Hash key function on the shipped WhereScape RED **Data Vault Templates**.

Hub, Link and Satellite Creation Wizard

This Wizard is launched when building a Data Vault object (**Hub**, **Link** or **Satellite** table) by dragging and dropping the source **Data Vault Stage** table from the right pane to the middle pane.



Select DataVault Columns - [Hub Table]

To create a new Hub table, the following requirements must be met:

- One hub hash key must be selected.
- Any number of attribute columns may be selected.

Available Columns:

- Hub Hash Keys
  - h_prod_code
  - h_prod_description
  - h_prod_line

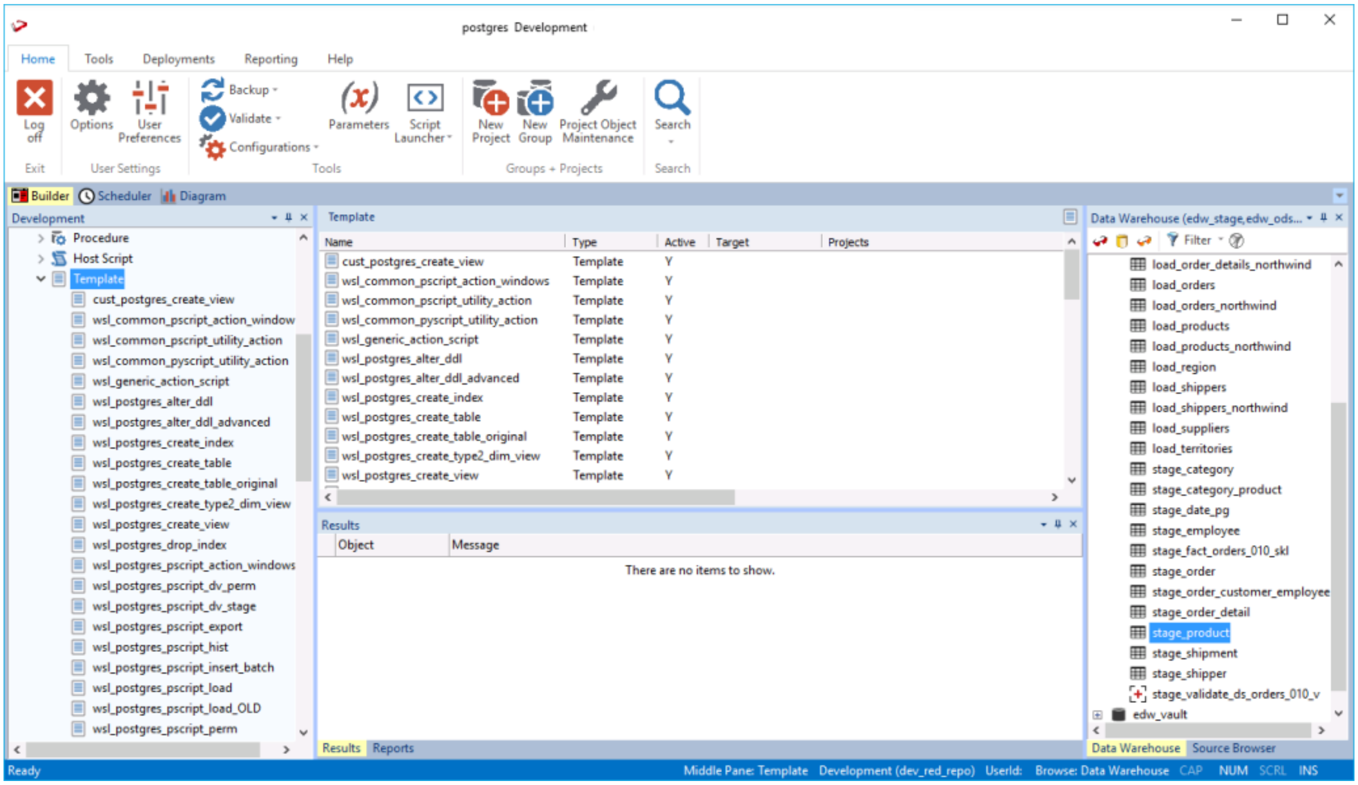
Selected Columns:

□ Show attribute columns

The detailed steps for using this Wizard is described **Creating the Hub, Link and Satellite Tables**.

Data Vault Templates

Templates are used to generate update procedures for **Data Vault** objects. Users must select a template to use when generating the update procedure for **Data Vault** objects created in WhereScape RED.



Development
- Procedure
- Host Script
- Template
  - cust_postgres_create_view
  - wsl_common_pscript_action_window
  - wsl_common_pscript_utility_action
  - wsl_common_pyscript_utility_action
  - wsl_generic_action_script
  - wsl_postgres_alter_ddl
  - wsl_postgres_alter_ddl_advanced
  - wsl_postgres_create_index
  - wsl_postgres_create_table
  - wsl_postgres_create_table_original
  - wsl_postgres_create_type2_dim_view
  - wsl_postgres_create_view
  - wsl_postgres_drop_index
  - wsl_postgres_pscript_action_windows
  - wsl_postgres_pscript_dv_perm
  - wsl_postgres_pscript_dv_stage
  - wsl_postgres_pscript_export
  - wsl_postgres_pscript_hist
  - wsl_postgres_pscript_insert_batch
  - wsl_postgres_pscript_load
  - wsl_postgres_pscript_load_OLD
  - wsl_postgres_pscript_perm

Template

Name	Type	Active	Target	Projects
cust_postgres_create_view	Template	Y		
wsl_common_pscript_action_windows	Template	Y		
wsl_common_pscript_utility_action	Template	Y		
wsl_common_pyscript_utility_action	Template	Y		
wsl_generic_action_script	Template	Y		
wsl_postgres_alter_ddl	Template	Y		
wsl_postgres_alter_ddl_advanced	Template	Y		
wsl_postgres_create_index	Template	Y		
wsl_postgres_create_table	Template	Y		
wsl_postgres_create_table_original	Template	Y		
wsl_postgres_create_type2_dim_view	Template	Y		
wsl_postgres_create_view	Template	Y		

Results

Object	Message

There are no items to show.

Data Warehouse (edw_stage,edw_ods...
- load_order_details_northwind
- load_orders
- load_orders_northwind
- load_products
- load_products_northwind
- load_region
- load_shippers
- load_shippers_northwind
- load_suppliers
- load_territories
- stage_category
- stage_category_product
- stage_date_pg
- stage_employee
- stage_fact_orders_010_skl
- stage_order
- stage_order_customer_employee
- stage_order_detail
- stage_product
- stage_shipment
- stage_shipper
- stage_validate_ds_orders_010_v
- edw_vault

Ready Middle Pane: Template Development (dev_red_repo) UserId: Browse: Data Warehouse CAP NUM SCRL INS

The **SQL Server Data Vault** templates are described below:

- **wsl_sqlserver_proc_dv_stage** – this template creates an SQL Server procedure for updating WhereScape RED **Data Vault Stage** tables.
- **wsl_sqlserver_proc_dv_perm** – this template creates an SQL Server procedure for updating WhereScape RED **Data Vault** objects (**Hub**, **Link** and **Satellite** tables).
- **wsl_sqlserver_utility_dv** – this utility template contains generic SQL Server macros that are used by the other two templates above.

The above templates are available in WhereScape RED version 6.9.1.0 and above. If the templates are not visible in the **Template** objects list after installing/upgrading WhereScape RED, use the WhereScape RED Setup Administrator to Validate the MetaData Repository.

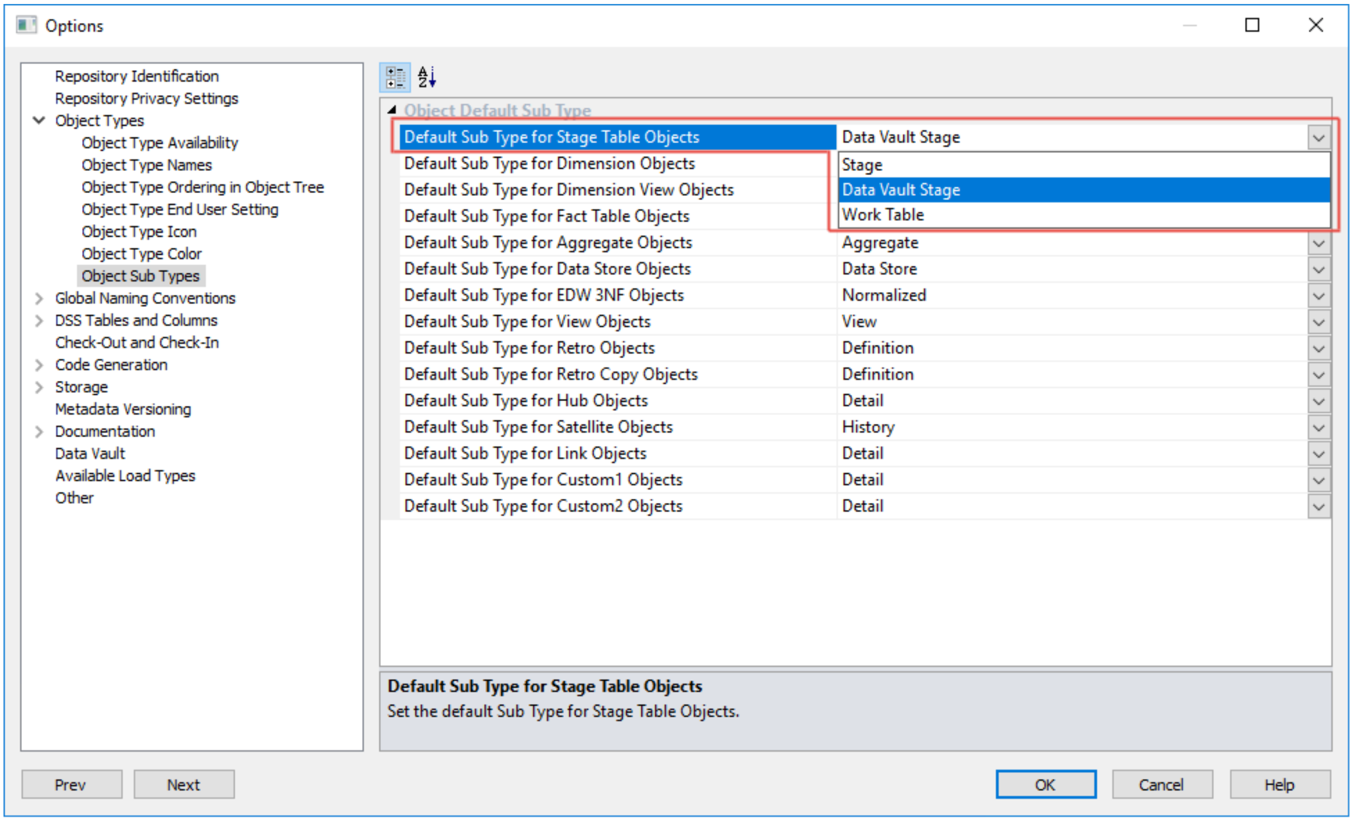
A Wizard to generate update procedures via templates is used to populate **Data Vault** tables. The detailed steps for using this Wizard is described in Generating Update Procedures for Hub, Link and Satellite Tables.

Data Vault Settings

Settings for Data Vault objects can be configured from the **Tools > Options** screen.

Object Types Settings

The **Default Sub Type for Stage Table Objects** drop-down includes the **Data Vault Stage** option.



Object Default Sub Type

Default Sub Type for Stage Table Objects	Data Vault Stage
Default Sub Type for Dimension Objects	Stage Data Vault Stage Work Table
Default Sub Type for Dimension View Objects	
Default Sub Type for Fact Table Objects	
Default Sub Type for Aggregate Objects	Aggregate
Default Sub Type for Data Store Objects	Data Store
Default Sub Type for EDW 3NF Objects	Normalized
Default Sub Type for View Objects	View
Default Sub Type for Retro Objects	Definition
Default Sub Type for Retro Copy Objects	Definition
Default Sub Type for Hub Objects	Detail
Default Sub Type for Satellite Objects	History
Default Sub Type for Link Objects	Detail
Default Sub Type for Custom1 Objects	Detail
Default Sub Type for Custom2 Objects	Detail

Default Sub Type for Stage Table Objects

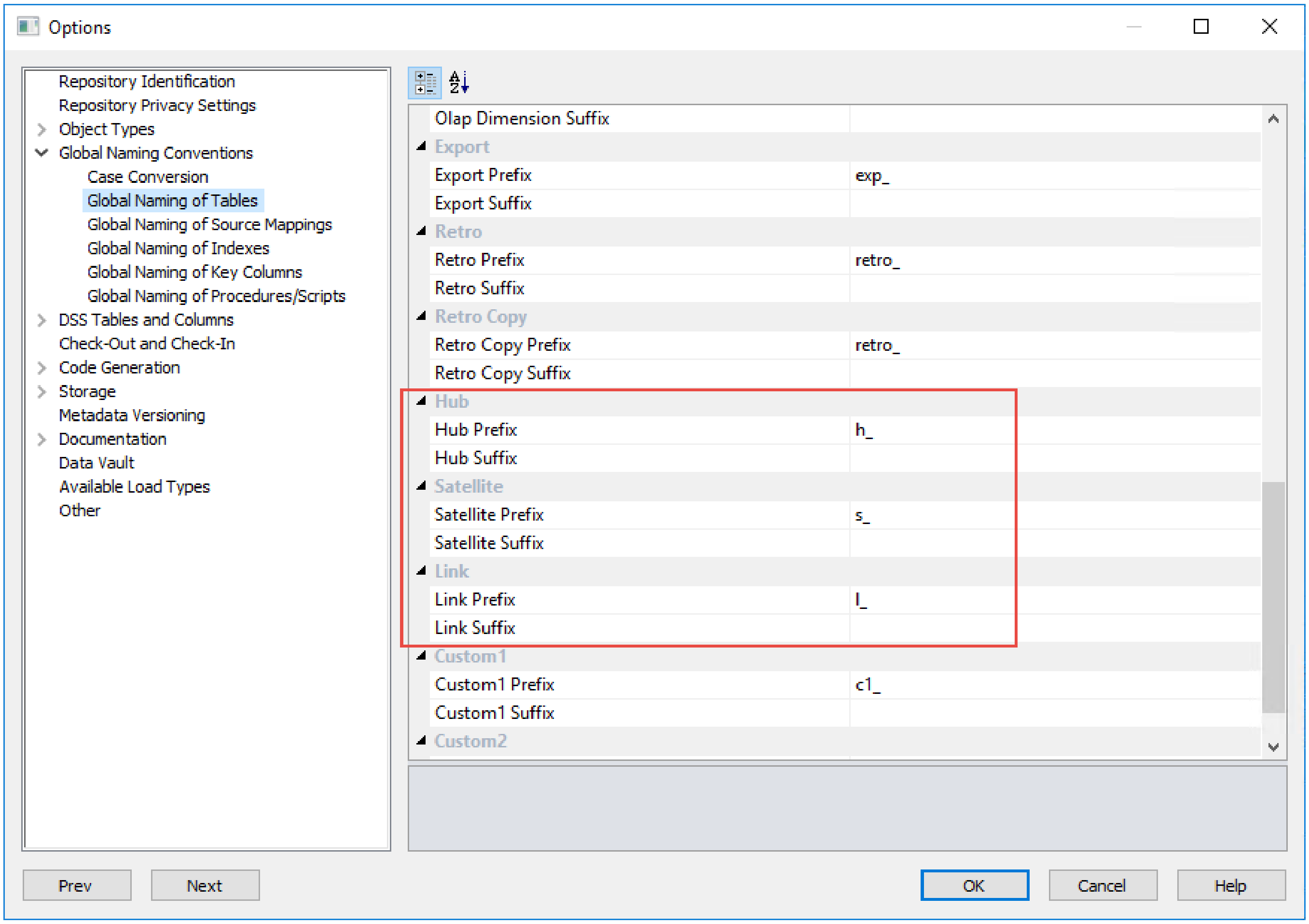
Set the default Sub Type for Stage Table Objects.

Configure this setting, if you want to set **Data Vault Stage** to be the default **Table Type** in the **Stage Table Properties** screen.

Global Naming Conventions Settings

Global Naming of Tables

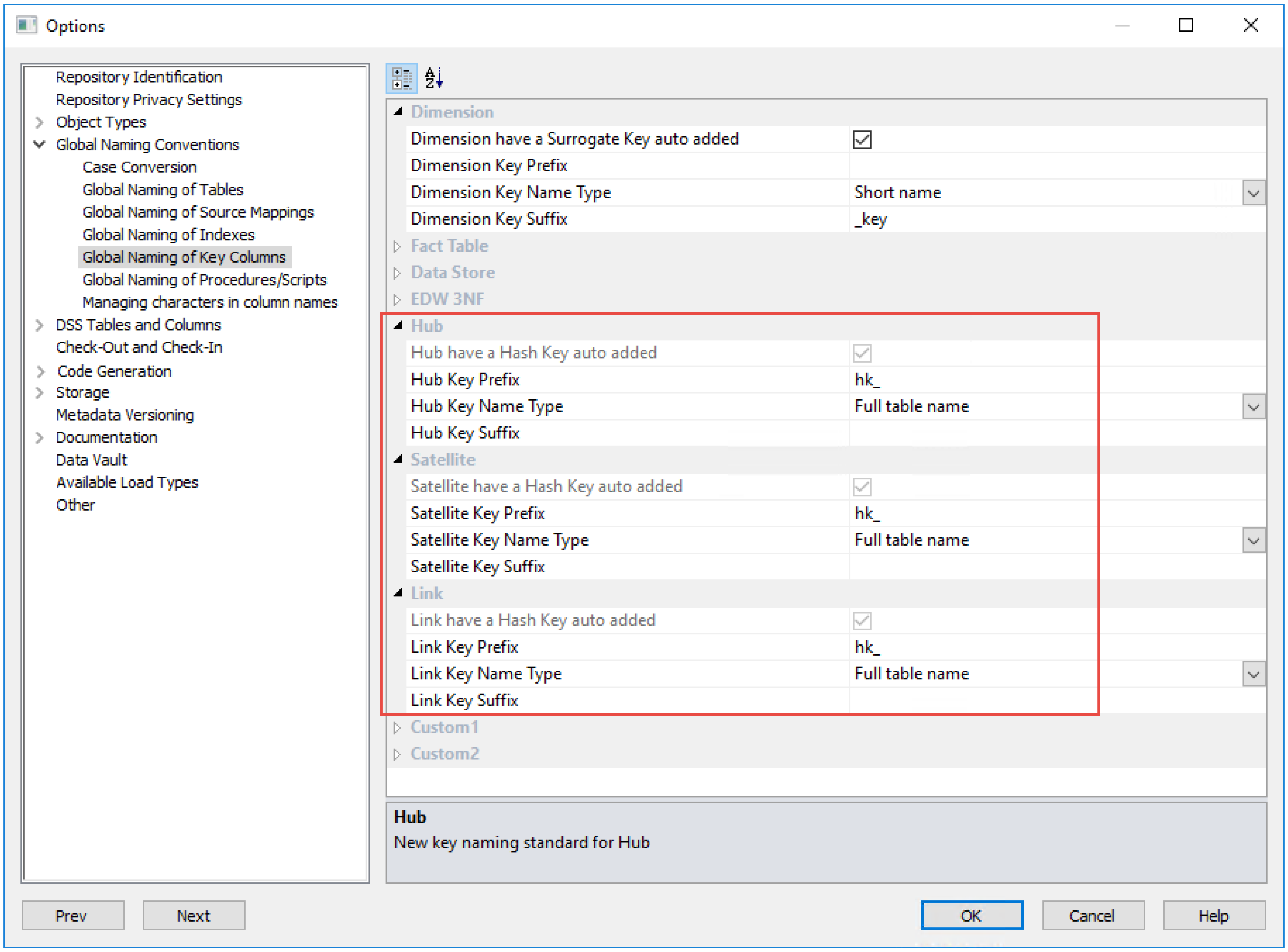
The **Global Naming of Tables** setting for **Hub**, **Satellite** and **Link** tables have been set to comply with the recommended standard naming convention for **Data Vault** tables. You can edit this setting to suit your requirements.



Olap Dimension Suffix	
Export	
Export Prefix	exp_
Export Suffix	
Retro	
Retro Prefix	retro_
Retro Suffix	
Retro Copy	
Retro Copy Prefix	retro_
Retro Copy Suffix	
Hub	
Hub Prefix	h_
Hub Suffix	
Satellite	
Satellite Prefix	s_
Satellite Suffix	
Link	
Link Prefix	l_
Link Suffix	
Custom1	
Custom1 Prefix	c1_
Custom1 Suffix	
Custom2	

Global Naming of Key Columns

The **Global Naming of Key Columns** setting for **Hub**, **Satellite** and **Link** tables enables you to define the default naming convention for the Hash Keys that are generated for these Data Vault objects (e.g. Key Prefix value plus the Source Column name). You can edit this setting to suit your requirements.



Dimension	
Dimension have a Surrogate Key auto added	☑
Dimension Key Prefix	
Dimension Key Name Type	Short name
Dimension Key Suffix	_key
Fact Table	
Data Store	
EDW 3NF	
Hub	
Hub have a Hash Key auto added	☑
Hub Key Prefix	hk_
Hub Key Name Type	Full table name
Hub Key Suffix	
Satellite	
Satellite have a Hash Key auto added	☑
Satellite Key Prefix	hk_
Satellite Key Name Type	Full table name
Satellite Key Suffix	
Link	
Link have a Hash Key auto added	☑
Link Key Prefix	hk_
Link Key Name Type	Full table name
Link Key Suffix	
Custom1	
Custom2	

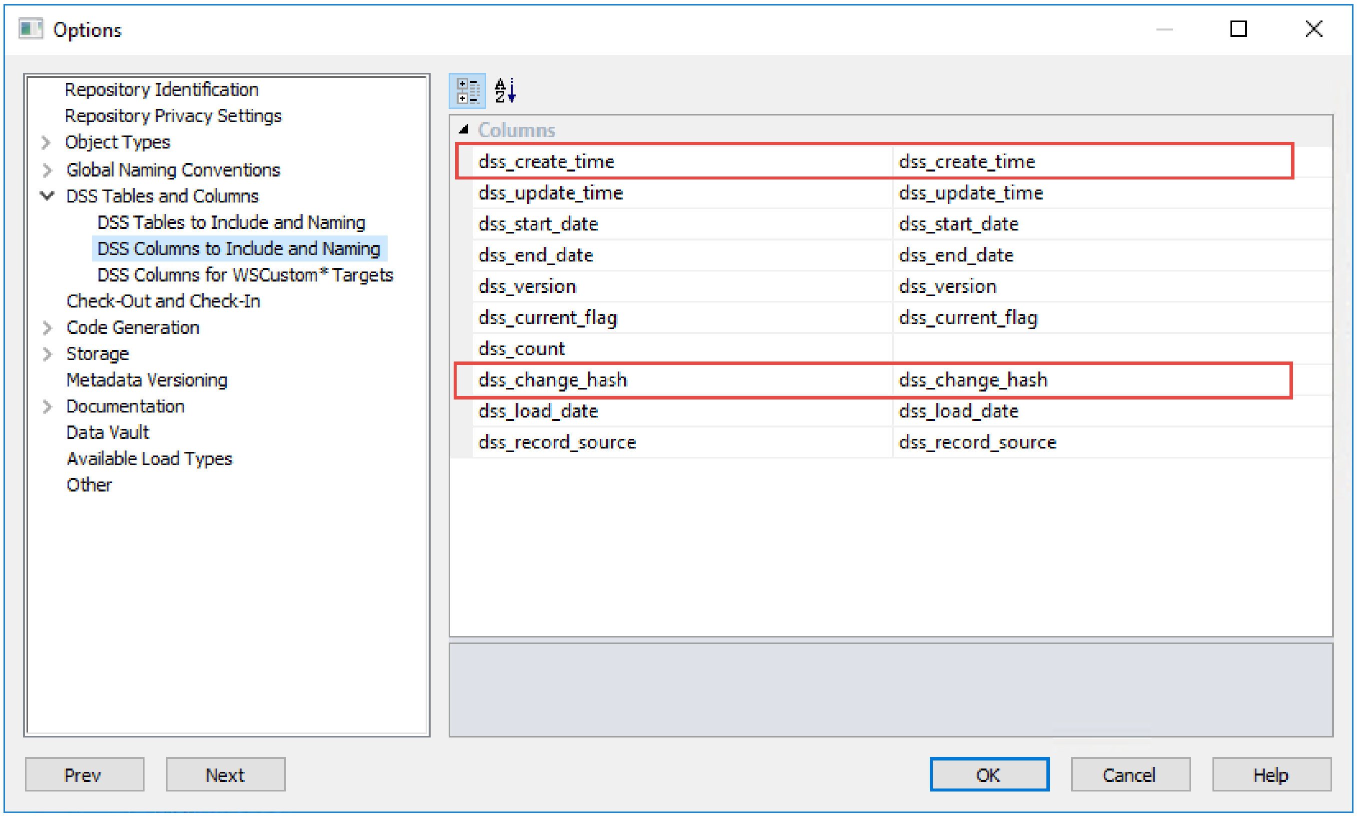
Hub
New key naming standard for Hub

Note

WhereScape RED was enhanced to comply with Data Vault 2.0 standards. The Hash keys are now generated as part of the **Data Vault Stage table creation wizard** which are then passed during the drag and drop creation of new Data Vault objects. Hence hash key columns are now mandatory and must not be added during the creation of a Data Vault object.

DSS Tables and Columns Settings

The **DSS Columns to Include and Naming** setting includes two additional columns which is described below:


Options
Repository Identification
Repository Privacy Settings
Object Types
Global Naming Conventions
DSS Tables and Columns
DSS Tables to Include and Naming
DSS Columns to Include and Naming
DSS Columns for WSCustom* Targets
Check-Out and Check-In
Code Generation
Storage
Metadata Versioning
Documentation
Data Vault
Available Load Types
Other
Prev
Next
OK
Cancel
Help

Columns

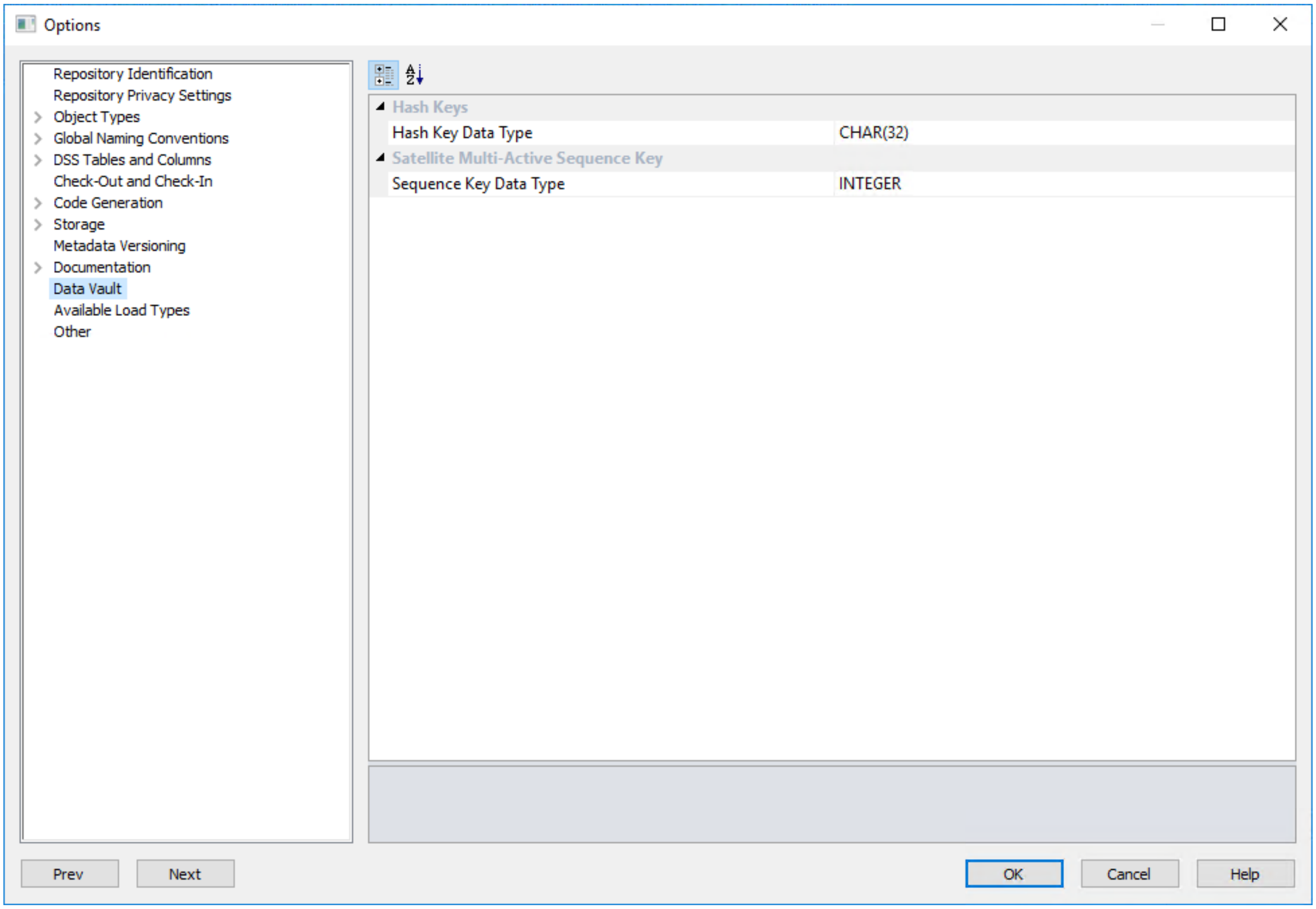
dss_create_time	dss_create_time
dss_update_time	dss_update_time
dss_start_date	dss_start_date
dss_end_date	dss_end_date
dss_version	dss_version
dss_current_flag	dss_current_flag
dss_count	
dss_change_hash	dss_change_hash
dss_load_date	dss_load_date
dss_record_source	dss_record_source

- **dss_create_time** – this column is added to all Stage, ODS, Normalized, Dimension, Fact and Aggregate tables for information purposes only. Leave blank to deactivate.
- **dss_change_hash** – this column is used to identify the differences in the descriptive columns of a Satellite table which is required for generating the change hash key for a Satellite table.

Data Vault

The **Data Vault** setting is used to define the data type for the generated Hash Keys in RED that are used to build the Hub, Link, and Satellite table objects.

Also, the Satellite Multi-Active Sequence Key option allows you to define how this key is represented in the Data Warehouse. The Sequence Key Data Type is INTEGER, by default.



Hash Keys

Hash Key Data Type	CHAR(32)

Satellite Multi-Active Sequence Key

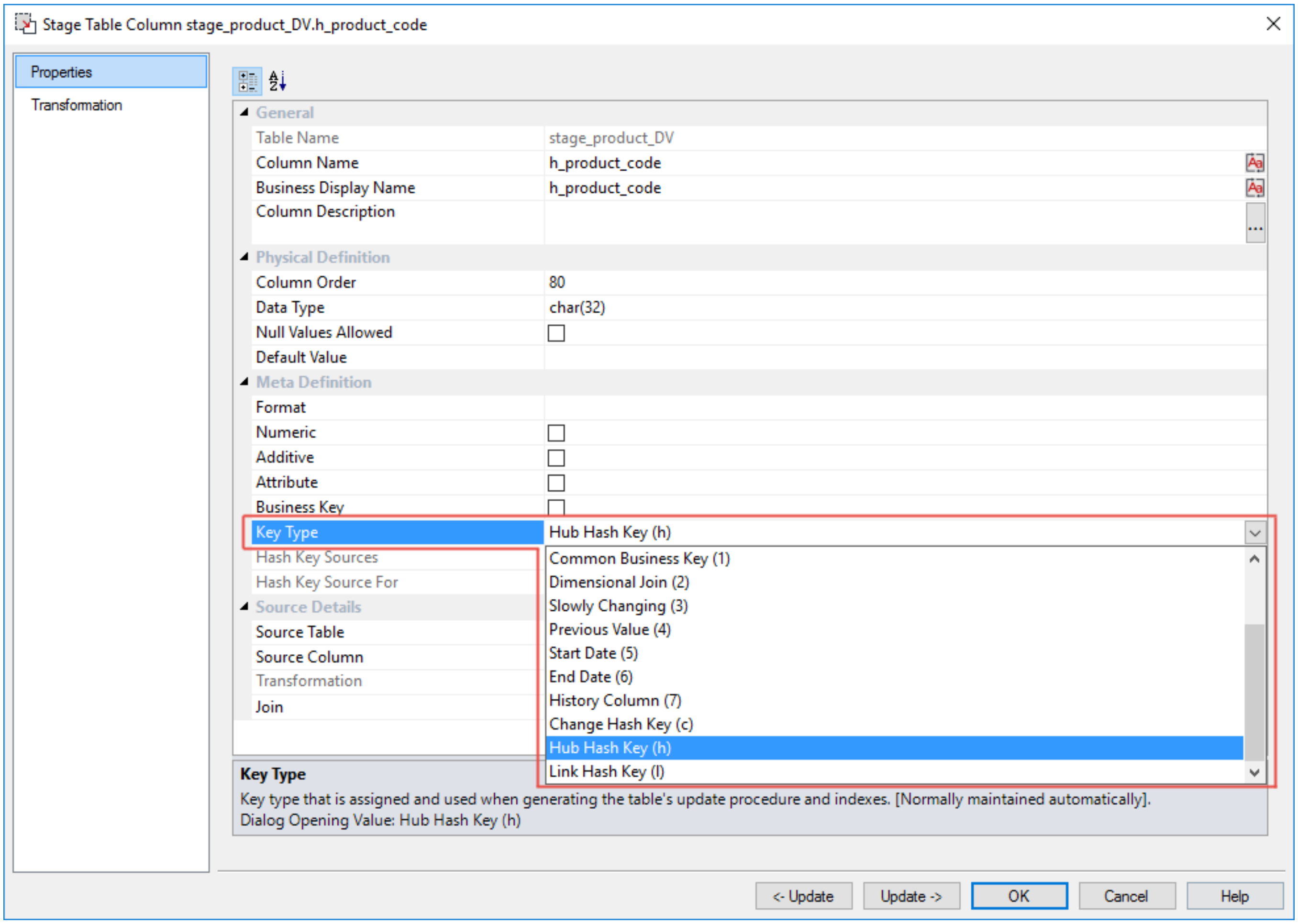
Sequence Key Data Type	INTEGER

Note

Changing this data type has no impact on any existing hash key columns.

Table Column Properties

The **Key Type** field drop-down include options for **Data Vault** hash keys, e.g. **Change Hash Key (c)**, **Hub Hash Key (h)** and **Link Hash Key (l)**.



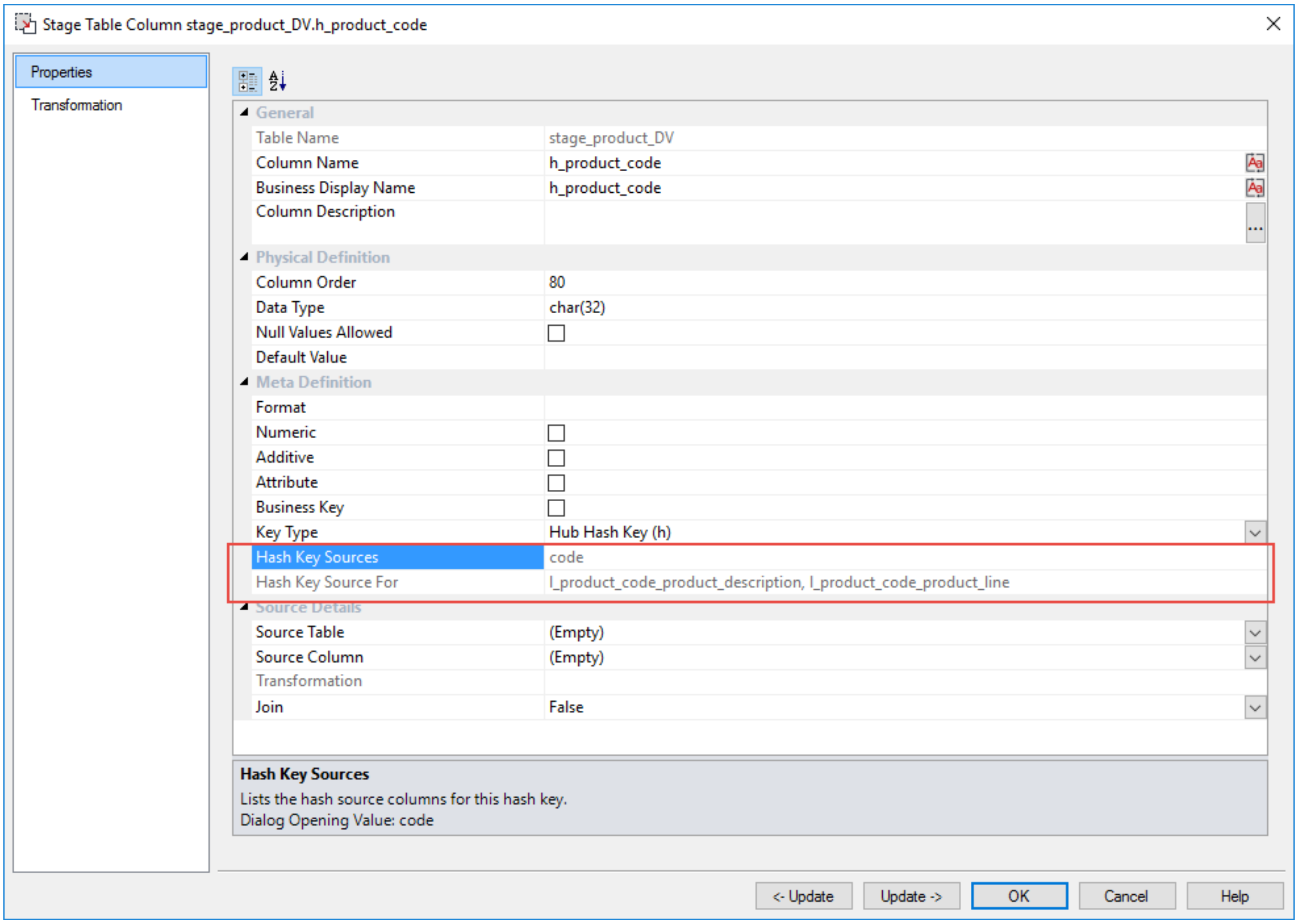
Stage Table Column stage_product_DV.h_product_code

General	
Table Name	stage_product_DV
Column Name	h_product_code
Business Display Name	h_product_code
Column Description	
Physical Definition	
Column Order	80
Data Type	char(32)
Null Values Allowed	□
Default Value	
Meta Definition	
Format	
Numeric	□
Additive	□
Attribute	□
Business Key	□
Key Type	Hub Hash Key (h)
Hash Key Sources	
Hash Key Source For	
Source Details	
Source Table	
Source Column	
Transformation	
Join	

Common Business Key (1)
Dimensional Join (2)
Slowly Changing (3)
Previous Value (4)
Start Date (5)
End Date (6)
History Column (7)
Change Hash Key (c)
Hub Hash Key (h)
Link Hash Key (l)

Key Type
Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically].
Dialog Opening Value: Hub Hash Key (h)

Hash Key source information are also displayed for these key types.



General	
Table Name	stage_product_DV
Column Name	h_product_code
Business Display Name	h_product_code
Column Description	
Physical Definition	
Column Order	80
Data Type	char(32)
Null Values Allowed	□
Default Value	
Meta Definition	
Format	
Numeric	□
Additive	□
Attribute	□
Business Key	□
Key Type	Hub Hash Key (h)
Hash Key Sources	code
Hash Key Source For	l_product_code_product_description, l_product_code_product_line
Source Details	
Source Table	(Empty)
Source Column	(Empty)
Transformation	
Join	False

Hash Key Sources
Lists the hash source columns for this hash key.
Dialog Opening Value: code

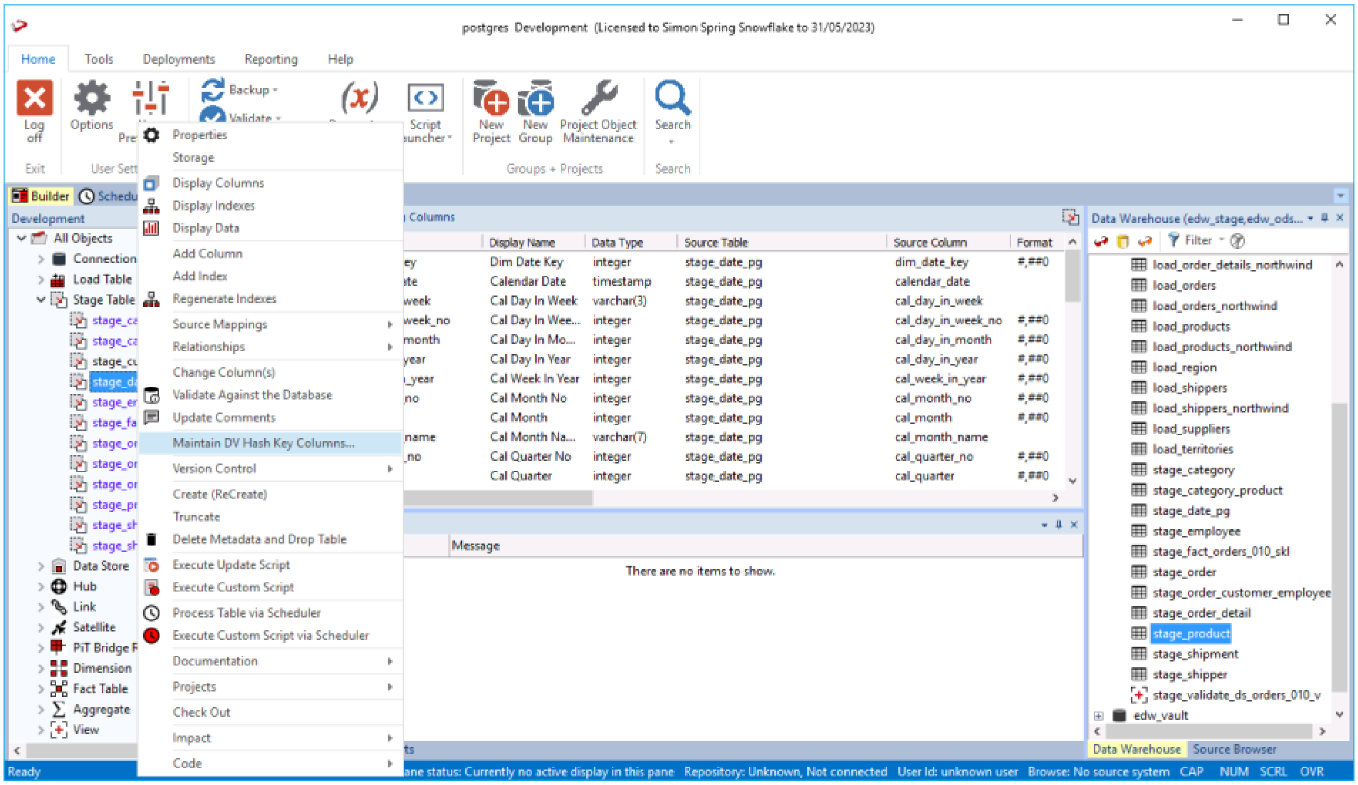
- **Hash Key Sources** – displays the source columns that are used to generate the selected **Hub**, **Link** or **Change** hash key.
- **Hash Key Source For** – displays the hash keys columns that use the displayed hash key sources.

The hash key generation Wizard enables you to specify the source columns to be used in defining the **Hub**, **Link** and **Change** hash key columns. The detailed steps for using this Wizard is described in Creating Data Vault Stage Tables.

Once the hash key columns have been defined, another Wizard is used to generate the procedure to populate the columns. The detailed steps for using this Wizard is described in Generating Update Procedures for the Data Vault Stage Table.

Maintain Hash Key Columns

The context menu for **Stage Table** objects, listed in the left pane provides an option for maintaining **Data Vault** hash key columns.



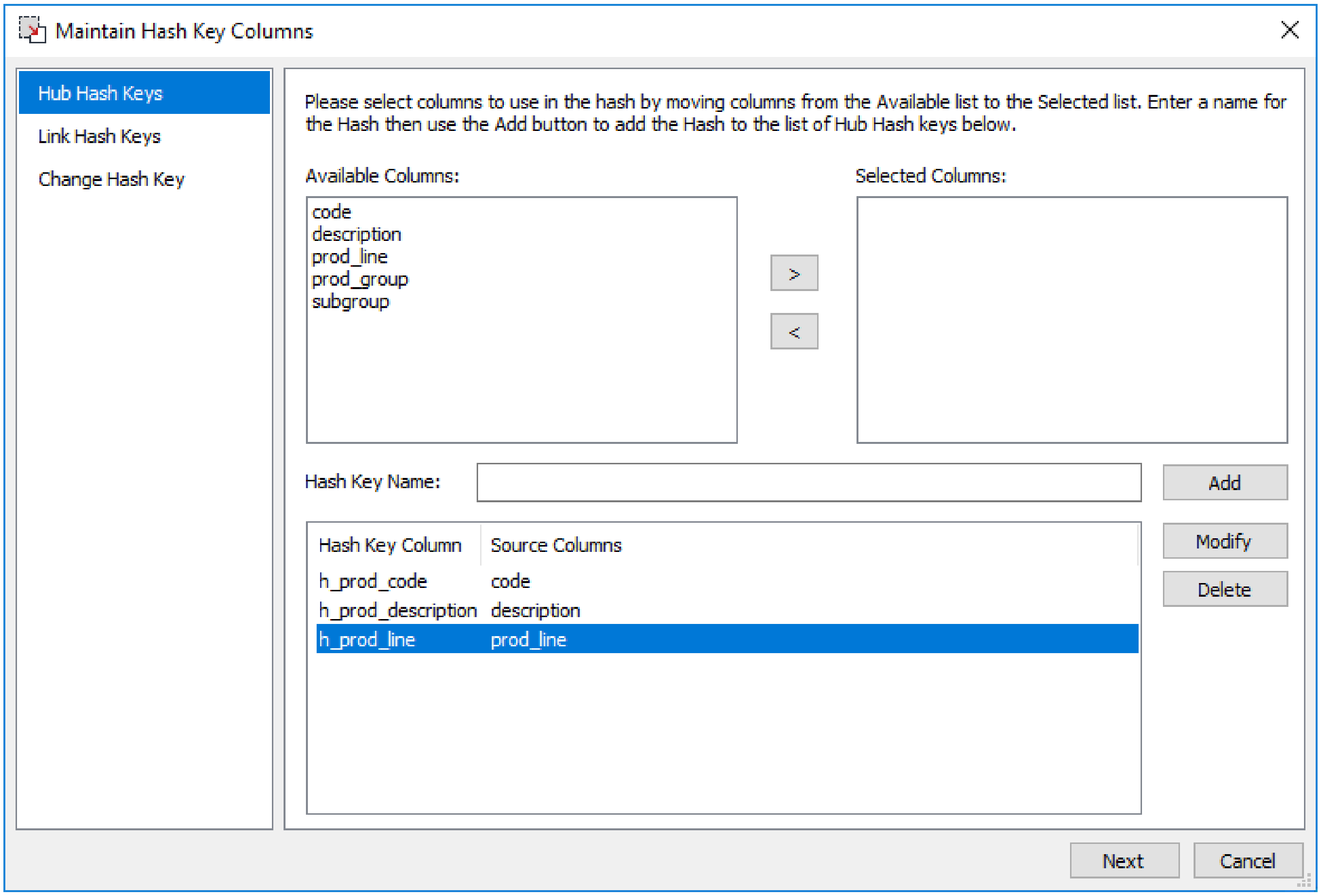
	Display Name	Data Type	Source Table	Source Column	Format
ey	Dim Date Key	integer	stage_date_pg	dim_date_key	#,##0
ite	Calendar Date	timestamp	stage_date_pg	calendar_date	
week	Cal Day In Week	varchar(3)	stage_date_pg	cal_day_in_week	
week_no	Cal Day In Wee...	integer	stage_date_pg	cal_day_in_week_no	#,##0
month	Cal Day In Mo...	integer	stage_date_pg	cal_day_in_month	#,##0
year	Cal Day In Year	integer	stage_date_pg	cal_day_in_year	#,##0
_year	Cal Week In Year	integer	stage_date_pg	cal_week_in_year	#,##0
no	Cal Month No	integer	stage_date_pg	cal_month_no	#,##0
	Cal Month	integer	stage_date_pg	cal_month	#,##0
name	Cal Month Na...	varchar(7)	stage_date_pg	cal_month_name	
_no	Cal Quarter No	integer	stage_date_pg	cal_quarter_no	#,##0
	Cal Quarter	integer	stage_date_pg	cal_quarter	#,##0

Message

There are no items to show.

ane status: Currently no active display in this pane Repository: Unknown, Not connected User Id: unknown user Browse: No source system

You can review the composition of existing hash keys for a **Data Vault Stage** table (Hub, Link and Satellite) and create additional hash keys by selecting the **Maintain DV Hash Key Columns** option from the selected **Data Vault Stage Table**'s context menu. This launches the hash key generation Wizard which enables you to maintain the source columns defined for the hash keys. Refer to [Creating Data Vault Stage Tables](#) for details.


Maintain Hash Key Columns

Please select columns to use in the hash by moving columns from the Available list to the Selected list. Enter a name for the Hash then use the Add button to add the Hash to the list of Hub Hash keys below.

Available Columns:

code
description
prod_line
prod_group
subgroup

Selected Columns:

Hash Key Name:

Hash Key Column	Source Columns
h_prod_code	code
h_prod_description	description
h_prod_line	prod_line

Note

To remove or change a hash key column, you need to delete it first, e.g. right click the column listed in the middle pane and then select **Delete Column** from the context menu.

Building Data Vault Objects

To build Data Vault objects (Hub, Link and Satellite) in WhereScape RED, involves the following procedures.

1. Creating Load Tables with the required DSS columns.
2. Creating Data Vault Stage tables.
3. Generating update procedures for the Stage table via templates.
4. Creating the Hub, Link and Satellite tables.
5. Generating update procedures for the Hub, Link and Satellite tables via templates.

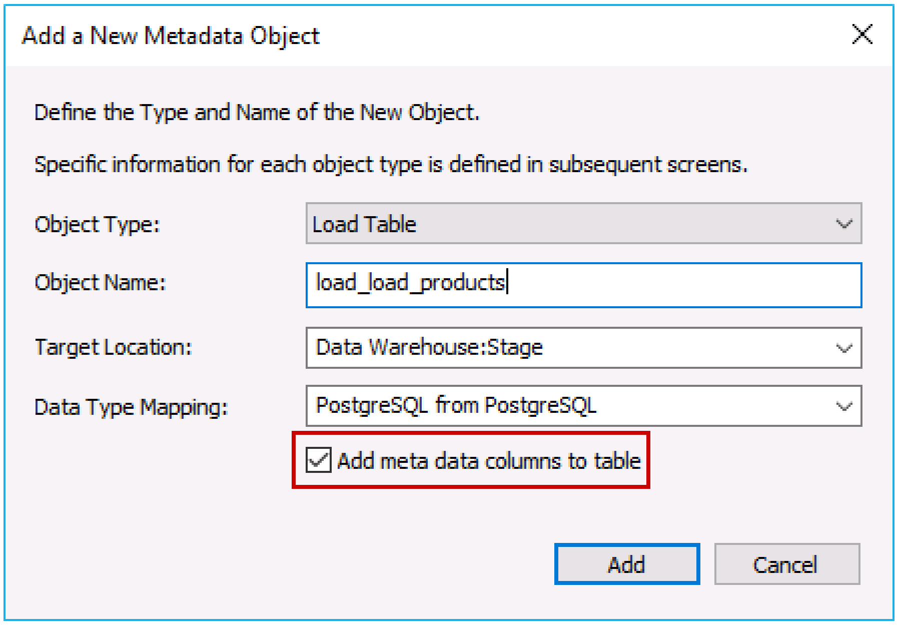
The detailed steps for each procedure are outlined in the following sections.

Creating Load Tables

The following describe the steps for creating a **Load** table:

1. Browse to the source system connection required (**Browse > Source Tables**).

2. Double-click the **Load Table** object group in the left pane, the middle pane displays a list of existing Load tables.
3. Click the source table from the right pane and drag it to the middle pane. You need to create the **Load** table with the required DSS columns—the option to add default meta data columns to the load table must be selected:



Add a New Metadata Object

Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: Load Table

Object Name: load_load_products

Target Location: Data Warehouse:Stage

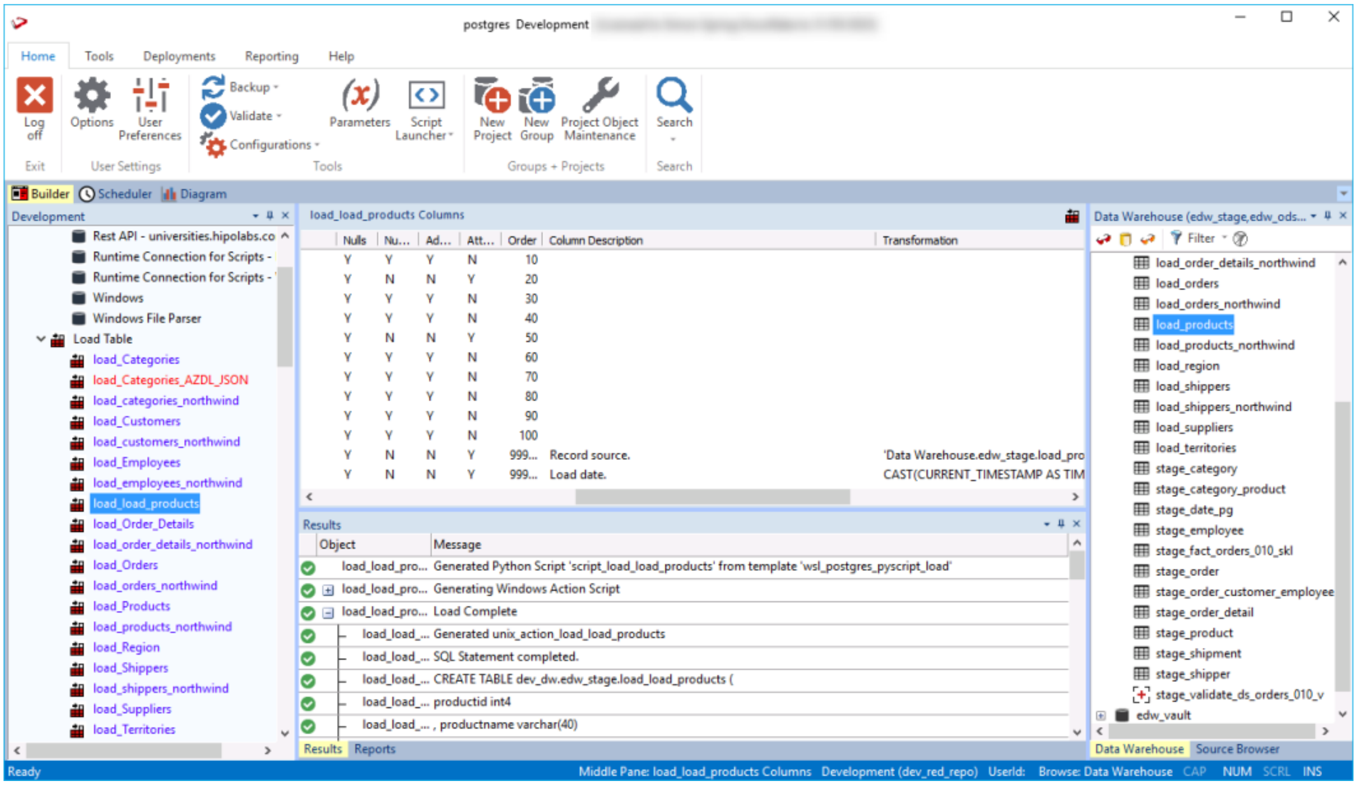
Data Type Mapping: PostgreSQL from PostgreSQL

☑ Add meta data columns to table

When the build table operation is performed, the Load table created has the two additional columns which are populated by transformations:

- **dss_record_source** – the connection or source for the load table.
- **dss_load_date** – the date when the data was loaded to the table. This is updated every time a load operation is performed.

These DSS columns added, include column description and transformation information.



load_load_products Columns

Nulls	Nu...	Ad...	Att...	Order	Column Description	Transformation
Y	Y	Y	N	10		
Y	N	N	Y	20		
Y	Y	Y	N	30		
Y	Y	Y	N	40		
Y	N	N	Y	50		
Y	Y	Y	N	60		
Y	Y	Y	N	70		
Y	Y	Y	N	80		
Y	Y	Y	N	90		
Y	Y	Y	N	100		
Y	N	N	Y	999...	Record source.	'Data Warehouse.edw_stage.load_pro
Y	N	N	Y	999...	Load date.	CAST(CURRENT_TIMESTAMP AS TIM

Results

Object	Message
load_load_pro...	Generated Python Script 'script_load_load_products' from template 'wsl_postgres_pyscript_load'
load_load_pro...	Generating Windows Action Script
load_load_pro...	Load Complete
load_load_...	Generated unix_action_load_load_products
load_load_...	SQL Statement completed.
load_load_...	CREATE TABLE dev_dw.edw_stage.load_load_products (
load_load_...	productid int4
load_load_...	, productname varchar(40)

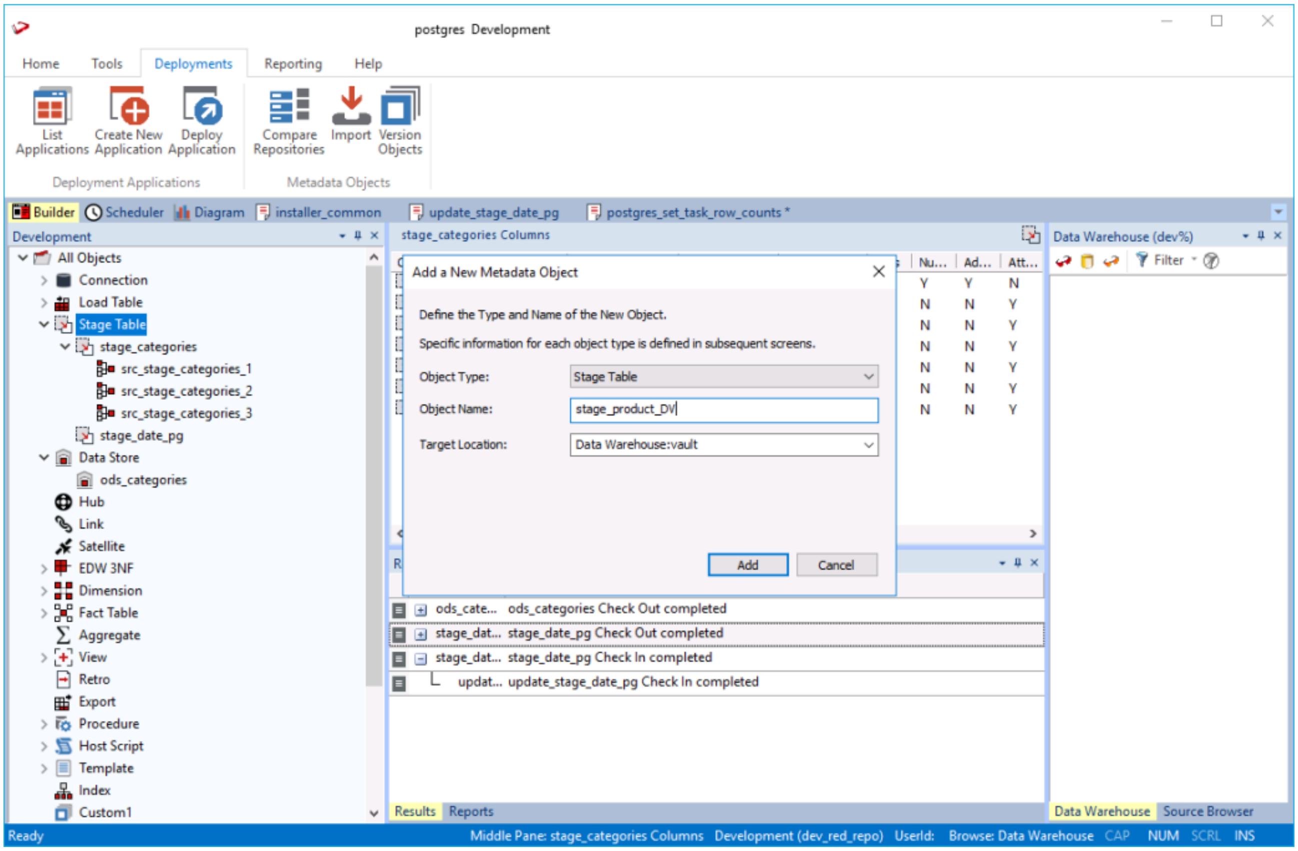
Notes

1. Custom aliases for the **dss_load_date** and **dss_record_source** columns can be defined in the **Tools > Options > DSS Tables and Columns > DSS Columns to Include and Naming** window. If custom DSS aliases are defined, then the aliases are used for the column names that are added to the new Load tables. Refer to **DSS Columns** for details.
2. The **dss_record_source** column transformation is set to the name of the source connection when the table is first created. If the source connection changes, the transformation value does not update automatically. You can manually update the transformation—refer to **Transformations** for details

Creating Data Vault Stage Tables

The following describe the steps for creating a **Data Vault Stage** table:

1. Browse to the Data Warehouse (click **Browse > Data Warehouse**) connection to create the **Data Vault Stage** table.
2. Double-click the **Stage** object group in the left pane, the middle pane displays a list of existing **Stage** tables.
3. Click the source **Load** table from the right pane and drag it to the middle pane. The selected **Load** table must have the required DSS columns (**dss_record_source** and **dss_load_date**).



Add a New Metadata Object

Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: Stage Table

Object Name: stage_product_DV

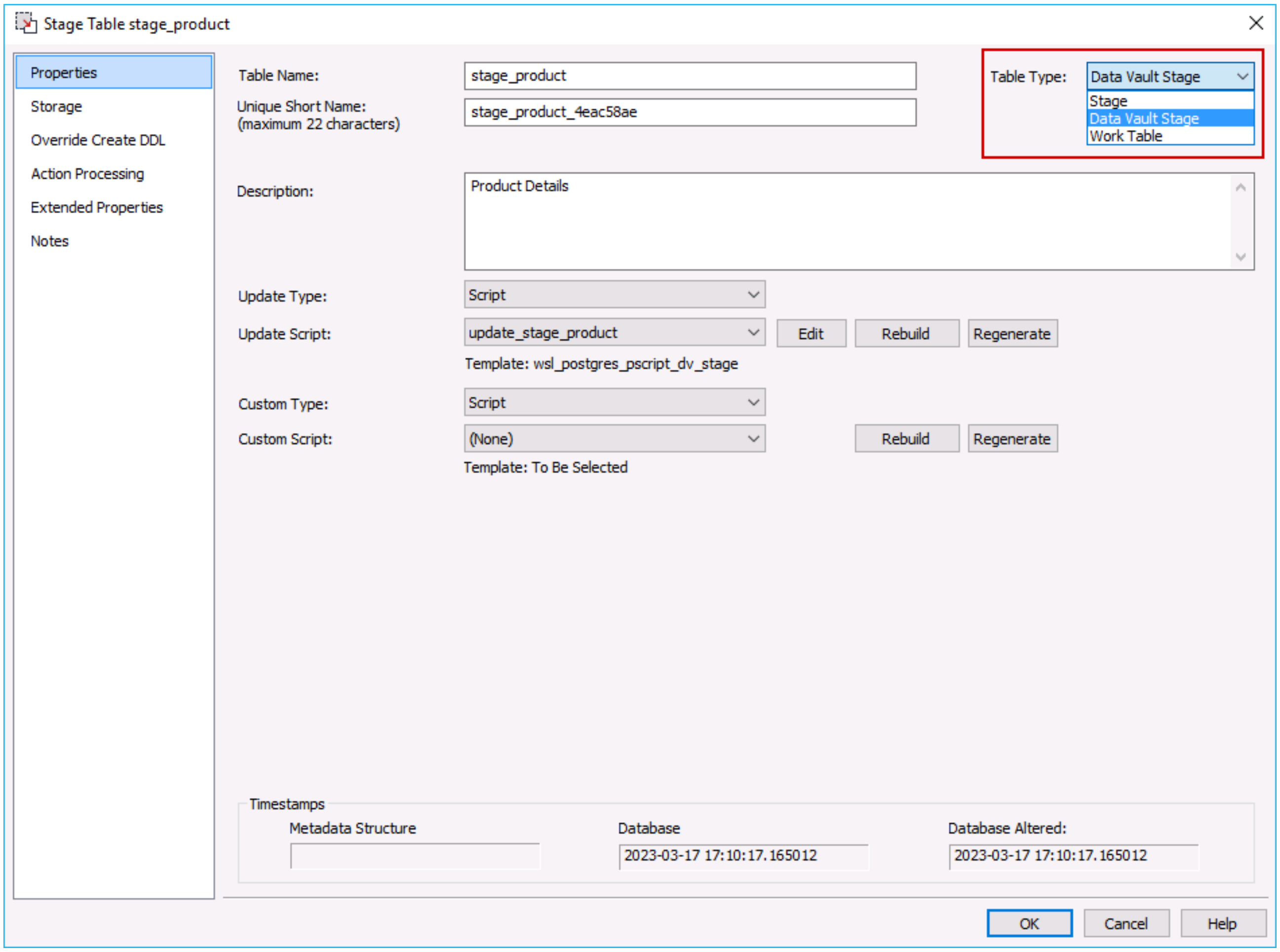
Target Location: Data Warehouse:vault

Add Cancel

```
ods_cate... ods_categories Check Out completed
stage_dat... stage_date_pg Check Out completed
stage_dat... stage_date_pg Check In completed
  updat... update_stage_date_pg Check In completed
```

Ready   Middle Pane: stage_categories Columns Development (dev_red_repo) UserId: Browse: Data Warehouse CAP NUM SCRL INS

4. The **Add a New Metadata Object** screen appears and classifies the new object as a **Stage** table. It provides a default name based on the **Load** table name. Accept the name or enter a new name for the **Stage** table and click **ADD** to continue.
5. On the **Table Properties** screen, select the **Data Vault Stage** option from the **Table Type** drop-down.



Stage Table stage_product

Table Name: stage_product

Unique Short Name: (maximum 22 characters) stage_product_4eac58ae

Table Type: Data Vault Stage

Description: Product Details

Update Type: Script

Update Script: update_stage_product

Template: wsl_postgres_pscript_dv_stage

Custom Type: Script

Custom Script: (None)

Template: To Be Selected

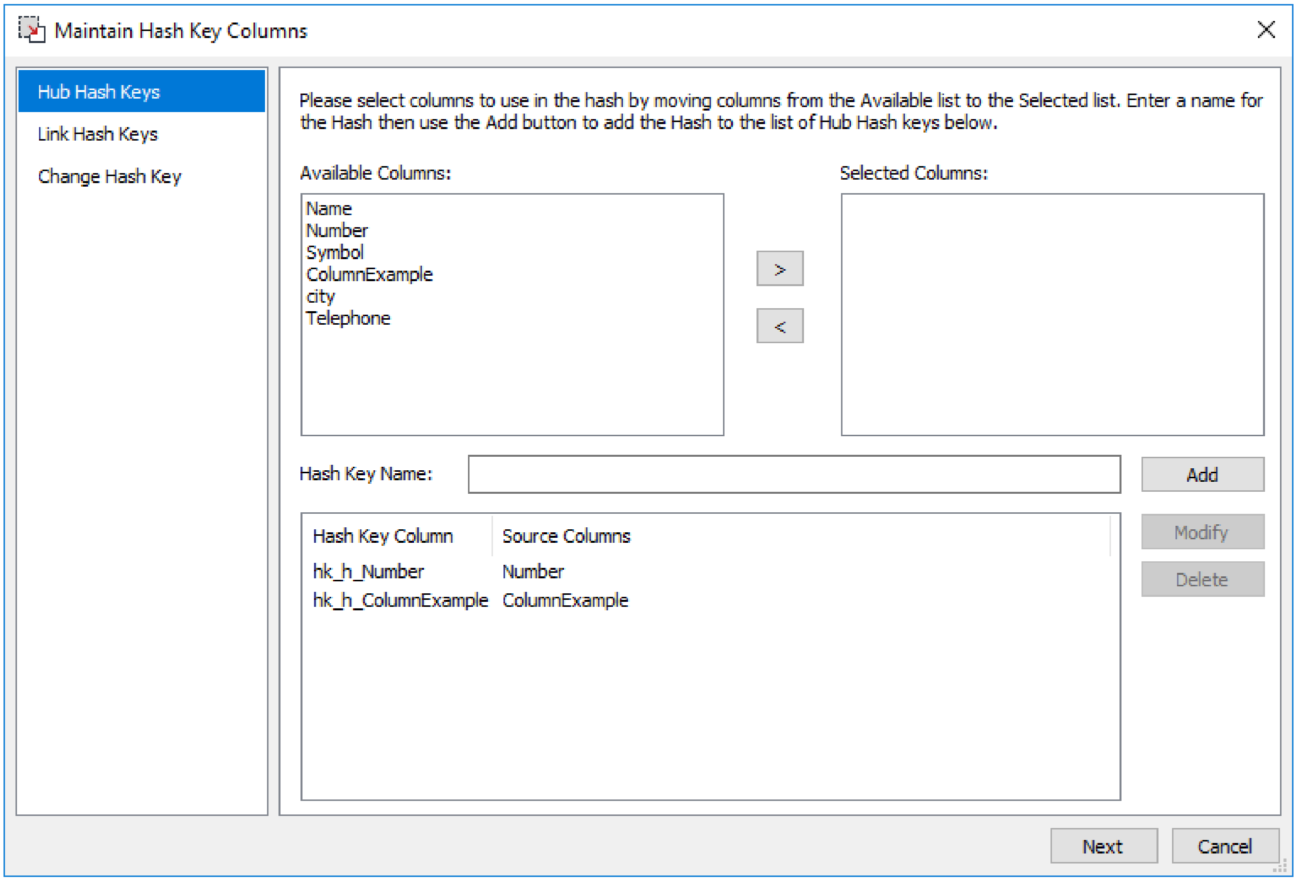
Timestamps

Metadata Structure	Database	Database Altered:
	2023-03-17 17:10:17.165012	2023-03-17 17:10:17.165012

Notes

1. You can set this table type to be the default by configuring the **Default Sub Type for Stage Table Objects** setting in the **Tools > Options > Object Types > Object Sub Types** screen.
2. RED displays the name of the previously used update procedure template by default, below the **Update Procedure** drop-down.

6. Click **OK** in the table **Properties** screen to launch the Wizard that enables you to define the source columns for the **Hub**, **Link** and **Change** hash key columns.

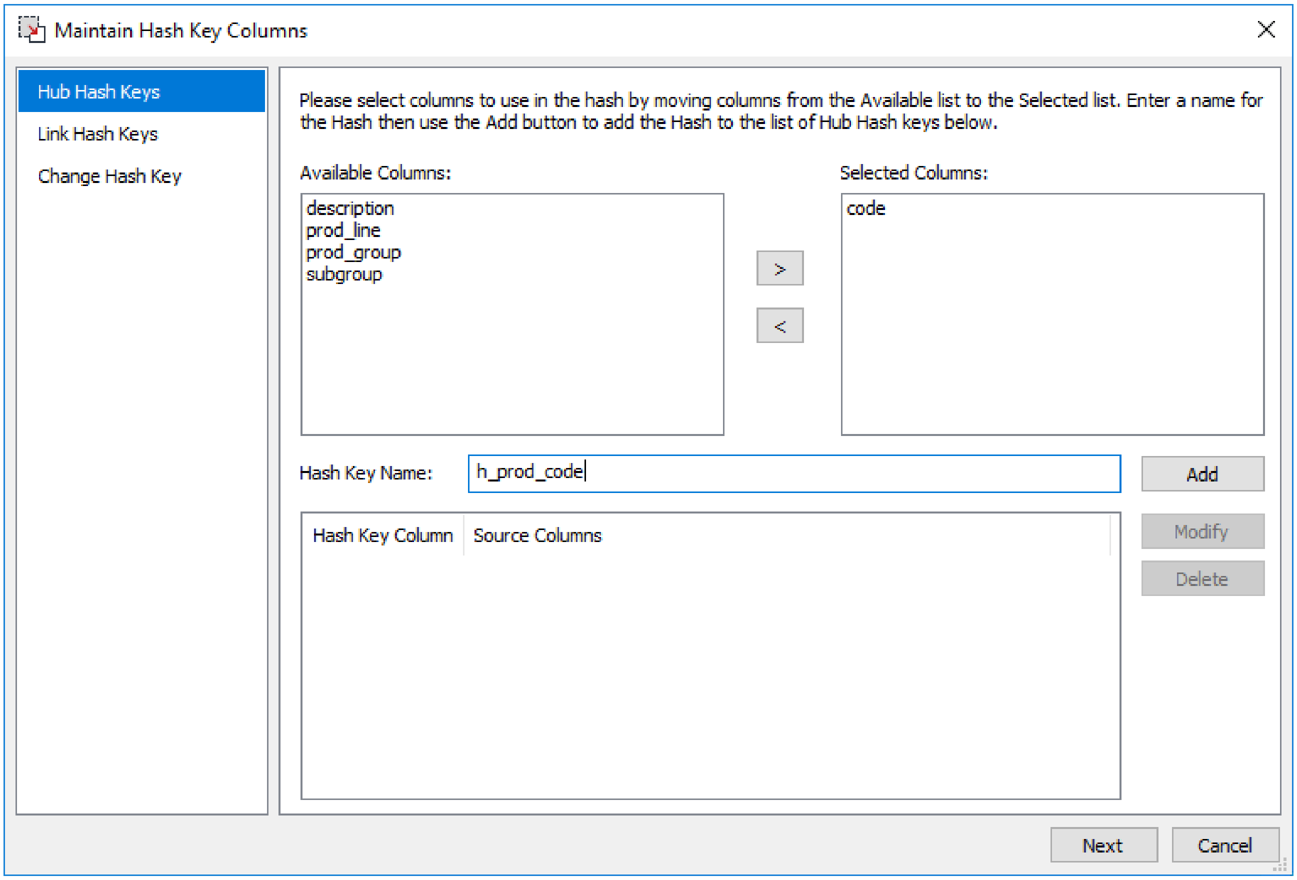


Please select columns to use in the hash by moving columns from the Available list to the Selected list. Enter a name for the Hash then use the Add button to add the Hash to the list of Hub Hash keys below.

Available Columns: Name, Number, Symbol, ColumnExample, city, Telephone

Hash Key Column	Source Columns
hk_h_Number	Number
hk_h_ColumnExample	ColumnExample

7. Select the source column(s) to use in defining the first Hub hash key. The **Hash Key Name** is formed based on the prefix (defined in the **Tools > Options > Global Naming Conventions > Global Naming of Key Columns** settings) and the source column(s) name. You can manually change the name if required.



Please select columns to use in the hash by moving columns from the Available list to the Selected list. Enter a name for the Hash then use the Add button to add the Hash to the list of Hub Hash keys below.

Available Columns:

description
prod_line
prod_group
subgroup

Selected Columns:

code

Hash Key Name: h_prod_code

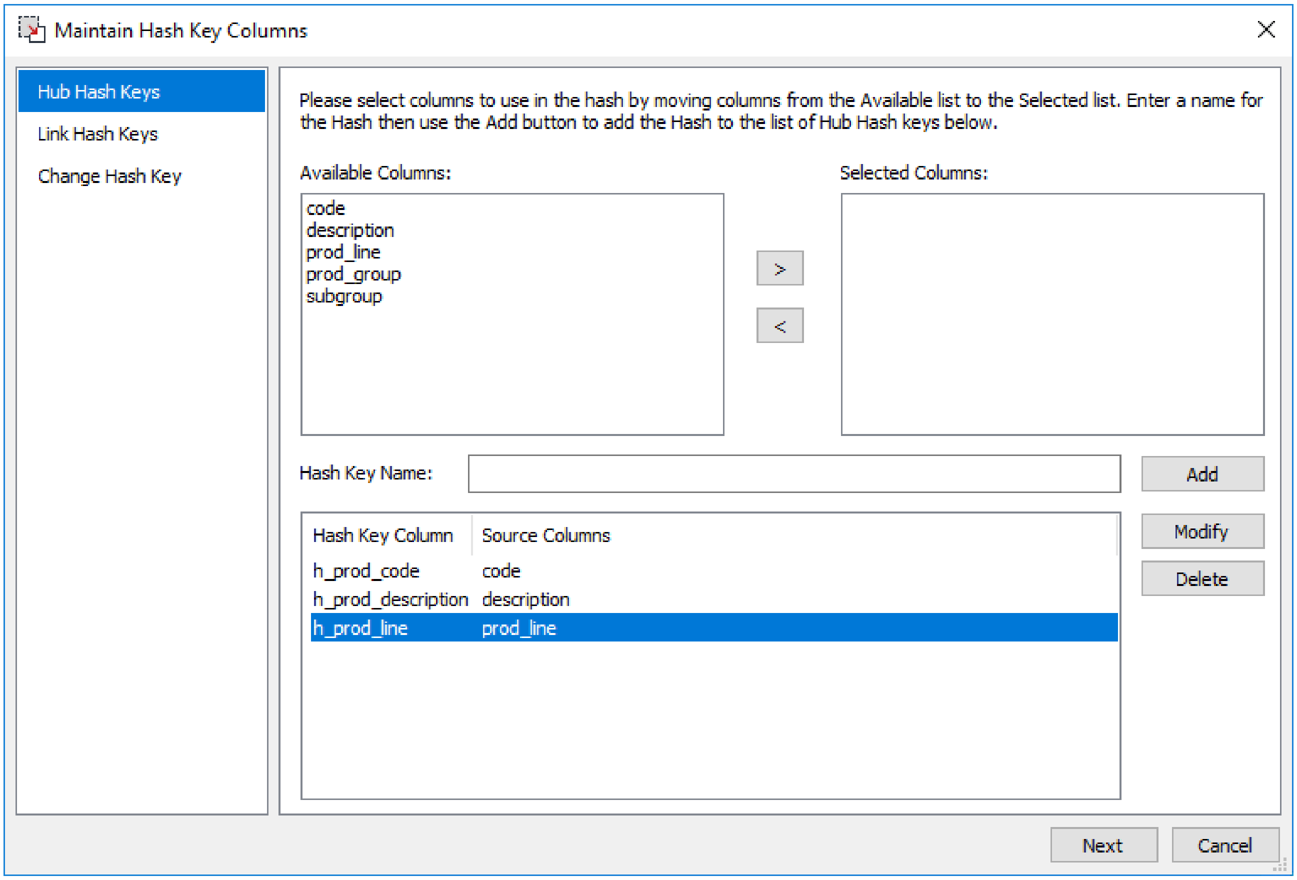
Hash Key Column	Source Columns

8. Click **Add** to create the first Hub hash key. Repeat the same steps as required. Use the **Modify** button to edit a selected hash key. The **Delete** button enables you to delete a selected hash key. Repeat the same steps as required.

Note

A confirmation message is displayed when you attempt to modify or delete a hash key that is being used by other objects in RED.

9. Once all the required **Hub Hash Keys** are created, click **Next** to progress to the **Link** hash key generation screen.



Please select columns to use in the hash by moving columns from the Available list to the Selected list. Enter a name for the Hash then use the Add button to add the Hash to the list of Hub Hash keys below.

Available Columns:

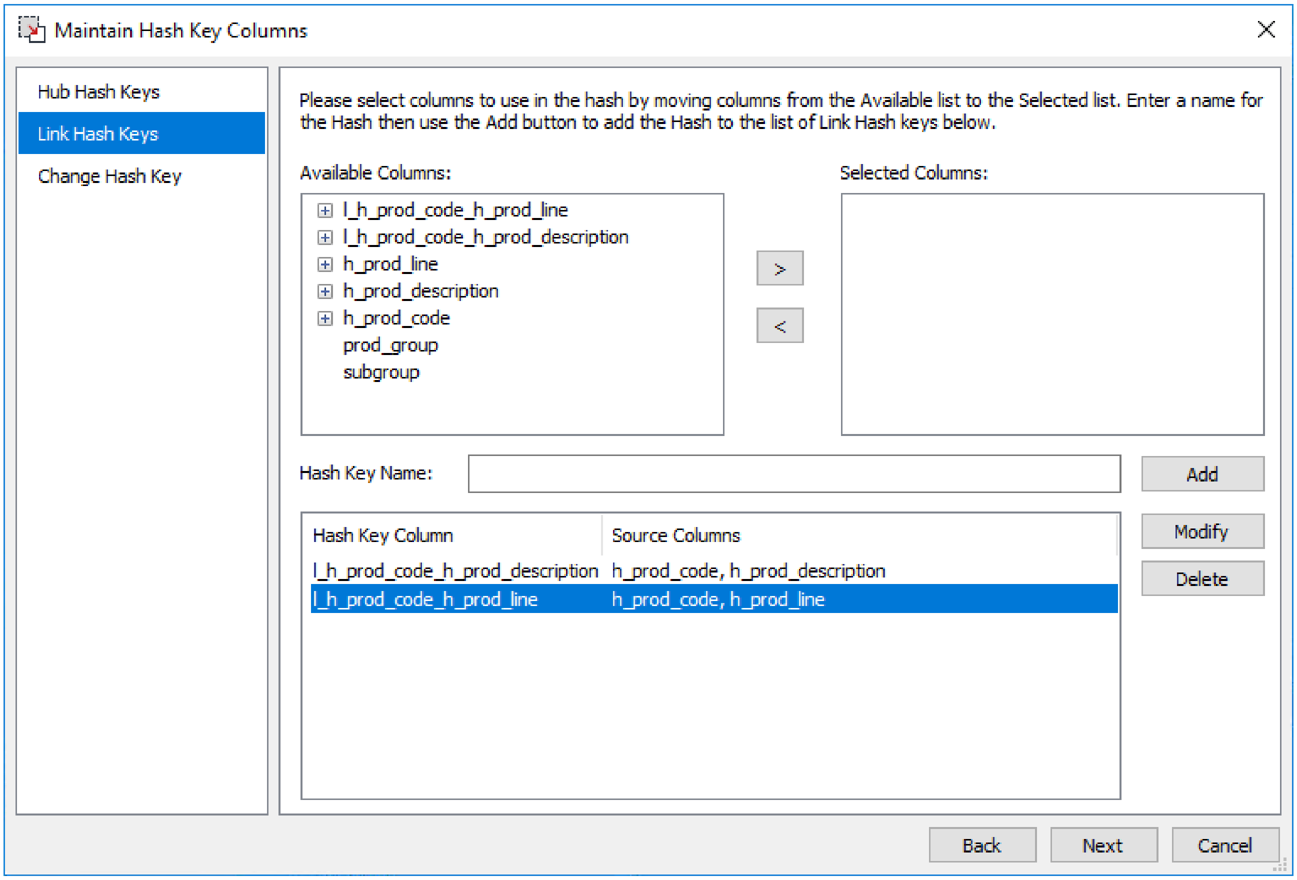
code
description
prod_line
prod_group
subgroup

Selected Columns:

Hash Key Name:

Hash Key Column	Source Columns
h_prod_code	code
h_prod_description	description
h_prod_line	prod_line

10. Creating the **Link Hash Keys** involves the same process, follow the previous steps (6 and 7) to select *multiple* source columns to combine to create the **Link** hash key. Click **Add** to create the first Link hash key. Repeat the same steps for any subsequent keys. Once all the required **Link Hash Keys** are created, click **Next** to progress to the **Change** hash key generation screen.



Please select columns to use in the hash by moving columns from the Available list to the Selected list. Enter a name for the Hash then use the Add button to add the Hash to the list of Link Hash keys below.

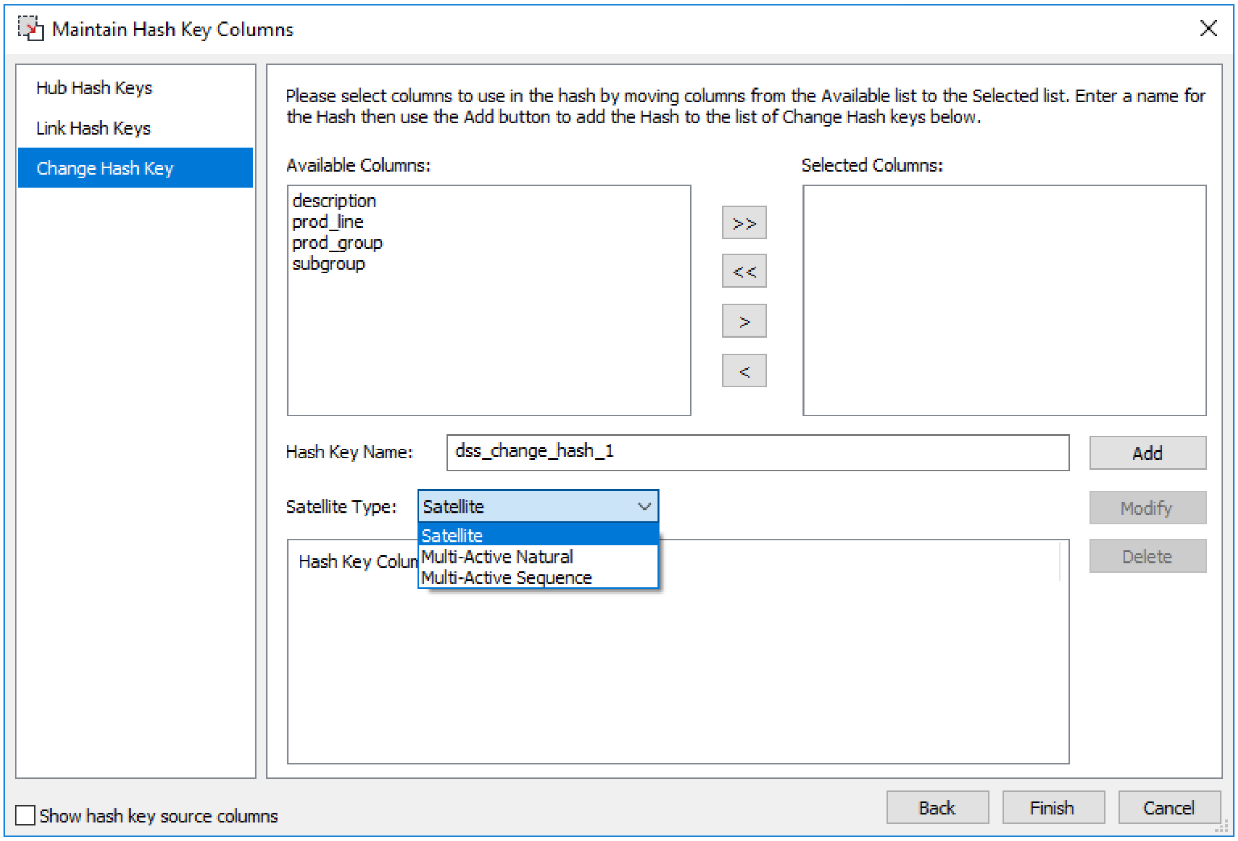
Hash Key Column	Source Columns
l_h_prod_code_h_prod_description	h_prod_code, h_prod_description
l_h_prod_code_h_prod_line	h_prod_code, h_prod_line

Notes

1. Custom aliases for the **dss_change_hash** column can be defined in the **Tools > Options > DSS Tables and Columns > DSS Columns to Include and Naming** window. If custom DSS aliases are defined then the aliases are used for the column names that are added to the new Load tables. Refer to **DSS Columns** for details.
2. The data type defined for the **dss_change_hash** column in **Tools > Options > Data Vault** window is applied when creating a satellite change hash key. Refer to **Data Vault Hash Keys setting** for details.

11. Follow the same steps to select the columns to use for the **Change Hash Key**. Select the **Show hash key source columns** check-box below the screen to display and use hash key source columns in defining the Change Hash keys. Also, select from the **Satellite Type** drop-down list the option to define your Multi-Active key:

    1. **Satellite**: Select the columns from the Available columns list by moving them to the Selected Columns box, type a name in the Hash Key Name, and click **Add**. Also, any Hash Key can be edited, by clicking **Modify**; otherwise, you can click **Delete** to remove it from the list.



Please select columns to use in the hash by moving columns from the Available list to the Selected list. Enter a name for the Hash then use the Add button to add the Hash to the list of Change Hash keys below.

Available Columns:
description
prod_line
prod_group
subgroup

Selected Columns:

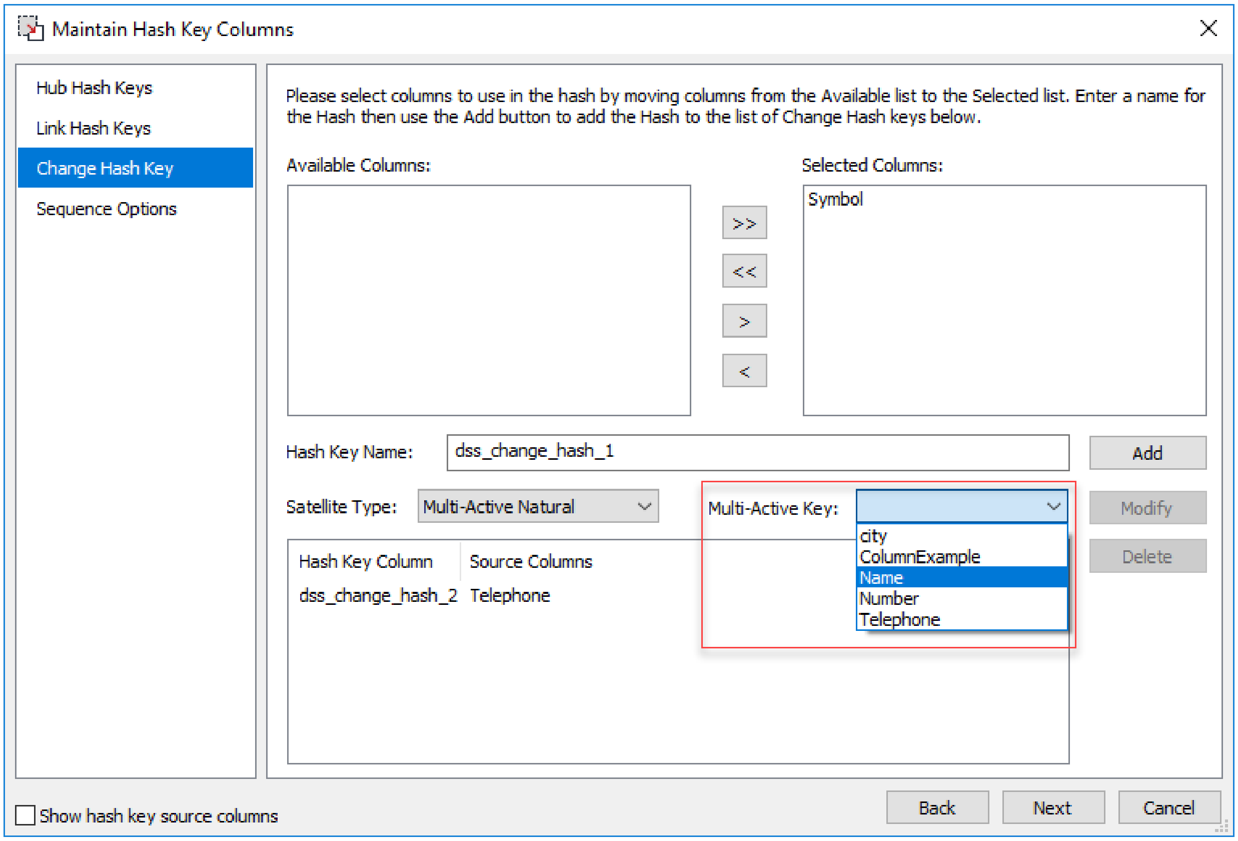
Hash Key Name: dss_change_hash_1

Satellite Type: Satellite
Satellite
Multi-Active Natural
Multi-Active Sequence

Hash Key Colun

□ Show hash key source columns

2. **Multi-Active Natural**: Select the columns from the Available columns list by moving them to the Selected Columns box, type a name in the Hash Key Name box, and click **Add**. Also, any Hash Key can be edited, by clicking **Modify**; otherwise, you can click **Delete** to remove it from the list.
   When you select this option, the **Multi-Active Key** drop-down list displays, allowing you to select any stage table column except for any DSS or selected column for the Change Hash Key sources.



Maintain Hash Key Columns

Please select columns to use in the hash by moving columns from the Available list to the Selected list. Enter a name for the Hash then use the Add button to add the Hash to the list of Change Hash keys below.

Available Columns:

Selected Columns: Symbol

Hash Key Name: dss_change_hash_1

Satellite Type: Multi-Active Natural

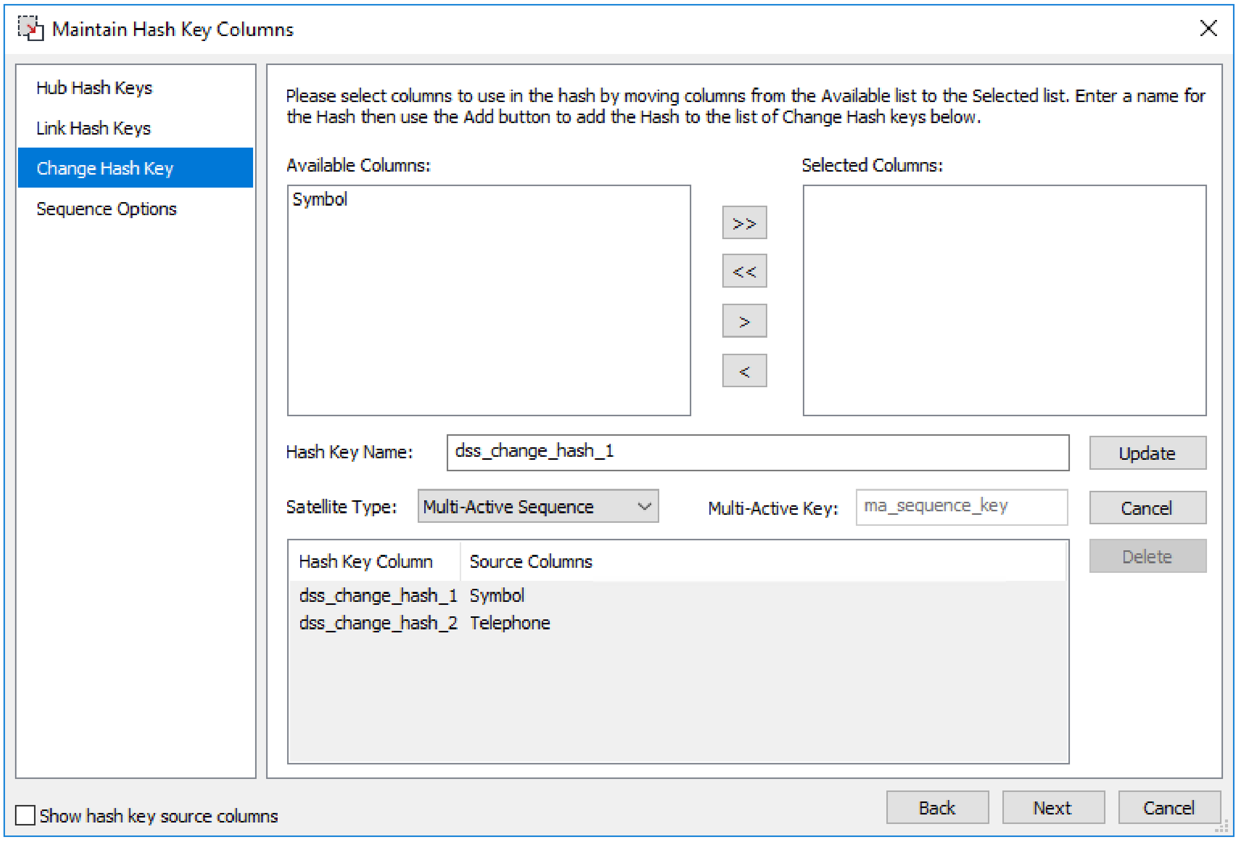
Multi-Active Key:

Hash Key Column	Source Columns
dss_change_hash_2	Telephone

☐ Show hash key source columns

3. **Multi-Active Sequence**:Select the columns from the Available columns list by moving them to the Selected Columns box, type a name in the Hash Key Name option, and click **Add**. Also, any Hash Key can be edited, by clicking **Modify**; otherwise, you can click **Delete** to remove it from the list.

   This option displays the **Multi-Active Key** box with the 'ma_sequence_key' name, by default. When a Multi-Active Sequence key is created, the **Sequence Options** tab appears.



Please select columns to use in the hash by moving columns from the Available list to the Selected list. Enter a name for the Hash then use the Add button to add the Hash to the list of Change Hash keys below.

Available Columns: Symbol

Selected Columns:

Hash Key Name: dss_change_hash_1

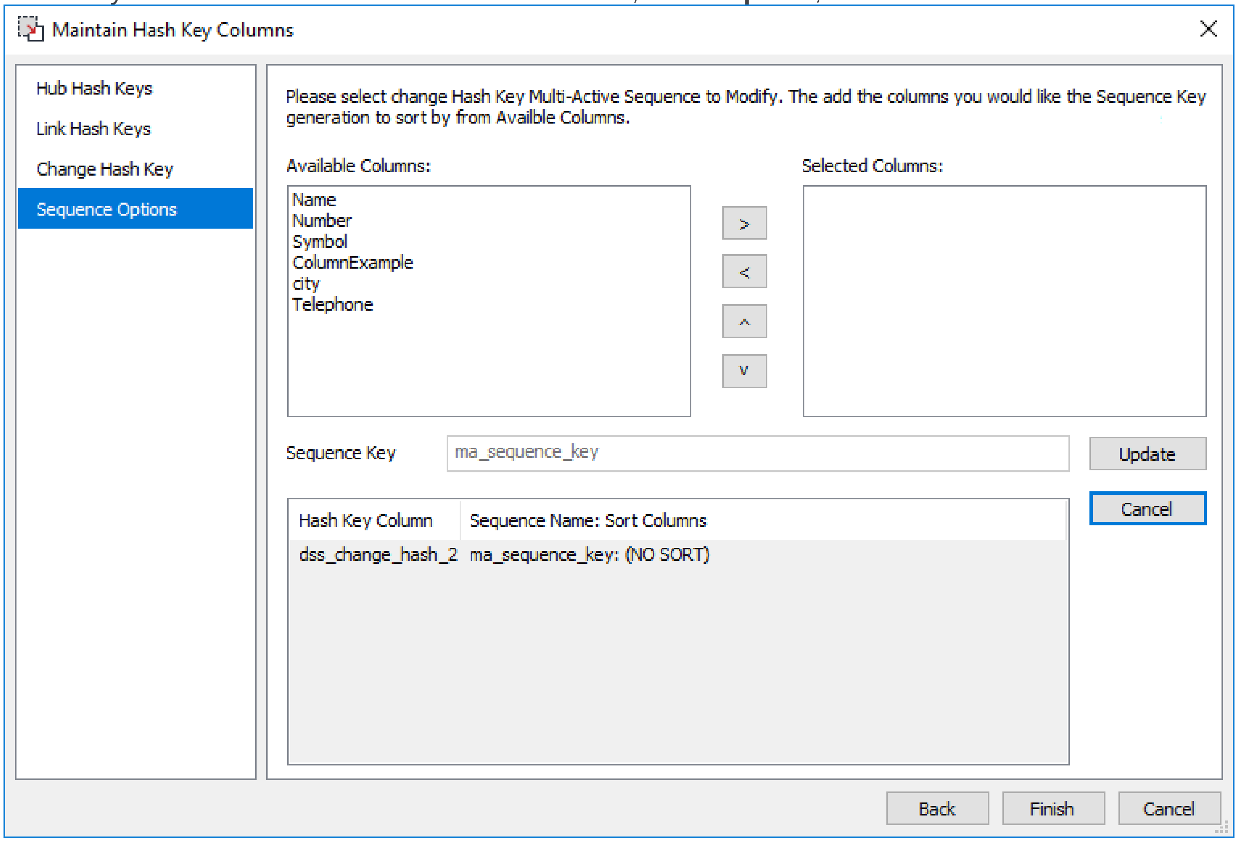
Satellite Type: Multi-Active Sequence

Multi-Active Key: ma_sequence_key

Hash Key Column	Source Columns
dss_change_hash_1	Symbol
dss_change_hash_2	Telephone

□ Show hash key source columns

4. **Sequence Options**: Select a Hash Key Multi-Active Key Sequence and click **Modify**. Add the columns you want to sort from the Available Columns, select **Update**, and click **Finish**.



Please select change Hash Key Multi-Active Sequence to Modify. The add the columns you would like the Sequence Key generation to sort by from Availble Columns.

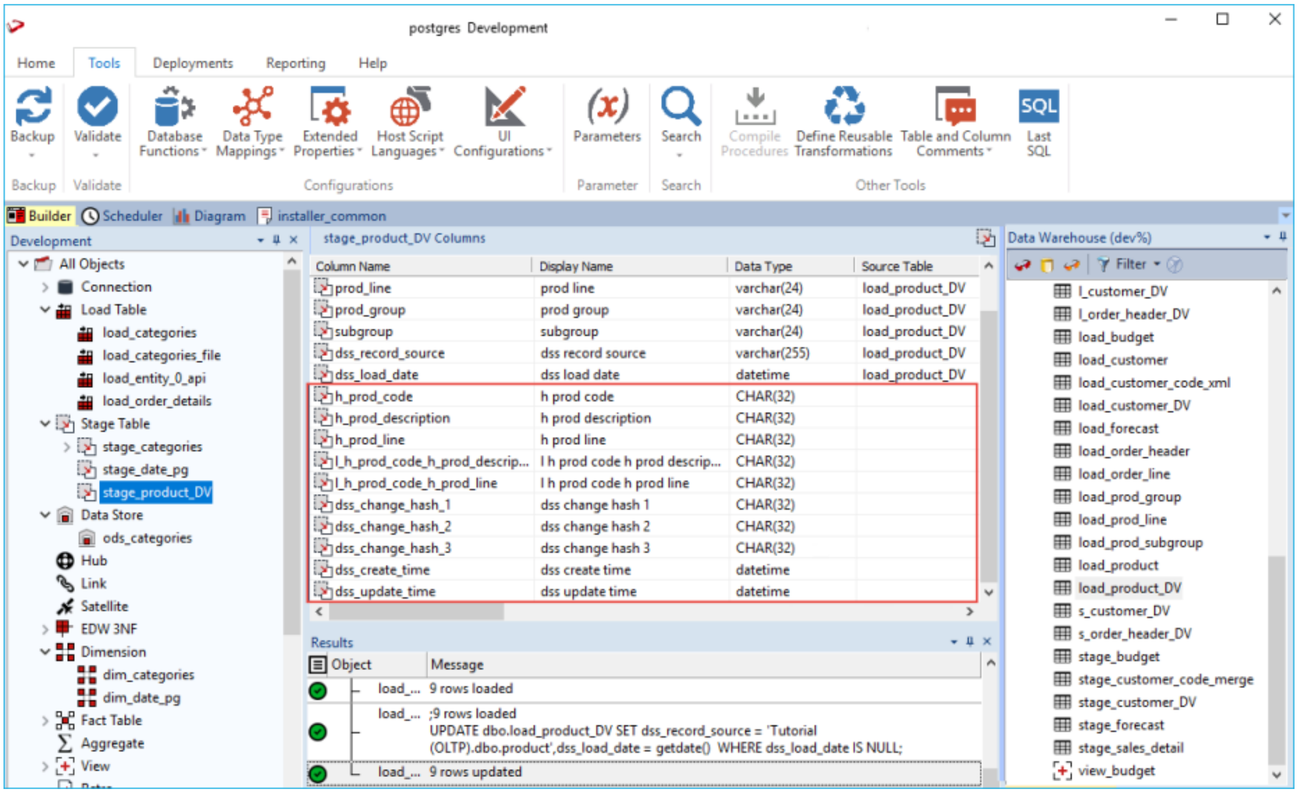
Available Columns:
Name
Number
Symbol
ColumnExample
city
Telephone

Selected Columns:

Sequence Key ma_sequence_key

Hash Key Column	Sequence Name: Sort Columns
dss_change_hash_2	ma_sequence_key: (NO SORT)

12. Click **Finish**, once you have defined the required list of descriptive columns for the Change hash key. The new **Data Vault Stage** table is added to the **Stage Table** objects list in the left pane and the columns included in the table are listed in the middle pane.



stage_product_DV Columns

Column Name	Display Name	Data Type	Source Table
prod_line	prod line	varchar(24)	load_product_DV
prod_group	prod group	varchar(24)	load_product_DV
subgroup	subgroup	varchar(24)	load_product_DV
dss_record_source	dss record source	varchar(255)	load_product_DV
dss_load_date	dss load date	datetime	load_product_DV
h_prod_code	h prod code	CHAR(32)	
h_prod_description	h prod description	CHAR(32)	
h_prod_line	h prod line	CHAR(32)	
l_h_prod_code_h_prod_descrip...	l h prod code h prod descrip...	CHAR(32)	
l_h_prod_code_h_prod_line	l h prod code h prod line	CHAR(32)	
dss_change_hash_1	dss change hash 1	CHAR(32)	
dss_change_hash_2	dss change hash 2	CHAR(32)	
dss_change_hash_3	dss change hash 3	CHAR(32)	
dss_create_time	dss create time	datetime	
dss_update_time	dss update time	datetime	

Results

Object	Message
load_...	9 rows loaded
load_...	;9 rows loaded UPDATE dbo.load_product_DV SET dss_record_source = 'Tutorial (OLTP).dbo.product',dss_load_date = getdate() WHERE dss_load_date IS NULL;
load_...	9 rows updated

In addition to the columns defined from the Load table, the following columns and their metadata have been added to the Data Vault Stage table:

- The Hub Hash Keys
- The Link Hash Keys
- The Change Hash Key
- The DSS_CREATE_TIME column
- The DSS_UPDATE_TIME column

The metadata for the hash columns include the source columns that were used to create them (used to generate the hash keys).

The hash keys created are used in the subsequent Data Vault object (Hub, Links and Satellites) creation Wizards.

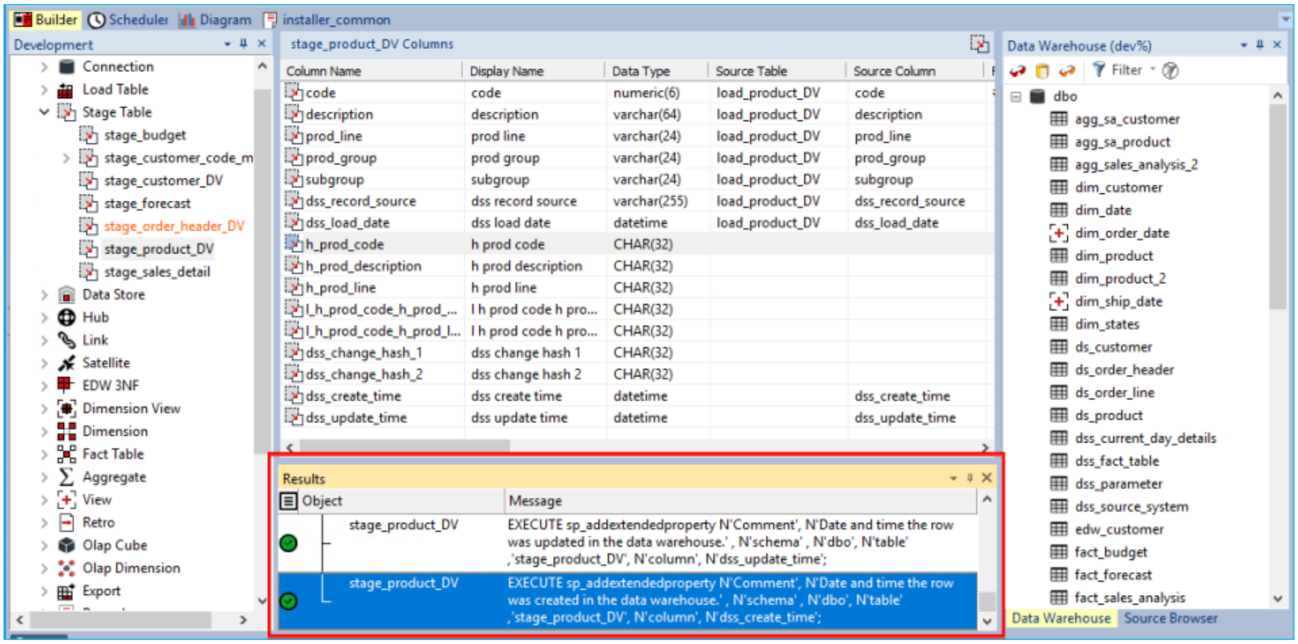
Tip

You can review the composition of existing hash keys for a Data Vault Stage table (Hub, Link and Satellite) and create additional hash keys by selecting the **Maintain DV Hash Key Columns** option from the selected **Data Vault Stage** table's context menu. This launches the hash key generation Wizard which enables you to maintain the source columns defined for the hash keys.

Note

To remove or change a hash key column, you need to delete it first, e.g. right click the column listed in the middle pane and then select **Delete Column** from the context menu.

13. Right-click the new **Data Vault Stage** table you defined from the left pane, under the **Stage Table** objects list and select **Create (ReCreate)** from the context menu to create the table. The **Results** pane displays confirmation that the **Data Vault Stage** table was successfully created.



stage_product_DV Columns

Column Name	Display Name	Data Type	Source Table	Source Column
code	code	numeric(6)	load_product_DV	code
description	description	varchar(64)	load_product_DV	description
prod_line	prod line	varchar(24)	load_product_DV	prod_line
prod_group	prod group	varchar(24)	load_product_DV	prod_group
subgroup	subgroup	varchar(24)	load_product_DV	subgroup
dss_record_source	dss record source	varchar(255)	load_product_DV	dss_record_source
dss_load_date	dss load date	datetime	load_product_DV	dss_load_date
h_prod_code	h prod code	CHAR(32)		
h_prod_description	h prod description	CHAR(32)		
h_prod_line	h prod line	CHAR(32)		
l_h_prod_code_h_prod_...	l h prod code h pro...	CHAR(32)		
l_h_prod_code_h_prod_l...	l h prod code h pro...	CHAR(32)		
dss_change_hash_1	dss change hash 1	CHAR(32)		
dss_change_hash_2	dss change hash 2	CHAR(32)		
dss_create_time	dss create time	datetime		dss_create_time
dss_update_time	dss update time	datetime		dss_update_time

Results

Object	Message
stage_product_DV	EXECUTE sp_addextendedproperty N'Comment', N'Date and time the row was updated in the data warehouse.' , N'schema' , N'dbo', N'table' ,'stage_product_DV', N'column', N'dss_update_time';
stage_product_DV	EXECUTE sp_addextendedproperty N'Comment', N'Date and time the row was created in the data warehouse.' , N'schema' , N'dbo', N'table' ,'stage_product_DV', N'column', N'dss_create_time';

Once the new Data Vault Stage table is defined and created, clicking the **Rebuild** button in the **Table Properties** screen launches the Wizard to generate the procedure to populate the table. This Wizard utilizes a template to create the procedure.

The detailed steps for using this Wizard is described in the next section, **Generating Update Procedures for the Data Vault Stage Table**.

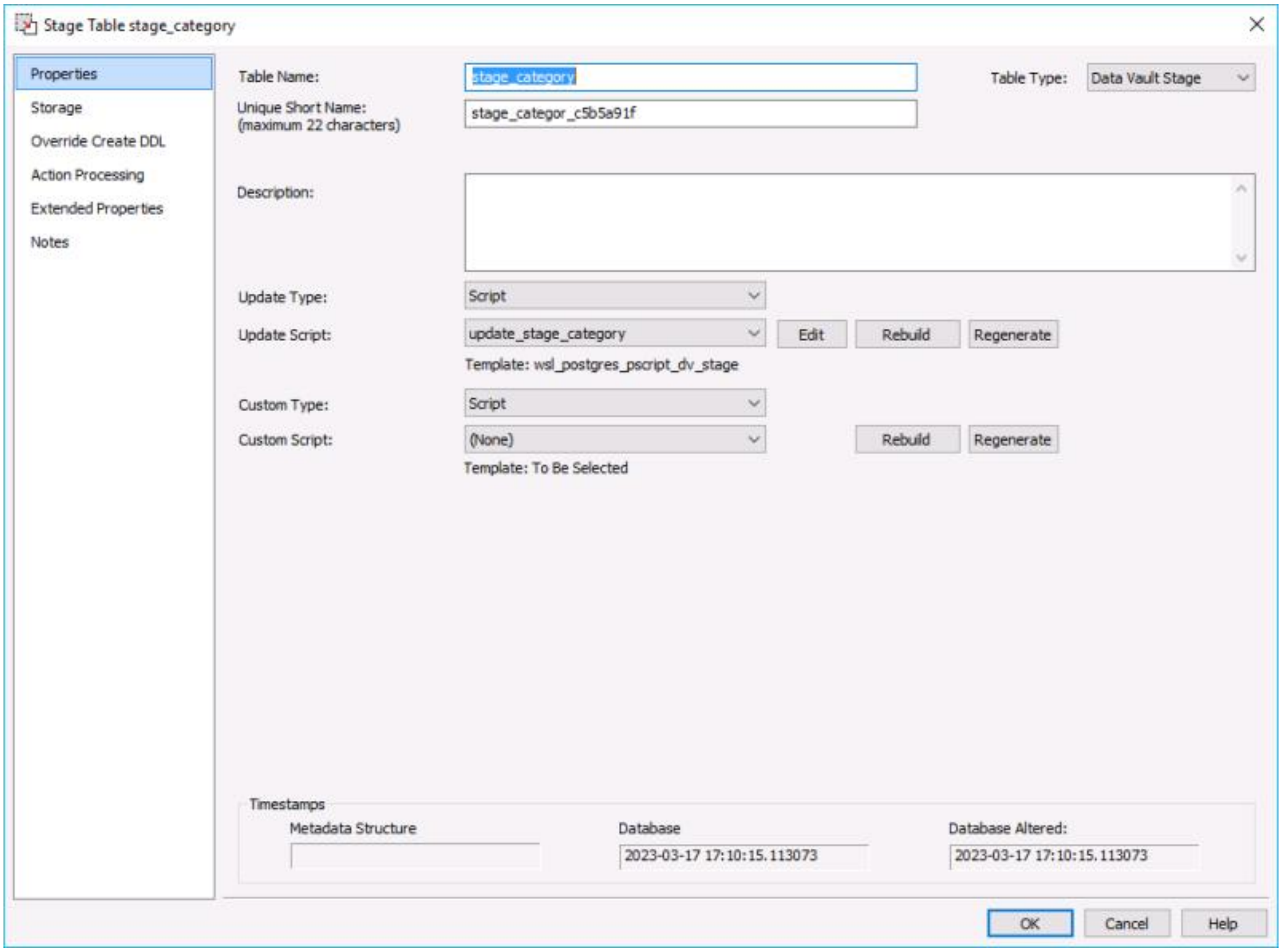
Generating Update Procedures for the Data Vault Stage Table

After successfully defining and creating the **Data Vault Stage** table, you can generate the update procedure via a template to populate the table.

Notes

- Please ensure that you have installed the WhereScape supplied templates (refer to **Data Vault Templates**) or created your own Data Vault templates, before performing the steps below.

1. Click the **Rebuild** button in the table **Properties** screen to launch the procedure generation Wizard to populate the table.



Stage Table stage_category

Table Name: stage_category

Table Type: Data Vault Stage

Unique Short Name: (maximum 22 characters) stage_categor_c5b5a91f

Description:

Update Type: Script

Update Script: update_stage_category

Template: wsl_postgres_pscript_dv_stage

Custom Type: Script

Custom Script: (None)

Template: To Be Selected

Timestamps

Metadata Structure:

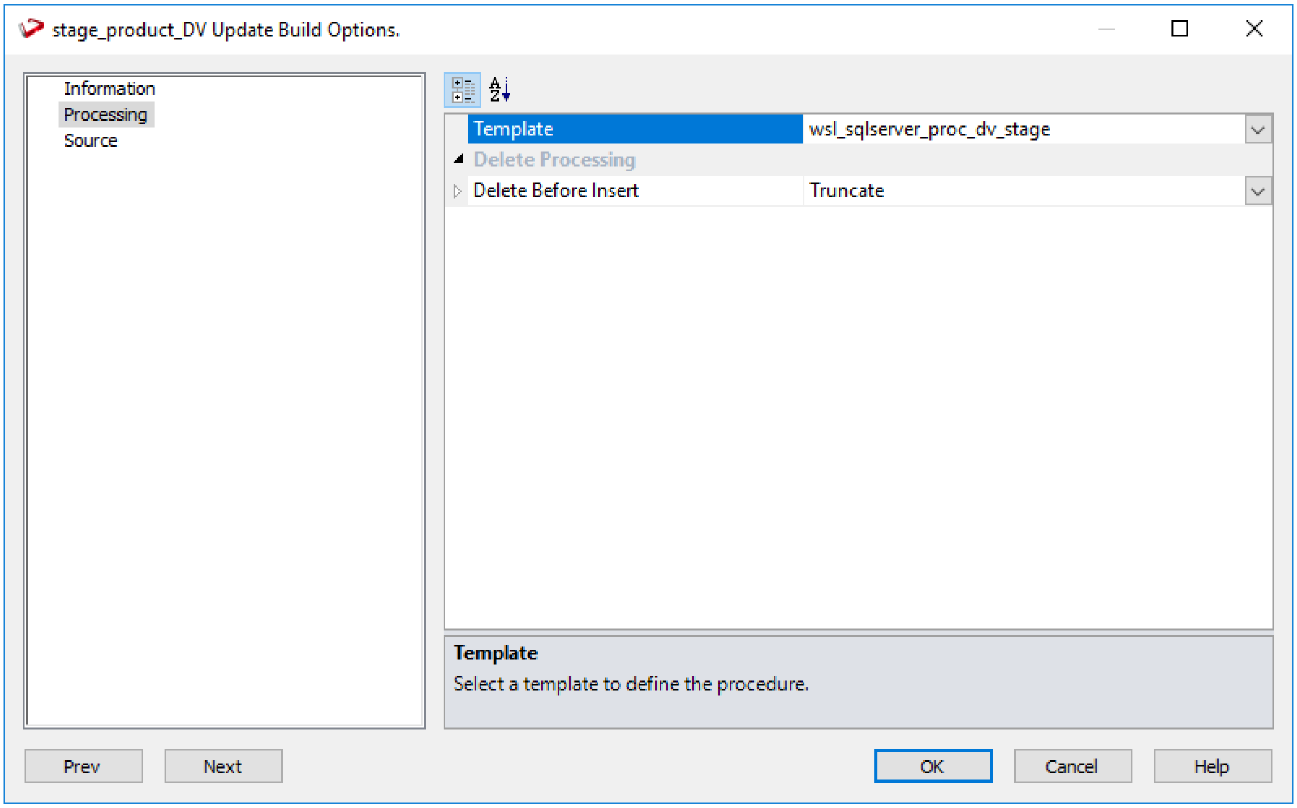
Database: 2023-03-17 17:10:15.113073

Database Altered: 2023-03-17 17:10:15.113073

Note

RED displays the name of the previously used update procedure template by default, below the **Update Procedure** drop-down.

2. On the **Processing** tab of **Table Update Build Options** screen, select the template to use from the **Template** drop-down or use the previous update procedure template.

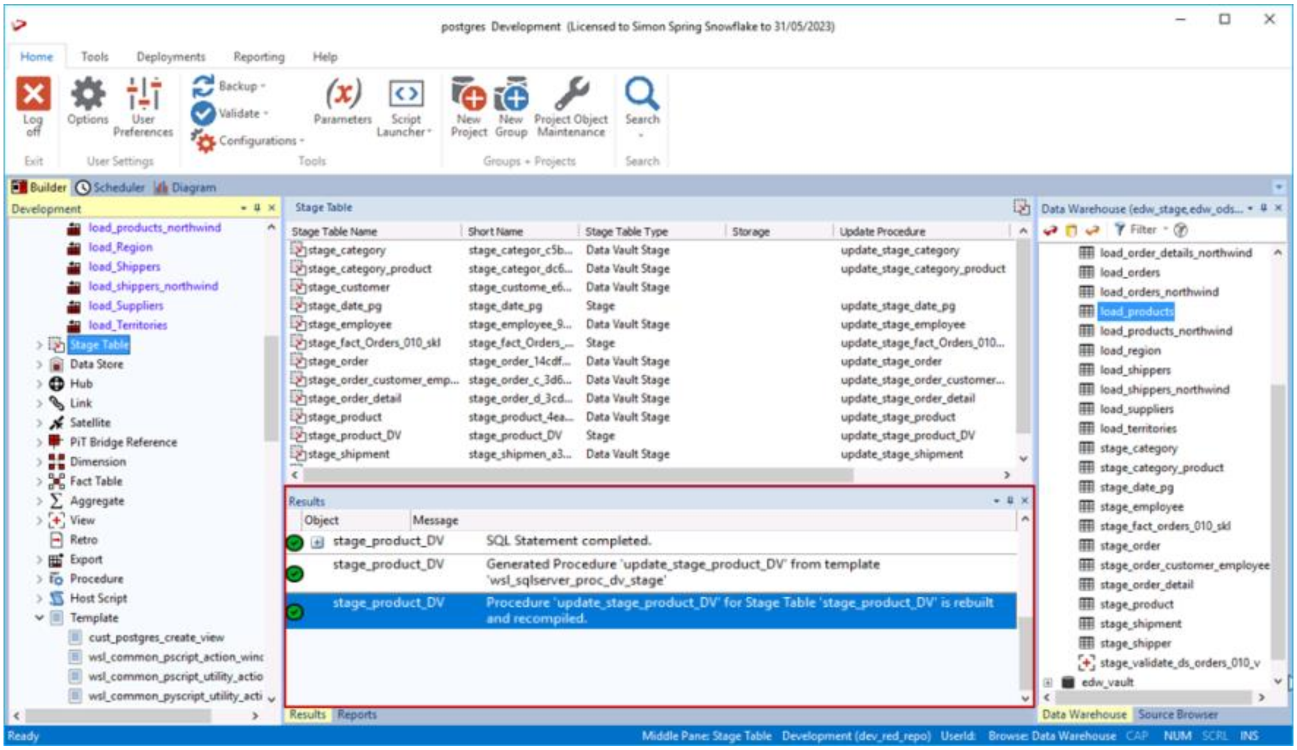


Template	wsl_sqlserver_proc_dv_stage
Delete Processing	
Delete Before Insert	Truncate

Template

Select a template to define the procedure.

3. Click **OK** to perform the procedure generation. The **Results** pane displays confirmation that the procedure was generated.



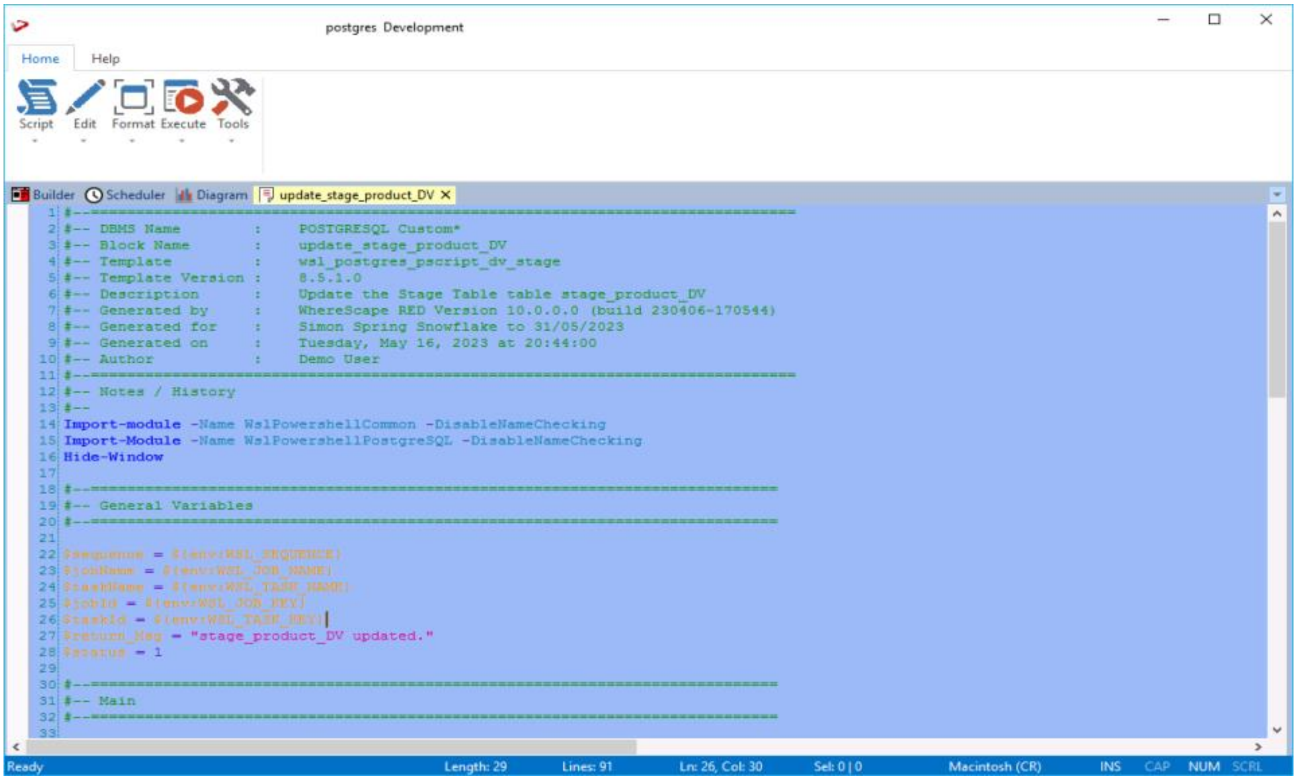
Stage Table

Stage Table Name	Short Name	Stage Table Type	Storage	Update Procedure
stage_category	stage_categor_c5b...	Data Vault Stage		update_stage_category
stage_category_product	stage_categor_dc6...	Data Vault Stage		update_stage_category_product
stage_customer	stage_custome_e6...	Data Vault Stage		
stage_date_pg	stage_date_pg	Stage		update_stage_date_pg
stage_employee	stage_employee_9...	Data Vault Stage		update_stage_employee
stage_fact_Orders_010_skl	stage_fact_Orders_...	Stage		update_stage_fact_Orders_010...
stage_order	stage_order_14cdf...	Data Vault Stage		update_stage_order
stage_order_customer_emp...	stage_order_c_3d6...	Data Vault Stage		update_stage_order_customer...
stage_order_detail	stage_order_d_3cd...	Data Vault Stage		update_stage_order_detail
stage_product	stage_product_4ea...	Data Vault Stage		update_stage_product
stage_product_DV	stage_product_DV	Stage		update_stage_product_DV
stage_shipment	stage_shipmen_a3...	Data Vault Stage		update_stage_shipment

Results

Object	Message
stage_product_DV	SQL Statement completed.
stage_product_DV	Generated Procedure 'update_stage_product_DV' from template 'wsl_sqlserver_proc_dv_stage'
stage_product_DV	Procedure 'update_stage_product_DV' for Stage Table 'stage_product_DV' is rebuilt and recompiled.

4. Right-click the **Data Vault Stage** table in the left pane, under the **Stage Table** objects list and select **Code > View update** from the context menu to view the code generated for the update procedure generated.



```
#--==============================================================================
#-- DBMS Name        :    POSTGRESQL Custom*
#-- Block Name       :    update_stage_product_DV
#-- Template         :    wsl_postgres_pscript_dv_stage
#-- Template Version :    8.5.1.0
#-- Description      :    Update the Stage Table table stage_product_DV
#-- Generated by     :    WhereScape RED Version 10.0.0.0 (build 230406-170544)
#-- Generated for    :    Simon Spring Snowflake to 31/05/2023
#-- Generated on     :    Tuesday, May 16, 2023 at 20:44:00
#-- Author           :    Demo User
#--==============================================================================
#-- Notes / History
#--
Import-module -Name WslPowershellCommon -DisableNameChecking
Import-Module -Name WslPowershellPostgreSQL -DisableNameChecking
Hide-Window

#--============================================================================
#-- General Variables
#--============================================================================

$sequence = ${env:WSL_SEQUENCE}
$jobName = ${env:WSL_JOB_NAME}
$taskName = ${env:WSL_TASK_NAME}
$jobId = ${env:WSL_JOB_KEY}
$taskId = ${env:WSL_TASK_KEY}
$return_Msg = "stage_product_DV updated."
$status = 1

#--============================================================================
#-- Main
#--============================================================================

```

Ready   Length: 29   Lines: 91   Ln: 26, Col: 30   Sel: 0 | 0   Macintosh (CR)   INS   CAP   NUM   SCRL

5. Right-click the **Data Vault Stage** table in the left pane and select **Execute Update Procedure** from the context menu to run the procedure. The **Results** pane displays the number of records created.

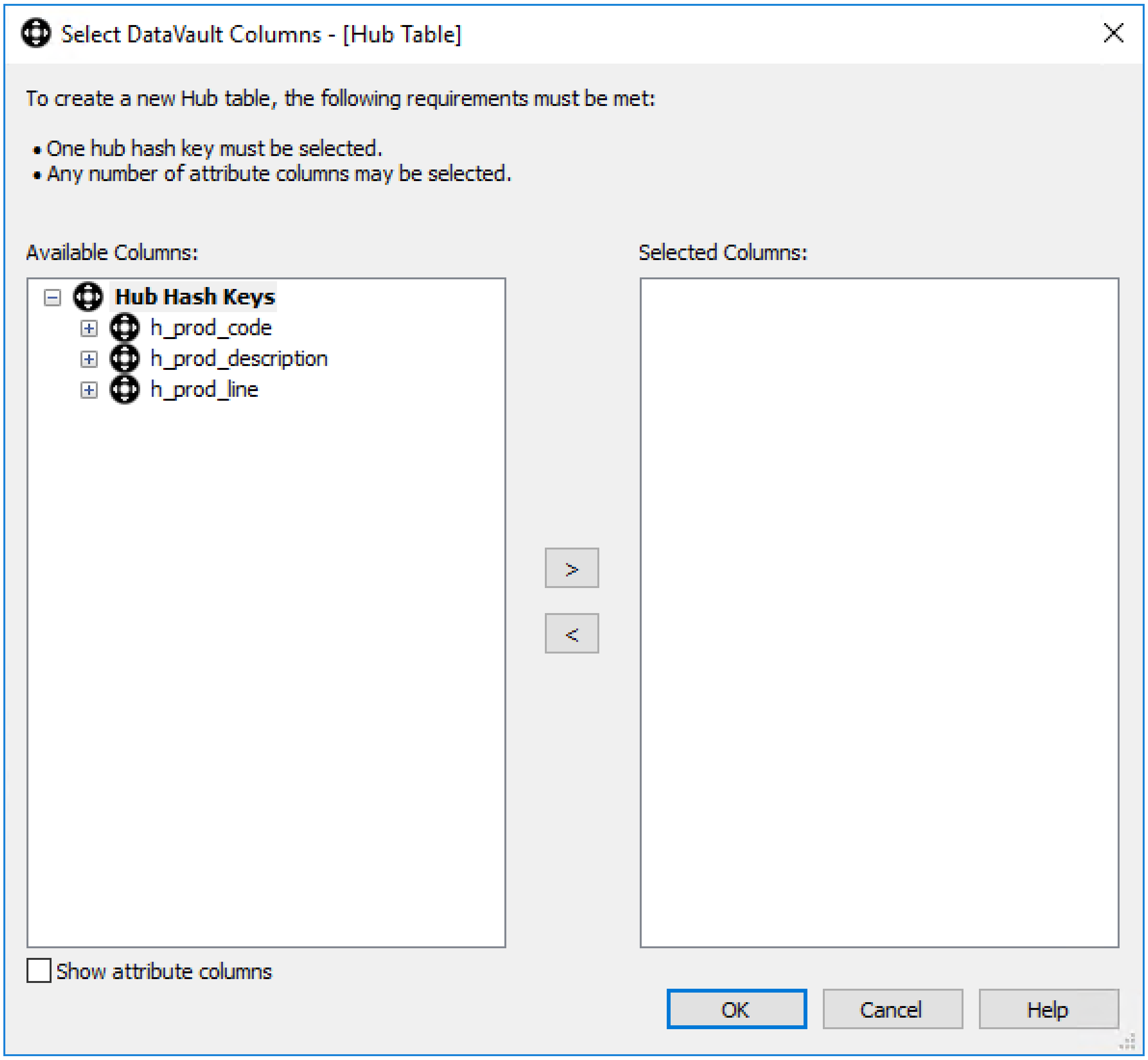
Creating the Hub, Link and Satellite Tables

After successfully creating and populating the **Data Vault Stage** table, you can now create the **Hub**, **Link** and **Satellite** tables. The **Hub**, **Link** and **Change** hash keys information stored in the **Data Vault Stage** table is used by the Wizard for building these **Data Vault** objects.

Creating the Hub table

The following describe the steps for creating a Hub table:

1. Browse to the Data Warehouse connection (click **Browse >Data Warehouse**) to create the Hub table.
2. Double-click the **Hub** object group in the left pane, the middle pane displays a list of existing Hub tables.
3. Click the source **Data Vault Stage** table from the right pane and drag it to the middle pane.
4. The Hub table creation Wizard appears and prompts you to select the Hash Keys to use from the **Available Columns** pane. The columns that comprise the selected Hash Key are displayed under the **Selected Columns** pane—these are the columns that will be populated by the Wizard in the new **Hub** table.



Select DataVault Columns - [Hub Table]

To create a new Hub table, the following requirements must be met:

- One hub hash key must be selected.
- Any number of attribute columns may be selected.

Available Columns:

- Hub Hash Keys
  - h_prod_code
  - h_prod_description
  - h_prod_line

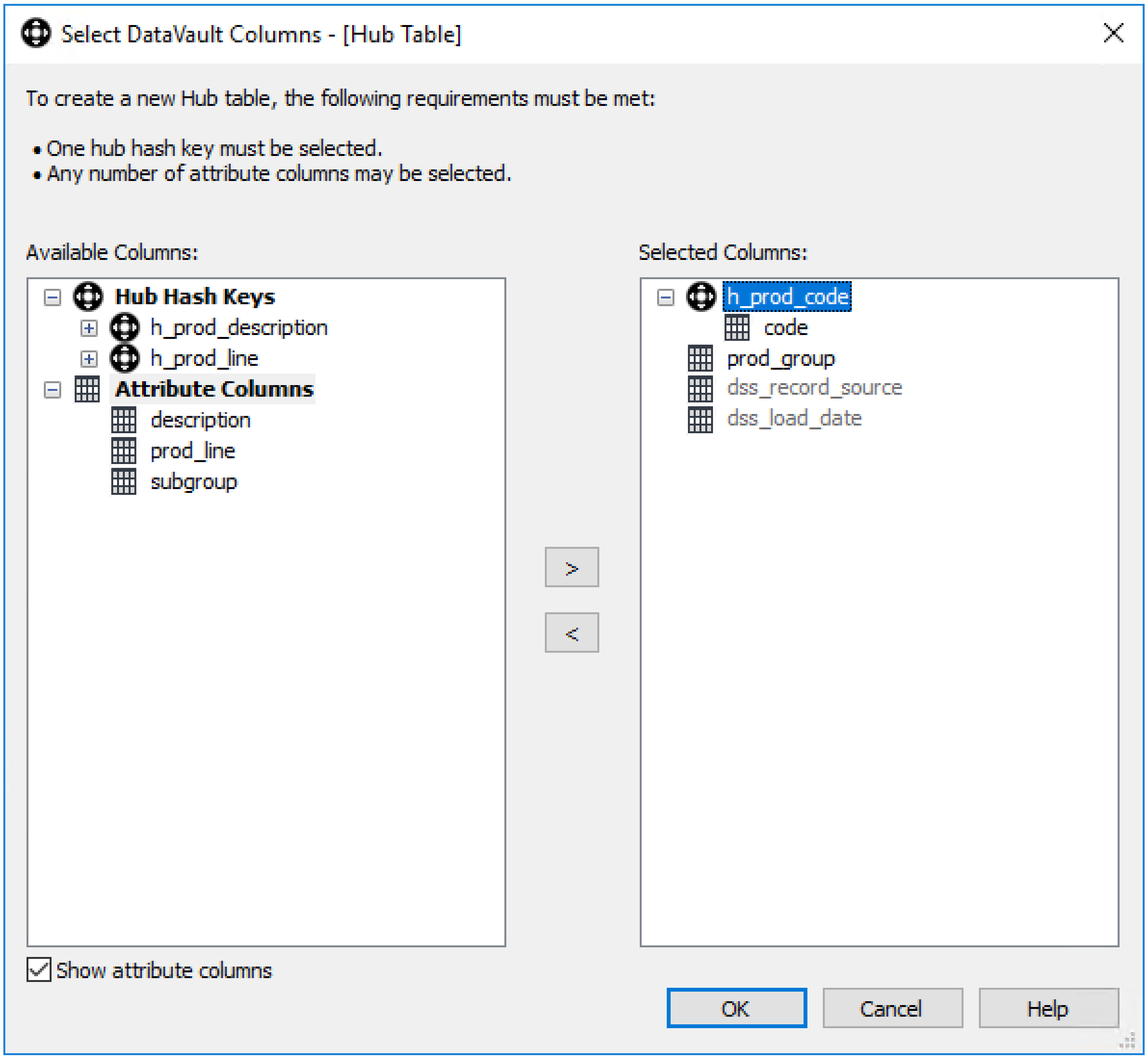
Selected Columns:

□ Show attribute columns

5. Select the Hash Keys you want to use from the **Available Columns** pane to see the columns that will be included in your Hub table under the **Selected Columns** pane. Click **OK** to continue.

Note:

In addition to the Hash Keys, you have the option to use attribute columns to create the Hub table. Select the **Show attribute columns** check box below the **Available Columns** pane to display and select the attribute columns that will be populated by the Wizard in the new Hub table.



Select DataVault Columns - [Hub Table]

To create a new Hub table, the following requirements must be met:

- One hub hash key must be selected.
- Any number of attribute columns may be selected.

Available Columns:

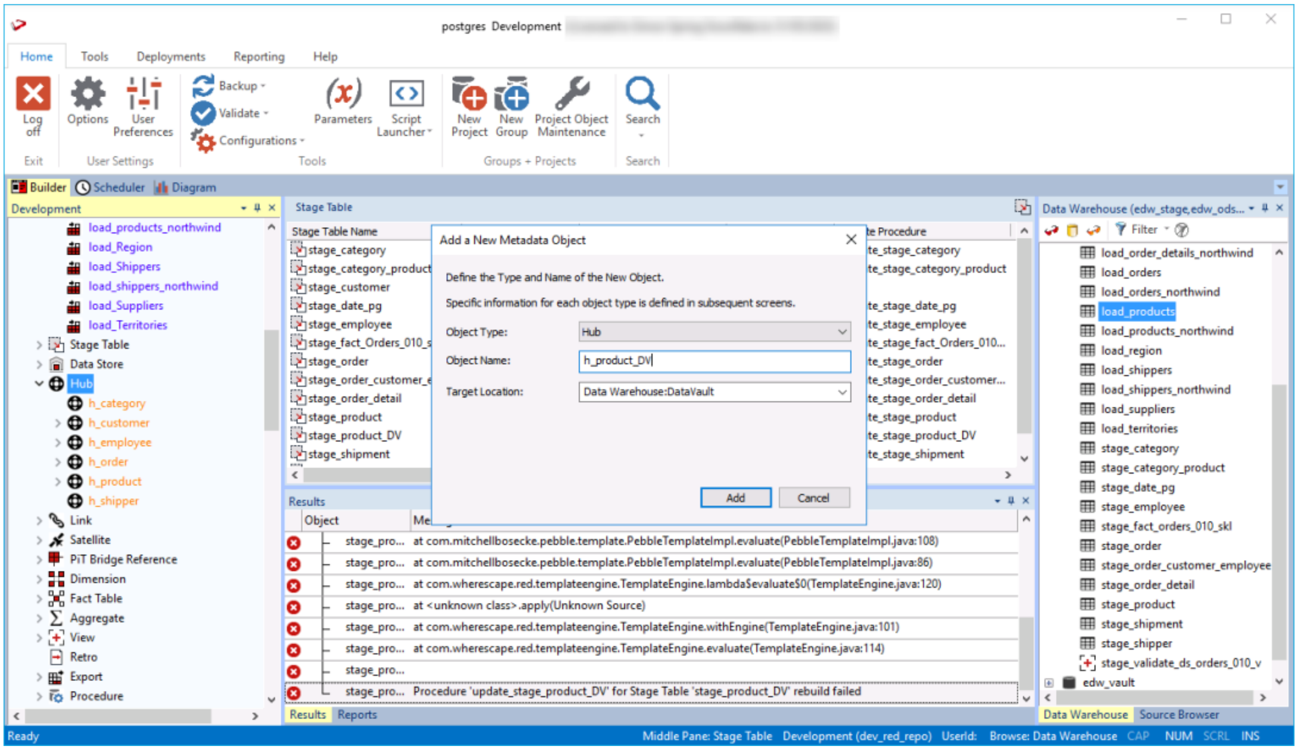
- Hub Hash Keys
  - h_prod_description
  - h_prod_line
- Attribute Columns
  - description
  - prod_line
  - subgroup

Selected Columns:

- h_prod_code
  - code
- prod_group
- dss_record_source
- dss_load_date

☑ Show attribute columns

6. The **Add a New Metadata Object** screen appears and classifies the new object as a Hub table. It provides a default name based on the source **Data Vault Stage** table name. Accept the name or enter a new name for the Hub table and click **ADD** to continue.



Add a New Metadata Object

Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: Hub

Object Name: h_product_DV

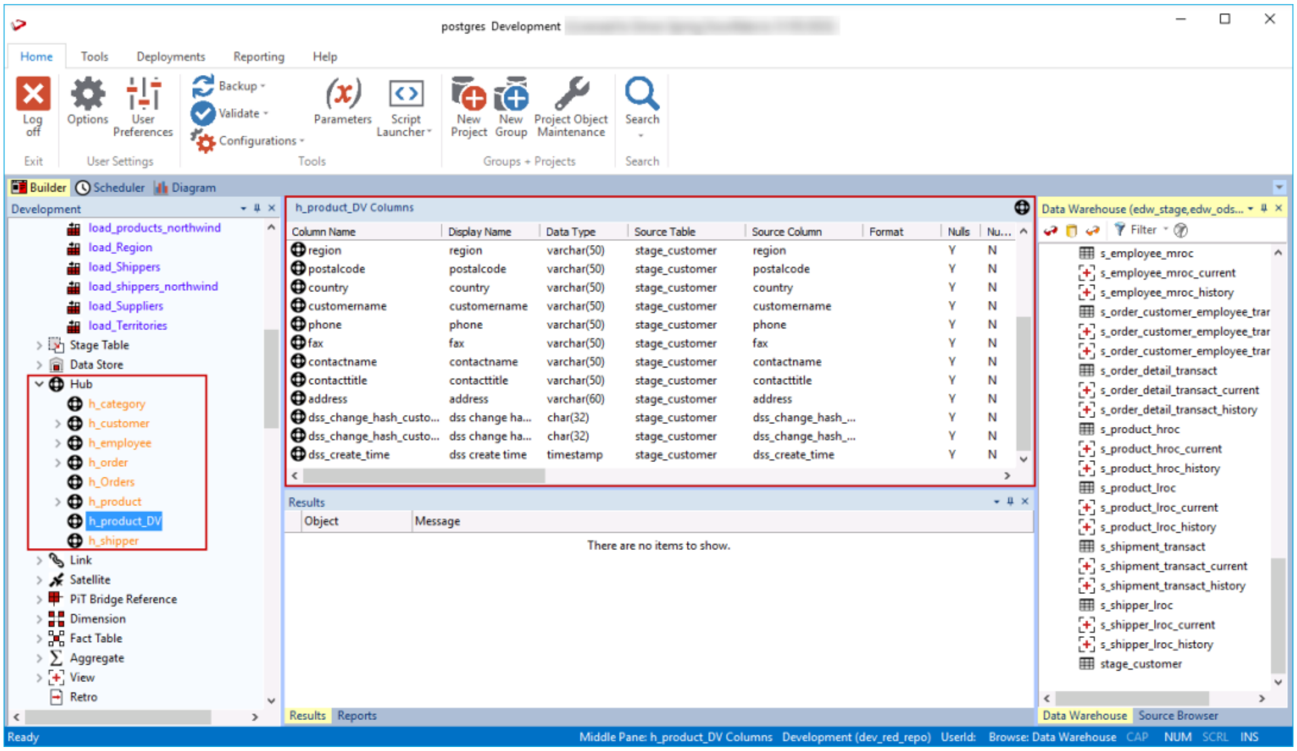
Target Location: Data Warehouse:DataVault

Add | Cancel

Results

```
Object          Message
stage_pro...    at com.mitchellbosecke.pebble.template.PebbleTemplateImpl.evaluate(PebbleTemplateImpl.java:108)
stage_pro...    at com.mitchellbosecke.pebble.template.PebbleTemplateImpl.evaluate(PebbleTemplateImpl.java:86)
stage_pro...    at com.wherescape.red.templateengine.TemplateEngine.lambda$evaluate$0(TemplateEngine.java:120)
stage_pro...    at <unknown class>.apply(Unknown Source)
stage_pro...    at com.wherescape.red.templateengine.TemplateEngine.withEngine(TemplateEngine.java:101)
stage_pro...    at com.wherescape.red.templateengine.TemplateEngine.evaluate(TemplateEngine.java:114)
stage_pro...
stage_pro...    Procedure 'update_stage_product_DV' for Stage Table 'stage_product_DV' rebuild failed
```

7. Click **OK** in the **Table Properties** screen to finish defining the meta data for the Hub table. The new Hub table is added to the **Hub** objects list in the left pane and the columns included in the table are listed in the middle pane.



h_product_DV Columns

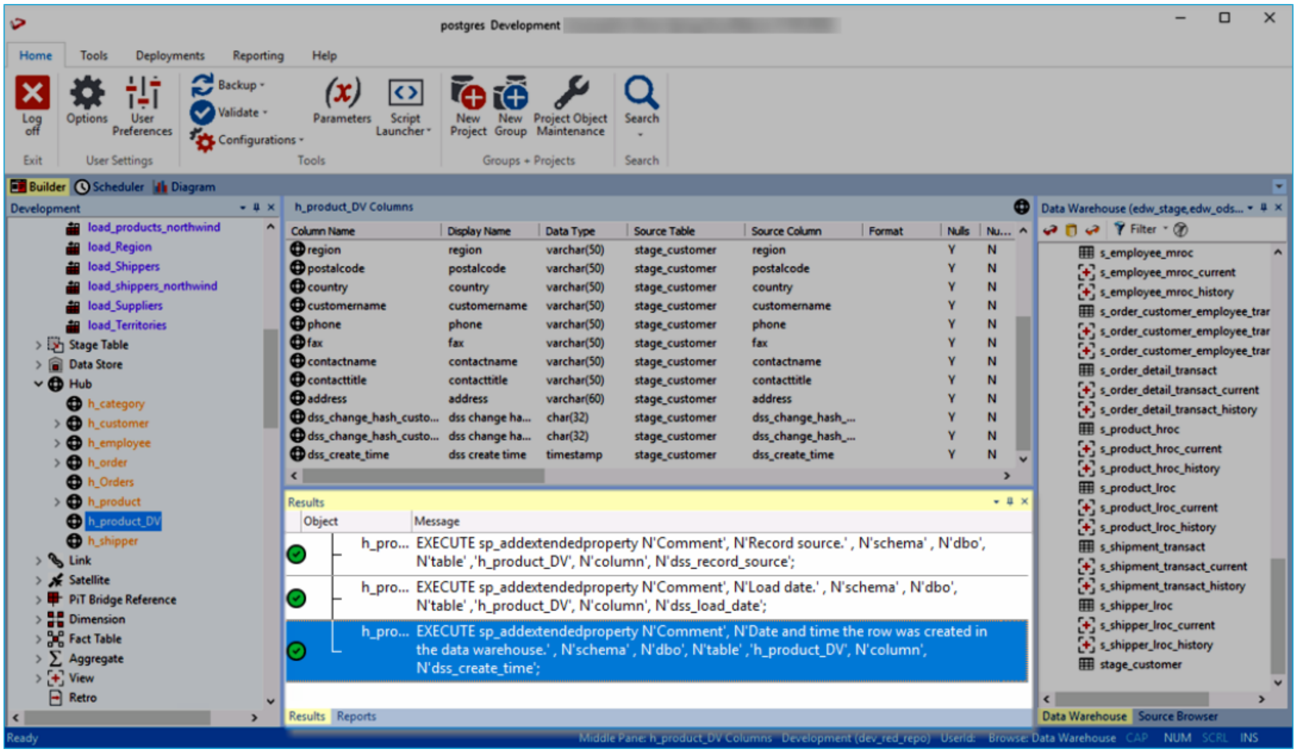
Column Name	Display Name	Data Type	Source Table	Source Column	Format	Nulls	Nu...
region	region	varchar(50)	stage_customer	region		Y	N
postalcode	postalcode	varchar(50)	stage_customer	postalcode		Y	N
country	country	varchar(50)	stage_customer	country		Y	N
customername	customername	varchar(50)	stage_customer	customername		Y	N
phone	phone	varchar(50)	stage_customer	phone		Y	N
fax	fax	varchar(50)	stage_customer	fax		Y	N
contactname	contactname	varchar(50)	stage_customer	contactname		Y	N
contacttitle	contacttitle	varchar(50)	stage_customer	contacttitle		Y	N
address	address	varchar(60)	stage_customer	address		Y	N
dss_change_hash_custo...	dss change ha...	char(32)	stage_customer	dss_change_hash_...		Y	N
dss_change_hash_custo...	dss change ha...	char(32)	stage_customer	dss_change_hash_...		Y	N
dss_create_time	dss create time	timestamp	stage_customer	dss_create_time		Y	N

Results

Object | Message

There are no items to show.

8. Right-click the new Hub table you defined in the left pane and select **Create (ReCreate)** from the context menu to create the table. The **Results** pane displays confirmation that the Hub table was successfully created.



h_product_DV Columns

Column Name	Display Name	Data Type	Source Table	Source Column	Format	Nulls	Nu...
region	region	varchar(50)	stage_customer	region		Y	N
postalcode	postalcode	varchar(50)	stage_customer	postalcode		Y	N
country	country	varchar(50)	stage_customer	country		Y	N
customername	customername	varchar(50)	stage_customer	customername		Y	N
phone	phone	varchar(50)	stage_customer	phone		Y	N
fax	fax	varchar(50)	stage_customer	fax		Y	N
contactname	contactname	varchar(50)	stage_customer	contactname		Y	N
contacttitle	contacttitle	varchar(50)	stage_customer	contacttitle		Y	N
address	address	varchar(60)	stage_customer	address		Y	N
dss_change_hash_custo...	dss change ha...	char(32)	stage_customer	dss_change_hash_...		Y	N
dss_change_hash_custo...	dss change ha...	char(32)	stage_customer	dss_change_hash_...		Y	N
dss_create_time	dss create time	timestamp	stage_customer	dss_create_time		Y	N

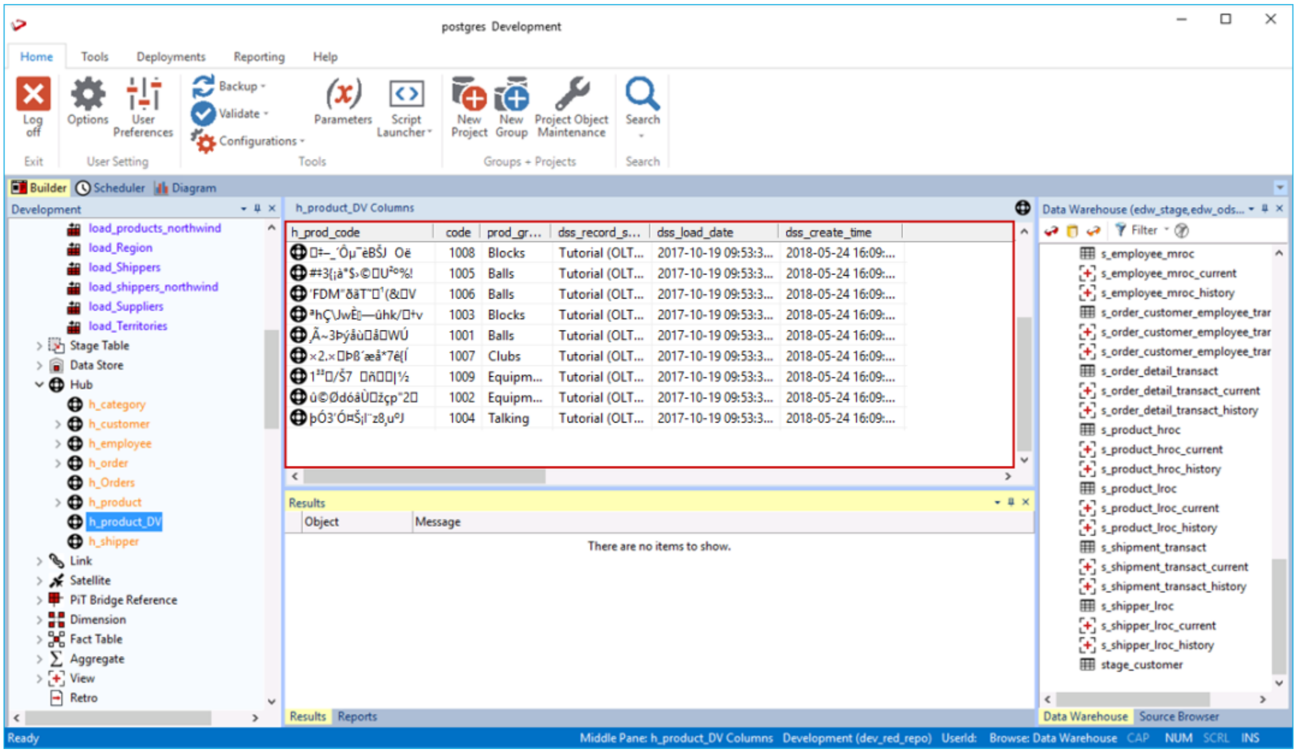
Results

Object	Message
h_pro...	EXECUTE sp_addextendedproperty N'Comment', N'Record source.' , N'schema' , N'dbo', N'table' ,'h_product_DV', N'column', N'dss_record_source';
h_pro...	EXECUTE sp_addextendedproperty N'Comment', N'Load date.' , N'schema' , N'dbo', N'table' ,'h_product_DV', N'column', N'dss_load_date';
h_pro...	EXECUTE sp_addextendedproperty N'Comment', N'Date and time the row was created in the data warehouse.' , N'schema' , N'dbo', N'table' ,'h_product_DV', N'column', N'dss_create_time';

After the new Hub table is defined and created, clicking the **Rebuild** button in the table **Properties** screen launches the Wizard for generating the update procedure to populate the table. This Wizard utilizes a template to create the procedure.

The detailed steps for using this Wizard is described in **Generating Update Procedures for the Hub, Link and Satellite Tables**.

Once you have run the generated update procedures, you can view the generated Hub Hash keys, by right-clicking the new **Hub** table you created in the left pane and then selecting **Display Data** from the context menu:



h_product_DV Columns

h_prod_code	code	prod_gr...	dss_record_s...	dss_load_date	dss_create_time
□‡–_´Ôµ¯èBŠJ Oë	1008	Blocks	Tutorial (OLT...	2017-10-19 09:53:3...	2018-05-24 16:09:...
#‡3{¡à*$›©□U²º%!	1005	Balls	Tutorial (OLT...	2017-10-19 09:53:3...	2018-05-24 16:09:...
'FDM"ðäT¨□¹(&□V	1006	Balls	Tutorial (OLT...	2017-10-19 09:53:3...	2018-05-24 16:09:...
ªhÇ\JwÈ□—ûhk/□+v	1003	Blocks	Tutorial (OLT...	2017-10-19 09:53:3...	2018-05-24 16:09:...
‚Ã~3þýåù□å□WÚ	1001	Balls	Tutorial (OLT...	2017-10-19 09:53:3...	2018-05-24 16:09:...
×2.×□Þß´æå*7ê{Í	1007	Clubs	Tutorial (OLT...	2017-10-19 09:53:3...	2018-05-24 16:09:...
1³³□/Š7 □ñ□□|½	1009	Equipm...	Tutorial (OLT...	2017-10-19 09:53:3...	2018-05-24 16:09:...
û©ØdóâÙ□žçp"2□	1002	Equipm...	Tutorial (OLT...	2017-10-19 09:53:3...	2018-05-24 16:09:...
þÓ3'Ó¤Š¡l¨z8‚uºJ	1004	Talking	Tutorial (OLT...	2017-10-19 09:53:3...	2018-05-24 16:09:...

Results

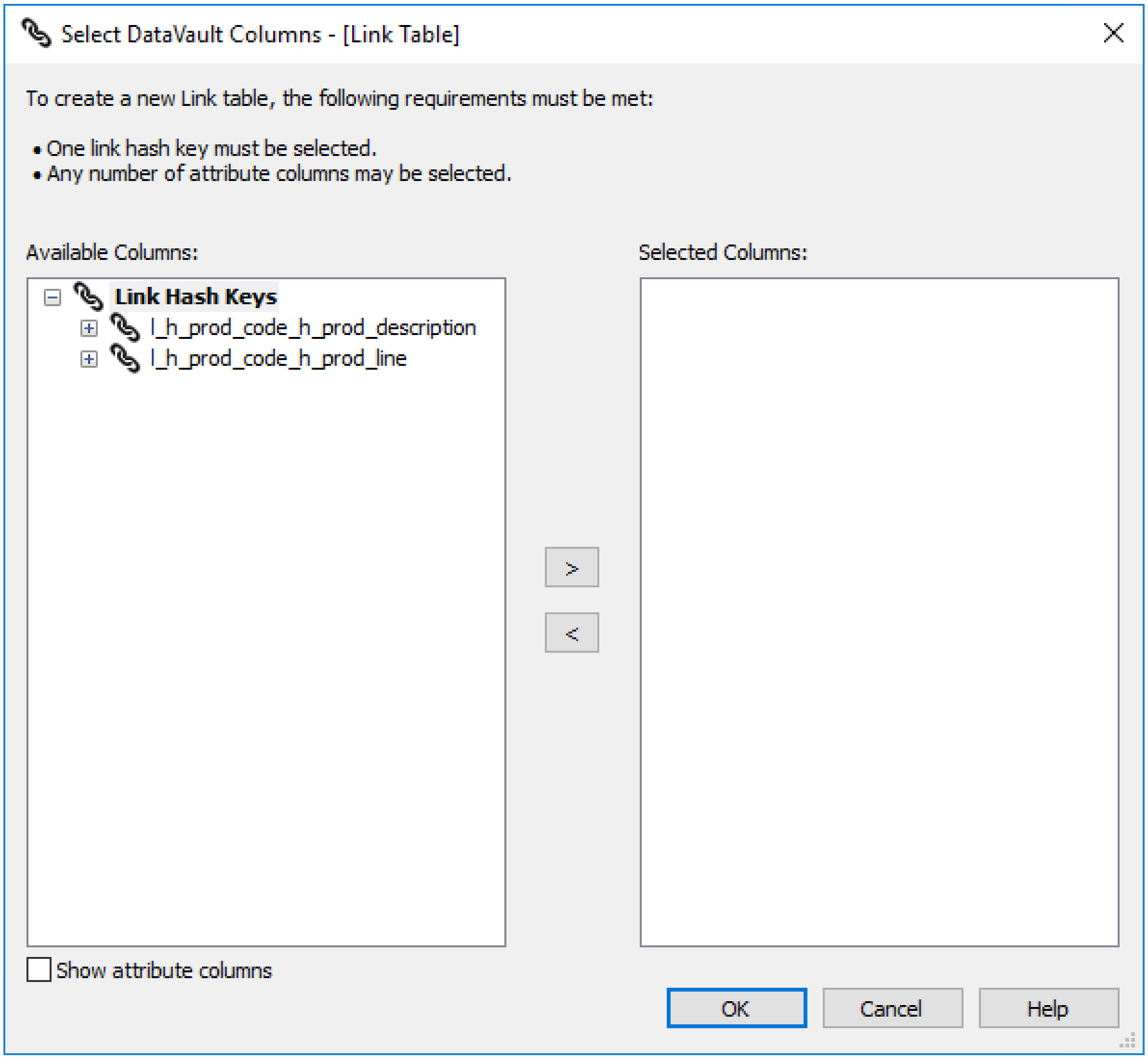
Object	Message

There are no items to show.

Creating the Link table

The steps for creating the Link table is similar to the steps used in creating the Hub table:

1. Browse to the Data Warehouse connection to create the Link table.
2. Double-click the **Link** object group in the left pane, the middle pane displays a list of existing Link tables.
3. Click the source **Data Vault Stage** table from the right pane and drag it to the middle pane.
4. The Link table creation Wizard appears and prompts you to select the Hash Key to use from the **Available Columns** pane. The columns that comprise the selected Hash Key are displayed under the **Selected Columns** pane—these are the columns that will be populated by the Wizard in the new **Link** table.



Select DataVault Columns - [Link Table]

To create a new Link table, the following requirements must be met:

- One link hash key must be selected.
- Any number of attribute columns may be selected.

Available Columns:

- Link Hash Keys
  - l_h_prod_code_h_prod_description
  - l_h_prod_code_h_prod_line

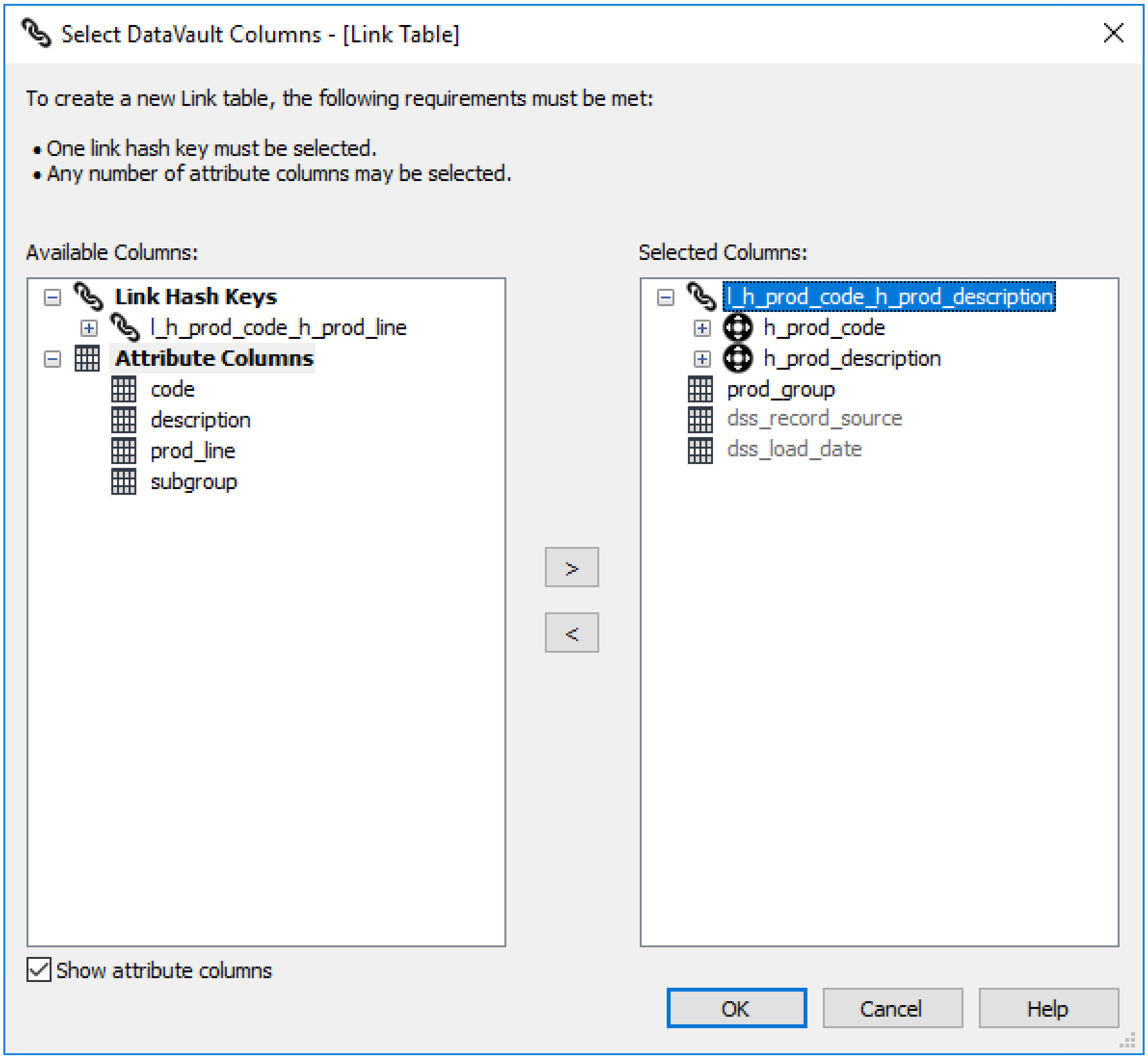
Selected Columns:

□ Show attribute columns

5. Select the Hash Key you want to use from the **Available Columns** pane to see the columns that will be included in your Link table under the **Selected Columns** pane. Click **OK** to continue.

Notes

In addition to the Hash Keys, you have the option to use attribute columns to create the Link table. Select the **Show attribute columns** check box below the **Available Columns** pane to display and select the attribute columns that will be populated by the Wizard in the new Link table.



Select DataVault Columns - [Link Table]

To create a new Link table, the following requirements must be met:

- One link hash key must be selected.
- Any number of attribute columns may be selected.

Available Columns:

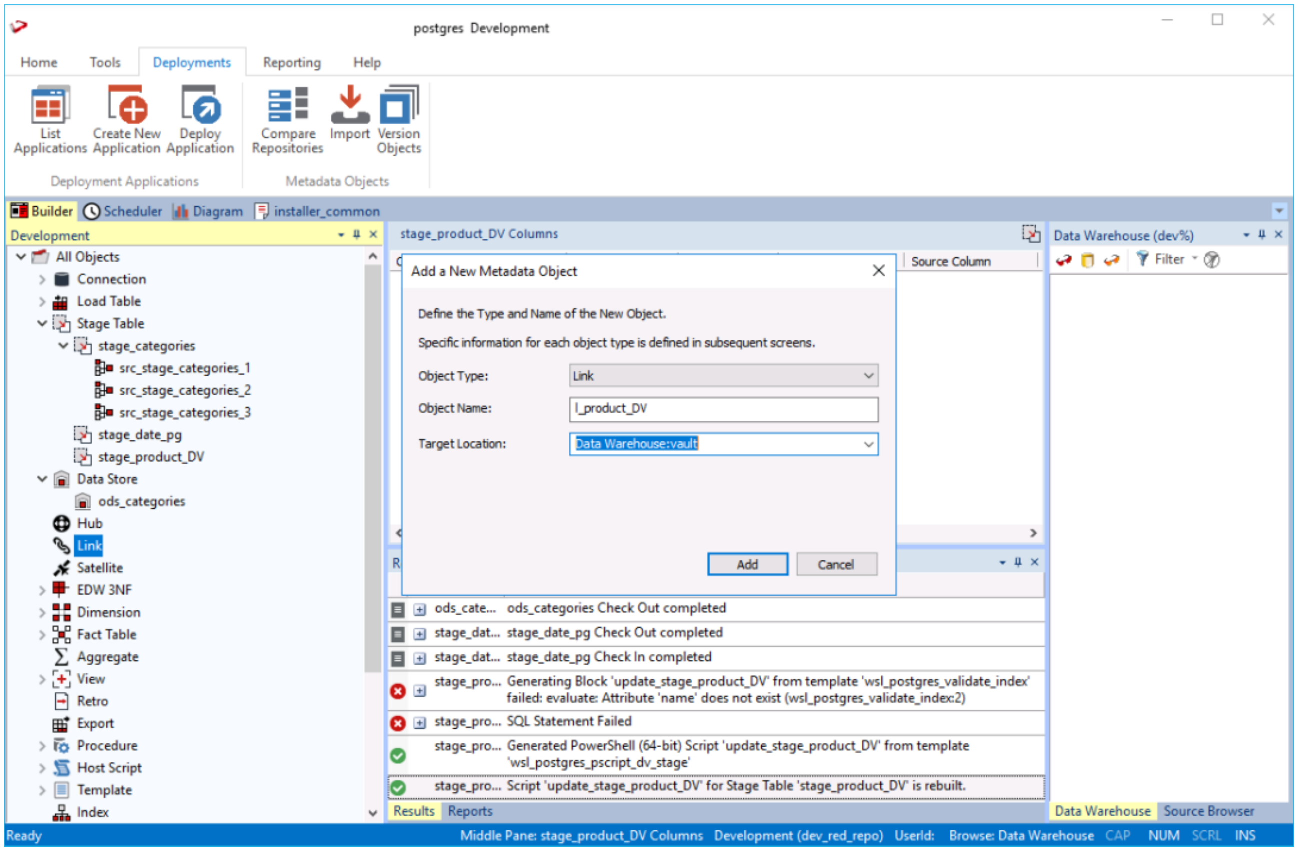
- Link Hash Keys
  - l_h_prod_code_h_prod_line
- Attribute Columns
  - code
  - description
  - prod_line
  - subgroup

Selected Columns:

- l_h_prod_code_h_prod_description
  - h_prod_code
  - h_prod_description
  - prod_group
  - dss_record_source
  - dss_load_date

☑ Show attribute columns

6. The **Add a New Metadata Object** screen appears and classifies the new object as a Link table. It provides a default name based on the source **Data Vault Stage** table name. Accept the name or enter a new name for the Link table and click **ADD** to continue.



Add a New Metadata Object

Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: Link

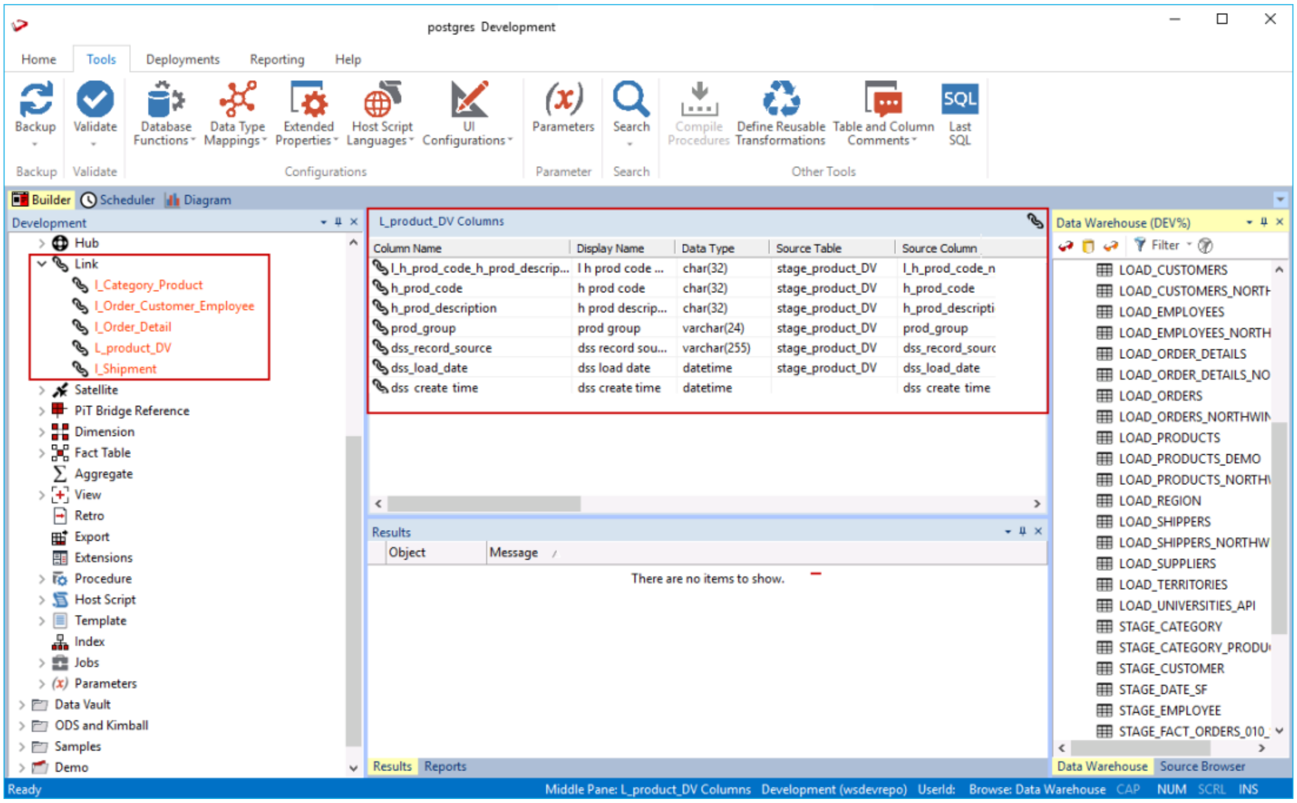
Object Name: l_product_DV

Target Location: Data Warehouse:vault

Add | Cancel

```
ods_cate...   ods_categories Check Out completed
stage_dat...  stage_date_pg Check Out completed
stage_dat...  stage_date_pg Check In completed
stage_pro...  Generating Block 'update_stage_product_DV' from template 'wsl_postgres_validate_index'
              failed: evaluate: Attribute 'name' does not exist (wsl_postgres_validate_index:2)
stage_pro...  SQL Statement Failed
stage_pro...  Generated PowerShell (64-bit) Script 'update_stage_product_DV' from template
              'wsl_postgres_pscript_dv_stage'
stage_pro...  Script 'update_stage_product_DV' for Stage Table 'stage_product_DV' is rebuilt.
```

7. Click **OK** in the table **Properties** screen to finish defining the meta data for the Link table. The new Link table is added to the **Link** objects list in the left pane and the columns included in the table are listed in the middle pane.



L_product_DV Columns

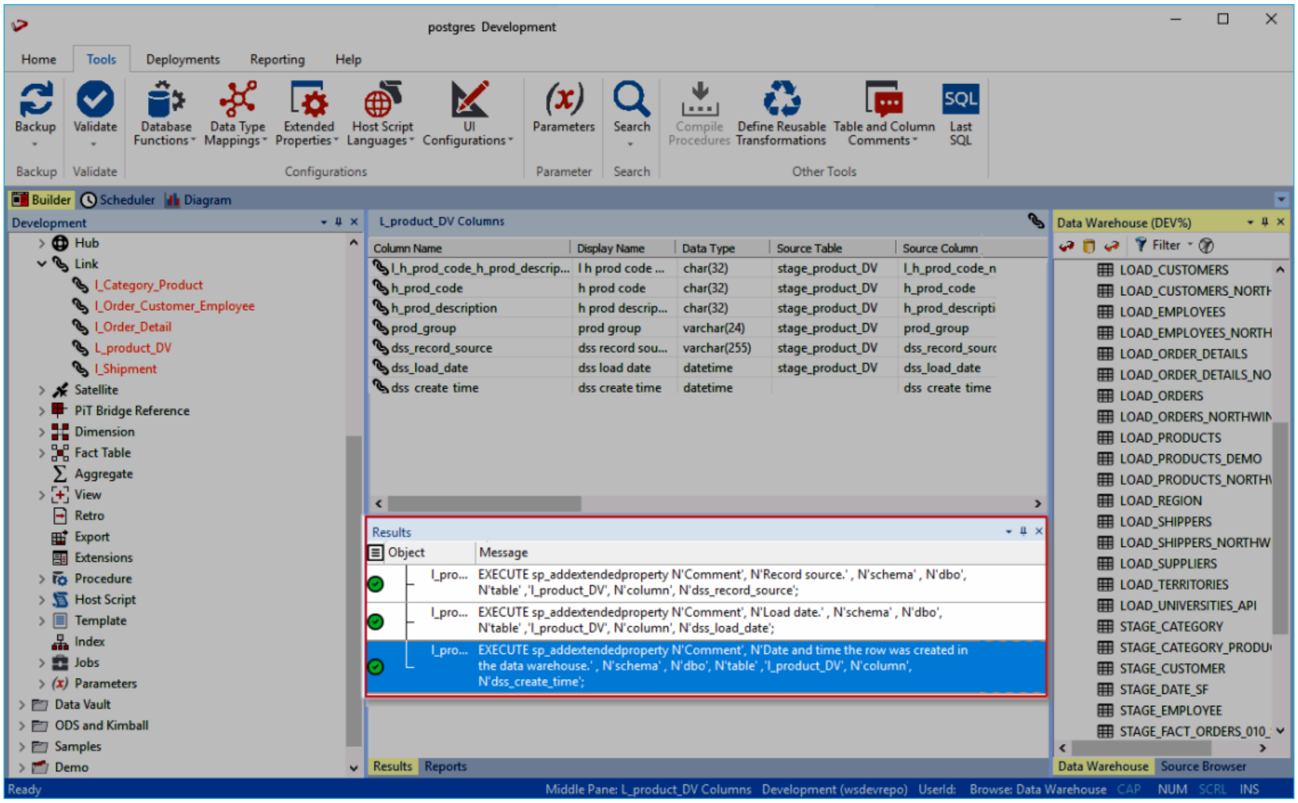
Column Name	Display Name	Data Type	Source Table	Source Column
l_h_prod_code_h_prod_descrip...	l h prod code ...	char(32)	stage_product_DV	l_h_prod_code_n
h_prod_code	h prod code	char(32)	stage_product_DV	h_prod_code
h_prod_description	h prod descrip...	char(32)	stage_product_DV	h_prod_descripti
prod_group	prod group	varchar(24)	stage_product_DV	prod_group
dss_record_source	dss record sou...	varchar(255)	stage_product_DV	dss_record_sourc
dss_load_date	dss load date	datetime	stage_product_DV	dss_load_date
dss_create_time	dss create time	datetime		dss_create_time

Results

Object	Message

There are no items to show.

8. Right-click the new Link table you defined in the left pane and select **Create (ReCreate)** from the context menu to create the table. The **Results** pane displays confirmation that the Link table was successfully created.



L_product_DV Columns

Column Name	Display Name	Data Type	Source Table	Source Column
l_h_prod_code_h_prod_descrip...	l h prod code ...	char(32)	stage_product_DV	l_h_prod_code_n
h_prod_code	h prod code	char(32)	stage_product_DV	h_prod_code
h_prod_description	h prod descrip...	char(32)	stage_product_DV	h_prod_descripti
prod_group	prod group	varchar(24)	stage_product_DV	prod_group
dss_record_source	dss record sou...	varchar(255)	stage_product_DV	dss_record_sourc
dss_load_date	dss load date	datetime	stage_product_DV	dss_load_date
dss_create_time	dss create time	datetime		dss_create_time

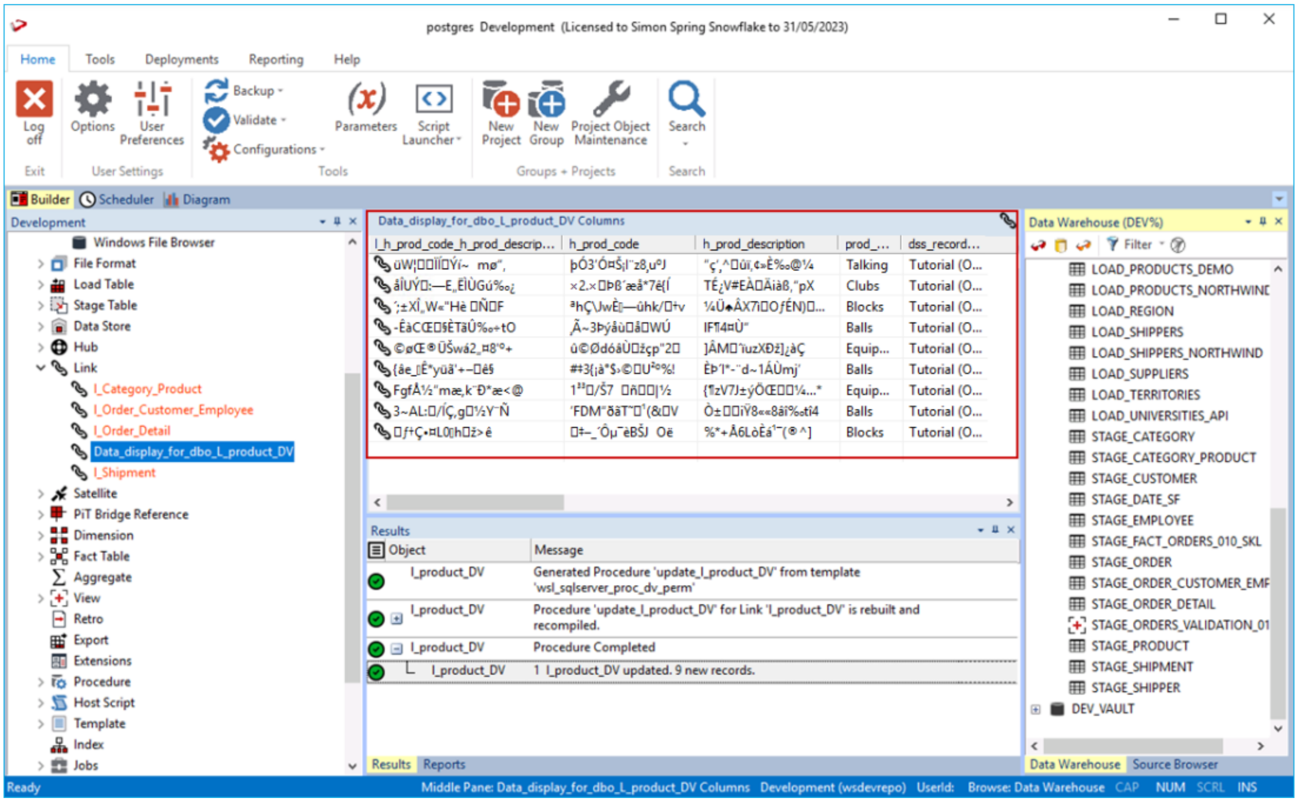
Results

Object	Message
l_pro...	EXECUTE sp_addextendedproperty N'Comment', N'Record source.' , N'schema' , N'dbo', N'table' ,'l_product_DV', N'column', N'dss_record_source';
l_pro...	EXECUTE sp_addextendedproperty N'Comment', N'Load date.' , N'schema' , N'dbo', N'table' ,'l_product_DV', N'column', N'dss_load_date';
l_pro...	EXECUTE sp_addextendedproperty N'Comment', N'Date and time the row was created in the data warehouse.' , N'schema' , N'dbo', N'table' ,'l_product_DV', N'column', N'dss_create_time';

After the new Link table is defined and created, clicking the **Rebuild** button in the table **Properties** screen launches the Wizard for generating the update procedure to populate the table. This Wizard utilizes a template to create the procedure.

The detailed steps for using this Wizard is described in Generating Update Procedures for the Hub, Link and Satellite Tables.

Once you have run the generated update procedures, you can view the generated Link Hash keys, by right-clicking the new Link table you created in the left pane and then selecting **Display Data** from the context menu:



Data_display_for_dbo_L_product_DV Columns

l_h_prod_code_h_prod_descrip...	h_prod_code	h_prod_description	prod_...	dss_record...
üW¦□□ĬÎ□Ýí~ mø",	þÓ3'Ó¤Š¡l¨z8,u°J	"ç',^□ûï,¢»È‰@¼	Talking	Tutorial (O...
åÎUÝ□:—E„ËÌÙGú‰¿	×2.×□Þß´æå*7ê{Í	TÉ¿V#EÀ□Äiàß,"pX	Clubs	Tutorial (O...
´;±XÎ„W«"Hè □Ñ□F	ªhÇ\JwÈ▯—ûhk/□+v	¼Ü♠ÂX7i□OƒÉN)□...	Blocks	Tutorial (O...
-ÊàCŒ□§ÈTãÛ‰+tO	‚Ã~3Þýåù□å□WÚ	IF¶4¤Ù"	Balls	Tutorial (O...
©øŒ®ŪŠwá2„¤8'º+	û©ØdóåÙ□žçp"2□	]ÂM□ˆïuzXÐž]¿àÇ	Equip...	Tutorial (O...
{âe_▯Ê*yüã'+–□ê§	#‡3{¡à*$›©□U²º%!	ÈÞ'l*-¨d~1ÁÙmj'	Balls	Tutorial (O...
FgfÅ½"mæ,k¨Ð*æ<@	1³³□/Š7 □ñ□□|½	{¶zV7J±ýÖŒ□□¼…*	Equip...	Tutorial (O...
3~AL:□/ÍÇ,g□½Y¯Ñ	'FDM"ðãT˜□¹(&□V	Ò±□□iŸ8««8äï‰tï4	Balls	Tutorial (O...
□ƒ†Ç•¤L0▯h□ž>ê	□‡–_´Ôµ¯èBŠJ Oë	%*+Å6LòÈá¹¯(®^]	Blocks	Tutorial (O...

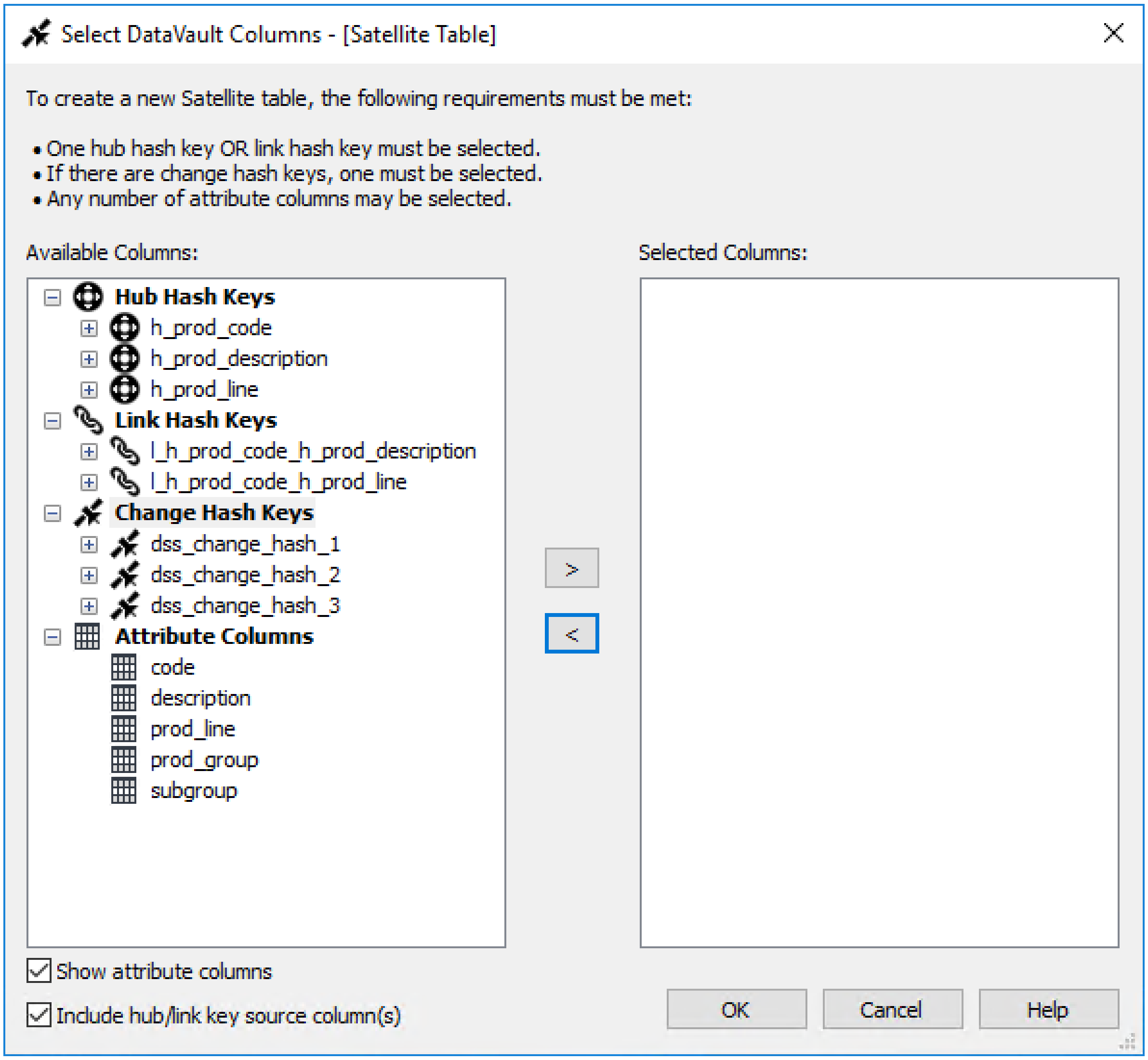
Results

Object	Message
l_product_DV	Generated Procedure 'update_l_product_DV' from template 'wsl_sqlserver_proc_dv_perm'
l_product_DV	Procedure 'update_l_product_DV' for Link 'l_product_DV' is rebuilt and recompiled.
l_product_DV	Procedure Completed
l_product_DV	1 l_product_DV updated. 9 new records.

Creating the Satellite table

The following describe the steps for creating a Satellite table:

1. Browse to the Data Warehouse connection to create the Satellite tables.
2. Double-click the **Satellite** object group in the left pane, the middle pane displays a list of existing Satellite tables.
3. Click the source **Data Vault Stage** table from the right pane and drag it to the middle pane.
4. The Satellite table creation Wizard appears and prompts you to select the Hash Key to use from the **Available Hash Keys** pane. The columns that comprise the selected Hash Key are displayed under the **Selected Columns** pane—these are the columns that will be populated by the Wizard in the new Satellite table.



Select DataVault Columns - [Satellite Table]

To create a new Satellite table, the following requirements must be met:

- One hub hash key OR link hash key must be selected.
- If there are change hash keys, one must be selected.
- Any number of attribute columns may be selected.

Available Columns:

- Hub Hash Keys
  - h_prod_code
  - h_prod_description
  - h_prod_line
- Link Hash Keys
  - l_h_prod_code_h_prod_description
  - l_h_prod_code_h_prod_line
- Change Hash Keys
  - dss_change_hash_1
  - dss_change_hash_2
  - dss_change_hash_3
- Attribute Columns
  - code
  - description
  - prod_line
  - prod_group
  - subgroup

Selected Columns:

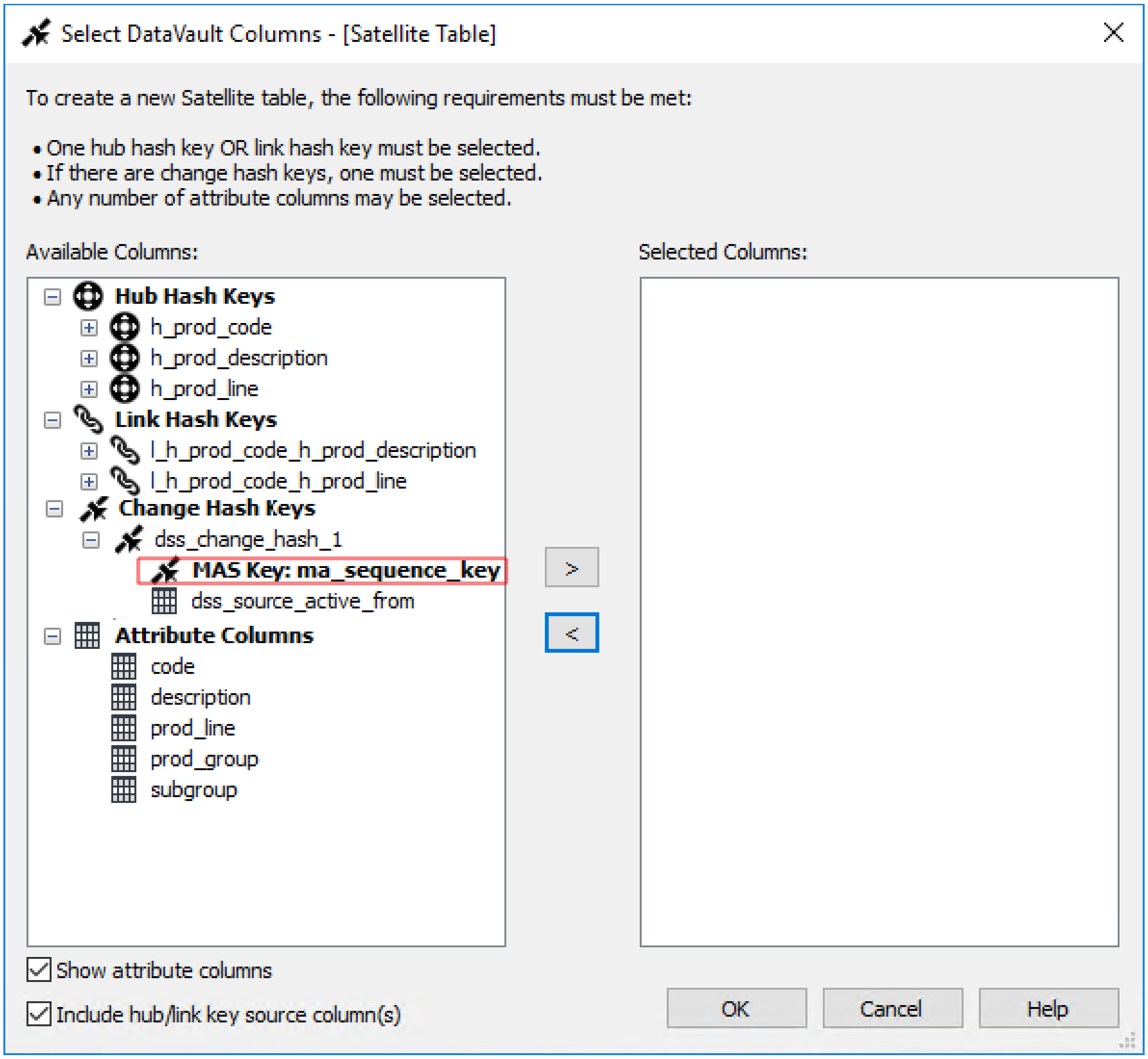
☑ Show attribute columns

☑ Include hub/link key source column(s)

Notes

The **'MAS Key: ma_sequence_key'**or **'MAS Key: natural_key_name'** under the Available Column **Change Hash Key** shows that a Multi-Active Sequence or Natural Key is linked to a change hash key.

Notes



Select DataVault Columns - [Satellite Table]

To create a new Satellite table, the following requirements must be met:

- One hub hash key OR link hash key must be selected.
- If there are change hash keys, one must be selected.
- Any number of attribute columns may be selected.

Available Columns:

- Hub Hash Keys
  - h_prod_code
  - h_prod_description
  - h_prod_line
- Link Hash Keys
  - l_h_prod_code_h_prod_description
  - l_h_prod_code_h_prod_line
- Change Hash Keys
  - dss_change_hash_1
    - MAS Key: ma_sequence_key
    - dss_source_active_from
- Attribute Columns
  - code
  - description
  - prod_line
  - prod_group
  - subgroup

Selected Columns:

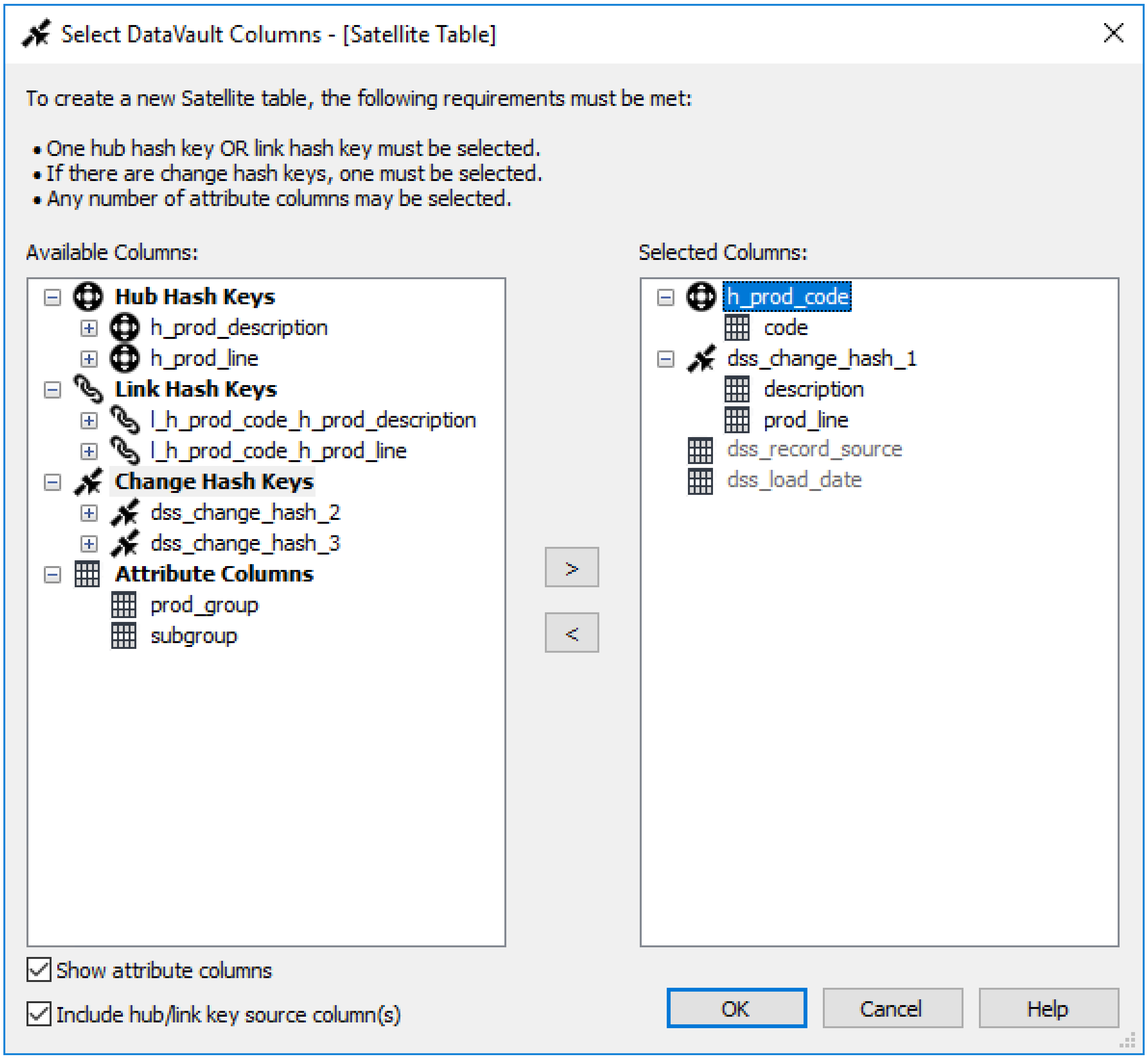
☑ Show attribute columns

☑ Include hub/link key source column(s)

5. Select the Hash Key you want to use from the **Available Hash Keys** pane to see the columns that will be included in your Satellite table under the **Selected Columns** pane. Click **OK** to continue.

Notes

1. In addition to the Hash Keys, you have the option to use attribute columns to create the Satellite table. Select the **Show attribute columns** check box below the **Available Columns** pane to display and select the attribute columns that will be populated by the Wizard in the new Satellite table.
2. Selecting the **Include hub/link key source columns** check box will include the hub hash key source column when the Satellite table is created.



Select DataVault Columns - [Satellite Table]

To create a new Satellite table, the following requirements must be met:

- One hub hash key OR link hash key must be selected.
- If there are change hash keys, one must be selected.
- Any number of attribute columns may be selected.

Available Columns:

- Hub Hash Keys
  - h_prod_description
  - h_prod_line
- Link Hash Keys
  - l_h_prod_code_h_prod_description
  - l_h_prod_code_h_prod_line
- Change Hash Keys
  - dss_change_hash_2
  - dss_change_hash_3
- Attribute Columns
  - prod_group
  - subgroup

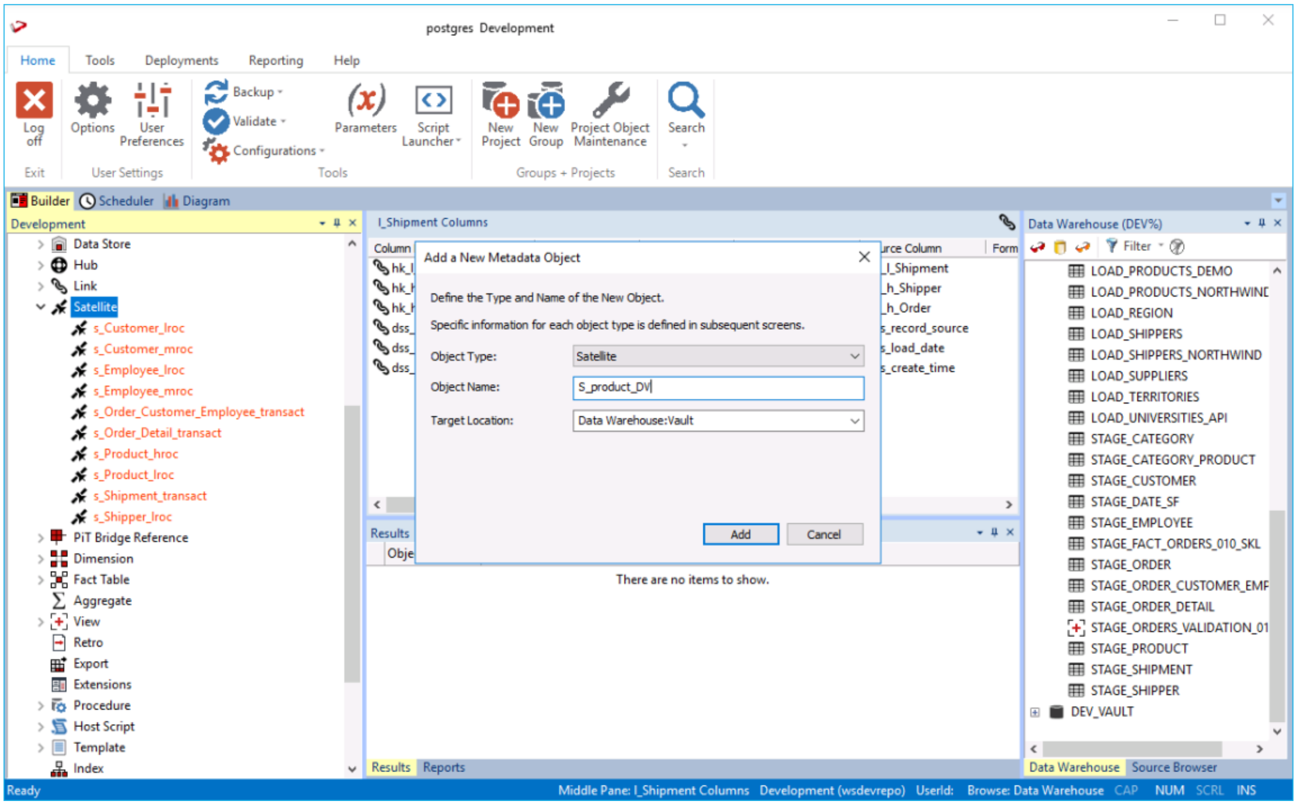
Selected Columns:

- h_prod_code
  - code
- dss_change_hash_1
  - description
  - prod_line
- dss_record_source
- dss_load_date

☑ Show attribute columns

☑ Include hub/link key source column(s)

6. The **Add a New Metadata Object** screen appears and classifies the new object as a Satellite table. It provides a default name based on the source **Data Vault Stage** table name. Accept the name or enter a new name for the Satellite table and click **ADD** to continue.



Add a New Metadata Object

Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: Satellite

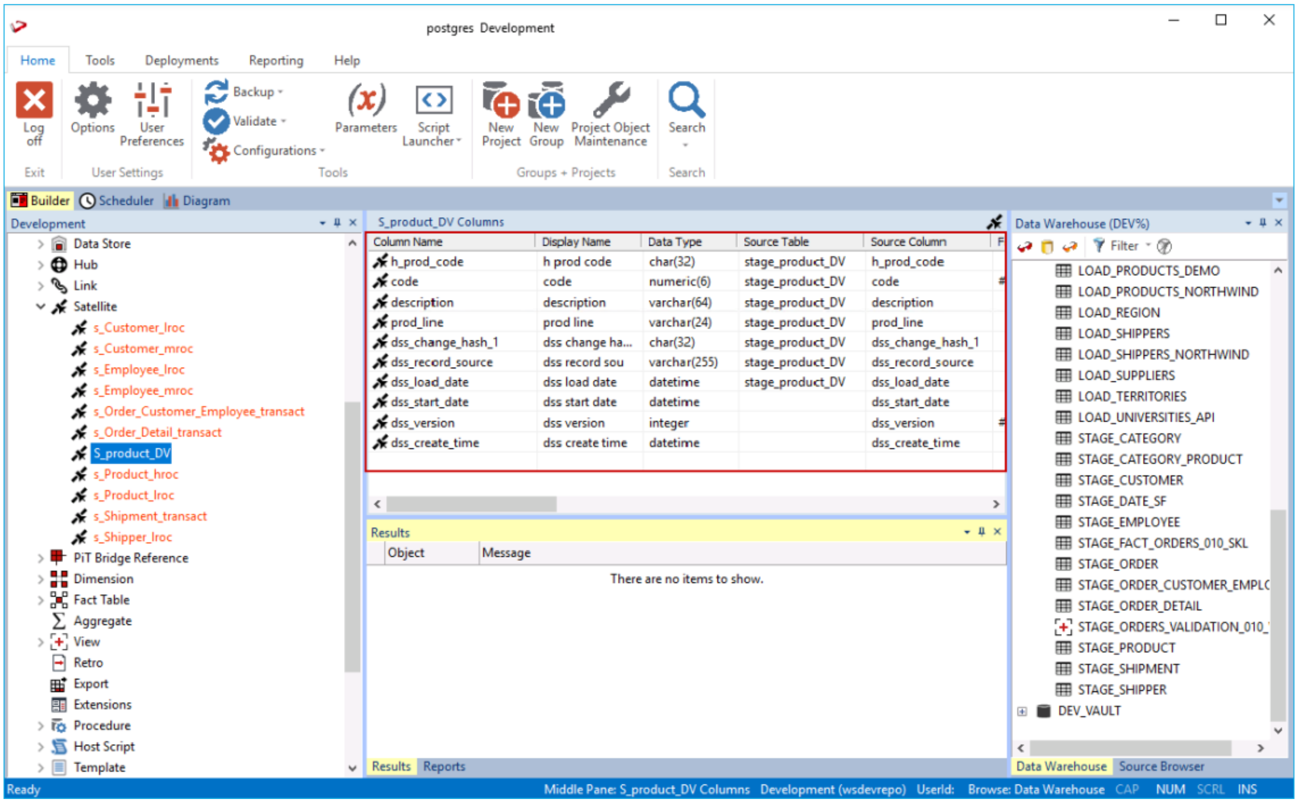
Object Name: S_product_DV

Target Location: Data Warehouse:Vault

Add | Cancel

There are no items to show.

7. Click **OK** in the table **Properties** screen to finish defining the meta data for the Satellite table. The new Satellite table is added to the **Satellite** objects list in the left pane and the columns included in the table are listed in the middle pane.

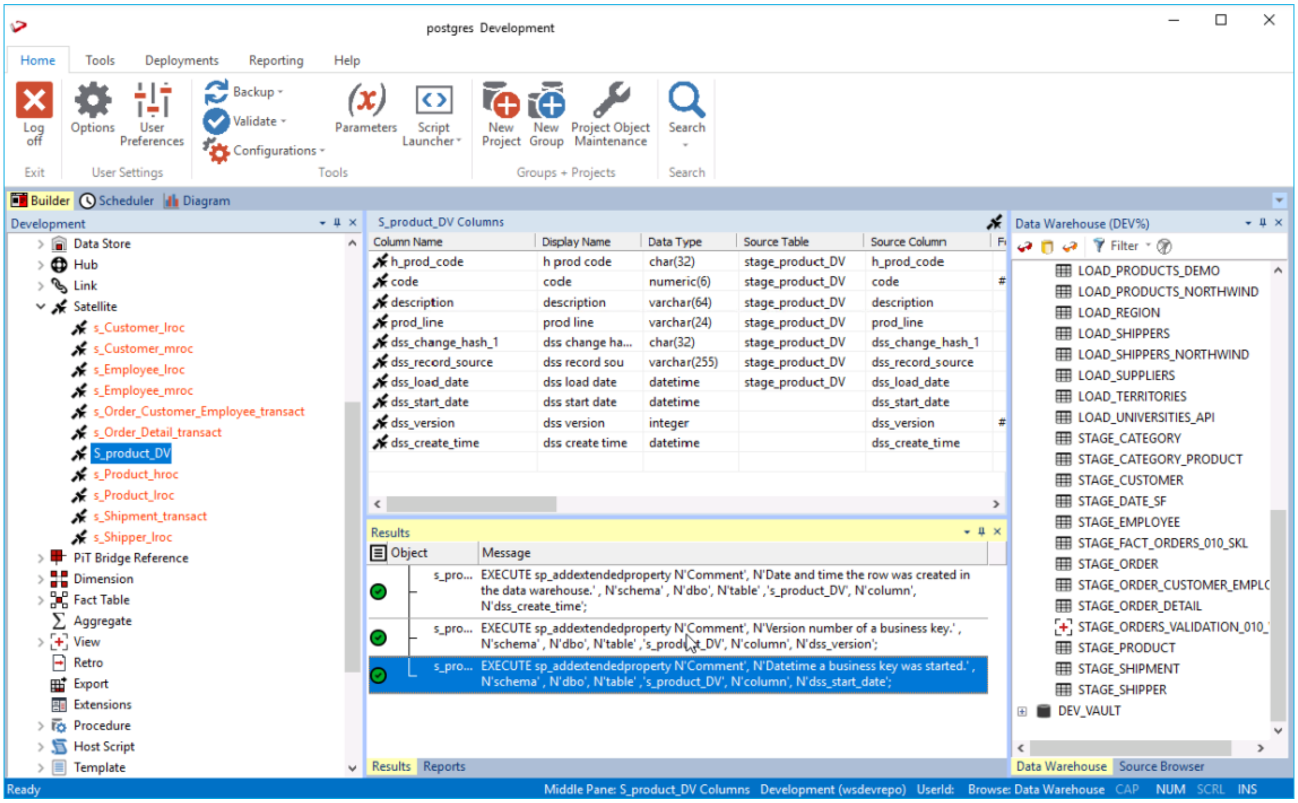


S_product_DV Columns

Column Name	Display Name	Data Type	Source Table	Source Column
h_prod_code	h prod code	char(32)	stage_product_DV	h_prod_code
code	code	numeric(6)	stage_product_DV	code
description	description	varchar(64)	stage_product_DV	description
prod_line	prod line	varchar(24)	stage_product_DV	prod_line
dss_change_hash_1	dss change ha...	char(32)	stage_product_DV	dss_change_hash_1
dss_record_source	dss record sou	varchar(255)	stage_product_DV	dss_record_source
dss_load_date	dss load date	datetime	stage_product_DV	dss_load_date
dss_start_date	dss start date	datetime		dss_start_date
dss_version	dss version	integer		dss_version
dss_create_time	dss create time	datetime		dss_create_time

There are no items to show.

8. Right-click the new Satellite table you defined in the left pane and select **Create (ReCreate)** from the context menu to create the table. The **Results** pane displays confirmation that the Satellite table was successfully created.



S_product_DV Columns

Column Name	Display Name	Data Type	Source Table	Source Column	Fi
h_prod_code	h prod code	char(32)	stage_product_DV	h_prod_code	
code	code	numeric(6)	stage_product_DV	code	#
description	description	varchar(64)	stage_product_DV	description	
prod_line	prod line	varchar(24)	stage_product_DV	prod_line	
dss_change_hash_1	dss change ha...	char(32)	stage_product_DV	dss_change_hash_1	
dss_record_source	dss record sou	varchar(255)	stage_product_DV	dss_record_source	
dss_load_date	dss load date	datetime	stage_product_DV	dss_load_date	
dss_start_date	dss start date	datetime		dss_start_date	
dss_version	dss version	integer		dss_version	#
dss_create_time	dss create time	datetime		dss_create_time	

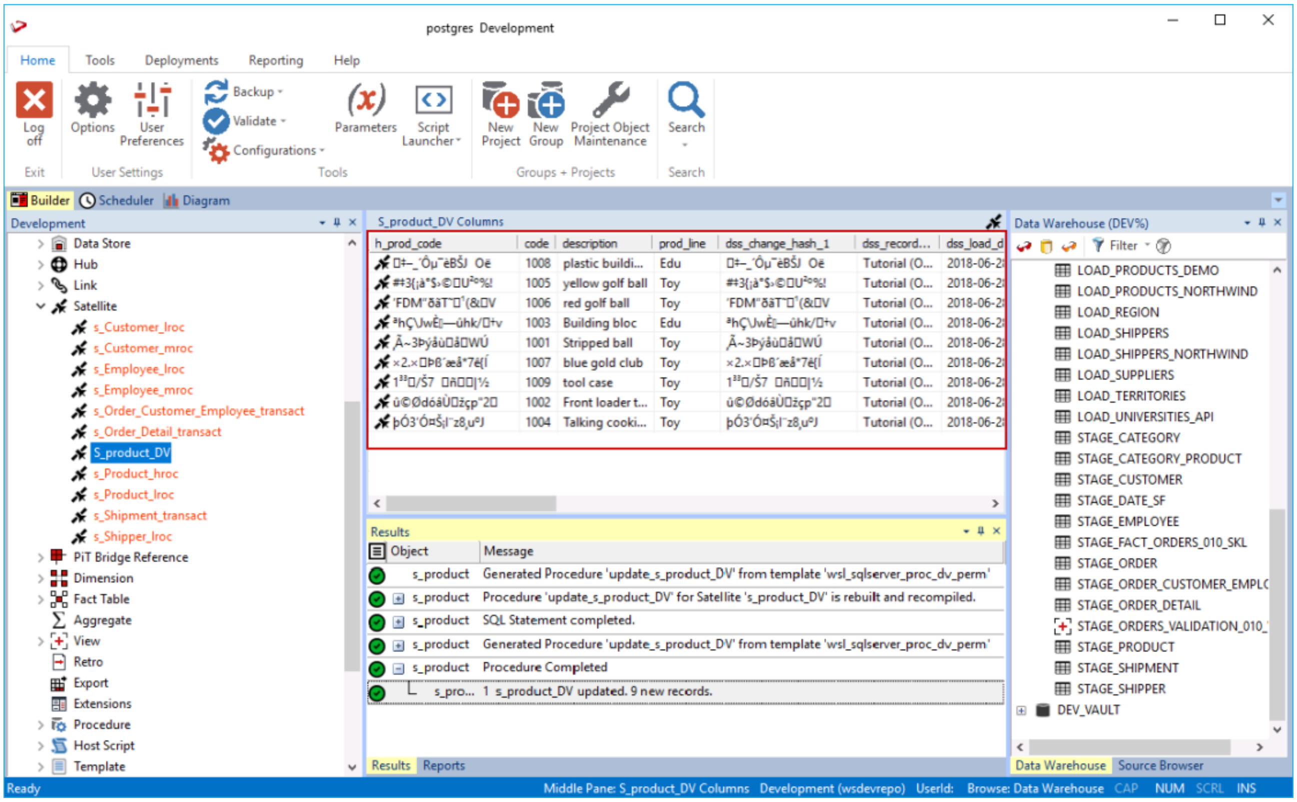
Results

Object	Message
s_pro...	EXECUTE sp_addextendedproperty N'Comment', N'Date and time the row was created in the data warehouse.' , N'schema' , N'dbo', N'table' ,'s_product_DV', N'column', N'dss_create_time';
s_pro...	EXECUTE sp_addextendedproperty N'Comment', N'Version number of a business key.' , N'schema' , N'dbo', N'table' ,'s_product_DV', N'column', N'dss_version';
s_pro...	EXECUTE sp_addextendedproperty N'Comment', N'Datetime a business key was started.' , N'schema' , N'dbo', N'table' ,'s_product_DV', N'column', N'dss_start_date';

After the new Satellite table is defined and created, clicking the **Rebuild** button in the **Table Properties** screen launches the Wizard for generating the update procedure to populate the table. This Wizard utilizes a template to create the procedure.

The detailed steps for using this Wizard is described in **Generating Update Procedures for the Hub, Link and Satellite Tables**.

Once you have run the generated update procedures, you can view the generated Satellite Hash keys, by right-clicking the new Satellite table you created in the left pane and then selecting **Display Data** from the context menu:



S_product_DV Columns

h_prod_code	code	description	prod_line	dss_change_hash_1	dss_record...	dss_load_d
□‡–_´Ôµ¯èBŠJ Oë	1008	plastic buildi...	Edu	□‡–_´Ôµ¯èBŠJ Oë	Tutorial (O...	2018-06-2
#‡3{¡à*$›©□U²º%!	1005	yellow golf ball	Toy	#‡3{¡à*$›©□U²º%!	Tutorial (O...	2018-06-2
'FDM"ðãT˜□¹(&□V	1006	red golf ball	Toy	'FDM"ðãT˜□¹(&□V	Tutorial (O...	2018-06-2
ªhÇ\JwÈ□—ûhk/□+v	1003	Building bloc	Edu	ªhÇ\JwÈ□—ûhk/□+v	Tutorial (O...	2018-06-2
Ã~3Þýåù□å□WÚ	1001	Stripped ball	Toy	Ã~3Þýåù□å□WÚ	Tutorial (O...	2018-06-2
×2.×□ÞB´æå*7ê{Í	1007	blue gold club	Toy	×2.×□ÞB´æå*7ê{Í	Tutorial (O...	2018-06-2
1³³□/Š7 □ñ□□|½	1009	tool case	Toy	1³³□/Š7 □ñ□□|½	Tutorial (O...	2018-06-2
û©ØdóâÙ□žçp"2□	1002	Front loader t...	Toy	û©ØdóâÙ□žçp"2□	Tutorial (O...	2018-06-2
þÓ3'Ó¤Š¡l¨z8,uºJ	1004	Talking cooki...	Toy	þÓ3'Ó¤Š¡l¨z8,uºJ	Tutorial (O...	2018-06-2

Results

Object	Message
s_product	Generated Procedure 'update_s_product_DV' from template 'wsl_sqlserver_proc_dv_perm'
s_product	Procedure 'update_s_product_DV' for Satellite 's_product_DV' is rebuilt and recompiled.
s_product	SQL Statement completed.
s_product	Generated Procedure 'update_s_product_DV' from template 'wsl_sqlserver_proc_dv_perm'
s_product	Procedure Completed
s_pro...	1 s_product_DV updated. 9 new records.

Generating Update Procedures for Hub, Link and Satellite Tables

The following describe the steps for generating update procedures via a template.

Hub table

After successfully defining and creating the Hub table, you can generate the update procedure via a template to populate the table.

1. Click the **Rebuild** button in the table **Properties** screen to launch the procedure generation Wizard to populate the table.

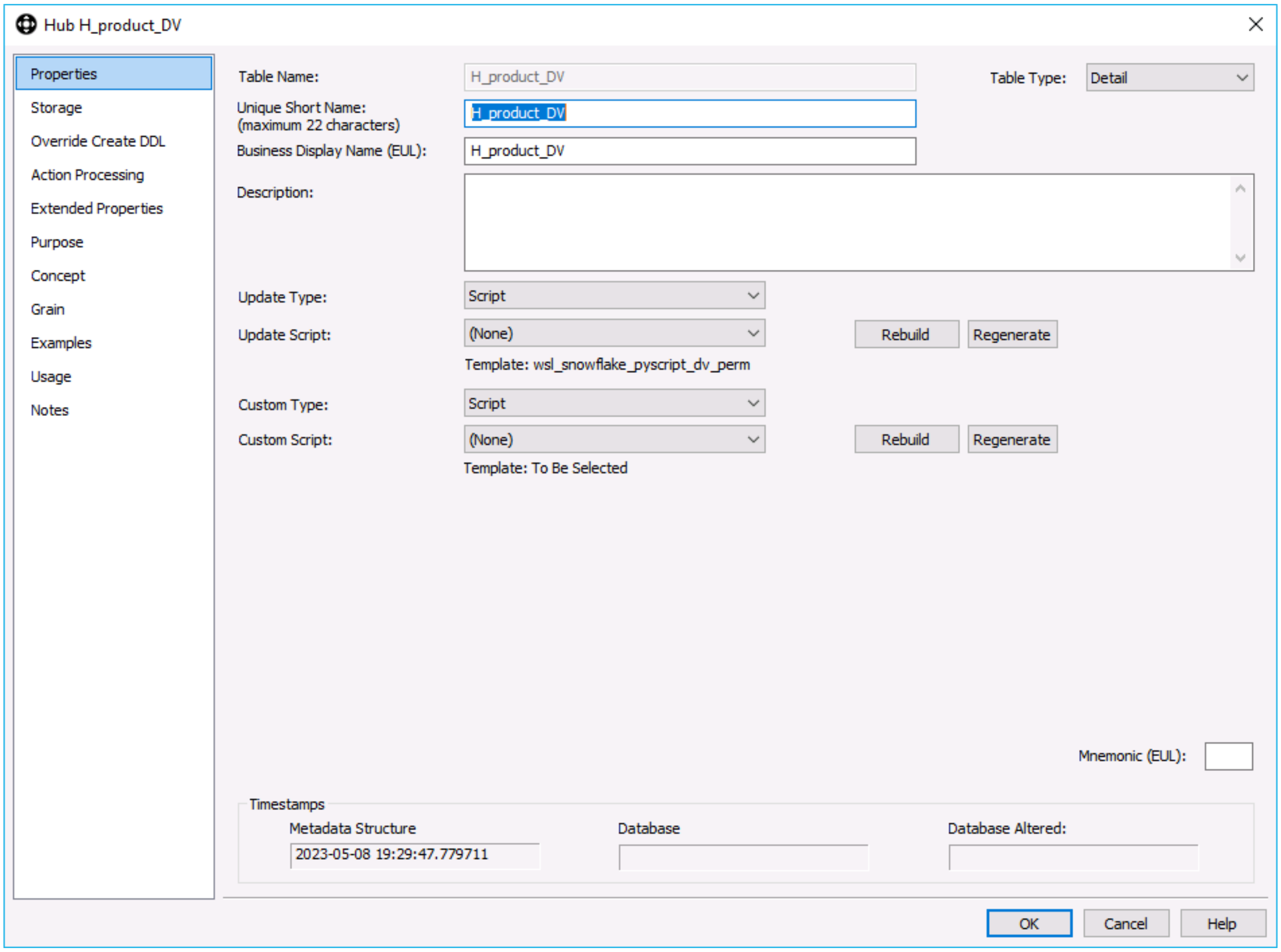


Table Name: H_product_DV

Table Type: Detail

Unique Short Name: (maximum 22 characters) H_product_DV

Business Display Name (EUL): H_product_DV

Description:

Update Type: Script

Update Script: (None)

Template: wsl_snowflake_pyscript_dv_perm

Custom Type: Script

Custom Script: (None)

Template: To Be Selected

Mnemonic (EUL):

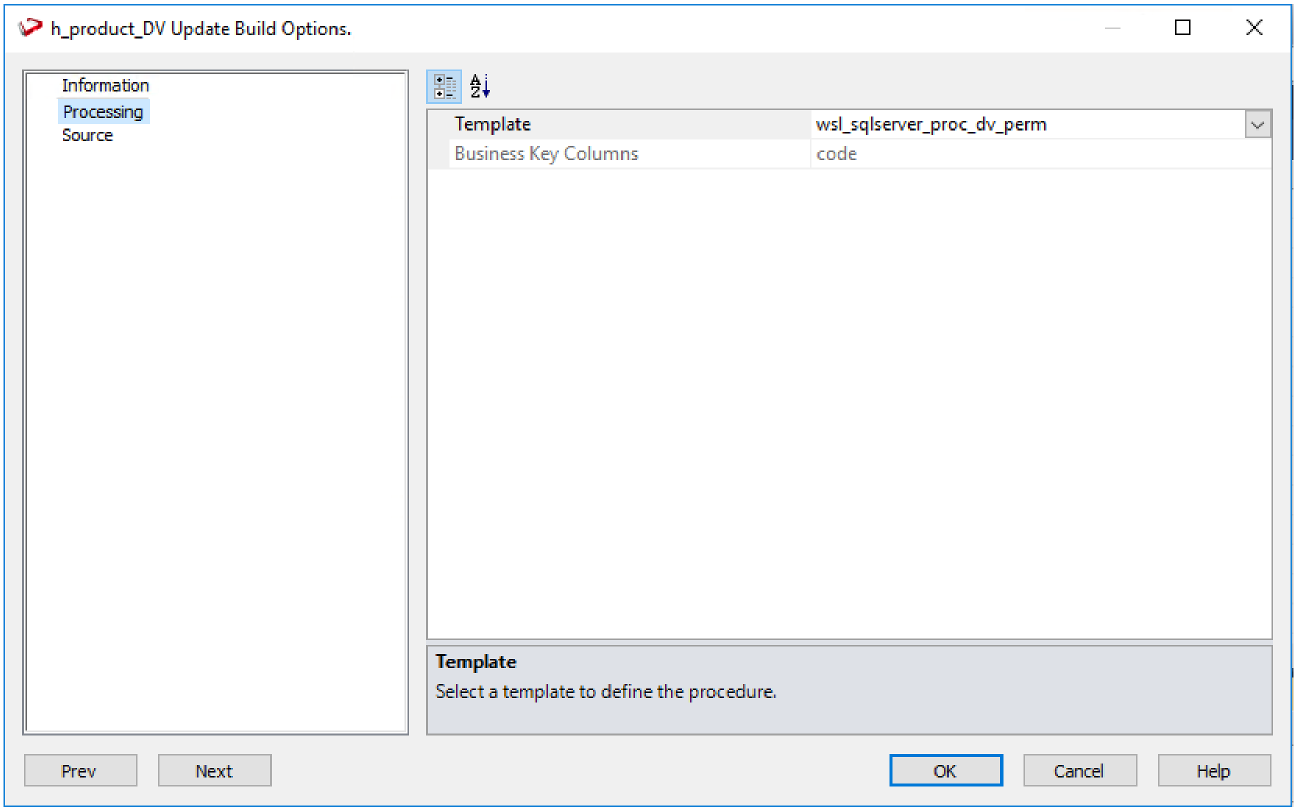
Timestamps

Metadata Structure	Database	Database Altered:
2023-05-08 19:29:47.779711		

Note

RED displays the name of the previously used update procedure template by default, below the **Update Procedure** drop-down.

2. On the **Processing** tab of **Table Update Build Options** screen, select the template to use from the **Template** drop-down or use the previous update procedure template.

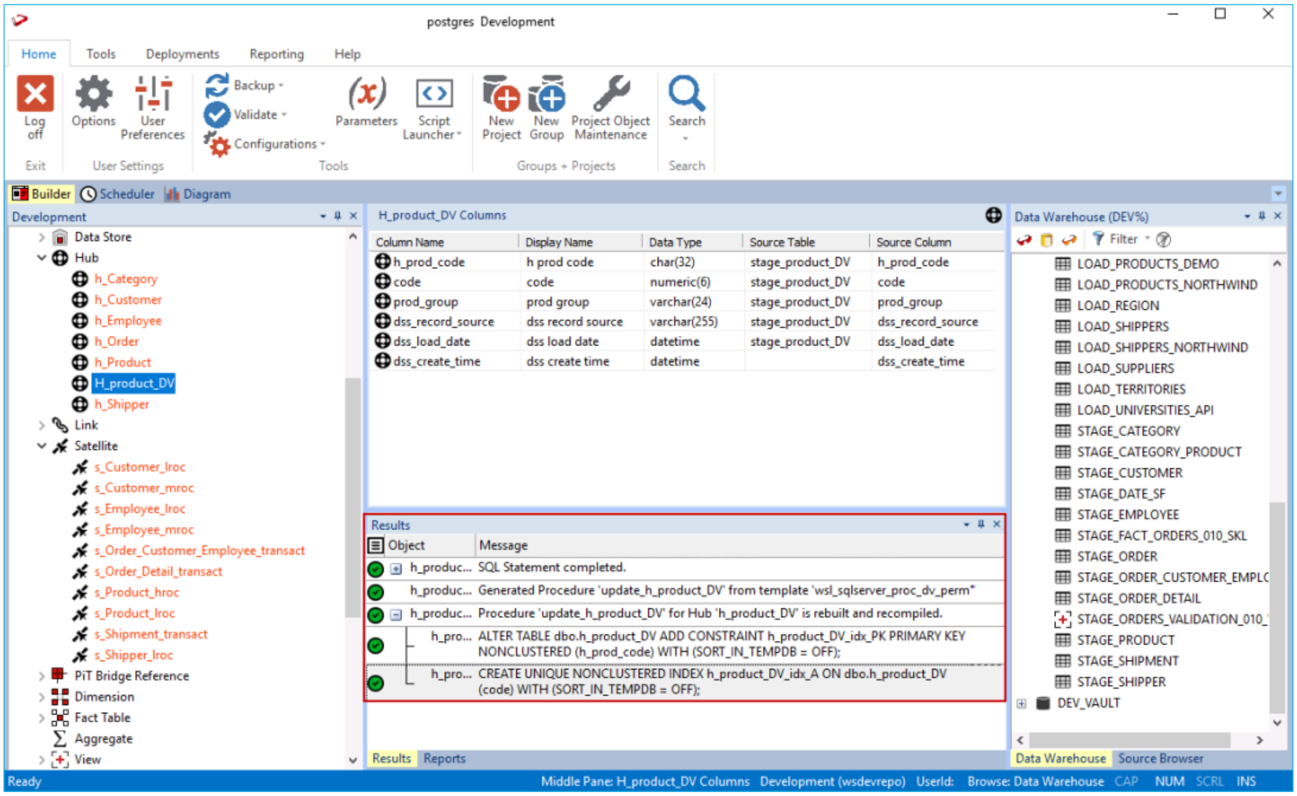


h_product_DV Update Build Options.

Template	wsl_sqlserver_proc_dv_perm
Business Key Columns	code

Template
Select a template to define the procedure.

3. Click **OK** to proceed with the procedure generation. The **Results** pane displays confirmation that the procedure was generated.



H_product_DV Columns

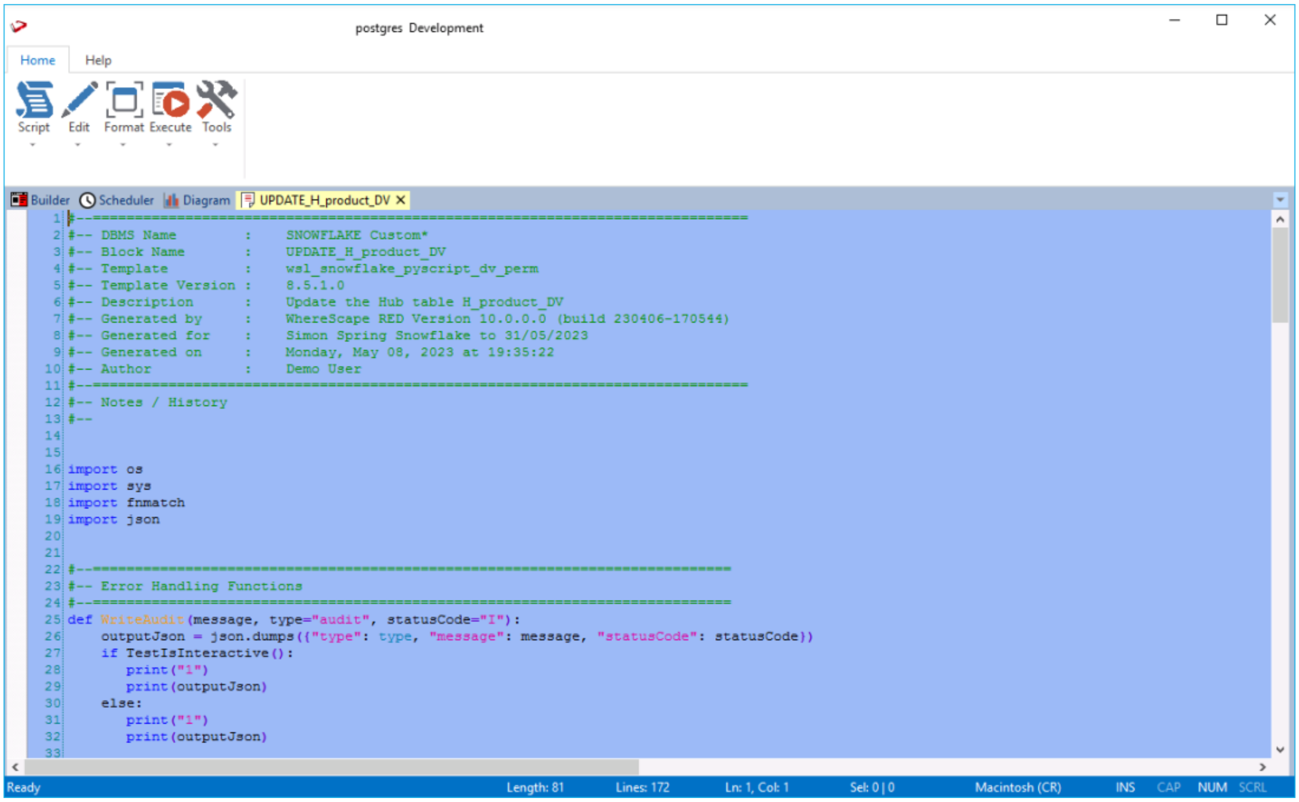
Column Name	Display Name	Data Type	Source Table	Source Column
h_prod_code	h prod code	char(32)	stage_product_DV	h_prod_code
code	code	numeric(6)	stage_product_DV	code
prod_group	prod group	varchar(24)	stage_product_DV	prod_group
dss_record_source	dss record source	varchar(255)	stage_product_DV	dss_record_source
dss_load_date	dss load date	datetime	stage_product_DV	dss_load_date
dss_create_time	dss create time	datetime		dss_create_time

Results

Object	Message
h_produc...	SQL Statement completed.
h_produc...	Generated Procedure 'update_h_product_DV' from template 'wsl_sqlserver_proc_dv_perm"
h_produc...	Procedure 'update_h_product_DV' for Hub 'h_product_DV' is rebuilt and recompiled.
h_pro...	ALTER TABLE dbo.h_product_DV ADD CONSTRAINT h_product_DV_idx_PK PRIMARY KEY NONCLUSTERED (h_prod_code) WITH (SORT_IN_TEMPDB = OFF);
h_pro...	CREATE UNIQUE NONCLUSTERED INDEX h_product_DV_idx_A ON dbo.h_product_DV (code) WITH (SORT_IN_TEMPDB = OFF);

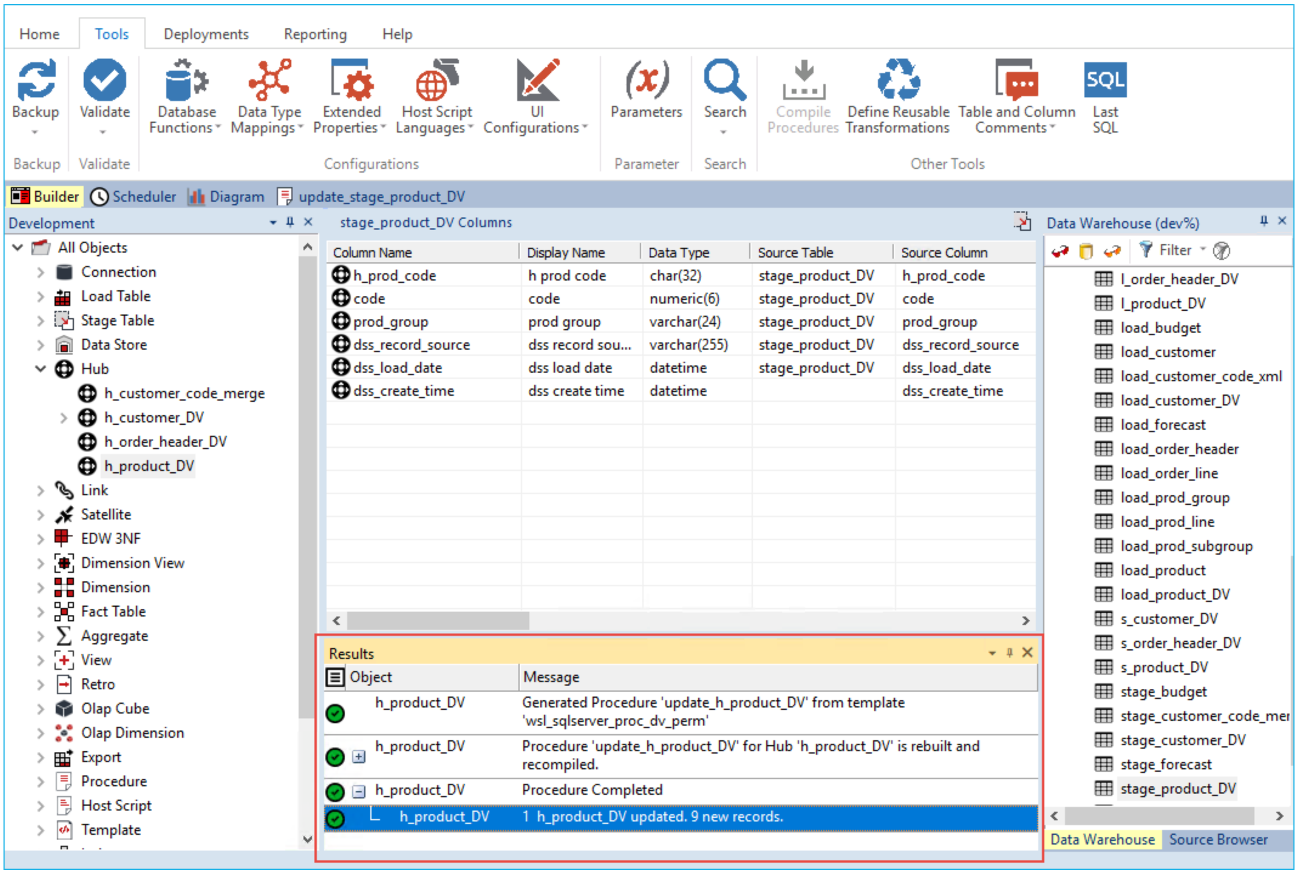
Ready   Middle Pane: H_product_DV Columns   Development (wsdevrepo)   UserId: Browse: Data Warehouse   CAP   NUM   SCRL   INS

4. Right-click the Hub table in the left pane, under the **Hub** objects list and select **Code > View update** from the context menu to view the contents of the update procedure generated.



```
#--==============================================================================
#-- DBMS Name          :    SNOWFLAKE Custom*
#-- Block Name         :    UPDATE_H_product_DV
#-- Template           :    wsl_snowflake_pyscript_dv_perm
#-- Template Version   :    8.5.1.0
#-- Description        :    Update the Hub table H_product_DV
#-- Generated by       :    WhereScape RED Version 10.0.0.0 (build 230406-170544)
#-- Generated for      :    Simon Spring Snowflake to 31/05/2023
#-- Generated on       :    Monday, May 08, 2023 at 19:35:22
#-- Author             :    Demo User
#--==============================================================================
#-- Notes / History
#--


import os
import sys
import fnmatch
import json


#--============================================================================
#-- Error Handling Functions
#--============================================================================
def WriteAudit(message, type="audit", statusCode="I"):
    outputJson = json.dumps({"type": type, "message": message, "statusCode": statusCode})
    if TestIsInteractive():
       print("1")
       print(outputJson)
    else:
       print("1")
       print(outputJson)
```

5. Right-click the Hub table in the left pane and select **Execute Update Procedure** from the context menu to run the procedure. The **Results** pane displays the number of records created.



Column Name	Display Name	Data Type	Source Table	Source Column
h_prod_code	h prod code	char(32)	stage_product_DV	h_prod_code
code	code	numeric(6)	stage_product_DV	code
prod_group	prod group	varchar(24)	stage_product_DV	prod_group
dss_record_source	dss record sou...	varchar(255)	stage_product_DV	dss_record_source
dss_load_date	dss load date	datetime	stage_product_DV	dss_load_date
dss_create_time	dss create time	datetime		dss_create_time

Object	Message
h_product_DV	Generated Procedure 'update_h_product_DV' from template 'wsl_sqlserver_proc_dv_perm'
h_product_DV	Procedure 'update_h_product_DV' for Hub 'h_product_DV' is rebuilt and recompiled.
h_product_DV	Procedure Completed
h_product_DV	1 h_product_DV updated. 9 new records.

Link and Satellite Tables

Follow the same steps described above to create and execute the update procedures for the Link and Satellite tables, via a template to populate the tables.

Changing the Data Vault Hash Key Function in WhereScape RED 6.9.1.0 and above

WhereScape RED ships templates (refer to Data Vault Templates) to generate the procedures for **Data Vault** objects, if the customer's license includes **Data Vault** object support for SQL Server.

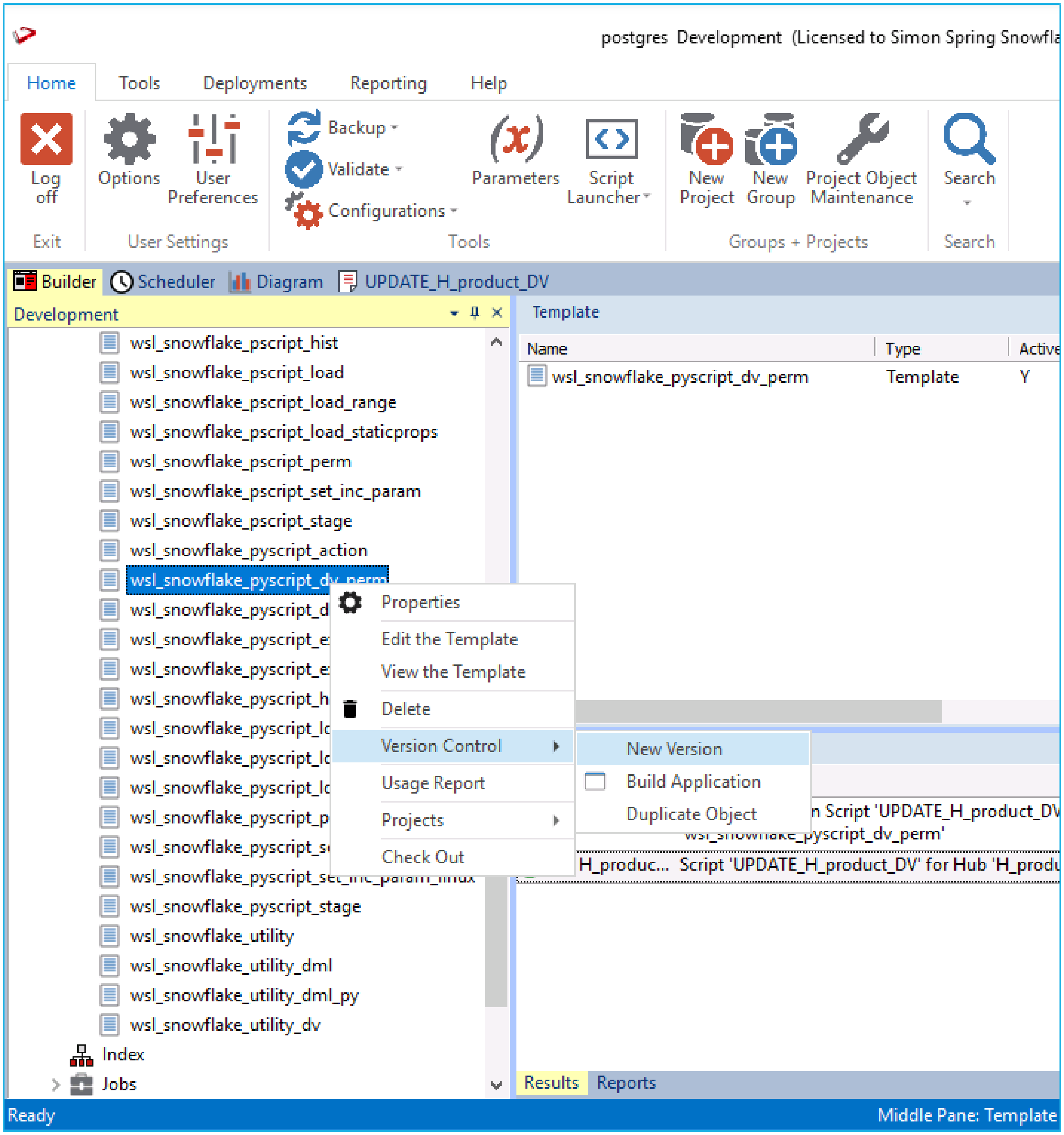
Included in the **Data Vault** utility (**wsl_<database>_utility_dv**) template is an **initial Hash key function**. This Hash key function must be evaluated before deployment, and if an alternate function is preferable for the customer's operating environment, then this function must be amended before Hash keys are created in a production environment.

The following outline the suggested steps to amend the shipped Hash key function, if required.

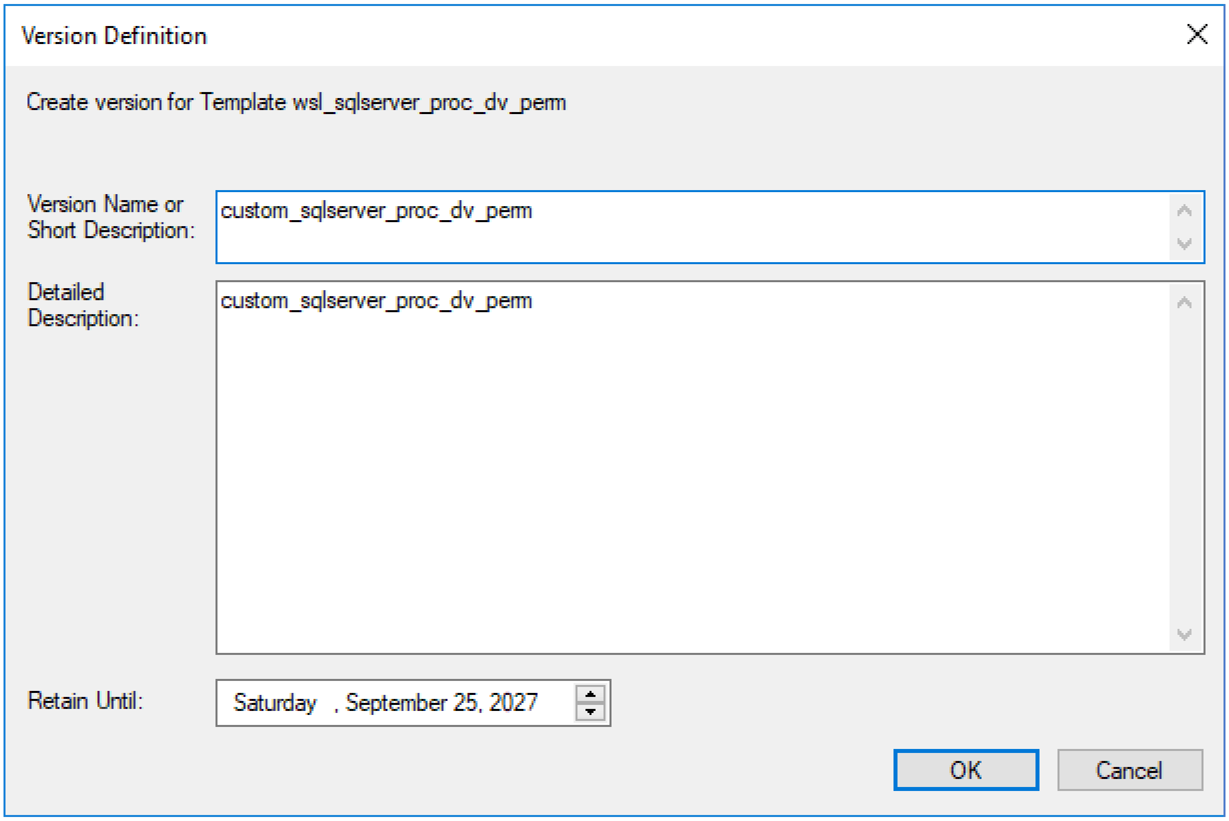
WhereScape **Data Vault** templates are Read-Only (to differentiate from custom templates), the first step is to make a copy of each of the three templates:

- **wsl_<database>_proc_dv_perm**
- **wsl_<database>_proc_dv_stage**
- **wsl_<database>_utility_dv**

1. Right-click the template from the **Template** objects list and select **Version Control > New Version** from the context menu.


postgres Development (Licensed to Simon Spring Snowfla
Home
Tools
Deployments
Reporting
Help
Log off
Options
User Preferences
Backup
Validate
Configurations
Parameters
Script Launcher
New Project
New Group
Project Object Maintenance
Search
Exit
User Settings
Tools
Groups + Projects
Search
Builder
Scheduler
Diagram
UPDATE_H_product_DV
Development
wsl_snowflake_pscript_hist
wsl_snowflake_pscript_load
wsl_snowflake_pscript_load_range
wsl_snowflake_pscript_load_staticprops
wsl_snowflake_pscript_perm
wsl_snowflake_pscript_set_inc_param
wsl_snowflake_pscript_stage
wsl_snowflake_pyscript_action
wsl_snowflake_pyscript_dv_perm
wsl_snowflake_pyscript_d
wsl_snowflake_pyscript_e
wsl_snowflake_pyscript_e
wsl_snowflake_pyscript_h
wsl_snowflake_pyscript_lo
wsl_snowflake_pyscript_lo
wsl_snowflake_pyscript_lo
wsl_snowflake_pyscript_p
wsl_snowflake_pyscript_s
wsl_snowflake_pyscript_set_inc_param_linux
wsl_snowflake_pyscript_stage
wsl_snowflake_utility
wsl_snowflake_utility_dml
wsl_snowflake_utility_dml_py
wsl_snowflake_utility_dv
Index
Jobs
Properties
Edit the Template
View the Template
Delete
Version Control
Usage Report
Projects
Check Out
New Version
Build Application
Duplicate Object
Template
Name
Type
Active
wsl_snowflake_pyscript_dv_perm
Template
Y
n Script 'UPDATE_H_product_DV
wsl_snowflake_pyscript_dv_perm'
H_produc... Script 'UPDATE_H_product_DV' for Hub 'H_produ
Results
Reports
Ready
Middle Pane: Template

2. Enter the version name and other details on the **Version Definition** window and then click **OK** to save the new version of the template.



Version Definition

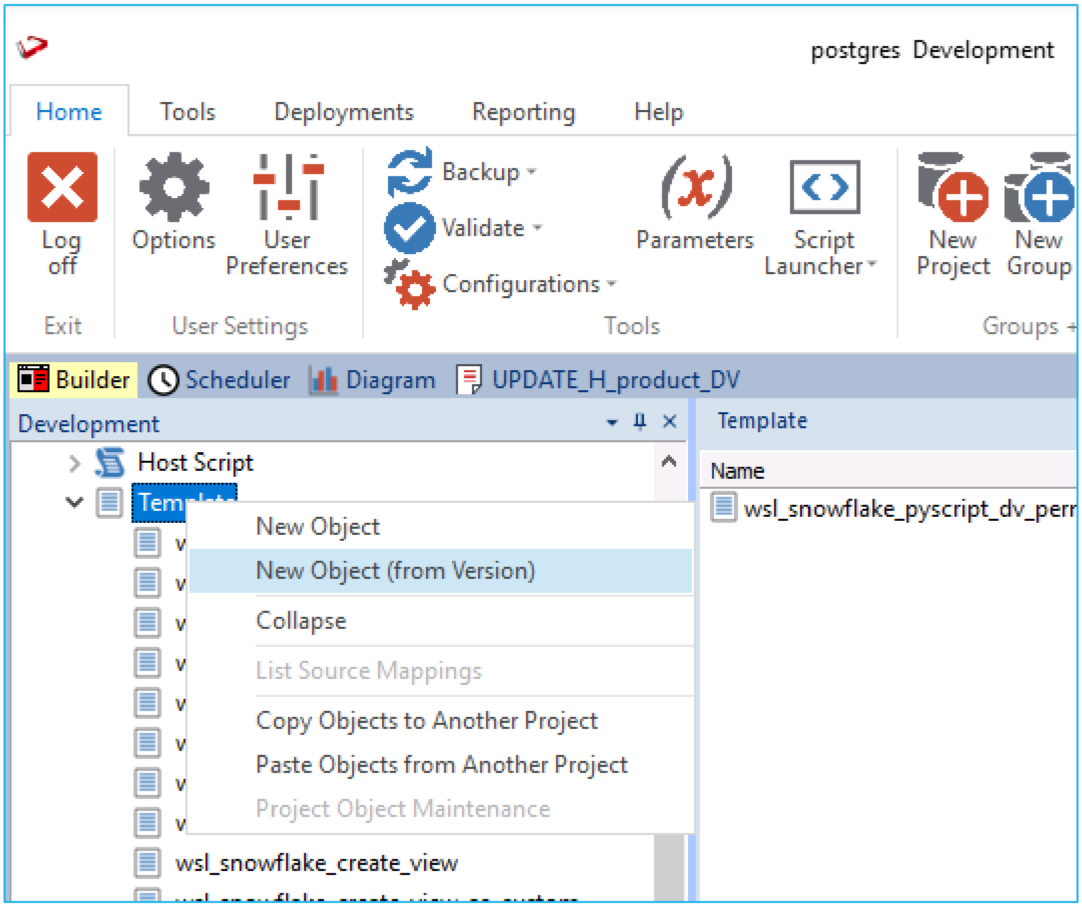
Create version for Template wsl_sqlserver_proc_dv_perm

Version Name or Short Description: custom_sqlserver_proc_dv_perm

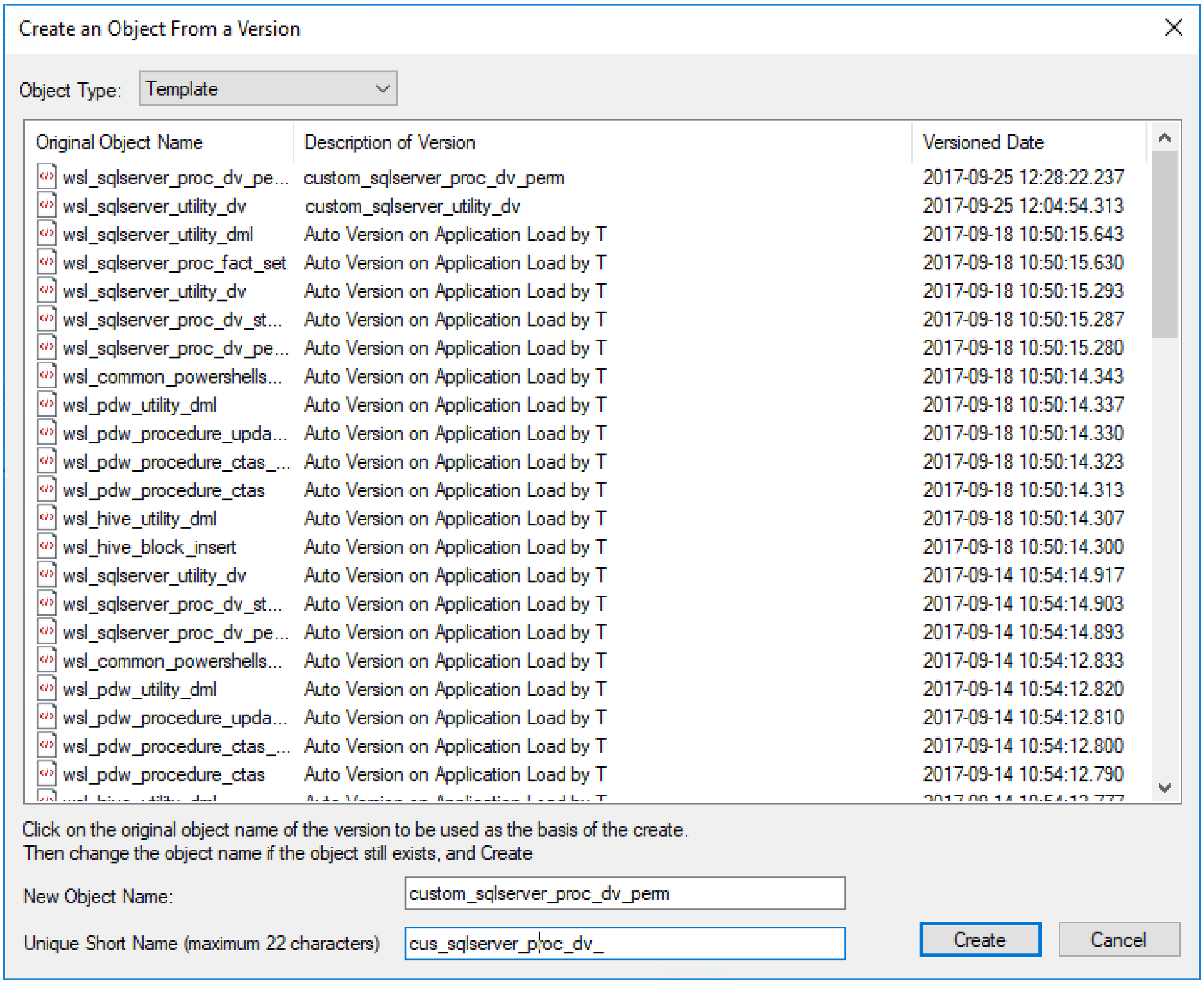
Detailed Description: custom_sqlserver_proc_dv_perm

Retain Until: Saturday , September 25, 2027

3. Right-click the **Template** object type and select **New Object (from version)** from the context menu.



4. Select the new version of the template you created from the list in the **Create an Object From a Version** window and then enter the new object name.



Create an Object From a Version

Object Type: Template

Original Object Name	Description of Version	Versioned Date
wsl_sqlserver_proc_dv_pe...	custom_sqlserver_proc_dv_perm	2017-09-25 12:28:22.237
wsl_sqlserver_utility_dv	custom_sqlserver_utility_dv	2017-09-25 12:04:54.313
wsl_sqlserver_utility_dml	Auto Version on Application Load by T	2017-09-18 10:50:15.643
wsl_sqlserver_proc_fact_set	Auto Version on Application Load by T	2017-09-18 10:50:15.630
wsl_sqlserver_utility_dv	Auto Version on Application Load by T	2017-09-18 10:50:15.293
wsl_sqlserver_proc_dv_st...	Auto Version on Application Load by T	2017-09-18 10:50:15.287
wsl_sqlserver_proc_dv_pe...	Auto Version on Application Load by T	2017-09-18 10:50:15.280
wsl_common_powershells...	Auto Version on Application Load by T	2017-09-18 10:50:14.343
wsl_pdw_utility_dml	Auto Version on Application Load by T	2017-09-18 10:50:14.337
wsl_pdw_procedure_upda...	Auto Version on Application Load by T	2017-09-18 10:50:14.330
wsl_pdw_procedure_ctas_...	Auto Version on Application Load by T	2017-09-18 10:50:14.323
wsl_pdw_procedure_ctas	Auto Version on Application Load by T	2017-09-18 10:50:14.313
wsl_hive_utility_dml	Auto Version on Application Load by T	2017-09-18 10:50:14.307
wsl_hive_block_insert	Auto Version on Application Load by T	2017-09-18 10:50:14.300
wsl_sqlserver_utility_dv	Auto Version on Application Load by T	2017-09-14 10:54:14.917
wsl_sqlserver_proc_dv_st...	Auto Version on Application Load by T	2017-09-14 10:54:14.903
wsl_sqlserver_proc_dv_pe...	Auto Version on Application Load by T	2017-09-14 10:54:14.893
wsl_common_powershells...	Auto Version on Application Load by T	2017-09-14 10:54:12.833
wsl_pdw_utility_dml	Auto Version on Application Load by T	2017-09-14 10:54:12.820
wsl_pdw_procedure_upda...	Auto Version on Application Load by T	2017-09-14 10:54:12.810
wsl_pdw_procedure_ctas_...	Auto Version on Application Load by T	2017-09-14 10:54:12.800
wsl_pdw_procedure_ctas	Auto Version on Application Load by T	2017-09-14 10:54:12.790
wsl_hive_utility_dml	Auto Version on Application Load by T	2017-09-14 10:54:12.777

Click on the original object name of the version to be used as the basis of the create.
Then change the object name if the object still exists, and Create

New Object Name: custom_sqlserver_proc_dv_perm

Unique Short Name (maximum 22 characters) cus_sqlserver_proc_dv_

5. Click **Create** to create the new Template object.
6. Follow the same steps to create the other new templates with unique names.
7. Modify the replacement utility template with the new function in this section:

```
{%- set hashFunctionPatBeg  = "CAST(HASHBYTES('md5',"       -%}

{%- set hashFunctionPatEnd  = ") AS CHAR(32))"  -%}

{%- set hashColTransPattern = "COALESCE(CAST([SRCCOL] AS VARCHAR(MAX)),'null')"  -%}

{%- set hashConcatPattern   = " +'||'+"         -%}
```

In each of the addHubHashKey, addLinkHashKey and addChangeHashKey macros.

8. In the two replacement proc_dv templates, change the import command to reference the new utility template:

```
{% import "custom_sqlserver_utility_dv" %}
```

9. Update existing **Data Vault Stage** table objects to reference the replacement **proc_dv_stage** template (the **Usage Report** on the context menu of the old template objects can help locate these).

10. After successfully testing the new templates, it is recommended to delete the original **WSL Data Vault** templates to avoid confusion.

Note

The above workflow does not provide allowance for the migration of any existing **Data Vault** records, created with a previous **Data Vault** function. This workflow can be used for new **Data Vault** implementations. In an existing **Data Vault** system, changing the Hash key function requires migration of existing Hash keys which is not provided in this workflow.

Custom Objects

Custom1 and **Custom2** objects are user defined objects. These Object types can be renamed in the **Tools > Options > Object Types > Object Names** menu.

A Custom object license is required for these object types.

Custom objects have the same options and properties as EDW 3NF tables. Refer to **EDW 3NF Tables** for details.

Fact Tables

A Fact table is normally defined, for our purposes, as a table with facts (measures) and dimensional keys that allow the linking of multiple dimensions. It is normally illustrated in the form of a Star Schema with the central Fact table and the outlying dimensions.

The ultimate goal of the Fact table is to provide business information to the end user community. In many cases, different types of Fact tables are required to address different end user requirements. For simplicity, the different types of Fact tables are grouped together under the following headings:

- Detail
- Rollup
- KPI
- Snapshot (essentially a type of Rollup fact table)
- Slowly Changing
- Partitioned

Detail Fact Tables

A Detail Fact table is normally a transactional table that represents the business data at its lowest level of granularity. In many ways, these tables reflect the business processes within the organization.
These Fact tables are usually large and are focused on a specified analysis area.

There may be quite a large number of Detail Fact tables in a data warehouse implementation, of which only a few are used on any regular basis by the end user community. The disadvantage of these Fact tables is that they provide isolated pools of information. Although joined by conformed dimensions, it is still often difficult to answer queries across the various analysis areas.
They do however provide the ultimate drill down for all information, and also the platform on which to build higher level and more complex Fact tables.

In terms of the time dimension, Detail Fact tables are typically at a daily or even hourly granular level.

An example of a Detail Fact table may be the sales, or orders Fact tables that have a daily granularity, and show all sales by product, by customer etc. on a given day.

Creating Detail Fact Tables

A Detail Fact table is typically created by dragging a staging table onto a fact table list. Once released, the dialog to create the fact table will commence.

When a staging table is dragged into the drop target pane (middle pane), all Fact tables are by default **Detail Fact tables**.

If manually creating a table, then the table type can be selected from the Properties window of the Fact table.

Detail Fact Columns

The columns for a Detail Fact table are typically those of the staging table. Such Fact tables typically contain a wide range of measures and attributes as well as the business keys used to look up the artificial dimension keys.

These business keys should be included whenever possible as they provide a means of rebuilding the dimensional link. If size prohibits their inclusion it is probably be necessary to backup or archive all source data to ensure that the Fact table can be rebuilt.

These Fact tables normally contain a large number of attributes such as dates, which have not been converted to dimensions. Also contained would be information, such as order numbers, invoice numbers etc.

Generating the Detail Fact Update Procedure

Once a detail fact table has been defined in the metadata and created in the database, an update procedure can be generated to handle the update of the fact table.

Notes

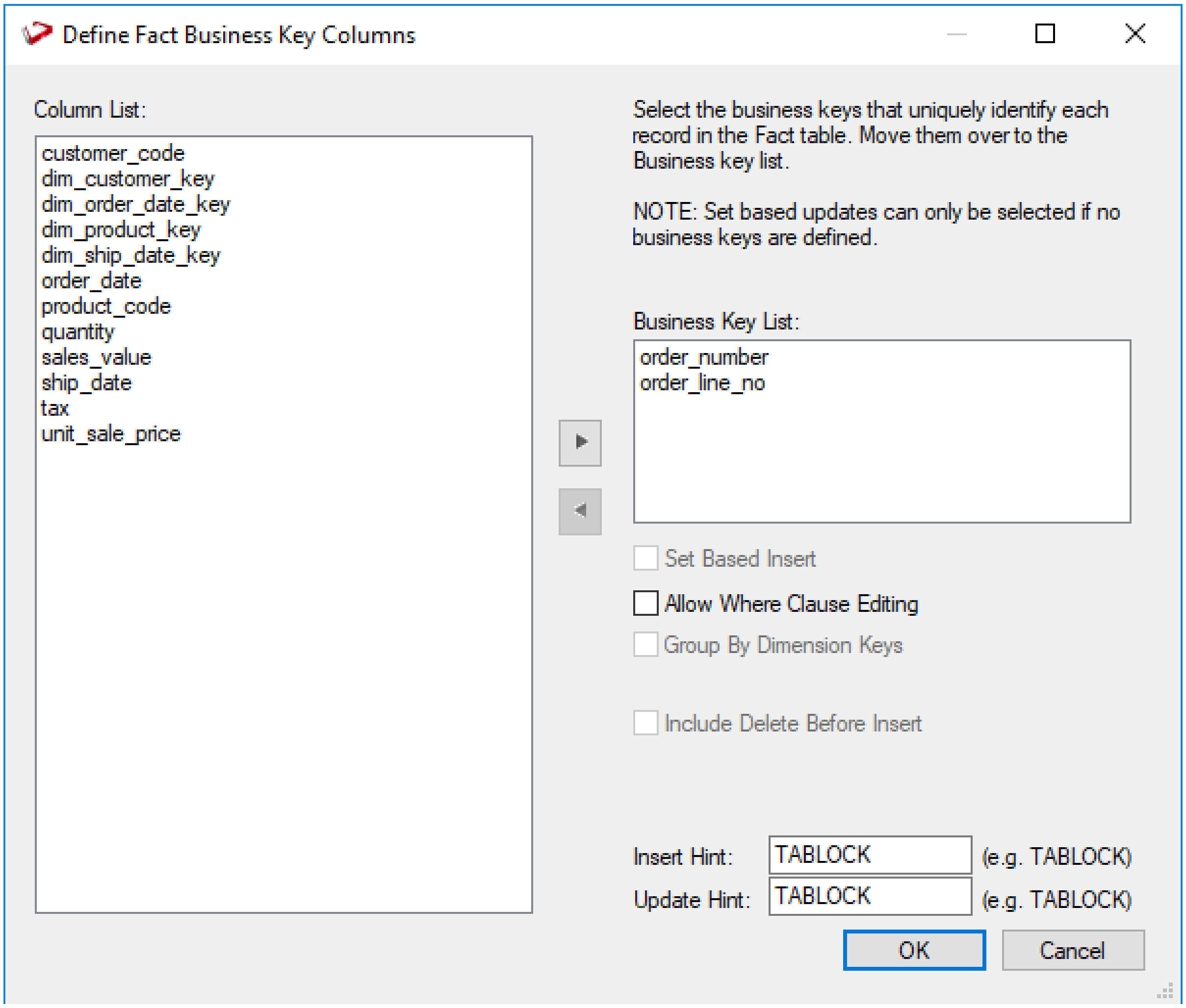
- You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.
- If the fact table is created in a custom database target, then an additional processing option is available—PowerShell script-based processing. Refer to **Script Templates for Custom Database Table Objects** for details.

Generating a Procedure

1. To generate a procedure, right-click on the fact table in the left pane and select **Properties**.
2. Click on the **Rebuild** button to start the process of generating the new procedure.
3. A series of prompts are displayed during the procedure generation based on the type of load information.

Define Fact Business Key Columns

The first dialog displayed when generating a detail fact table update procedure is the define fact business key columns dialog, asking for the business key that will uniquely identify each fact table record.
The source table from which the fact table is derived would normally have some form of unique constraint applied. In most cases this will be the business key. In the example below, the order_id and order_line_no are selected for the business key list.



Define Fact Business Key Columns

Column List:

- customer_code
- dim_customer_key
- dim_order_date_key
- dim_product_key
- dim_ship_date_key
- order_date
- product_code
- quantity
- sales_value
- ship_date
- tax
- unit_sale_price

Select the business keys that uniquely identify each record in the Fact table. Move them over to the Business key list.

NOTE: Set based updates can only be selected if no business keys are defined.

Business Key List:

- order_number
- order_line_no

□ Set Based Insert

□ Allow Where Clause Editing

□ Group By Dimension Keys

□ Include Delete Before Insert

Insert Hint: TABLOCK (e.g. TABLOCK)

Update Hint: TABLOCK (e.g. TABLOCK)

A business key can be made up of multiple columns, but it must provide a unique identifier. Where multiple columns separately uniquely identify rows in the fact table, choose one to act as the primary business key.

For example a source table may have a unique constraint on both a product code and a product description. Therefore the description as well as the code must be unique.

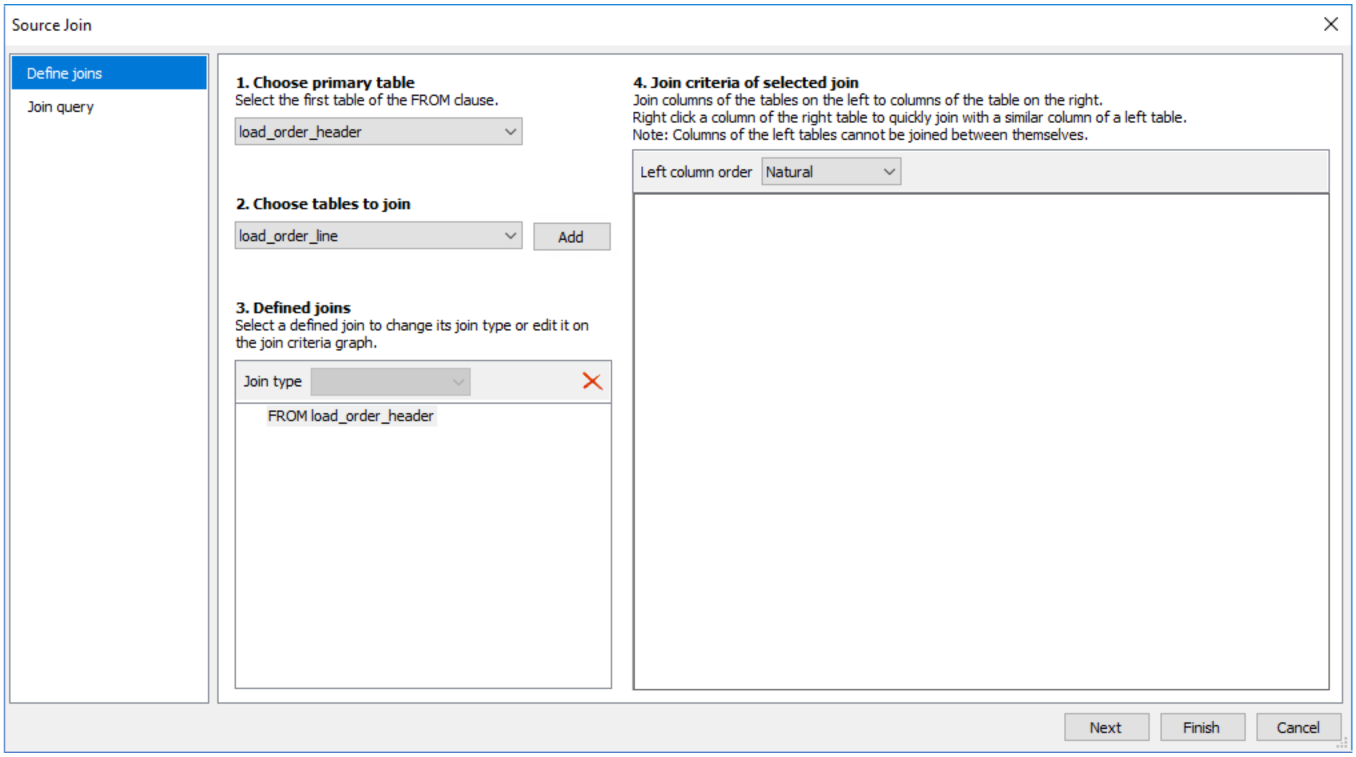
Note

None of the columns chosen as the business key should ever contain a NULL value.

Fields	Description
Allow Where Clause Editing	If the **Allow Where Clause Editing** option is selected, then the next dialog to be displayed is the **Source Join** wizard, else this is skipped.
Insert Hint	Enter a database hint to be used in the INSERT statement.
Update Hint	Enter a database hint to be used in the UPDATE statement.

Source Join Wizard

This wizard is used to join source tables and add 'Where' clauses.


Source Join
Define joins
Join query
1. Choose primary table
Select the first table of the FROM clause.
load_order_header
2. Choose tables to join
load_order_line
Add
3. Defined joins
Select a defined join to change its join type or edit it on the join criteria graph.
Join type
FROM load_order_header
4. Join criteria of selected join
Join columns of the tables on the left to columns of the table on the right.
Right click a column of the right table to quickly join with a similar column of a left table.
Note: Columns of the left tables cannot be joined between themselves.
Left column order
Natural
Next
Finish
Cancel

Source table joins should have been performed in the stage table, refer to **Generating the Staging Update Procedure** for details.

Since most of the real work is done in the stage table, this is all that is needed to build the Detail Fact Update Procedure.

Rollup or Combined Fact Tables

Rollup fact tables are typically the most heavily used fact tables in a data warehouse implementation. They provide a smaller set of information at a higher level of granularity.
These fact tables typically have a monthly granularity in the time dimension and reflect figures for the month or month-to-date. They often combine multiple analysis areas (detail fact tables) into one super set of information. This allows simpler comparisons or joining of otherwise isolated pools of information.

An example may be the combination of daily sales, weekly demand forecasts, and monthly budgets, into one rollup fact table. Such a fact table provides the ability to compare monthly or month-to-date sales against budget and forecast.

These rollup/combined fact tables may and often do contain different levels of granularity, as well as different sets of dimensions.

For example, it may be possible to look at sales and forecasts by promotion, but the budget information does not have a populated promotion dimension, so we will not get a budgeted value for the promotion.

Creating Rollup Fact Tables

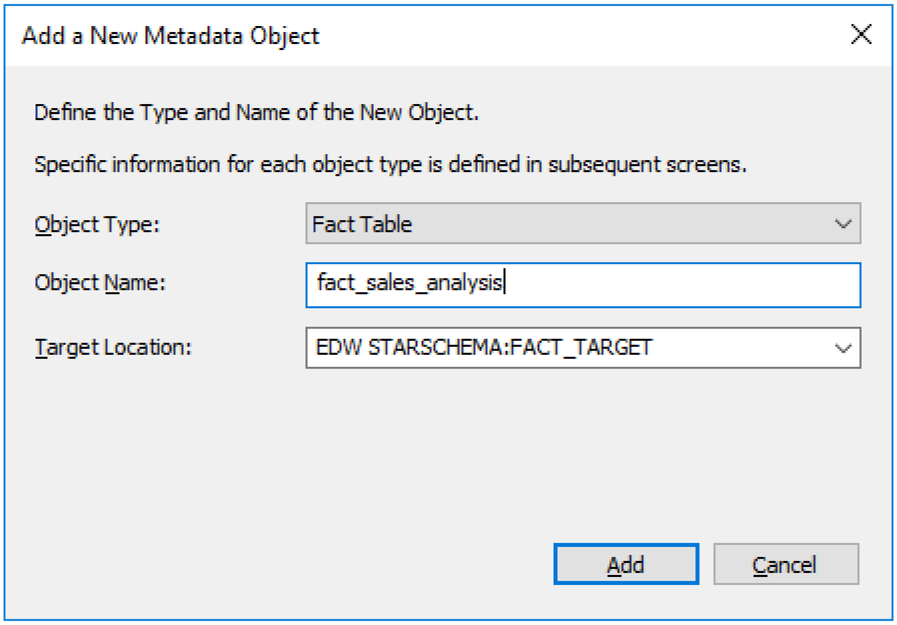
A rollup or combined Fact table is typically created by dragging a Fact table onto a Fact table list. In the example below, a double-click on the Fact table object group under the Sales project produced a list in the middle pane, showing the existing Fact tables.

A **fact_sales_analysis** rollup Fact table is created after selecting and dragging the **fact_sales_detail** table in the right pane into the middle pane.

The initial object create window appears. Note that the name implies that this is a rollup Fact table.

The name is changed to the one chosen above.

A list of columns is displayed and those not required are deleted (everything except additive measures and dimensions).



Add a New Metadata Object

Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: Fact Table

Object Name: fact_sales_analysis

Target Location: EDW STARSCHEMA:FACT_TARGET

Drag and drop additional columns from other Fact tables into the middle pane if required. Columns that have the same name need to be changed.

For example, if we acquired a quantity column from **fact_sales_detail** and a column of the same name from **fact_forecast** we may choose to change the names to **actual_quantity** and **forecast_quantity**.

Once all columns have been defined:

1. Create the table.
2. Edit the properties of the table and request that a new procedure be created.
3. Click **OK** and the window to create a template procedure is commenced. This process can include the definition of the date dimension granularity. Monthly rollup is the norm.
4. To just combine Fact tables with no rollup, select the date dimension key as directed in the window. Other forms of dimension rollup require some alteration to the produced template procedure.

Rollup Fact Columns

The columns for a rollup Fact table are typically the dimensions, and only those key measures that are added.

For example, in the sales world, measures may simply be quantity, kilograms, $value, forecast_quantity, forecast_$value, budget_quantity, budget_$value.

Generating the Rollup Fact Update Procedure

Once a rollup fact table has been defined in the metadata and created in the database, an update procedure can be generated to handle the update of the fact table.

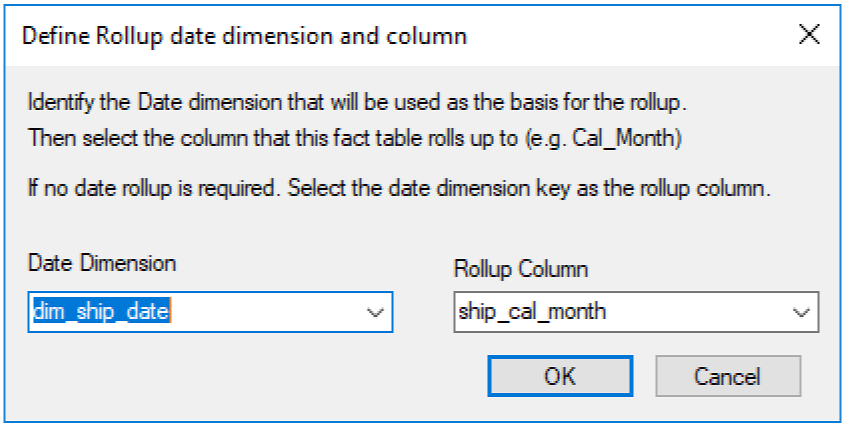
Note: You can also generate an update procedure via a template, refer to Rebuilding Update Procedures for details.

Generating a Procedure

1. To generate a procedure, right-click on the fact table in the left pane and select **Properties** to edit the properties for the fact table.
2. Click the **Rebuild** button to start the process of generating the new procedure.
3. A series of prompts are displayed during the procedure generation based on the type of load information.

Define Rollup date dimension and column

The first dialog displayed when generating a detail fact table update procedure is the define rollup date dimension and column window:


Define Rollup date dimension and column

Identify the Date dimension that will be used as the basis for the rollup.
Then select the column that this fact table rolls up to (e.g. Cal_Month)

If no date rollup is required. Select the date dimension key as the rollup column.

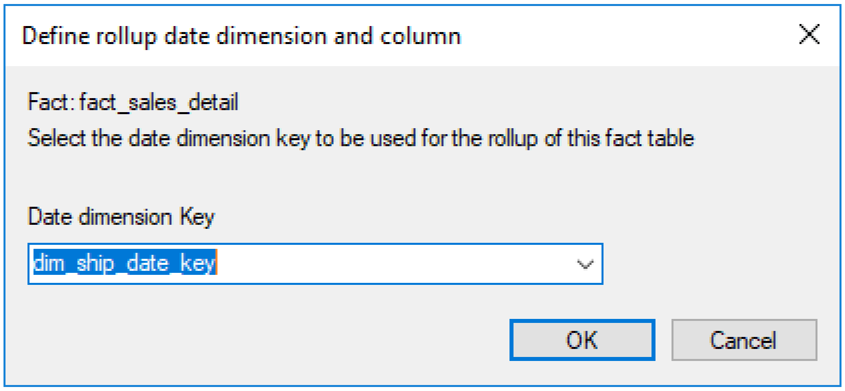
Date Dimension: dim_ship_date

Rollup Column: ship_cal_month

RED needs to know which date is driving the rollup and to which level of detail we wish to set as our base.

In this example, we have chosen **dim_ship_date** as our date dimension and **ship_cal_month** as our rollup column to set the level to summarize all transactions. Click **OK**.

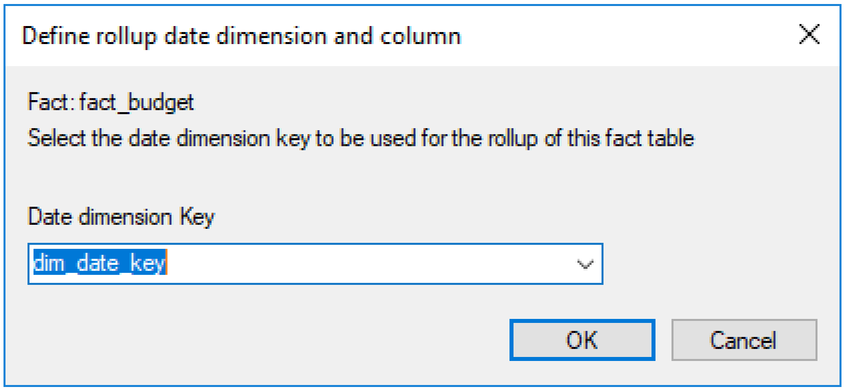
RED also needs to understand which date field in the actuals should be used to base the rollup; we have chosen **dim_ship_date_key** and click **OK**.


Define rollup date dimension and column

Fact: fact_sales_detail
Select the date dimension key to be used for the rollup of this fact table

Date dimension Key: dim_ship_date_key

Once again, select the date dimension key and click **OK**.


Define rollup date dimension and column

Fact: fact_budget
Select the date dimension key to be used for the rollup of this fact table

Date dimension Key: dim_date_key

KPI Fact Tables

Key Performance Indicator (KPI) fact tables share a lot of similarities with the rollup fact tables in that they are typically at a monthly level of granularity and they combine information from multiple detail fact tables. They differ in that they provide specific sets of pre-defined information (KPIs). Such KPIs are often difficult to answer and require the implementation of some form of business rule. Also they often provide pre-calculated **year-to-date**, **same month last year**, and **previous year-to-date** values for easier comparisons. They have a KPI dimension which provides a list of all the specific statistics or performance indicators we are tracking.

An example from our previous sales examples may be a set of KPIs dealing with customer acquisition and loss. We may have a specific KPI that records the number of new customers during the period and the $value of their business during that period. Another KPI may record the percentage growth of all customers over the last six months. Another may show the number of customers who closed their accounts in the period, and the value of their business over the preceding 12 months.

KPI fact tables require the specific definition of each KPI.

Creating the KPI Dimension

The KPI dimension essentially provides a record of each KPI or statistic we are tracking. There should be one row in this table for each such element.

The KPI fact table requires that this dimension contain at the least an artificial dimension key and a business key. We will refer to the business key as the KPI Identifier.

The following example covers the manual creation of a basic KPI dimension:

1. Right-click the **Dimension** object group in the left pane and select **New Object**.
2. Give the new dimension a name for example **dim_kpi**.
3. Select a **Normal** dimension and click **OK** on the Properties screen to define the dimension table.
4. Right-click the dimension name in the left pane and select **Add Column**, add the three columns defined in the following table.

Field	Column 1	Column 2	Column 3
Column Name	dim_kpi_key	kpi_identifier	kpi_name
Business display	kpi artificial key	kpi unique id	kpi name
Datatype • **SQL Server**	integer	varchar(12)	varchar(256)
Nulls	Clear	Clear	Checked
Key Type	0	A	Clear
Artificial Key	Checked	Clear	Clear
Primary Business	Clear	Checked	Clear

The first two columns are required, and should be set up as shown above, although a different data type can be used for the second column, and the names can be chosen to suit the requirements.

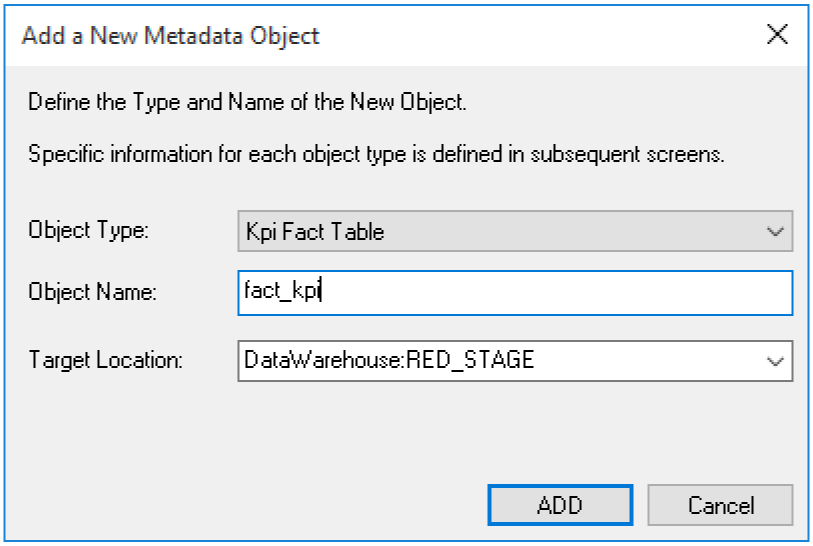
Creating KPI Fact Tables

A KPI fact table is normally created manually.

1. Right-click the Fact Table object group in the left pane and select **New Object**.
2. A dialog displays as shown below.

Note

The default object type is **Fact Table**. This must be changed to **KPI Fact Table** and a table name entered.



Add a New Metadata Object

Define the Type and Name of the New Object.

Specific information for each object type is defined in subsequent screens.

Object Type: Kpi Fact Table

Object Name: fact_kpi

Target Location: DataWarehouse:RED_STAGE

Adding columns

Once created, the KPI fact table need to have columns added. New columns can be added by dragging columns from the browser window, or by manually adding from the right mouse menu. Dimension keys should normally be dragged to ensure that all the properties are correctly set.

The columns for a KPI fact table are typically the dimensions and generic measures. For example, valid measures could be **kpi_count**, **kpi_quantity** and **kpi_value**. The interpretation of these measures depend on the KPI or group of KPIs being queried. As with the rollup fact table it is normal and desirable to keep the number of measures as small as possible. Attributes should not be included in these fact tables. These measures must allow Null values for the generated update procedure to function correctly.

Notes

- The kpi measure column order is important. For example if we have mtd count, quantity and value in that order then the prev month, prev ytd and ytd figures also need count first followed by quantity and finally value. If this order is not adhered to then the generated procedure will load the wrong values into the wrong columns.
- A kpi fact table should also have the column dss_update_time added. This column is required if the kpi table is to be used to populate higher level fact tables, aggregates or cubes. This column should be populated with the current date time whenever the particular row is updated.

Setup of KPI Fact Tables

The set-up of a KPI fact table is somewhat more complex than the other types of fact table. Once all columns have been defined, the following steps need to be taken:

1. Create and populate a KPI dimension that provides at least an id (unique identifier) for each KPI and a description. The key from this dimension must be one of the columns in the fact table.
2. Physically create the KPI fact table in the database once all columns including the KPI dimension key have been added.
3. Define the column type and default value for ALL columns in the table. There must be at least one date dimension key and one KPI dimension key. These attributes are set by selecting a column and modifying its properties.
4. Create the update procedure for the fact table.
5. Set-up each individual KPI.

KPI Setup

KPI Dimension

The KPI dimension provides a means of selecting a specific KPI. This dimension needs to be built in advance of the KPI fact table. It does not need to be populated, as the KPI fact table will add entries to this table as required. A key and some form of unique identifier for the KPI is required, e.g. they may be numbered or given short names.

Create the KPI Fact Table

Right-click the KPI table and select **Create/Recreate** to physically create the table in the database.

Defining column types and default values

1. List all the columns for the KPI fact table.
2. Amend the properties for each column. The column types can be one of the following:

Type	Description
Dimension Key	Any dimension that is not the time or KPI dimension
Time Dimension Key	Time dimension key
KPI Dimension Key	KPI dimension key
Month-to-date Measure	Used to identify the normal KPI measures. These are calculated to populate the KPI during each run.
Year-to-date Measure	Used to identify year-to-date measures for the KPI. Optionally calculated or derived from existing rows and measures in the KPI fact table.
Last year same month Measure	Used to identify measures for same month last year comparisons. Derived from existing rows and measures in the KPI fact table.
Previous year-to-date Measure	Used to identify previous year-to-date measures. Derived from existing rows and measures in the KPI fact table.
Fact Table Key	Not normally used
Date	Not normally used
Attribute	Not normally used
Measure	Not normally used

Tip

If an **Attribute** or **Measure** column is specified then this column will just be set to its default value. In this way an update_time column can be set by making its default value **sysdate**.

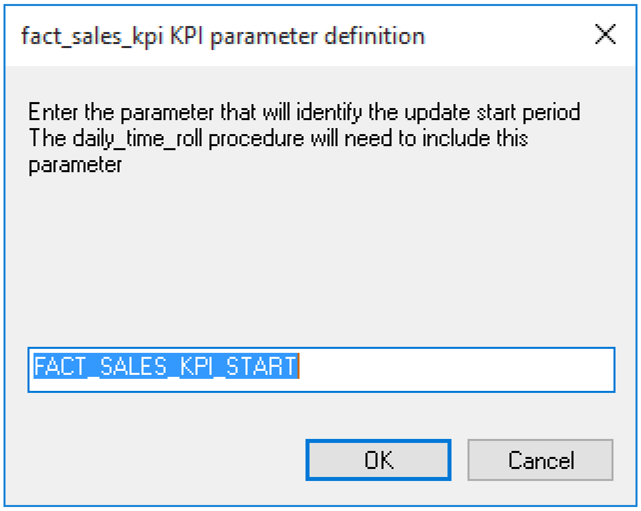
For each column set a default value.

- For measures and dimension keys this would normally be zero (0). Nulls can be represented as two single quotation marks, i.e. ' '
- **sysdate** can be used for a default date value
- These default values populate the columns where and when no other information is available. For example if a KPI only makes use of 4 of the 6 dimensions available then the remaining 2 dimensions will be set to their default values.

Create the 'Update' procedure

Right-click the KPI fact table name and select **Properties**. In the Properties window, set the update procedure list to the **(Build Procedure...)** option. Click **OK** to initiate a series of window prompts to create a procedure used to update the kpi fact table.

The first two dialogs ask for parameter names to define the start and end periods for which the KPIs will be updated. The default naming convention is fact_table_name_START and fact_table_name_END. Accept these default names if possible. These two parameters need to be created, and need to be updated once the KPIs have been defined and tested. Under normal operation these parameters should probably be set daily from the daily_time_roll procedure.



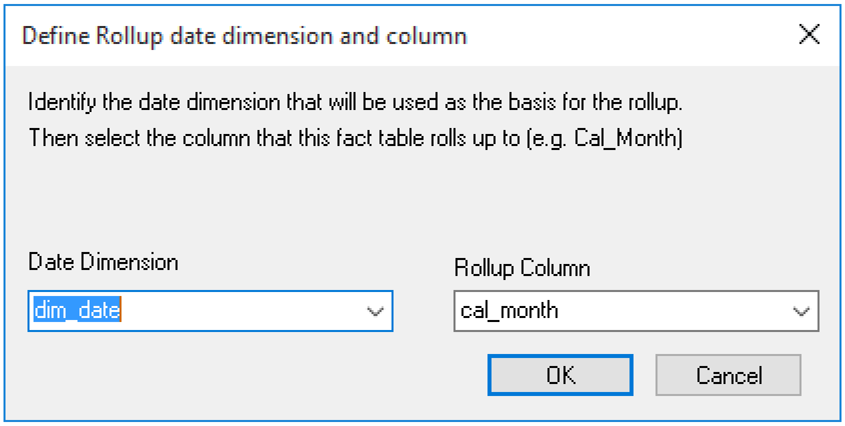
fact_sales_kpi KPI parameter definition

Enter the parameter that will identify the update start period
The daily_time_roll procedure will need to include this parameter

FACT_SALES_KPI_START

The date dimension and column for rollup is required. Normally this would be cal_month or fin_month.

This is based on the assumption that most KPIs are monthly or annual. Subsequently a rollup year and date will be required from the same date dimension.



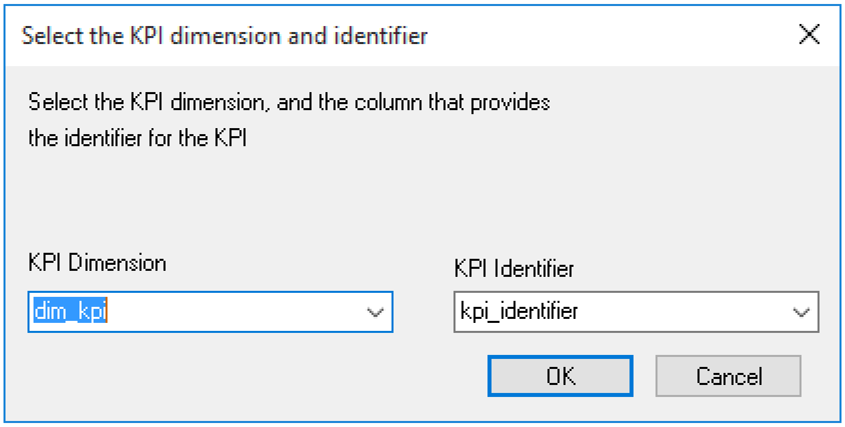
Define Rollup date dimension and column

Identify the date dimension that will be used as the basis for the rollup.
Then select the column that this fact table rolls up to (e.g. Cal_Month)

Date Dimension: dim_date

Rollup Column: cal_month

The KPI dimension is required. Also the identifier column (not the key) that uniquely identifies the KPI. This would normally be a number or short name (max 64 chars).



Select the KPI dimension and identifier

Select the KPI dimension, and the column that provides the identifier for the KPI

KPI Dimension: dim_kpi

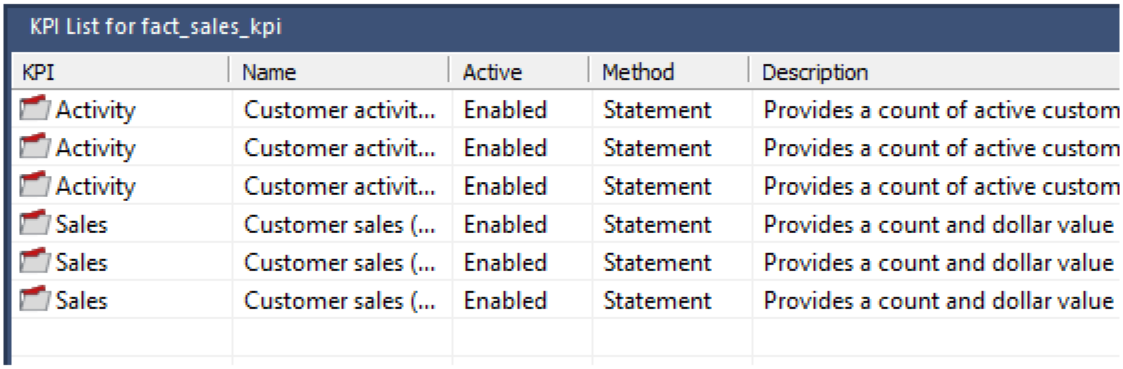
KPI Identifier: kpi_identifier

The procedure is then created and compiled.

Define each KPI

Once the procedure has successfully compiled, we are ready to start setting up each of the KPIs. Right-click the KPI table name to get the pop-up menu.

This menu is similar to the other fact table menus except for the additional options to display and add KPIs. A display of the KPIs will list any KPIs that are currently defined. The result set is as appears below:



KPI List for fact_sales_kpi

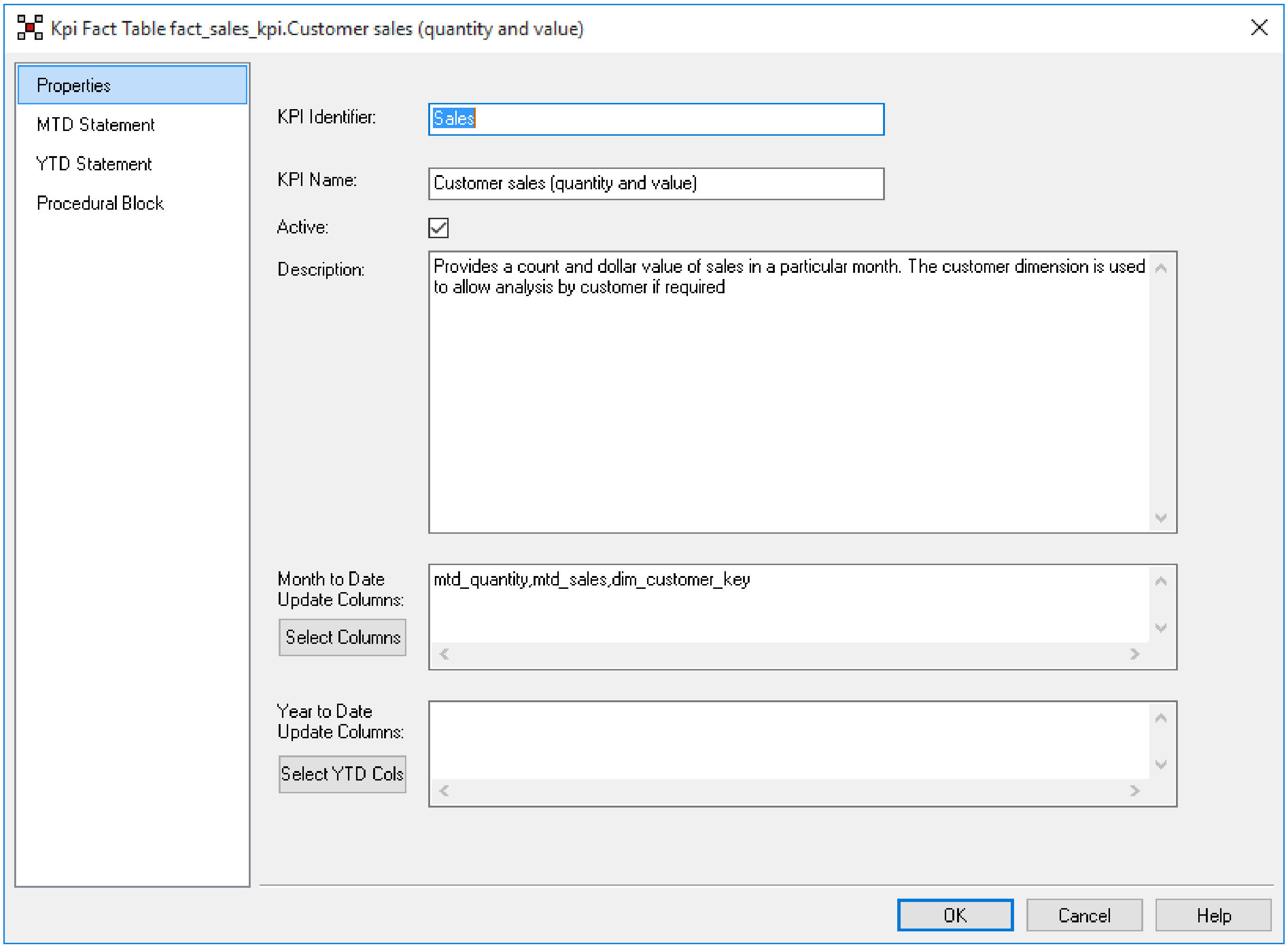
KPI	Name	Active	Method	Description
Activity	Customer activit...	Enabled	Statement	Provides a count of active custom
Activity	Customer activit...	Enabled	Statement	Provides a count of active custom
Activity	Customer activit...	Enabled	Statement	Provides a count of active custom
Sales	Customer sales (...	Enabled	Statement	Provides a count and dollar value
Sales	Customer sales (...	Enabled	Statement	Provides a count and dollar value
Sales	Customer sales (...	Enabled	Statement	Provides a count and dollar value

The ‘Id’ is the same unique identifier that will be found in the kpi dimension. The **Name** is a name entered in the properties of the KPI and does not relate to the dimension in any way. The **Active** option indicates if the KPI will be processed when the **Update** or **Process** menu options are selected. Each KPI can be defined as a SQL Statement, a Procedural Block or a Procedure. The current method used to update the KPI is one of these three and is displayed under **Method**. The description is a comment or Description field entered in the Properties of the KPI and in no way relates in the KPI dimension.

Right-click anywhere in the List window to access the KPI add option. Clicking the right mouse button while positioned over the ‘Id’ opens a pop-up menu which provide options to copy, modify, delete, add and view the KPI statement.

When the add KPI option is selected, a Properties screen appears as shown below. The id and name fields are required. If the **Active** check box is selected ,the procedure is enabled for update.

The **Select Columns** button enables the selection of the columns that this KPI will update during the month to date update pass. The date and KPI dimensions will be updated for you so are not provided as choices. The only choices provided are the ‘Month to date measures’ and the other ‘Dimensions’. Select those measures and dimensions that will be updated. All non-selected measures and dimensions will be set to their default values. The resultant choice is shown in the **Update Columns** window. This window should only be updated via the **Select Columns** button.



Kpi Fact Table fact_sales_kpi.Customer sales (quantity and value)

KPI Identifier: Sales

KPI Name: Customer sales (quantity and value)

Active: ☑

Description: Provides a count and dollar value of sales in a particular month. The customer dimension is used to allow analysis by customer if required

Month to Date Update Columns: mtd_quantity,mtd_sales,dim_customer_key

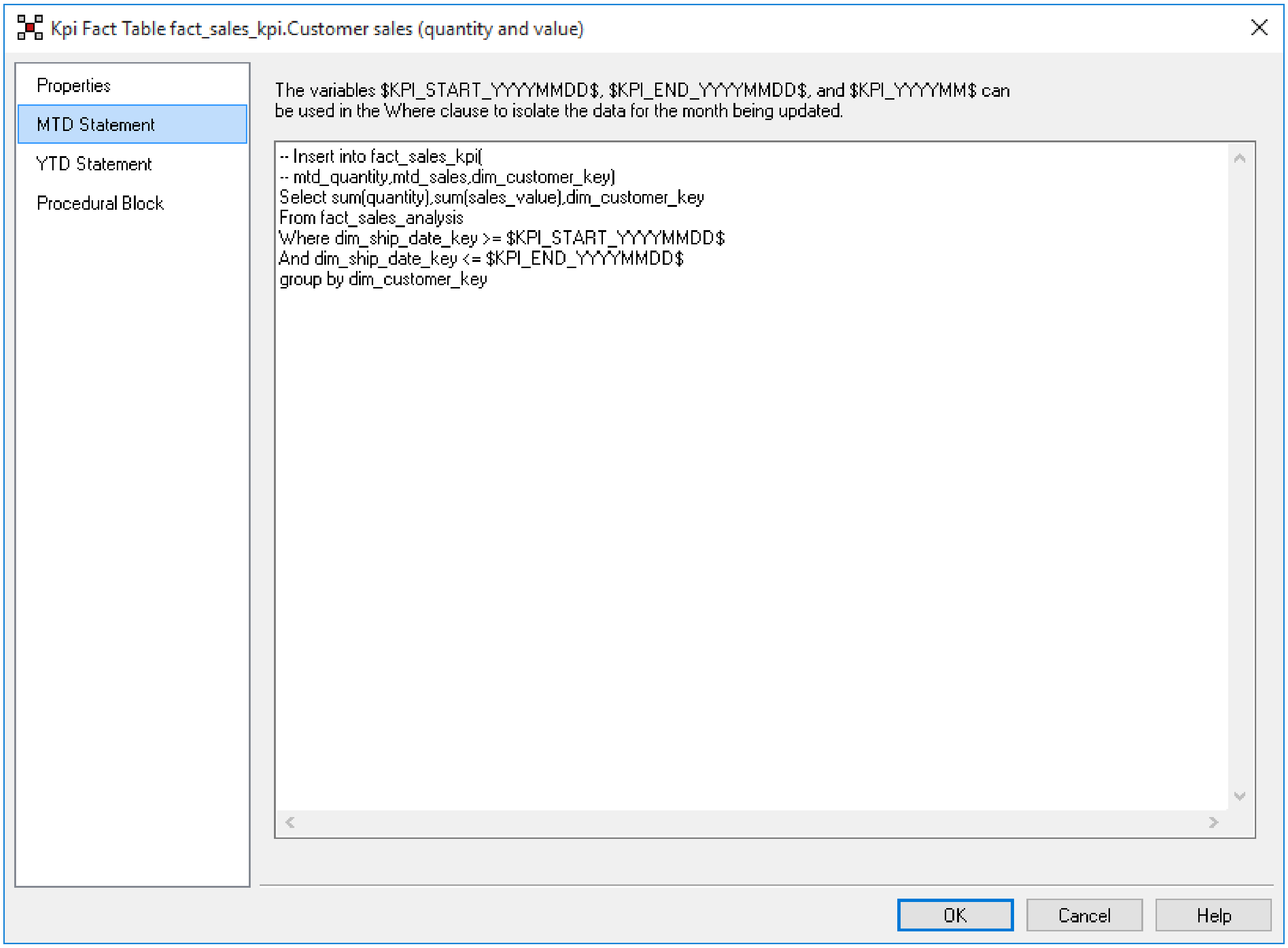
Year to Date Update Columns:

Once these initial values are defined, the **Statement** tabs can be selected to define the update SQL statement. If a procedural block or procedure is required for the update (i.e. too complex for a single SQL statement), then select the **Procedural Block** tab.

The KPI Statement

The **Statement** tabs present edit windows that enable the input of a SELECT statement that will return the values for the columns being updated.

They must return the values for the columns selected in the **Properties** tab in the order shown in the **Update Columns** window. An example of a completed KPI statement follows:


Kpi Fact Table fact_sales_kpi.Customer sales (quantity and value)
Properties
MTD Statement
YTD Statement
Procedural Block
OK
Cancel
Help

The variables $KPI_START_YYYYMMDD$, $KPI_END_YYYYMMDD$, and KPI_YYYYMM can be used in the Where clause to isolate the data for the month being updated.

```
-- Insert into fact_sales_kpi(
-- mtd_quantity,mtd_sales,dim_customer_key)
Select sum(quantity),sum(sales_value),dim_customer_key
From fact_sales_analysis
Where dim_ship_date_key >= $KPI_START_YYYYMMDD$
And dim_ship_date_key <= $KPI_END_YYYYMMDD$
group by dim_customer_key
```

The first two lines are comments inserted (when modifying) to provide a reminder of the columns and order required by the select statement. Anything prior to the initial SELECT statement is ignored.

The statement must make use of at least one of the variables provided to select the appropriate period. This variable is replaced with the values for the period being updated at run time. These values are derived from the start and end parameters defined for the KPI fact table.

In order to group by the non-selected columns and any attributes, etc. WhereScape RED must be able to break this statement into its component Select, From, Where, Group By, and Having components (where present). If a complex statement is used that has more than one of these key words then the primary one must be in UPPER case. For example if a sub query is being used then the main SELECT and FROM key words must be in upper case and must be the only occurrence of these key words in uppercase. WhereScape RED checks and issues a warning if it is unable to resolve the main key words.

To test a statement, right-click the KPI Id and select **View Statement**. This provides a window with the full statement with the default columns inserted and the parameters converted. This statement can be cut and run in SQL Admin (shipped with WhereScape RED) or in a database specific utility for testing.

Procedural Block

The **Procedural Block** tab presents an edit window. If there is anything in this window it is actioned, instead of the statement. A button in the lower right allows a rudimentary template block to be built. If used, this block must do the actual insert statement rather than just the select. It must also handle ALL dimension, month to date measure and non standard columns. In addition, it will need to put default values in columns not actively being updated. The KPI dimension and date dimension keys must also be updated. The output from a ‘View Statement’ option for another KPI gives an indication of what the block must do. The same month last year, year to date and previous year to date measures are still handled automatically. The same $KPI_ variables as shown above the statement edit window can and must be used.

Procedure

If for some reason it is not possible to implement the KPI in either the statement or procedural block then a procedure may be created. The name of such a procedure can be entered in the window at the bottom of the **Procedural Block** tab. If any value is present in this window, then the procedure takes precedence over both the statement and block.

This procedure must adhere to the standard set of parameters as with all other procedures to be executed by the scheduler. It must undertake the same tasks as defined above in the **Procedural Block** section, and as with those tasks it does not need to set the year to date, last year same month and previous year to date values.

Copying KPIs

Right-click the KPI Id and select **Insert Copy** to insert an exact duplicate of the KPI. This copy can then have its Id and Name changed along with its statement to create a new KPI. This is particularly useful when only small differences exist between a number of KPIs.

Testing/Development

During testing it would be normal to DISABLE all KPIs except the one currently being developed. In this way the update or process options will only utilize that KPI. As mentioned above the 'View Statement' option of the KPI popup menu displays, and allows the cutting of, the statement that will be executed to acquire the KPI information. This statement can be tested in SQL*Plus or some other environment if problems are encountered.

Generating the KPI Fact Update Procedure

Once a Detail Fact table has been defined in the metadata and created in the database, an update procedure can be generated to handle the update of the Fact table.

Note

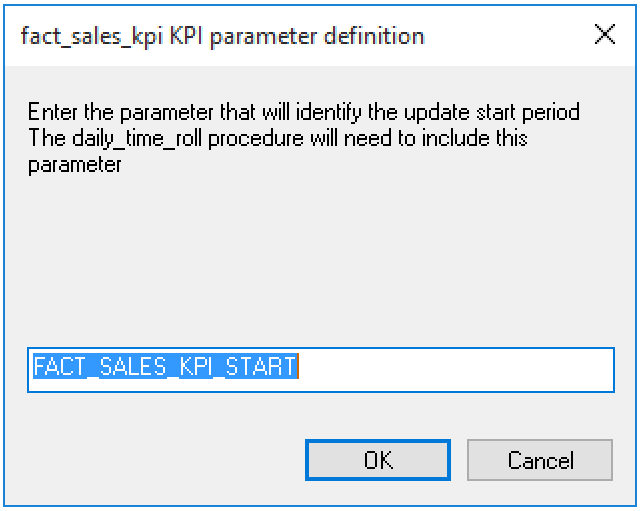
You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.

Generating a Procedure

1. To generate a procedure, right-click the Fact table in the left pane and select **Properties**.
2. Click the **Rebuild** button to start the process of generating the new procedure.
3. A series of prompts are displayed during the procedure generation based on the type of load information.

KPI parameter definition

The first window displayed when generating a Detail Fact table update procedure is the KPI parameter definition dialog:



fact_sales_kpi KPI parameter definition

Enter the parameter that will identify the update start period
The daily_time_roll procedure will need to include this parameter

FACT_SALES_KPI_START

The **KPI parameter definition** window prompts for a parameter to hold the start period that we are interested in. This period sets the first period that is processed when the KPI Fact table is updated. Once entered, click **OK**.

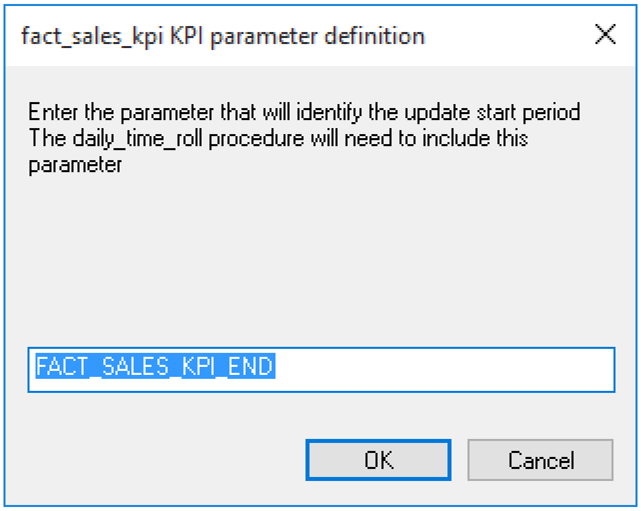
Note

If this parameter is not set, the default current period is set from dim_date. The format is YYYYMM.

Tip

The **dim_date** custom update procedure **daily_date_roll** can be updated to set these parameters automatically if you do not wish to control them manually. This is good practice.

Similarly, a **KPI parameter definition** dialog prompts for a parameter to hold the end period. Once entered, click **OK**.

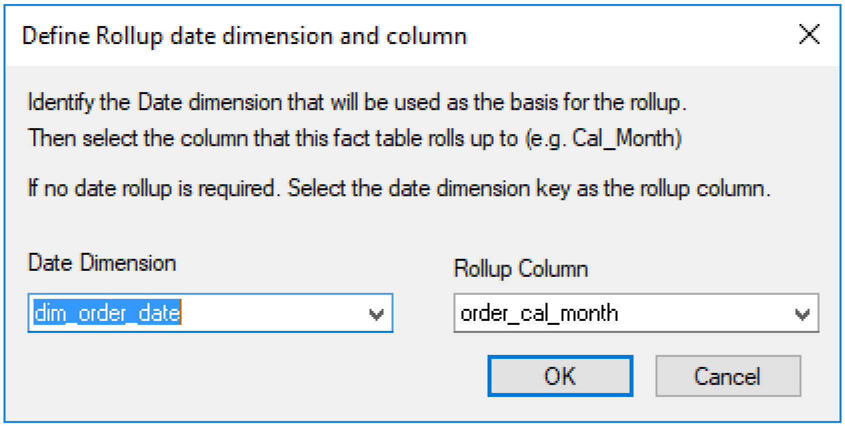

fact_sales_kpi KPI parameter definition

Enter the parameter that will identify the update start period
The daily_time_roll procedure will need to include this parameter

FACT_SALES_KPI_END

Define Rollup date dimension and column

As with Rollup tables, we now need to specify which date in the transactional data we will use to control the rollups and to which level – in this example, for the Monthly level rollup, we choose **dim_order_date** as our **Date Dimension** and **order_cal_month** as our **Rollup Column** and click **OK**.


Define Rollup date dimension and column

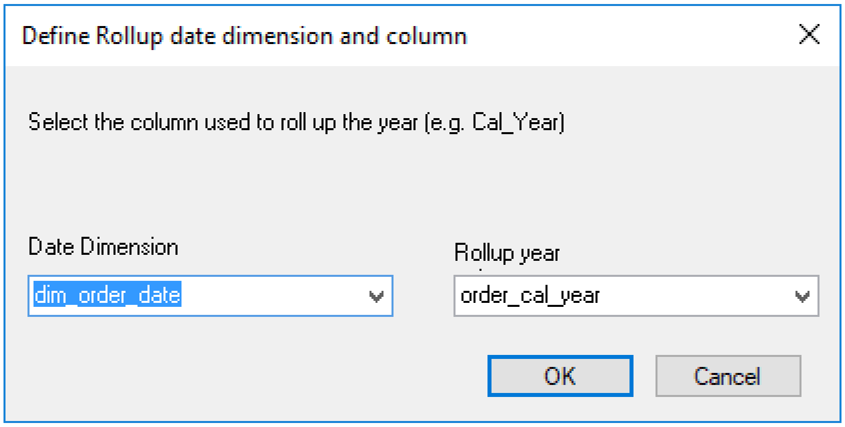
Identify the Date dimension that will be used as the basis for the rollup.
Then select the column that this fact table rolls up to (e.g. Cal_Month)

If no date rollup is required. Select the date dimension key as the rollup column.

Date Dimension: dim_order_date

Rollup Column: order_cal_month

Once again for our Yearly rollup, in this example we choose **dim_order_date** as our **Date Dimension** and **order_cal_year** as our **Rollup Year** and click **OK**.



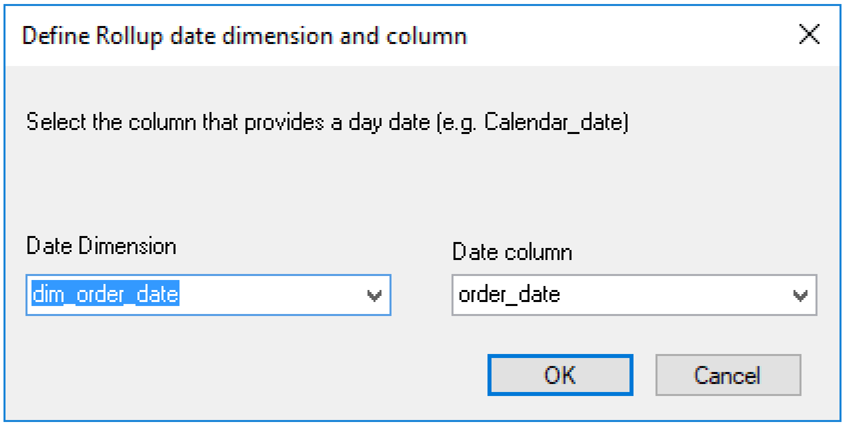
Define Rollup date dimension and column

Select the column used to roll up the year (e.g. Cal_Year)

Date Dimension: dim_order_date

Rollup year: order_cal_year

The next screen sets the Daily level for the update procedure to understand the lower level detail, we choose **dim_order_date** as our **Date Dimension** and **order_date** as our **Date Column** and click **OK**.



Define Rollup date dimension and column

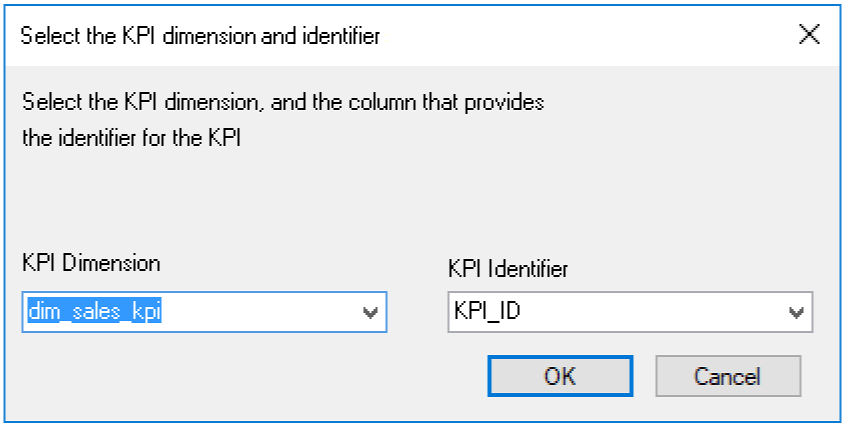
Select the column that provides a day date (e.g. Calendar_date)

Date Dimension: dim_order_date

Date column: order_date

Select the KPI dimension and identifier

Finally we need to specify where the KPIs are defined and how they are defined, so in this example we choose **dim_sales_kpi** as the **KPI Dimension** and **KPI_ID** as the **KPI Identifier**. Click **OK.**



Select the KPI dimension and identifier

Select the KPI dimension, and the column that provides the identifier for the KPI

KPI Dimension: dim_sales_kpi

KPI Identifier: KPI_ID

Snapshot Fact Tables

Snapshot fact tables provide an ‘as at’ view of some part of the business. They always have a time dimension that represents the ‘as at’ date. For example, we may have a weekly or monthly snapshot of our inventory information. Such a snapshot would allow us to see what inventory we had at a particular point in time and allow us to compare our holdings over time. Snapshots are often used when dealing with people profiles or client/customer profiles to enable us to view the state of our relationship with an entity at a given point in time e.g. to see a monthly trend of a client's outstanding balance.

Creating Snapshot Fact Tables

The creation of snapshot fact tables is very dependent on the method used to create the snapshot. In some cases they may be created in exactly the same way as detail fact tables.
These cases are typically where the OLTP application provides the 'as at' state. For example, an inventory system may provide a weekly stock take table which is the basis for our snapshot fact table. In such cases the table is created as per a detail fact table, but with additional tailoring around the time dimension.

In other situations, the snapshot is built from an existing transactional fact table, where all information up to a specific date is summarized or frozen to create the snapshot. In these instances, the snapshot can be created as a rollup fact table.

Regardless of whether a detail or rollup method is used for the creation of the snapshot table, a significant amount of work will be required to tailor the update procedure to reflect the particular requirements of the snapshot table.

Snapshot Fact Columns

The columns for a snapshot Fact table vary to a large degree. A date dimension is always present to reflect the 'as at' date. Some snapshots may have a large number of measures, reflecting $values quantities etc., whereas others may be rich in attribute (non-additive) facts.

Generating the Snapshot Fact Update Procedure

Once a snapshot Fact table has been defined in the metadata and created in the database, an update procedure can be generated to handle the update of the Fact table.

Note

You can also generate an update procedure via a template, refer to **Rebuilding Update Procedures** for details.

Generating a Procedure

1. To generate a procedure, right-click the Fact table in the left pane and select **Properties**.
2. Click the **Rebuild** button to start the process of generating the new procedure.
3. A series of questions is asked during the procedure generation based on the type of load information.
4. If a **detail** method was used for the creation of the snapshot table, refer to **Generating the Detail Fact Update Procedure** for details.
5. However, if a **rollup** method was used for the creation of the snapshot table, refer to **Generating the Rollup Fact Update Procedure** for details.

Slowly Changing Fact Tables

Slowly changing fact tables provide a complete history of changes for some part of the business. Typically, such fact tables are used for situations where a relatively small number of changes occur over time, but we need to track all of these changes.

A good example is an inventory stock holding that does not change very often. In this case, a slowly changing fact table can record both the current inventory levels as well as providing a means of seeing the state of inventory holdings at any point in time.

Creating Slowly Changing Fact Tables

WhereScape RED does not provide any automated way for creating a slowly changing fact table. The best method is to proceed as follows.

1. Drag and drop the Stage table into a dimension target and create a dummy slowly changing dimension with the name we will be using for our fact table. (i.e. fact_...).
2. Answer the questions to the pop-up dialog boxes. Select a join if asked, but do not bother to join any of the tables. Select the columns to be managed as slowly changing. Normally these will be the measures such as quantity and all of the dimension keys. The generated procedure will fail to compile.

3. Inspect the additional columns added for the dimension and make a note of them. Specifically dss_end_date, dss_start_date, dss_current_flag and dss_version.
4. Inspect the indexes created to help manage the slowly changing dimension and make a note of their columns.
5. Delete the dummy dimension you have created.
6. Delete the get_.._key procedure created for the dimension. Do not delete the update procedure.
7. Create a detail fact table in the normal manner.
8. Add in the special columns as created for the dimension. Namely dss_end_date, dss_start_date, dss_current_flag and dss_version.
9. Recreate the table
10. Create indexes as per the dimension table to help maintain the slowly changing fact table.
11. Modify the update procedure for the fact table. Bring up the old version of the procedure that was created for the dimension of the same name. This would have been automatically versioned out when the latest procedure was created and can be seen through the procedure viewer.
12. This new procedure needs to merge the code from the old procedure, which will provide a guide in how to build the code for a slowly changing fact table.

Partitioned Fact Tables

Partitioned fact tables are normally used for performance reasons. They enables you to break a large fact table into a number of smaller tables (partitions).
WhereScape RED supports partitioned fact tables. The following section explains their creation and support.

Creating a Partitioned Fact Table in SQL Server

In WhereScape RED, a partitioned fact table can only be created from an existing fact table. The process therefore is to create a fact table normally, including the update procedure and indexes.

The table is then partitioned through the **Storage** screen of the fact table's Properties window. WhereScape RED assists in creating the exchange table, modifying the indexes and building the procedure to handle the update of the partitions.

Create a detail or rollup/combined fact table

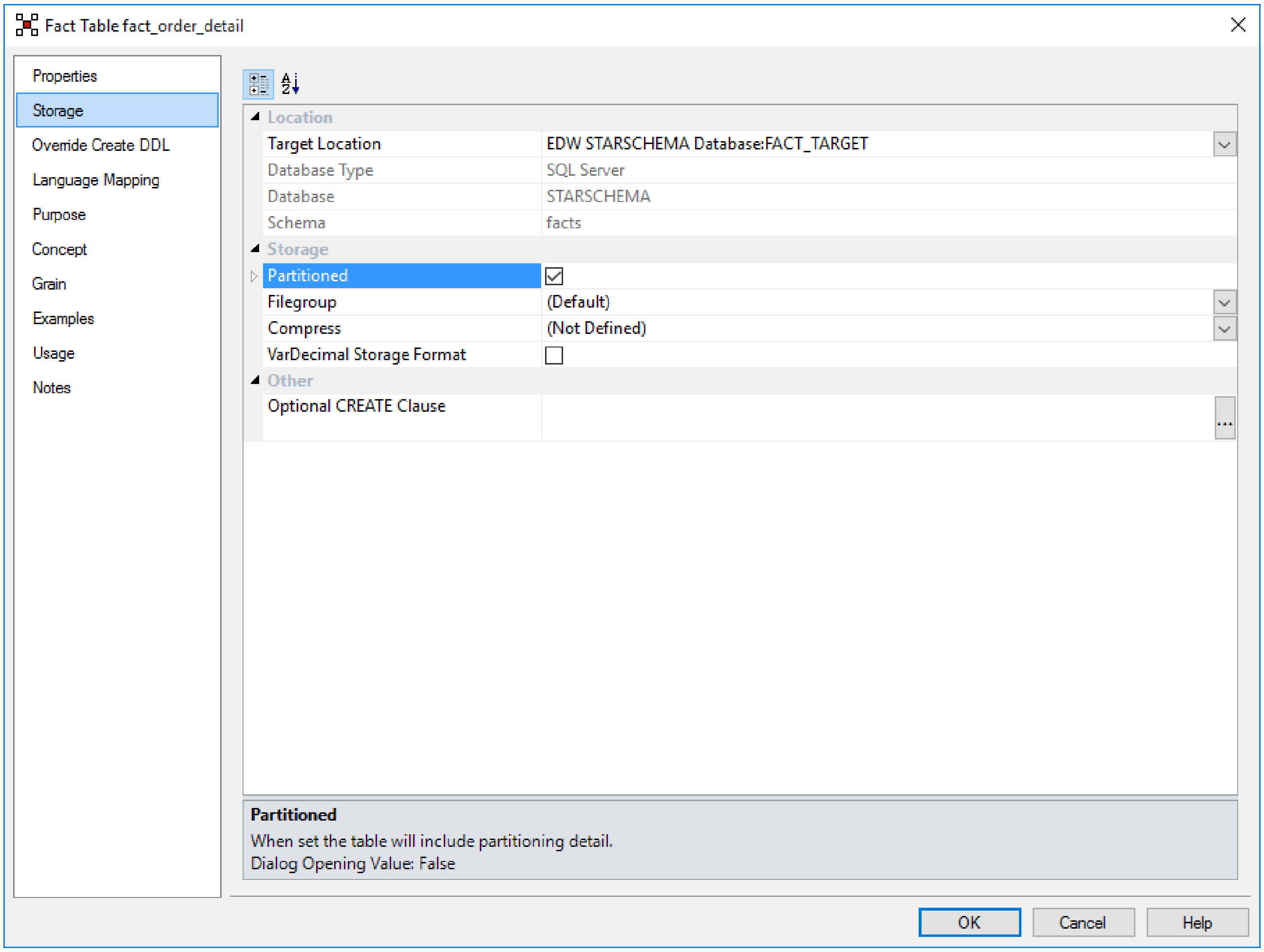
- Create the fact table that you wish to partition in the usual way.
- You must create an update procedure but do not need to load any data. The process of creating the update procedure will also build any indexes for the fact table.
- RED will be using a partition exchange table to move data to and from the partitioned fact table. This exchange table must have the same columns, column order and indexes as the fact table. It is therefore recommended to get the fact table into its final form before partitioning it.
- Add any indexes to avoid extra work later.

Convert to a partitioned fact table

1. Double-click the fact table that you want to partition to bring up its properties screen.
2. Click the **Storage** tab and tick the **Partitioned** check-box (see example in the screens below).
3. A prompt is displayed warning you that this table needs to be recreated to change the current fact to a partitioned table and offering assistance to create the partitioned table. The exchange table is then used to replace the relevant period in the fact table. The fact table remains in a queryable state throughout the process.
4. Click **Yes** to launch the wizard that will assist you through the partitioning of the table.

Notes:

Converting an existing non-partitioned table to a partitioned table cannot be done using a deployment application. To convert non-partitioned to partitioned tables using deployment applications, may require some manual intervention to update the target databases to match the new metadata.



Fact Table fact_order_detail

Location

Target Location	EDW STARSCHEMA Database:FACT_TARGET
Database Type	SQL Server
Database	STARSCHEMA
Schema	facts

Storage

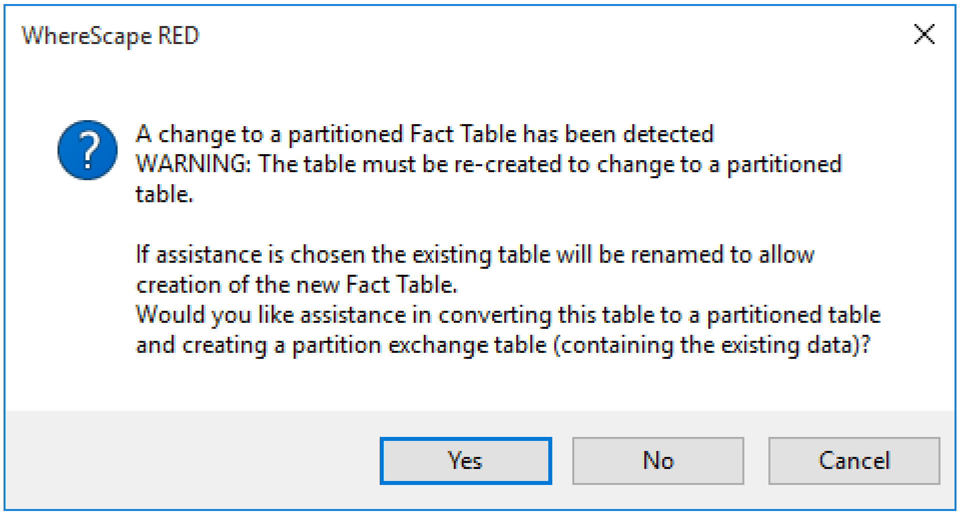
Partitioned	☑
Filegroup	(Default)
Compress	(Not Defined)
VarDecimal Storage Format	□

Other

Optional CREATE Clause	

Partitioned
When set the table will include partitioning detail.
Dialog Opening Value: False

5. As you click the Partitioned check-box, a dialog pop-up is displayed as follows asking for confirmation that the table is to be converted to a partitioned fact table.
If you click **Yes**, the existing fact table is renamed to become the exchange partition table. Any current data will remain in this table.

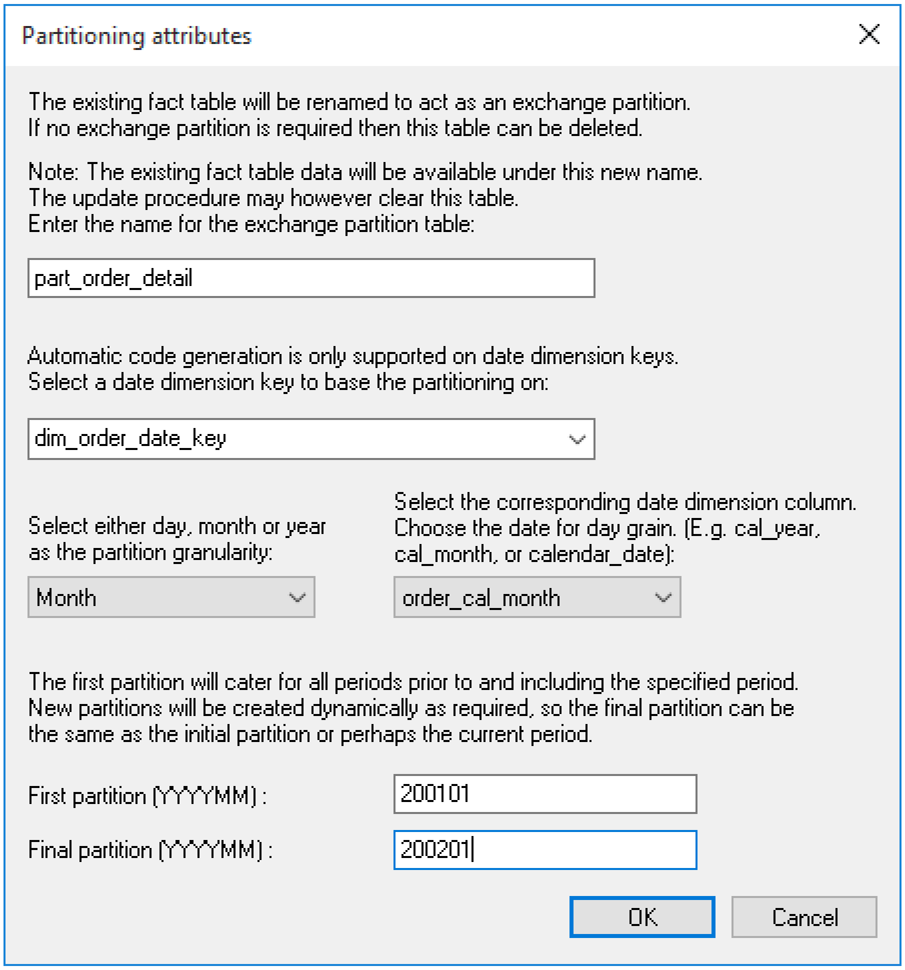


WhereScape RED

A change to a partitioned Fact Table has been detected
WARNING: The table must be re-created to change to a partitioned table.

If assistance is chosen the existing table will be renamed to allow creation of the new Fact Table.
Would you like assistance in converting this table to a partitioned table and creating a partition exchange table (containing the existing data)?

6. The parameters for the partitioned fact table are then prompted for and are described below:



Partitioning attributes

The existing fact table will be renamed to act as an exchange partition.
If no exchange partition is required then this table can be deleted.

Note: The existing fact table data will be available under this new name.
The update procedure may however clear this table.
Enter the name for the exchange partition table:

part_order_detail

Automatic code generation is only supported on date dimension keys.
Select a date dimension key to base the partitioning on:

dim_order_date_key

Select either day, month or year as the partition granularity:

Month

Select the corresponding date dimension column. Choose the date for day grain. (E.g. cal_year, cal_month, or calendar_date):

order_cal_month

The first partition will cater for all periods prior to and including the specified period.
New partitions will be created dynamically as required, so the final partition can be the same as the initial partition or perhaps the current period.

First partition (YYYYMM) : 200101

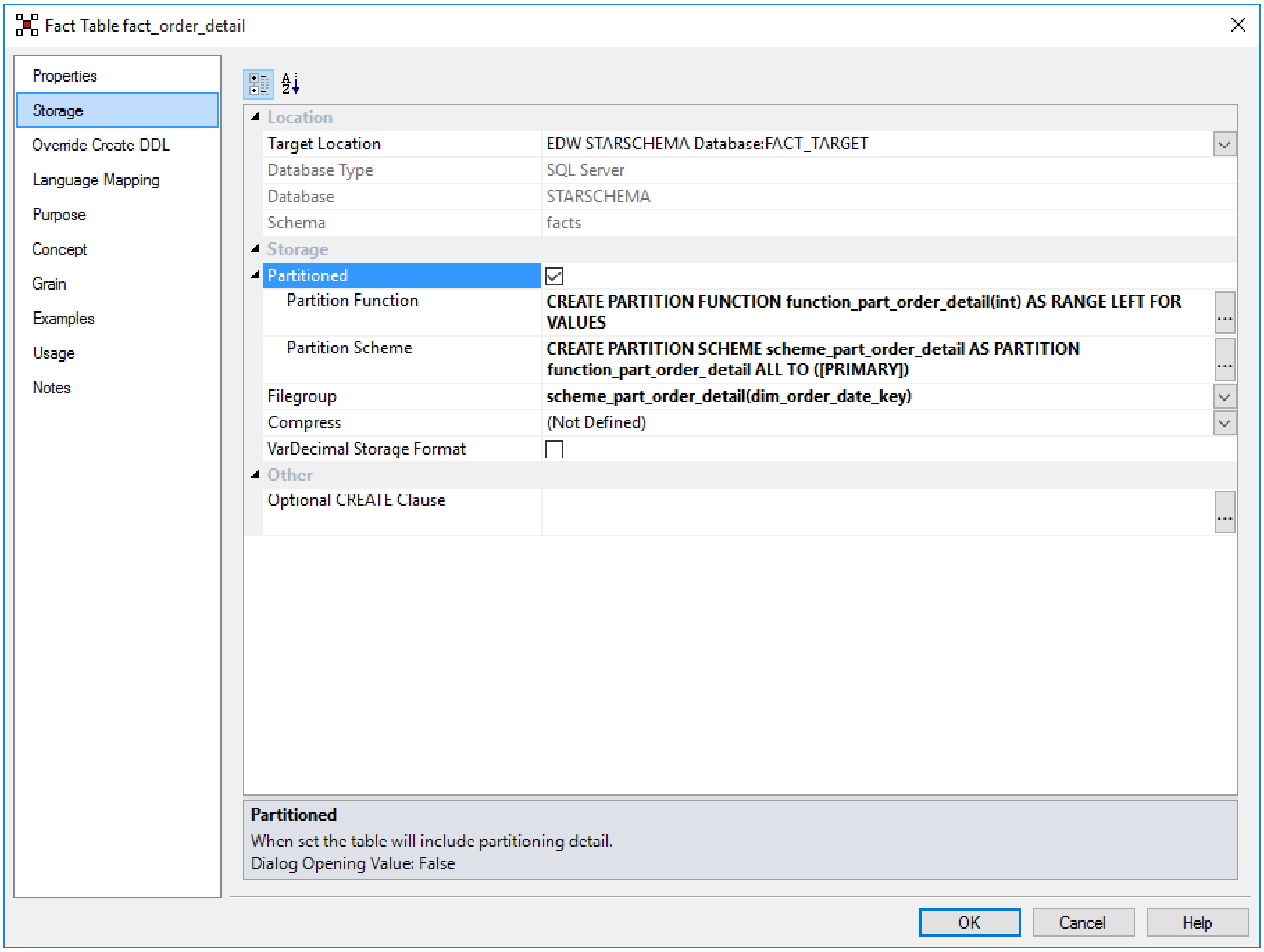
Final partition (YYYYMM) : 200201

7. The partition exchange table name is prompted for. This table is identical to the fact table and is used to swap partitions to and from the fact table. Enter a **name** for the exchange partitioned table.
   - The fact table will be partitioned by the date dimension key. Select a **date dimension key** that will be used as the basis for the partitioning of the fact table. It is assumed that this key has the standard WhereScape format of YYYYMMDD.
   - Select a partition granularity of day, month or year along with the corresponding column in the date dimension.
     - For a **daily partition** a column with a date data type must be chosen. The date dimension key will actually be used for the partitioning but the date column is required.
     - For a **monthly partition** a column with the format YYYYMM must be chosen. In the standard WhereScape date dimension this is the column cal_month.
     - For a **yearly partition** a column with the format YYYY must be chosen. In the standard WhereScape date dimension this is the column cal_year.
   - The first and last partition should then be selected. The last partition is not that important as additional partitions will be created as required. Normally just select a partition a few on from the first. The first partition is however important as any data that is dated prior to the first partition will be loaded into this first partition. The partition must be entered in the format shown. For example monthly partitions require the format YYYYMM.
   - After all the relevant fields have been completed, click **OK** and the conversion process will populate the **Partition Function, Partition Scheme** and **Table Partition Scheme** fields of the table's storage properties.

Note

This field is limited in size, so if too many partitions are chosen only the first few will be added to this field. The other partitions will be created dynamically by the update procedure.

8. Click the **OK** button on the table's **Storage** screen to begin the process of converting the fact table, creating the new fact table and building the new update procedure.
   Examine the results screen to see what has been done.



Fact Table fact_order_detail

Location
- Target Location: EDW STARSCHEMA Database:FACT_TARGET
- Database Type: SQL Server
- Database: STARSCHEMA
- Schema: facts

Storage
- Partitioned: ☑
  - Partition Function: CREATE PARTITION FUNCTION function_part_order_detail(int) AS RANGE LEFT FOR VALUES
  - Partition Scheme: CREATE PARTITION SCHEME scheme_part_order_detail AS PARTITION function_part_order_detail ALL TO ([PRIMARY])
- Filegroup: scheme_part_order_detail(dim_order_date_key)
- Compress: (Not Defined)
- VarDecimal Storage Format: ☐

Other
- Optional CREATE Clause:

Partitioned
When set the table will include partitioning detail.
Dialog Opening Value: False

The exchange partition table will now have any current data, to populate the fact table, copy the data from partition table to the fact table, drop and recreate fact index.
The date dimension key used to base the partitioning on is not allowed to have nulls. WhereScape RED has modified the table property, however the partition table needs to be validated and recreated to be valid.

Then the fact table can be processed.

The partitioned fact table update procedure

The generated procedure will handle the creation of new partitions as required.

Any data for periods prior to the first partition is placed in the first partition. If data is received for a period after the last partition then new partitions are created.

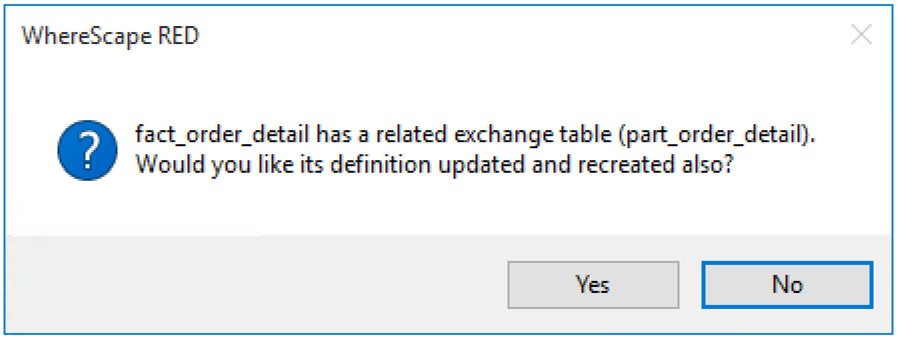
Empty partitions are created if necessary to skip periods where no data is present. For example, if we have a table partitioned by month and the latest partition is 200204 (April/2002). If data is then received for say September 2002 the update procedure will build partitions for May, June, July, August and September.

There is a variable at the start of the update procedure called v_max_skip_periods. This variable is by default set to 12. This defines the maximum number of continuous partitions we will leave empty.

From our previous example if our latest partition was April 2002 and we received data for July 2003 with no interim data then the update procedure will fail as it would have to skip 14 partitions. This check is present to prevent erroneous data from creating hundreds of partitions into the future.

Note

Recreating or Altering a partitioned fact table prompts with an offer to resync and recreate the related exchange table and indexes. This will resync the columns and indexes from the fact table to its exchange partition table so that partition swapping works correctly.



WhereScape RED

fact_order_detail has a related exchange table (part_order_detail).
Would you like its definition updated and recreated also?

Generating the Partitioned Fact Update Procedure

Once a partitioned fact table has been defined in the metadata and created in the database, an update procedure can be generated to handle the update of the fact table.

Note

You can also generate an update procedure via a template, refer to Rebuilding Update Procedures for details.

Generating a Procedure

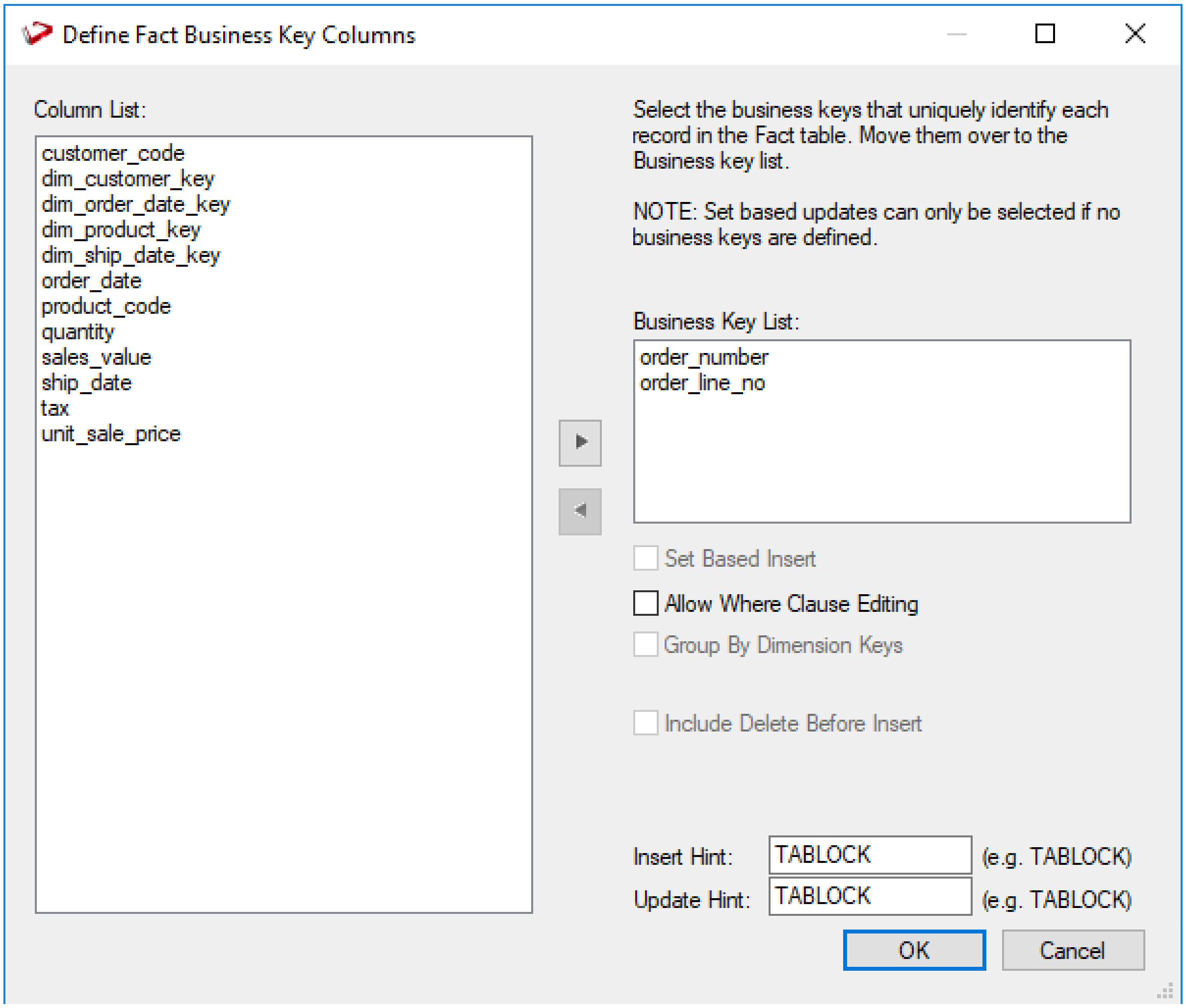
1. To generate a procedure, right-click on the fact table in the left pane and select **Properties**.
2. Click on the **Rebuild** button to start the process of generating the new procedure.
3. A series of prompts are displayed during the procedure generation based on the type of load information.

Define Fact Business Key Columns

The first dialog displayed when generating a partitioned fact table update procedure is the define fact business key columns dialog, asking for the business key that will uniquely identify each fact table record.

The source table from which the fact table is derived would normally have some form of unique constraint applied. In most cases this will be the business key.

In the example below the order_id and order_line_no are selected for the business key list.



Select the business keys that uniquely identify each record in the Fact table. Move them over to the Business key list.

NOTE: Set based updates can only be selected if no business keys are defined.

□ Set Based Insert

□ Allow Where Clause Editing

□ Group By Dimension Keys

□ Include Delete Before Insert

Insert Hint: TABLOCK (e.g. TABLOCK)

Update Hint: TABLOCK (e.g. TABLOCK)

A business key can be made up of multiple columns, but it must provide a unique identifier. Where multiple columns separately uniquely identify rows in the fact table, choose one to act as the primary business key.

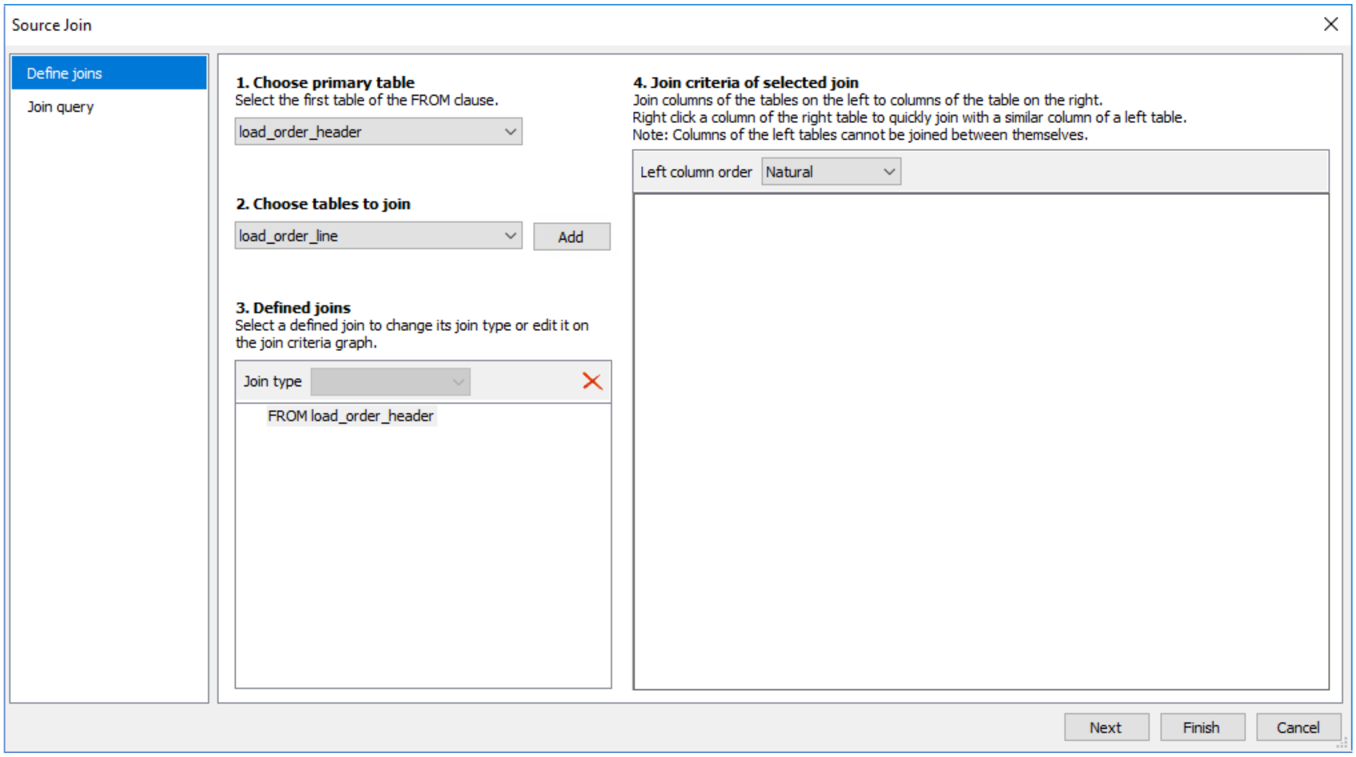
For example a source table may have a unique constraint on both a product code and a product description. Therefore the description as well as the code must be unique.

None of the columns chosen as the business key should ever contain a **NULL** value.

Fields	Description
Allow Where Clause Editing	If the **Allow Where Clause Editing** option is selected, then the next dialog displayed is the **Source Join** wizard, otherwise this is skipped.
Insert Hint	SQL Server only. Enter a database hint to be used in the INSERT statement.
Update Hint	SQL Server only. Enter a database hint to be used in the UPDATE statement.

Source Join Wizard

This wizard is used to join source tables and add 'Where' clauses.



1. Choose primary table
Select the first table of the FROM clause.
load_order_header

2. Choose tables to join
load_order_line

3. Defined joins
Select a defined join to change its join type or edit it on the join criteria graph.
FROM load_order_header

4. Join criteria of selected join
Join columns of the tables on the left to columns of the table on the right.
Right click a column of the right table to quickly join with a similar column of a left table.
Note: Columns of the left tables cannot be joined between themselves.

Source table joins should have been performed in the stage table, refer to **Generating the Staging Update Procedure** for details.

Since most of the real work is done in the stage table, this is all that is needed to build the Partitioned Fact Update Procedure.

Fact Table Column Properties

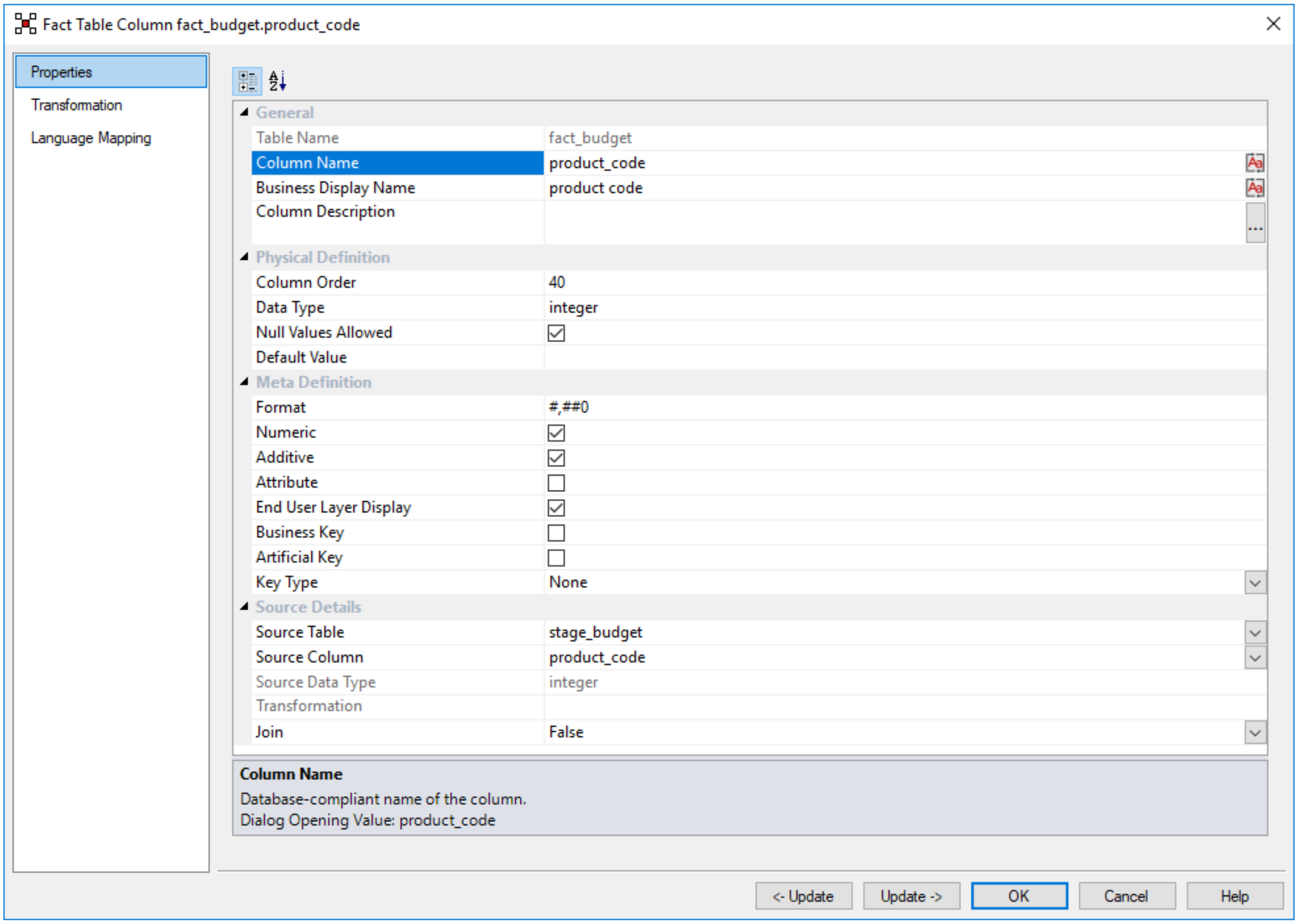
Each fact table column has a set of associated properties. The definition of each property is defined below:

- If the **Column name** or **Data type** is changed for a column then the metadata will differ from the table as recorded in the database.
- Use the **Validate > Validate Table Create Status** menu option to compare the metadata to the table in the database.
  A context menu option **Alter Table** is available when your right click the table name, after the validate has completed. This option will alter the database table to match the metadata definition.

Tip

If a database table's definition is changed in the metadata, then the table needs to be altered in the database. Use the **Validate > Validate Table Create Status** to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table, through a pop-up menu option from the validated table name.

A sample **Properties** screen is as follows:



General	
Table Name	fact_budget
Column Name	product_code
Business Display Name	product code
Column Description	
Physical Definition	
Column Order	40
Data Type	integer
Null Values Allowed	☑
Default Value	
Meta Definition	
Format	#,##0
Numeric	☑
Additive	☑
Attribute	□
End User Layer Display	☑
Business Key	□
Artificial Key	□
Key Type	None
Source Details	
Source Table	stage_budget
Source Column	product_code
Source Data Type	integer
Transformation	
Join	False

Column Name
Database-compliant name of the column.
Dialog Opening Value: product_code

Tip

The two special update keys allow you to update the column and step either forward or backward to the next column's properties.
ALT-Left Arrow and **ALT-Right Arrow** can also be used instead of the two special update keys..

Fields	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Business Display Name / Column Title	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.

Fields	Description
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example, if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col .** This field is also stored as a comment against the column in the database.
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. See the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.
Numeric	Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view ws_admin_v_dim_col which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Additive	Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view ws_admin_v_dim_col which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Attribute	Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or a invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables. This checkbox does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
End User Layer Display	Indicates whether the table column is available/visible to end users. If set the documentation will include the column in the glossary and in the user documentation. It is also used to decide what columns appear in the view ws_admin_v_dim_col. Typically columns such as the artificial key would not be enabled for end user display.

Business Key	Indicates whether the column is part of the BUSINESS key, which is defined during the update procedure build. [Normally maintained automatically]. Multiple columns can form the primary business key.

<table>
<tr><td><b>Key Type</b></td><td>Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested. The supported values are:
<table>
<tr><th>Key type</th><th>Meaning</th></tr>
<tr><td><b>0</b></td><td>The artificial key. Set when the key is added during drag and drop table generation.</td></tr>
<tr><td><b>1</b></td><td>Component of all business keys. Indicates that this column is used as part of any business key.</td></tr>
<tr><td><b>A</b></td><td>Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation.</td></tr>
<tr><td><b>B-Z</b></td><td>Indicates that the column is part of a secondary business key. Only used during index generation and not normally set.</td></tr>
</table>
</td></tr>
</table>

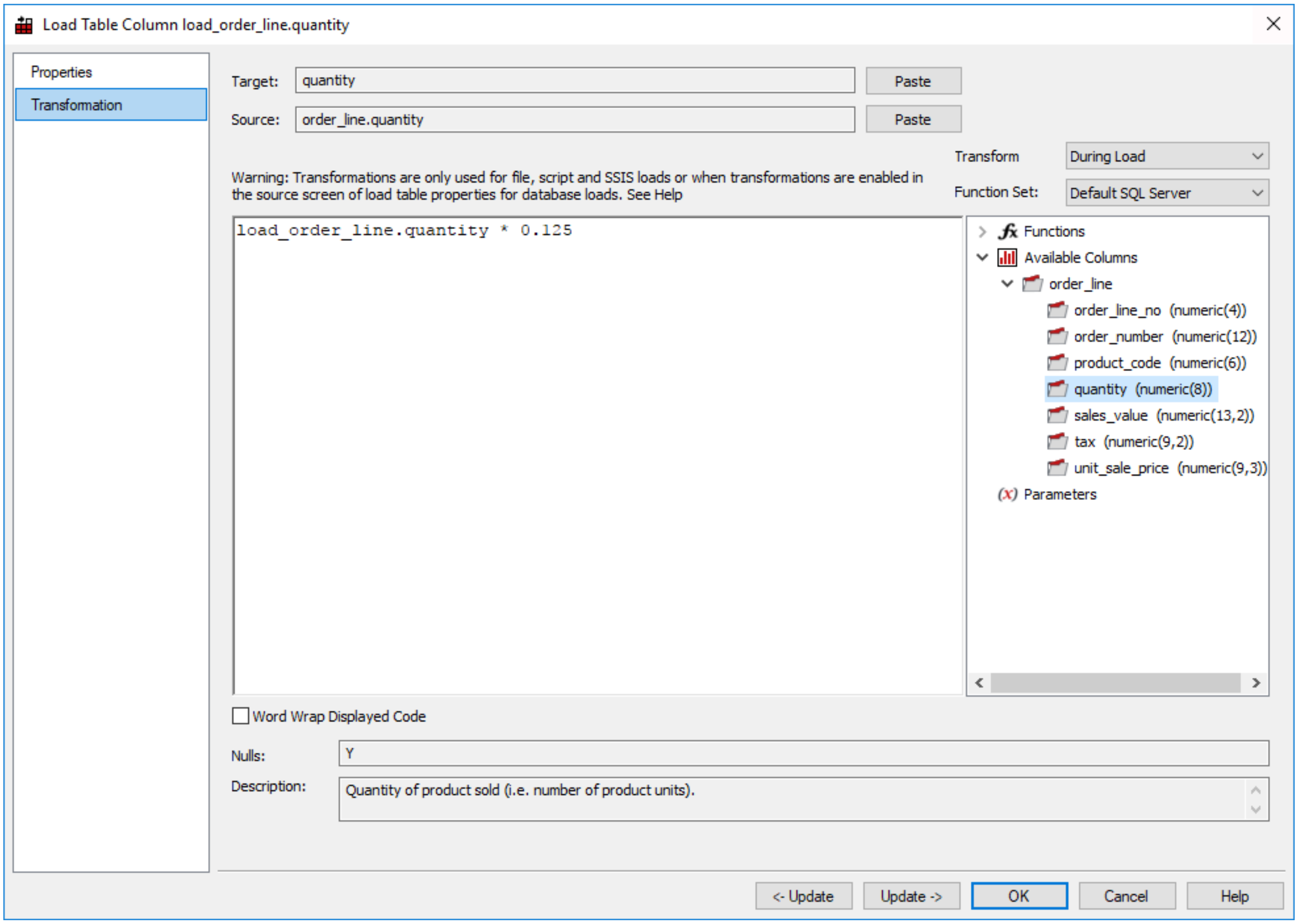
KPI Column Type	Only used by **KPI** fact tables. This field defines the column type for the KPI Fact Table. Refer to the KPI table creation section for more details on this field.
Source Table	Identifies the source table where the column's data comes from. This source table is normally a stage table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the Source strategy field. This field is used when generating a procedure to update the fact table. It is also used in the track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a stage table column, which in turn may have been a transformation or the combination of multiple columns. This may also be a dimensional key where a dimension is being joined.
Source Data Type	Identifies the source column's data type. [Read-only].
Transformation	Refer to **Fact Table Column Transformations**. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source table** and **Source column** fields will provide the dimension table's side of the join. The options for this field are: False, True, Manual and Pre Join. • Setting this field to Manual changes the way the dimension table is looked up during the update procedure build. It allows you to join the dimension manually in the Cursor mapping dialog (where the 'Where' clause is built). • Setting this field to Pre Join activates the **Pre Join Source Table** field and allows you to select a table from the drop-down list.
Pre Join Source Table	Indicates the table from which the pre joined column was sourced. When the Join option is set to False, this field becomes inactive. When the Join option is set to True or Manual, this field is set to the current table name. When the Join option is set to Pre Join, then you can select the required table from the drop-down list.

Fact Table Column Transformations

Each fact table column can have a transformation associated with it. The transformation is included in the generated procedure and executed as part of the procedure update.
The transformation must therefore be a valid SQL construct that can be included in a **SELECT** statement.

For example, we could have a transformation of 'load_order_line.quantity * 0.125' to calculate a tax column of 12.5%.

Click the **Transformation** tab to enter a transformation.



Load Table Column load_order_line.quantity

Target: quantity

Source: order_line.quantity

Transform: During Load

Function Set: Default SQL Server

Warning: Transformations are only used for file, script and SSIS loads or when transformations are enabled in the source screen of load table properties for database loads. See Help

```
load_order_line.quantity * 0.125
```

- Functions
- Available Columns
  - order_line
    - order_line_no (numeric(4))
    - order_number (numeric(12))
    - product_code (numeric(6))
    - quantity (numeric(8))
    - sales_value (numeric(13,2))
    - tax (numeric(9,2))
    - unit_sale_price (numeric(9,3))
- (x) Parameters

☐ Word Wrap Displayed Code

Nulls: Y

Description: Quantity of product sold (i.e. number of product units).

Notes

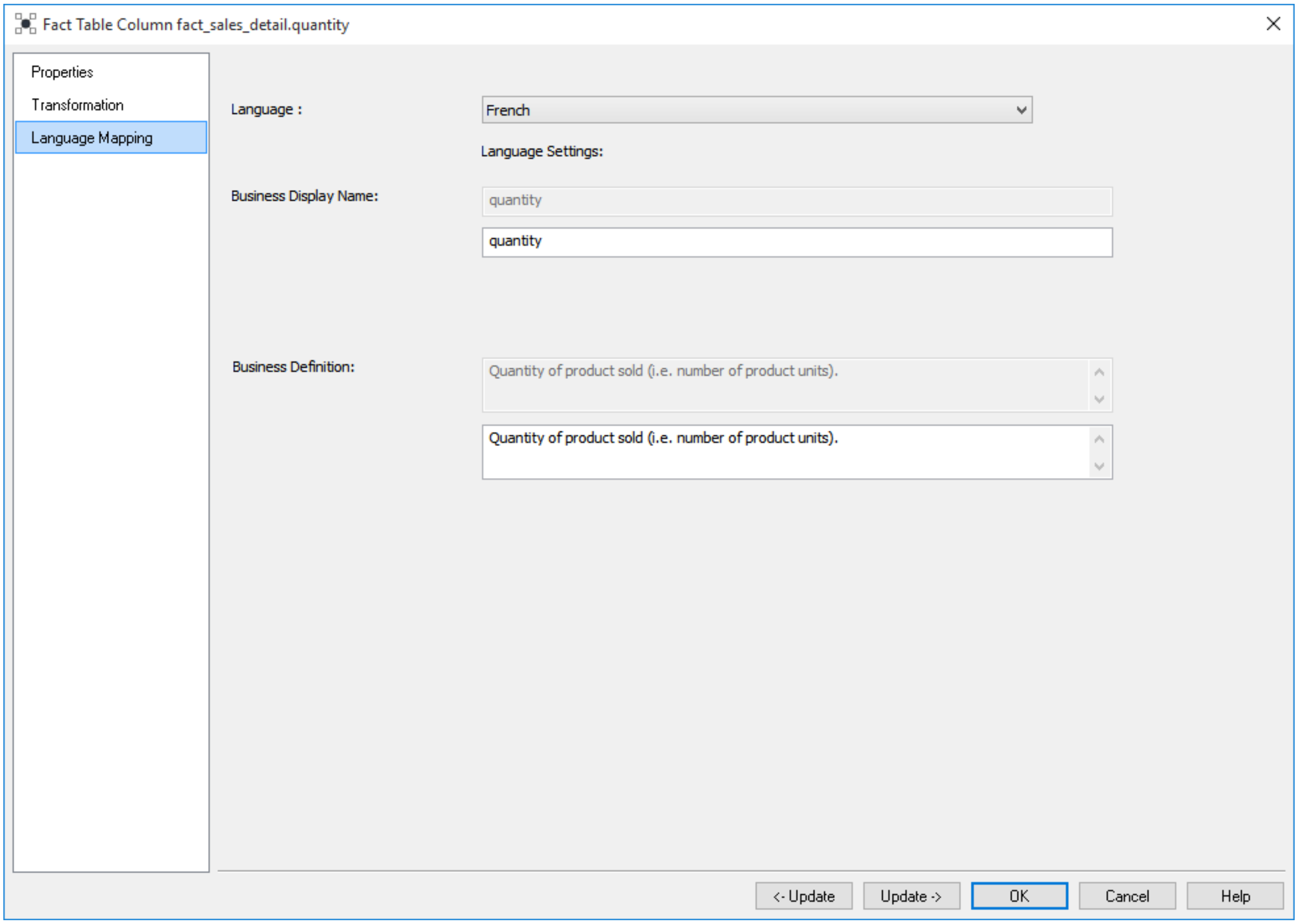
- Transformations are only put into effect when the procedure is re-generated.
- **Microsoft Analysis Services 2005+ Tabular Mode Tables:** For Tabular Mode table column transformations, **Default DAX** is the only applicable Function Set for **after load** transformations.

Refer to **Transformations** for details.

Fact Table Language Mapping

The Fact table Properties screen has a tab called **Language Mapping**.

Select the language from the drop-down list and then enter the translations for the **Business Display Name** and the **Description** in the chosen language.
The translations for these fields can then be pushed through into OLAP cubes.



Language : French

Language Settings:

Business Display Name: quantity

quantity

Business Definition: Quantity of product sold (i.e. number of product units).

Quantity of product sold (i.e. number of product units).

Aggregation

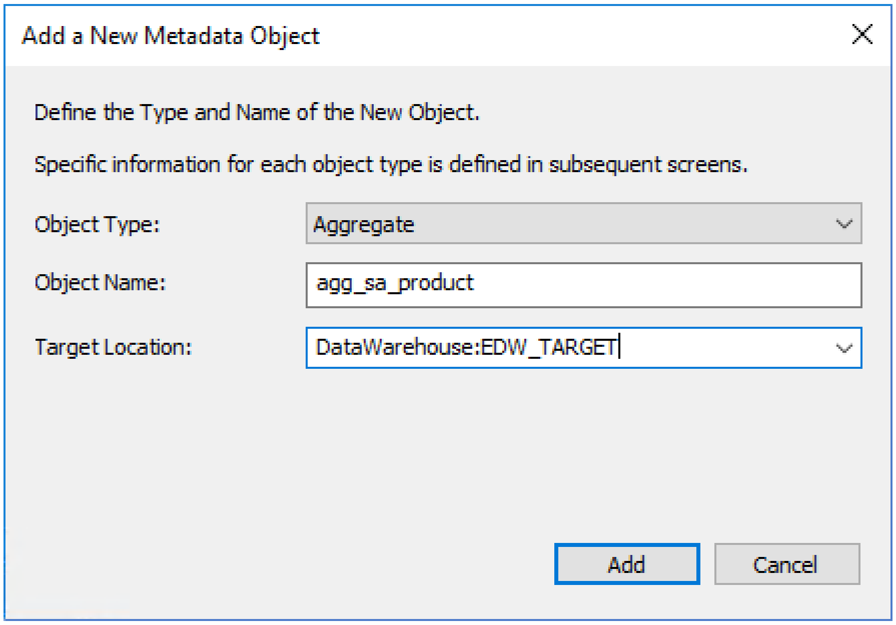
Two types of aggregate tables are discussed.

The first is where all non-additive facts and one or more dimensions are removed from a fact table. Typically this results in a smaller table that can answer a subset of the queries that could be posed against the fact table. This aggregate table still maintains full integrity to the remaining dimensions, and consequently reflects all changes to those dimensions.

The second type, we will call an aggregate summary, or summary table. This table includes additive measures and in some cases hierarchical elements of one or more of the dimensions providing a rolled-up summary of the fact table data. For example we may choose to deal at product group level rather than product SKU which is the granularity of the dimension.

Creating an Aggregate Table

1. In the left pane double-click on the **aggregate** group to list the aggregates in the middle pane and set aggregates as the drop target.
2. From the Data Warehouse browse (right) pane drag a fact table into the middle pane and enter the aggregate name. Click **ADD**.



Add a New Metadata Object

Define the Type and Name of the New Object.

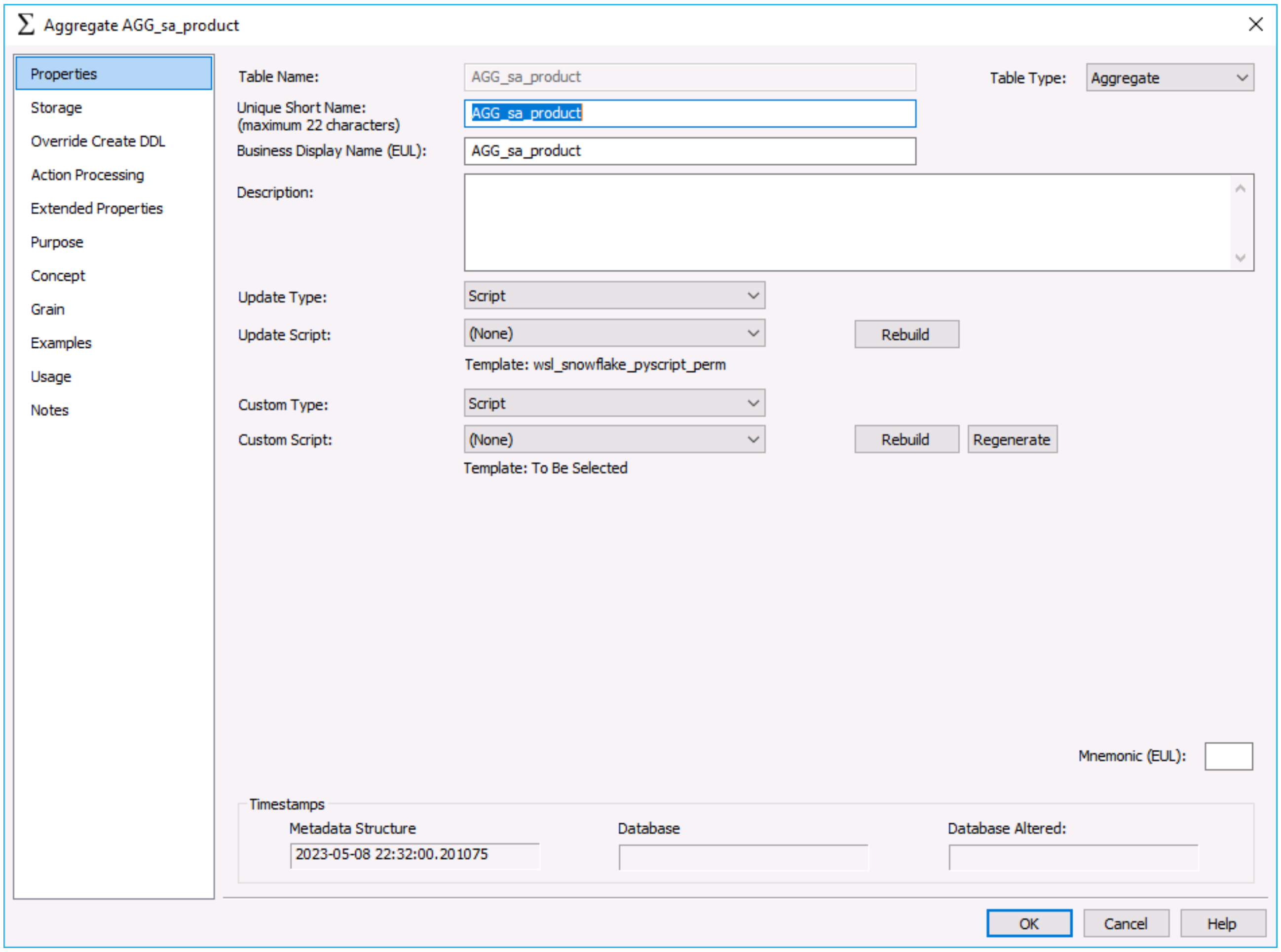
Specific information for each object type is defined in subsequent screens.

Object Type: Aggregate

Object Name: agg_sa_product

Target Location: DataWarehouse:EDW_TARGET

3. The aggregate properties are displayed. Click **OK**.



Aggregate AGG_sa_product

Table Name: AGG_sa_product

Table Type: Aggregate

Unique Short Name: (maximum 22 characters) AGG_sa_product

Business Display Name (EUL): AGG_sa_product

Description:

Update Type: Script

Update Script: (None)

Template: wsl_snowflake_pyscript_perm

Custom Type: Script

Custom Script: (None)

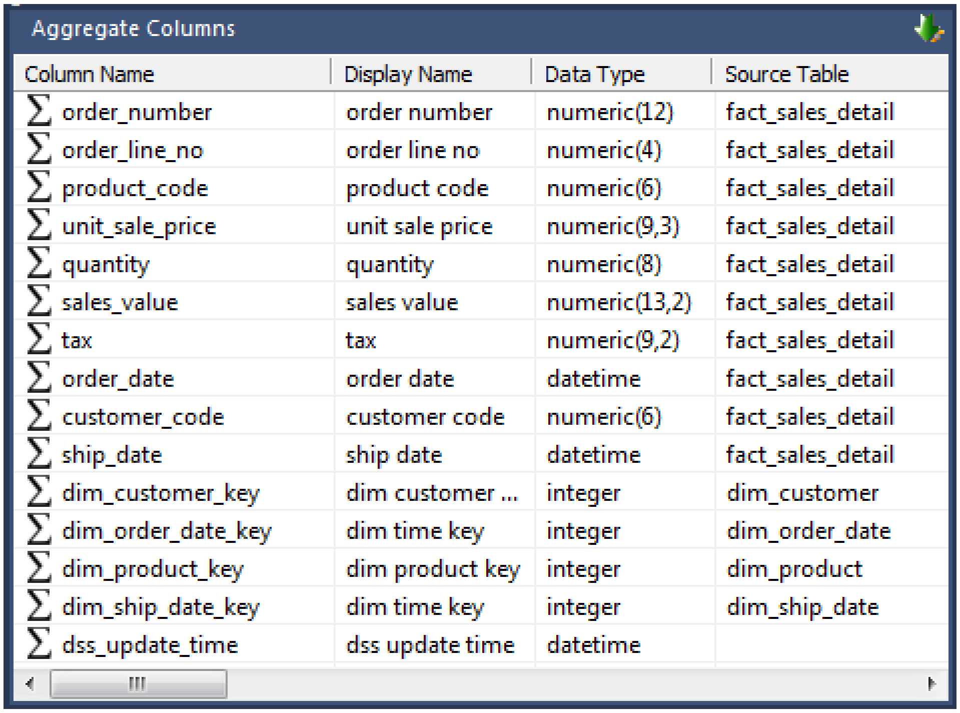
Template: To Be Selected

Mnemonic (EUL):

Timestamps

Metadata Structure	Database	Database Altered:
2023-05-08 22:32:00.201075		

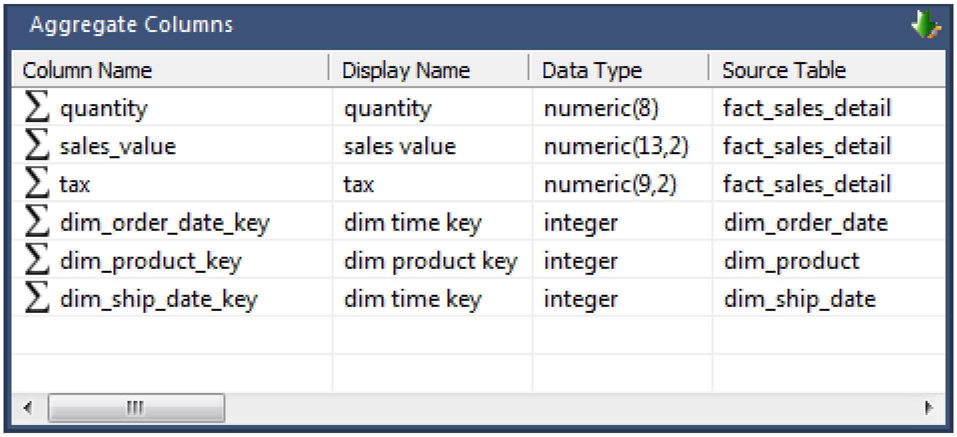
4. The list of columns in the Aggregate is displayed in the middle pane:



Aggregate Columns

Column Name	Display Name	Data Type	Source Table
order_number	order number	numeric(12)	fact_sales_detail
order_line_no	order line no	numeric(4)	fact_sales_detail
product_code	product code	numeric(6)	fact_sales_detail
unit_sale_price	unit sale price	numeric(9,3)	fact_sales_detail
quantity	quantity	numeric(8)	fact_sales_detail
sales_value	sales value	numeric(13,2)	fact_sales_detail
tax	tax	numeric(9,2)	fact_sales_detail
order_date	order date	datetime	fact_sales_detail
customer_code	customer code	numeric(6)	fact_sales_detail
ship_date	ship date	datetime	fact_sales_detail
dim_customer_key	dim customer ...	integer	dim_customer
dim_order_date_key	dim time key	integer	dim_order_date
dim_product_key	dim product key	integer	dim_product
dim_ship_date_key	dim time key	integer	dim_ship_date
dss_update_time	dss update time	datetime	

5. Remove any columns that will not make sense at an aggregated level. For example, dss_fact_table_key, any business keys, any non-additive facts, any measures that relate to detail (e.g. unit price) and any dimension keys not required:



Column Name	Display Name	Data Type	Source Table
∑ quantity	quantity	numeric(8)	fact_sales_detail
∑ sales_value	sales value	numeric(13,2)	fact_sales_detail
∑ tax	tax	numeric(9,2)	fact_sales_detail
∑ dim_order_date_key	dim time key	integer	dim_order_date
∑ dim_product_key	dim product key	integer	dim_product
∑ dim_ship_date_key	dim time key	integer	dim_ship_date

Note

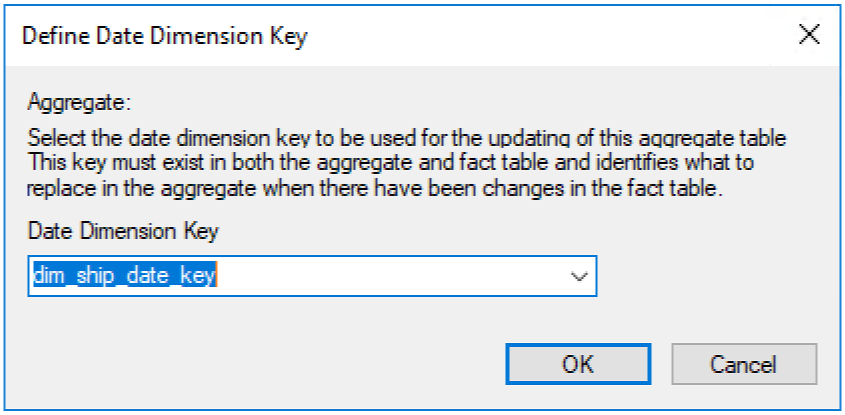
All aggregate table columns (other than dss columns) should always have a source table specified, even if a column transformation is in use.

6. Create the aggregate table in the database by right-clicking on the aggregate and selecting **Create(ReCreate)**.
7. Create a procedure to update the aggregate by right-clicking on the aggregate, selecting **Properties** and selecting **(Build Procedure...)** in the Update Procedure field.

Note

WhereScape RED generated update procedures for aggregates are incremental. Incremental updates are based on a date dimension key and number of look back days. The aggregate update process looks at any records that have been updated in the fact table in the last 7 days (by default)..

To support this, a date dimension key must be selected. The columns chosen must be in both the source fact table and the aggregate. Select this column and click **OK**.


Define Date Dimension Key

Aggregate:

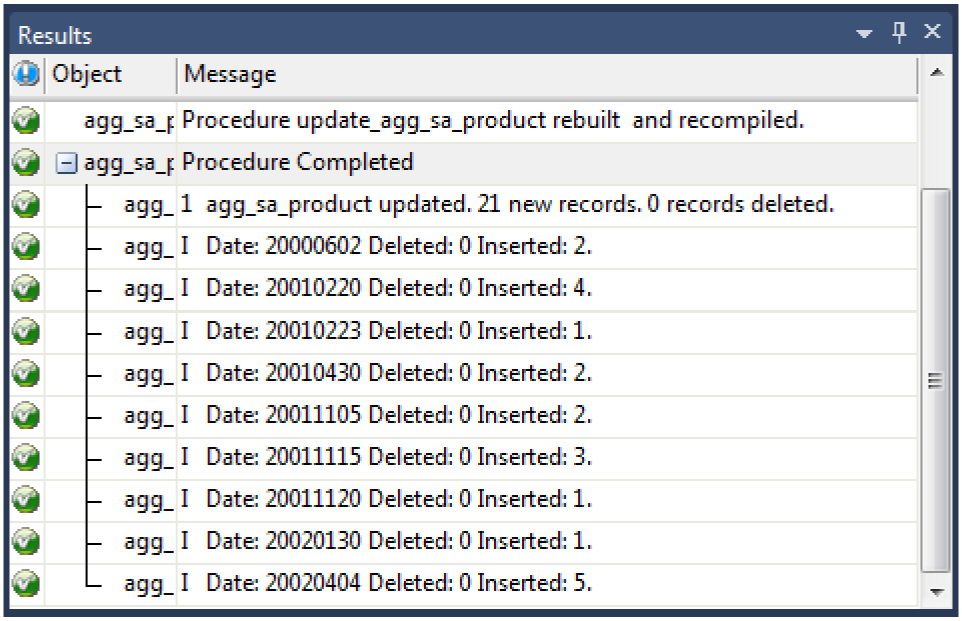
Select the date dimension key to be used for the updating of this aggregate table This key must exist in both the aggregate and fact table and identifies what to replace in the aggregate when there have been changes in the fact table.

Date Dimension Key: dim_ship_date_key

8. Update the table by right-clicking and choosing **Execute Update Procedure.**

Note

Any column transformations added to a measure column of an aggregate table must always include an aggregate function, usually **SUM**. For example, an ISNULL added to forecast_quantity should be entered as: SUM(ISNULL(forecast_quantity,0)).



	Object	Message
✓	agg_sa_p	Procedure update_agg_sa_product rebuilt and recompiled.
✓	⊟ agg_sa_p	Procedure Completed
✓	agg_	1 agg_sa_product updated. 21 new records. 0 records deleted.
✓	agg_	I Date: 20000602 Deleted: 0 Inserted: 2.
✓	agg_	I Date: 20010220 Deleted: 0 Inserted: 4.
✓	agg_	I Date: 20010223 Deleted: 0 Inserted: 1.
✓	agg_	I Date: 20010430 Deleted: 0 Inserted: 2.
✓	agg_	I Date: 20011105 Deleted: 0 Inserted: 2.
✓	agg_	I Date: 20011115 Deleted: 0 Inserted: 3.
✓	agg_	I Date: 20011120 Deleted: 0 Inserted: 1.
✓	agg_	I Date: 20020130 Deleted: 0 Inserted: 1.
✓	agg_	I Date: 20020404 Deleted: 0 Inserted: 5.

- **DB2:** When the procedure runs it first removes the materialized query table, then updates any changes from the fact table into the aggregate. It then re-establishes the materialized query table, enabling query rewrite.

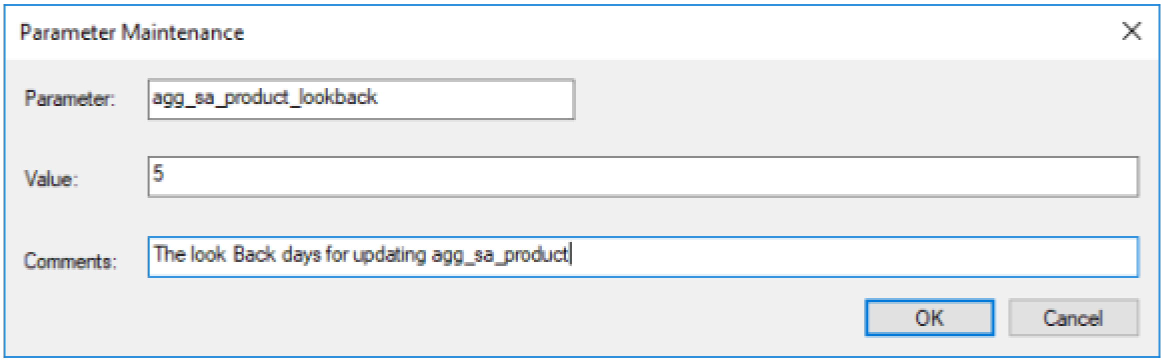
Change an Aggregate's Default Look Back Days

WhereScape RED generate update procedures for aggregates that are incremental. Incremental updates are based on a date dimension key and number of look back days.

The aggregate update process looks at any records that have been updated in the fact table in the last number of look back days.

The default number of look back days is 7. The default is set in each update procedure. To change the number of look back days for an aggregate table, create a WhereScape RED parameter called AggregateName_lookback and enter the required number of look back days as the parameter value.

For example, add the following parameter to change the look back days to 5 for agg_sa_product:



Parameter: agg_sa_product_lookback

Value: 5

Comments: The look Back days for updating agg_sa_product

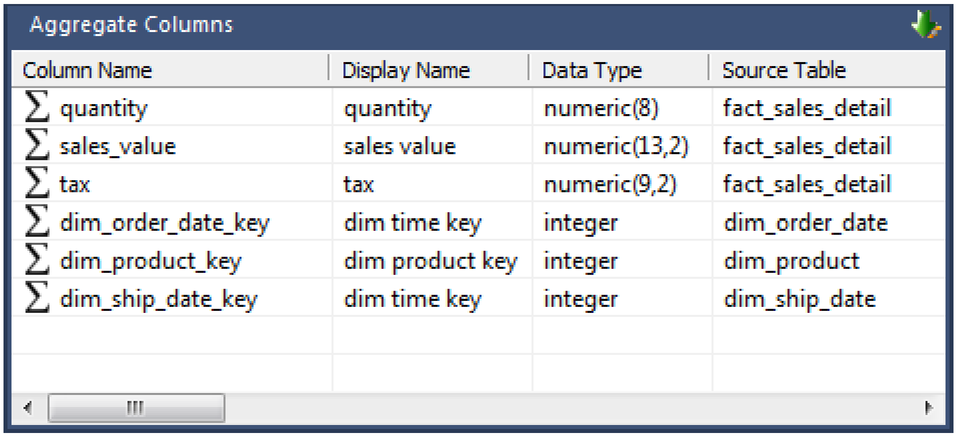
Refer to **Parameters** for more information on WhereScape RED parameters.

Creating an Aggregate Summary Table

The creation of a summary table proceeds initially in the same way as an aggregate table.

1. In the left pane double-click the **Aggregate** group to list the aggregates in the middle pane and set aggregates as the drop target.

2. From the Data Warehouse browse (right) pane drag a fact table into the middle pane. Remove any columns that will not make sense at an aggregated level. For example, dss_fact_table_key, any business keys, any non-additive facts, any measures that relate to detail (e.g. unit price).
3. Drag over columns from dimensions that are linked to the fact table. Delete the dimension keys to allow a rollup to the level of the dimension elements.



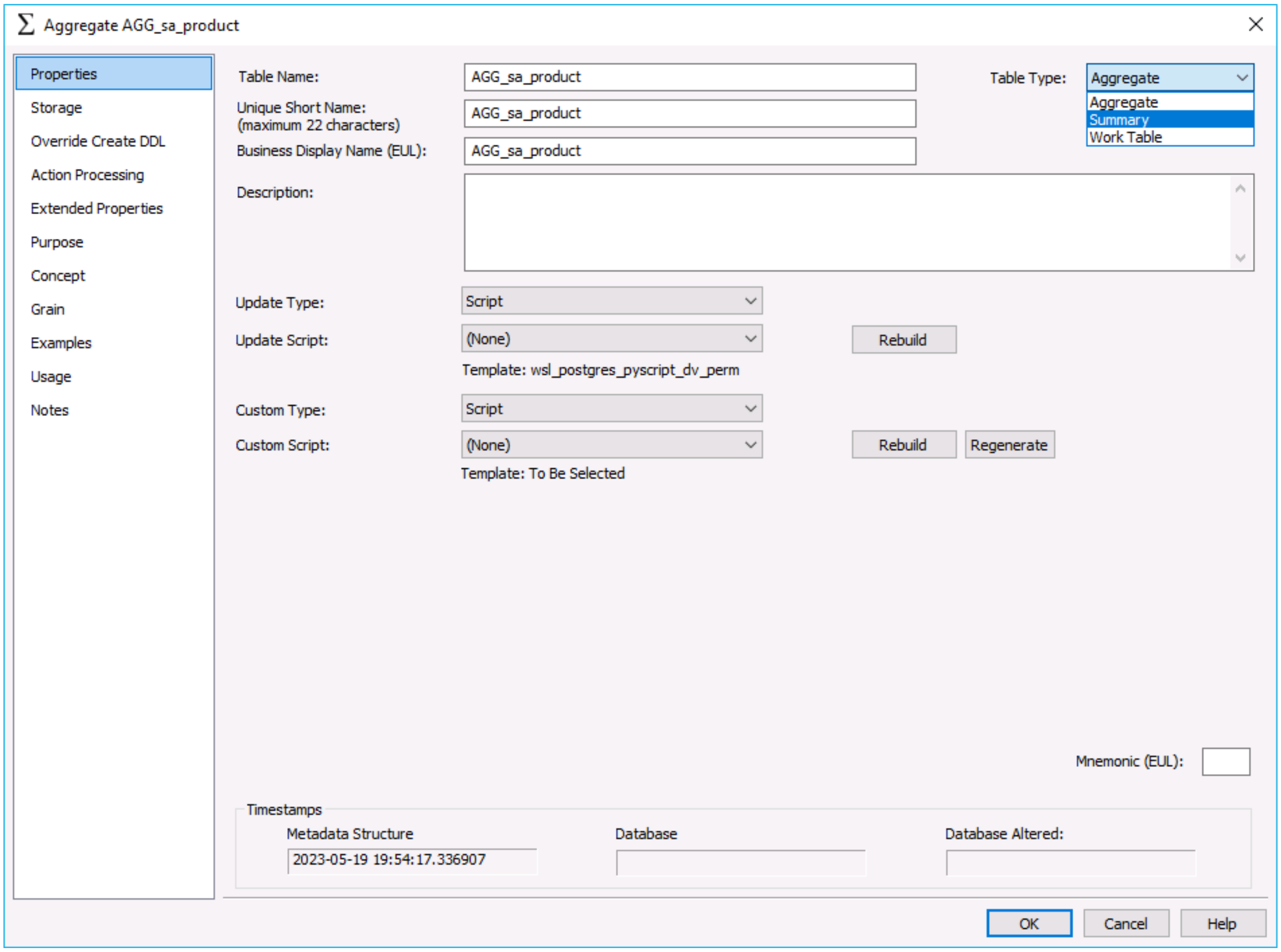
Aggregate Columns

Column Name	Display Name	Data Type	Source Table
∑ quantity	quantity	numeric(8)	fact_sales_detail
∑ sales_value	sales value	numeric(13,2)	fact_sales_detail
∑ tax	tax	numeric(9,2)	fact_sales_detail
∑ dim_order_date_key	dim time key	integer	dim_order_date
∑ dim_product_key	dim product key	integer	dim_product
∑ dim_ship_date_key	dim time key	integer	dim_ship_date

Note

All aggregate summary table columns (other than dss columns) should always have a source table specified, even if a column transformation is in use.

4. In the **Properties** of the aggregate table change the **Table Type** to **Summary**:



∑ Aggregate AGG_sa_product

Table Name: AGG_sa_product

Unique Short Name: (maximum 22 characters) AGG_sa_product

Business Display Name (EUL): AGG_sa_product

Table Type: Aggregate

Description:

Update Type: Script

Update Script: (None)

Template: wsl_postgres_pyscript_dv_perm

Custom Type: Script

Custom Script: (None)

Template: To Be Selected

Mnemonic (EUL):

Timestamps

Metadata Structure: 2023-05-19 19:54:17.336907

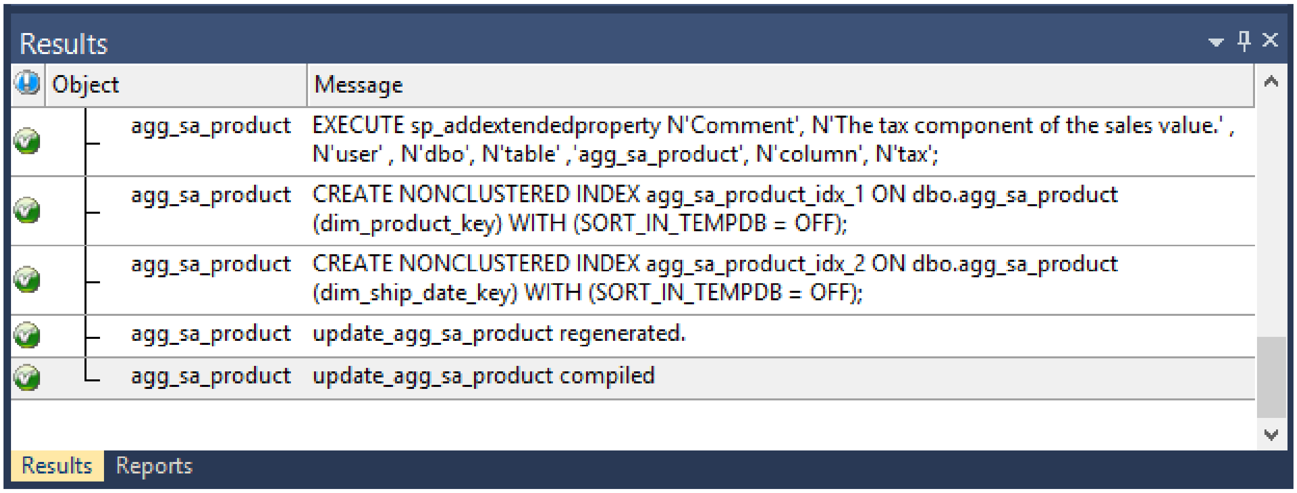
Database:

Database Altered:

5. Create the aggregate summary table in the database by right-clicking on the aggregate and selecting **Create(ReCreate)**.
6. Create a procedure to update the aggregate summary by right-clicking the aggregate, selecting **Properties** and selecting **(Build Procedure...)** in the **Update Procedure** field. The aggregate summary table is totally rebuilt each time the procedure is executed.

Note

Any column transformations added to a measure column of an aggregate summary table must always include an aggregate function, usually **SUM**. For example, an ISNULL added to forecast_quantity should be entered as: SUM(ISNULL(forecast_quantity,0))..



	Object	Message
✓	agg_sa_product	EXECUTE sp_addextendedproperty N'Comment', N'The tax component of the sales value.' , N'user' , N'dbo', N'table' ,'agg_sa_product', N'column', N'tax';
✓	agg_sa_product	CREATE NONCLUSTERED INDEX agg_sa_product_idx_1 ON dbo.agg_sa_product (dim_product_key) WITH (SORT_IN_TEMPDB = OFF);
✓	agg_sa_product	CREATE NONCLUSTERED INDEX agg_sa_product_idx_2 ON dbo.agg_sa_product (dim_ship_date_key) WITH (SORT_IN_TEMPDB = OFF);
✓	agg_sa_product	update_agg_sa_product regenerated.
✓	agg_sa_product	update_agg_sa_product compiled

Aggregate Table Column Properties

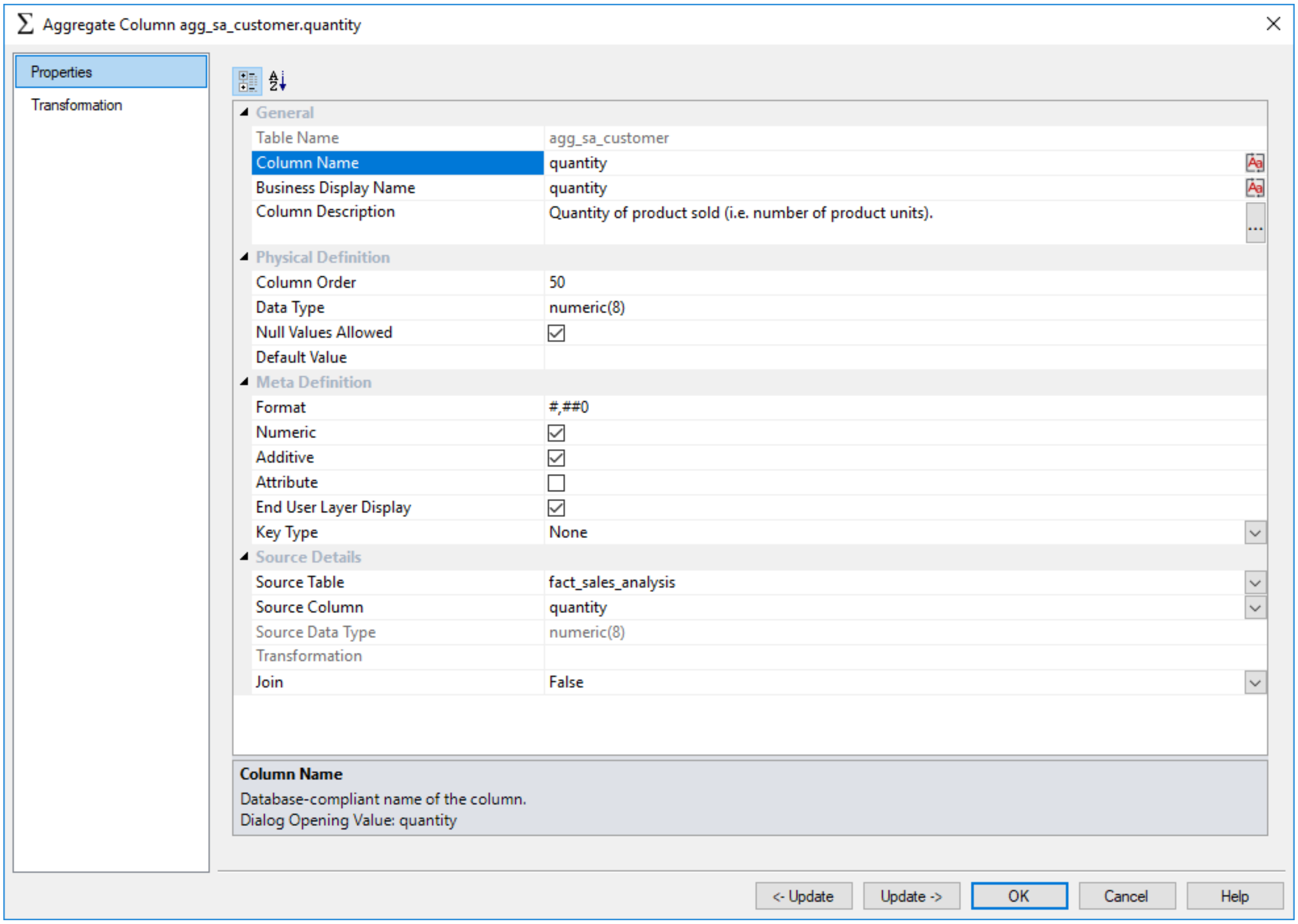
Each aggregate table column has a set of associated properties. The definition of each property is defined below:

If the **Column name** or **Data type** is changed for a column, then the metadata will differ from the table as recorded in the database. Use the **Validate > Validate Table Create Status** menu option to compare the metadata to the table in the database. A right click menu option of **Alter Table** is available when positioned on the table name after the validate has completed. This option will alter the database table to match the metadata definition.

Tip

If a database table's definition is changed in the metadata then the table will need to be altered in the database. Use the **Validate > Validate Table Create Status** to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table, through a pop-up menu option from the validated table name.

A sample **Properties** screen:



General	
Table Name	agg_sa_customer
Column Name	quantity
Business Display Name	quantity
Column Description	Quantity of product sold (i.e. number of product units).
Physical Definition	
Column Order	50
Data Type	numeric(8)
Null Values Allowed	☑
Default Value	
Meta Definition	
Format	#,##0
Numeric	☑
Additive	☑
Attribute	□
End User Layer Display	☑
Key Type	None
Source Details	
Source Table	fact_sales_analysis
Source Column	quantity
Source Data Type	numeric(8)
Transformation	
Join	False

Column Name
Database-compliant name of the column.
Dialog Opening Value: quantity

The two special update keys allow you to update the column and step either forward or backward to the next column's properties. **ALT-Left Arrow** and **ALT-Right Arrow** can also be used instead of the two special update keys.

General

Fields	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Business Display Name	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the

Fields	Description
	sequence each conversion.
Column Description	This field contains the description for the column. It may be a description from a business user's point of view. This field might additionally contain information on where and how the column was acquired. For example if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col .** This field is also stored as a comment against the column in the database.

Physical Definition

Fields	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. For SQL Server common types are integer, numeric, varchar() and datetime. See the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.

Meta Definition

Fields	Description
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.
Numeric	Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Additive	Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.

Fields	Description
Attribute	Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or a invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables. This checkbox does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
End User Layer Display	Indicates whether the table column is available/visible to end users. If set the documentation will include the column in the glossary and in the user documentation. It is also used to decide what columns appear in the view **ws_admin_v_dim_col.** Typically columns such as the artificial key would not be enabled for end user display.
Key Type	Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested. The supported values are:

Key type	Meaning
0	The artificial key. Set when the key is added during drag and drop table generation.
1	Component of all business keys. Indicates that this column is used as part of any business key. For example: By default the dss_source_system_key is added to every dimension table. It is considered part of any lookup on that table and has the key type set to 1. Set when the column is added during drag and drop table generation.
2	Indicates that this column is a dimensional join. Used on fact tables to indicate the dimension keys. Results in bitmap indexes being built for the columns. Set during the update procedure generation for a fact table, based on information from the staging table.
3	Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation.
4	Previous value column indicator. Used on dimension tables to indicate that the column is being managed as a previous value column. The source column identifies the parent column. Set during the dimension creation.
A	Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation.
B-Z	Indicates that the column is part of a secondary business key. Only used during index generation and not normally set.

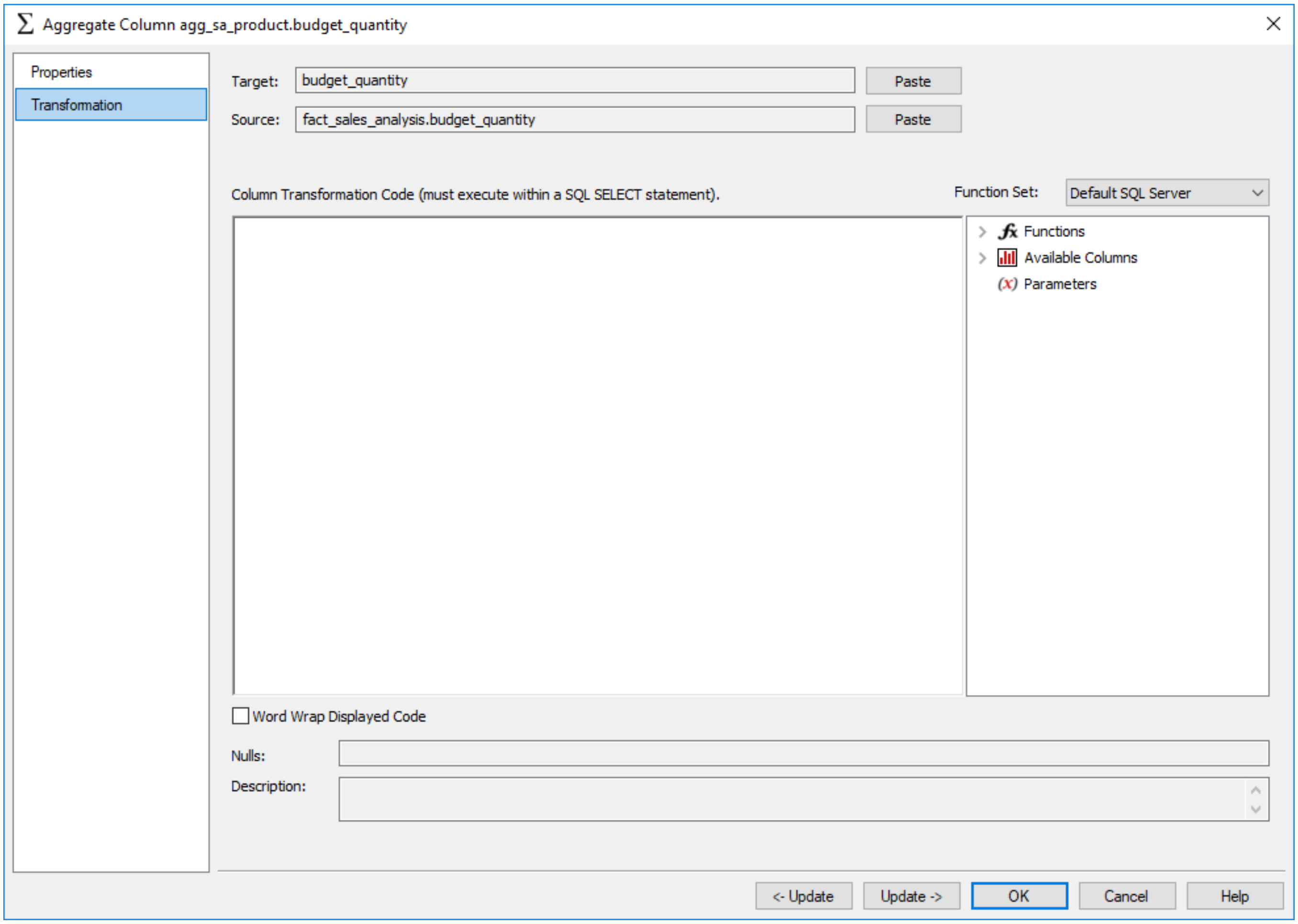
Source Details

Fields	Description
Source Table	Identifies the source table where the column's data comes from. This source table is normally a fact table or a dimension table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the **Source strategy** field. This field is used when generating a procedure to update the aggregate table. It is also used in the

Fields	Description
	track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a fact table column or a dimension table column, which in turn may have been a transformation or the combination of multiple columns.
Source Data Type	Identifies the source column's data type. [Read-only].
Transformation	Refer to **Aggregate Table Column Transformations**. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source Table** and **Source Column** fields will provide the dimension table's side of the join. The options for this field are: False, True, Manual and Pre Join. • Setting this field to Manual changes the way the dimension table is looked up during the update procedure build. It allows you to join the dimension manually in the **Source Join** wizard (used to build the 'Where' clause). • Setting this field to Pre Join activates the **Pre Join Source Table** field and allows you to select a table from the drop-down list.
Pre Join Source Table	Indicates the table from which the pre joined column was sourced. When the Join option is set to False, this field becomes inactive. When the Join option is set to True or Manual, this field is set to the current table name. When the Join option is set to Pre Join, then you can select the required table from the drop-down list.

Aggregate Table Column Transformations

Each aggregate table column can have a transformation associated with it. The transformation will be included in the generated procedure and will be executed as part of the procedure update. The transformation must therefore be a valid SQL construct that can be included in a **Select** statement. For example we could have a transformation of 'load_order_line.quantity * 0.125' to calculate a tax column of 12.5%. Click the **Transformation** tab to enter a transformation.



Target: budget_quantity

Source: fact_sales_analysis.budget_quantity

Column Transformation Code (must execute within a SQL SELECT statement).

Function Set: Default SQL Server

□ Word Wrap Displayed Code

Nulls:

Description:

Note

Transformations are only put into effect when the procedure is re-generated.

Refer to **Transformations** for details.

Views

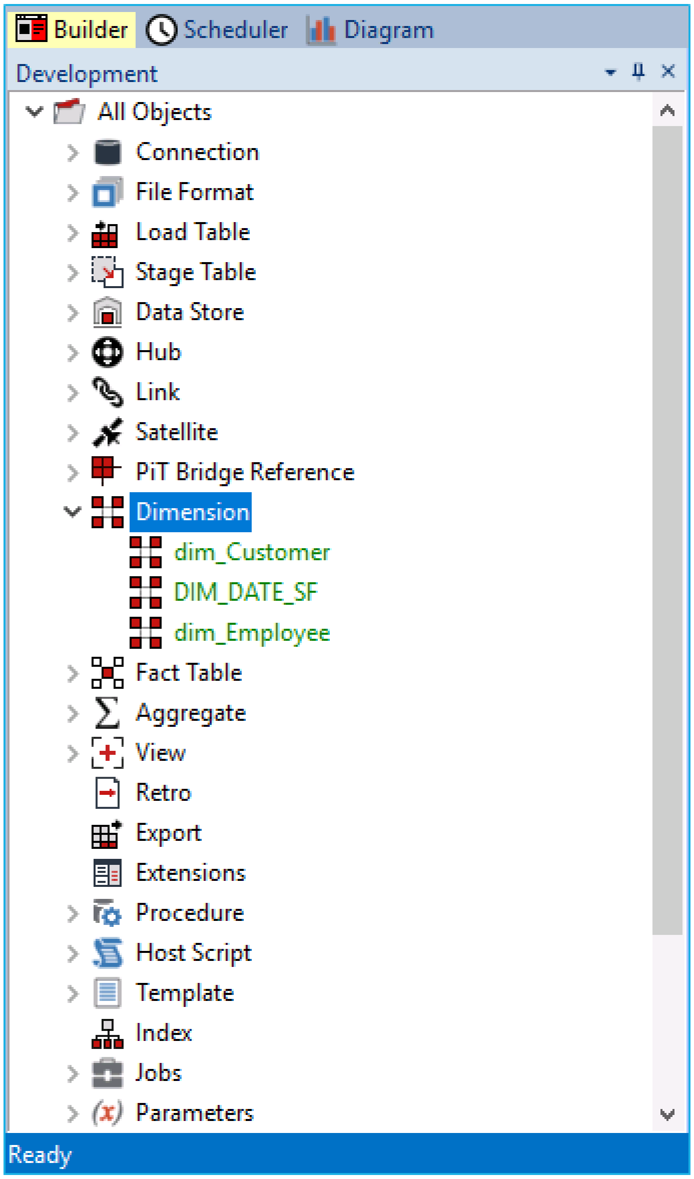
Views are normally created from Dimensions. These Views are then used to define Dimensions in a Fact table where the Dimension appears multiple times. A classic example is the date dimension. To facilitate the creation of the Fact table we create two dimension views to allow the date dimension to be joined to the Fact table twice.

Views can also be created from any other table type. By default, views are displayed as part of the Dimension object group. If Fact table views or other views are being used, then views can be displayed as a separate object type which can be configured from **Tools > Options > User Preferences > Look and Feel > General**.

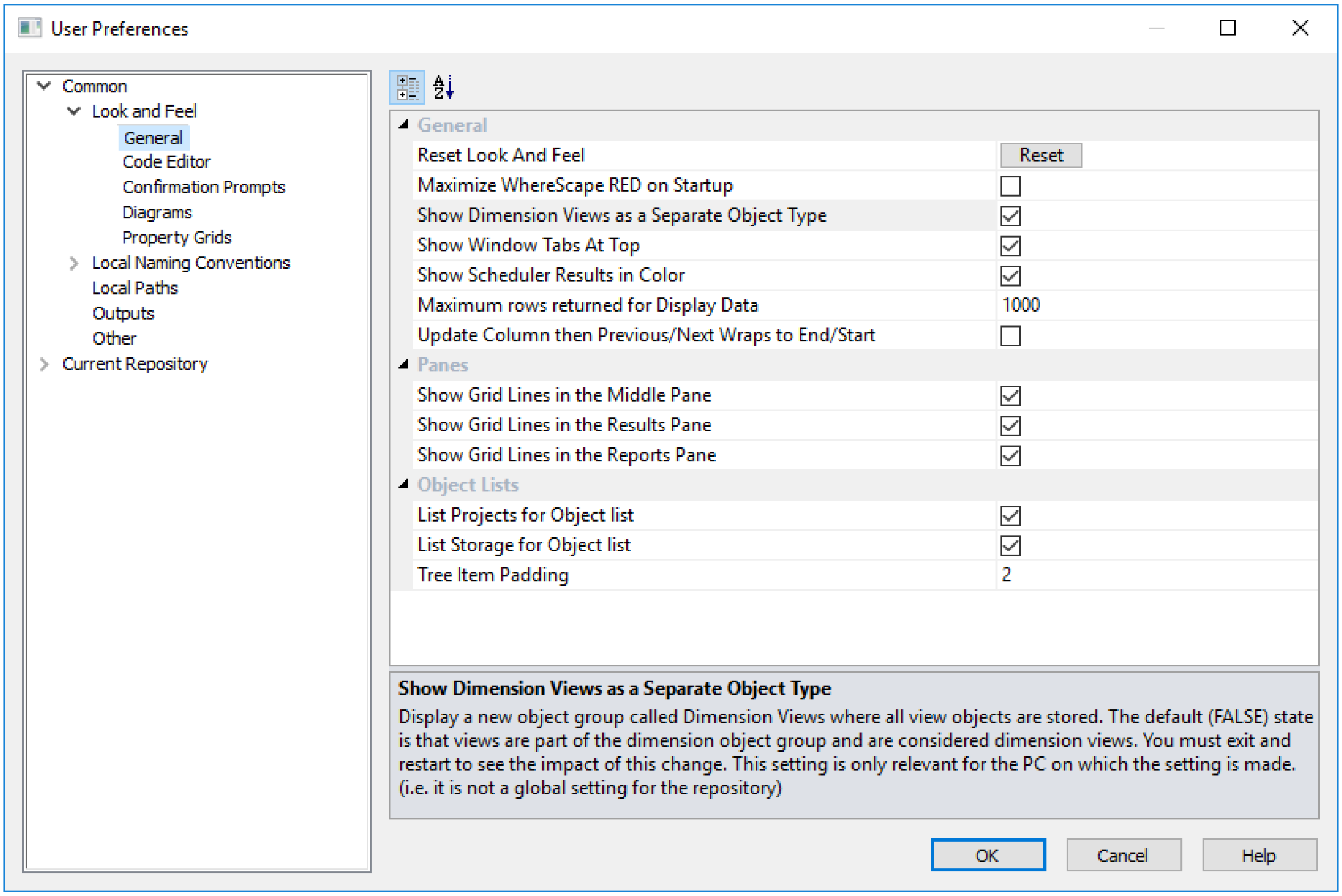
Dimension Views

A Dimension view is a database view of a Dimension table. It may be a full or partial view. It is typically used in cases such as date dimensions, where multiple date dimensions exist for one Fact table.
In many data warehouses Dimension views are built as part of the end user layer but creating them in the data warehouse means they are available regardless of the end user tools used.

Views are displayed in WhereScape RED either as part of the **Dimension** object group or in a separate object group—**Dimension View**.
The default is to view Dimensions and Dimension Views together:



This visualization option is set from the **Tools > User Preferences > Look and Feel** menu item:



General

Reset Look And Feel	Reset
Maximize WhereScape RED on Startup	□
Show Dimension Views as a Separate Object Type	☑
Show Window Tabs At Top	☑
Show Scheduler Results in Color	☑
Maximum rows returned for Display Data	1000
Update Column then Previous/Next Wraps to End/Start	□

Panes

Show Grid Lines in the Middle Pane	☑
Show Grid Lines in the Results Pane	☑
Show Grid Lines in the Reports Pane	☑

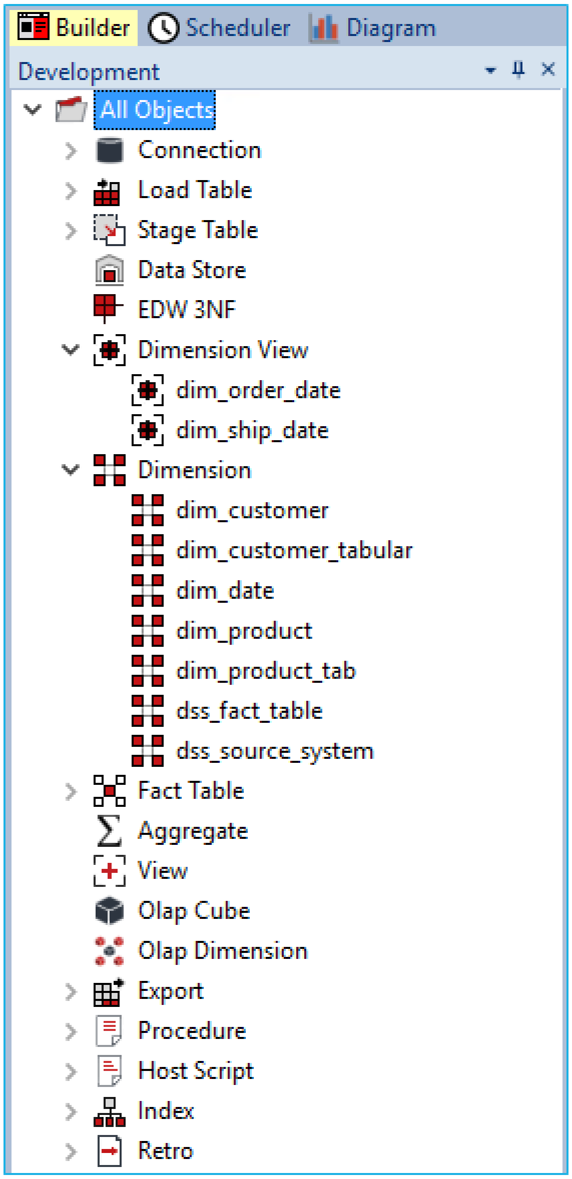
Object Lists

List Projects for Object list	☑
List Storage for Object list	☑
Tree Item Padding	2

Show Dimension Views as a Separate Object Type

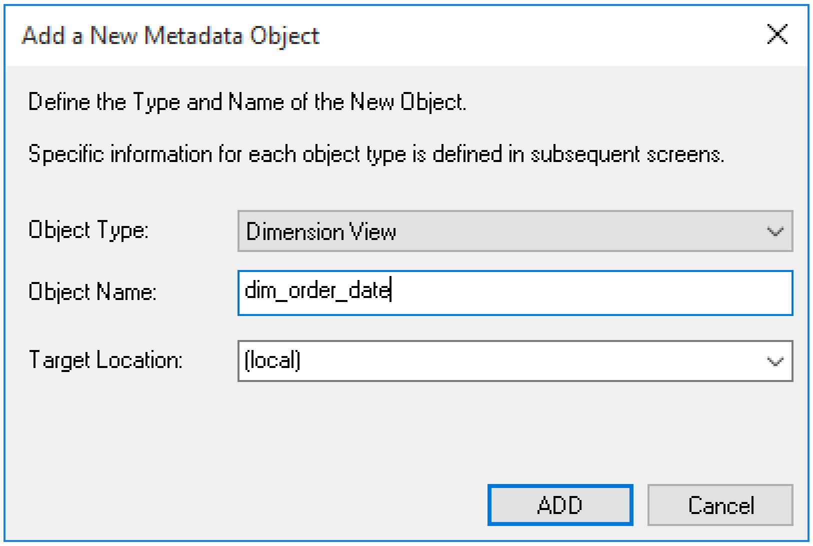
Display a new object group called Dimension Views where all view objects are stored. The default (FALSE) state is that views are part of the dimension object group and are considered dimension views. You must exit and restart to see the impact of this change. This setting is only relevant for the PC on which the setting is made. (i.e. it is not a global setting for the repository)

This example shows **Dimensions** and **Dimension Views** as separate object types:



Creating a Dimension View:

1. Double-click Dimension (or Dimension View) in the left pane.
2. Browse the data warehouse in the right pane.
3. Drag a table from the right pane into the middle pane.
   - The **Add New Metadata Object** window defaults the object type to a Dimension view
   - Change the view name as required and then click **Add**.



Add a New Metadata Object

Define the Type and Name of the New Object.

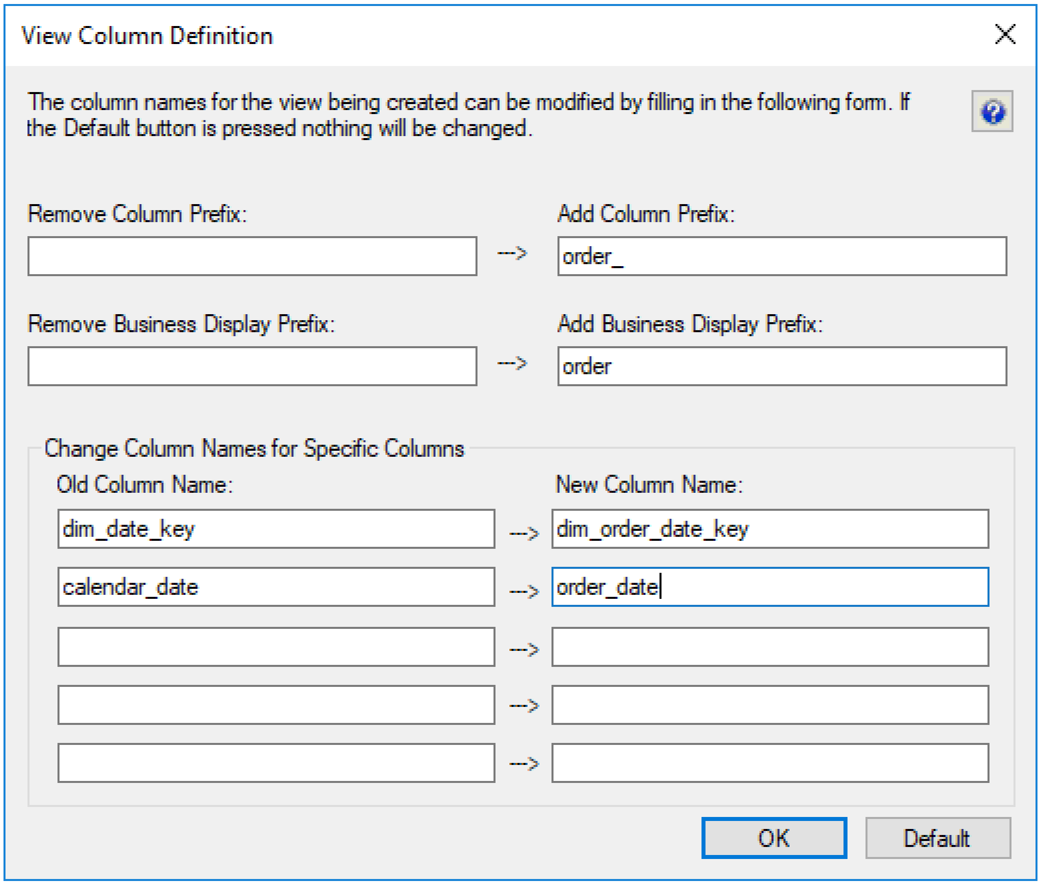
Specific information for each object type is defined in subsequent screens.

Object Type: Dimension View

Object Name: dim_order_date

Target Location: (local)

4. The **View Column Definition** window appears to provide a means of re-mapping some of the column names in the view if required.



View Column Definition

The column names for the view being created can be modified by filling in the following form. If the Default button is pressed nothing will be changed.

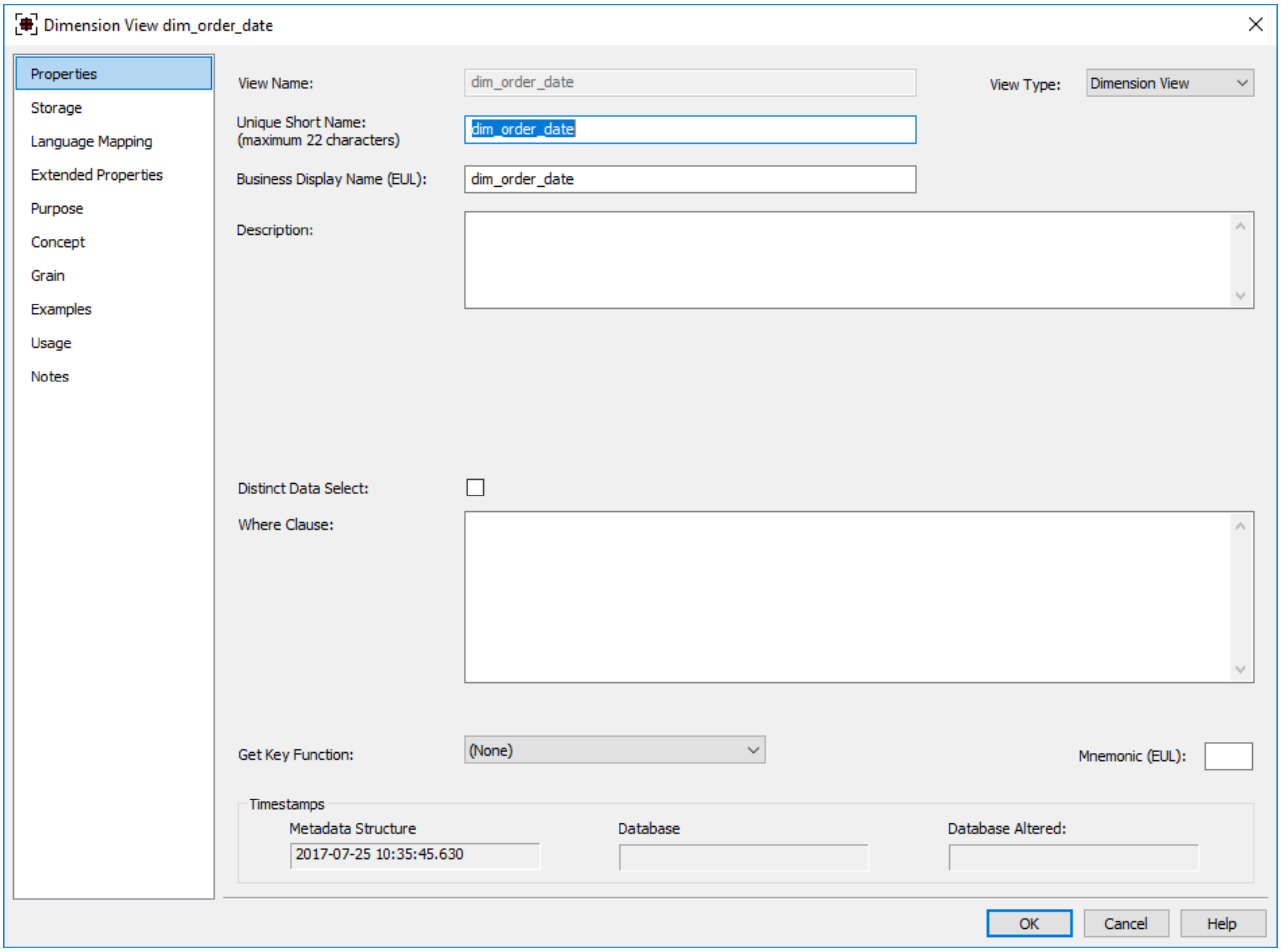
Remove Column Prefix: --> Add Column Prefix: order_

Remove Business Display Prefix: --> Add Business Display Prefix: order

Change Column Names for Specific Columns

Old Column Name:		New Column Name:
dim_date_key	-->	dim_order_date_key
calendar_date	-->	order_date
	-->	
	-->	
	-->	

5. Click **OK**, the **View** properties window is displayed:
   - Select the **Distinct Data Select** check box if you want the view to return only different values.
   - A **'Where'** clause could be entered, but this can be done later. Normally you would just click **OK**.



Dimension View dim_order_date

View Name: dim_order_date

View Type: Dimension View

Unique Short Name: (maximum 22 characters) dim_order_date

Business Display Name (EUL): dim_order_date

Description:

Distinct Data Select: □

Where Clause:

Get Key Function: (None)

Mnemonic (EUL):

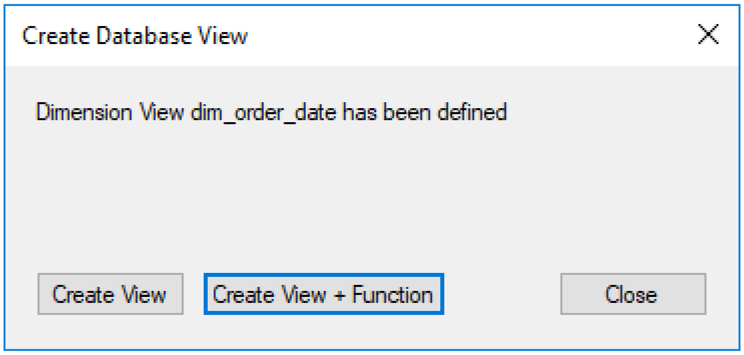
Timestamps

Metadata Structure: 2017-07-25 10:35:45.630

Database:

Database Altered:

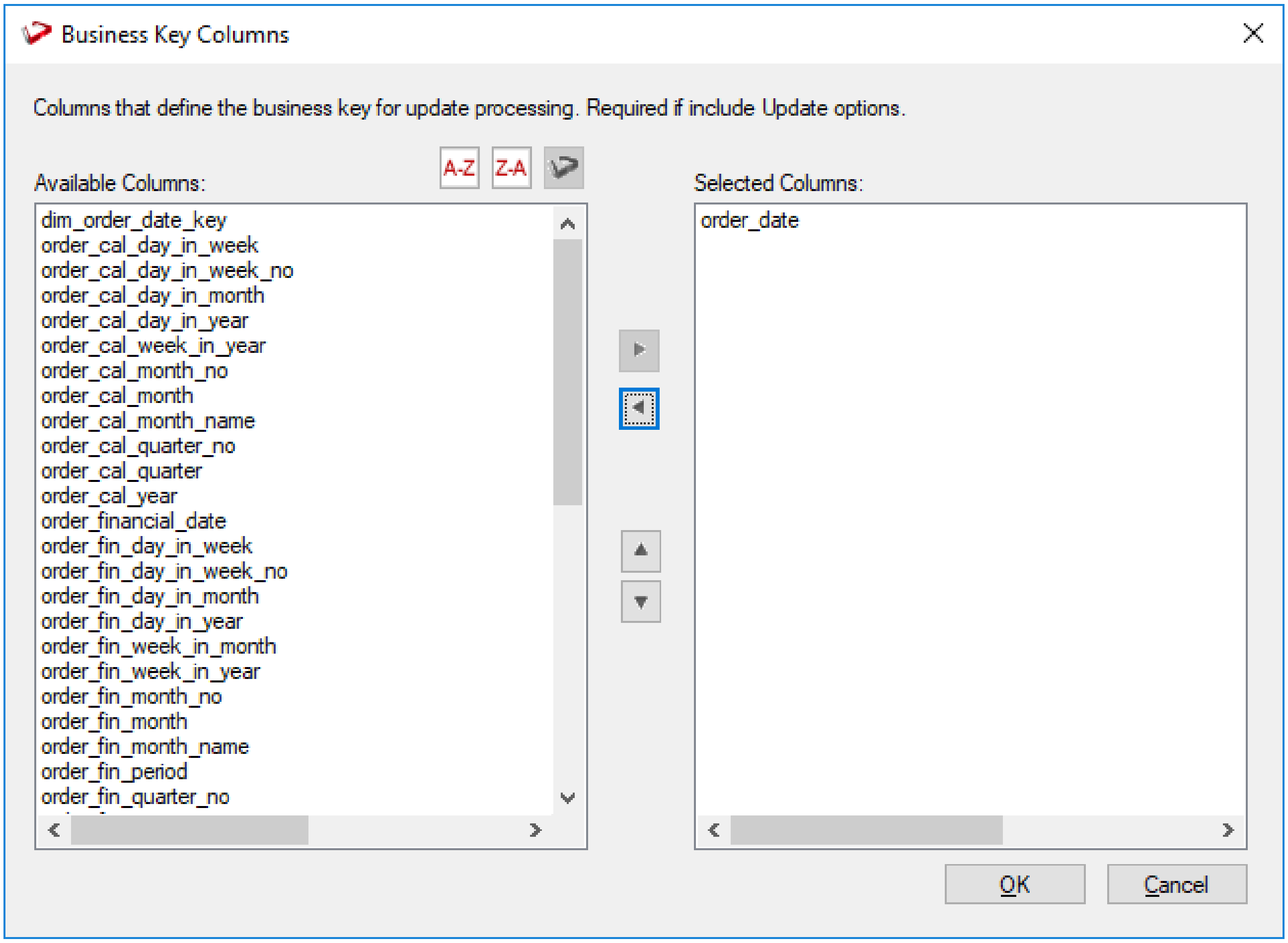
6. A prompt appears to indicate that the view has been defined.
   - Because this is a Dimension view, you are given the option to create both the view and an associated Get Key function. For any other view type, a Get Key function is not created.



Create Database View

Dimension View dim_order_date has been defined

7. If you click **Create View + Function**, the following window appears which enables you to select a business (natural) key for the Dimension view.



Columns that define the business key for update processing. Required if include Update options.

Available Columns:

dim_order_date_key
order_cal_day_in_week
order_cal_day_in_week_no
order_cal_day_in_month
order_cal_day_in_year
order_cal_week_in_year
order_cal_month_no
order_cal_month
order_cal_month_name
order_cal_quarter_no
order_cal_quarter
order_cal_year
order_financial_date
order_fin_day_in_week
order_fin_day_in_week_no
order_fin_day_in_month
order_fin_day_in_year
order_fin_week_in_month
order_fin_week_in_year
order_fin_month_no
order_fin_month
order_fin_month_name
order_fin_period
order_fin_quarter_no

Selected Columns:

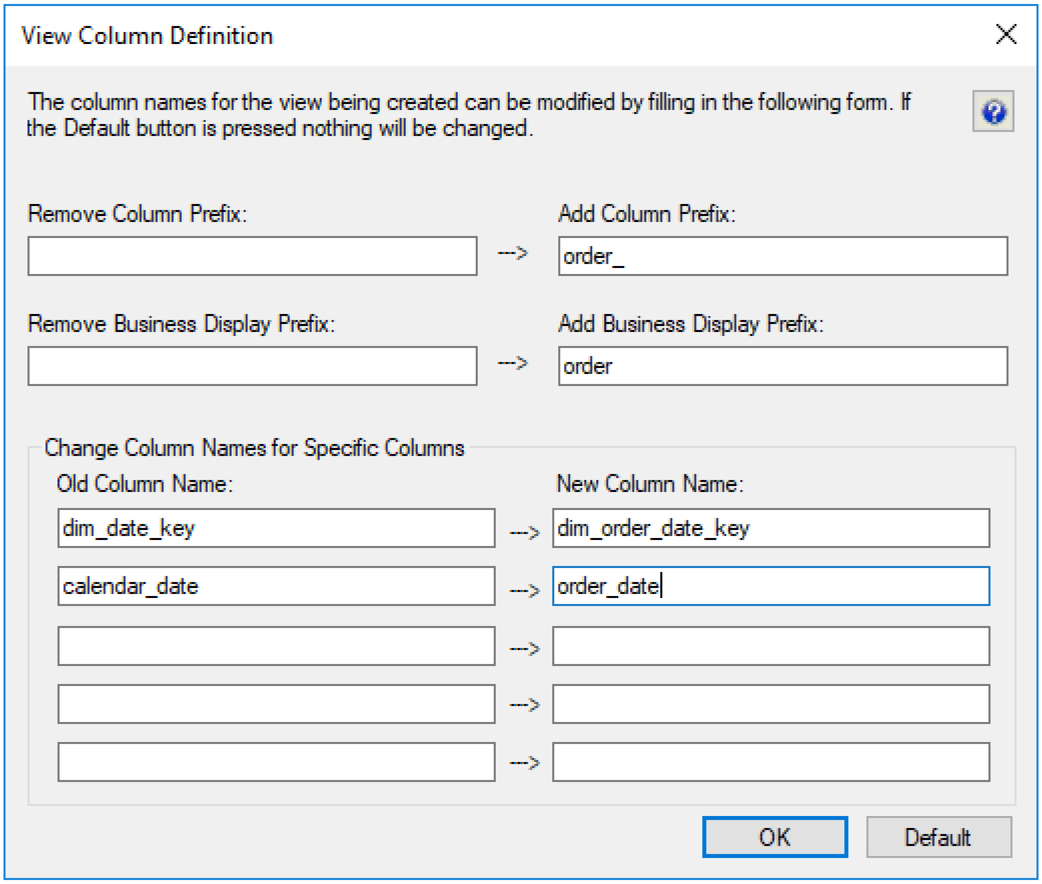
order_date

View Column Re-mapping

The **View Column Definition** window is used for the automated re-mapping of certain column names. It provides an easy method for changing the column names for a large number of columns, when creating a View.

The various actions undertaken as a result of entries in this window can all be done or reversed manually by changing the individual column properties.

The various fields are described below:



The column names for the view being created can be modified by filling in the following form. If the Default button is pressed nothing will be changed.

Remove Column Prefix: --> Add Column Prefix: order_

Remove Business Display Prefix: --> Add Business Display Prefix: order

Change Column Names for Specific Columns

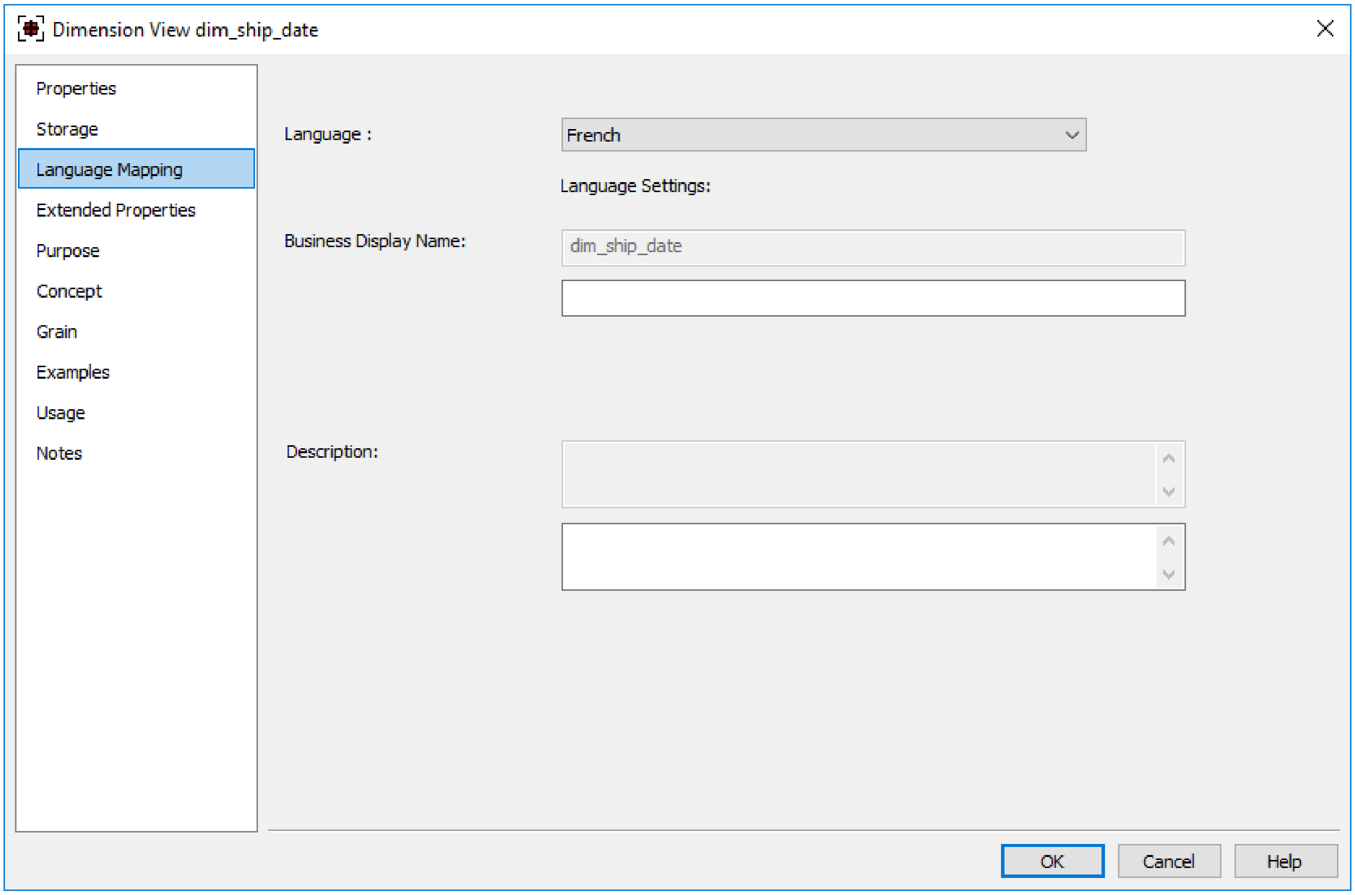
Old Column Name:		New Column Name:
dim_date_key	-->	dim_order_date_key
calendar_date	-->	order_date
	-->	
	-->	
	-->	

Fields	Description
Remove Column Prefix	If the columns in the source table are prefixed then that prefix can be removed by entering it in this field. An example may be a date dimension which has each column prefixed with date_ (e.g. date_month, date_year etc.). If this field is left blank then no removal action is taken.
Add Column Prefix	If required a prefix can be added to each column name. This is particularly useful when defining a date dimension view where you would like each column to be prefixed by the date type.
Remove Business Display Prefix	As per the column names, it may be required to remove a prefix from the business display fields. If so enter the prefix to remove in this column.
Add Business Display Prefix	The business display fields are used in the creation of the glossary. It is therefore quite useful to prefix these display fields with an identifier for the view being created. It is assumed that these business display names will be carried forward to the end user layer. Enter a value in this field to prefix the business display name fields for each column. It is normal to include a space at the end of this field.
Old Column Name	Up to five individual column names can be re mapped. Enter the column name as it appears in the source table in one of the **Old Column Name** fields in order to re map that column name. The business display name is also changed to match.
New Column Name	Place a new column name alongside any existing column name you wish to re map. In the example dialog above a column named **calendar_date** is being renamed to **order_date** in the view.

Dimension View Language Mapping

The **Dimension View Properties** screen has a **Language Mapping** tab.

1. Select the language from the drop-down list and then enter the translations for the **Business Display Name** and the **Description** in the chosen language.
2. The translations for these fields can then be pushed through into OLAP cubes.



Language : French

Language Settings:

Business Display Name: dim_ship_date

Description:

Tip

Languages can be added and maintained from the **Tools > Language Options** menu. Refer to **Language Options** for details.

Fields	Description
Business Display Name	Translated Value of Business Display Name.
Description	Translated Value of Description.

Non-Dimension Object Views

WhereScape RED also supports creating views of objects other than dimensions (Load tables, Facts, Stage tables, Aggregates, etc.).

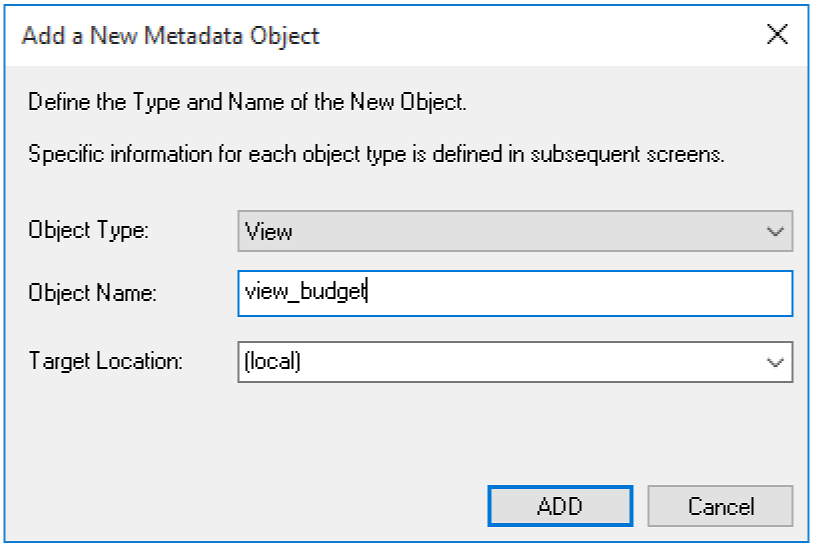
A Fact View is a database view of a Fact table. It may be a full, partial view, or a view incorporating both the fact data and some dimensional attributes.

It is typically used in cases where a subset of information is to be provided for specific end user communities. It is common that the 'Where' clause be populated in fact views, either to join the dimension tables or restrict the data that the view represents.

Similarly, views may be created over Staging tables or any other **table** object.

The creation of non-dimension table views is the same process as that for Dimension views:

1. Drag a table from the right pane into the middle pane.
   - The **Add New Metadata Object** window defaults the object type to a **View**.
   - Change the View name as required, and click **Add**.



Add a New Metadata Object

Define the Type and Name of the New Object.

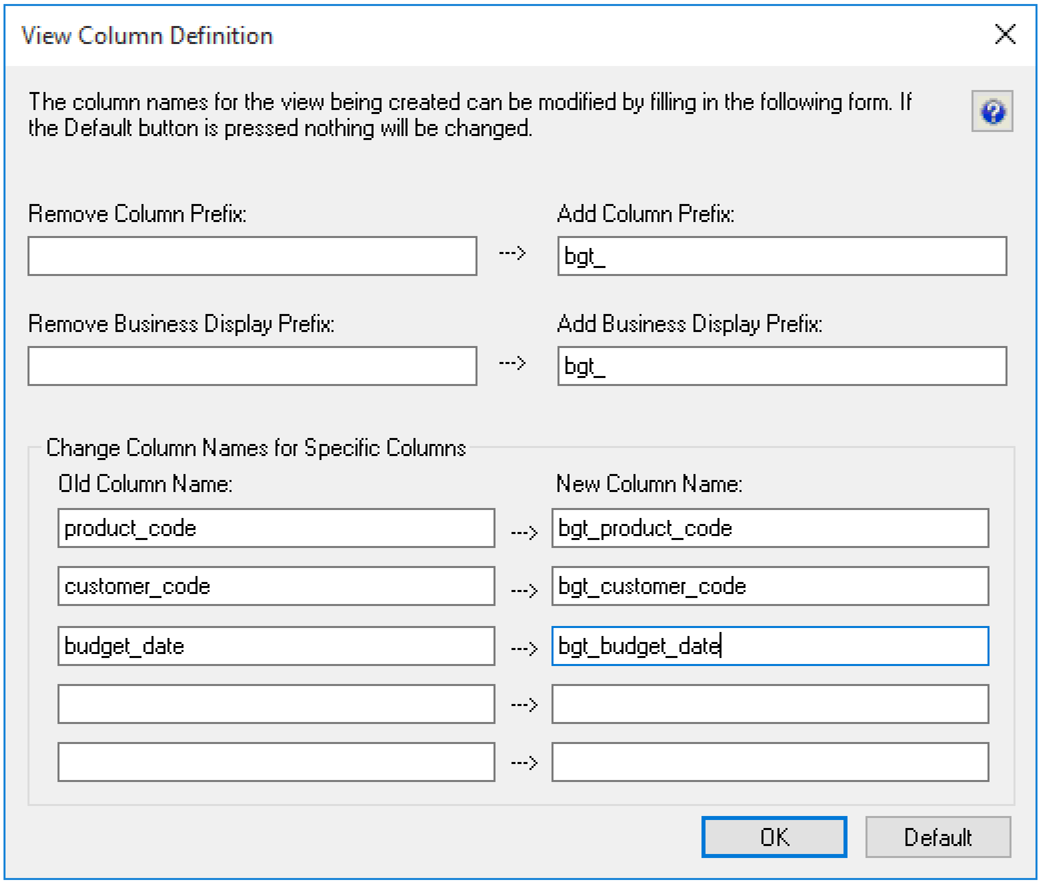
Specific information for each object type is defined in subsequent screens.

Object Type: View

Object Name: view_budget

Target Location: (local)

2. The **View Column Definition** window enables the automated re-mapping of certain column names. It provides an easy method for changing the column names for a large number of columns when creating a view.



View Column Definition

The column names for the view being created can be modified by filling in the following form. If the Default button is pressed nothing will be changed.

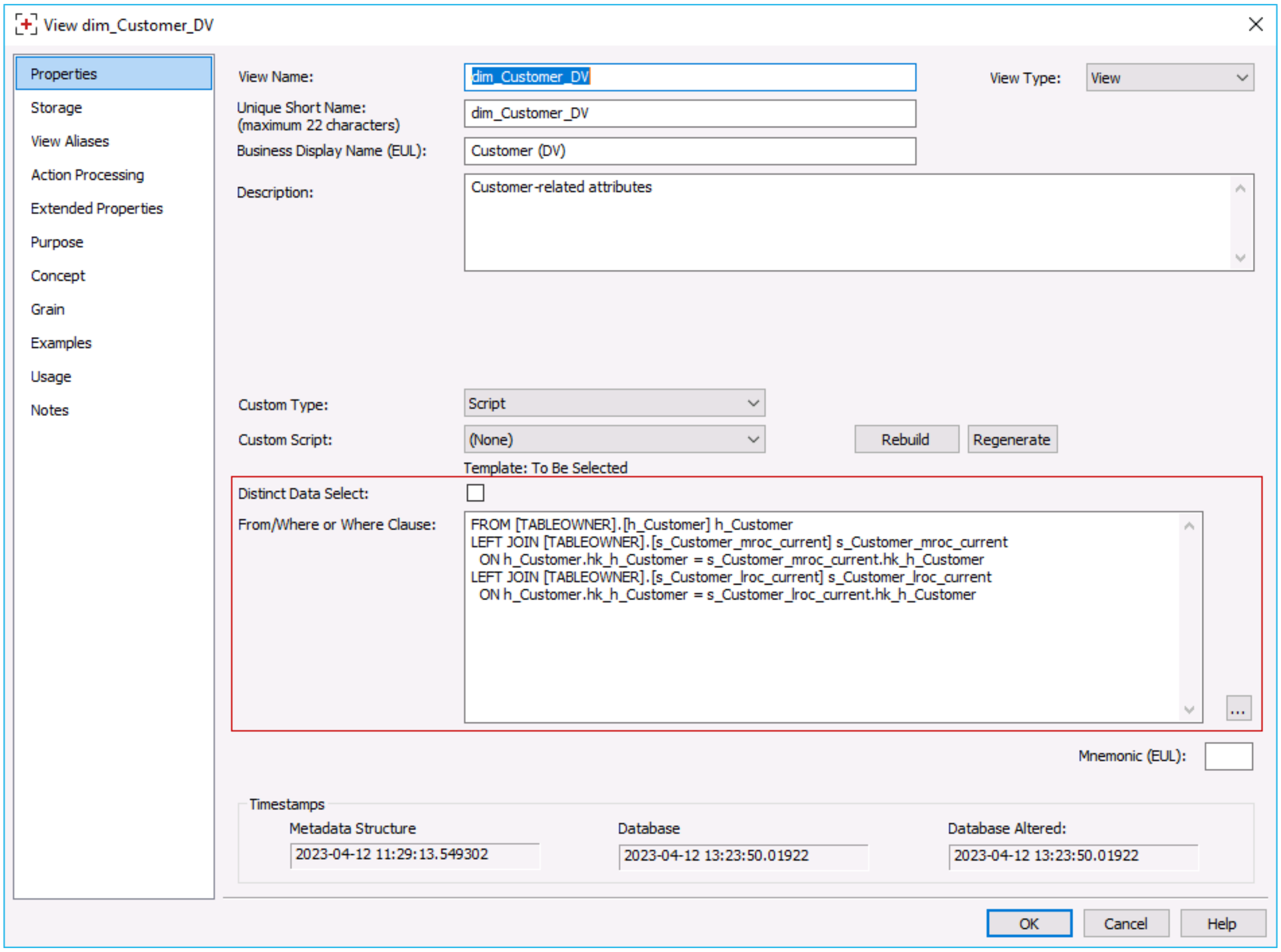
Remove Column Prefix: --> Add Column Prefix: bgt_

Remove Business Display Prefix: --> Add Business Display Prefix: bgt_

Change Column Names for Specific Columns

Old Column Name:		New Column Name:
product_code	--->	bgt_product_code
customer_code	--->	bgt_customer_code
budget_date	--->	bgt_budget_date
	--->	
	--->	

3. Click **OK**, the **View** properties window is displayed:


View dim_Customer_DV

View Name: dim_Customer_DV

View Type: View

Unique Short Name: (maximum 22 characters) dim_Customer_DV

Business Display Name (EUL): Customer (DV)

Description: Customer-related attributes

Custom Type: Script

Custom Script: (None)

Template: To Be Selected

Distinct Data Select: □

From/Where or Where Clause:

```
FROM [TABLEOWNER].[h_Customer] h_Customer
LEFT JOIN [TABLEOWNER].[s_Customer_mroc_current] s_Customer_mroc_current
  ON h_Customer.hk_h_Customer = s_Customer_mroc_current.hk_h_Customer
LEFT JOIN [TABLEOWNER].[s_Customer_lroc_current] s_Customer_lroc_current
  ON h_Customer.hk_h_Customer = s_Customer_lroc_current.hk_h_Customer
```

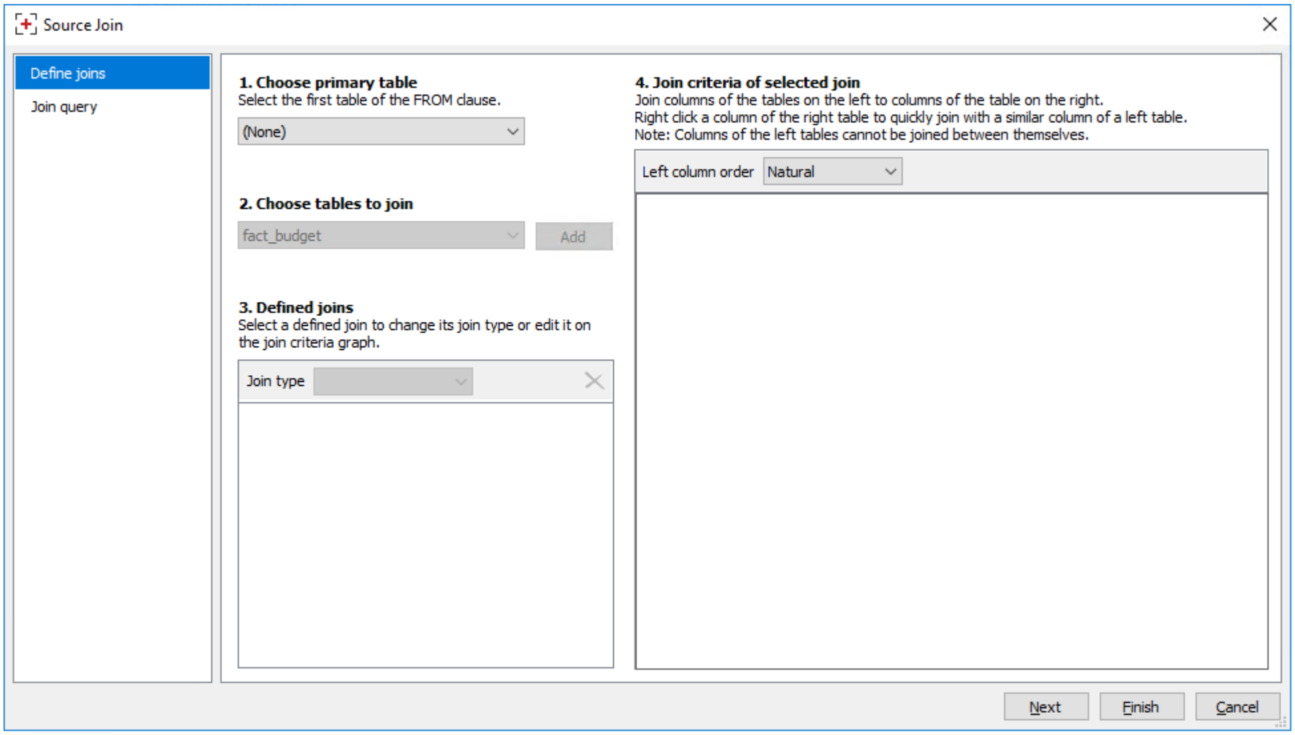
Mnemonic (EUL):

Timestamps

Metadata Structure	Database	Database Altered:
2023-04-12 11:29:13.549302	2023-04-12 13:23:50.01922	2023-04-12 13:23:50.01922

Change the following properties, if required:

- Select the **Distinct Data Select** check box if you want the view to return only distinct values.
- A **From/Where** clause could be entered but this can also be done later.
  Clicking the ellipses button opens the **Source Join** wizard shown below. Refer to **Generating the Dimension Update Procedure > Source tab > Joining multiple source tables** for details.



1. Choose primary table
Select the first table of the FROM clause.
(None)

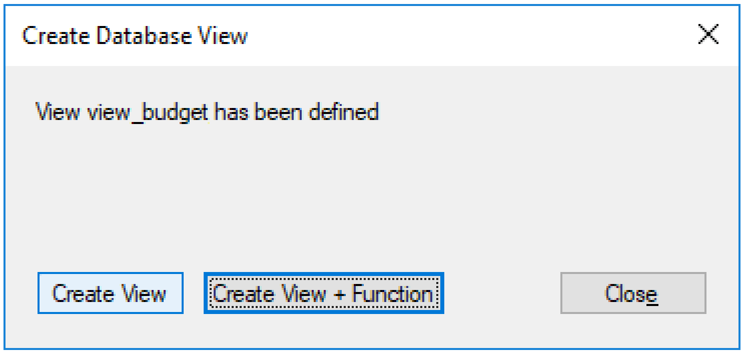
2. Choose tables to join
fact_budget

3. Defined joins
Select a defined join to change its join type or edit it on the join criteria graph.

4. Join criteria of selected join
Join columns of the tables on the left to columns of the table on the right.
Right click a column of the right table to quickly join with a similar column of a left table.
Note: Columns of the left tables cannot be joined between themselves.

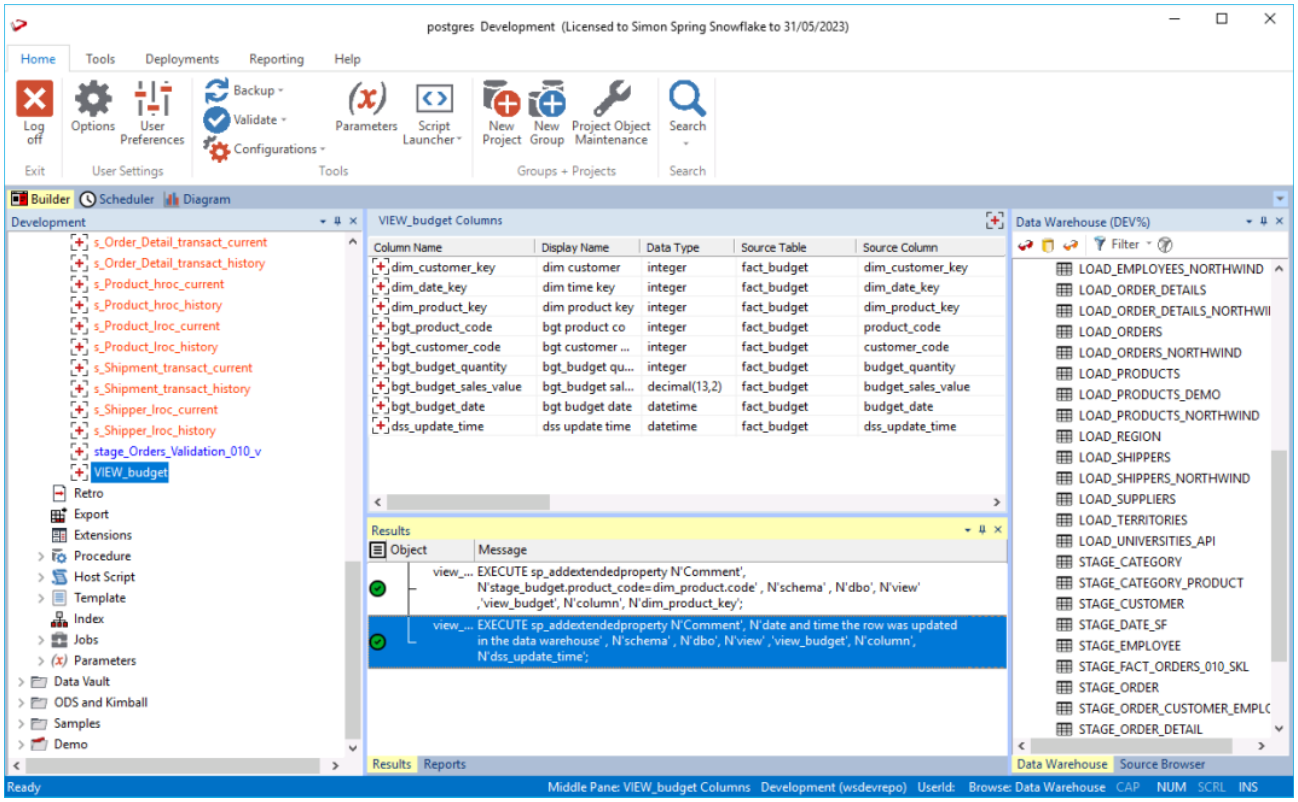
Left column order: Natural

4. Click **OK** in the **View** properties window, after completing the view definition. A prompt is displayed to indicate that the view has been defined, click **Create View**.



View view_budget has been defined

The columns of the **View** table is displayed in the middle pane.



VIEW_budget Columns

Column Name	Display Name	Data Type	Source Table	Source Column
dim_customer_key	dim customer	integer	fact_budget	dim_customer_key
dim_date_key	dim time key	integer	fact_budget	dim_date_key
dim_product_key	dim product key	integer	fact_budget	dim_product_key
bgt_product_code	bgt product co	integer	fact_budget	product_code
bgt_customer_code	bgt customer ...	integer	fact_budget	customer_code
bgt_budget_quantity	bgt_budget qu...	integer	fact_budget	budget_quantity
bgt_budget_sales_value	bgt_budget sal...	decimal(13,2)	fact_budget	budget_sales_value
bgt_budget_date	bgt budget date	datetime	fact_budget	budget_date
dss_update_time	dss update time	datetime	fact_budget	dss_update_time

Results

Object	Message
view_...	EXECUTE sp_addextendedproperty N'Comment', N'stage_budget.product_code=dim_product.code' , N'schema' , N'dbo', N'view' ,'view_budget', N'column', N'dim_product_key';
view_...	EXECUTE sp_addextendedproperty N'Comment', N'date and time the row was updated in the data warehouse' , N'schema' , N'dbo', N'view' ,'view_budget', N'column', N'dss_update_time';

Note

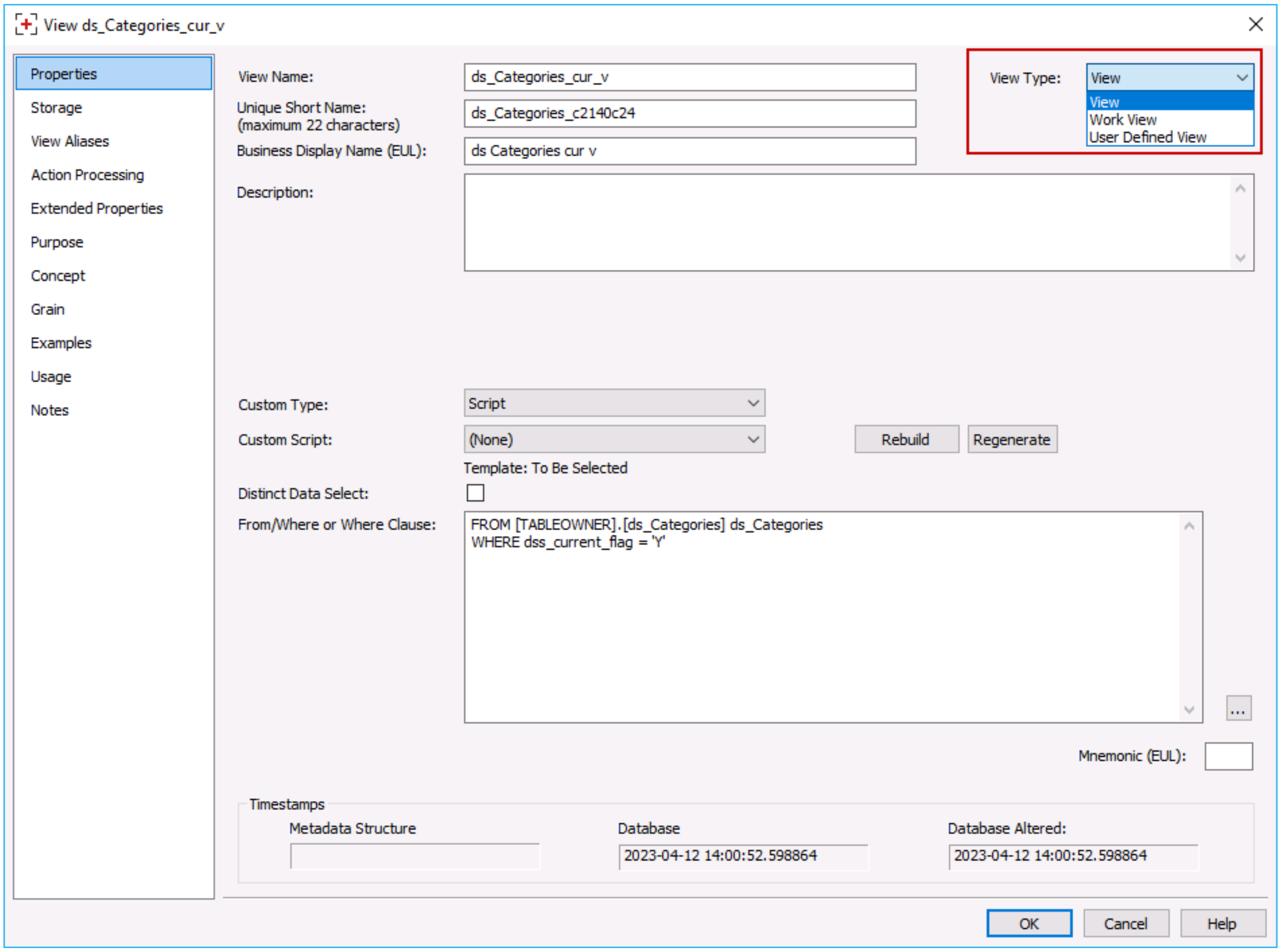
If Dimension attributes are being added (e.g.: to a Fact view) then drag these attributes into the views column list, create the view and edit the 'Where' clause for the view.

Creating a Custom View

A custom view can be created within WhereScape RED to handle views that are not strictly one to one such as where multiple tables are joined or where a complex condition is placed on the view. There are two options for custom views, the first where the columns are defined in WhereScape RED and the 'Select' component of the view is customized. The second option is where the view is totally customized and no columns need to be defined in WhereScape RED, although it is good practice to still define the columns for documentation purposes.

Creating a Custom or 'User Defined' view:

1. Create a View either by dragging a table in or adding a new object.
2. Change the table type to **User Defined View** from the **View Type** drop-down list.



View Name: ds_Categories_cur_v

Unique Short Name: (maximum 22 characters) ds_Categories_c2140c24

Business Display Name (EUL): ds Categories cur v

Description:

View Type: View (View / Work View / User Defined View)

Custom Type: Script

Custom Script: (None)

Template: To Be Selected

Distinct Data Select: □

From/Where or Where Clause:

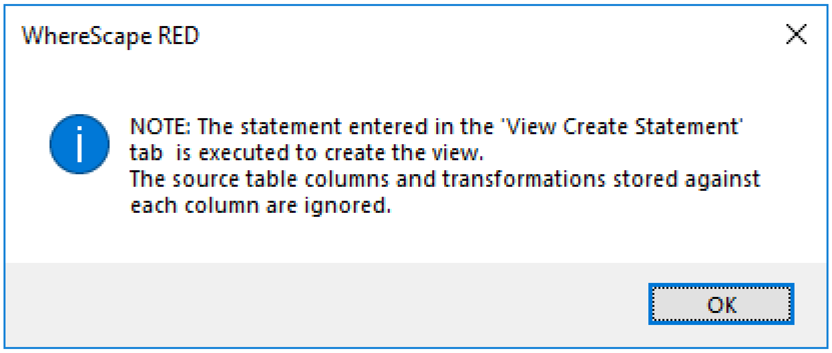
```
FROM [TABLEOWNER].[ds_Categories] ds_Categories
WHERE dss_current_flag = 'Y'
```

Mnemonic (EUL):

Timestamps

Metadata Structure	Database	Database Altered:
	2023-04-12 14:00:52.598864	2023-04-12 14:00:52.598864

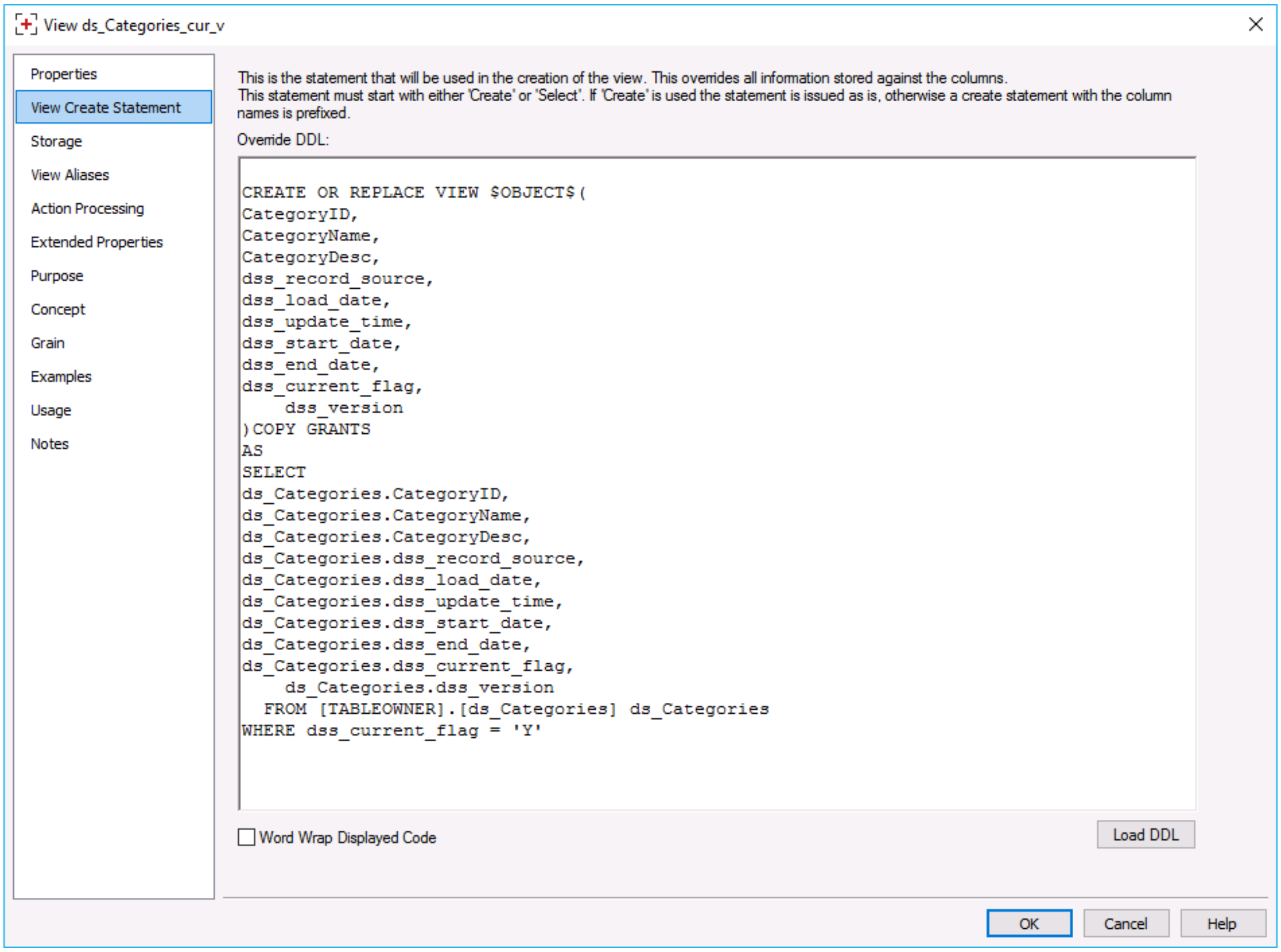
3. The following message is displayed. Click **OK**.



WhereScape RED

NOTE: The statement entered in the 'View Create Statement' tab is executed to create the view.
The source table columns and transformations stored against each column are ignored.

4. Edit the new tab **View Create Statement** and insert the SQL Statement that will be used to create the view.
   - This SQL Statement must start with either **Create** or **Select**.
   - If **Create** is used then the columns in the view are ignored and the statement will be issued to create the view.
   - If the statement starts with **Select** then WhereScape RED will build up a view create statement from the column names and the supplied **Select** clause.
   - There is also a **Load DDL** button at the bottom right corner to get a sample **Select** statement, based on the columns in the view and any transformations.



This is the statement that will be used in the creation of the view. This overrides all information stored against the columns.
This statement must start with either 'Create' or 'Select'. If 'Create' is used the statement is issued as is, otherwise a create statement with the column names is prefixed.

Override DDL:

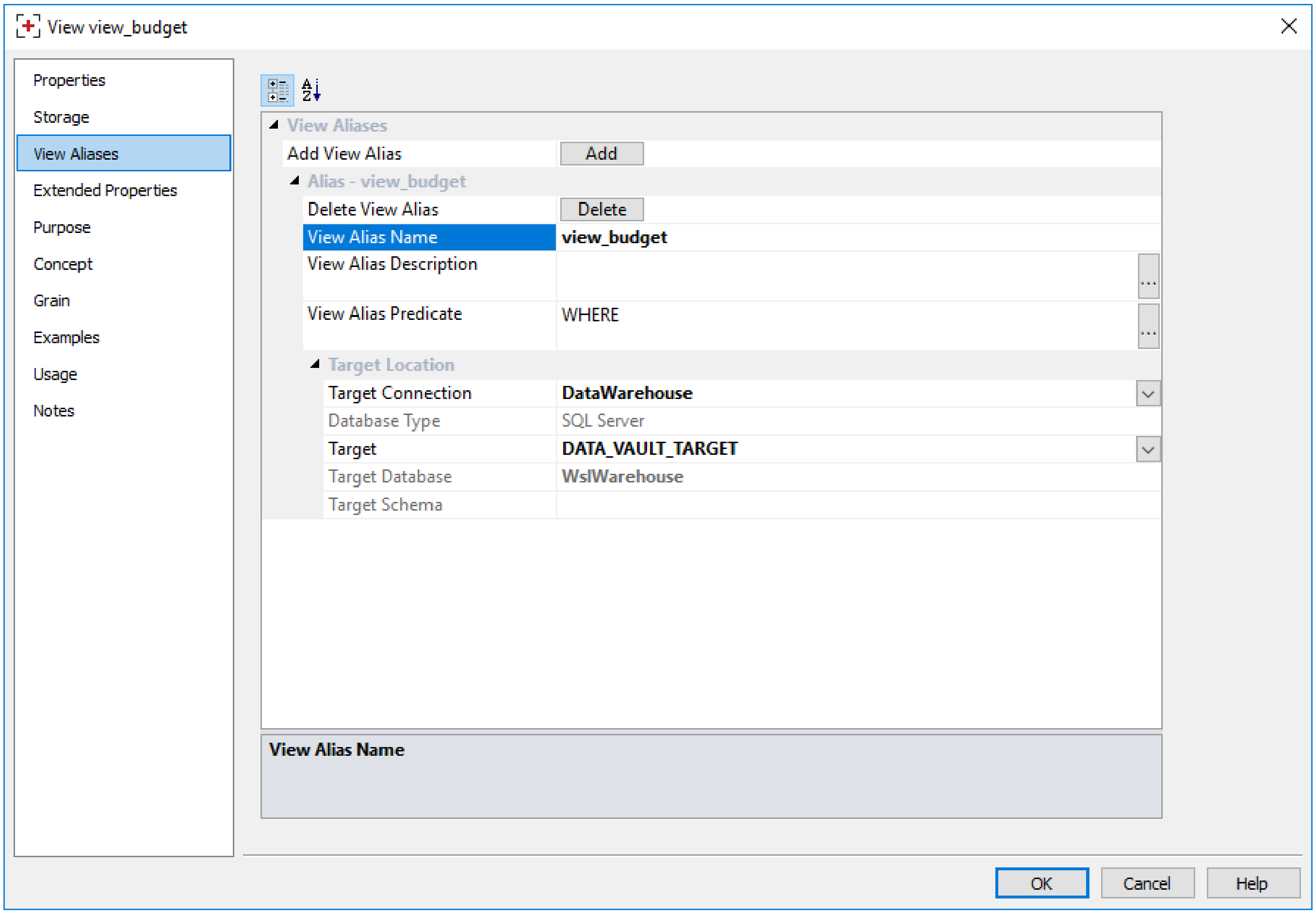
```
CREATE OR REPLACE VIEW $OBJECT$(
CategoryID,
CategoryName,
CategoryDesc,
dss_record_source,
dss_load_date,
dss_update_time,
dss_start_date,
dss_end_date,
dss_current_flag,
    dss_version
)COPY GRANTS
AS
SELECT
ds_Categories.CategoryID,
ds_Categories.CategoryName,
ds_Categories.CategoryDesc,
ds_Categories.dss_record_source,
ds_Categories.dss_load_date,
ds_Categories.dss_update_time,
ds_Categories.dss_start_date,
ds_Categories.dss_end_date,
ds_Categories.dss_current_flag,
    ds_Categories.dss_version
  FROM [TABLEOWNER].[ds_Categories] ds_Categories
WHERE dss_current_flag = 'Y'
```

☐ Word Wrap Displayed Code

View Aliases

A View Alias provides the ability to create security views on SQL Server in an alternate schema.

The **View Aliases** tab enables you to define additional/replica views.



Add View Alias: Add

Delete View Alias: Delete

View Alias Name: view_budget

View Alias Description:

View Alias Predicate: WHERE

Target Connection: DataWarehouse

Database Type: SQL Server

Target: DATA_VAULT_TARGET

Target Database: WslWarehouse

Target Schema:

View Alias Name

Fields	Description
Add button	Enables you to add a View Alias which adds a new copy of the view. This enables you to define new View Alias which is an alternate definition that will ultimately exist as another database view with a different name, predicate and/or target location.
Delete button	Enables you to delete a View Alias and remove its metadata details after closing the screen.
View Alias Name	The view alias name that is used as the alternate database view name.
View Alias Description	Description of the view alias.
View Alias Predicate	Optional 'Where' clause to include in the alternate view definition.
Target Connection	The name of the WhereScape RED connection that identifies the target location to create the alternate view.
Database Type	[Read Only] The database type of the selected target connection.
Target	The target that defines the database and schema for the table. Leave blank for the connection default.
Target Database	[Read Only] Database Name to reference the alternate view.
Target Schema	[Read Only] The target Database/User schema for the alternate view. Leave blank to use the default schema,

View Alias Through Templates

RED supports template based Create View DDL Statement generation for view aliases.

The CREATE VIEW statement needs to reference `$OBJECT$` or `$DATABASE$.$SCHEMA$.$TABLE$` tokens or use template variables to explicitly form the qualified view name. You cannot use the `[TABLEOWNER]` markup in the create statement as it cannot be correctly replaced at run-time for each View Alias. This means your existing create view templates may need updating to work with View Aliases.

For example, use:

```
CREATE OR REPLACE VIEW $OBJECT$
```

Instead of:

```
CREATE OR REPLACE VIEW [TABLEOWNER].[{{table.name}}]
```

Additional steps to create View Aliases for template based Create View DDL statement are the following:

- Configure a Create View DDL Template in the view object.
- Configure Drop View SQL in the connection.

The Default Create View DDL Template for new View Objects can be configured in the connection.

Note

The same CREATE VIEW DDL and Drop View SQL are used for both View Objects and View Aliases.

View Aliases are not meta objects and they cannot be found when applying `[TABLEOWNER]` markup, but `[TABLEOWNER]` can still be used elsewhere in the CREATE VIEW statement such as in FROM clause when used to markup source tables.

It is also possible to branch the logic in the Create View DDL Template by inspecting a new template variable `table.viewAliasParent`. This variable is only defined for view aliases, so the logic can be used only for View Aliases:

```
{% if table.viewAliasParent is defined %}
```

The `table.viewAliasParent` variable identifies the view object that owns the view alias.

Exporting Data

Export objects are used in WhereScape RED to produce ascii files from a single database table or view for a downstream feed. Some or all of the columns in a table or view can be exported.

There are three ways of performing exports in RED:

- **File export** - an export where most of the processing is managed and controlled by the scheduler
- **Script-based export** - an export where a Windows script file is executed to perform the export. Script-based exports on Windows supports both DOS Batch and PowerShell scripts.
- **Integration Services Export** - an export processed using a Windows connection where the processing is handled via an Integration Services Package that is generated and executed dynamically at run time. SSIS exports to UNIX/Linux connections and processed via the UNIX/Linux scheduler are currently not supported.

Building an Export Object

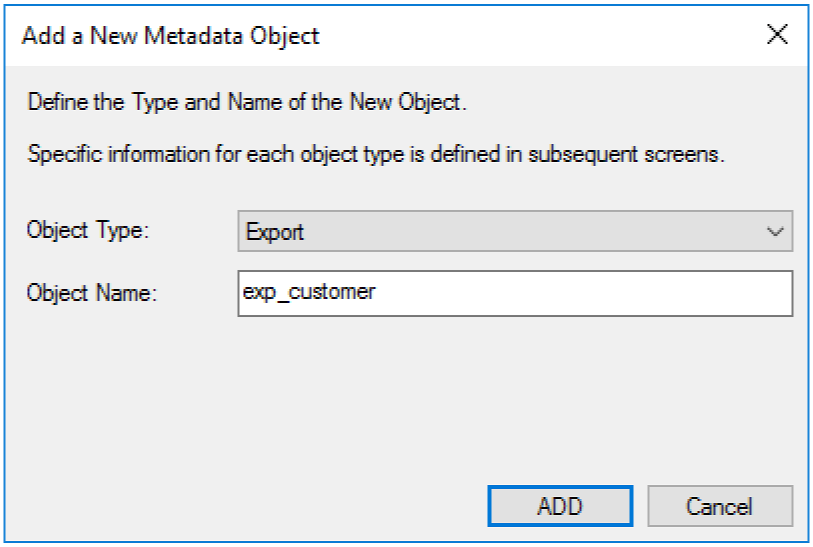
Tips

- The relevant client software for each database type must be installed on each machine you want to export to.

The simplest way to create an export object is to use the **drag and drop** functionality of WhereScape RED.

Creating an Export:

1. Browse to the Datawarehouse connection (**Browse > Source Tables**).
2. Create a drop target by double-clicking on the **Export** object group in the left pane. The middle pane should have a column heading of **Export Objects** for the leftmost column.
3. Select a table or view in the right pane and drag it into the middle pane. Drop the table or view anywhere in the middle pane. The following dialog appears:



Add a New Metadata Object

Define the Type and Name of the New Object.

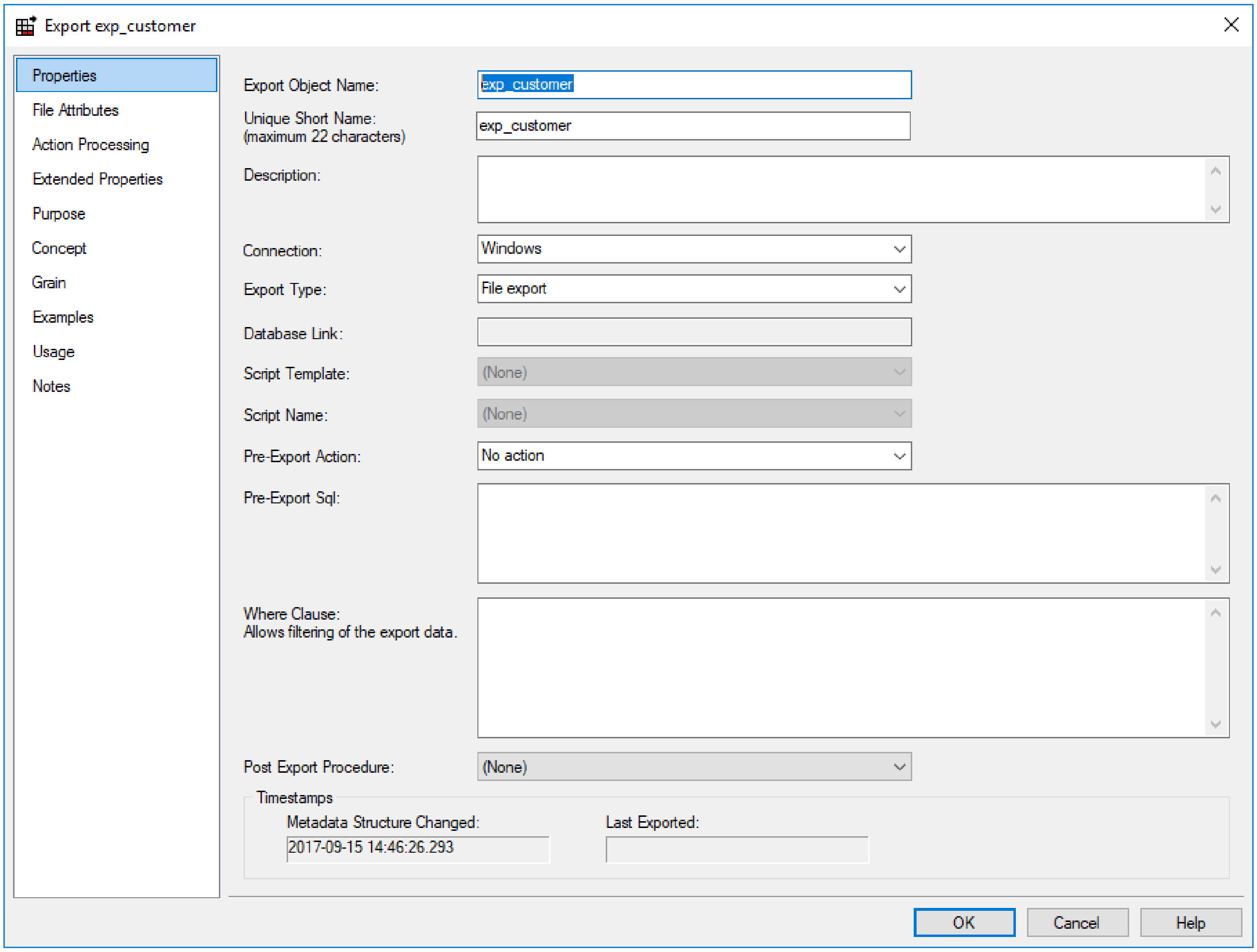
Specific information for each object type is defined in subsequent screens.

Object Type: Export

Object Name: exp_customer

4. If the export object needs to be renamed, rename it and then click **ADD**.

   The Properties window appears.



Export exp_customer

Export Object Name: exp_customer

Unique Short Name: (maximum 22 characters) exp_customer

Description:

Connection: Windows

Export Type: File export

Database Link:

Script Template: (None)

Script Name: (None)

Pre-Export Action: No action

Pre-Export Sql:

Where Clause: Allows filtering of the export data.

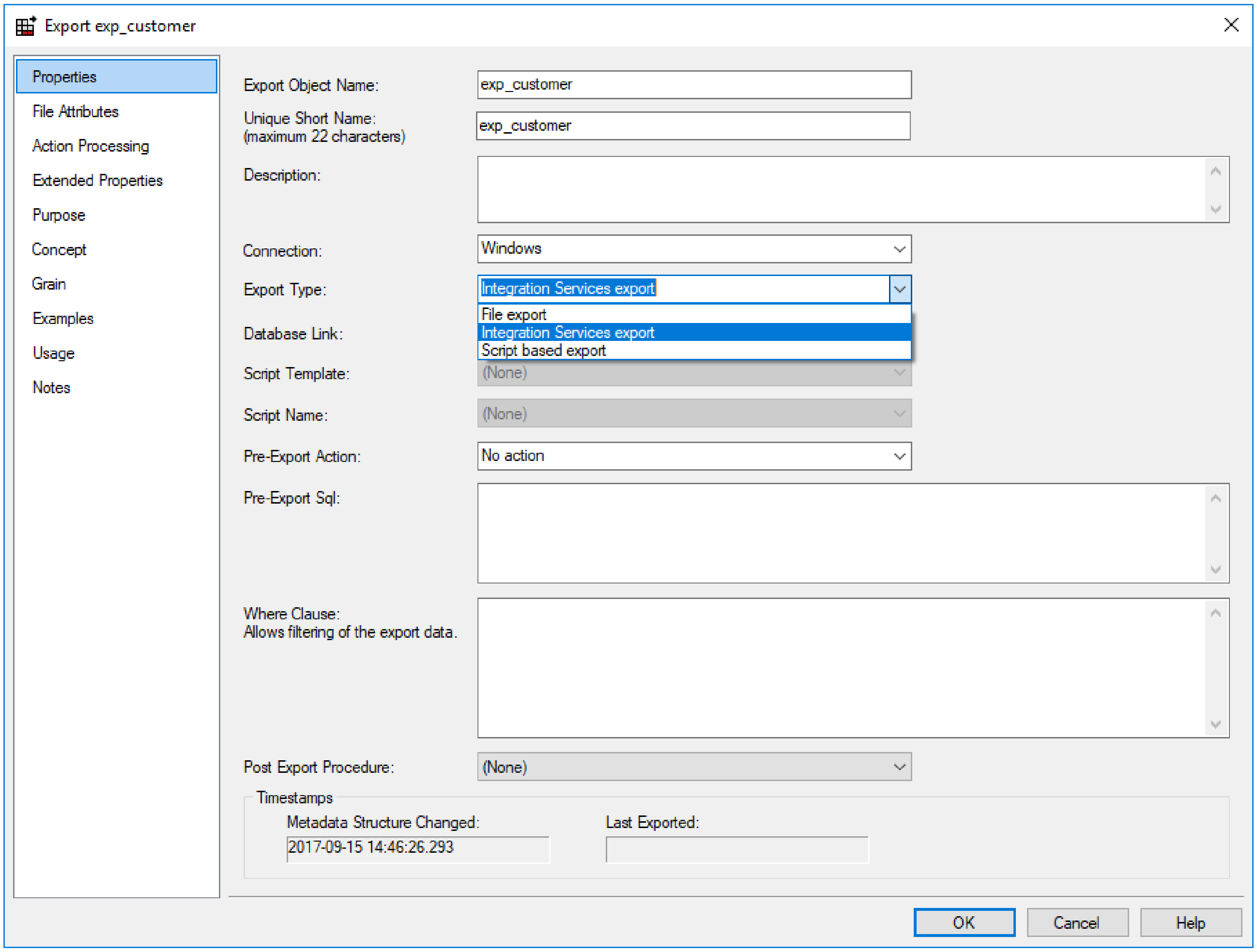
Post Export Procedure: (None)

Timestamps

Metadata Structure Changed: 2017-09-15 14:46:26.293

Last Exported:

5. Select the **Connection** that you want to perform the export from the Connection drop-down list. In this example, the Connection is **Windows.**
   - Select the preferred Export type: **File export**, **Script-based export**



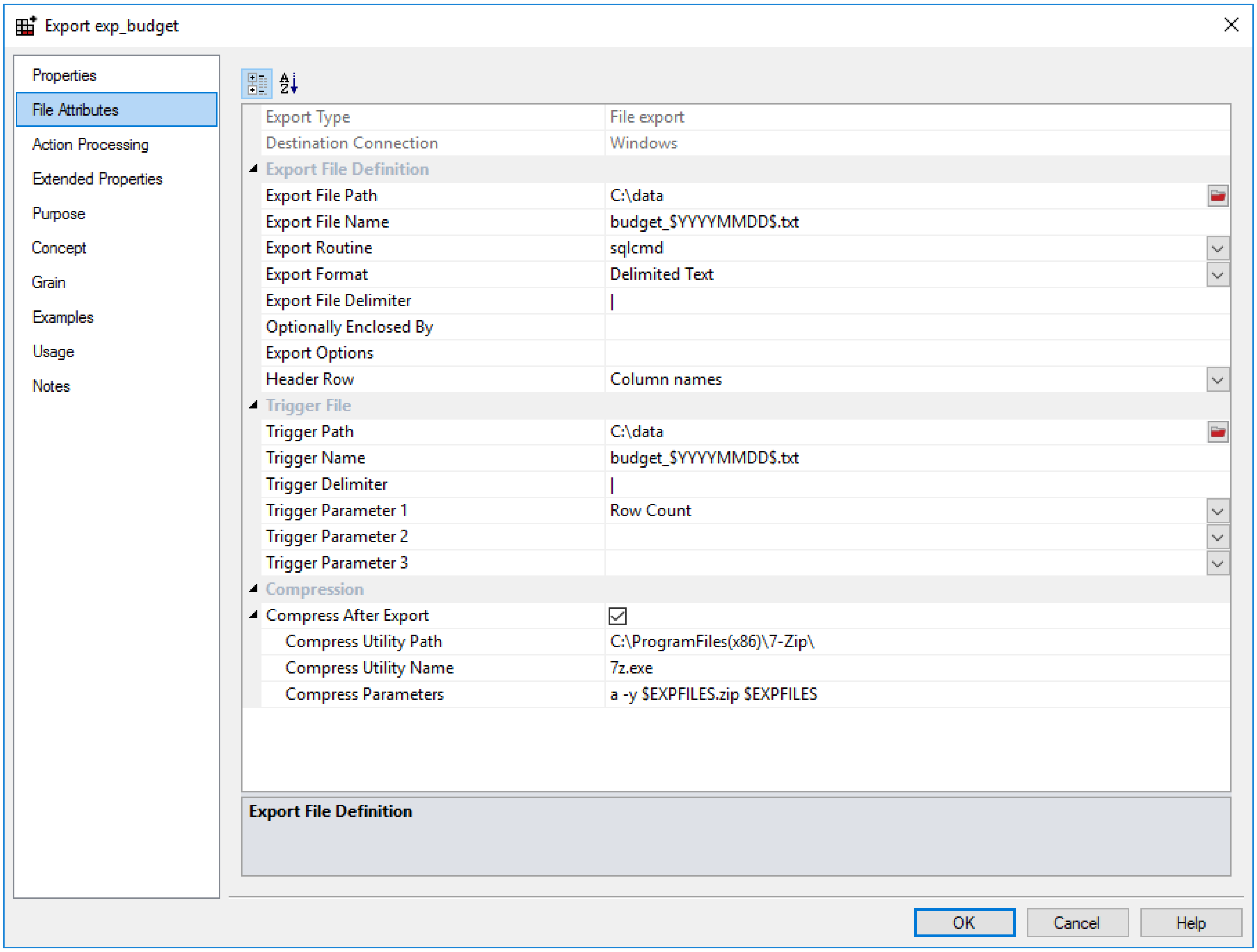
Export Object Name: exp_customer
Unique Short Name: (maximum 22 characters) exp_customer
Description:
Connection: Windows
Export Type: Integration Services export
Database Link:
Script Template: (None)
Script Name: (None)
Pre-Export Action: No action
Pre-Export Sql:
Where Clause: Allows filtering of the export data.
Post Export Procedure: (None)
Timestamps
Metadata Structure Changed: 2017-09-15 14:46:26.293
Last Exported:

6. Click the **File Attributes** tab and fill in the fields as described in the next section.
7. Finally, run the export by right-clicking it and selecting **Export**.

File Attributes

The following fields are available to define the location, name and contents of the exported data file:

SQL Server File Attributes example screen:



Export Type: File export
Destination Connection: Windows

Export File Definition
Export File Path: C:\data
Export File Name: budget_$YYYYMMDD$.txt
Export Routine: sqlcmd
Export Format: Delimited Text
Export File Delimiter: |
Optionally Enclosed By:
Export Options:
Header Row: Column names

Trigger File
Trigger Path: C:\data
Trigger Name: budget_$YYYYMMDD$.txt
Trigger Delimiter: |
Trigger Parameter 1: Row Count
Trigger Parameter 2:
Trigger Parameter 3:

Compression
Compress After Export: ☑
Compress Utility Path: C:\ProgramFiles(x86)\7-Zip\
Compress Utility Name: 7z.exe
Compress Parameters: a -y $EXPFILES.zip $EXPFILES

Export File Definition

Fields	Description
Export Type	Method of exporting data from the table. The available options are dependent on the Destination connection that can be specified via the Properties page.
Destination Connection	Destination to the file system to which data will be exported. The destination connection can be specified on the Properties screen.

Export File Definition

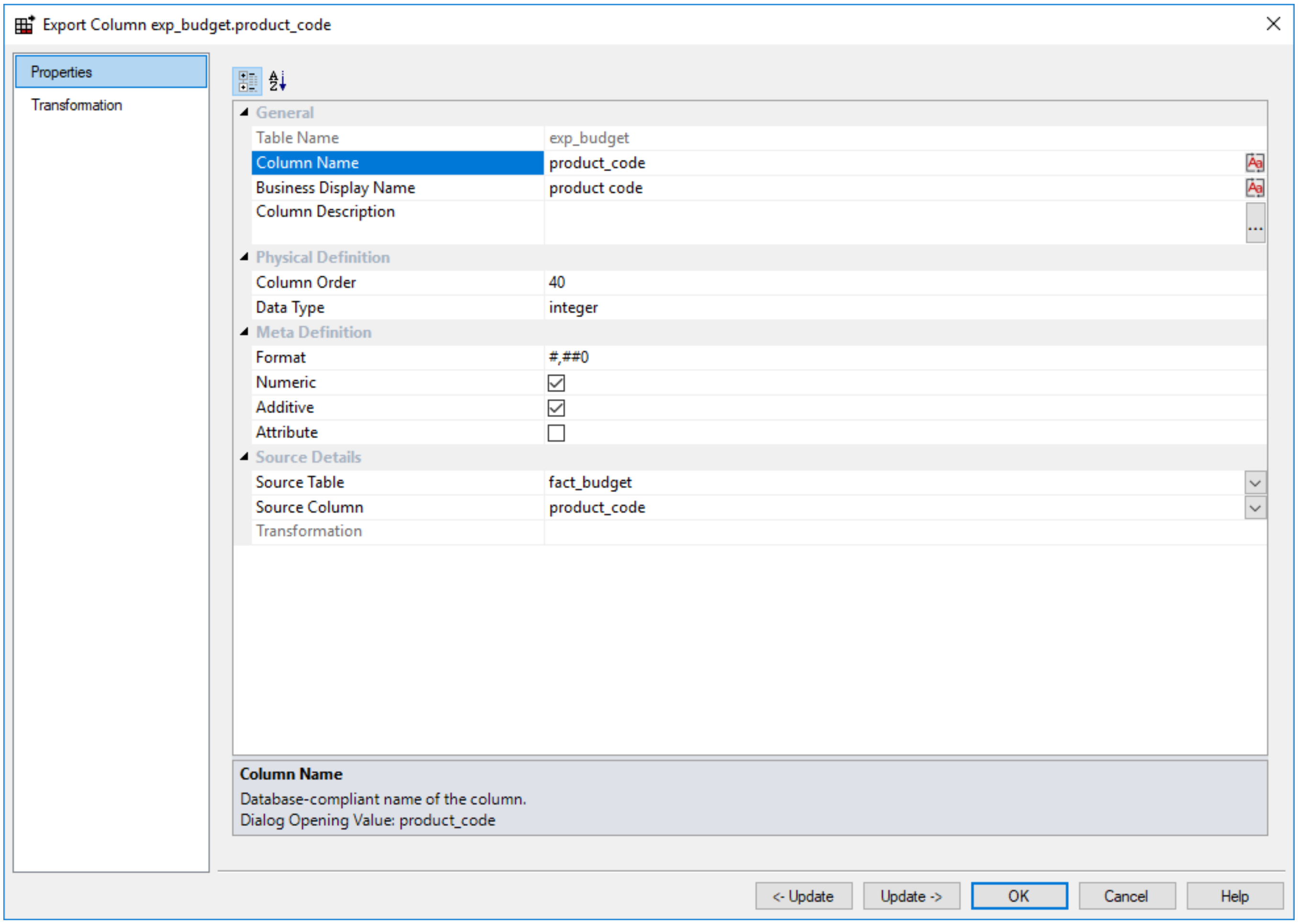
Fields	Description
Export File Path	The full path (absolute path) of the folder/directory where the File is to be created on the Windows or UNIX/Linux system.
Export File Name	Name of the Export File to which the data will be exported. The variable $SEQUENCE$ can be used to provide a unique sequence number for the export file. Also the data/file components YYYY, MM, HH, MI, SS can be used when enclosed with the $ character.For example an export file name might be budget_$YYYYMMDD$.txt which would result in a file name like budget_20150520.txt.
Export Routine	Database-specific routine to use to export the data.
Export File Delimiter	Character that separates the fields within each record of the Export File for Delimited formats. The delimiter identifies the end of which field. Common field delimiters are tab, comma, colon, semi-colon, pipe. To enter a special character enter the uppercase string CHAR with the ASCII value in brackets (e.g. CHAR(9)). This is only available if the Export

Fields	Description
	Format is Delimited Text.
Optionally Enclosed by	Character that brackets text fields within each record of the Export File for Delimited formats. A common example is ". This is only available if the Export Format is Delimited Text.
Header Row in Export	If a header line is required, choose business names or column names from this drop-down list.
Export Options	Allows the entry of export utility options. If more than one option is required, then a semi-colon should be used between options.

Trigger File

Fields	Description
Trigger file	If a trigger file is specified then this file is created after the export file. It will normally contain a row count and a check sum.
Trigger Path	The purpose of the trigger file is to indicate that the export to the main file has completed and that it is now safe to load the file. Secondly the trigger file may contain control sums to validate the contents of the main load file. This field should contain the full path name to the directory in which a trigger file is to be generated on the destination system.
Trigger Name	Refers to the name of the file that is to be created as a trigger file. A trigger file typically contains check sums (row count or the sum of a numeric column). The variable $SEQUENCE$ can be used to provide a unique sequence number for the trigger file. Also the data/file components YYYY, MM, HH, MI, SS can be used when enclosed with the $ character. For example a trigger file name might be budget_$YYYYMMDD$.txt which would result in a file name like budget_20150520.txt.
Trigger Delimiter	Multiple fields in the trigger file are to be separated by the trigger delimiter.
Trigger Parameter 1,2,3	The checksums to be put in the trigger file. One of the row count and the sum of any numeric fields in the source data.

Export Column Properties



General	
Table Name	exp_budget
Column Name	product_code
Business Display Name	product code
Column Description	
Physical Definition	
Column Order	40
Data Type	integer
Meta Definition	
Format	#,##0
Numeric	☑
Additive	☑
Attribute	□
Source Details	
Source Table	fact_budget
Source Column	product_code
Transformation	

Column Name

Database-compliant name of the column.

Dialog Opening Value: product_code

General

Fields	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Business Display Name	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Column Description	This field contains the description for the column. It may be a description from a business

Fields	Description
	user's point of view. This field might additionally contain information on where and how the column was acquired. For example if the column is sourced from multiple tables or is a composite or derived column then this definition would normally describe the process used to populate the column. This field is used in the documentation and is available via the view **ws_admin_v_dim_col .** This field is also stored as a comment against the column in the database.

Physical Definition

Fields	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. For SQL Server common types are integer, numeric, varchar() and datetime. Refer to the database documentation for a description of the data types available. Changing this field alters the table's definition.

Meta Definition

Fields	Description
Format	Not relevant for Export Objects.
Numeric	Not relevant for Export Objects.
Additive	Not relevant for Export Objects.
Attribute	Not relevant for Export Objects.

Source Details

Fields	Description
Source Table	Identifies the source table where the column's data comes from. This source table is normally a stage table within the data warehouse. If the column was sourced from multiple tables, then the normal practice is to record one of the tables in this field and a comment listing all of the other tables in the **Source Strategy** field. This field is used when generating a procedure to update the fact table. It is also used in the track back diagrams and in the documentation.
Source Column	Identifies the source column where the column's data comes from. Such a column is normally a stage table column, which in turn may have been a transformation or the combination of multiple columns. This may also be a dimensional key where a dimension is being joined.
Transformation	Transformation. [Read-only].

Script Based Exports

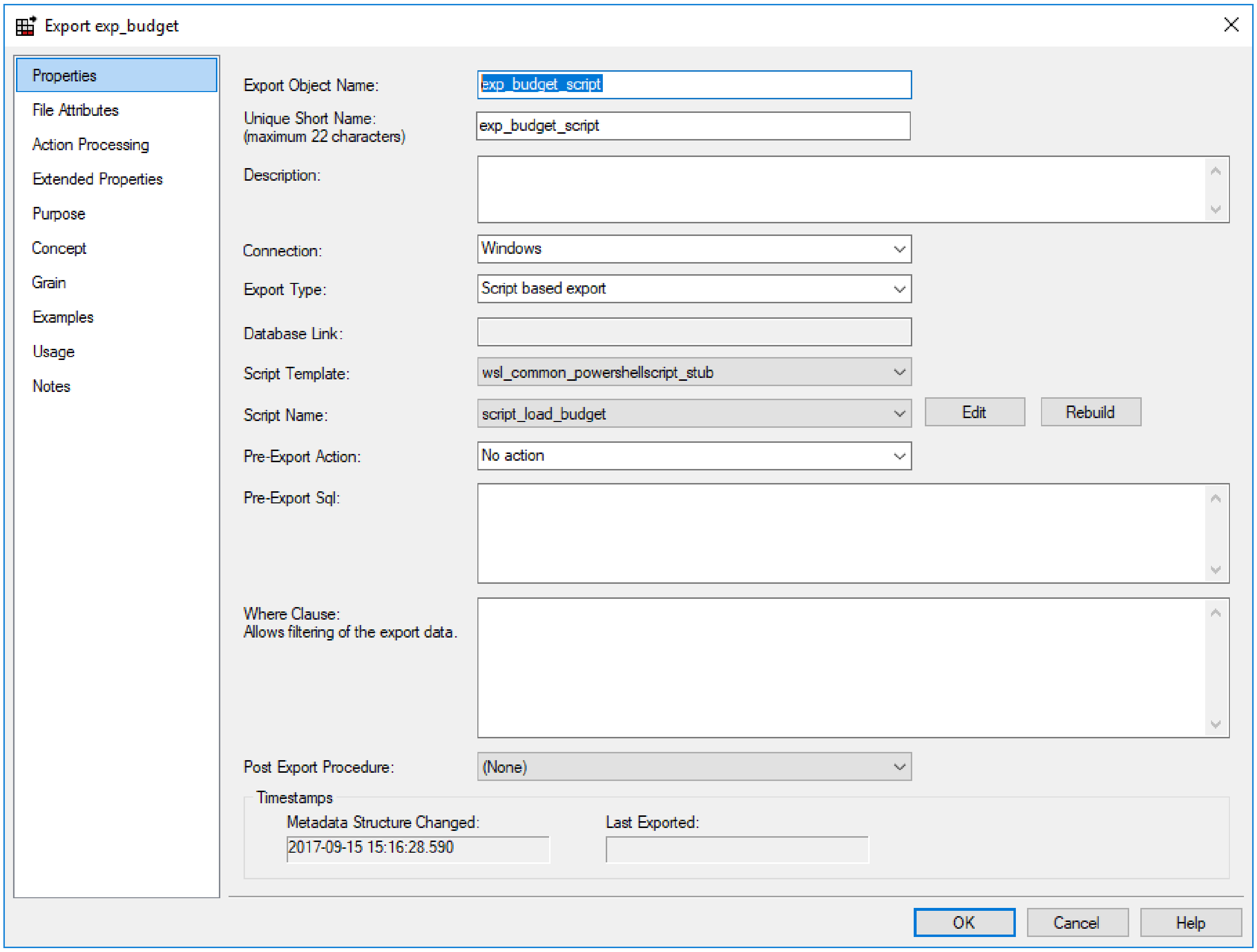
A script based export object will have a Host Script defined. During the export process, the host script is executed and the results returned.
During the **drag and drop** creation of an export object from a single table or view, a script can be generated by selecting one of the 'Script based' export options. This script can then be edited to more fully meet any requirements.

Note

Script-based exports on Windows supports both DOS Batch and **PowerShell scripts**.

There are a number of conventions that must be followed if these host scripts are to be used by the RED Scheduler.

1. The first line of data in **standard out** must contain the resultant status of the script. Valid values are '1' to indicate success, '-1' to indicate a Warning condition occurred but the result is considered a success, '-2' to indicate a handled Error occurred and subsequent dependent tasks should be held, -3 to indicate an unhandled Failure and that subsequent dependent tasks should be held.
2. The second line of data in **standard out** must contain a resultant message of no more than 256 characters.
3. Any subsequent lines in **standard out** are considered informational and are recorded in the audit trail. The normal practice is to place a minimum of information in the audit trail. All bulk information should be output to **standard error.**
4. Any data output to **standard error** is written in the error/detail log. Both the audit log and detail log can be viewed from the WhereScape RED tool under the Scheduler window.
5. When performing **Script based exports**, it is easy to use the **Rebuild** button beside Script Name field to rebuild the scripts.



Export exp_budget

Export Object Name: exp_budget_script

Unique Short Name: (maximum 22 characters) exp_budget_script

Description:

Connection: Windows

Export Type: Script based export

Database Link:

Script Template: wsl_common_powershellscript_stub

Script Name: script_load_budget

Pre-Export Action: No action

Pre-Export Sql:

Where Clause: Allows filtering of the export data.

Post Export Procedure: (None)

Timestamps

Metadata Structure Changed: 2017-09-15 15:16:28.590

Last Exported:

Transformations

Standard **column transformations** can be used in WhereScape RED to perform calculations, change data types or format data.

Re-using complex transformations can save a significant amount of time. These can be achieved in the following ways with WhereScape RED:

- Database Functions
- WhereScape RED Re-usable Transformations
- Teradata User Defined Functions (UDFs)

Column Transformations

Each table or export object column can have a transformation associated with it. For all table types, except for load tables, the transformation will be included in the generated procedure for the table. These are executed as part of the procedure update. The transformation must therefore be a valid SQL construct that can be included in a **Select** statement.
For example, we could have a transformation of 'load_order_line.quantity * 0.125' to calculate a tax column of 12.5%. Click the **Transformation** tab on the Column Properties window to enter a transformation.

Notes:

- Transformations added to an existing table object that have an update procedure are only put into effect when the procedure is re-generated and re-compiled.
- A common transformation can be applied to several columns via the **Change Columns** window.

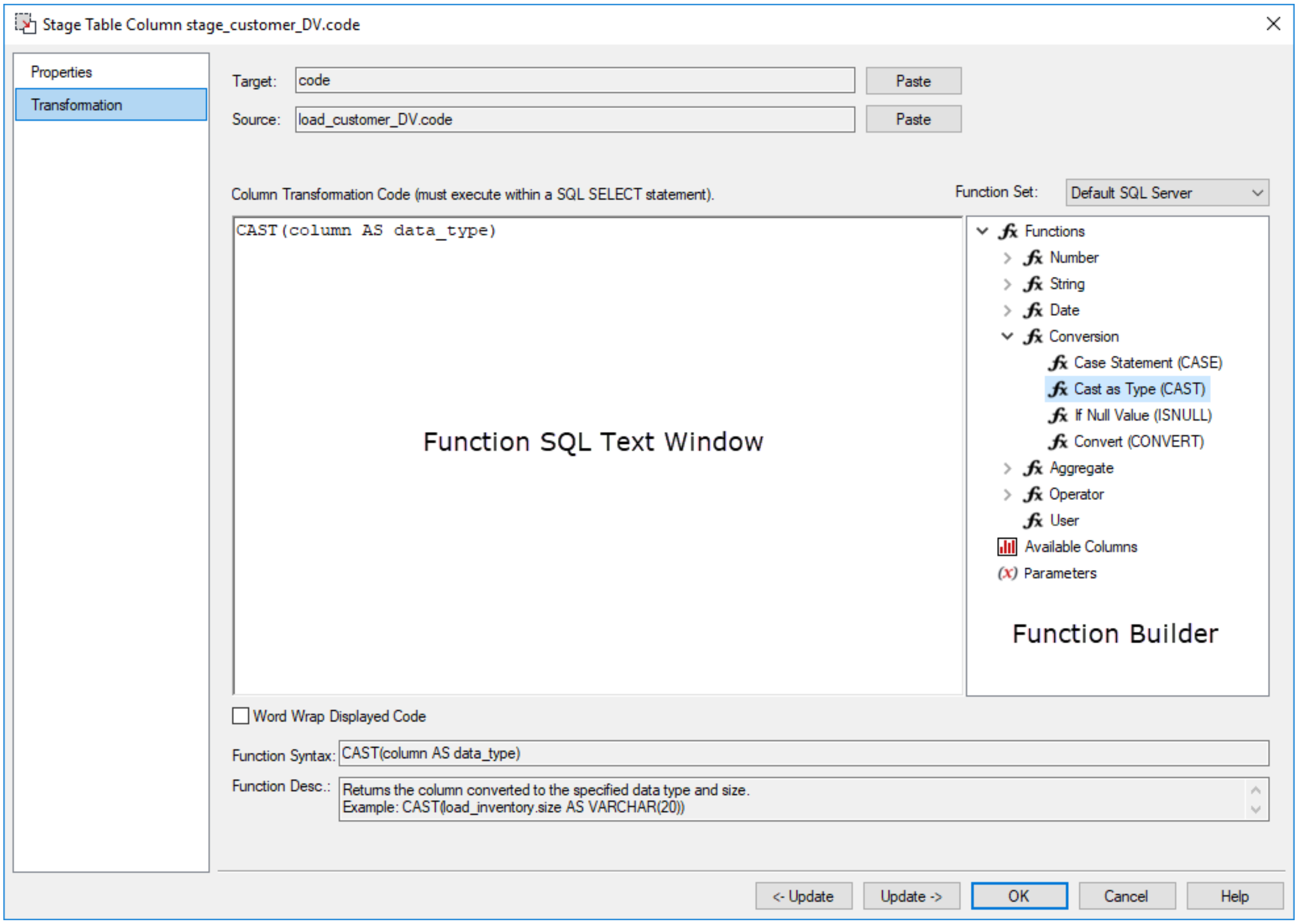
Column transformations in Load tables are more complex, due to the unique nature of Load tables. Refer to **Load Table Column Transformations** for details.

Dimension View transformations are included in the Database Definition Language (DDL) that creates the view in the database. Any changes to Dimension View column transformations require the view to be dropped and recreated.

Export object column transformations are dynamically applied for file exports. If the export object is executed via a Host Script, then the script needs to be regenerated for changes to transformations to take effect. Refer to **Exporting Data** for details.

Column Transformation Properties

An example below shows the transformation **Properties** screen with a simple transformation:



```
CAST(column AS data_type)
```

Function Syntax: CAST(column AS data_type)

Function Desc.: Returns the column converted to the specified data type and size.
Example: CAST(load_inventory.size AS VARCHAR(20))

The two special update keys allow you to update the column and step either forward or backward to the next column's properties. **ALT-Left Arrow** and **ALT-Right Arrow** can also be used instead of the two special update keys.

Fields	Description
Function SQL Text Window	The **Function SQL Text Window** contains the SQL used in the transformation. It can be directly entered, built up using the **Function Builder** and **Add** buttons or a combination of both.
Function Builder	The **Function Builder** contains a list of standard database functions, operators, user defined functions, all columns belonging to all source tables and **parameters** defined in the meta data repository. • Expanding the **Function Heading** displays the **Function Groups** (Number, String, Data, Conversion, etc.) and the **Re-usable Function Heading**. Similarly, expanding the other headings displays the **Available Source Columns** or **Parameters**. • Each function group, source column or parameter can in turn be expanded to show individual **Functions** , **Source Columns** and **Parameters**. • Double-clicking on a function adds the **Function Model** to the **Function SQL Text Window**. The first variable (almost always the source column) is left highlighted in the **Function SQL Text Window**. You can insert additional Functions, Source Columns or Parameters in the correct place within the **Function SQL Text Window** by double-clicking them from the **Function Builder**pane.
Target Paste Button	The **Target Paste** button adds the current column in the form ColumnName to the **Function SQL Text Window** at the location of the cursor.

Fields	Description
Source Paste Button	The **Source Paste** button adds the source table and column in the form TableName.ColumnName to the **Function SQL Text Window** at the location of the cursor.
Transform Stage	Only visible on Load table column transformations. Refer to **Load Table Column Transformations** for details.
Function Set	This drop-down list enables the user to select which set of functions are to be displayed in the tree view when creating a transformation on a column of a table.
Update Buttons	The Update Buttons: **Update <-** and **Update ->** are used to move from the current column to previous and next columns respectively in the current table. The alternative is to exit the Column Transformation Properties, choose the next column, re-enter the Column Properties and choose the **Transformation** tab.
Function Syntax	The syntax guide for the **Function,** visible when you click on the function. Essentially the same as the Function model; loaded into the **Function SQL Text Window** when you click on the function. Read only.
Function Desc	The description of the **Function** visible when the function is clicked. Read only.
Function Model	The model (template SQL code) for a **Re-usable Transformations**. This is only visible for Re-usable Transformations. Read only.
Localize Transformation	The **Localize transformation** button breaks the link between a **column transformation** and the **Re-usable Transformation** it is based on. If this button is clicked, changes to the underlying re-usable transformation cannot be automatically propagated to the column transformation. This is only visible for Re-usable Transformations.

Load Table Column Transformations

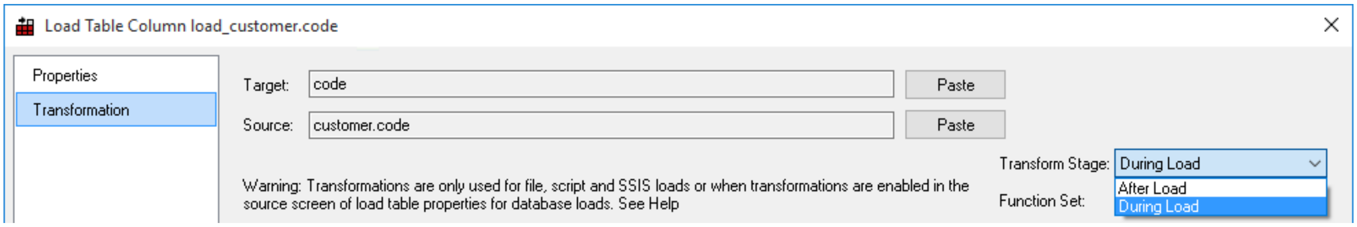
Data entering the data warehouse can be manipulated if required. This manipulation can occur at any stage, but is supported via a number of methods during the **Load** stage. Load tables provide options to transform data. If multiple pass transformations are required, then a Load table can be created from another Load table, e.g. multiple Load tables can be supported in the data flow path.

The options available differ depending on the type of load but in most cases the **after** transformation and post load procedure can be utilized. Specifically:

Database Link Load	• During Load transformations • After Load transformations • Post Load procedure
ODBC Based Load	• During Load transformations • After Load transformations • Post load procedure
File Based Load	• After Load transformations • Post load procedure
Integration Services Load	• During Load transformations • After Load transformations
Script Based Load	• During Load transformations • After Load transformations • Post load procedure

Externally Loaded	• After Load transformations • Post Load procedure

The **Transformation** tab of a column's properties is used to define **during** and **after** load transformations. It can only be one or the other for a specific column. One column can be used to build another, so an **after** can be based on the results of a **during**, if different columns are used.



Load Table Column load_customer.code

Target: code

Source: customer.code

Transform Stage: During Load

Function Set:

Warning: Transformations are only used for file, script and SSIS loads or when transformations are enabled in the source screen of load table properties for database loads. See Help

Note

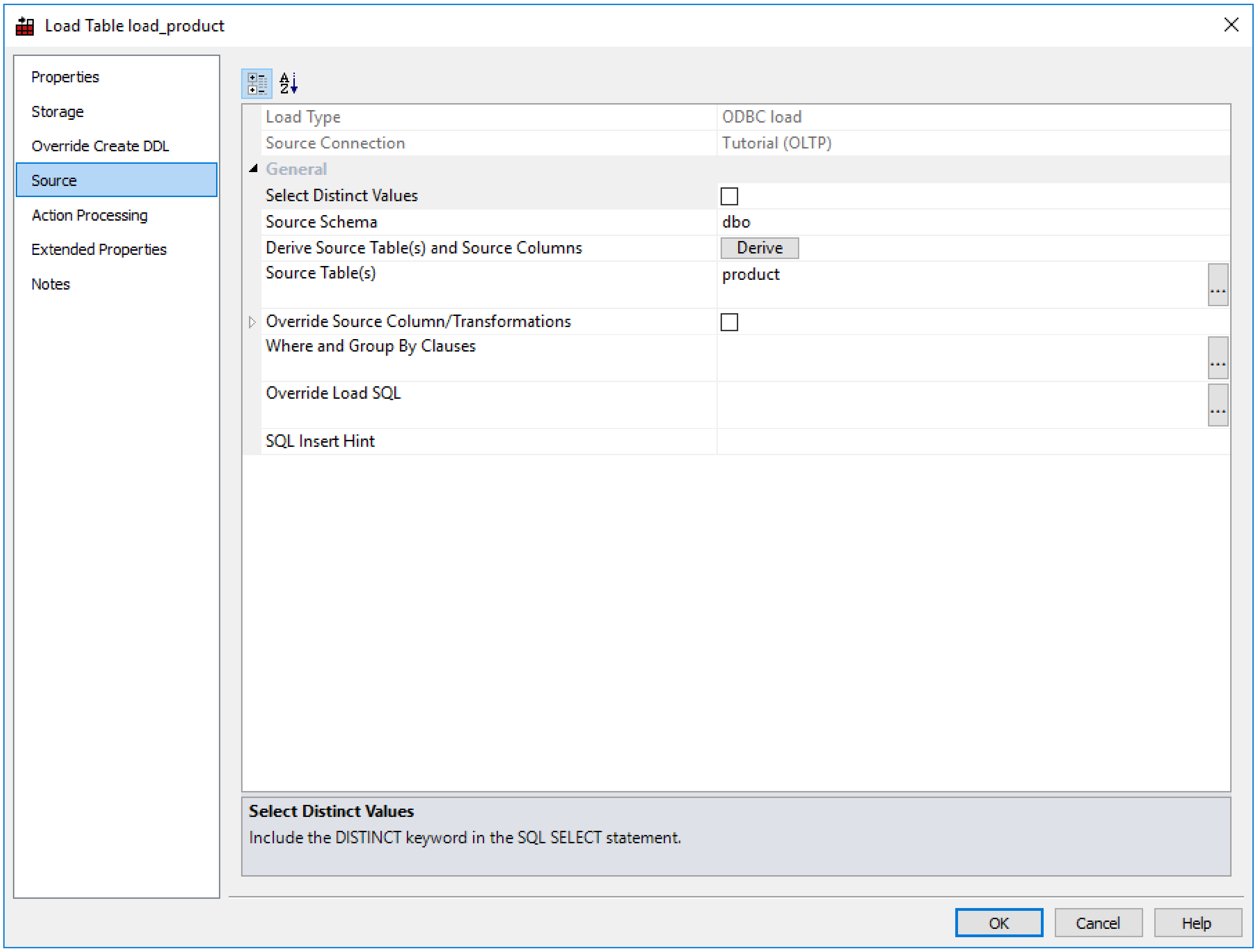
The **During** transformations use **Source Table** columns. The **After** transformations use the **Load Table** columns.

All **After** transformations take place in the native SQL language of the data warehouse database. The **During** transformations differ in terms of which language is used. This is particularly true for file based loads. Normally the 'During' transformation will occur in the native SQL language of the source database.

Database Link During Load Transformations

The **during** load transformation allows the manipulation of column data as it enters the data warehouse.

By default, **Database link loads** and **ODBC based loads** have **During** and **After** transformations enabled.



Load Type	ODBC load
Source Connection	Tutorial (OLTP)
General	
Select Distinct Values	□
Source Schema	dbo
Derive Source Table(s) and Source Columns	Derive
Source Table(s)	product
Override Source Column/Transformations	□
Where and Group By Clauses	
Override Load SQL	
SQL Insert Hint	

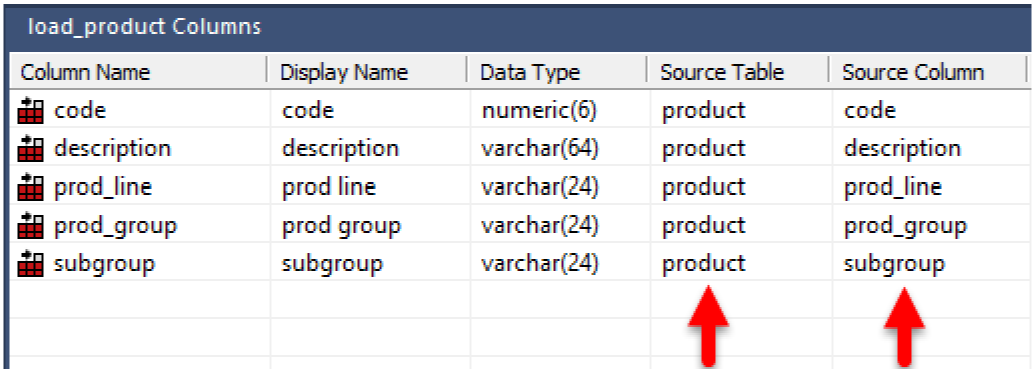
Select Distinct Values
Include the DISTINCT keyword in the SQL SELECT statement.

When transformations are enabled, the contents of a source table/source column for each column are used as the basis of the loading statement.

- If source table and source column are null, then a null is used.
- If data exists in the **Transformation** tab of a column's properties, then this transformation data is used instead of the source table/source column combination.

Example

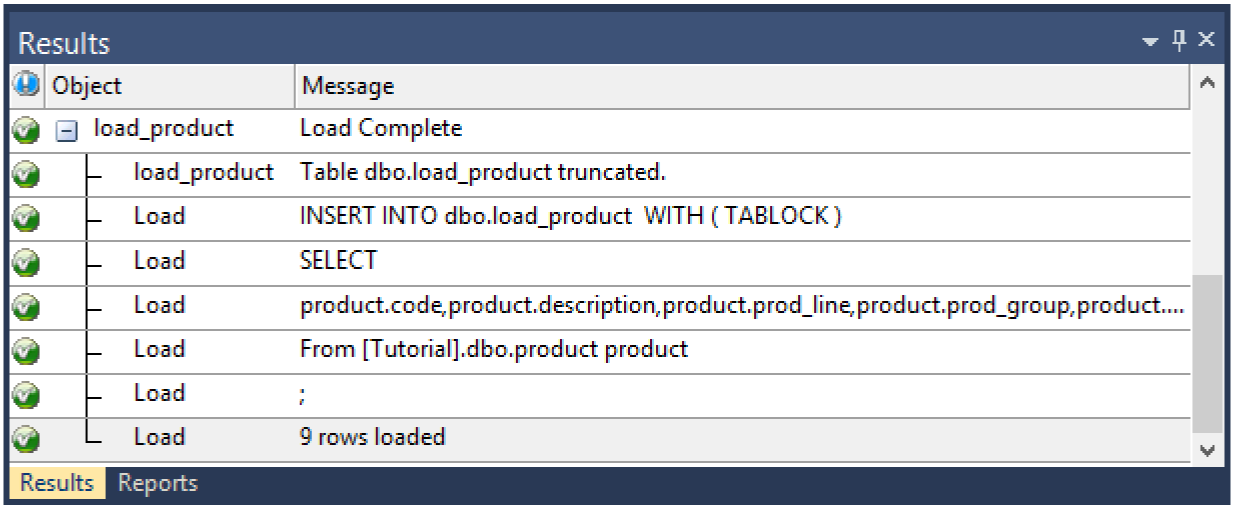
The following load table columns will generate the load sql statement if no transformation data is present against these columns.



load_product Columns

Column Name	Display Name	Data Type	Source Table	Source Column
code	code	numeric(6)	product	code
description	description	varchar(64)	product	description
prod_line	prod line	varchar(24)	product	prod_line
prod_group	prod group	varchar(24)	product	prod_group
subgroup	subgroup	varchar(24)	product	subgroup

The SQL code from the load results is:

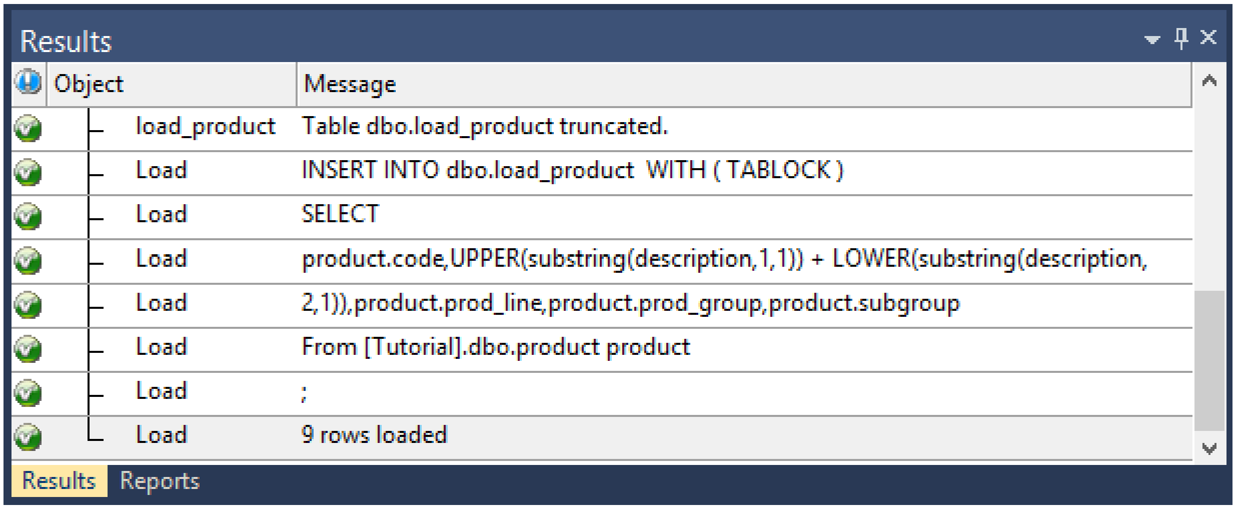


Object	Message
load_product	Load Complete
load_product	Table dbo.load_product truncated.
Load	INSERT INTO dbo.load_product WITH (TABLOCK)
Load	SELECT
Load	product.code,product.description,product.prod_line,product.prod_group,product....
Load	From [Tutorial].dbo.product product
Load	;
Load	9 rows loaded

If the Column has a transformation defined as follows:

UPPER(substring(description,1,1)) + LOWER(substring(description,2,1))

then the following SQL statement will be executed.



Object	Message
load_product	Table dbo.load_product truncated.
Load	INSERT INTO dbo.load_product WITH (TABLOCK)
Load	SELECT
Load	product.code,UPPER(substring(description,1,1)) + LOWER(substring(description,
Load	2,1)),product.prod_line,product.prod_group,product.subgroup
Load	From [Tutorial].dbo.product product
Load	;
Load	9 rows loaded

Database Functions

Database functions can be created in the data warehouse database (as procedure objects in WhereScape RED) and used in column transformation in WhereScape RED. Refer to SQL Server documentation on functions and **Procedures and Scripts** for more information on creating functions inside WhereScape RED.

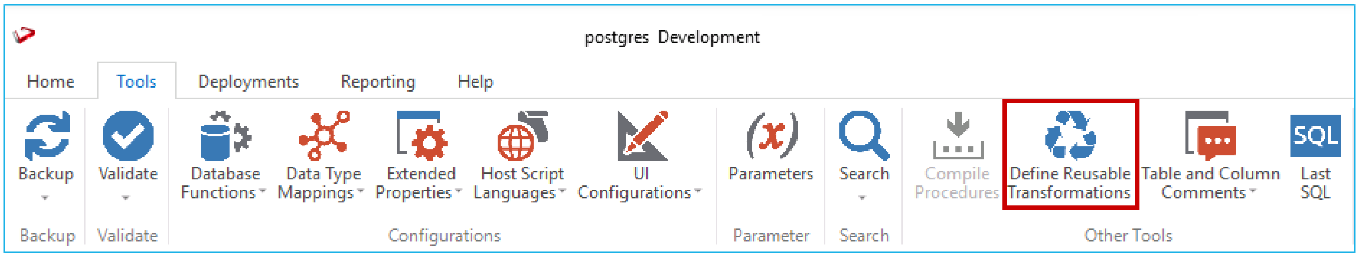
Database function sets can also be created, edited, deleted, imported and exported, using the RED **Database Function sets** tool.

Re-usable Transformations

WhereScape RED Re-usable Transformations allow a complex SQL transformation using standard database functions (including User Defined Functions) to be built once and reused in multiple column transformations. Functionality is included to manage and propagate changes to re-usable transformations.

Creating a New Re-usable Transformation

New **re-usable transformations** are created from the **Tools > Define Re-Usable Transformations** menu.

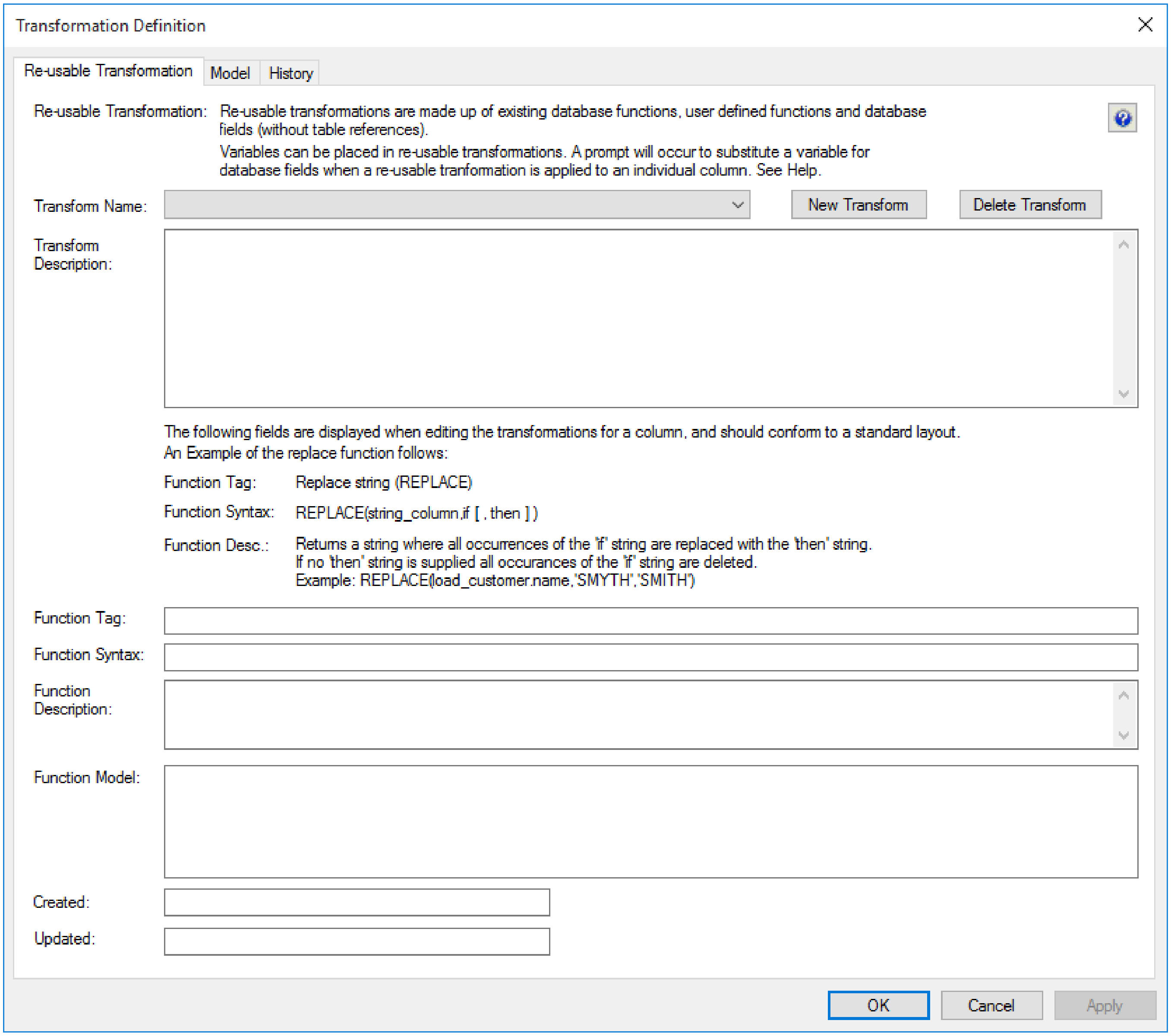


Creating a new re-usable transformation is a three step process:

1. Specify the name of the transformation
2. Enter metadata for the transformation
3. Define the transformation

Specify the Name of the Transformation

After selecting **Define Re-Usable Transformations** from the **Tools** menu, the following screen is displayed:



Transformation Definition

Re-usable Transformation | Model | History

Re-usable Transformation: Re-usable transformations are made up of existing database functions, user defined functions and database fields (without table references).
Variables can be placed in re-usable transformations. A prompt will occur to substitute a variable for database fields when a re-usable tranformation is applied to an individual column. See Help.

Transform Name:

New Transform

Delete Transform

Transform Description:

The following fields are displayed when editing the transformations for a column, and should conform to a standard layout.
An Example of the replace function follows:

Function Tag: Replace string (REPLACE)

Function Syntax: REPLACE(string_column,if [, then])

Function Desc.: Returns a string where all occurrences of the 'if' string are replaced with the 'then' string.
If no 'then' string is supplied all occurances of the 'if' string are deleted.
Example: REPLACE(load_customer.name,'SMYTH','SMITH')

Function Tag:

Function Syntax:

Function Description:

Function Model:

Created:

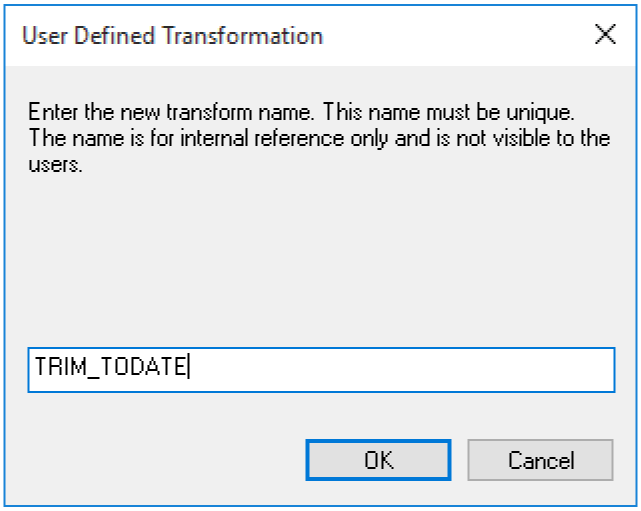
Updated:

OK

Cancel

Apply

Click **New Transform** and enter a name for the re-usable transformation:



User Defined Transformation

Enter the new transform name. This name must be unique. The name is for internal reference only and is not visible to the users.

TRIM_TODATE

Note

This is the internal WhereScape RED name for the transformation, not the name developers reference when utilizing the transformation on column transformations.

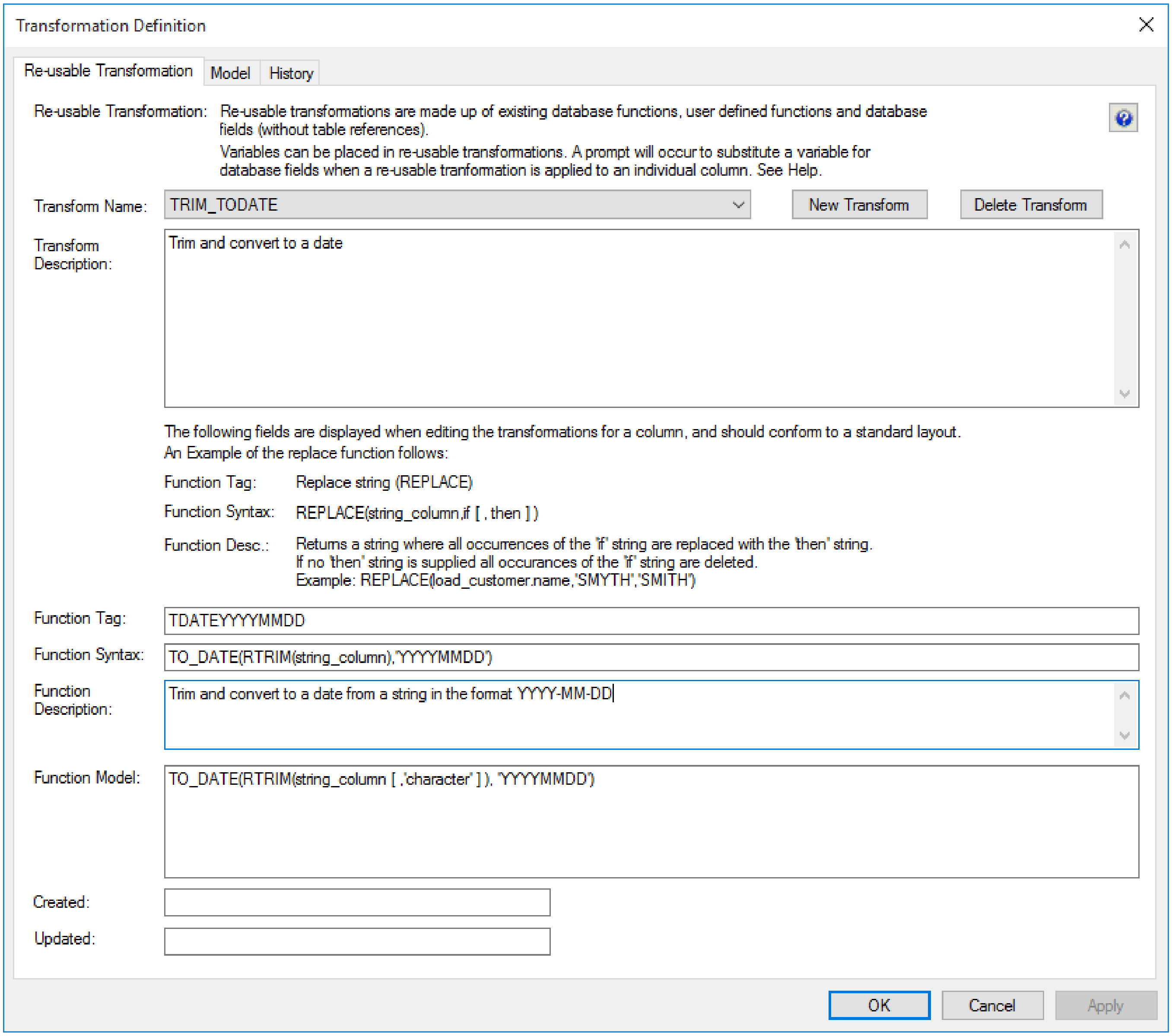
Click **OK**.

Enter Re-usable Transformation Metadata

Enter the following metadata for the transformation to describe the transformation for developers.

Field	Description
Transform Description	A general description of the transformation.
Function Tag	This is the function name visible to the users, which can be selected from the function builder when building column transformations.
Function Syntax	The syntax guide for the function. This is visible in the function builder when clicking on the User defined function.
Function Description	The description of the function visible in the function builder when clicking on the User defined function.
Function Model	The model (template SQL code) for a User Defined Transformations. This is only visible for User Defined Transformations. [Read only].

Completed Re-usable Transformation



Re-usable Transformation: Re-usable transformations are made up of existing database functions, user defined functions and database fields (without table references).

Variables can be placed in re-usable transformations. A prompt will occur to substitute a variable for database fields when a re-usable tranformation is applied to an individual column. See Help.

Transform Name: TRIM_TODATE

Transform Description: Trim and convert to a date

The following fields are displayed when editing the transformations for a column, and should conform to a standard layout.
An Example of the replace function follows:

Function Tag: Replace string (REPLACE)

Function Syntax: REPLACE(string_column,if [, then])

Function Desc.: Returns a string where all occurrences of the 'if' string are replaced with the 'then' string.
If no 'then' string is supplied all occurances of the 'if' string are deleted.
Example: REPLACE(load_customer.name,'SMYTH','SMITH')

Function Tag: TDATEYYYYMMDD

Function Syntax: TO_DATE(RTRIM(string_column),'YYYYMMDD')

Function Description: Trim and convert to a date from a string in the format YYYY-MM-DD

Function Model: TO_DATE(RTRIM(string_column [,'character']), 'YYYYMMDD')

Created:

Updated:

Changing a Re-usable Transformation

To change a re-usable transformation:

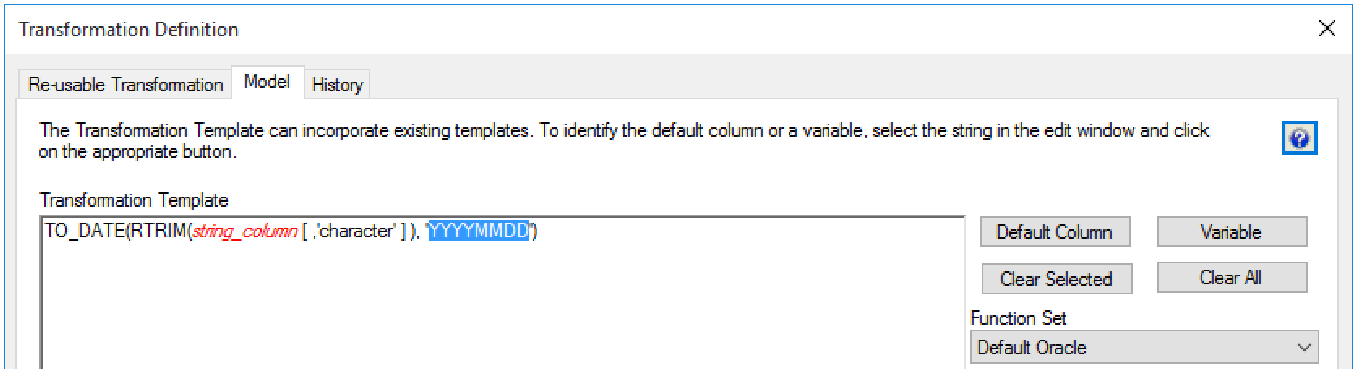
1. Select **Re-Usable Transformations** from the **Tools** menu.
2. Choose the transformation from the **Transform Name** drop-down list.
3. Click on the **Model** Tab.
4. Change the SQL as required.
5. Click **OK.**

Example of a change to the Model SQL

In the example used in Creating a New Re-usable Transformation, the SQL was:

TO_DATE(RTRIM(string_column),'YYYYMMDD')

Change the SQL to allow the format to be specified when the transformation is used by changing **YYYYMMDD** to **format**.



The Transformation Template can incorporate existing templates. To identify the default column or a variable, select the string in the edit window and click on the appropriate button.

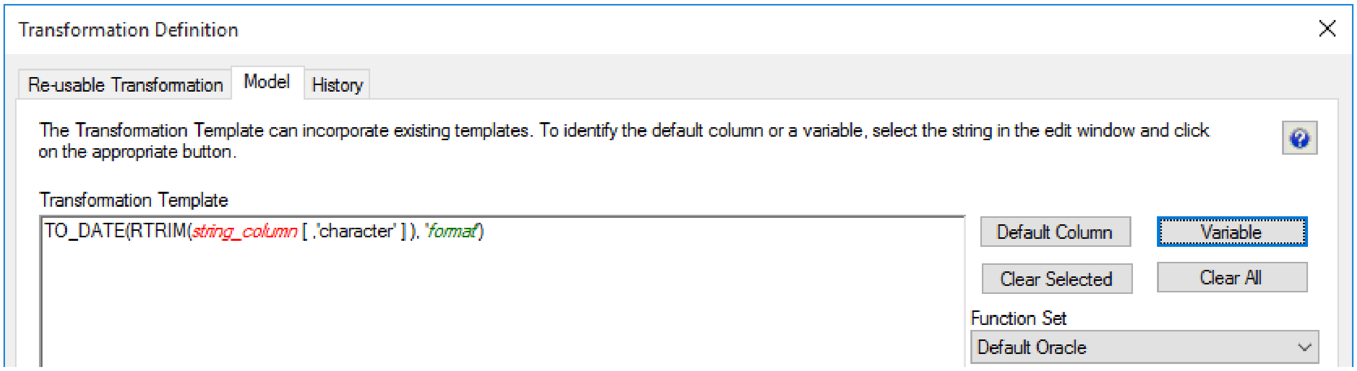
Transformation Template

```
TO_DATE(RTRIM(string_column [ ,'character' ] ), 'YYYYMMDD')
```

Function Set: Default Oracle

Then highlight the word **format** and click on the **Variable** button. This makes the word **format** a variable than can be substituted when the Re-usable Transformation is used.

Now **format** is green and in italics:



The Transformation Template can incorporate existing templates. To identify the default column or a variable, select the string in the edit window and click on the appropriate button.

Transformation Template

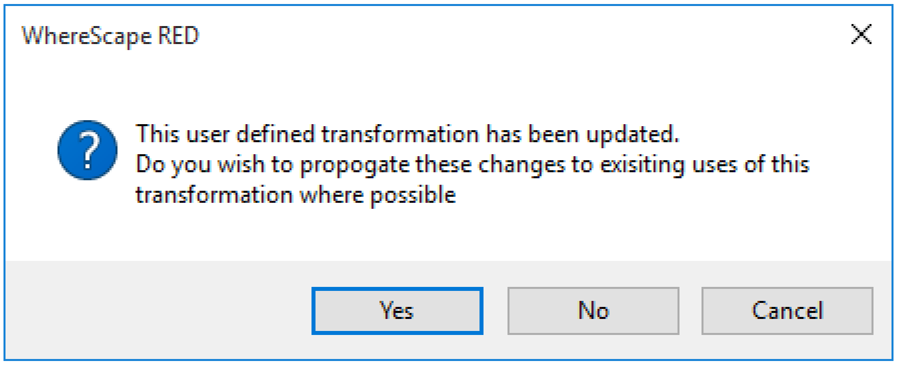
```
TO_DATE(RTRIM(string_column [ ,'character' ] ), 'format')
```

Function Set: Default Oracle

Click **OK**.

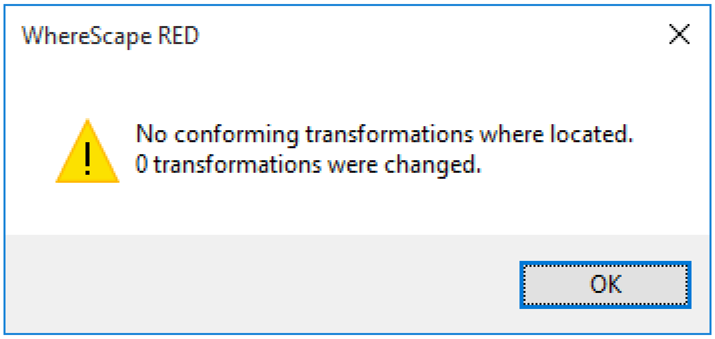
Applying Changes to Dependant Transformations

After changing a **Re-usable Transformation**, a dialog appears asking to confirm that changes should be applied to individual columns using the transformation, where possible:



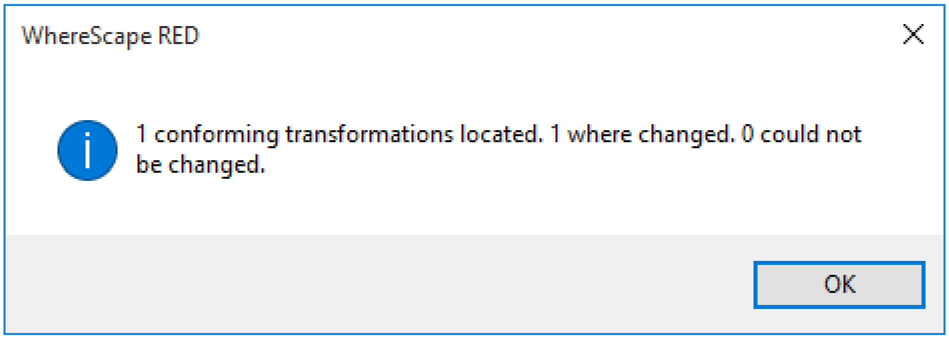
This user defined transformation has been updated.
Do you wish to propogate these changes to exisiting uses of this transformation where possible

If the **Re-usable Transformation** doesn't have any dependant columns, then the following message is displayed:



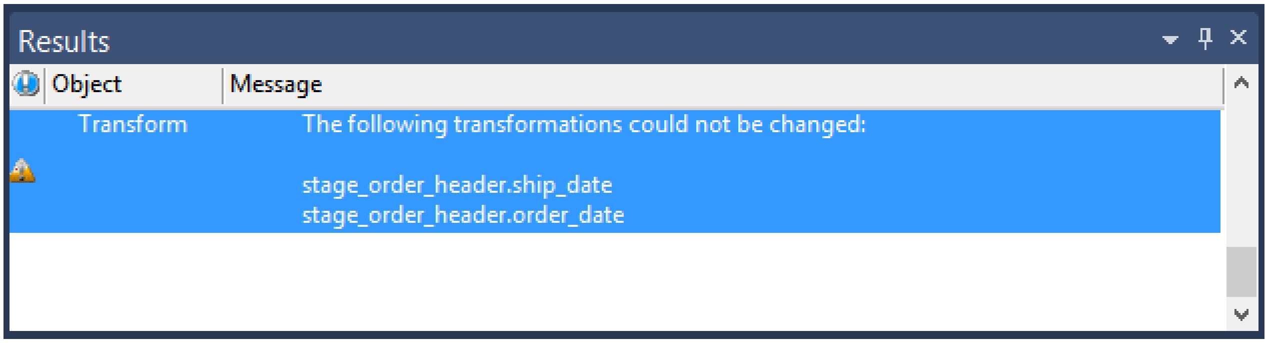
No conforming transformations where located.
0 transformations were changed.

If the **Re-usable Transformation** has been used for dependant columns then this message is displayed:



1 conforming transformations located. 1 where changed. 0 could not be changed.

When an attempt is made to update a dependant Transformation, and the transformation has been modified in such a way as to make it impossible for the changes to the Re-usable Transformation to be applied, the error message above will include a count of the failures. The Results pane will detail the columns (and tables) where the update failures have occurred:



Object	Message
Transform	The following transformations could not be changed: stage_order_header.ship_date stage_order_header.order_date

Using Re-usable Transformations

Re-usable transformations are used exactly the same way as any standard database **Function**. They can be used on any object type. Refer to **Column Transformation Properties** for details.

Procedures and Scripts

WhereScape RED has a **Procedure** object group for database stored procedures and a **Script** object group for host system scripts, such as UNIX shell scripts, Windows batch, PowerShell, Python, and user defined host script languages.

Procedures

WhereScape RED generates the bulk of the procedures via templates for supported Enablement Pack Platforms during a data warehouse build, these templates can be customized as required. Generated procedures can then be modified to meet specific requirements. The procedure object group refers to the concept of database-stored procedures.

Procedure objects in RED are only able to be executed via Action Processing Scripts. Procedure objects of type Procedure and Function must have been compiled before they can be run. The SQL Block subtype does not require compilation.

Scripts

Host scripts are generated either manually or via templates. Most Target Enablement Packs provide a rich set of templates for data warehouse operations that automate the generation of either PowerShell or Python scripts for all object types and supported sources. Scripts can also be created manually provided the rules for their inclusion into the scheduling process are followed.

This chapter covers the generation, editing and compilation of procedures, as well as the generation, editing and testing of host scripts as well as explaining the components required to allow host scripts to work in the scheduler environment.

Procedure Generation

Prerequisites

Procedures can only be generated from templates so you must either have installed an Enablement Pack that supports Procedures, which will have templates included, or create your own procedure templates.

On supported platforms, WhereScape RED generates template procedures to assist in the various phases of the data warehouse build process. A procedure is generated by selecting the type as Procedure and then selecting **(Build Procedure...)** option from a drop-down list field in a table's Properties window to configure the update, custom, or post load procedures.

Procedures can either be manually generated from a RED provided procedure outline or generated leveraging a RED template—refer to **Rebuilding Update Procedures** for details.

If a new procedure is created from scratch (i.e. not auto generated), then an outline of the syntax required by the WhereScape scheduler can be generated by selecting the **Tools > Create Procedure Outline** menu option in the procedure editor.

Wrapper procedures

In some cases, multiple procedures will be required to update a table. In such cases, it is best to create a top level procedure that is seen by the scheduler as the 'Update' procedure. This procedure can in turn, call other procedures.

Procedure Placeholders

The following procedure placeholders are used for compiling procedures to the required target database location and referencing other data warehouse objects defined in the RED metadata. Placeholders are particularly useful when deploying between environments ensuring the objects referenced are fully qualified to suit the environment being deployed to:

1. **[SCHEMA.procedure_name].procedure_name** = resolves to **schema.procedure_name** during the compilation process of the procedure. This placeholder is used in the CREATE PROCEDURE statement to resolve the schema location for the procedure.
2. **[PROCEDUREOWNER].[procedure_name] =** resolves to **database.schema.procedure_name** during the compilation process of the procedure. This placeholder is used to fully qualify procedures, based on the target location of the referenced procedure.
3. **[TABLEOWNER].[table_name] =** resolves to **database.schema.table_name** or **schema.table_name** depending on the target location settings of the referenced object.

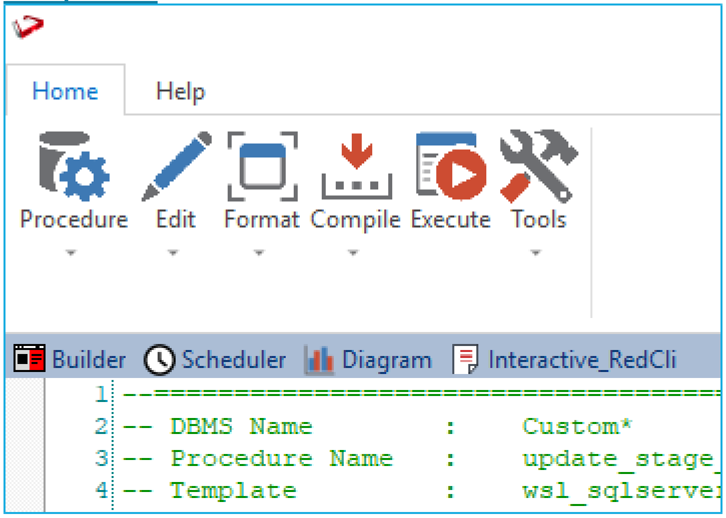
Procedure Editing

WhereScape RED includes a procedure editor which allows the maintenance of the various procedures, functions and packages within the data warehouse. The editor is invoked by double-clicking on a procedure name in the left pane or by right-clicking on the procedure name and selecting **Edit the Procedure**.

This section discuss some of the features of the procedure editor which includes the following:

- syntax highlighting
- improved find feature (repeated find, up and down, etc.)
- toggle the display of line numbering
- a status bar that includes line number, line length, selected text length, etc.
- a context sensitive toolbar
- tab to space conversion and vice versa
- uppercase, lowercase, titlecase text conversion command
- increase/decrease test indent command
- bookmark support

A procedure can be compiled by selecting the **Compile > Compile Procedures** menu option. Refer to **Procedure Compilation** for details.



```
--====================================
-- DBMS Name          :    Custom*
-- Procedure Name     :    update_stage_
-- Template           :    wsl_sqlserve
```

In the following sections reference is made to a selected block of text. A selected block of text is a normal windows selection where the text in question is highlighted. Normally achieved by holding down the left mouse button and moving the cursor.

Indenting code

The **Tab** character inserts four spaces into the text. A **Shift+Tab** removes four spaces.

Cut, Copy, Paste and Delete

The normal Windows cut, copy, paste and delete functions are available either through the toolbar or via the right mouse pop-up menu.

Indenting a block of text

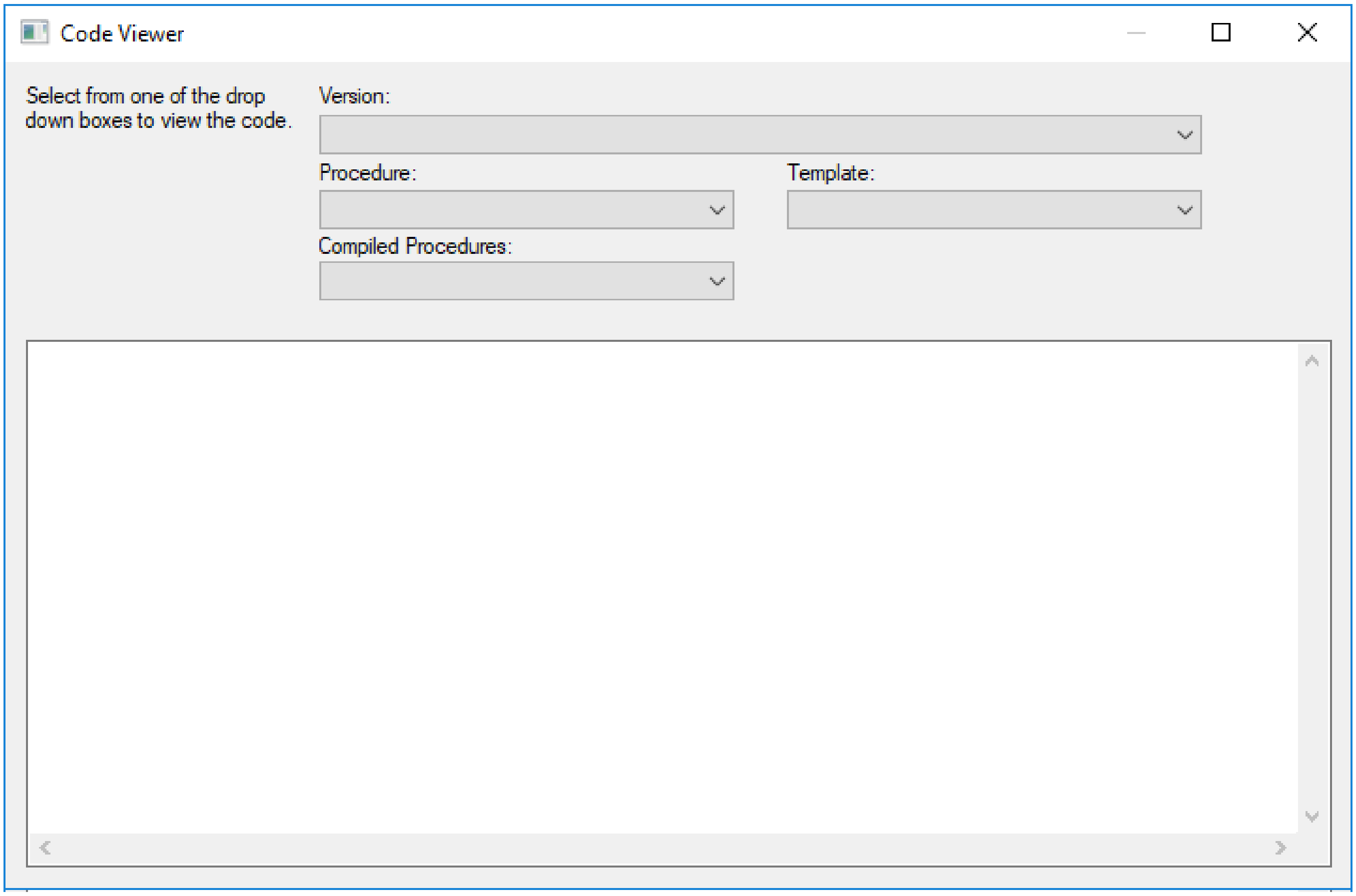
A selected block of text can be indented by four spaces by depressing the **Tab** character. Each tab will indent by a further four spaces. A **Shift+Tab** removes four spaces from the front of each line in the selected block.

Commenting out a block of text

A selected block of text can be turned into a comment by right-clicking and selecting **Comment**. The editor places two dashes '--' at the front of each line in the selected block. In the same way a block of text can be uncommented by choosing the **Uncomment** option.

Viewing other procedural code

During the editing process it is possible to pop up a window containing other procedural code. This window allows cut and paste operations. In such a way it can be used as a work area or as a source of code. Select the **Tools > View Procedure or Template** menu option to bring the viewer window up. A dialog will appear as follows:



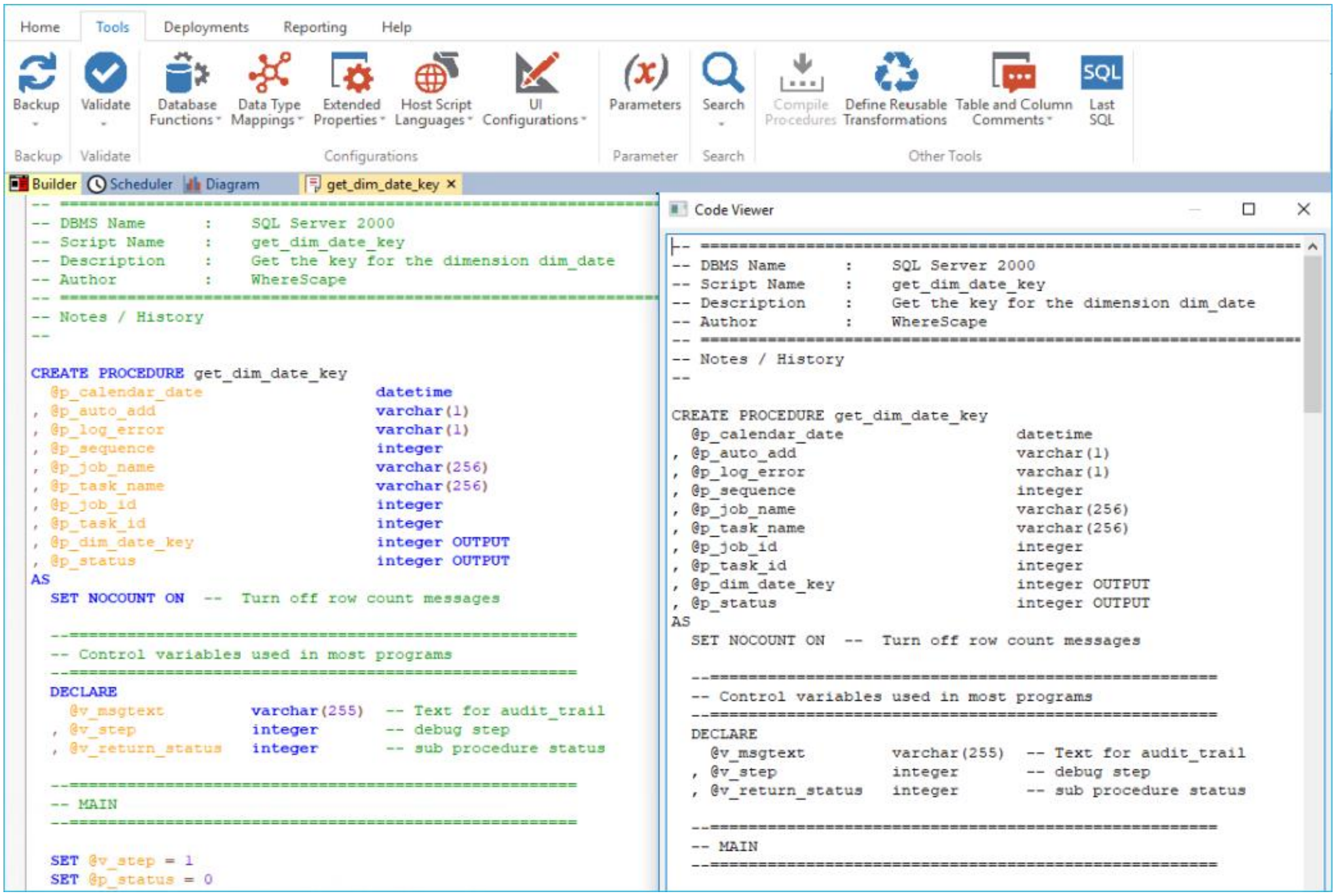
Select from one of the drop down boxes to view the code.

There are a number of drop-down list boxes to choose from. Once an item is selected, the viewer loads the code and moves to the right side of the edit window. The various options are:

- Version: A previously saved version of this procedure. The description and time are shown.
- Procedure: Some other procedure stored within the WhereScape RED metadata.
- Template: A template procedure as defined in the WhereScape RED metadata.

Once an item is chosen, the viewer appears on the right side of the editor window. The viewer or the main editor window can receive the focus by clicking the mouse within the appropriate window.

Code can be cut from one window and pasted into the other. Any changes made in the viewer window cannot be saved. An example of an active view window is as follows:



```
-- ===============================================================
-- DBMS Name      :   SQL Server 2000
-- Script Name    :   get_dim_date_key
-- Description    :   Get the key for the dimension dim_date
-- Author         :   WhereScape
-- ===============================================================
-- Notes / History
--

CREATE PROCEDURE get_dim_date_key
  @p_calendar_date                   datetime
, @p_auto_add                        varchar(1)
, @p_log_error                       varchar(1)
, @p_sequence                        integer
, @p_job_name                        varchar(256)
, @p_task_name                       varchar(256)
, @p_job_id                          integer
, @p_task_id                         integer
, @p_dim_date_key                    integer OUTPUT
, @p_status                          integer OUTPUT
AS
  SET NOCOUNT ON  --  Turn off row count messages

  --=====================================================
  -- Control variables used in most programs
  --=====================================================
  DECLARE
    @v_msgtext          varchar(255)   -- Text for audit_trail
  , @v_step             integer        -- debug step
  , @v_return_status    integer        -- sub procedure status

  --=====================================================
  -- MAIN
  --=====================================================

  SET @v_step = 1
  SET @p_status = 0
```

```
-- ===============================================================
-- DBMS Name      :   SQL Server 2000
-- Script Name    :   get_dim_date_key
-- Description    :   Get the key for the dimension dim_date
-- Author         :   WhereScape
-- ===============================================================
-- Notes / History
--

CREATE PROCEDURE get_dim_date_key
  @p_calendar_date                   datetime
, @p_auto_add                        varchar(1)
, @p_log_error                       varchar(1)
, @p_sequence                        integer
, @p_job_name                        varchar(256)
, @p_task_name                       varchar(256)
, @p_job_id                          integer
, @p_task_id                         integer
, @p_dim_date_key                    integer OUTPUT
, @p_status                          integer OUTPUT
AS
  SET NOCOUNT ON  --  Turn off row count messages

  --=====================================================
  -- Control variables used in most programs
  --=====================================================
  DECLARE
    @v_msgtext          varchar(255)   -- Text for audit_trail
  , @v_step             integer        -- debug step
  , @v_return_status    integer        -- sub procedure status

  --=====================================================
  -- MAIN
  --=====================================================
```

Note

Editing, deleting or compiling Locked for Edit or opened procedures/scripts

- Procedures or scripts cannot be deleted if they are Locked for Edit by any user, checked out by another user or if there is another object that has the same associated procedures or scripts.
- Saving or Compiling in the procedure or script edit window cannot be performed if the procedures or scripts become Locked for Edit by other users after the edit window has been opened.
- Procedures or scripts cannot be deleted or modified after the editor window has been opened, unless the Edit Lock has been released. Edit Locks can be released by any user in the Script or Procedure Properties screen.
- To prevent updates, deletes and modifications to certain procedures or scripts, it is best to use the Check Out functionality instead. For more information about this functionality, refer to **Object Check Outs and Check Ins** for details.

Procedure Loading and Saving

Procedures are normally stored in the WhereScape RED metadata tables. When a procedure is opened within WhereScape RED then the data is retrieved from the meta tables. Likewise when the procedure is saved, it overwrites the existing data in the meta tables.

When a procedure is compiled, it is also written to the specified target database platform.

Loading data

As mentioned above when a procedure is opened for editing the information stored in the metadata is loaded. Additional text can be loaded into the procedure edit window by selecting the **Procedure > Insert from File** menu option which allows the browsing and inserting of a PC based file. Also paste buffer data can be inserted in the normal manner. In the previous 'Editing' section the viewer window was discussed. This window can also be a source of data via cut and paste.

Saving the Procedure

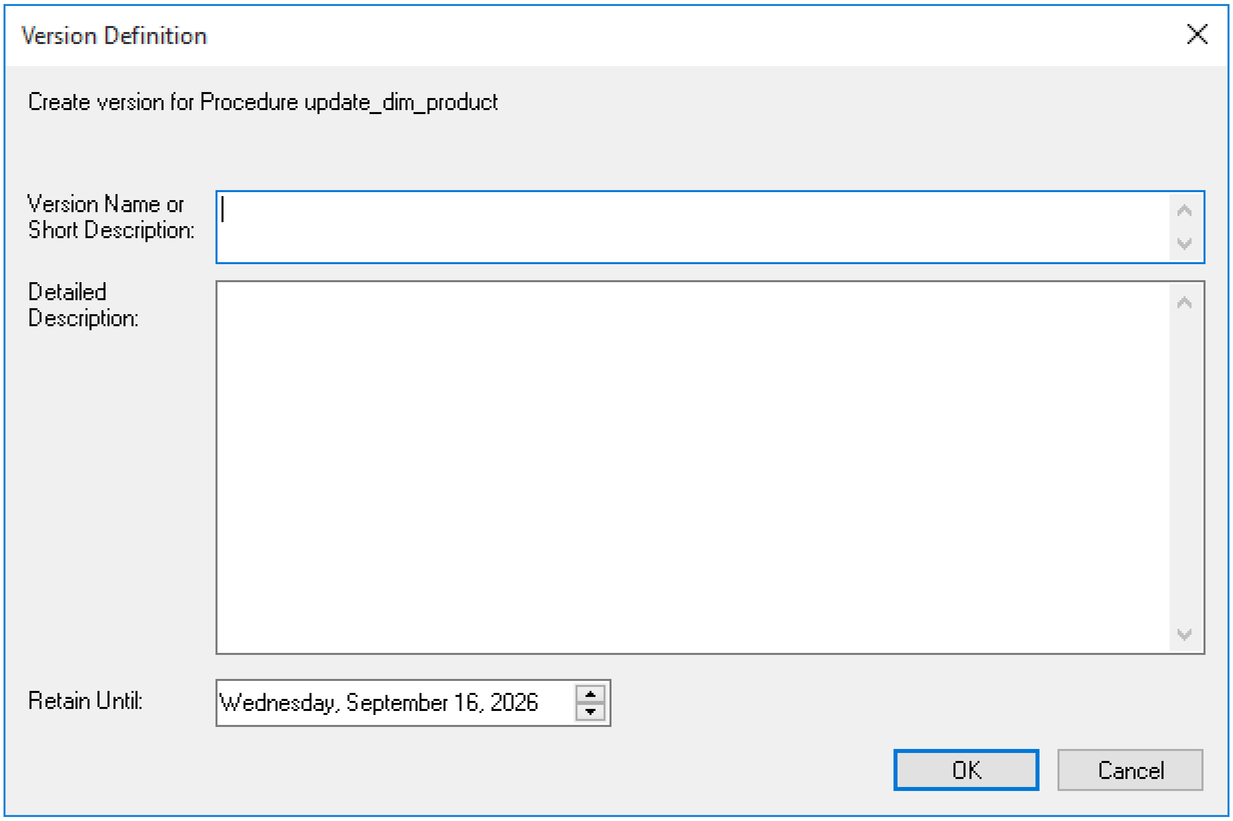
The default **Procedure > Save Procedure** menu option overwrites the existing procedure in the metadata. In addition, a procedure can be saved to a Windows disk by selecting the **Procedure > Write procedure to disk** menu option. All procedures can be written to disk from the main Builder menu option **Backup > Save procedures to disk**. This option allows the selection of a directory. The procedures are then written individually into that directory

Versions

Multiple versions of a procedure can be stored. Once a version is created that version may be read but may not be updated. Only the current procedure can be edited. There are a number of ways of creating a version of a procedure. These are:

1. By setting **Auto-Version before Procedure Compile** under **Home > Options > Metadata Versioning.** If set to true, a new version of a procedure will be created whenever the procedure is compiled.
2. The Procedure Editor menu option **Procedure > Save and Version** will save a procedure and create a version at the same time.
3. By selecting the **Version Control > New Version** menu option from the right-click context menu of a procedure in the main Builder window.
4. By selecting the **Tools > Version Objects** menu option.

When a version is created via method (2) or (3) above the following screen appears, to enable the definition of the version. If an auto version is created, then the person creating the version is recorded along with the reason for the version. (e.g. Version on compile, Version on procedure delete)

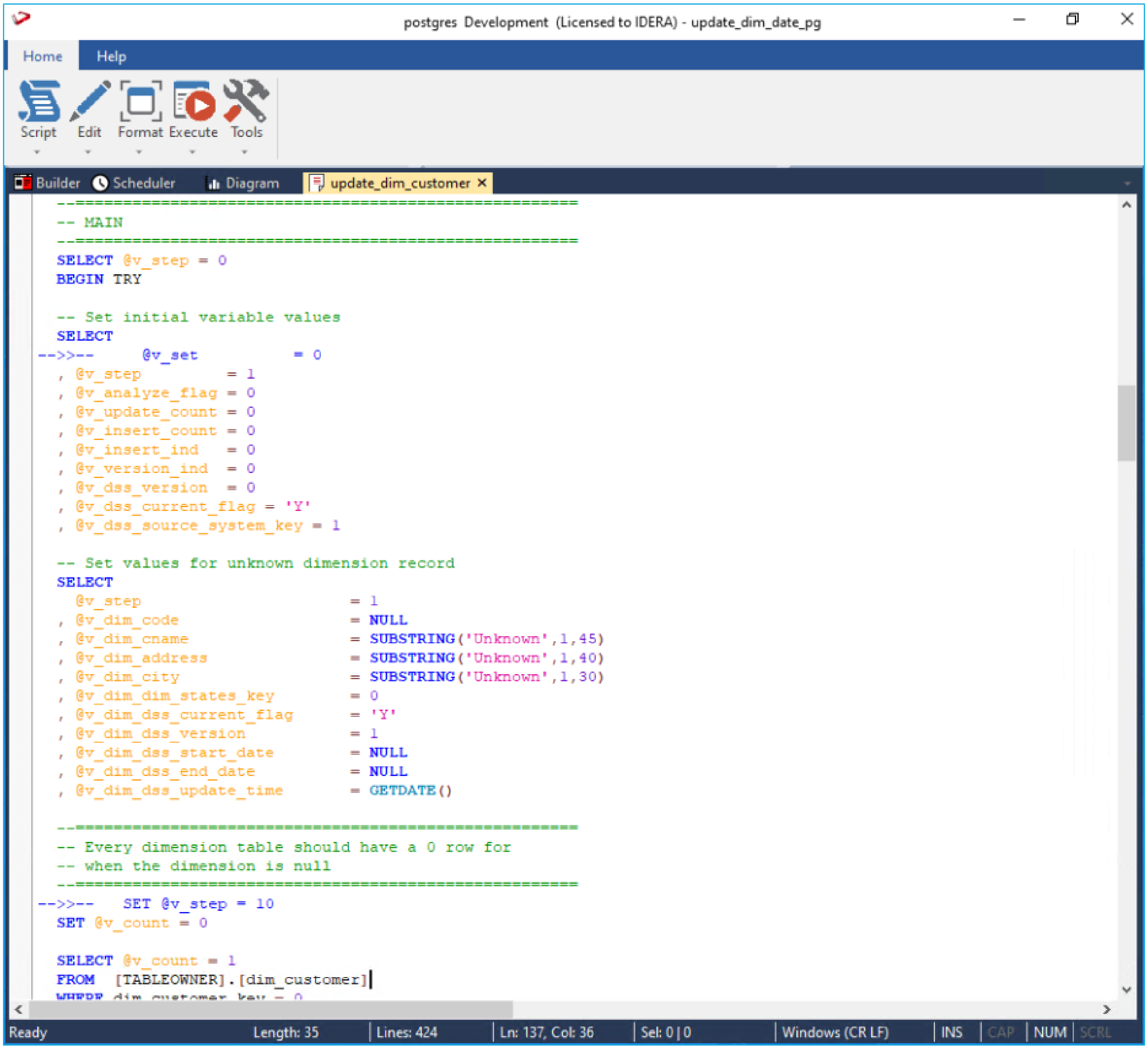

Version Definition
Create version for Procedure update_dim_product
Version Name or Short Description:
Detailed Description:
Retain Until:
OK
Cancel

Retain Until: Wednesday, September 16, 2026

- The version name/description appears when the versions are subsequently browsed.
- The Retain Until date is set ten years in the future by default.
- The automated deletion of versions is not supported.

Procedure Comparisons

A procedure can be compared to an earlier version. If a viewer window is open (see the procedure editing section) then the **Tools > Compare to Viewer** menu option compares the contents of the viewer window with the current code. Therefore, to compare against an older version, we first load the viewer window with the older version and perform a **Compare to Viewer**.



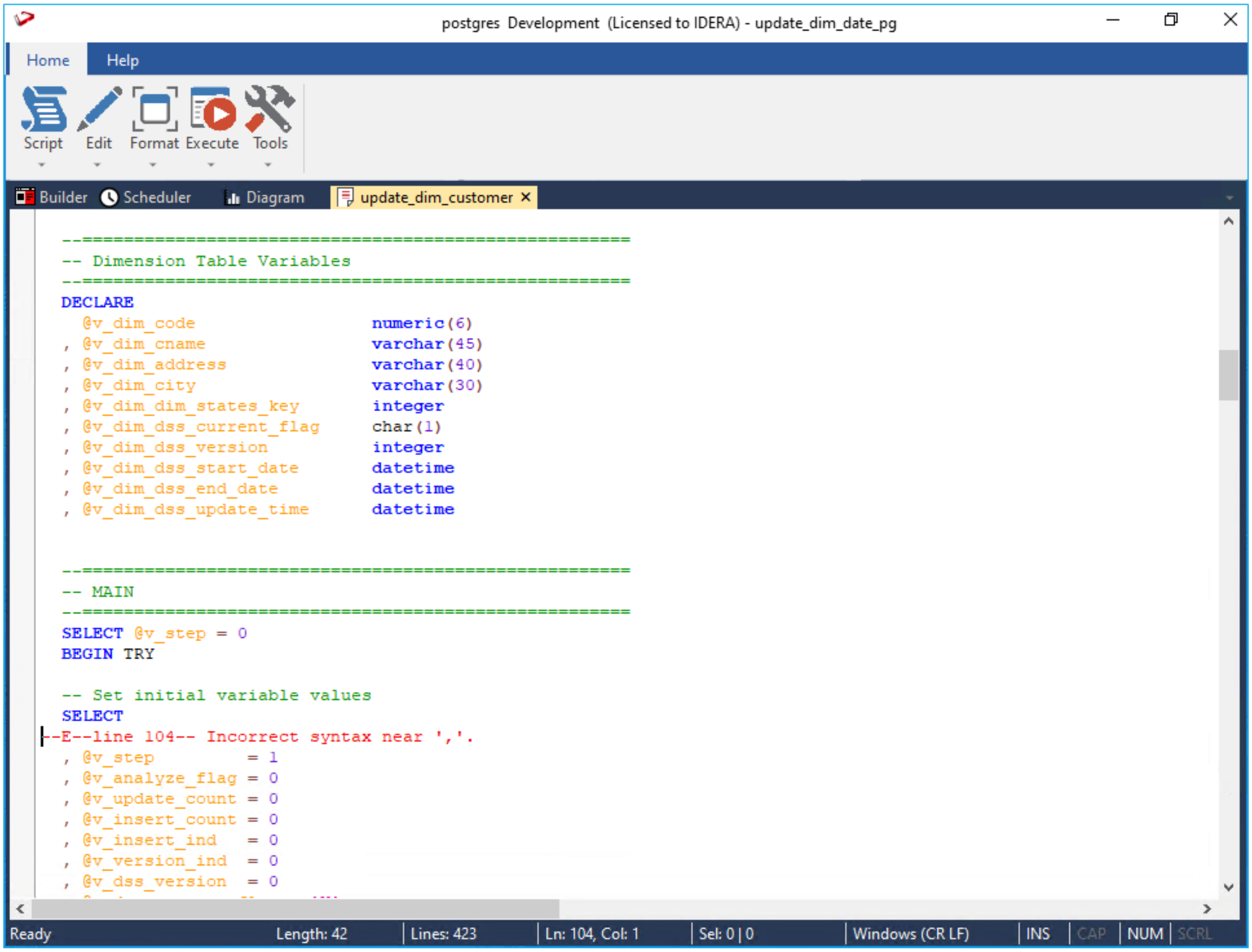
```
   --=====================================================
   -- MAIN
   --=====================================================
   SELECT @v_step = 0
   BEGIN TRY

   -- Set initial variable values
   SELECT
--->>--      @v_set          = 0
   , @v_step          = 1
   , @v_analyze_flag = 0
   , @v_update_count = 0
   , @v_insert_count = 0
   , @v_insert_ind   = 0
   , @v_version_ind  = 0
   , @v_dss_version  = 0
   , @v_dss_current_flag = 'Y'
   , @v_dss_source_system_key = 1

   -- Set values for unknown dimension record
   SELECT
     @v_step                         = 1
   , @v_dim_code                     = NULL
   , @v_dim_cname                    = SUBSTRING('Unknown',1,45)
   , @v_dim_address                  = SUBSTRING('Unknown',1,40)
   , @v_dim_city                     = SUBSTRING('Unknown',1,30)
   , @v_dim_dim_states_key           = 0
   , @v_dim_dss_current_flag         = 'Y'
   , @v_dim_dss_version              = 1
   , @v_dim_dss_start_date           = NULL
   , @v_dim_dss_end_date             = NULL
   , @v_dim_dss_update_time          = GETDATE()

   --=====================================================
   -- Every dimension table should have a 0 row for
   -- when the dimension is null
   --=====================================================
--->>--    SET @v_step = 10
   SET @v_count = 0

   SELECT @v_count = 1
   FROM   [TABLEOWNER].[dim_customer]
   WHERE dim_customer_key = 0
```

In this example, the lines **select @v_step = 0** and **set @v_step = 10** have been removed from the current code in the edit window.

Once the comparison has been completed, you can either remove the compare comments or accept the compare changes. The menu option **Tools > Remove Compare Comments** removes the added blue comments and code. The menu option **Tools > Accept Compare Changes** implements the changes highlighted. For the above example the line **select @v_step = 0** and **set @v_step = 10** would be added.

Procedure Compilation

From within the procedure editor a procedure can be compiled by selecting the menu option **Compile > Compile**. If the procedure compiles successfully, a window appears confirming a successful compile. If the compile fails, then the compile failure message is displayed.



```
--=======================================================
-- Dimension Table Variables
--=======================================================
DECLARE
  @v_dim_code                  numeric(6)
, @v_dim_cname                 varchar(45)
, @v_dim_address               varchar(40)
, @v_dim_city                  varchar(30)
, @v_dim_dim_states_key        integer
, @v_dim_dss_current_flag      char(1)
, @v_dim_dss_version           integer
, @v_dim_dss_start_date        datetime
, @v_dim_dss_end_date          datetime
, @v_dim_dss_update_time       datetime


--=======================================================
-- MAIN
--=======================================================
SELECT @v_step = 0
BEGIN TRY

-- Set initial variable values
SELECT
--E--line 104-- Incorrect syntax near ','.
, @v_step           = 1
, @v_analyze_flag = 0
, @v_update_count = 0
, @v_insert_count = 0
, @v_insert_ind   = 0
, @v_version_ind  = 0
, @v_dss_version  = 0
```

Depending on the target platform, error comments may be is inserted at each error point. **A compile will delete any previous error comments**. Error comments can also be removed through the menu option **Compile > Delete Error messages.**

Note

In some instances, the error comments may not be positioned on the correct line. This can occur as the result of one or more procedure lines being wrapped. Therefore, ensure the procedure editor window is maximized when dealing with compile errors.

If a procedure fails to compile then it is invalidated in the database, and will not run until successfully compiled.

Procedure Running

Procedure objects in RED are only able to be executed via Action Processing Scripts. Procedure objects of type Procedure and Function must have been compiled before they can be run. The Block sub-type does not require compilation.

For procedure based Update, Custom, Post Load or Post Export routines the Action Processing Script associated to the table object must contain the code to execute the procedure and return the results. Generally table action processing scripts are generated by default (via templates) for tables when a relevant change is made that will affect the action script. It is important to ensure your action script is generated if you have added a procedure based routine to the object.

Update, Custom, Post Load and Post Export procedures should be executed via the parent object's context menu.

Standalone procedures require their Action Processing Scripts to be manually generated (or an existing script selected) via the Procedure Properties - Action Processing tab. Once a Procedure object has an Action Script associated it can be run directly through the procedure object context menu or via the Procedure Editor window.

Select the **Execute > Execute** menu option or click the Execute icon to run the procedure. The results of the procedure execution are displayed in the main builder results window. The result code and result message is displayed, as well as any additional messages.

The results of the procedure is displayed in the main builder result window.

Procedure Syntax

The procedures managed by the WhereScape RED scheduler require the following general standards, however, each target platform may have specific variations of this standard which will be documented in the Enablement Pack user guide. If a function or procedure is being developed that is not called directly by the scheduler, then it does not need to conform with this standard. If however, such a procedure or function wants to log messages to the audit or error logs then it needs the input parameters included in its parameter list.

Note

The WhereScape RED procedure editor includes support for syntax highlighting.

Parameters

The procedure must have the following parameters in the following order:

Parameter name	Input/Output	Example Data Type
p_sequence	Input	Integer
p_job_name	Input	Varchar(256)
p_task_name	Input	Varchar(256)
p_job_id	Input	Integer
p_task_id	Input	Integer
p_return_msg	Output	Varchar(1024)
p_status	Output	Integer

The input parameters are passed to the procedure by the scheduler. If the procedure is called outside the scheduler then the normal practice is to pass zero (0) in the sequence, job_id and task_id. A description of the run can be passed in the job name and the task name is typically the name of the procedure.

The output parameters must be populated by the procedure on completion. The return_msg can be any string up to 1024 characters long that describes the result of the procedures execution. The status typically should be one of the following values:

Status	Meaning	Description
1	Success	Normal completion
-1	Warning	Completion with warnings
-2	Error	Hold subsequent tasks dependent on this task
-3	Fatal Error	Hold all subsequent tasks

Note

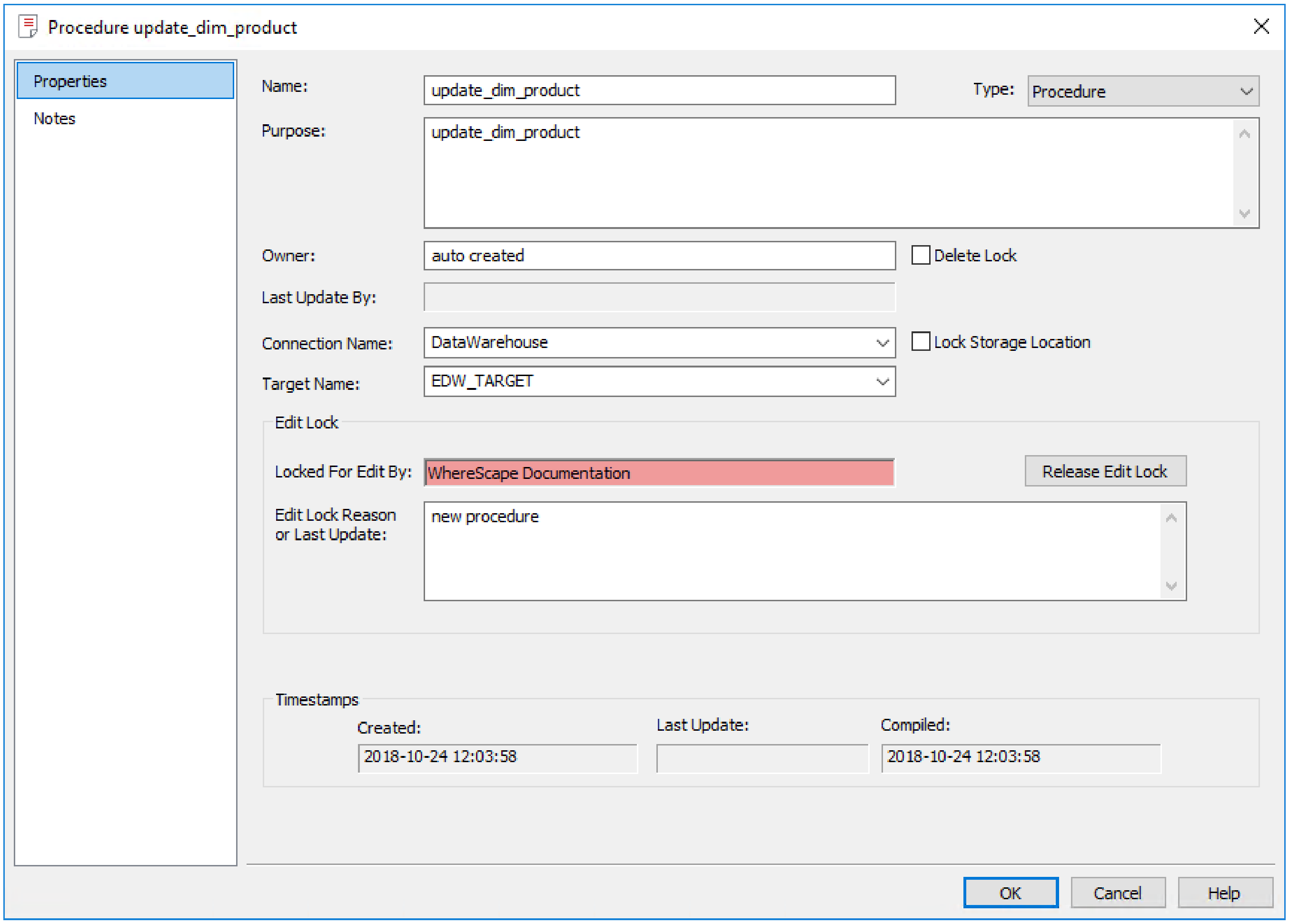
Note

Multiple SQL statements can be separated using the "end of statement" indicator. This is <EOS> by default but can be configured in**Home > Options > Code Generation > General**.

Procedure Properties

The **Properties** screen for procedures and scripts is the same. A procedure can be renamed by changing the name field.

If a procedure is renamed, then it is also necessary to change the procedure name within the actual code. The **Purpose** and **Owner** fields are purely informational.



Name: update_dim_product
Type: Procedure
Purpose: update_dim_product
Owner: auto created
□ Delete Lock
Last Update By:
Connection Name: DataWarehouse
□ Lock Storage Location
Target Name: EDW_TARGET
Locked For Edit By: WhereScape Documentation
Edit Lock Reason or Last Update: new procedure
Created: 2018-10-24 12:03:58
Last Update:
Compiled: 2018-10-24 12:03:58

In the above example, the **Delete Lock** option is not selected. Selecting this check box prevents the procedure from being deleted through the **Delete** menu option. It also prevents the procedure from being overwritten, if a new procedure generation is requested.

Selecting the **Lock Storage Location** option enables you to retain the current location of the procedure, e.g. no changes will occur if the Target location of the procedure is manually changed, and a **Rebuild** or **Regenerate** operation is performed via the Table's **Properties** window or from the Table's right click **Code** context menu.

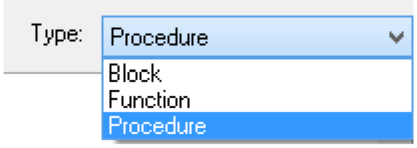
In the above example, the procedure is currently being edited, and is shown as **Locked for Edit** by "WhereScape Documentation". If procedures or scripts have already been opened for editing, they can only subsequently be opened for viewing.

Check out and delete operations for procedures and scripts, as well as the regeneration and drop of procedures are not permitted; if the object is currently Locked For Edit by another user.

The **Release Edit Lock** button to the right of the edit lock message, enables the edit lock to be cleared. If WhereScape RED, the database or the PC crashes when a procedure is open, then the check out must be cleared through this screen.

The **Edit Lock Reason** is for information only, and can be used as another comment field if desired.

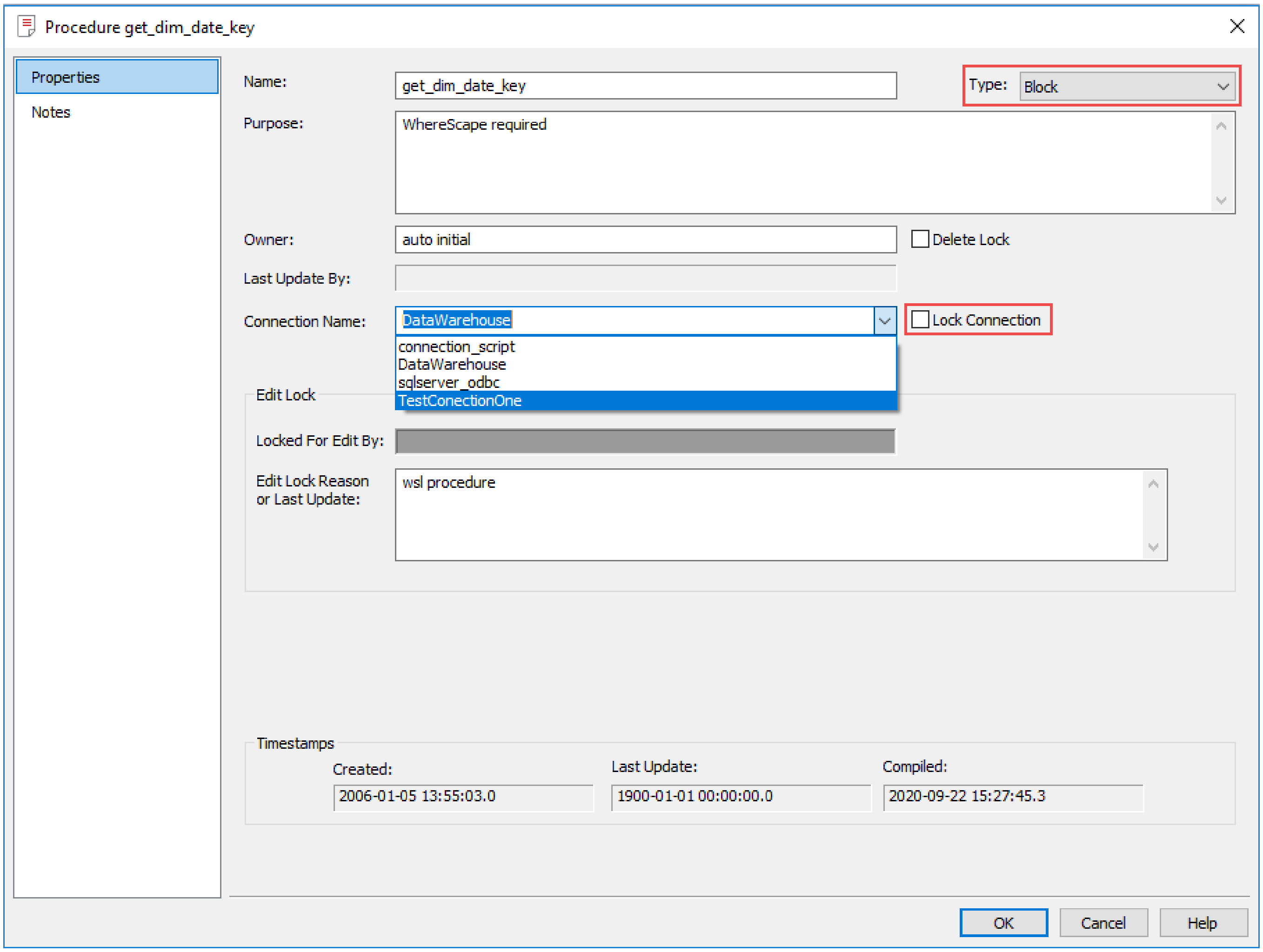
The **Type** drop-down list provides the selection of **Block**, **Function** or **Procedure**:



Selecting a type of **Block** enables you to execute a SQL block against another connection.

The **Connection Name** drop-down field enables you to select the connection against which the SQL block is executed.

The **Lock Connection** check-box displays next to the Connection Name option, enabling locking the selected connection.



Name: get_dim_date_key

Type: Block

Purpose: WhereScape required

Owner: auto initial

□ Delete Lock

Last Update By:

Connection Name: DataWarehouse

□ Lock Connection

Locked For Edit By:

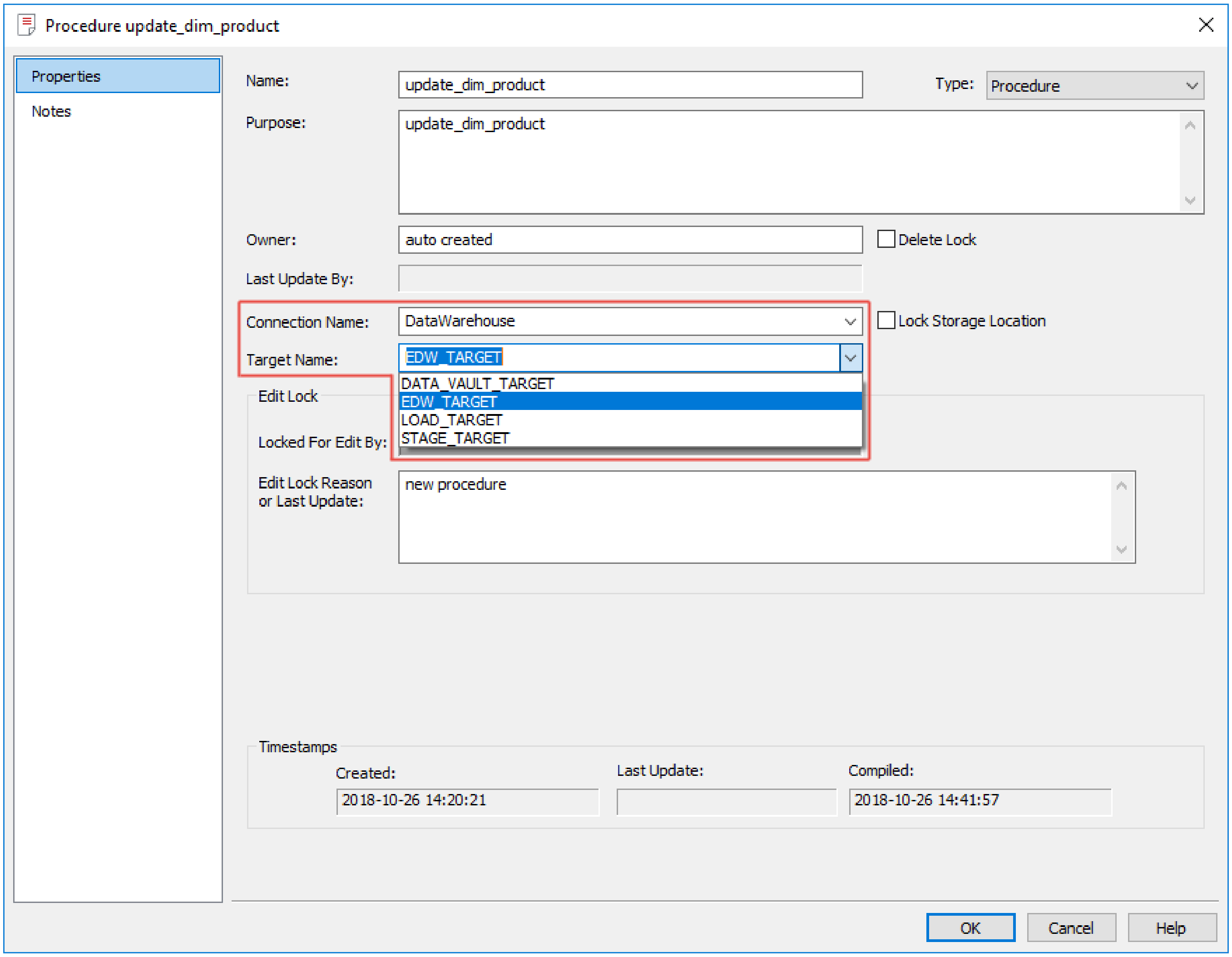
Edit Lock Reason or Last Update: wsl procedure

Created: 2006-01-05 13:55:03.0

Last Update: 1900-01-01 00:00:00.0

Compiled: 2020-09-22 15:27:45.3

For **Procedure** or **Function** type, the **Connection Name** drop-down field enables you to select the connection against which a procedure or function is executed and the target database where the update procedure or function is stored, via the **Target Name** drop-down field.


Procedure update_dim_product

Name: update_dim_product
Type: Procedure
Purpose: update_dim_product
Owner: auto created
□ Delete Lock
Last Update By:
Connection Name: DataWarehouse
□ Lock Storage Location
Target Name: EDW_TARGET

Edit Lock
Locked For Edit By:
Edit Lock Reason or Last Update: new procedure

Timestamps
Created: 2018-10-26 14:20:21
Last Update:
Compiled: 2018-10-26 14:41:57

Notes

- Selecting the **Lock Storage Location** option in the procedure **Properties** window, enables you to retain the current location of the procedure, if the Target location of the procedure is manually changed and a **Rebuild** or **Regenerate** operation is performed, via the Table object's **Properties** window or from the Table object's right click **Code** context menu

Script Generation

Host scripts are generated either manually or via templates. Most Target Enablement Packs provide a rich set of templates for data warehouse operations that automate the generation of either PowerShell or Python scripts for all object types and supported sources. Scripts can also be created manually provided the rules for their inclusion into the scheduling process are followed.

Tip

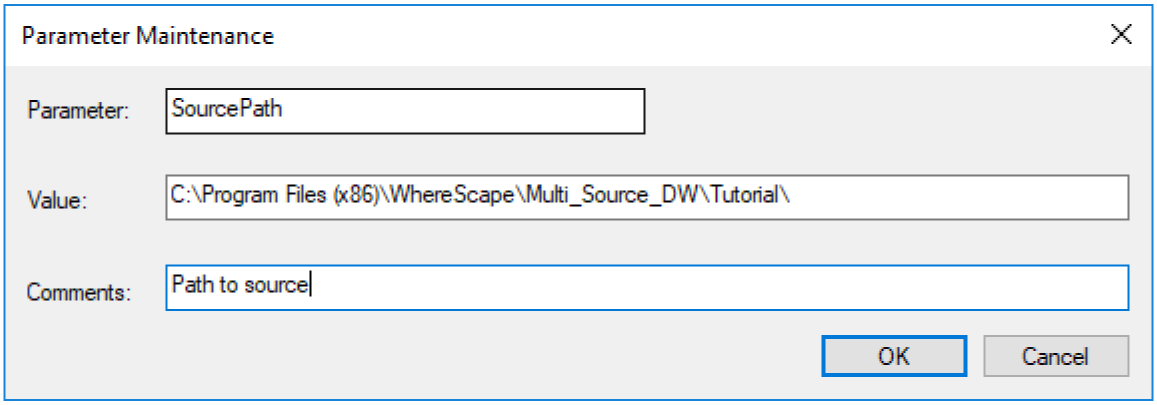
WhereScape RED TIP: Parameters

Parameters can also be added to Scripts to facilitate deployment processes or environment changes without the

Tip

need to regenerate scripts.
For example: add $P<ParameterName>$ to the script where P is the parameter indicator as shown below.

Before adding the Parameter to the script, create the desired parameter in **Tools > Parameters > Add Parameter Source** (context menu).


Parameter Maintenance

Parameter: SourcePath

Value: C:\Program Files (x86)\WhereScape\Multi_Source_DW\Tutorial\

Comments: Path to source

```
SET LOAD_FILE="$PSourcePath$\forecast.txt"
SET TRIG_FILE="$PSourcePath$\forecast.trg"
```

The script makes use of several environmental variables. These variables are acquired from both the Table and Connection properties. These variables are established in the environment by either WhereScape RED or the scheduler. If the script is to be executed outside of WhereScape RED or scheduler control, these variables must be assigned.

Script Editing

WhereScape RED includes a script editor which allows the maintenance of any host scripts within the data warehouse. The editor is invoked by double-clicking on a script name in the left pane or by right-clicking the script name and then selecting **Edit the Script** from the context menu.

Similar to the procedure editor, the script editor includes the following features:

- bash/DOS/PowerShell syntax highlighting
- improved find feature (repeated find, up and down, etc.)
- toggle the display of line numbering
- a status bar that includes line number, line length, selected text length, etc.
- a context sensitive toolbar
- tab to space conversion and vice versa
- uppercase, lowercase, titlecase text conversion command
- increase/decrease test indent command
- bookmark support

Indenting code

The tab character inserts four spaces into the text. A shift/tab removes four spaces.

Cut, Copy, Paste and Delete

The normal Windows cut, copy, paste and delete functions are available either through the toolbar or via the right mouse click context menu.

Indenting a block of text

A selected block of text can be indented by four spaces by depressing the tab character. Each tab will indent by a further four spaces. A shift/tab will remove four spaces from the front of each line in the selected block.

Viewing other scripts

During the editing process it is possible to pop up a window containing other scripts. This window allows cut and paste operations. In such a way it can be used as a work area or as a source of code. Select the **Tools > View Script or Template** menu option to bring up the viewer window.

A number of drop-down list boxes can be chosen from. Once an item is selected the viewer loads the code and moves to the right side of the edit window. The various options are:

- Version: A previously saved version of this script. The description and time are shown.
- Script: Some other script stored within the WhereScape RED metadata.
- Template: A template script as defined in the WhereScape RED metadata.

Once an item is chosen, the viewer appears on the right side of the edit window. The viewer or the main edit window can receive the focus by clicking the mouse within the appropriate window. Code can be cut from one window and pasted into the other. Any changes made in the viewer window cannot be saved.

Notes

Editing, deleting or compiling Locked for Edit or opened procedures/scripts

- Procedures or scripts cannot be deleted if they are Locked for Edit by any user, checked out by another user or if there is another object that has the same associated procedures or scripts.
- Saving or Compiling in the procedure or script edit window cannot be performed if the procedures or scripts become Locked for Edit by other users after the edit window has been opened.
- Procedures or scripts cannot be deleted or modified after the editor window has been opened, unless the Edit Lock has been released. Edit Locks can be released by any user in the Script or Procedure Properties screen.
- To prevent updates, deletes and modifications to certain procedures or scripts, it is best to use the Check Out functionality instead. For more information about this functionality, refer to **Object Check Outs and Check Ins** for details.

Script Testing

When a host script is scheduled, it is run in the scheduler environment. Therefore a UNIX scheduler must be available to run UNIX scripts and Windows scheduler available to run Windows scripts.

It is possible to test a script interactively. For a Windows script, this is achieved by running the script on the current Windows environment. For a UNIX script, RED will attempt to start a telnet window to the UNIX host (as defined in the Connection set for the script) and run the script within this telnet window. Since this method will transfer the necessary scripts files to UNIX via the telnet connection it can take a long time to execute depending on your environment, so it is recommended that UNIX script testing is done via the scehduler for best results.

A script is invoked via the **Execute > Execute the Script** menu option. You can also right click the script from the **Host Script** objects list on the left pane and select **Execute Script** from the context menu.

The output from the script is shown in the results pane.

Note

A connection must be correctly set-up to support UNIX script testing from Windows. Refer to the section on UNIX connections in the RED Installation Guide for details.

Script Output Protocol

The RED UI and the Scheduler have different conventions when it comes to script output and Exit Codes. Scripts generated via Enablement Pack templates already handle this difference in output protocol and switch between the two depending on the context of the run determined by the provided environment variables. When running scripts interactively you also generally do not want to write to the audit and detail log tables but instead only display the output. It is best practice in scripts to provide central logging functions and also a script exit function and in these functions handle the different output protocols. There are complete examples of these functions at the end of this section.

Determining the run context

To test at run-time whether the script run context is interactive the following method is used:

When a script is run via RED UI the environment variables for WSL_JOB_KEY and WSL_JOB_NAME will be set as ‘0’ and ‘Develop’ respectively. When run through the Scheduler these environment variables will match the actual Job ID and Job Name.

To test for interactive execution the following sample functions can be used:

Python

```
def is_red_interactive():
    if os.environ.get('WSL_JOB_KEY','') == '0' and os.environ.get('WSL_JOB_NAME','') == 'Develop':
        return True
    else:
        return False
```

Powershell

```
function Test-IsInteractive {
  if (($ENV:WSL_JOB_KEY -eq '0') -and ($ENV:WSL_JOB_NAME -eq 'Develop')) {
    return $true
  }
  else {
    return $false
  }
}
```

RED UI Script Output Protocol

For interactive scripts the Exit Code must be 0 always.

The interactive script result is determined by reading the status value from the first line of the standard-out, therefore all standard-out must be suppressed/collected until the final status value and status result messages are written. The second line of standard-out is taken as the status result message. Any further lines of standard-out are considered audit log messages. Any standard-error messages are considered detail/error logs and are printed after the audit logs in the results pane.

<table>
<tr><th>Output Channel</th><th>Purpose</th></tr>
<tr><td><b>Exit Code</b></td><td>Must be ‘0’, non-zero will throw an error dialog and suppress the logging.</td></tr>
<tr><td><b>Standard Output</b></td><td>Line 1 - Numeric result code (1,-1,-2,-3)
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1</td><td>Success</td></tr>
<tr><td>-1</td><td>Success with Warnings</td></tr>
<tr><td>-2</td><td>Failure</td></tr>
<tr><td>-3</td><td>Fatal Unexpected Error</td></tr>
</table>
Line 2 - Result message
Lines 3+ - Audit messages</td></tr>
</table>

Standard Error	Lines 1+ - Detail messages

Scheduler Script Output Protocol

For Scheduler execution of scripts the Exit Code is what determines the actual run status of any script. An Exit Code of ‘0’ (the default when nothing has gone wrong) signals a successful script execution.

Audit, Error, and Detail logging can be output at any time provided each line of output conforms to a certain JSON output protocol defined below. Any output not conforming to the JSON structure will be treated as error/detail logs.

The final result message must also conform to the JSON output protocol.

<table>
<tr><th>Output Channel</th><th>Purpose</th><th>Protocol</th></tr>
<tr><td><b>Exit Code</b></td><td>Result Code</td><td>0 - Success 1 (or non-zero) - Error</td></tr>
<tr><td><b>Standard Output</b></td><td>Structured messages</td><td>UTF-8
Sequence of JSON objects (not a JSON array or objects, just many objects next to each other)
Each object represents one message, either audit, detail or result depending on the contents.
<table>
<tr><th>Field</th><th>Mandatory</th><th>Type</th><th>Meaning</th></tr>
<tr><td><b>type</b></td><td>Y</td><td>string</td><td>Message type, one of "result", "audit", "detail"</td></tr>
<tr><td><b>message</b></td><td>Y</td><td>string (1024)</td><td>Message text</td></tr>
<tr><td><b>statusCode</b></td><td>N</td><td>string (1)</td><td>"I" (information), "S" (success), "W" (warning), "E" (error), (defaults to "I")</td></tr>
</table>
</td></tr>
<tr><td><b>Standard Error and Output</b></td><td>Unstructured detail messages</td><td>UTF-8 Each line is equivalent to { "type": "detail", "message": "LINE" }</td></tr>
</table>

Example Scheduler Script Output

Standard Output	Standard Error

`{"type":"audit","message":"Starting Load of load_budget"}`
`{"type":"audit","statusCode":"W",message:"Table load_budget is not empty"`
`{"type":"detail","message":"Loading from file \"budget.txt\""}`
`{"type":"audit","message":"Load completed"}`
`{"type":"result","message":"Loaded 11 rows, woohoo!"}`
	Some message from psql.exe
	Some message from other tools

Example Python Logging Functions

```
def is_red_interactive():
    if os.environ.get('WSL_JOB_KEY','') == '0' and os.environ.get('WSL_JOB_NAME','') == 'Develop':
        return True
    else:
        return False

def write_audit(message = '', logType = 'audit', statusCode = 'I'):
    # statusCodes 'E' = error, 'W' Warning, 'I' information, 'S' Success
    global interactiveLog
    if is_red_interactive():
        interactiveLog = '\n'.join([interactiveLog, message])
    else:
        outputJson = json.dumps({"type": logType, "message": message, "statusCode": statusCode})
        print(outputJson, flush=True)

def write_error(message = ''):
    write_audit(message, 'audit', 'E')

def write_detail(message = '', statusCode = 'I'):
    if debugMode:
        write_audit(message, 'detail', statusCode)

def write_result(message = '', statusCode = 'S'):
    write_audit(message, 'result', statusCode)

def exit_script(exitCode = 0, message = 'Executed the script'):
    if is_red_interactive():
        if exitCode != 0:
            print(-2, flush=True)
        else:
            print(1, flush=True)
        print(message, flush=True)
        print(interactiveLog, flush=True)
        sys.exit(0)
    else:
        if exitCode != 0:
            write_result(message,'E')
        else:
            write_result(message,'S')
        sys.exit(exitCode)
```

Example PowerShell Logging Functions

```
function WriteAudit($message, $type="audit", $statusCode="I") {
    $outputJson = ConvertTo-Json @{message = $message; type = $type; statusCode = $statusCode} -
Compress
        if(Test-IsInteractive){
      $logStream.WriteLine($outputJson)
    }
    else {
      [Console]::WriteLine($outputJson)
    }
}

function Test-IsInteractive {
  # determines if the script is being run from RED or a Scheduler
  if ( ${env:WSL_JOB_KEY} -eq '0' -and ${env:WSL_JOB_NAME} -eq 'Develop' ) {
    $true
  }
  else {
    $false
  }
}
```

```
function Exit-Script([int]$scrResCode=1, $scrResMsg="Success") {
    if(Test-IsInteractive){
      $scriptExitCode = 0
            [Console]::WriteLine($scrResCode)
      [Console]::WriteLine($scrResMsg)
    }
    elseif($scrResCode -eq 1) {
      $scriptExitCode = 0
      WriteAudit $scrResMsg
    }
    else{
      $scriptExitCode = $scrResCode
            if ($scrResCode -eq -1) {
        WriteAudit $scrResMsg "audit" "W"
      }
      else {
        WriteAudit $scrResMsg "audit" "E"
      }
    }
        Print-Log
    Exit $scriptExitCode
}

function Print-Log { {%- br %}
    $logStream.Dispose(){%- br %}
    $logReader = New-Object IO.StreamReader($fileAud){%- br %}
    {%- br %}
    while( ! $logReader.EndOfStream) { {%- br %}
        [Console]::WriteLine($logReader.ReadLine()){%- br %}
    }{%- br %}
    {%- br %}
    $logReader.Dispose(){%- br %}
}
```

Script Environment Variables

The following environment variables are available for all script loads and script exports, both Windows and UNIX/Linux.

All load scripts

The following variables are available in all load scripts:

Environment variable	Description
WSL_LOAD_FULLNAME	The fully-qualified load table name.
WSL_LOAD_TABLE	The unqualified load table name.
WSL_LOAD_SCHEMA	The schema for the load table.
WSL_LOAD_DB	The name of the database for the load table.
WSL_TEMP_DB	• **PDW:** The name of the staging database for the load. **Others:** Not Used.
WSL_TGT_DSN	The ODBC data source name (DSN) for the load table's storage connection.

Environment variable	Description
WSL_TGT_SERVER	The server for the load table's storage connection.
WSL_TGT_DBPORT	The database port for the load table's storage connection.
WSL_TGT_DBID	The **Database ID** property of the load table's storage connection.
WSL_TGT_USER	The user id for the load table's storage connection.
WSL_TGT_PWD	The password for the load table's storage connection.

All load scripts from Database or ODBC connections

In **addition** to the variables in the previous table, the following variables are available in all load scripts from Database or ODBC connections:

<table>
<tr><th>Environment variable</th><th>Description</th></tr>
<tr><td>WSL_SRC_DSN</td><td>The ODBC data source name (DSN) for the source connection.</td></tr>
<tr><td>WSL_SRC_SERVER</td><td>The server for the source connection.</td></tr>
<tr><td>WSL_SRC_DBPORT</td><td>The database port for the source connection.</td></tr>
<tr><td>WSL_SRC_DBID</td><td>The <b>Database ID</b> property of the source connection.</td></tr>
<tr><td>WSL_SRC_DB</td><td>The name of the database for the source connection.</td></tr>
<tr><td>WSL_SRC_SCHEMA</td><td>The Source Schema property of the load.
<b>Note</b>
The property is fetched without modification, so there may or may not be a trailing dot depending on how it is configured.
However, it is better not to assume the trailing dot is or isn't appended by using the variable like this, when it is not empty:
<table>
<tr><th>OS</th><th>If no trailing dot is wanted</th><th>If a trailing dot is wanted</th></tr>
<tr><td><b>Windows</b></td><td>!WSL_SRC_SCHEMA:.=!</td><td>!WSL_SRC_SCHEMA:.=!.</td></tr>
<tr><td><b>UNIX/ Linux</b></td><td>${SRC_SCHEMA%.}</td><td>${SRC_SCHEMA%.}.</td></tr>
</table>
</td></tr>
<tr><td>WSL_SRC_USER</td><td>The user id for the source connection.</td></tr>
<tr><td>WSL_SRC_PWD</td><td>The password for the source connection.</td></tr>
</table>

All export scripts

The following variables are available in all export scripts:

Environment variable	Description
WSL_EXP_NAME	The export object name.
WSL_EXP_FULLNAME	The fully-qualified export table name.
WSL_EXP_TABLE	The unqualified export table name.
WSL_EXP_SCHEMA	The schema for the export table.

Environment variable	Description
WSL_EXP_DB	The name of the database for the export table.
WSL_TEMP_DB	- **Others:** Not used.
WSL_SRC_DSN	The ODBC data source name (DSN) for the export table's storage connection.
WSL_SRC_SERVER	The server for the export table's storage connection.
WSL_SRC_DBPORT	The database port for the export table's storage connection.
WSL_SRC_DBID	- The **Database ID** property of the export table's storage connection.
WSL_SRC_USER	The user id for the export table's storage connection.
WSL_SRC_PWD	The password for the export table's storage connection.

All scripts

In **addition** to the specific variables in the previous tables, the following variables are available in all scripts:

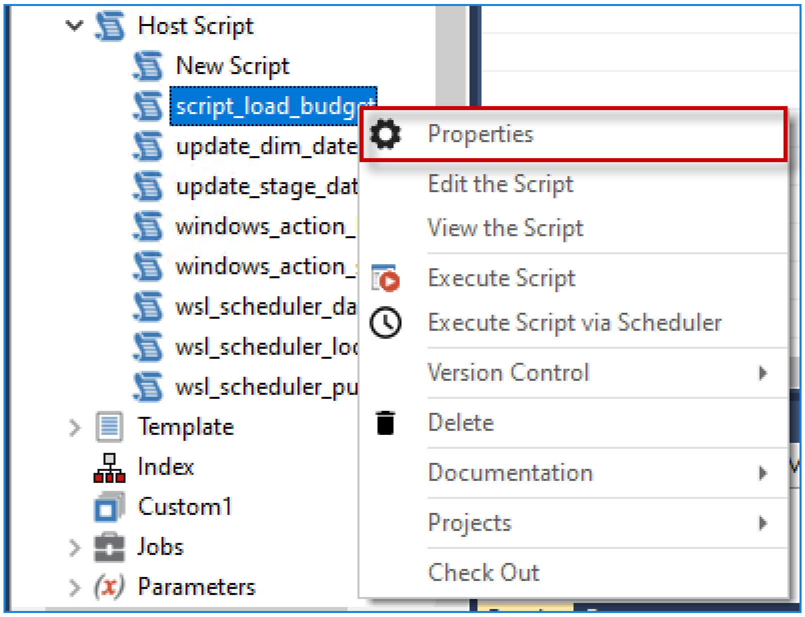
Environment variable	Description
WSL_META_DSN	The ODBC data source name (DSN) for the meta-repository connection.
WSL_META_SERVER	The server for the meta-repository connection.
WSL_META_DBID	The **Database ID** property of the meta-repository connection.
WSL_META_DB	The name of the database for the meta-repository connection.
WSL_META_SCHEMA	The meta-repository schema name, with a trailing dot.
WSL_META_USER	The user id for the meta-repository connection.
WSL_META_PWD	The password for the meta-repository connection.
WSL_WORKDIR	- **Windows:** The work directory defined in the Windows connection. - **UNIX/Linux:** The work directory defined in the UNIX/Linux or Hadoop connection.
WSL_SEQUENCE	A unique sequence number for the load or export task.
WSL_PARAMnnn	Any parameters that start with the load table or export object name. **Example:** A table called **load_abc** has a parameter called **load_abc_server** defined. In this case, a variable called **WSL_PARAM_SERVER** (Windows) or **PARAM_SERVER** (UNIX/Linux) will be created.

Scheduling Scripts

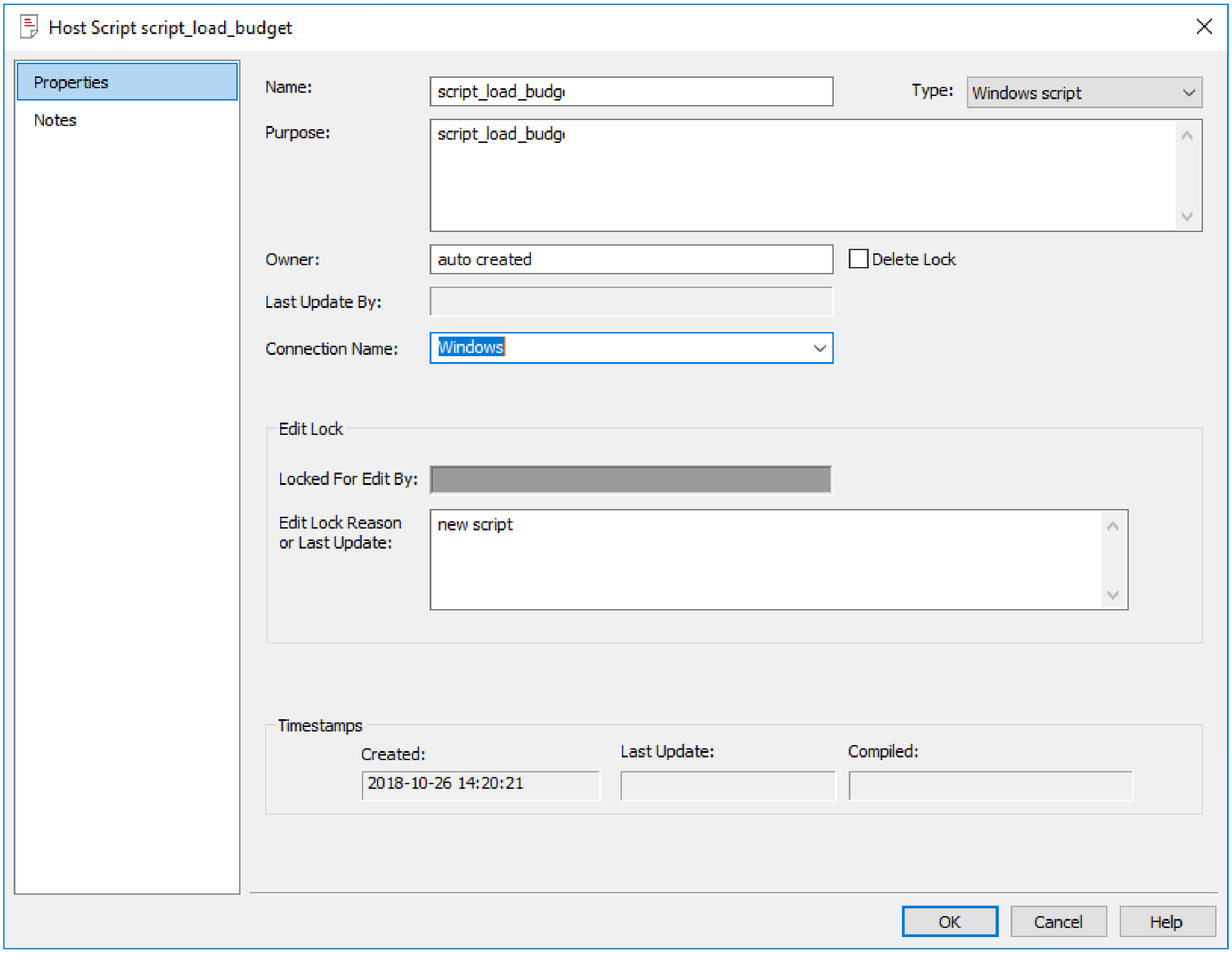
When a host script is scheduled, it is run in the scheduler environment. Therefore, a UNIX/Linux scheduler must be available to run UNIX/Linux scripts and a Windows scheduler available to run Windows script.

It is important to set the **connection** on the **Properties** screen for the script.

Right-click on the host script in the left pane and select **Properties**.



Set the **Connection Name** to either **Windows** or **Unix** and then click **OK**.


Host Script script_load_budget

Name:	script_load_budg
Type:	Windows script
Purpose:	script_load_budg
Owner:	auto created
□ Delete Lock	
Last Update By:	
Connection Name:	Windows

Edit Lock

Locked For Edit By:	
Edit Lock Reason or Last Update:	new script

Timestamps

Created:	Last Update:	Compiled:
2018-10-26 14:20:21		

Note

If you fail to set the default connection for the host script, you will receive a return message of **Invalid Host Type** when the host script is executed.

There are a number of conventions that must be followed if a host script is to be used by the WhereScape RED scheduler. Refer to the section **Script Output Protocol** for details.

Manually Created Scripts

Individual scripts can also be manually created in RED to perform and schedule tasks that are not related to load tables.

Note, it is important to follow the RED **Script Output Protocols** when developing scripts of your own.

Action Processing Scripts

RED Action Processing Scripts are scripts that perform a set of scheduler actions for a given object in RED. Action Processing Scripts are required for each RED object involved in a scheduled job task except stand-alone scripts. An Action Processing Script is generated by a template (or manually) for a particular object and contains the code required to run each available action on the external scheduler for that object type.

When the Scheduler runs a particular task on an object it passes the task action code as an argument to the Action Processing Script so that the script runs the corresponding action routine in the script.

Action Processing Scripts for a table object would typically contain code to perform:

- Processing Actions
  - Pre Actions - Index drop, Pre-SQL, Truncate etc
  - Primary Routine - Load/Update Script and Procedure execution
  - Secondary Routine - Post-Load/Custom Script and Procedure execution
  - Post Actions - After-Load Transformations, Index rebuild
- Object DDL
  - Create
  - Drop

Working with Action Processing Scripts

Enablement Pack Templates

Action Processing Scripts will normally be auto-generated using templates but can also be created by hand. Your **WhereScape Target Enablement Pack** will have suitable templates included for Action Processing. Typically these templates are named with the following naming convention:

wsl_<target_platform>_<pyscript|pscript>_action

- **<target_platform>** - the Enablement Pack target platform
- **<pyscript|pscript>** - either a python or powershell based template

While RED provides the ability to define separate templates for Linux processing vs Windows processing, WhereScape Target Enablement Packs will in most cases only supply a single Action Processing Script template that is suitable for both Windows and Linux/Unix job processing environments.

Object types supported

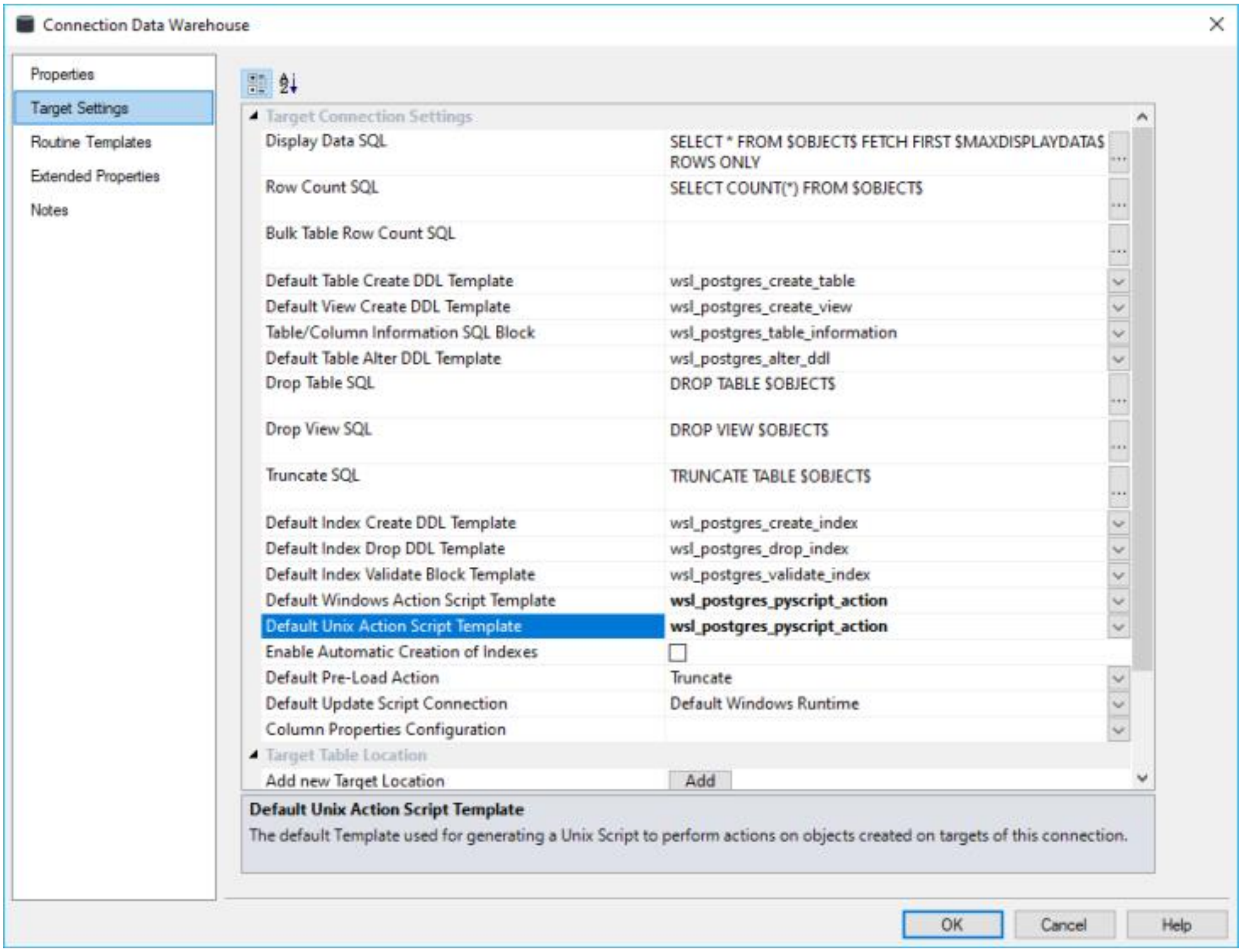
The below RED object types can have Action Processing Scripts assigned:

- Table - all except Retro tables
- View
- Export
- Source Mapping
- Index - only required if you perform tasks directly on indexes
- Procedure/SQL Block - only required for stand-alone Procedures or SQL Blocks

Settings

Connection - Target Settings

Default Action Processing Script Templates are set in the Target Settings tab of your Data Warehouse Connections (RED Connections with Target Table Location).



Target Connection Settings	
Display Data SQL	SELECT * FROM $OBJECT$ FETCH FIRST $MAXDISPLAYDATA$ ROWS ONLY
Row Count SQL	SELECT COUNT(*) FROM $OBJECT$
Bulk Table Row Count SQL	
Default Table Create DDL Template	wsl_postgres_create_table
Default View Create DDL Template	wsl_postgres_create_view
Table/Column Information SQL Block	wsl_postgres_table_information
Default Table Alter DDL Template	wsl_postgres_alter_ddl
Drop Table SQL	DROP TABLE $OBJECT$
Drop View SQL	DROP VIEW $OBJECT$
Truncate SQL	TRUNCATE TABLE $OBJECT$
Default Index Create DDL Template	wsl_postgres_create_index
Default Index Drop DDL Template	wsl_postgres_drop_index
Default Index Validate Block Template	wsl_postgres_validate_index
Default Windows Action Script Template	**wsl_postgres_pyscript_action**
Default Unix Action Script Template	**wsl_postgres_pyscript_action**
Enable Automatic Creation of Indexes	☐
Default Pre-Load Action	Truncate
Default Update Script Connection	Default Windows Runtime
Column Properties Configuration	
Target Table Location	
Add new Target Location	Add

Default Unix Action Script Template
The default Template used for generating a Unix Script to perform actions on objects created on targets of this connection.

Default Unix Action Processing Script Template:

- When set to a valid template then this template is assigned to new objects created on this target at creation time enabling the auto-generation of the Unix/Linux Action Processing Script to occur.
- Set this when you have Azkaban Executors on Unix/Linux

Default Unix Action Processing Script Template:

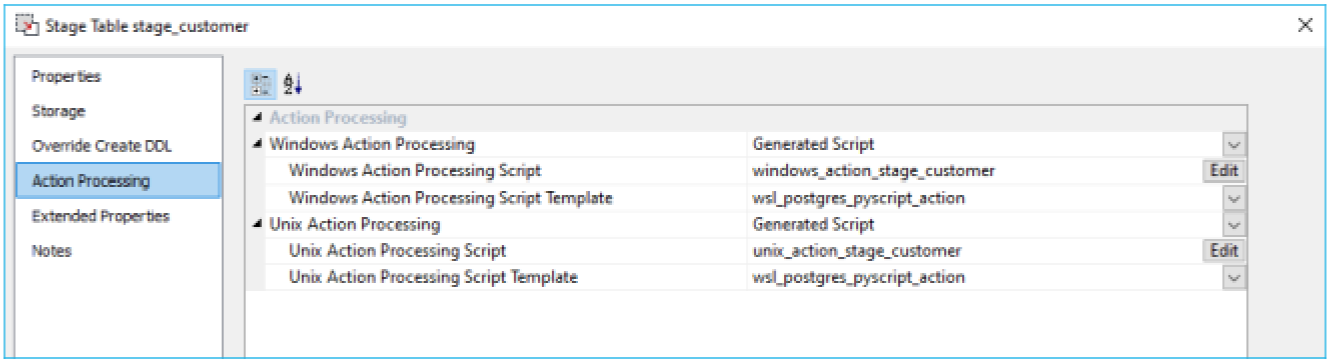
- When set to a valid template then this template is associated to any new objects at creation time enabling the auto-generation of the Unix/Linux Action Processing Script to occur.
- Set this when you have Azkaban Executors on Windows

Properties - Action Processing tab

To adjust or set an Action Processing template on an existing object open the Properties of the object and navigate to the Action Processing tab. From this tab you can choose the Windows and Unix processing options from:

- **Generated Script** - templated code generation, requires a template to be selected
- **Existing Script** - requires an existing script to be selected
- **None** - when no Action Script is required

The **edit** button opens the script in the script editor and closes the properties dialog.

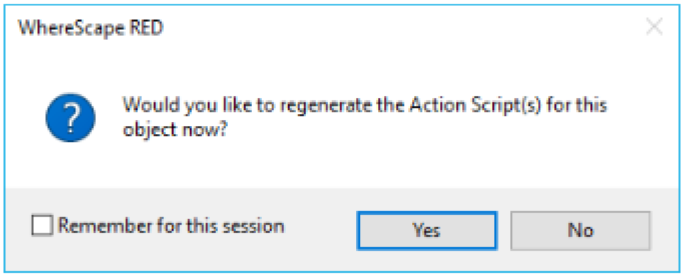


Action Processing	
Windows Action Processing	Generated Script
Windows Action Processing Script	windows_action_stage_customer
Windows Action Processing Script Template	wsl_postgres_pyscript_action
Unix Action Processing	Generated Script
Unix Action Processing Script	unix_action_stage_customer
Unix Action Processing Script Template	wsl_postgres_pyscript_action

Generating Action Processing Scripts

Auto-generation

In most cases, RED will prompt to generate action scripts for newly created objects and for changes to existing objects which may affect the contents of the action script. If both Windows and Unix Action Processing are configured then both scripts will be generated or regenerated from the single prompt.



Would you like to regenerate the Action Script(s) for this object now?

□ Remember for this session

If you prefer to defer generation of action scripts to a bulk regenerate at a later stage or if you know you need to make further changes to the object which will affect the action script also then select ‘No’ at the prompt.

To avoid the prompt recurring in the same circumstances throughout your RED session then select the ‘Remember for this session’ check box before selecting Yes or No.

Note that there is a setting that affects the prompting behavior under User Preferences.

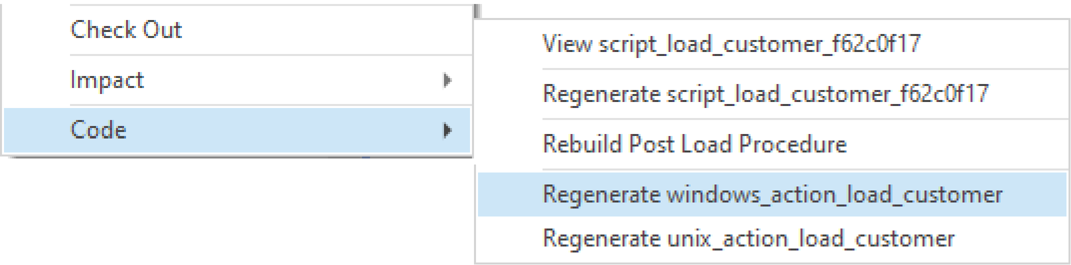
The setting can be found under *User Preferences->Common->Look and Feel->Confirmation Prompts*

Setting: **‘Prompt to Regenerate Action Scripts’ default=checked (on)**

When unchecked this will cause Action Scripts to always be regenerated as required without prompting.

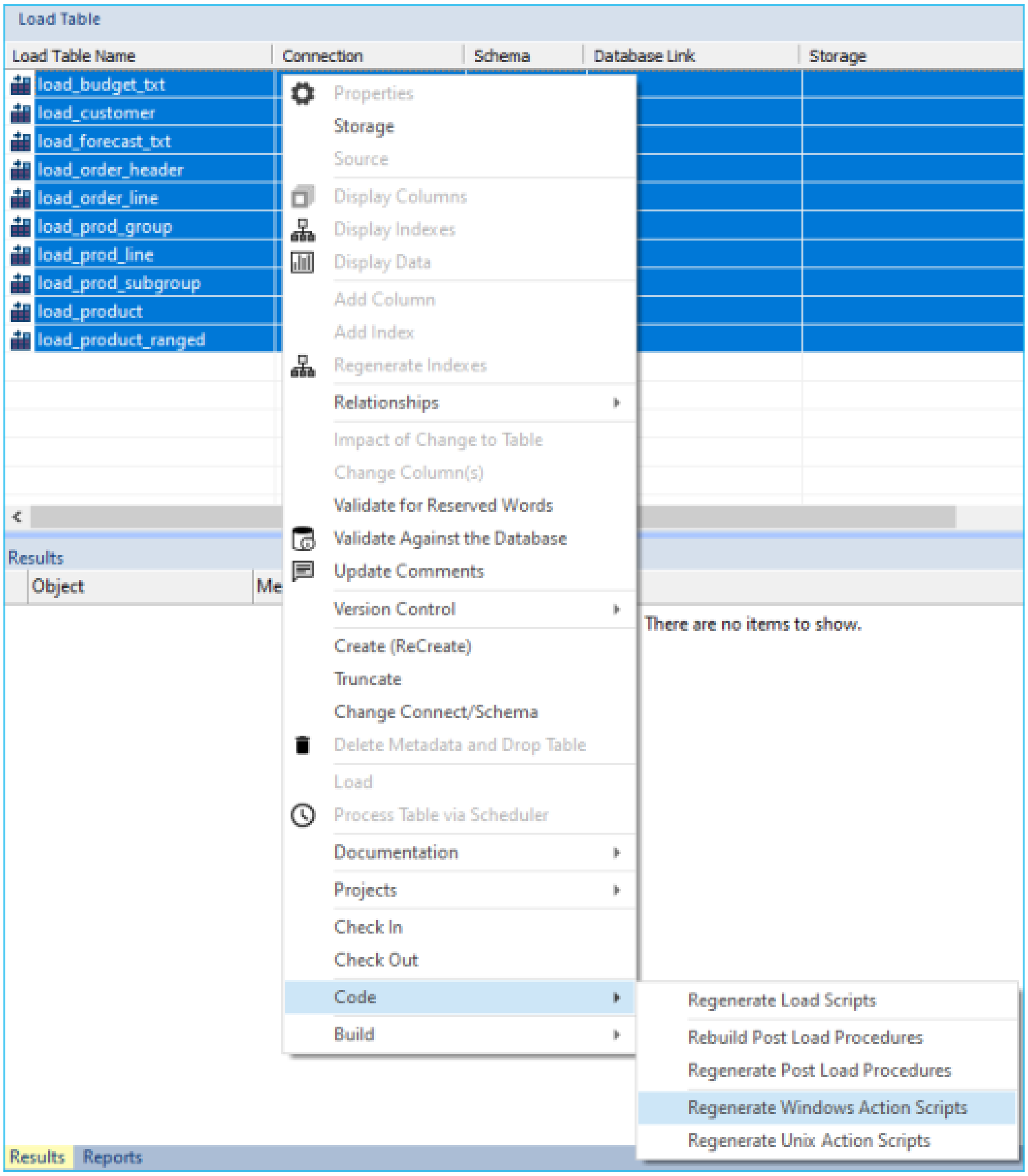
Manual Generation

Each object that supports Action Processing Scripts will have an additional item added to its context menu under the **Code** sub-menu for Windows and Unix Actions Scripts provided there are Action Processing templates set on the object.


Check Out
Impact
Code
View script_load_customer_f62c0f17
Regenerate script_load_customer_f62c0f17
Rebuild Post Load Procedure
Regenerate windows_action_load_customer
Regenerate unix_action_load_customer

Bulk generation

If you would like to generate or regenerate the Windows or Unix Action Processing Scripts on multiple objects at once this can be done by first listing those objects in the middle pane and then multi-selecting the objects you'd like to regenerate. Then the same context menu can be used as in single object regeneration under the Code submenu items.

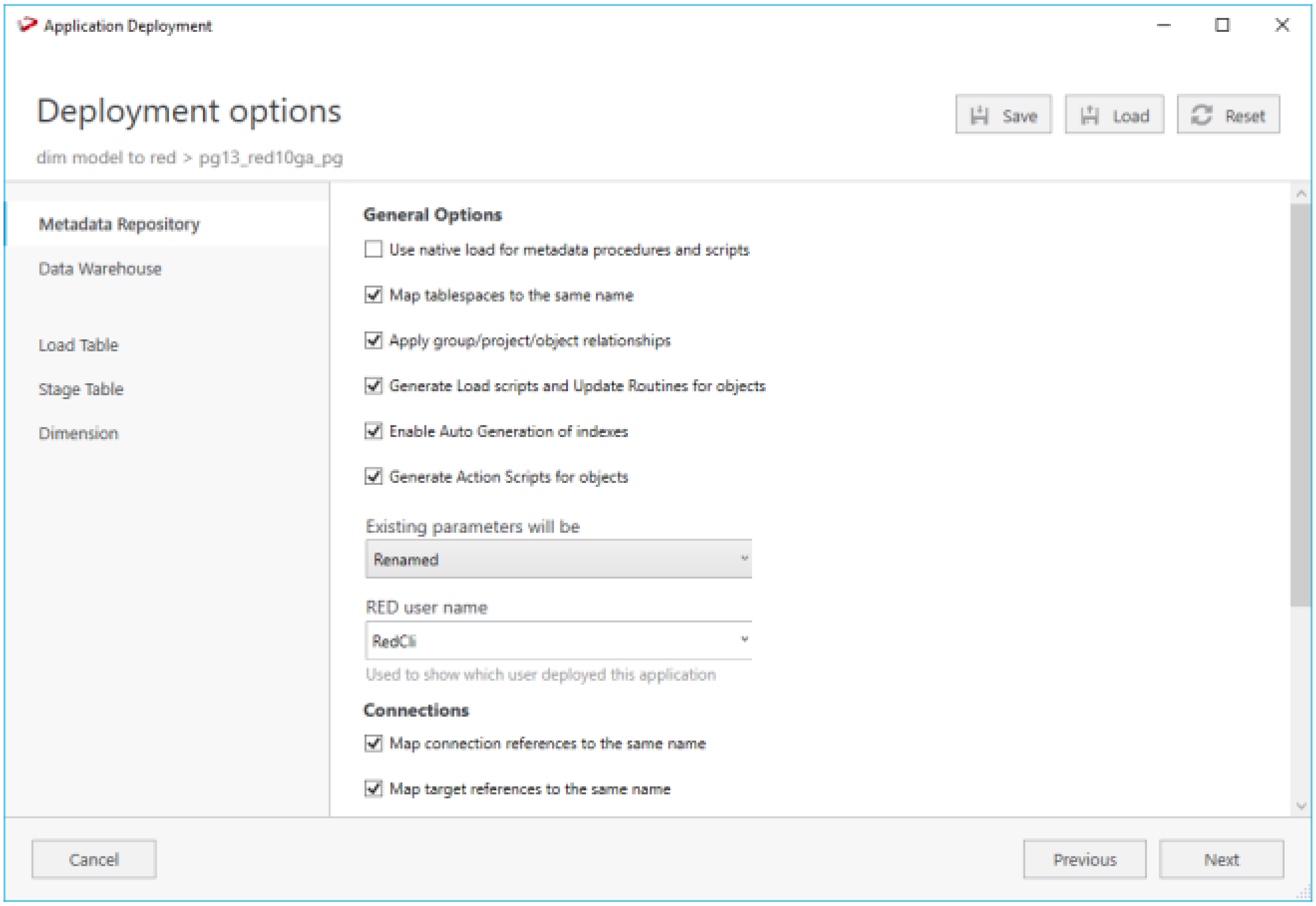


Application Deployment

For command line driven deployments there is an option which can be set in deployment options XML to turn on auto generation of Actions Scripts during deployment:

`<Generate_Action_Scripts>`YES`</Generate_Action_Scripts>`

For interactive deployments using **RedDeploy.exe** this option is set in the UI as shown, using the **Generate Action Scripts for objects** option:



Deployment options

dim model to red > pg13_red10ga_pg

General Options

☐ Use native load for metadata procedures and scripts

☑ Map tablespaces to the same name

☑ Apply group/project/object relationships

☑ Generate Load scripts and Update Routines for objects

☑ Enable Auto Generation of indexes

☑ Generate Action Scripts for objects

Existing parameters will be: Renamed

RED user name: RedCli

Used to show which user deployed this application

Connections

☑ Map connection references to the same name

☑ Map target references to the same name

When this option is set to yes any given object in the deployment application will be assessed for regeneration of its action script.

Regeneration Conditions:

When the setting is 'YES' generate the action script only if the deployment application does not include either a Windows or Unix action script for the object being deployed and when an object is being created or modified by the deployment if either:

- For modifications - If there is an associated action template set on the object, either within the application itself or in the target repository.
- For new objects - if a template is not already associated (in the application itself) then use the default action template(s) in the Target Connection of the object.

RedCli object generate-routine

Action Processing Scripts, along with other template generated scripts, can also be generated from the command line. The following would be the typical format of the command:

```
RedCli.exe object generate-routine --routine-owner-type <[object|source-mapping]> --
routine-owner-name <parent_object> --routine-operation <[windows-action|unix-
action]> --version-and-modify-existing-routine
```

For command line help and other options refer to the inbuilt help:

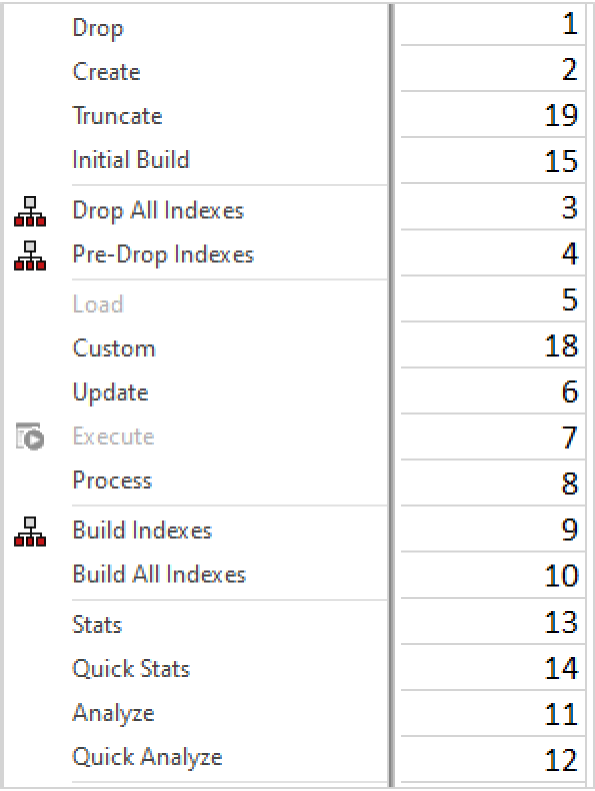
```
RedCli.exe --help
```

Available Actions

The supported codeable Actions directly map to the Object Task Actions context menu in the Task Definition page of Jobs in the Scheduler.

When the Scheduler runs a particular task on an object it passes the task action code as an argument to the Action Processing Script so that the script runs the corresponding action routine in the script.

The image below shows the Action Name to Action Code mappings:



Drop	1
Create	2
Truncate	19
Initial Build	15
Drop All Indexes	3
Pre-Drop Indexes	4
Load	5
Custom	18
Update	6
Execute	7
Process	8
Build Indexes	9
Build All Indexes	10
Stats	13
Quick Stats	14
Analyze	11
Quick Analyze	12

Primary and Secondary Routine script execution

- Primary = Load, Export and Update routines
- Secondary = Post-Load and Custom routines

When an action script is executed by the Scheduler with an Action that leads to the execution of another script such as Load, Process, Custom, and Update then the corresponding script as well as the Action Processing Script are written to the work directory for execution. The main action script will be set up with the special environment variables that contain the script run command as defined in the Host Script Language Definition for the script type. When the Primary or Secondary routine is a Procedure object then the actual SQL statements are written to disk and the file made available to the Action Processing Script.

Action Processing Scrip Environment Variables:

When the Routine Type is script based:

- WSL_SCRIPT_1_COMMAND - The Primary Routine (load, update) run command
- WSL_SCRIPT_2_COMMAND - The Secondary Routine (post-load, custom) run command

When the Routine Type is a Procedure or Function and only for the Action type ‘Create/Compile’:

- WSL_SCRIPT_1_COMMAND - The full path to the file containing the Procedural code

When the Routine Type is an SQL Block

- WSL_SCRIPT_1_COMMAND - The full path to the file containing the Primary SQL Block code
- WSL_SCRIPT_2_COMMAND - The full path to the file containing the Secondary SQL Block code

Audit and Error Logging

Refer to WhereScape Azkaban Job and Task Logging and the **Script Output Protocol** sections

Templates

Templates provide the ability to customize automatically generated code within RED. This feature is most suited to users that would like to customize automatically generated code or would like to expand RED to support non-native database platforms.

Creating templates is an advanced function that requires intimate knowledge of RED operations and metadata structure. WhereScape recommends that you contact our consulting team to assist with this feature. However, should you wish to use this feature independently, example templates and up-to-date reference information is available on our website:

https://www1.wherescape.com/support/software-downloads-documentation/enablement-packs/

Each template is assigned a type and a target database, these properties are used to assist with filtering when associating table operations to templates. RED supports templates for the following operations:

Operation	Database	Template Type
Alter Table DDL	Custom (on SQL Server)	DDL
Create Table DDL	All database types	DDL
Create Index DDL	Custom (on SQL Server)	DDL
Drop Index DDL	Custom (on SQL Server)	DDL
Export Script	All database types	Windows Script
		PowerShell Script
		Unix Script
Load Script	All database types	Windows Script
		PowerShell Script
		Unix Script
Update Procedure	Custom (on SQL Server)	Block
		PowerShell Script
	Hive	Block
	SQL Server and PDW	Block
		Procedure
Custom Update Procedure	SQL Server	Procedure
Update Script	Custom (on SQL Server)	PowerShell Script
Post Load Procedure	Custom (on SQL Server)	Block
	Hive	Block
	SQL Server and PDW	Block
	SQL Server	Procedure
Post Load Script	Custom (on SQL Server)	PowerShell Script

Notes:

- Refer to **Rebuilding Update Scripts** for details on generating an update procedure via a template.

Utility type templates contain common code for use by other templates.

Templates are written in the Pebble template language—refer to **Pebble Templates** and **Pebble Syntax** sections for details.

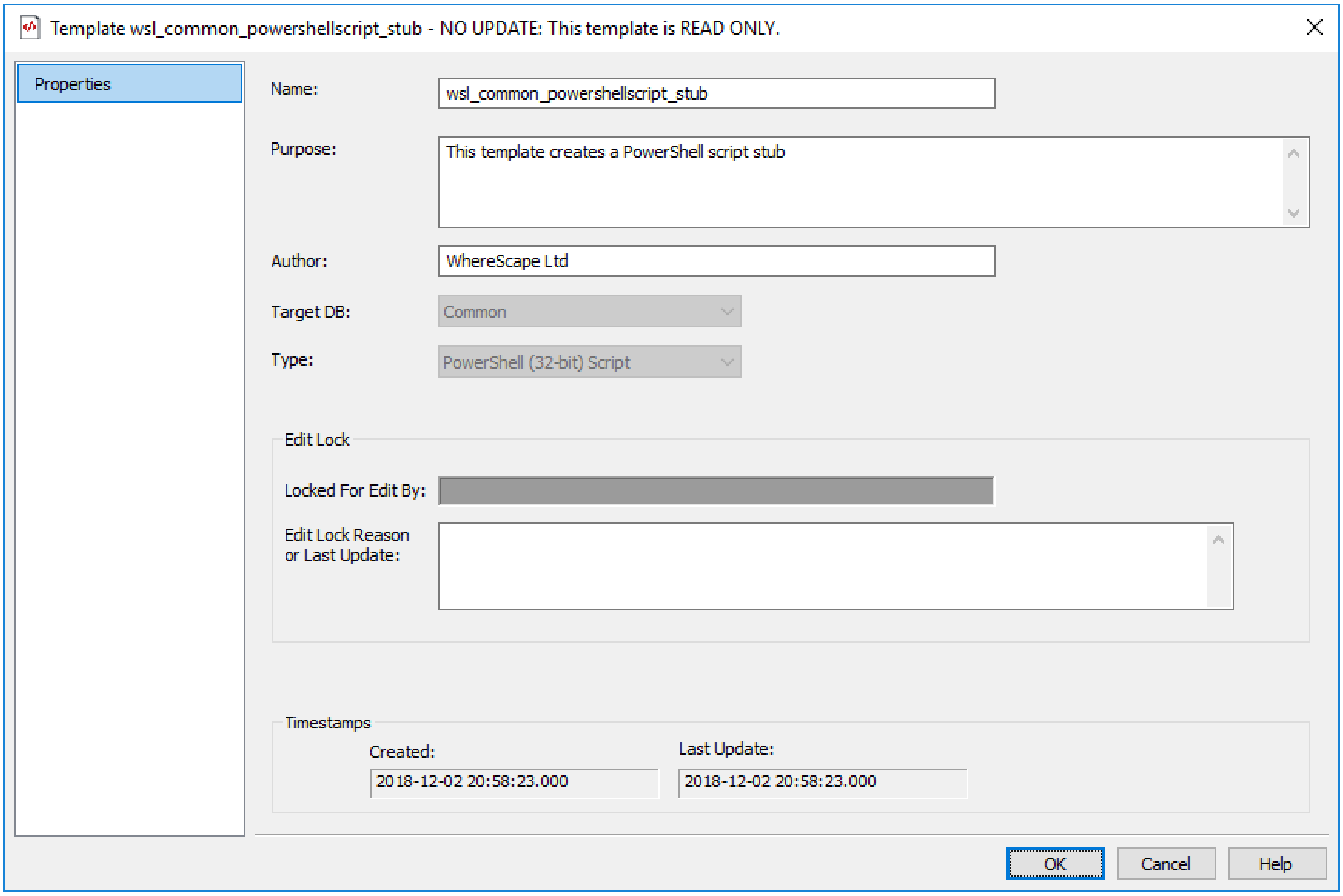
You can also define default templates to use for the update routines of table objects in RED by Connection type—refer to **Connection Routine Templates** for details.

Tip:

Detailed logs can be produced during template evaluation by typing **FULLLOG** in the Notes of the relevant connection.

Template Properties

The properties screen for a template is shown below.



Template wsl_common_powershellscript_stub - NO UPDATE: This template is READ ONLY.

Name: wsl_common_powershellscript_stub

Purpose: This template creates a PowerShell script stub

Author: WhereScape Ltd

Target DB: Common

Type: PowerShell (32-bit) Script

Edit Lock

Locked For Edit By:

Edit Lock Reason or Last Update:

Timestamps

Created: 2018-12-02 20:58:23.000

Last Update: 2018-12-02 20:58:23.000

Name, **Purpose** and **Author** fields should be completed to provide background information on the template; these fields are purely informational.

Created and **Last Update** fields provide date information on the template.

The **Target DB** sets the type of database connections for this template. The template is only selectable for an operation if the **Target DB** field matches that of the object. Target DB is restricted, based on your license.

- Common (applies to all databases)
- Licensed DB Type

The **Type** field specifies what the template can be used for in RED. This can be set to one of the following:

- Alter
- Block
- DDL
- Function

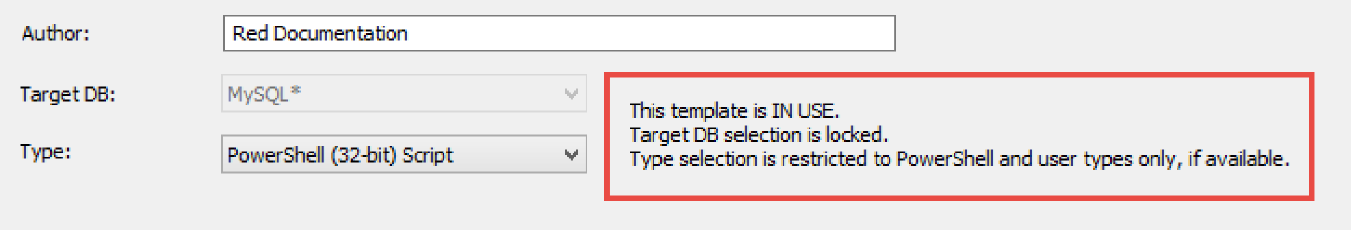
- PowerShell Script
- Procedure
- Unix Script
- Utility
- Windows Script

Refer to **Template Usage** for more details.

Note

Additional host script language types which can be applied to templates can be defined in RED via the **Tools > Host Script Languages** option.

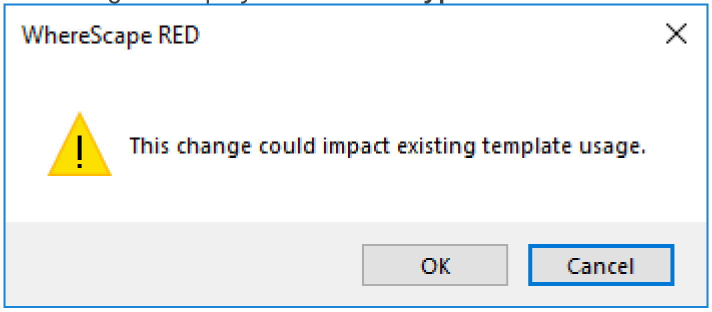
The **Type** field setting of a template that is currently IN USE can be changed but is restricted to PowerShell and user defined script types only.



Author:	Red Documentation
Target DB:	MySQL*
Type:	PowerShell (32-bit) Script

This template is IN USE.
Target DB selection is locked.
Type selection is restricted to PowerShell and user types only, if available.

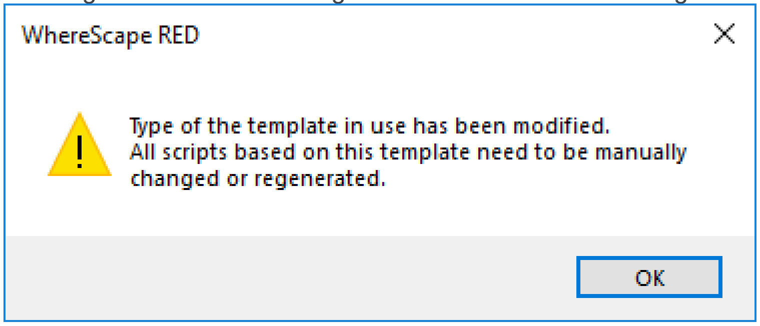
A message is displayed when the **Type** field is modified:



WhereScape RED

This change could impact existing template usage.

Clicking **OK** saves the change and a confirmation message is displayed:



WhereScape RED

Type of the template in use has been modified.
All scripts based on this template need to be manually changed or regenerated.

The **Edit Lock** fields provide information if the template is currently locked for editing by another user.

Templates cannot be deleted or modified after the editor window has been opened, unless the Edit Lock has been released. Edit Locks can be released by any user in the template Properties screen.

Properties Screen Banner

The banner message varies depending on the state of the template object.

- *<TEMPLATE NAME> - NO UPDATE: This template is READ ONLY*
  indicates that this is a shipped master template that may not be changed.
- *<TEMPLATE NAME> - NO UPDATE: Checked out by <USER>*
  indicates that the template has been checked out by a user and thus may not be edited until checked back in.
- *<TEMPLATE NAME> - NO UPDATE: Checked out by <USER>*
  indicates that the template is in use by procedures and therefore cannot be altered. The **Type** field is also locked. See notes above for exceptions.

Pebble Templates

Pebble is a scripting language that generates text output, based on provided meta data. The usage of Pebble templates in WhereScape RED does not require knowledge of Java. When looking at the Pebble documentation ignore the references to Java, including information about setting up Pebble in Java.

Operation Types

Create DDL

- env
- table
- options
- settings
- Types

Custom Procedure

- env
- table
- options
- settings
- Types

Export Script

- env
- table
- options
- settings
- Types

Index Create DDL

- env
- index
- options
- settings
- Types

Index Drop DDL

- env
- index
- options
- settings
- Types

Load Script

- env
- table
- options
- settings
- Types

Table Alter DDL

- env
- changes
- actualTable
- table
- options
- settings
- Types

Update Procedure

- env
- table
- options
- settings
- Types

Types

ActualTable

Name	Nullable	Type	Notes
catalog		Scalar	
schema		Scalar	
name		Scalar	
additionalInformation		Array of **AdditionalDatum**	
columns		Array of **Column**	

AdditionalDatum

Name	Nullable	Type	Notes
key		Scalar	
description		Scalar	
value		Scalar	

ChangeType

Name	Nullable	Type	Notes
Types.ChangeType.TableCreation		**TableCreation**	
Types.ChangeType.TableAdditionalInformationChange		**TableAdditionalInformationChange**	

Name	Nullable	Type	Notes
Types.ChangeType.ColumnAddition		ColumnAddition	
Types.ChangeType.ColumnDeletion		ColumnDeletion	
Types.ChangeType.ColumnNameChange		ColumnNameChange	
Types.ChangeType.ColumnDataTypeChange		ColumnDataTypeChange	
Types.ChangeType.ColumnAdditionalInformationChange		ColumnAdditionalInformationChange	

Column

Name	Nullable	Type	Notes
additive		Scalar	
artificial		Scalar	
baseDataType		Scalar	
businessKey		Scalar	
changeHashKey		Scalar	
columnId		Scalar	
createTime		Scalar	
currentFlag		Scalar	
dataType		Scalar	
dataTypeScale		Scalar	
dataTypeSize		Scalar	
defaultValue		Scalar	
description		Scalar	
displayName		Scalar	
dss		Scalar	
dssEndDate		Scalar	
dssLoadDate		Scalar	
dssRecordSource		Scalar	
dssStartDate		Scalar	
dssVersion		Scalar	
format		Scalar	
fullDataType		Scalar	
hashKeyImmediateSources	Y	Array of column	
hashKeySources	Y	Array of column	
hubHashKey		Scalar	
keyType		enum	
linkHashKey		Scalar	
lookupKey		Scalar	
name		Scalar	
nullAllowed		Scalar	

Name	Nullable	Type	Notes
numeric		Scalar	
partition		Scalar	
rangeEnd		Scalar	
rangeStart		Scalar	
slowlyChanging		Scalar	
source		Scalar	
sourceColumn	Y	Link to column identified by name	
sourceSystem		Scalar	
sourceTable	Y	Link to table identified by name	
table		table	
transform		Scalar	
transformType		enum	
updateTime		Scalar	
zeroKeyValue		Scalar	

ColumnAddition

Name	Nullable	Type	Notes
columnName		Scalar	

ColumnAdditionalInformationChange

Name	Nullable	Type	Notes
columnName		Scalar	
changedKey		Scalar	
changedDescription		Scalar	
expectedValue		Scalar	
actualValue		Scalar	

ColumnDataTypeChange

Name	Nullable	Type	Notes
columnName		Scalar	
dataType		Scalar	
actualDataType		Scalar	

ColumnDeletion

Name	Nullable	Type	Notes
columnName		Scalar	

ColumnJoin

Name	Nullable	Type	Notes
fromColumn		Link to column identified by name	
toColumn		Link to column identified by name	

ColumnNameChange

Name	Nullable	Type	Notes
columnName		Scalar	
actualColumnName		Scalar	

DateBasedPartitioningInfo

Name	Nullable	Type	Notes
partitionType		enum	
dateDimensionColumn	Y	Link to column identified by name	
startDateKey		Scalar	
endDateKey		Scalar	

EnumValue

Name	Nullable	Type	Notes
name		Scalar	
code		Scalar	

Env

Name	Nullable	Type	Notes
licensedTo		Scalar	
productVersion		Scalar	
userName		Scalar	
metaSchemaPrefix		Scalar	
loginName		Scalar	
currentTimestamp		Scalar	

ExportInfo

Name	Nullable	Type	Notes
connectionType		enum	
exportConnection		Link to connection identified by name	
sourceDbType		enum	
file		**FileInfo**	

Name	Nullable	Type	Notes
routine		Scalar	
format		Scalar	
headerRow	Y	enum	
options		Scalar	
triggerFile		**FileInfo**	
triggerParameter1		Scalar	
triggerParameter2		Scalar	
triggerParameter3		Scalar	
compressAfterExport		Scalar	
compressUtility		**FileInfo**	
compressParameters		Scalar	
whereClause		Scalar	

ExtendedPropertyDefinition

Name	Nullable	Type	Notes
id		Scalar	
variableName		Scalar	
displayName		Scalar	
description		Scalar	
isMasked		Scalar	
scopes		Array of **ExtendedPropertyScope**	

ExtendedPropertyScope

Name	Nullable	Type	Notes
id		Scalar	
connectionType		enum	
objectType		enum	
databaseType		enum	

FK

Name	Nullable	Type	Notes
fromTable		Link to table identified by name	
fromColumn		Link to column identified by name	
toTable		Link to table identified by name	
toColumn		Link to column identified by name	

FileInfo

Name	Nullable	Type	Notes

Name	Nullable	Type	Notes
path		Scalar	
name		Scalar	
charSet		Scalar	
fieldDelimiter		Scalar	
fieldEnclosure		Scalar	
recordDelimiter		Scalar	
headerLine		Scalar	
escapeEncoding		Scalar	
nullEncoding		Scalar	
nonStringNullEncoding		Scalar	

HiveNativeDelimitedRowFormat

Name	Nullable	Type	Notes
delimiter		Scalar	
escapedBy		Scalar	
nullAs		Scalar	
lineTerminator		Scalar	
additionalProperties		Scalar	

HiveNativeFileFormat

Name	Nullable	Type	Notes
storedAs		Scalar	
inputFormatClass		Scalar	
outputFormatClass		Scalar	

HiveNativeSerdeRowFormat

Name	Nullable	Type	Notes
serdeName		Scalar	
serdeProperties		Scalar	

HiveNativeStorage

Name	Nullable	Type	Notes
rowFormat		Scalar	
delimitedRowFormat		**HiveNativeDelimitedRowFormat**	
serdeRowFormat		**HiveNativeSerdeRowFormat**	
fileFormat		**HiveNativeFileFormat**	

HiveNonNativeStorage

Name	Nullable	Type	Notes
storageClass		Scalar	
serdeProperties		Scalar	

HiveStorage

Name	Nullable	Type	Notes
external		Scalar	
externalLocation		Scalar	
clustered		Scalar	
clusterColumns		Array of column	
clusterSortColumns		Array of column	
clusterBuckets		Scalar	
skew		Scalar	
skewAsDirectories		Scalar	
skewColumns		Array of column	
skewValues		Scalar	
skipFirstRow		Scalar	
native		Scalar	
nativeStorage		**HiveNativeStorage**	
nonNativeStorage		**HiveNonNativeStorage**	
additionalTableProperties		Map of Scalar	

LoadInfo

Name	Nullable	Type	Notes
loadType		enum	
sourceConnection	Y	Link to connection identified by name	
sourceConnectionType		enum	
scriptConnection	Y	Link to connection identified by name	
fileParsed		Scalar	
sourceFile		**FileInfo**	
sqoopSourceFileEnclosureType		enum	
triggerFile		**FileInfo**	
triggerParameterNamePrefix		Scalar	
wait		Scalar	
waitSeconds		Scalar	
waitAction		enum	
archiveFile		**FileInfo**	
triggerArchiveFile		**FileInfo**	
tptHadoopHost		Scalar	

Name	Nullable	Type	Notes
pdwOptions		PDWLoadOptions	
selectDistinctValues		Scalar	
sourceSchema		Scalar	
sourceTables		Scalar	
useOverrideSourceColumns		Scalar	
overrideSourceColumns		Scalar	
whereAndGroupByClauses		Scalar	
overrideLoadSQL		Scalar	
fileLoaderOptions		Scalar	
tempDatabase		Scalar	
tptOptions		TPTLoadOptions	
ssisOptions		SSISLoadOptions	
sqoopOptions		SqoopLoadOptions	

Options

Name	Nullable	Type	Notes
endOfStatement		Scalar	
extendedPropertyScopesById		Map of ExtendedPropertyScope	
extendedPropertyDefinitionsByName		Map of ExtendedPropertyDefinition	
extendedPropertyDefinitions		Array of definitions	

PDWLoadOptions

Name	Nullable	Type	Notes
loadMode		Scalar	
upsertKeyColumns		Scalar	
useStagingDatabase		Scalar	
stagingDatabase		Scalar	
useMultiTransMode		Scalar	

Relation

Name	Nullable	Type	Notes
joinLookup		Scalar	
fk		Scalar	
fks		Array of FK	
ansi		Scalar	
noWarningForUnmatchedBusinessKeys		Scalar	
detailLogForUnmatchedBusinessKeys		Scalar	
useDateColForTrackingChangingDimTable		Scalar	

Name	Nullable	Type	Notes
autoAddUnmatchedBusinessKeys		Scalar	
autoAddProcedure		Scalar	
identifyUnmatchedBusinessKeyMethod		Scalar	
logWhenAutoAdd		Scalar	
currentFlag		Scalar	
trackingDateColumn	Y	column	

Relationship

Name	Nullable	Type	Notes
joinType		enum	
fromTable		Link to table identified by name	
toTable		Link to table identified by name	
columnJoins		Array of **ColumnJoin**	
primaryJoin		Scalar	

SSISLoadOptions

Name	Nullable	Type	Notes
sourceIdentifierEncapsulation		Scalar	
sourceAlwaysUseDefaultCodePage		Scalar	
destinationAlwaysUseDefaultCodePage		Scalar	
useRowCountLog		Scalar	
loadMode		Scalar	
upsertKeyColumns		Scalar	
rollbackOnTableFailure		Scalar	
useStagingDatabase		Scalar	
stagingDatabase		Scalar	
acquireTableLock		Scalar	
commitInterval		Scalar	
sourceIdentifierCaseConversion		enum	
destinationIdentifierCaseConversion		enum	
sourceCodePage	Y	Scalar	
destinationCodePage	Y	Scalar	
rowsPerBatch	Y	Scalar	

SourceJoinDetails

Name	Nullable	Type	Notes
distinct		Scalar	
groupBy		Scalar	
join		Scalar	

Name	Nullable	Type	Notes
where		Scalar	

SourceTableInfo

Name	Nullable	Type	Notes
sourceTable		Link to table identified by name	
lockingMethod	Y	Scalar	

SqoopLoadOptions

Name	Nullable	Type	Notes
baseTempDir		Scalar	
genericArguments		Scalar	
toolArguments		Scalar	
splitByColumn	Y	Scalar	

Storage

Name	Nullable	Type	Notes
dataBlockSize		Scalar	
distributionColumns		Array of column	
distributionMethod		Scalar	
fallback		Scalar	
freeSpace		Scalar	
multiSet		Scalar	
optionalCreateClause		Scalar	
organizingColumns		Array of column	
partitionFunction		Scalar	
partitionScheme		Scalar	
partitioned		Scalar	
partitioningColumns		Array of column	
storageIndex	Y	index	
hiveStorage		**HiveStorage**	

TPTLoadOptions

Name	Nullable	Type	Notes
tptRoutine		Scalar	
characterSet		Scalar	
odbcOperatorAttributes		Scalar	
loadOperatorAttributes		Scalar	
jobName		Scalar	

Name	Nullable	Type	Notes
tbuildOptions		Scalar	
tlogviewOptions		Scalar	
sharedMemorySize		Scalar	
sessionsMin		Scalar	
readInstances		Scalar	
writeInstances		Scalar	
sessionsMax	Y	Scalar	

Table

Name	Nullable	Type	Notes
id		Scalar	
columns		Array of column	
columnsByName		Map of **Column**	
database		Scalar	
dbType		enum	
indexes		Array of Link to index identified by name	
name		Scalar	
objectType		enum	
relations		Array of **Relation**	
relationships		Array of **Relationship**	
schema		Scalar	
shortName		Scalar	
displayName		Scalar	
description		Scalar	
sourceJoinDetails		**SourceJoinDetails**	
storage		**Storage**	
subType		Scalar	
target		Link to target identified by name	
updateProcedure	Y	Link to procedure identified by name	
customProcedure	Y	Link to procedure identified by name	
getFunction	Y	Link to procedure identified by name	
postLoadProcedure	Y	Link to procedure identified by name	
loadScript	Y	Link to host script identified by name	
postExportProcedure	Y	Link to procedure identified by name	

Name	Nullable	Type	Notes
exportScript	Y	Link to host script identified by name	
loadInfo	Y	**LoadInfo**	
exportInfo	Y	**ExportInfo**	
viewInfo	Y	**ViewInfo**	
extendedPropertyValuesByName		Map of Scalar	
sourceMappings	Y	Array of Link to source mapping identified by sourceMappingName	

TableAdditionalInformationChange

Name	Nullable	Type	Notes
changedKey		Scalar	
changedDescription		Scalar	
expectedValue		Scalar	
actualValue		Scalar	

TableCreation

Name	Nullable	Type	Notes

UpdateProcSettings

Name	Nullable	Type	Notes
batchDateGrain		Scalar	
batchJoinTableColumn		Scalar	
batchProcessingField		Scalar	
changeDetectionEndCurrent		Scalar	
changeDetectionEndExpiring		Scalar	
changeDetectionStartInitial		Scalar	
changeDetectionStartNew		Scalar	
changeDetectionUpdateCurrentRecordsOnly		Scalar	
createStorageIndexAfterCTAS		Scalar	
dateBasedPartitioningInfo		**DateBasedPartitioningInfo**	
deleteBeforeInsert		Scalar	
deleteBeforeInsertMultiPass		Scalar	
deleteBeforeInsertTruncate		Scalar	
deleteRowsNotInSource		Scalar	
deleteTruncateOption		Scalar	

Name	Nullable	Type	Notes
deleteWhereClause		Scalar	
dssCurrentFlagColumnName		Scalar	
dssEndDateColumnName		Scalar	
dssStartDateColumnName		Scalar	
enableParallelDml		Scalar	
includeExplicitLock		Scalar	
includeInitialLoadInsert		Scalar	
insert		Scalar	
insertHint		Scalar	
insertMinusHint		Scalar	
insertNewRowIdentificationMethodExcept		Scalar	
insertNewRowsOnly		Scalar	
insertZeroKeyRecord		Scalar	
lockClause		Scalar	
merge		Scalar	
mergeChangedRowsOnly		Scalar	
mergeHint		Scalar	
mergeMinusHint		Scalar	
mergeNewRowIdentificationMethodExcept		Scalar	
minimalLogging		Scalar	
nullSupport		Scalar	
parameters		Array of Scalar	
partitionExchangeTable	Y	Link to table identified by name	
procedureName		Scalar	
processByBatch		Scalar	
selectHint		Scalar	
sourceJoinDetails		**SourceJoinDetails**	
sourceTablesByName		Map of **SourceTableInfo**	
template	Y	template	
update		Scalar	
updateChangeRowIdentificationMethodExcept		Scalar	
updateChangedRowsOnly		Scalar	
updateCurrentRowsOnly		Scalar	DEPRECATED: Use updateChangedRowsOnly instead
updateHint		Scalar	
updateMinusHint		Scalar	
warnOnDelete		Scalar	

VanillaSettings

Name	Nullable	Type	Notes
template	Y	template	

ViewInfo

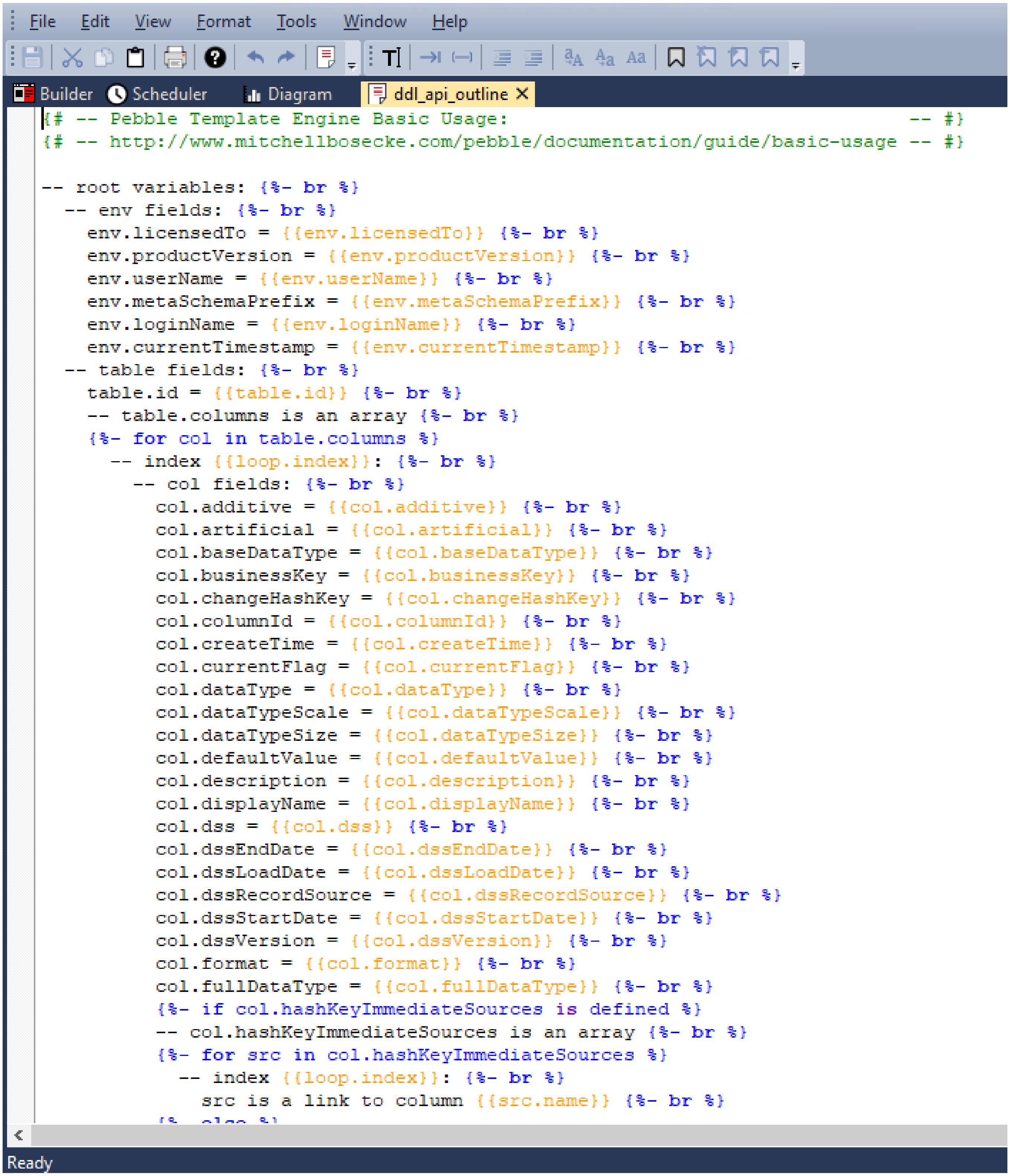
Name	Nullable	Type	Notes
viewType		enum	
distinct		Scalar	
whereClause		Scalar	
userDefined		Scalar	
userDefinition		Scalar	

Template Editor

Right-click a template and select **Edit the Template** or **View Template** to open the Template editor.

Similar to the procedure and script editors, the template editor includes the following features:

- Pebble syntax highlighting
- improved find feature (repeated find, up and down, etc.)
- toggle the display of line numbering
- a status bar that includes line number, line length, selected text length, etc.
- a context sensitive toolbar
- tab to space conversion and vice versa
- uppercase, lowercase, titlecase text conversion command
- increase/decrease test indent command
- bookmark support



```
{# -- Pebble Template Engine Basic Usage:                                   -- #}
{# -- http://www.mitchellbosecke.com/pebble/documentation/guide/basic-usage -- #}

-- root variables: {%- br %}
  -- env fields: {%- br %}
    env.licensedTo = {{env.licensedTo}} {%- br %}
    env.productVersion = {{env.productVersion}} {%- br %}
    env.userName = {{env.userName}} {%- br %}
    env.metaSchemaPrefix = {{env.metaSchemaPrefix}} {%- br %}
    env.loginName = {{env.loginName}} {%- br %}
    env.currentTimestamp = {{env.currentTimestamp}} {%- br %}
  -- table fields: {%- br %}
    table.id = {{table.id}} {%- br %}
    -- table.columns is an array {%- br %}
    {%- for col in table.columns %}
      -- index {{loop.index}}: {%- br %}
        -- col fields: {%- br %}
          col.additive = {{col.additive}} {%- br %}
          col.artificial = {{col.artificial}} {%- br %}
          col.baseDataType = {{col.baseDataType}} {%- br %}
          col.businessKey = {{col.businessKey}} {%- br %}
          col.changeHashKey = {{col.changeHashKey}} {%- br %}
          col.columnId = {{col.columnId}} {%- br %}
          col.createTime = {{col.createTime}} {%- br %}
          col.currentFlag = {{col.currentFlag}} {%- br %}
          col.dataType = {{col.dataType}} {%- br %}
          col.dataTypeScale = {{col.dataTypeScale}} {%- br %}
          col.dataTypeSize = {{col.dataTypeSize}} {%- br %}
          col.defaultValue = {{col.defaultValue}} {%- br %}
          col.description = {{col.description}} {%- br %}
          col.displayName = {{col.displayName}} {%- br %}
          col.dss = {{col.dss}} {%- br %}
          col.dssEndDate = {{col.dssEndDate}} {%- br %}
          col.dssLoadDate = {{col.dssLoadDate}} {%- br %}
          col.dssRecordSource = {{col.dssRecordSource}} {%- br %}
          col.dssStartDate = {{col.dssStartDate}} {%- br %}
          col.dssVersion = {{col.dssVersion}} {%- br %}
          col.format = {{col.format}} {%- br %}
          col.fullDataType = {{col.fullDataType}} {%- br %}
          {%- if col.hashKeyImmediateSources is defined %}
          -- col.hashKeyImmediateSources is an array {%- br %}
          {%- for src in col.hashKeyImmediateSources %}
            -- index {{loop.index}}: {%- br %}
              src is a link to column {{src.name}} {%- br %}
          {%- else %}
```

Notes

Editing or deleting Locked for Edit or opened Templates

- Templates cannot be deleted if they are Locked for Edit or checked out by another user.
- Saving the template cannot be performed if the template is Locked for Edit by other users after the edit window has been opened.
- Templates cannot be deleted or modified after the editor window has been opened, unless the Edit Lock

Notes

has been released. Edit Locks can be released by any user in the Template Properties screen.

- To prevent updates, deletes and modifications to certain Templates, it is best to use the Check Out functionality instead. For more information about this functionality, refer to **Object Check Outs and Check Ins** for details.

Evaluating an API Outline Template

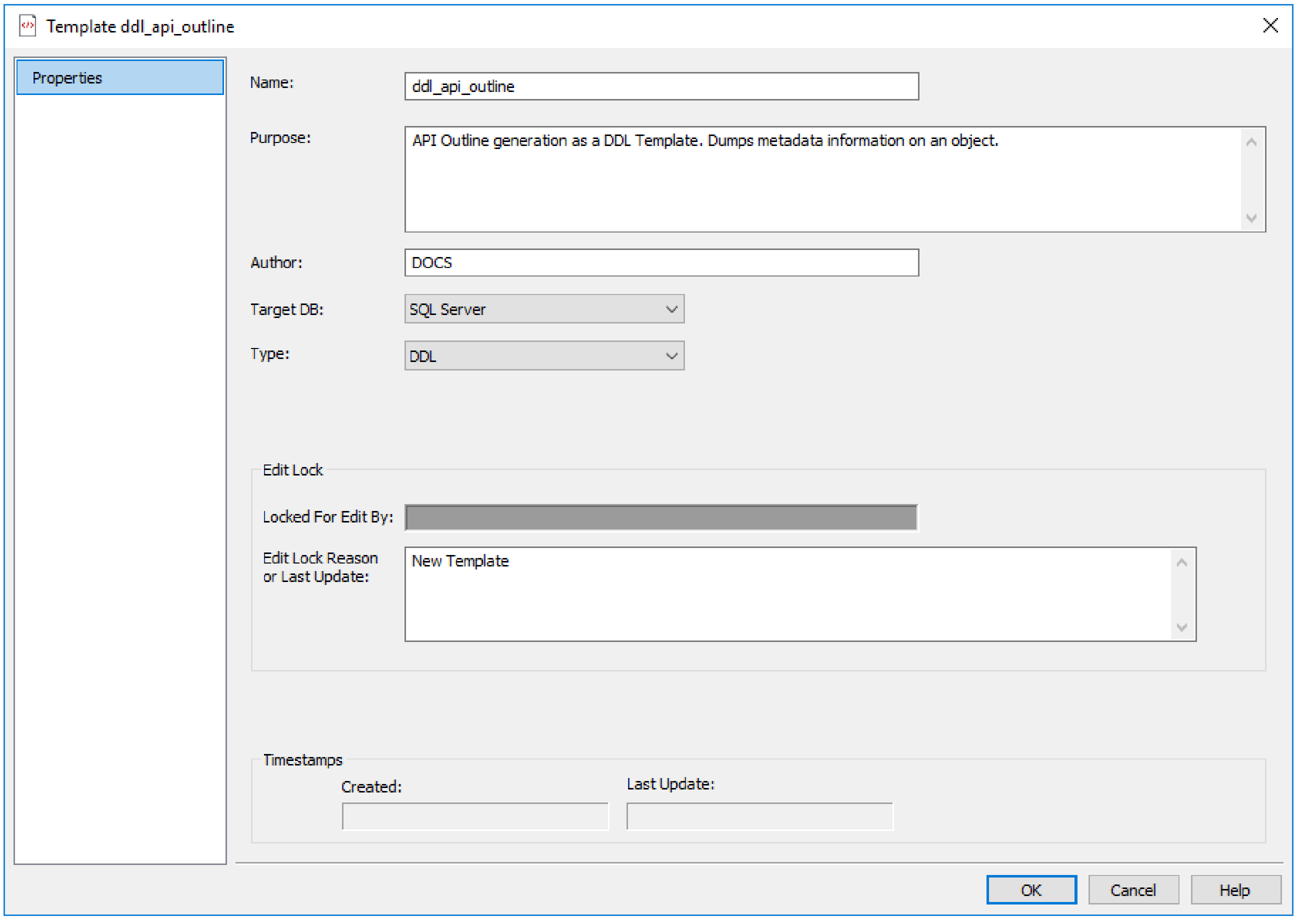
An API Outline Template is available to output all object properties relevant to the current object. Upon evaluation of this template, the status of each property is generated and printed to the script or procedure file.

Note

Template evaluation usually generates a script or procedure file, but the API Outline Template generates plain text. The output of this template is intended to be viewed or copied to a text file, it cannot be executed as a script.

To evaluate an API Outline Template:

1. Create a **new template**. The template can normally be of any type, but in this example we'll use the DDL template type as viewing the evaluation of DDL templates is simple.
   Set **Target DB** to your source connection database type. In this example, we will set the Target DB to SQL Server because the load table we are evaluating is stored on a SQL Server connection.



Template ddl_api_outline

Name: ddl_api_outline

Purpose: API Outline generation as a DDL Template. Dumps metadata information on an object.

Author: DOCS

Target DB: SQL Server

Type: DDL

Edit Lock

Locked For Edit By:

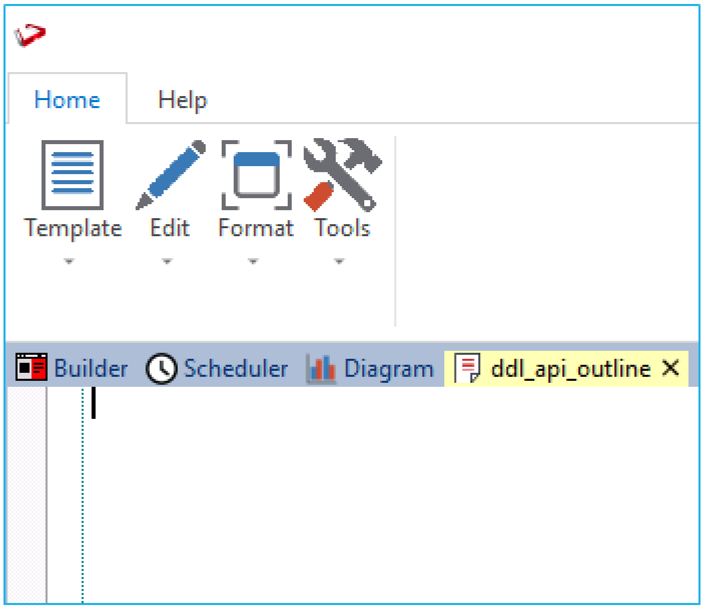
Edit Lock Reason or Last Update: New Template

Timestamps

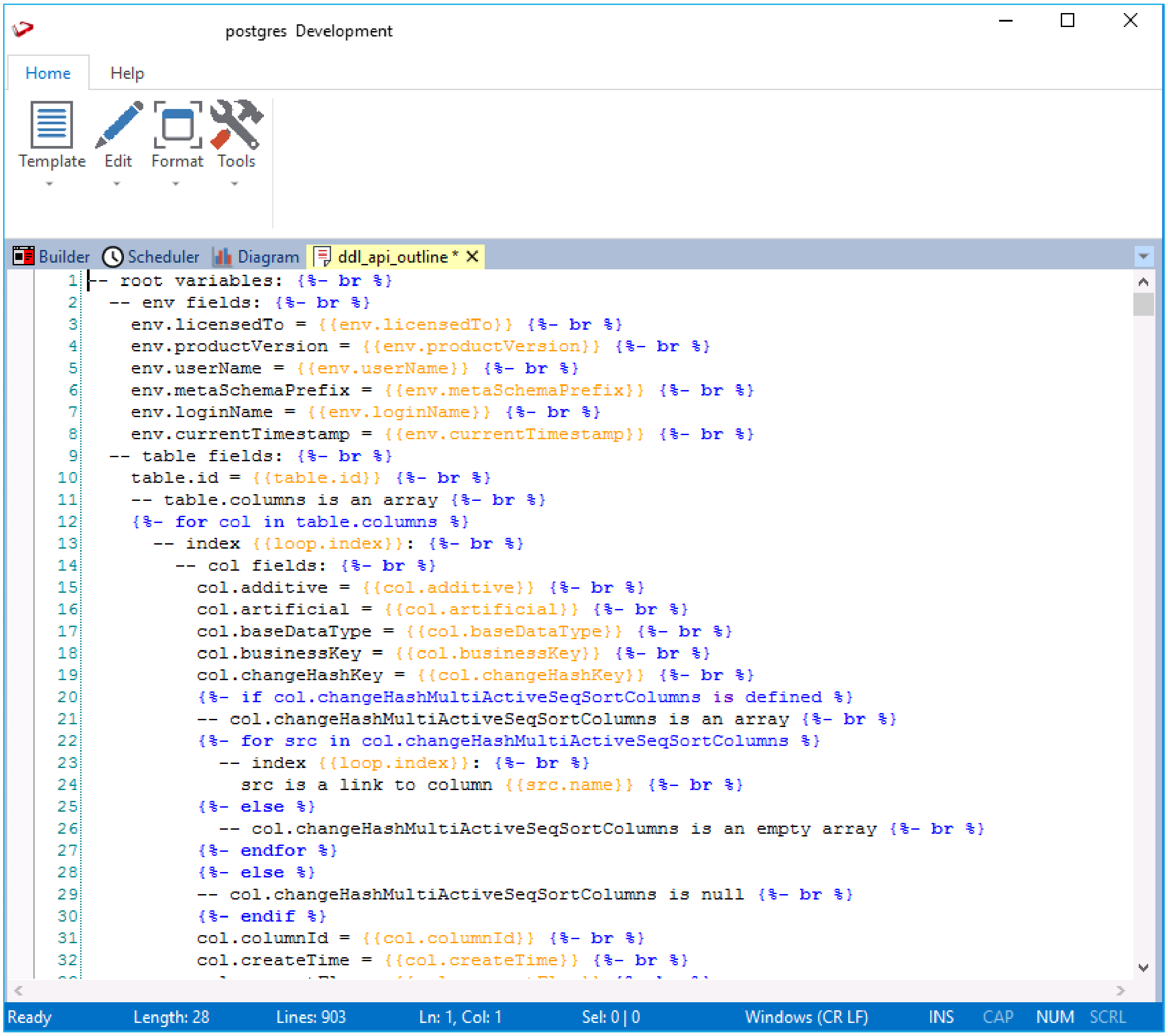
Created:

Last Update:

2. Open the template in the Template Editor.



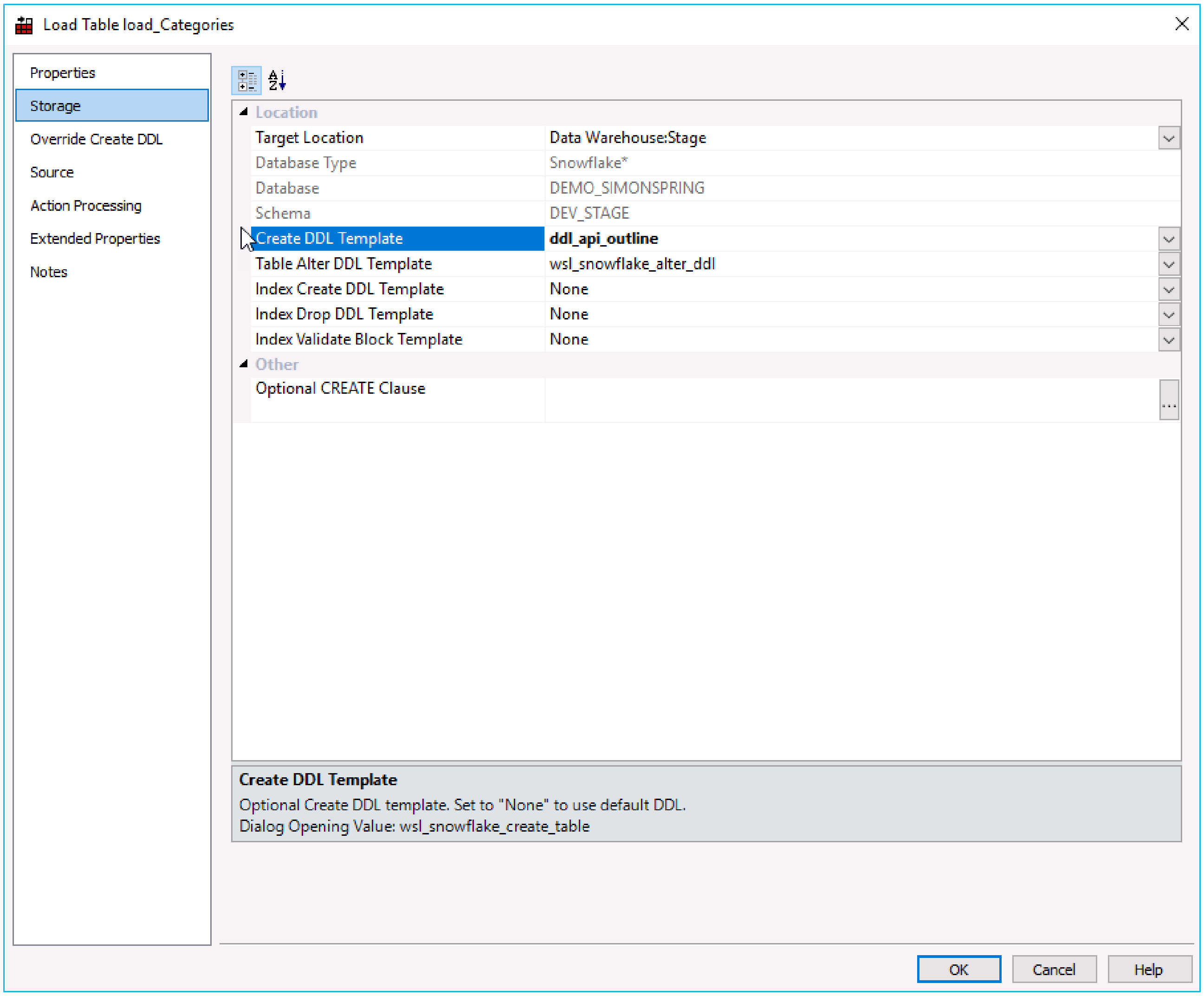
3. Click **Tools > Create API Example Outline**. The API Example Outline text is added to the blank template.



```
-- root variables: {%- br %}
  -- env fields: {%- br %}
    env.licensedTo = {{env.licensedTo}} {%- br %}
    env.productVersion = {{env.productVersion}} {%- br %}
    env.userName = {{env.userName}} {%- br %}
    env.metaSchemaPrefix = {{env.metaSchemaPrefix}} {%- br %}
    env.loginName = {{env.loginName}} {%- br %}
    env.currentTimestamp = {{env.currentTimestamp}} {%- br %}
  -- table fields: {%- br %}
    table.id = {{table.id}} {%- br %}
    -- table.columns is an array {%- br %}
    {%- for col in table.columns %}
      -- index {{loop.index}}: {%- br %}
        -- col fields: {%- br %}
          col.additive = {{col.additive}} {%- br %}
          col.artificial = {{col.artificial}} {%- br %}
          col.baseDataType = {{col.baseDataType}} {%- br %}
          col.businessKey = {{col.businessKey}} {%- br %}
          col.changeHashKey = {{col.changeHashKey}} {%- br %}
          {%- if col.changeHashMultiActiveSeqSortColumns is defined %}
          -- col.changeHashMultiActiveSeqSortColumns is an array {%- br %}
          {%- for src in col.changeHashMultiActiveSeqSortColumns %}
            -- index {{loop.index}}: {%- br %}
              src is a link to column {{src.name}} {%- br %}
          {%- else %}
            -- col.changeHashMultiActiveSeqSortColumns is an empty array {%- br %}
          {%- endfor %}
          {%- else %}
          -- col.changeHashMultiActiveSeqSortColumns is null {%- br %}
          {%- endif %}
          col.columnId = {{col.columnId}} {%- br %}
          col.createTime = {{col.createTime}} {%- br %}
```

Ready Length: 28 Lines: 903 Ln: 1, Col: 1 Sel: 0 | 0 Windows (CR LF) INS CAP NUM SCRL

4. **Save** and **close** the template.
5. Open the Properties dialog for the **Load Table** you wish to evaluate.
6. In the **Storage** tab, select the template you created in the **Create DDL Template** drop-down box.



Load Table load_Categories

Location

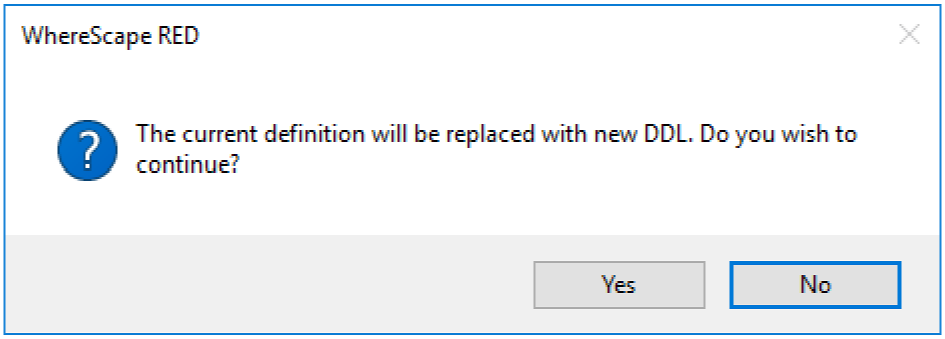
Target Location	Data Warehouse:Stage
Database Type	Snowflake*
Database	DEMO_SIMONSPRING
Schema	DEV_STAGE
Create DDL Template	ddl_api_outline
Table Alter DDL Template	wsl_snowflake_alter_ddl
Index Create DDL Template	None
Index Drop DDL Template	None
Index Validate Block Template	None

Other

Optional CREATE Clause	

Create DDL Template
Optional Create DDL template. Set to "None" to use default DDL.
Dialog Opening Value: wsl_snowflake_create_table

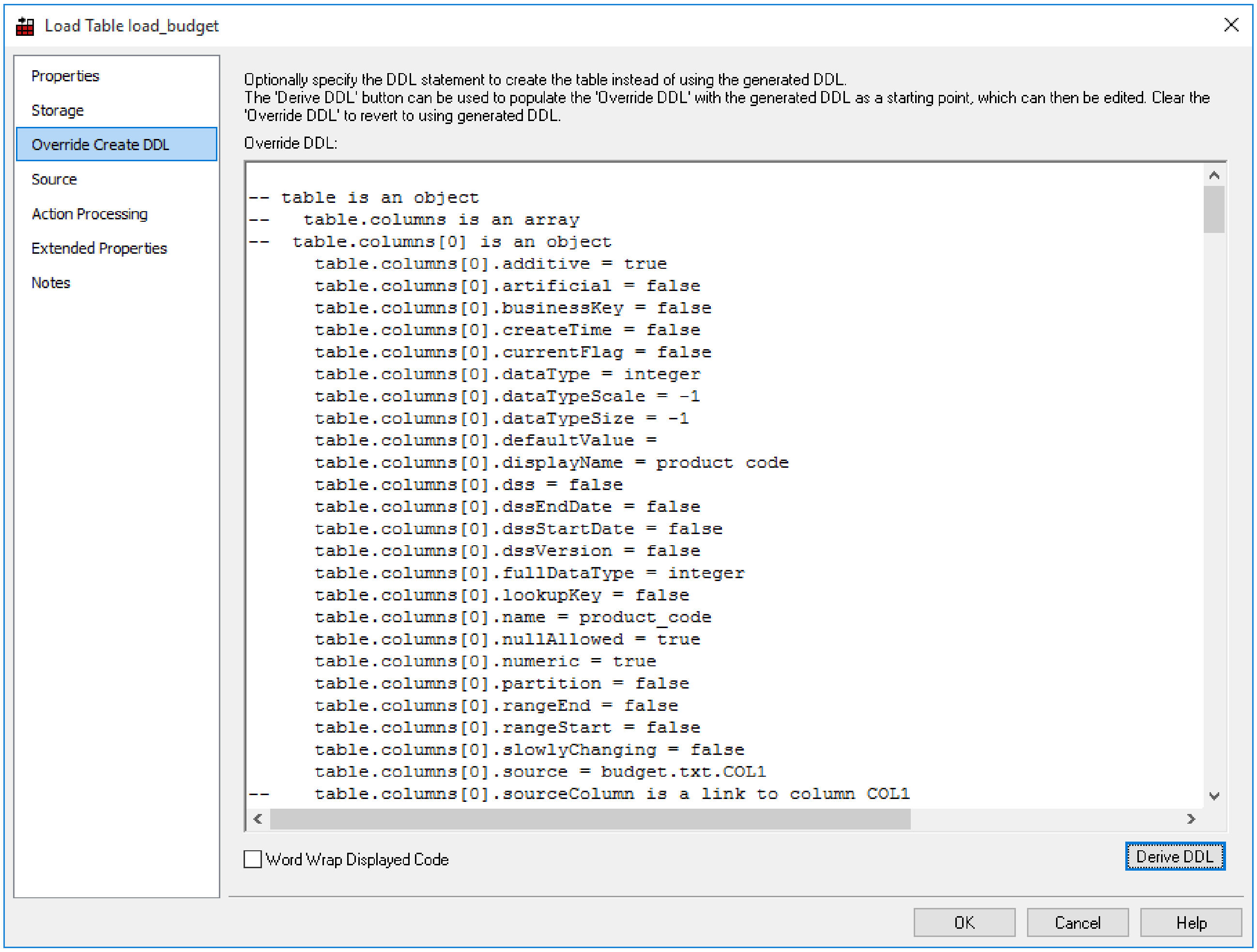
7. Open the **Override Create DDL** tab. If the **Override DDL** field is populated with a custom DDL statement, copy and paste this statement to a text file for backup purposes.
8. Click the **Derive DDL** button. A warning dialog box displays informing you that the association of the DDL template with this table will be saved to the metadata, click **OK**.



WhereScape RED

The current definition will be replaced with new DDL. Do you wish to continue?

9. The results of the API Outline Template are printed to the **Override DDL** text box. Cut or copy this text to a text editor and save as a text file for reference purposes.



Optionally specify the DDL statement to create the table instead of using the generated DDL.
The 'Derive DDL' button can be used to populate the 'Override DDL' with the generated DDL as a starting point, which can then be edited. Clear the 'Override DDL' to revert to using generated DDL.

Override DDL:

```
-- table is an object
--   table.columns is an array
--  table.columns[0] is an object
      table.columns[0].additive = true
      table.columns[0].artificial = false
      table.columns[0].businessKey = false
      table.columns[0].createTime = false
      table.columns[0].currentFlag = false
      table.columns[0].dataType = integer
      table.columns[0].dataTypeScale = -1
      table.columns[0].dataTypeSize = -1
      table.columns[0].defaultValue =
      table.columns[0].displayName = product code
      table.columns[0].dss = false
      table.columns[0].dssEndDate = false
      table.columns[0].dssStartDate = false
      table.columns[0].dssVersion = false
      table.columns[0].fullDataType = integer
      table.columns[0].lookupKey = false
      table.columns[0].name = product_code
      table.columns[0].nullAllowed = true
      table.columns[0].numeric = true
      table.columns[0].partition = false
      table.columns[0].rangeEnd = false
      table.columns[0].rangeStart = false
      table.columns[0].slowlyChanging = false
      table.columns[0].source = budget.txt.COL1
--    table.columns[0].sourceColumn is a link to column COL1
```

□ Word Wrap Displayed Code

10. Click **Cancel** in the Load table Properties dialog.

Note

Ensure the Load table properties are returned to their previous state. The default value for **Storage > Create DDL Template** is **None** and the **Override Create DDL > Override DDL** field is left blank to use the automatically generated DDL statement or ensure your custom DDL statement remains.

Pebble Syntax

Notes

- Pebble syntaxes documented in this section are common between WhereScape products, but some syntaxes are unique to WhereScape 3D and WhereScape RED.
- It is advisable that you do not copy contents of a Pebble template from one WhereScape product to another.

Controls

Controls	Description	Example

Controls	Description	Example

{%...%}	This is a tag for control flow. It should contain a tag and may be additional parameters, as shown in the example.	`{% if ${condition} %}`
		`${content}`
		`{% else %}`
		`${content}`
		`{% endif %}`

{%-...-%}	This is a tag for control flow. It should contain a tag and may be additional parameters, as shown in the example. If there is a **-** at the start or at the end, this will trim the leading or trailing whitespace respectively.	`{%- if ${condition} -%}`
		`${content}`
		`{%- else -%}`
		`${content}`
		`{%- endif -%}`

{{...}}	This tag is for expressions. It should contain an expression that will be outputted, as shown in the example.	`{{ max(5, 3) }}`
{{-...-}}	This tag is for expressions. It should contain an expression that will be outputted, as shown in the example. If there is a **-** at the start or at the end, this will trim the leading or trailing whitespace respectively.	`{{- range(3, 6) -}}`
{#...#}	This tag is for comments.	`{# this is a comment. #}`
{#-...-#}	This tag is for comments. If there is a **-** at the start or at the end, this will trim the leading or trailing whitespace respectively.	`{#- No whitespace on the left #}`

Pebble Template Tags

autoescape

The autoescape tag can be used to temporarily disable/re-enable the autoescaper as well as change the escaping strategy for a portion of the template.

Example

```
{% autoescape "js" %}
${content}
{% endautoescape %}
```

block

A section that can be overridden by a child template.

Template

```
{% block "${blockName}" %}
```

```
${content}
{% endblock %}
```

extends

The extends tag is used to declare a parent template. It should be the very first tag used in a child template and a child template can only extend up to one parent template.

Template

```
{% extends "${templateName}" %}
```

filter

The filter tag allows you to apply a filter to a large chunk of template.

Template

```
{% filter ${filter} %}
${content}
{% endfilter %}
```

Template 2

```
{% filter ${filter1} | ${filter2} %}
${content}
{% endfilter %}
```

for

The for tag is used to iterate through a collection or map.

Template

```
{% for ${i} in ${collection} %}
${content}
{% endfor %}
```

for else

The for tag is used to iterate through a collection or map with a convenient way to check for emptiness.

Template

```
{% for ${i} in ${collection} %}
${content}
{% else %}
${content}
{% endfor %}
```

if

The if tag allows you to designate a chunk of content as conditional depending on the result of an expression.

Template

```
{% if ${condition} %}
${content}
{% endif %}
```

if else

The if tag allows you to designate a chunk of content as conditional depending on the result of an expression.

Template

```
{% if ${condition} %}
${content}
{% else %}
${content}
{% endif %}
```

if elseif

The if tag allows you to designate a chunk of content as conditional depending on the result of an expression.

Template

```
{% if ${condition} %}
${content}
{% elseif %}
${content}
{% else %}
${content}
{% endif %}
```

import

The import tag allows you to use macros defined in another template.

Template

```
{% import "${templateName}" %}
```

include

The include tag allows you to insert the rendered output of another template directly into the current template. The included template will have access to the same variables that the current template does.

Template

```
{% include "${templateName}" %}
```

include with

The include tag allows you to insert the rendered output of another template directly into the current template. The included template will have access to the same variables that the current template does with additional variables.

Template

```
{% include "${templateName}" with { ${mapOfVariables} } %}
```

macro

The macro tag allows you to create a chunk of reusable and dynamic content. The macro can be called multiple times in the current template or even from another template with the help of the import tag.

Example

```
{% macro concat(left, right) %}
{{ left }} {{ right }}
{% endmacro %}
```

parallel

The parallel tag allows you to designate a chunk of content to be rendered using a new thread.

Template

```
{% parallel %}
${content}
{% endparallel %}
```

set

The set tag allows you to define a variable in the current context, whether it currently exists or not.

Will initialize the variable or override the current value in the variable.

Template

```
{% set ${variable} = ${value} %}
```

verbatim

The verbatim tag allows you to write Pebble syntax that won't be parsed.

Example

```
{% verbatim %}
{% if not varA is defined %}
{% set varA = "Abc" %}
{% endif %}
{% endverbatim %}
```

br

The br tag adds a line break token at the current position.

Use an additional parameter to change the type of line break: lf (Default), crlf, other "token".

Template

```
{% br %}
{% br lf %}
{% br crlf %}
{% br other "|" %}
```

counter

The counter tag outputs an integer value that is incremented by one for each use.

Use the parameter local to make the counter increments per used template.

Example

```
{% counter %} pencil
{% counter %} pens
```

Output

```
1 pencil
2 pens
```

Example 2

```
Template "Other": {% counter local %}
Main template: {% counter local %}{% include "Other" %}
```

Outputs

```
11
```

declare

The declare tag allows you to define a variable in the current context if it does not exist. If the variable does exist it will not be changed.

Note

The following are the same:

```
{% declare encapsulation = true %}{% set encapsulation = encapsulation | default(true) %}
```

Example

```
{% set varA = "aa" %}
{% declare varA = "bb" %}
{% declare varB = "cc" %}
```

```
{{ varA }} {{ varB }}
```

Outputs

```
aa cc
```

error

The error tag allows you to fail generating the template with an error message, e.g. if something unexpected occurs.

Example

```
{% if not requiredVar is defined %}
{% error "Something terrible has occurred" %}
```

indent

The indent tag adds one or more 4-space indents at the current position.

Example

```
{% indent %}
```

Example 2

```
{% indent 3 %}
```

from

The from tag is used to iterate a subset of a collection defined by the where condition.

Template

```
{% from ${collection} as ${var} where ${test} %}
${content}
{% endfrom %}
```

Example

```
{% from range(1,10) as num where num is even %}
{{- num }}{% br %}
{% endfrom %}
```

Output

```
2
4
6
8
10
```

from else

The from tag is used to iterate a subset of a collection defined by the where condition with a convenient way to check for emptiness.

Template

```
{% from ${collection} as ${var} where ${test} %}
${content}
{% else %}
${content}
{% endfrom %}
```

Example

```
{% from [] as num where num is even -%}
Even numbers{% br %}
{% else -%}
Empty collection{% endfrom %}
```

Output

```
Empty collection
```

list add

The list with add tag is used to append a value to a list.

Example

```
{% set alphabet = ["a", "b", "c"] %}
{{ alphabet }}{% br %}
{% list alphabet add "d" -%}
{{ alphabet }}
```

Outputs

```
[a, b, c]
[a, b, c, d]
```

list adds

The list with add tag is used to append one or more values to a list.

Example

```
{% set alphabet = ["a", "b", "c"] %}
{{ alphabet }}{% br %}
{% list alphabet add "d", "e", "f" -%}
{{ alphabet }}
```

Outputs

```
[a, b, c]
[a, b, c, d, e, f]
```

list remove

The list with remove tag is used to remove a value from a list.

Example

```
{% set alphabet = ["a", "b", "c", "d", "e", "f"] %}
{{ alphabet }}{% br %}
{% list alphabet remove "d" -%}
{{ alphabet }}
```

Outputs

```
[a, b, c, d, e, f]
[a, b, c, e, f]
```

list removes

The list with remove tag is used to remove one or more values from a list.

Example

```
{% set alphabet = ["a", "b", "c", "d", "e", "f"] %}
{{ alphabet }}{% br %}
{% list alphabet remove "d", "e", "f" -%}
{{ alphabet }}
```

Outputs

```
[a, b, c, d, e, f]
[a, b, c]
```

map put

The map put tag is used to set a key-value pair into a map, whether it currently exists or not.

Template

```
{% map ${mapVar} put ${key} = ${value} %}
```

map remove

The map remove tag is used to remove a key from a map.

Template

```
{% map ${mapVar} remove ${key} %}
```

placebottom

The placebottom tag allows defining a block in the template that will be placed at the end of the output. The priority argument defines the order of multiple place bottom blocks.

Template

```
{% placebottom ${priority} %}
${content}
{% endplacebottom %}
```

get

The get tag defines a variable as a subset of a collection based on the given condition.

Example

```
{% get evenNums from range(1,10) as num where num is even %}
{{ evenNums }}
```

Output

```
[2, 4, 6, 8, 10]
```

WhereScape RED Specific Tag

fetch

The fetch tag is used to manually load attributes of objects that which aren’t loaded automatically, eg. for an alternate table.

Example

```
{% for col in table.columns %}
 {% fetch col.sourceTable %}
 {% set srcTab = col.sourceTable %}
 {% for srcCol in srcTab.columns  %}
{{ srcCol.source }}{% br %}
 {% endfor %}
{% endfor %}
```

Pebble Template Functions

block

The block function is used to render the contents of a block tag more than once.

Example

```
{% block "testBlock" %}Some content
{% endblock %}
{{ block("testBlock") }}
```

Output

```
Some contentSome content
```

max

The max function will return the largest of it's numerical arguments.

Example

```
{{ max(5, 3) }}
```

Output

```
5
```

min

The min function will return the smallest of it's numerical arguments.

Example

```
{{ min(5, 3) }}
```

```
3
```

parent

The parent function is used inside of a block to render the content that the parent template would have rendered inside of the block had the current template not overriden it.

Template

```
{% block "${blockName}" %}
{{ parent() }}
{% endblock %}
```

range

The range function will return a list containing an arithmetic progression of numbers.

Example

```
{{ range(3, 6) }}
```

Output

```
[3, 4, 5, 6]
```

range with step

The range function will return a list containing an arithmetic progression of numbers.

Example

```
{{ range(2, 10, 2) }}
```

Output

```
[2, 4, 6, 8, 10]
```

bool

The bool function will return a boolean from parsing a value.

Example

```
{{ bool("T") }}
```

Output

```
true
```

Example 2

```
{{ bool(0) }}
```

Output

```
false
```

codepoint

The codepoint function will return the character for the specified Unicode code point.

Example

```
{{ codepoint(34) }}
```

Output

```
"
```

combine lists

The `combineLists` function will return a merged list of the two arguments.

Example

```
{% set listA = [ "a", "b", "c" ] %}
{% set listB = [ "d", "c", "e" ] %}
{{ combineLists(false, listA, listB) }}
{{ combineLists(true, listA, listB) }}
```

Output

```
[a, b, c, d, c, e][a, b, c, d, e]
```

double

The double function will return an int for the provided Number argument or attempt to parse a String into a double.

Example

```
{{ double(3.145) }}
```

Output

```
3.145
```

Example 2

```
{{ double("3.145") }}
```

Output

```
3.145
```

int

The int function will return an int for the provided Number argument or attempt to parse a String into an int.

Example

```
{{ int(3.145) }}
```

Output

```
3
```

Example 2

```
{{ int("3.145") }}
```

Output

```
3
```

replace start

The `replaceStart` function will return a String where **str** will be checked if it starts with **match** and if so be replaced with **replace**.

Example

```
{{ replaceStart("prefix", "pre", "suf", false) }}
```

Output

```
suffix
```

replace end

The `replaceEnd` function will return a String where **str** will be checked if it ends with **match** and if so be replaced with **replace**.

Example

```
{{ replaceEnd("completion", "ion", "ed", false) }}
```

Output

```
completed
```

str is equals

The strIsEquals function will return a boolean for whether the provided String arguments are equal. Safely handles null values.

Example

```
{{ strIsEquals("a", "A", true) }}
```

Output

```
true
```

Example 2

```
{{ strIsEquals("a", null) }}
```

Output

```
false
```

starts with ignore case

The `startsWithIgnoreCase` function will return a boolean for whether the provided **str** begins with **match** while ignoring casing.

Example

```
{{ startsWithIgnoreCase("SuperMan", "super") }}
```

Output

```
true
```

ends with ignore case

The `endsWithIgnoreCase` function will return a boolean for whether the provided **str** ends with **match** while ignoring casing.

Example

```
{{ endsWithIgnoreCase("SuperMan", "man") }}
```

Output

```
true
```

Pebble Template Filters

abbreviate

The abbreviate filter will abbreviate a string using an ellipsis. It takes one argument which is the max width of the desired output including the length of the ellipsis.

Example

```
{{ "This is a sentence" | abbreviate(7) }}
```

Output

```
This...
```

abs

The abs filter is used to obtain the absolute value.

Example

```
{{ -5 | abs }}
```

Output

```
5
```

capitalize

The capitalize filter will capitalize the first letter of the string.

Example

```
{{ "this is a sentence" | capitalize }}
```

Output

```
This is a sentence
```

date

The date filter is used to format an existing `java.util.Date` object. The filter will construct a `java.text.SimpleDateFormat` using the provided pattern and then use this newly `created SimpleDateFormat` to format the provided Date object.

The alternative way to use this filter is to use it on a string but then provide two arguments. The first is the desired pattern for the output and the second is the existing format used to parse the input string into a `java.util.Date` object.

Example : If todayDate is a java.util.Date object

```
{{ todayDate | date("yyyy-MM-dd") }}
```

Output

```
2017-12-25
```

Example 2

```
{{ "December 25, 2017" | date("yyyy-MM-dd", existingFormat="MMMM dd, yyyy") }}
```

Output

```
2017-12-25
```

default

The default filter will render a default value if and only if the object being filtered is empty. A variable is empty if it is null, an empty string, an empty collection, or an empty map.

Example

```
{%- set str = "" -%}
{{ str | default("NonEmpty") }}
```

Output

```
NonEmpty
```

escape

The escape filter will turn special characters into safe character references in order to avoid XSS vulnerabilities.

Example

```
{{ "<content>" | escape }}
```

Output

```
<content>
```

first

The first filter will return the first item of a collection, or the first letter of a string.

Example

```
{{ "Some letters" | first }}
```

Output

```
S
```

Example 2

```
{{ [ "b", "a", "c" ] | first }}
```

Output

```
b
```

join

The join filter will concatenate all items of a collection into a string. An optional argument can be given to be used as the separator between items.

Example

```
{{ [ "This", "is", "a", "sentence" ] | join }}
```

Output

```
This is a sentence
```

Example 2

```
{{ [ "This", "is", "a", "sentence" ] | join(" ") }}
```

Output

```
This is a sentence
```

last

The last filter will return the last item of a collection, or the last letter of a string.

Example

```
{{ "Some letters" | last }}
```

Output

```
s
```

Example 2

```
{{ [ "b", "a", "c" ] | last }}
```

Output

```
c
```

length

The length filter returns the number of items of collection, map or the length of a string.

Example

```
{{ "Some letters" | length }}
```

Output

```
12
```

Example 2

```
{{ [ "b", "a", "c" ] | length }}
```

Output

```
3
```

lower

The lower filter makes an entire string lower case.

Example

```
{{ "THIS IS A SENTENCE" | lower }}
```

Output

```
this is a sentence
```

merge

The merge filter returns a new map, list or array that is the result of merging two maps, lists or arrays together.

Note

Keys in the second map will overwrite the first map keys.

Example

```
{%- set mapa = { "a": "aa", "b": "bb", "c": "cc" } -%}
{%- set mapb = { "c": "ccc", "d": "ddd", "e": "eee" } -%}
{{ mapa | merge(mapb) }}
```

Output

```
{a=aa, b=bb, c=ccc, d=ddd, e=eee}
```

numberformat

The `numberformat` filter is used to format a decimal number. Behind the scenes it uses `java.text.DecimalFormat`.

Example

```
{{ 3.14678 | numberformat("#.##") }}
```

Output

```
3.15
```

raw

The raw filter prevents the output of an expression from being escaped by the autoescaper. The raw filter must be the very last operation performed within the expression otherwise the autoescaper will deem the expression as potentially unsafe and escape it regardless.

Note

If the raw filter is not the last operation performed then the expression will be escaped.

Example

```
{{ "<content>" | raw }}
```

Output

```
<content>
```

replace

The replace filter returns a string that is the result of replacing any substrings matching keys in a map with the respective values.

Example

```
{{ "This is a sentence" | replace({ "i": "eye" }) }}
```

Output

```
Theyes eyes a sentence
```

rsort

The rsort filter will sort a list in reversed order. The items of the list must implement Comparable.

Example

```
{{ [ "b", "a", "c" ] | rsort }}
```

Output

```
[c, b, a]
```

slice

The slice filter returns a portion of a list, array, or string. The indexing starts at 0.

Example

```
{{ "This is a sentence" | slice(5, 9) }}
```

Output

```
is a
```

Example 2

```
{{ [ "b", "a", "c" ] | slice(1, 2) }}
```

Output

```
[a[
```

sort

The sort filter will sort a list. The items of the list must implement Comparable.

Example

```
{{ [ "b", "a", "c" ] | sort }}
```

Output

```
[a, b, c[
```

title

The title filter will capitalize the first letter of each word.

Example

```
{{ "This is a sentence" | title }}
```

Output

```
This Is A Sentence
```

trim

The trim filter is used to trim whitespace off the beginning and end of a string.

Example

```
{{ " Some letters " | trim }}
```

Output

```
Some letters
```

upper

The upper filter makes an entire string upper case.

Example

```
{{ "this is a sentence" | upper }}
```

Output

```
THIS IS A SENTENCE
```

urlencode

The urlencode filter translates a string into `application/x-www-form-urlencoded` format using the `UTF-8` encoding scheme.

Example

```
{{ "list=[a,b,c]" | urlencode }}
```

Output

```
list%3D%5Ba%2Cb%2Cc%5D
```

lines

The lines filter will split a string by its line breaks with each line becoming an element in a list.

Example

```
{%- set aStr = "Something
Is
On
Lots
Of
Lines" -%}
{{ aStr | lines }}
```

Output

```
[Something, Is, On, Lots, Of, Lines]
```

Pebble Template Tests

defined

The defined test checks if a variable exists.

Example

```
{%if someVar is defined%}   ...
{%endif %}
```

empty

The empty test checks if a variable is empty. A variable is empty if it is null, an empty string, an empty collection, or an empty map.

Example

```
{% if "" is empty %}
...
```

```
{% endif  %}
```

even

The even test checks if an integer is even.

Example

```
{% if 2 is even %}
...
{% endif  %}
```

iterable

The iterable test checks if a variable implements `java.lang.Iterable`.

Example

```
{% if [ "a", "b", "c" ] is iterable %}
...
{% endif  %}
```

map

The map test checks if a variable is an instance of a map.

Example

```
{% if { "a": "aa", "b": "bb" } is map %}
...
{% endif  %}
```

null

The null test checks if a variable is null.

Example

```
{% if someVar is null %}
...
{% endif  %}
```

odd

The odd test checks if an integer is odd.

Example

```
{% if "" is odd %}
...
{% endif  %}
```

boolean

The boolean test checks if the input is a Boolean or viable to be casted as a Boolean.

Example

```
{% if "yes" is boolean %}   ...
{% endif  %}
```

Example 2

```
{% if 1 is boolean %}   ...
{% endif  %}
```

instanceof

The instanceof test checks if the input is of the specified class.

Example

```
{% if "a" is instanceof ("String") %}   ...
{% endif  %}
```

Example 2

```
{% if "a" is instanceof ("java.lang.String") %}   ...
{% endif  %}
```

number

The number test checks if the input is a Number or viable to be casted as a Number. The input can be either a numeric value or a String.

Example

```
{% if3.14 is number %}   ...
{% endif  %}
```

Example 2

```
{% if "3.14"  is number %}   ...
{% endif  %}
```

string

The string test checks if the input is a String.

Example

```
{% if "a" is string %}...
{% endif  %}
```

Pebble Template Variables

StringUtil

Provides common string function support.

Function	Description	Datatype	

capitalize(String str)	Capitalizes a String, Changing the first character to title case.	String	**Example**
			`{{ StringUtils.capitalize("word") }}`
			Output
			`Word`

compare(String str1, String str2)	Compare two Strings lexicographically.
	- Returns -1 when str1 is less than str2.
	- Returns 0 when str1 and str2 are equal.
	- Returns 1 when str1 is greater than str2.
		int	**Example**
			`{% if (StringUtils.compare("lower", "upper") > 0) %}`
			...
			`{% endif %}`

compare(String str1, String str2, boolean nullIsLess)	Compare two Strings lexicographically.
	- Returns -1 when str1 is less than str2.
	- Returns 0 when str1 and str2 are equal.
	- Returns 1 when str1 is greater than str2.
		int	**Example**
			`{% if (StringUtils.compare("lower", "upper", false) > 0) %}`
			...
			`{% endif %}`

compareIgnoreCase(String str1, String str2)	Compare two Strings lexicographically, ignoring case differences.
	- Returns -1 when str1 is less than str2.
	- Returns 0 when str1 and str2 are equal.
	- Returns 1 when str1 is greater than str2.
		int	**Example**
			`{% if (StringUtils.compareIgnoreCase("lower", "upper") > 0) %}`
			...
			`{% endif %}`

compareIgnoreCase(String str1, String str2, boolean nullIsLess)	Compare two Strings lexicographically, ignoring case differences. - Returns -1 when str1 is less than str2. - Returns 0 when str1 and str2 are equal.	int	**Example** `{% if (StringUtils.compareIgnoreCase("lower", "upper", false) > 0) %}`

Function	Description	Datatype	
	- Returns 1 when str1 is greater than str2.		... `{% endif %}`
countMatches(String str, String pattern)	Returns the number of times the pattern appears in the string	int	**Example** `{{ StringUtils.countMatches( "AaaaaaA", "a") }}` **Output** 5
indexOf(String str, String search)	Returns the index (Starts with zero) within the String of the first occurrence of the specified search String.	int	**Example** `{{ StringUtils.indexOf("abcabcabc", "c") }}` **Output** 2
indexOf(String str, String search, int startPos)	Returns the index (Starts with zero) within the String of the first occurrence of the specified search String starting from the provided zero based index	int	**Example** `{{ StringUtils.indexOf("abcabcabc", "c", int(4)) }}` **Output** 5
indexOfIgnoreCase(String str, String search)	Returns the case insensitive index (Starts with zero) within the String of the first occurrence of the specified search String.	int	**Example** `{{ StringUtils.indexOfIgnoreCase("ABCABCABC", "c") }}` **Output** 2
indexOfIgnoreCase(String str, String search, int startPos)	Returns the case insensitive index (Starts with zero) within the String of the first occurrence of the specified search String starting from the provided zero based index.	int	**Example** `{{ StringUtils.indexOfIgnoreCase("ABCABCABC", "c", int(4)) }}` 5

isAllLowerCase(String str)	Checks if the String contains only lowercase characters.	boolean	**Example**
			`{{ StringUtils.isAllLowerCase("letters") }} {% br %}`
			`{{ StringUtils.isAllLowerCas`

Function	Description	Datatype	
			e("leTTers") }} **Output** true false
isAllUpperCase(String str)	Checks if the String contains only uppercase characters.	boolean	**Example** {{ StringUtils.isAllUpperCase("LETTERS") }} {% br %} {{ StringUtils.isAllUpperCase("leTTers") }} **Output** true false
isAlpha(String str)	Checks if the String contains only unicode letters.	boolean	**Example** {{ StringUtils.isAlpha("letters") }}{% br %} {{ StringUtils.isAlpha("Num1") }} **Output** true false
isAlphanumeric(String str)	Checks if the String contains only unicode letters or digits.	boolean	**Example** {{ StringUtils.isAlphanumeric("Number3") }}{% br %} {{ StringUtils.isAlphanumeric("Number 3") }} **Output** true false
isAlphanumericSpace(String str)	Checks if the String contains only unicode letters, digits or space.	boolean	**Example** {{ StringUtils.isAlphanumericSpace("Number 3") }}

Function	Description	Datatype	
			Output true
isAlphaSpace(String str)	Checks if the string contains only unicode letters or space.	boolean	**Example** {{ StringUtils.isAlphaSpace("next letter") }}{% br %} {{ StringUtils.isAlphaSpace("Number 3") }} **Output** true false
isNumeric(String str)	Checks if the String contains only unicode digits.	boolean	**Example** {{ StringUtils.isNumeric("123") }}{% br %} {{ StringUtils.isNumeric("Number2") }} **Output** true false
isNumericSpace(String str)	Checks if the String contains only unicode digits or space.	boolean	**Example** {{ StringUtils.isNumericSpace("1 2 3") }}{% br %} {{ StringUtils.isNumericSpace("Johnny 5") }} **Output** true false

isWhitespace(String str)	Checks if the String contains only whitespace.	boolean	**Example**
			{{ StringUtils.isWhitespace(" ") }}{% br %}
			{{ StringUtils.isWhitespace(" letter ") }}

Function	Description	Datatype	

			Output
			`true`
			`false`

lastIndexOf(String str, String search)	Returns the index (Starts with zero) within the String of the last occurrence of the specified search String.		**Example**
			`{{ StringUtils.lastIndexOf("abcabcabc", "a") }}`
			Output
			`6`

lastIndexOf(String str, String search, int startPos)	Returns the index (Starts with zero) within the String of the last occurrence of the specified search String starting from the provided zero based index.		**Example**
			`{{ StringUtils.lastIndexOf("abcabcabc", "a", int(2)) }}`
			Output
			`0`

lastIndexOfIgnoreCase(String str, String search)	Returns the case insensitive index (Starts with zero) within the String of the last occurrence of the specified search String.	int	**Example**
			`{{ StringUtils.lastIndexOfIgnoreCase("ABCABCABC", "a") }}`
			Output
			`6`

lastIndexOf(String str, String search, int startPos)	Returns the case insensitive index (Starts with zero) within the String of the last occurrence of the specified search String starting from the provided zero based index.	int	**Example**
			`{{ StringUtils.lastIndexOfIgnoreCase("ABCABCABC", "a", int(2)) }}`
			Output
			`0`

leftPad(String str, int size)	Left pad a String with spaces.	String	**Example**
			`T{{ StringUtils.leftPad("1", int(5) }}T`
			Output
			`T 1T`

repeat(String str, int repeat)	Repeat a String repeat times to form a new String.	String.	**Example** `{{ StringUtils.repeat("haha", int(5)) }}`

Function	Description	Datatype	
			Output hahahahahahahahaha
repeat(String str, String separator, int repeat)	Repeat a String repeat times to form a new String, with a String separator injected each time.		**Example** {{ StringUtils.repeat("haha", " ", int(5)) }} **Output** haha haha haha haha haha
reverse(String str)	Reverses a String	String	**Example** {{ StringUtils.reverse("abcde") }} **Output** edcba
rightPad(String str, int size)	Right pad a String with spaces.	String	**Example** T{{ StringUtils.rightPad("1", int(5) }}T **Output** T1 T
splitByCharacterType(String str)	Splits a String into an array of String with each element containing a substring based on the character type. Character types include: lower case, upper case, numbers, whitespace, math operators	String[]	**Example** {% set array = StringUtils.splitByCharacterType(" AbcdEFG%++-/3432 ") %} Split content:{%- br -%} {%- for item in array -%} {{ item }}{%- br -%} {%- endfor -%} EndSplit **Output** Split content: A bcd EFG % ++ - / 3432

Function	Description	Datatype	
			EndSplit
splitByCharacterTypeCamelCase(String str)	Splits a String into an array of String with each element containing a substring based on the character type with the special case to maintain one upper case letter followed by some lower case letters as an element. Character types include: lower case, upper case, numbers, whitespace, math operators	String[]	**Example** {% set array = StringUtils.splitByCharacterTypeCamelCase(" AbcdEFGhij%++-/3432 ") %} Split content:{%- br -%} {%- for item in array -%} {{ item }}{%- br -%} {%- endfor -%} EndSplit **Output** Split content: Abcd EF Ghij % ++ - / 3432 EndSplit
swapCase(String str)	Swaps the case of a String changing upper and title case to lower case, and lower case to upper case.	String	**Example** {{ StringUtils.swapCase("UPPER lower") }} **Output** upper LOWER
trim(String str)	Removes whitespace from both ends of the String.	String	**Example** T{{ StringUtils.trim(" island ") }}T **Output** TislandT
uncapitalize(String str)	Uncapitalizes a String, changing the first character to lower case .	String	**Example** {{ StringUtils.uncapitalize("WORD") }} **Output**

Function	Description	Datatype	
			`wORD`

Template Usage

If a template exists of the correct **Type** and **Target DB**, templates can be specified and evaluated as follows:

Operation	Location to Specify the Template

Operation	Location to Specify the Template
Alter Table DDL	**Connection Properties > Target Settings > Default Table Alter DDL Template**
Create DDL	**Table Properties > Storage > Create DDL Template** **Connection Properties > Target Settings > Default Table Create DDL Template** **Connection Properties > Target Settings > Default View Create DDL Template**
Create Index DDL	**Connection Properties > Target Settings > Default Index Create DDL Template**
Drop Index DDL	**Connection Properties > Target Settings > Default Index Drop DDL Template**
Export Script	**Export Object Properties > Script Template**
Load Script	**Load Table Properties > Script Template** **Connection Properties > New Table Default Load Script Template**
Update Procedure	When building the Update Procedure, specify a template to use in the **Template** drop-down field of the **Update Build Option > Processing tab**.
Custom Update Procedure	**Table Properties > Custom Procedure**
Update Script	**Table Properties > Update Script**
Post Load Procedure	**Load Table Properties > Post Load Procedure**
Post Load Script	**Load Table Properties > Post Load Script**

To check which Operations are supported by Templates on your Target database, refer to **Templates** for details.

Notes:

- The template of the right type and database controls the availability of the template options above, e.g. the template selection fields are only available if valid templates exists.
- Most of the default settings for Templates are configurable in the **Connection Properties > Target Setting** tab. Refer to **Connection Target Settings** for details.
- The template object context menu that is displayed when you right click a Template object from the Objects list on the left pane provides a Usage Report option, which enables you to list the table objects that are currently associated with the selected template. Refer to **Working with Objects > Templates** for details.
- Additional host script language types which can be applied to templates can be defined in RED. Refer to **Host Script Languages** for details.

Script Templates for Custom Database Table Objects

The update process for data warehouse table objects stored in a custom database target can be generated via a SQL block or a Windows PowerShell script. In both cases, templates are utilized to build the update procedures to populate the tables.

WhereScape RED provides PowerShell script templates that are designed to work for Table objects with a custom database target in a SQL Server metadata repository. Refer to the custom database enablement packs provided by WhereScape for details about these PowerShell script templates.

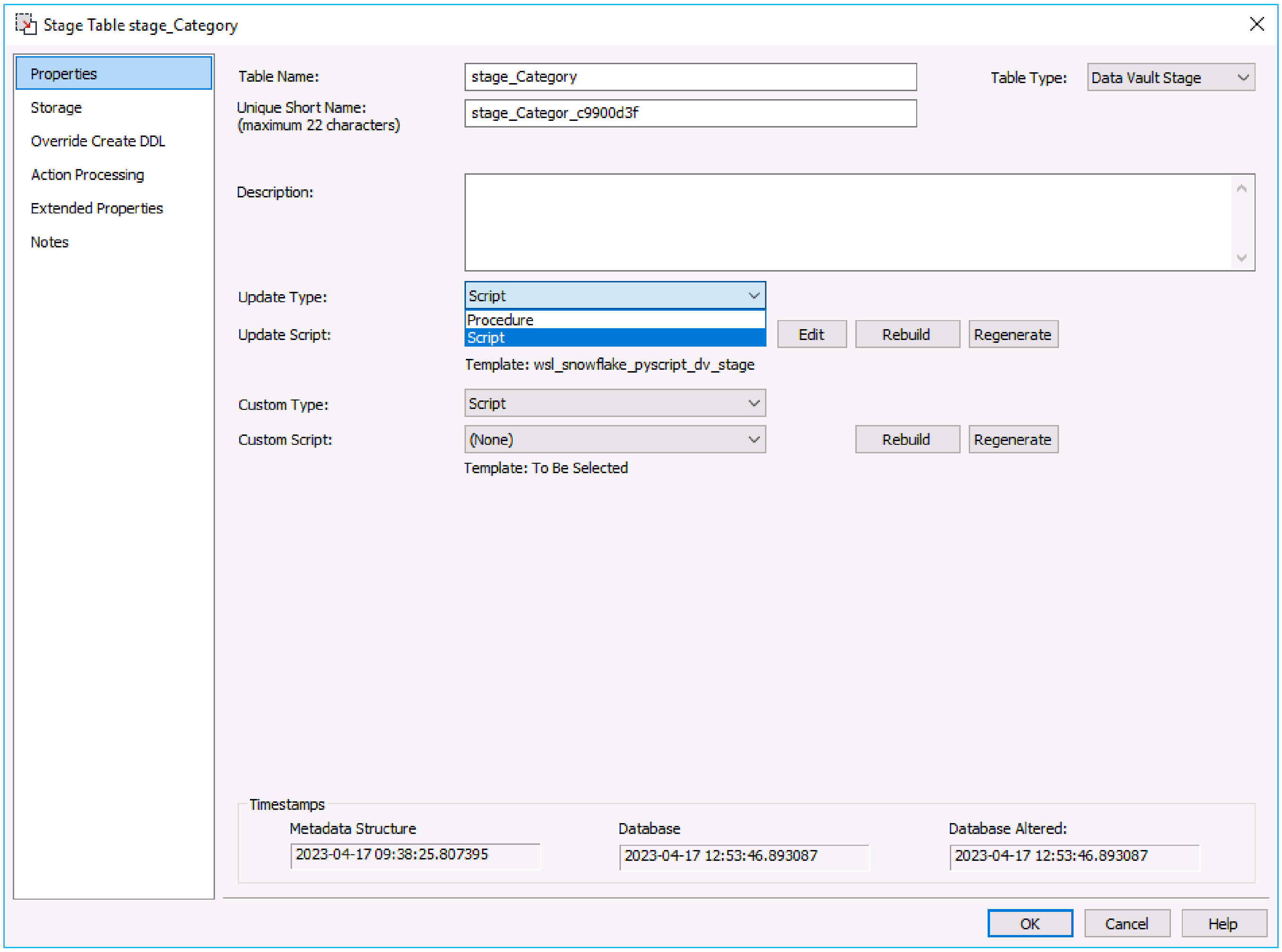
Moreover, additional host script language types that can be applied to Templates can be defined in RED via the

Tools > Host Script Languages tool. User defined script templates can also be used for building the update procedures of Table objects stored in a custom database target.

The **Properties** window for table objects that support script based update has an **Update Type** drop-down field, which enables you to select between **Procedure** or **Script** based type processing for the table's update procedures.

Note:

For new data warehouse table objects in a custom database target, the default **Update Type** is set to **Script**.



Stage Table stage_Category

Table Name: stage_Category

Table Type: Data Vault Stage

Unique Short Name: (maximum 22 characters) stage_Categor_c9900d3f

Description:

Update Type: Script

Update Script:

Template: wsl_snowflake_pyscript_dv_stage

Custom Type: Script

Custom Script: (None)

Template: To Be Selected

Timestamps

Metadata Structure	Database	Database Altered:
2023-04-17 09:38:25.807395	2023-04-17 12:53:46.893087	2023-04-17 12:53:46.893087

Selecting **Script** from the **Update Type** drop-down displays the **Update Script** drop-down which enables you to select a script to use from the list of available script templates. The same applies to the **Custom Type** drop-down field.

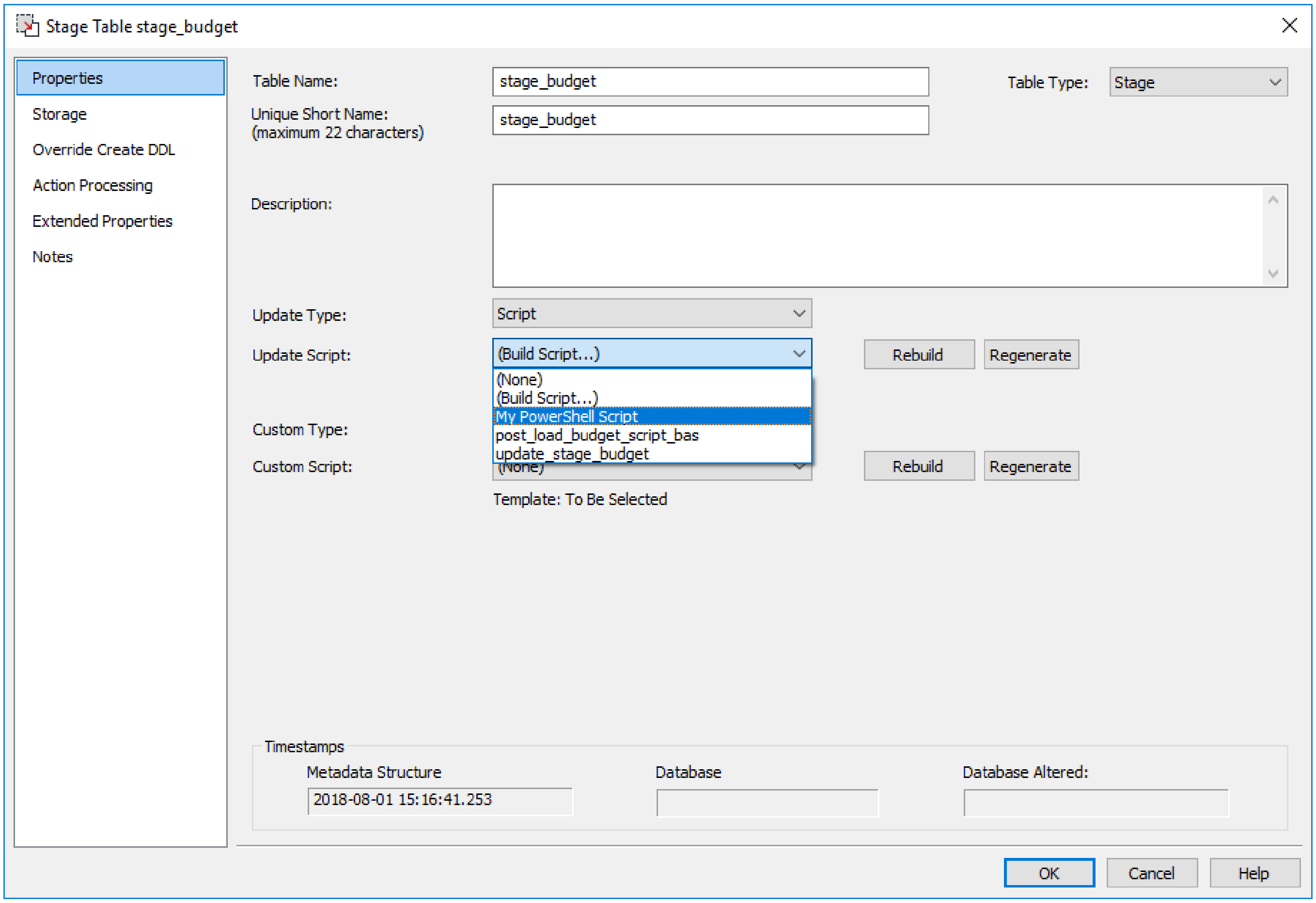


Table Name: stage_budget

Unique Short Name: (maximum 22 characters) stage_budget

Table Type: Stage

Description:

Update Type: Script

Update Script: (Build Script...)

(None)
(Build Script...)
My PowerShell Script
post_load_budget_script_bas
update_stage_budget

Custom Type:

Custom Script: (None)

Template: To Be Selected

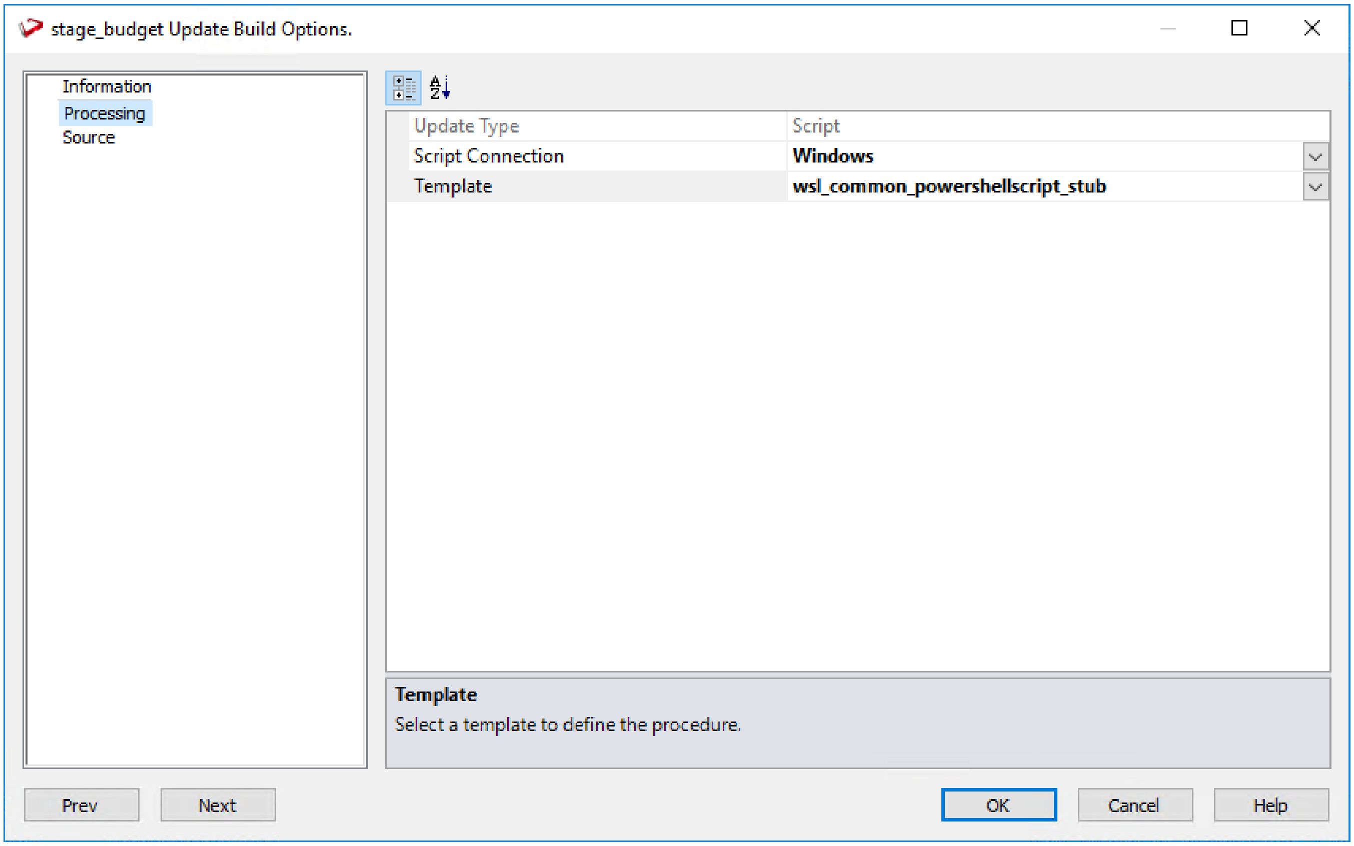
Timestamps

Metadata Structure: 2018-08-01 15:16:41.253

Database:

Database Altered:

Selecting the **(Build Script...)** option and then clicking the **Rebuild** or **OK** button opens the table **Update Build Options** window which enables you to define the **Script Connection** and **Template** to use. The same applies to the **Custom Script** processing.



Update Type	Script
Script Connection	Windows
Template	wsl_common_powershellscript_stub

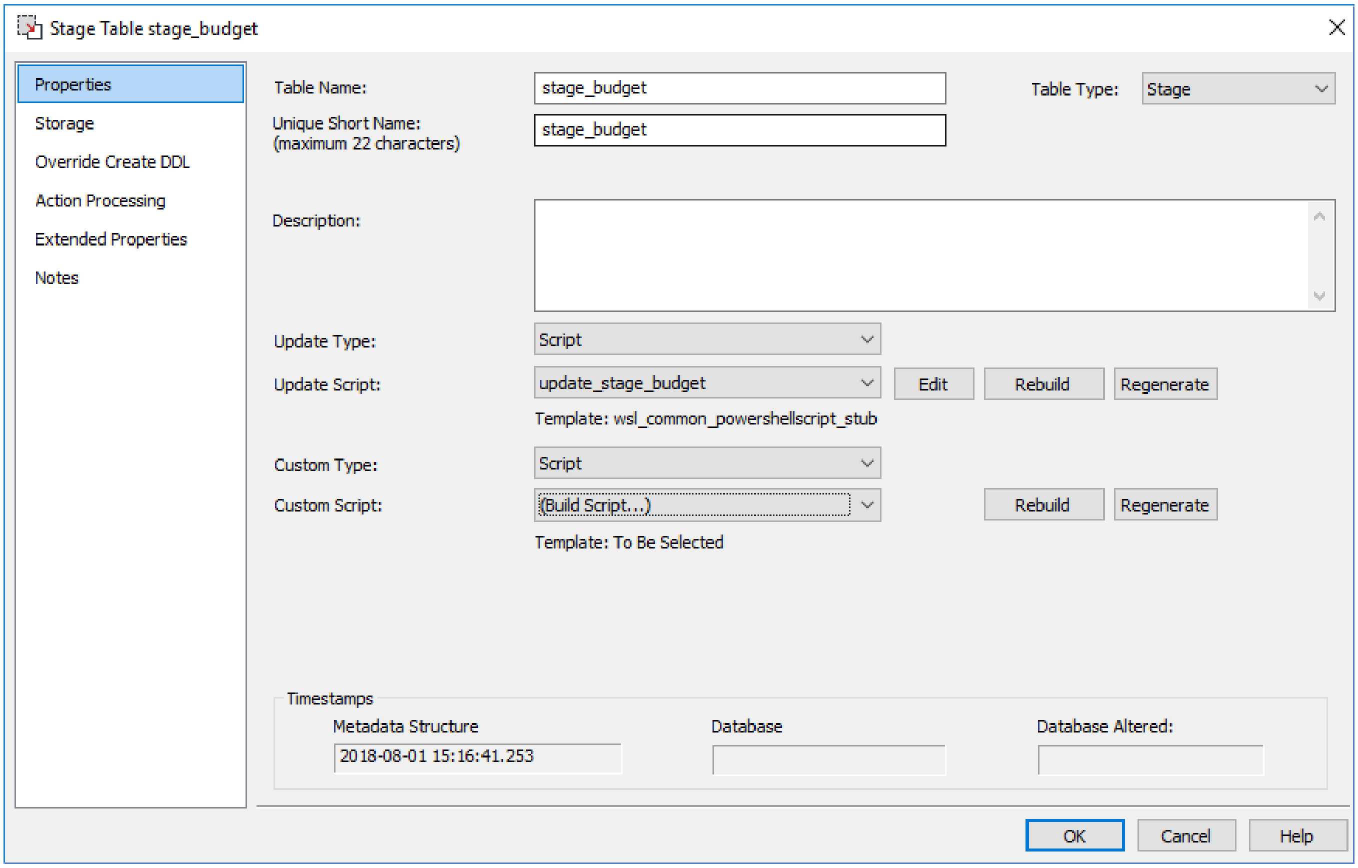
Template

Select a template to define the procedure.

- Clicking the **Edit** button opens the selected update script template in the WhereScape RED **Script Editor** which enables you to edit the contents as required.
- Clicking the **Regenerate** button regenerates the selected update script template without any prompts, unless you made changes that require further input.
  The same applies to the **Custom Script** processing.

Note:

If a script (e.g. update script, custom script, load script, export script, etc.) is modified, the update time for that script is reflected in the **Script Editor** and the ### MODIFIED ### notice is displayed in the table object's Properties window when an update script is selected.



Stage Table stage_budget

Table Name: stage_budget

Unique Short Name: (maximum 22 characters) stage_budget

Table Type: Stage

Description:

Update Type: Script

Update Script: update_stage_budget

Template: wsl_common_powershellscript_stub

Custom Type: Script

Custom Script: (Build Script...)

Template: To Be Selected

Timestamps

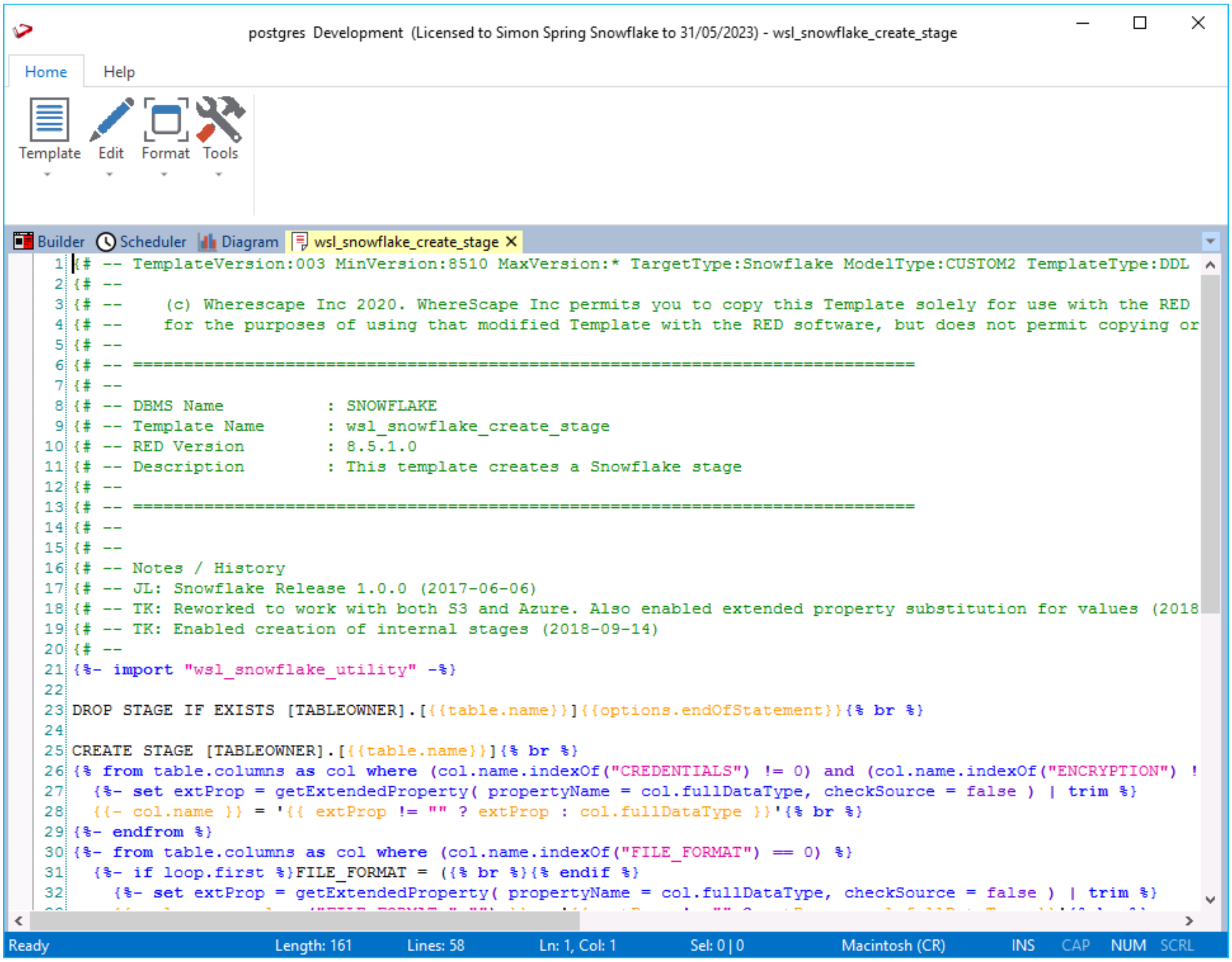
Metadata Structure	Database	Database Altered:
2018-08-01 15:16:41.253		

Windows PowerShell Templates

PowerShell Stub Template (wsl_common_powershellscript_stub)

You can use the basic PowerShell stub template available in WhereScape RED that serves as a guide in using a template to generate a PowerShell script.

Additional PowerShell Templates can be downloaded from the WhereScape website (**https://www1.wherescape.com/support/software-downloads-documentation/templates/**).


postgres Development (Licensed to Simon Spring Snowflake to 31/05/2023) - wsl_snowflake_create_stage
Home
Help
Template
Edit
Format
Tools
Builder
Scheduler
Diagram
wsl_snowflake_create_stage

```
{# -- TemplateVersion:003 MinVersion:8510 MaxVersion:* TargetType:Snowflake ModelType:CUSTOM2 TemplateType:DDL
{# --
{# --    (c) Wherescape Inc 2020. WhereScape Inc permits you to copy this Template solely for use with the RED
{# --    for the purposes of using that modified Template with the RED software, but does not permit copying or
{# --
{# -- =============================================================================
{# --
{# -- DBMS Name          : SNOWFLAKE
{# -- Template Name      : wsl_snowflake_create_stage
{# -- RED Version        : 8.5.1.0
{# -- Description        : This template creates a Snowflake stage
{# --
{# -- =============================================================================
{# --
{# --
{# -- Notes / History
{# -- JL: Snowflake Release 1.0.0 (2017-06-06)
{# -- TK: Reworked to work with both S3 and Azure. Also enabled extended property substitution for values (2018
{# -- TK: Enabled creation of internal stages (2018-09-14)
{# --
{%- import "wsl_snowflake_utility" -%}

DROP STAGE IF EXISTS [TABLEOWNER].[{{table.name}}]{{options.endOfStatement}}{% br %}

CREATE STAGE [TABLEOWNER].[{{table.name}}]{% br %}
{% from table.columns as col where (col.name.indexOf("CREDENTIALS") != 0) and (col.name.indexOf("ENCRYPTION") !
  {%- set extProp = getExtendedProperty( propertyName = col.fullDataType, checkSource = false ) | trim %}
  {{- col.name }} = '{{ extProp != "" ? extProp : col.fullDataType }}'{% br %}
{%- endfrom %}
{%- from table.columns as col where (col.name.indexOf("FILE_FORMAT") == 0) %}
  {%- if loop.first %}FILE_FORMAT = ({% br %}{% endif %}
    {%- set extProp = getExtendedProperty( propertyName = col.fullDataType, checkSource = false ) | trim %}
```

Ready Length: 161 Lines: 58 Ln: 1, Col: 1 Sel: 0 | 0 Macintosh (CR) INS CAP NUM SCRL

Notes:

- The PowerShell stub template (wsl_common_powershellscript_stub) is included under the **Template** objects list pane, along with the other templates available in WhereScape RED. If this stub template is not visible after installing/upgrading WhereScape RED, use the WhereScape RED Setup Administrator to Validate the Metadata Repository. For more information, please refer to the WhereScape RED Installation Guide.
- WhereScape provides a Windows application extension called WslMetadataServiceClient that can be used from PowerShell templates to simplify access to RED metadata content. This DDL is accessed from WSL provided templates in custom database enablement packs.
  Please contact your WhereScape customer representative, if you require details on how to use this DDL.

Scheduler

The scheduler is accessible by clicking the **Scheduler** button Scheduler from the toolbar. It is also available as a stand alone utility. In this way, operators can be given access to the scheduler without gaining full access to the data warehouse.

The scheduler runs on either a UNIX host or under Windows (as a system service). It processes predefined jobs, recording the success, failure or otherwise of the job.

Audit trail

Specific information relating to the tasks in the job are recorded to the audit trail. Generally, only summary information is written to this audit trail. The contents of the audit trail are maintained even after a job is deleted.

Error trail

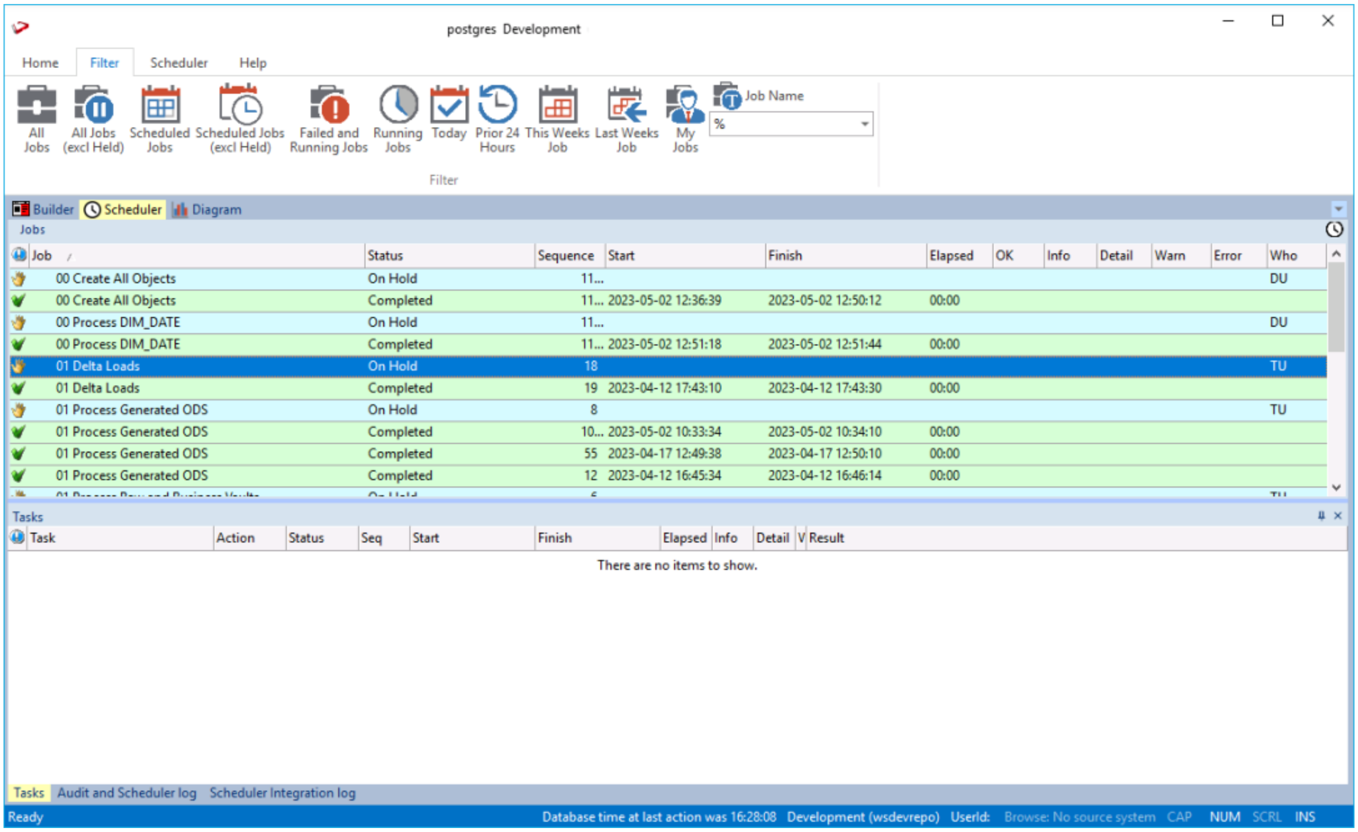
Detail or error information is written to the error trail. The contents of the error trail are deleted when the job is deleted.

Administration Views

It is possible to view the status of a job without using the WhereScape RED product. Three views are provided to assist in this undertaking. They are **ws_admin_v_audit**, **ws_admin_v_error** and **ws_admin_v_sched**. Queries can be issued using these views to see the results or status of a job.

Scheduler Window

An example of the **Scheduler** screen is shown below.



Job Name: %

Job	Status	Sequence	Start	Finish	Elapsed	OK	Info	Detail	Warn	Error	Who
00 Create All Objects	On Hold	11...									DU
00 Create All Objects	Completed	11...	2023-05-02 12:36:39	2023-05-02 12:50:12	00:00						
00 Process DIM_DATE	On Hold	11...									DU
00 Process DIM_DATE	Completed	11...	2023-05-02 12:51:18	2023-05-02 12:51:44	00:00						
01 Delta Loads	On Hold	18									TU
01 Delta Loads	Completed	19	2023-04-12 17:43:10	2023-04-12 17:43:30	00:00						
01 Process Generated ODS	On Hold	8									TU
01 Process Generated ODS	Completed	10...	2023-05-02 10:33:34	2023-05-02 10:34:10	00:00						
01 Process Generated ODS	Completed	55	2023-04-17 12:49:38	2023-04-17 12:50:10	00:00						
01 Process Generated ODS	Completed	12	2023-04-12 16:45:34	2023-04-12 16:46:14	00:00						

Task	Action	Status	Seq	Start	Finish	Elapsed	Info	Detail	V	Result

There are no items to show.

Ready Database time at last action was 16:28:08 Development (wsdevrepo) UserId: Browse: No source system CAP NUM SCRL INS

Toolbar/Jobs menu

Quick access to some job categories are in the toolbar. The complete options are listed under the **Jobs** menu and while most are self-explanatory they are described below:

Object	Description
Job Name Filter	Lists the jobs whose names match the filter supplied. This filter only works as an appended filter to the main filter selected under **Jobs**, e.g. first enter a filter for **Job Name Filter** or select a filter from the drop-down list; and then choose your main filter under **Jobs**—e.g. All Jobs, Scheduled Jobs, etc.
All jobs	Lists all jobs in the middle pane
Scheduled	Lists the jobs that are waiting to run or are on hold.
Run/Fail	Lists the jobs that are running and those that have failed.
My Jobs	Lists the jobs you have scheduled or have run.
Today	Lists the jobs you have scheduled for today.
Last 24 hours	Lists the jobs that have run or started to run during the last 24 hours.
Prior 24 hours	Lists the jobs that ran during the previous 24 hours.
This Weeks Jobs	Lists the jobs that have run or are scheduled to run during the current week
Last Weeks Jobs	Lists the jobs that ran during the last week.
Recent log entries	Displays the Audit and Scheduler log which details the audit trail information of a task/job.
Scheduler status	Lists all schedulers and displays their current status. The status is updated every few minutes, and a right menu option enables the polling of a scheduler for status, and the termination of a scheduler.

Top pane

The top pane shows the details of the jobs. Information covers:

Column	Description
Job name	The name given to the job when created.
Status	The status of the job. Refer to the following section for the various status values.
Sequence	This is a unique number assigned to each job iteration and job. If you enter a new job, it acquires a new sequence number. In normal daily processing when no new jobs have been created, sequence numbers will be sequential.
Start and Finish times	The start and finish dates and times for the job.
Elapsed time	The time that elapses from start to finish of a job.
Ok	These are success messages written to the audit trail.
Info	Informational messages written to the audit trail about the running of the job.
Detail	Lines written to the detail or error logs.
Warn	The number of warnings written to the audit trail.
Error	The number of error messages written to the audit trail.
Who	The initials of the person who scheduled the job.

Additional fields can be added via the **Tools > Select Job Report Fields** option.

Middle pane

The middle pane shows the tasks related to a selected job. Task information available includes:

Column	Description
Task	The object name.
Action	The action to be done to the object.
Status	Status of the task.

Column	Description
Sequence	This is a unique number assigned to each job iteration and job. If you enter a new job, it acquires a new sequence number. In normal daily processing when no new jobs have been created, sequence numbers will be sequential.
Start and Finish times	As the names suggest, the start and finish dates and times for the task
Elapsed time	The time that elapses from start to finish of a task.
Info	Informational messages written to the audit trail about the running of the job.
Detail	Lines written to the detail or error logs.
Warning	The number of warnings written to the audit trail.
Result	Result of the task.

Additional fields can be added via the **Tools > Select Task Report Fields** option

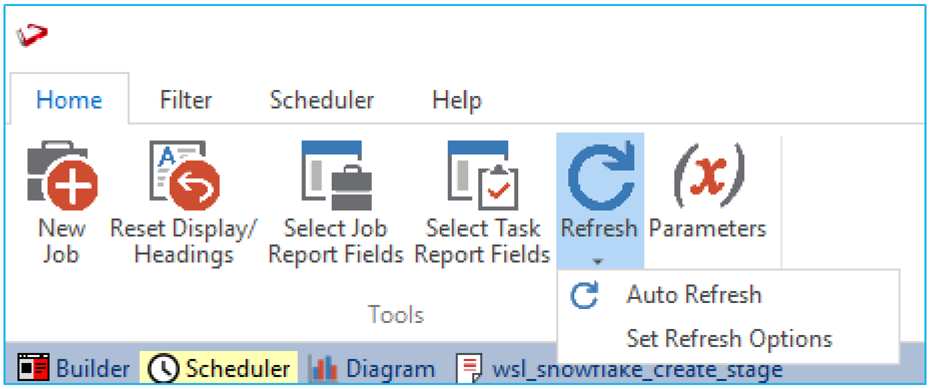
Bottom pane

The bottom pane shows the audit trail of a selected task/job. Audit trail information includes:

Column	Description
Task	The object name.
Status	The status of the message.
Sequence	This is a unique number assigned to each job iteration and job. If you enter a new job, it acquires a new sequence number. In normal daily processing when no new jobs have been created, sequence numbers will be sequential.
Timestamp	Time of message output.
Message	The message.
DB Message	Message from database.
Job	Job relating to this message.

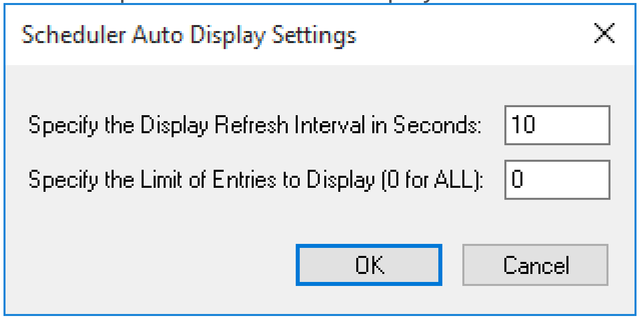
Auto Menu

The **Auto** menu provides options for refreshing the scheduler display.



The following options are available:

Option	Description
Auto Refresh	Use auto refresh to automatically refresh the display status of all jobs.
Set Refresh	Displays the Scheduler Auto Display Settings window which enables you to specify auto

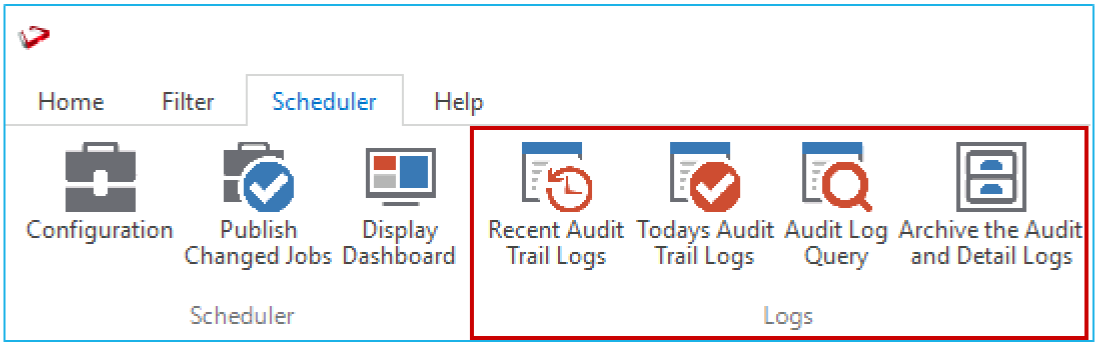
Option	Description
Options	display settings. Use this setting to control the maximum number of rows that are displayed via the Auto Refresh option as well as the display refresh interval.  Scheduler Auto Display Settings Specify the Display Refresh Interval in Seconds: 10 Specify the Limit of Entries to Display (0 for ALL): 0 When clicking the **Set Refresh Options**, the settings dialog allows adding a display limit. The jobs displayed when on auto refresh will stop at the selected number of rows, so if the limit is set to 100, the refresh will stop after the first 100 rows (jobs) are returned. If **0** is selected in the Specify the Limit of Entries to Display, then all rows (jobs) is displayed.

Logs Menu

The **Logs** menu provides options for viewing the Audit Trail logs and enables you to archive logs when required.

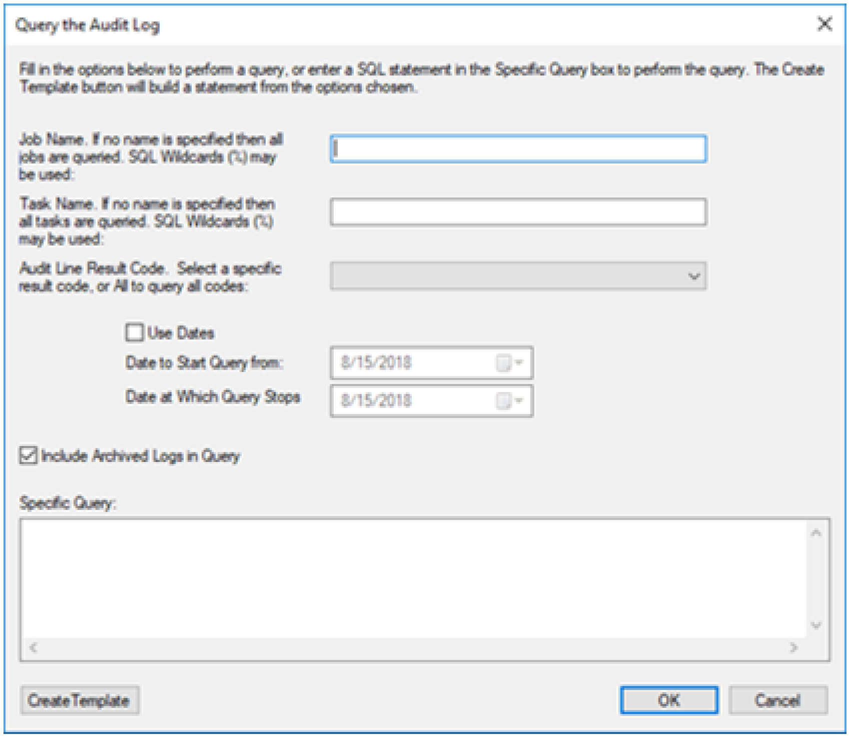
The logs information is useful when a job fails to start or enters some other unknown state. Generally, the audit trail entries for a job can be found by drilling down into the job itself.

The Audit Trail log retains all the task information for a job, even if its associated job log is deleted. This log is backed up, as part of the metadata and should be periodically archived.



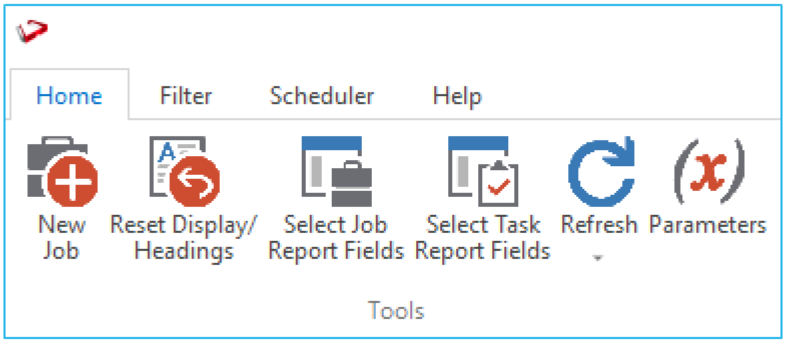
The following options are available:

Option	Description
Recent Audit Trail Logs	Displays the Audit and Scheduler log which details the most recent audit trail information of a task/job.
Todays Audit Trail Logs	Displays the logs for the jobs that have run in the current day.
Audit Log Query	Displays the **Query the Audit Log** window which enables you to specify the parameters to perform the log query.

Option	Description
	 Query the Audit Log Fill in the options below to perform a query, or enter a SQL statement in the Specific Query box to perform the query. The Create Template button will build a statement from the options chosen. Job Name. If no name is specified then all jobs are queried. SQL Wildcards (%) may be used: Task Name. If no name is specified then all tasks are queried. SQL Wildcards (%) may be used: Audit Line Result Code. Select a specific result code, or All to query all codes: ☐ Use Dates Date to Start Query from: 8/15/2018 Date at Which Query Stops 8/15/2018 ☑ Include Archived Logs in Query Specific Query: Create Template OK Cancel
Archive the Audit and Detail Logs	Archives the Audit and Detail logs to an audit trail archive table. The archive process moves the records from the logs to an audit trail archive table. This table can be queried but is not backed up, as part of the metadata backup routines.

Tools Menu

The **Tools** menu provides options for configuring the scheduler display.



The following options are available:

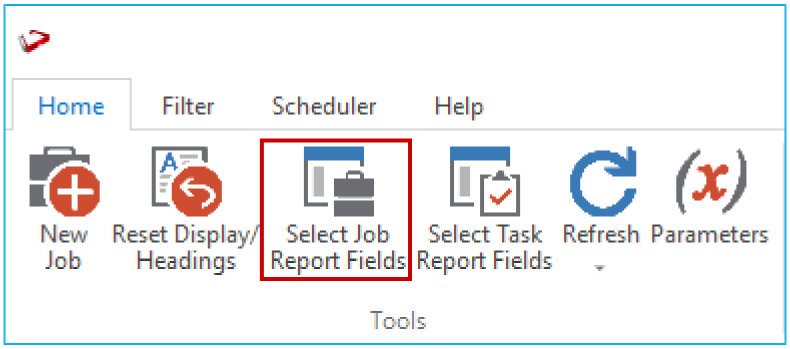
Option	Description
Parameters	Click to display all parameters defined in the meta data repository.
Select Job Report Fields	Click to choose the fields to include in the job report (see Select Job Report Fields below).
Select Task Report Fields	Click to choose the fields to include in the task report.

Option	Description
Background Line Color	Click to display/hide the background line colors in the Jobs and Tasks panes.

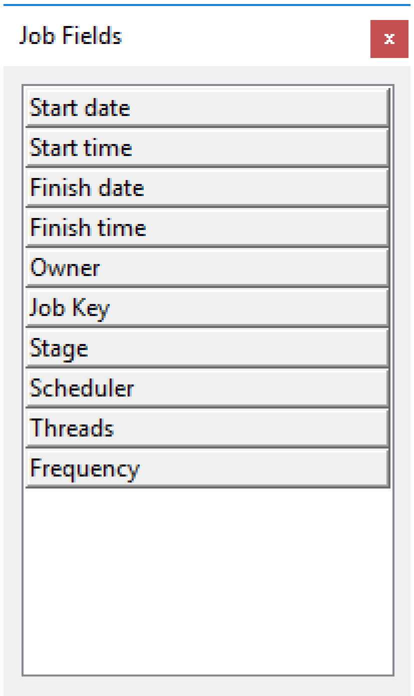
Select Job Report Fields

The **Select Job Report Fields** menu option in the Scheduler window enables you to select extra fields (e.g. Scheduler, Threads and Frequency) to include in the job report. such as fields into the job report.

1. To make these fields available, click **Tools > Select job report fields** from the top pane.



2. Select the fields you want to add to your job report from the menu and drag them to where you want to place them on the report.



Note

The "Frequency" field is only populated for scheduled jobs. If selected, it returns blank results for running or completed jobs.

Scheduler States

A scheduled **job** can have the following states:

- Hold
- Waiting
- Blocked
- Pending
- Running
- Failed
- Failed - Aborted
- Completed

State	Description
Hold	The job is on hold. It can be edited and its state changed in order to release the job.
Waiting	The job is waiting to start, or waiting for its scheduled time to arrive, or waiting for a scheduler to become available.
Blocked	The job is blocked as a previous instance of the same job is still running. When the currently running instance completes this job will start.
Pending	This is the first stage of a running job. The scheduler has identified the job as ready to start and has allocated a thread, or sub task to process the job. A job is in this state until the thread or sub task begins processing. If a job stays in this state then the scheduler thread has failed for some reason. The logs can be checked on either the UNIX or Windows server on which the scheduler is running.
Running	The job is currently running. Double-click on the job name in the right pane to drill down into the specific tasks.
Failed	A failed job is one that had a problem. It can be restarted from the point of failure and is considered to be running unless subsequently aborted.
Failed - Aborted	The job has been aborted after having failed. Once in this state a job cannot be restarted. The job exists then only as a log of what occurred and is no longer regarded as a job.
Completed	The job has successfully completed, possibly with warnings. Once in this state a job cannot be restarted. The job exists then only as a log of what occurred and is no longer regarded as a job.

Note

When a job fails and drilling down does not show any errors against the tasks, right-click on the job and "**View Audit Trail**". The job may have failed because of an error in the Job level.

A scheduled **task** can have the following states:

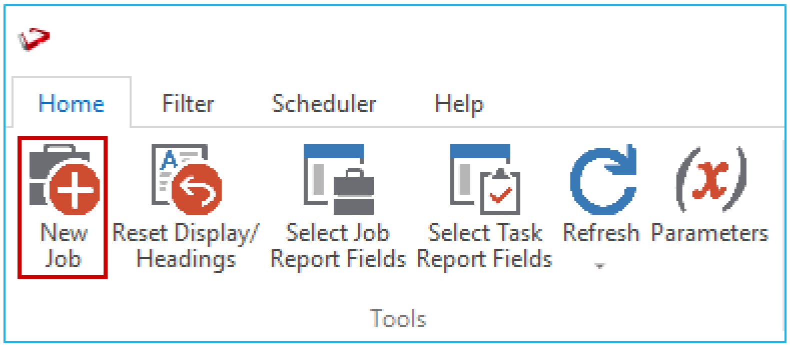
- Waiting or Blank
- Held
- Running
- Failed
- Completed
- Error Completion
- Bad Return Status

State	Description
Held	The task has been held due to a prior dependency failure. The problem must be rectified and the job restarted.
Waiting (Blank)	Tasks that are waiting to run either due to a shortage of threads, or prior dependencies

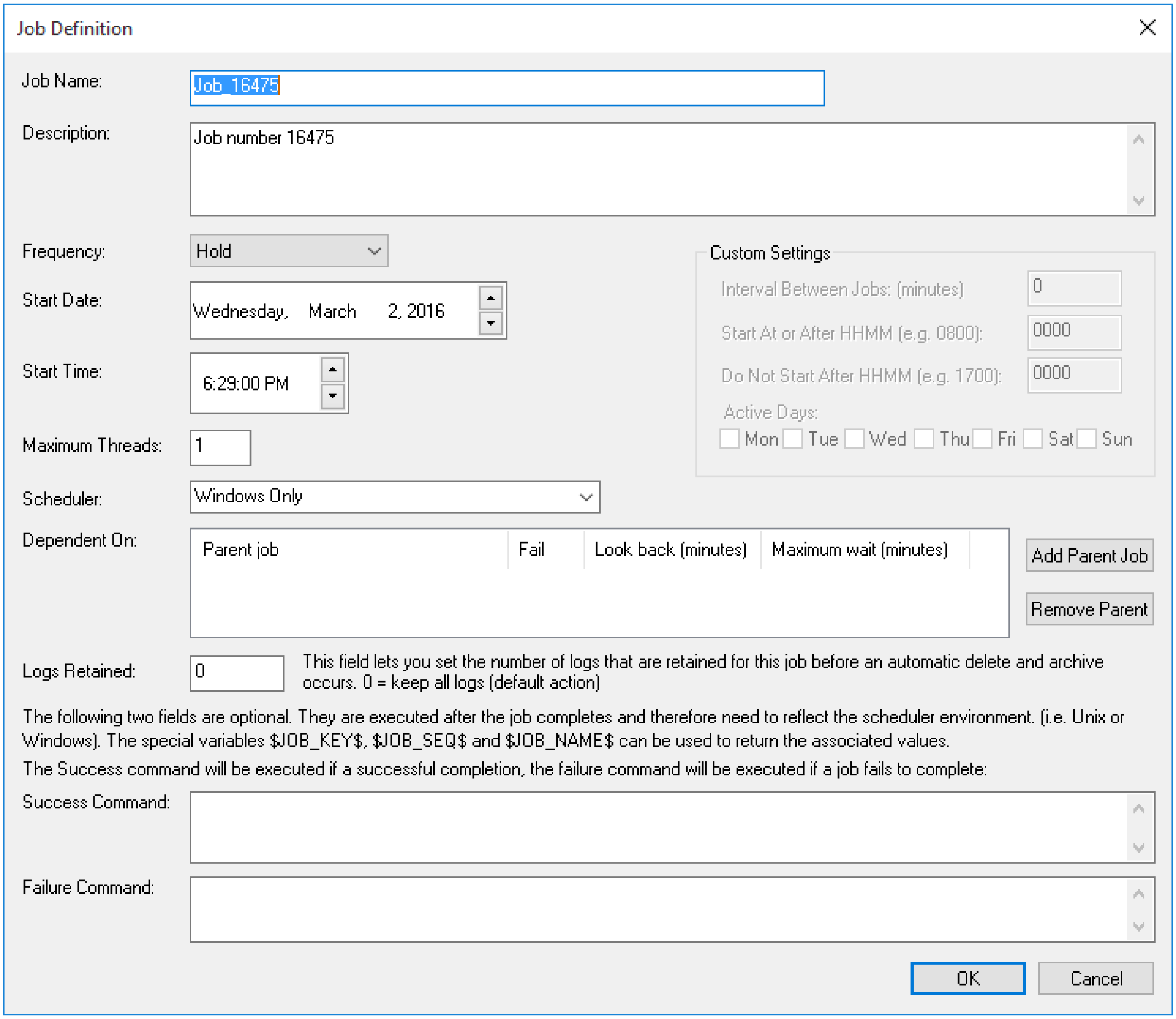
State	Description
	normally have a blank status.
Running	The task is currently running.
Failed	The task has had a fatal error. Any dependencies on this task will be held. Double-click the task to see more detail error information or review the audit and error/detail log for the job.
Completed	The task has completed successfully.
Error Completion	The task has completed with a handled Error. Any dependent tasks will be held, and the job must be restarted when the problem is rectified.
Bad Return Status	The task has returned an unknown status. This normally occurs with script files that produce unexpected information. The rule for scripts is that the first line returned must be a status of either 1, -1, -2, or -3. The second line is a message detailing the result. If the first line does not contain one of these four values, then this status will be returned and dependent tasks held. Run the script manually to view the output or check the logs.

Scheduling a Job

1. Open the Scheduler by clicking the **Scheduler** button from the toolbar.
2. Click **File > New Job** from the menu in the top of the screen, or click the **New Job** button from the toolbar.



3. Enter the required details in the **Job Definition** window.



Job Definition

Job Name: Job_16475

Description: Job number 16475

Frequency: Hold

Start Date: Wednesday, March 2, 2016

Start Time: 6:29:00 PM

Maximum Threads: 1

Scheduler: Windows Only

Custom Settings

Interval Between Jobs: (minutes) 0

Start At or After HHMM (e.g. 0800): 0000

Do Not Start After HHMM (e.g. 1700): 0000

Active Days:

□ Mon □ Tue □ Wed □ Thu □ Fri □ Sat □ Sun

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)

Logs Retained: 0

This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

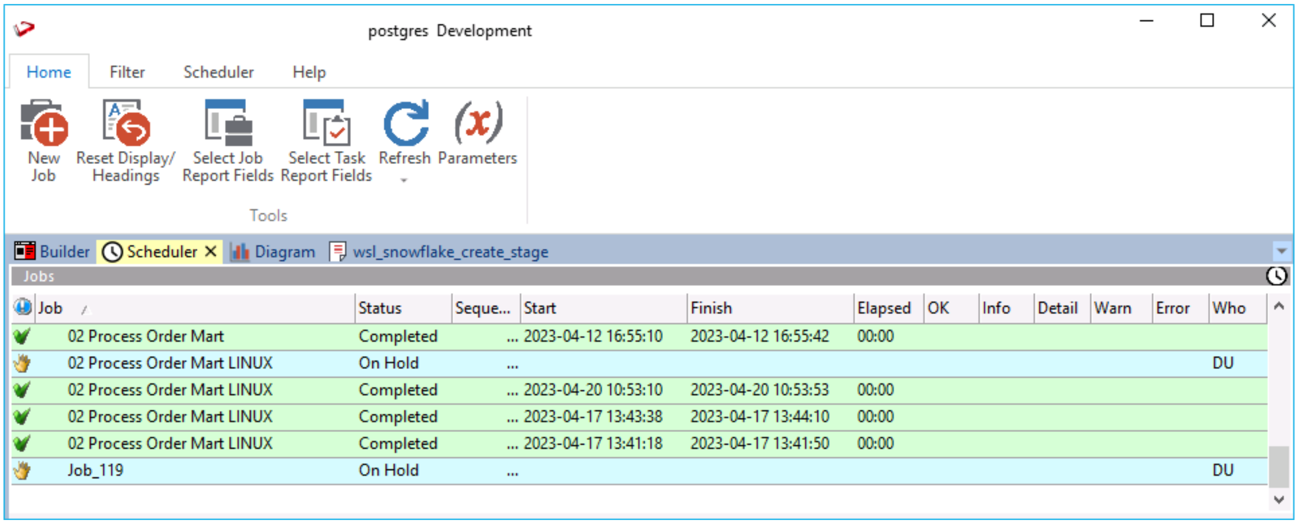
The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

Success Command:

Failure Command:

Refer to **Creating a Job** for more details on how to create a new job.

4. Once the job has been created, click the **All Jobs** button from the toolbar. The newly created job is now displayed in the Scheduler window.



	Job	Status	Seque...	Start	Finish	Elapsed	OK	Info	Detail	Warn	Error	Who
✓	02 Process Order Mart	Completed	...	2023-04-12 16:55:10	2023-04-12 16:55:42	00:00						
✋	02 Process Order Mart LINUX	On Hold	...									DU
✓	02 Process Order Mart LINUX	Completed	...	2023-04-20 10:53:10	2023-04-20 10:53:53	00:00						
✓	02 Process Order Mart LINUX	Completed	...	2023-04-17 13:43:38	2023-04-17 13:44:10	00:00						
✓	02 Process Order Mart LINUX	Completed	...	2023-04-17 13:41:18	2023-04-17 13:41:50	00:00						
✋	Job_119	On Hold	...									DU

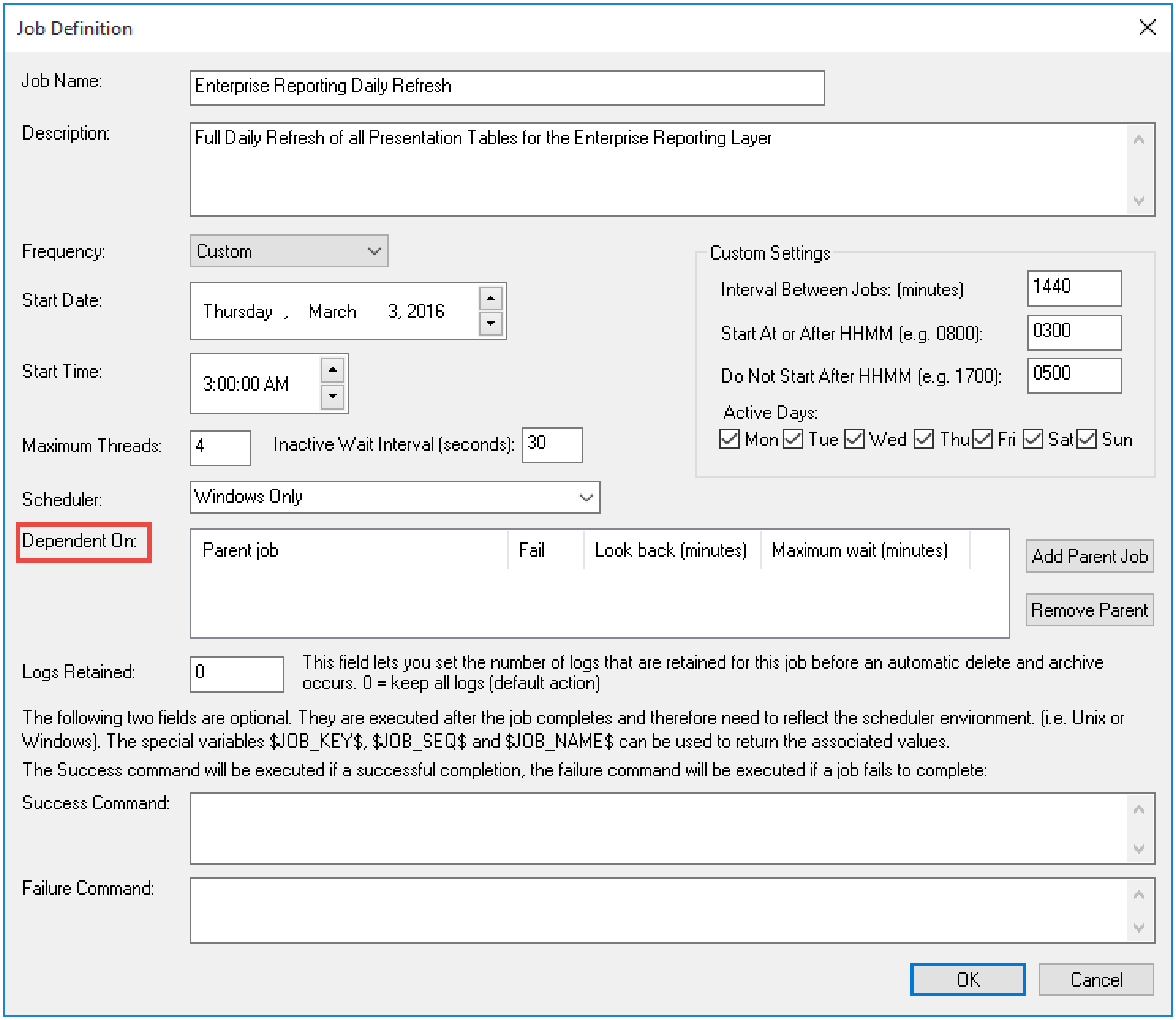
To create a job within a job

It is possible to schedule one job from another job. There are however some limitations and rules that must be understood when doing this.

1. A job that is called from another job is only ever allocated one thread. All tasks within the called job will therefore run sequentially.
2. A job can only have one running iteration. Therefore, a called job is blocked if that job is already running independently or as part of another job.
3. Any job dependencies for the called job are ignored. The parent's job dependencies are the only ones that are used.
4. A called job essentially runs as a separate job, so that if it fails both it and the parent job shows in a failed state. Once the problem is fixed, the parent job should be restarted which restarts the called job.

To create a job dependency

It is possible to make a job dependent on another job, using the **Dependent On** field in the **Job Definition** window.



Job Name: Enterprise Reporting Daily Refresh

Description: Full Daily Refresh of all Presentation Tables for the Enterprise Reporting Layer

Frequency: Custom

Start Date: Thursday , March 3, 2016

Start Time: 3:00:00 AM

Maximum Threads: 4

Inactive Wait Interval (seconds): 30

Scheduler: Windows Only

Custom Settings

Interval Between Jobs: (minutes) 1440

Start At or After HHMM (e.g. 0800): 0300

Do Not Start After HHMM (e.g. 1700): 0500

Active Days:

☑ Mon ☑ Tue ☑ Wed ☑ Thu ☑ Fri ☑ Sat ☑ Sun

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)

Logs Retained: 0

This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

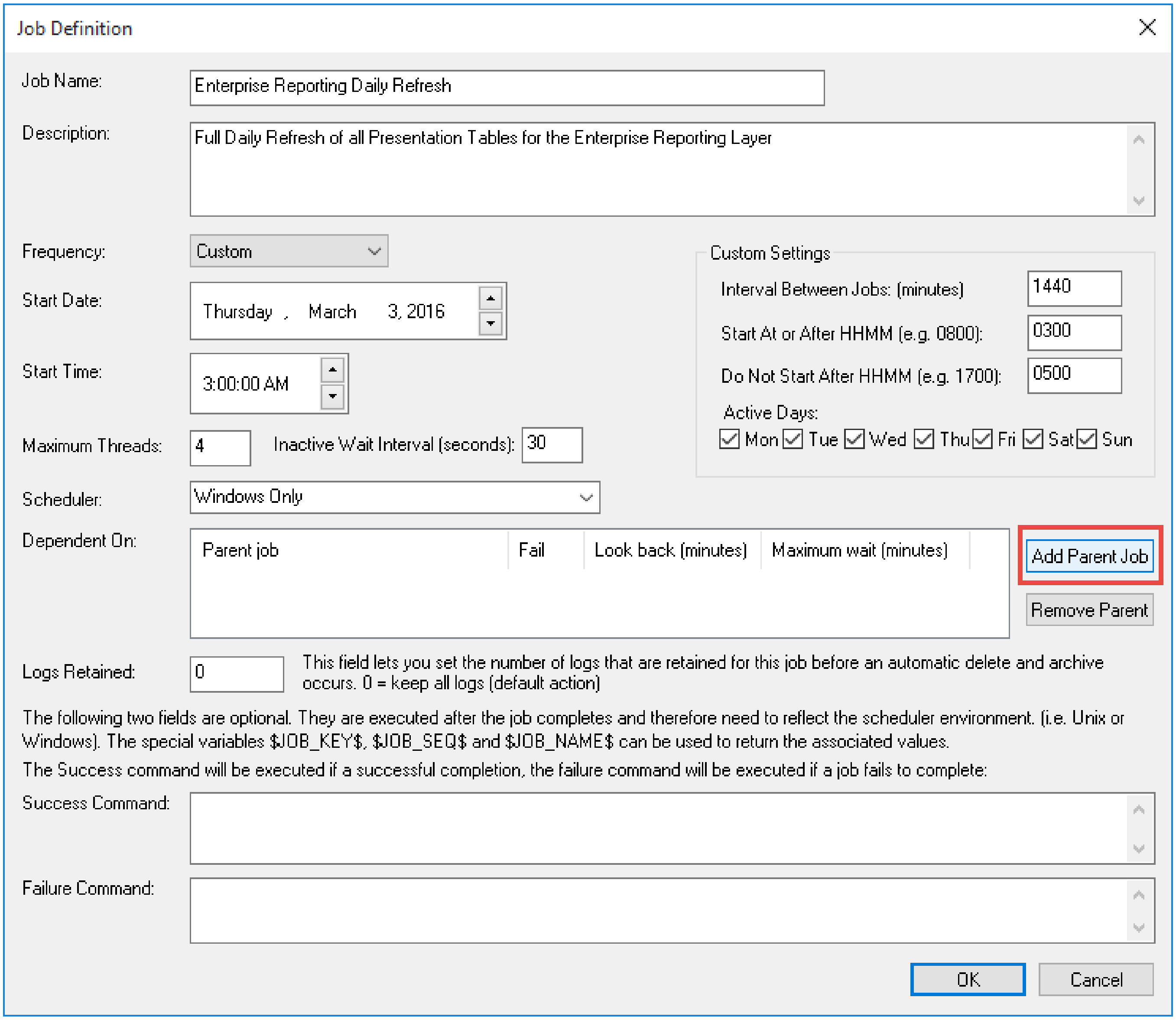
The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

Success Command:

Failure Command:

Click the **Add Parent Job** button.



Job Definition

Job Name: Enterprise Reporting Daily Refresh

Description: Full Daily Refresh of all Presentation Tables for the Enterprise Reporting Layer

Frequency: Custom

Start Date: Thursday , March 3, 2016

Start Time: 3:00:00 AM

Maximum Threads: 4 Inactive Wait Interval (seconds): 30

Scheduler: Windows Only

Custom Settings

Interval Between Jobs: (minutes) 1440

Start At or After HHMM (e.g. 0800): 0300

Do Not Start After HHMM (e.g. 1700): 0500

Active Days:
☑ Mon ☑ Tue ☑ Wed ☑ Thu ☑ Fri ☑ Sat ☑ Sun

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)

Logs Retained: 0 This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

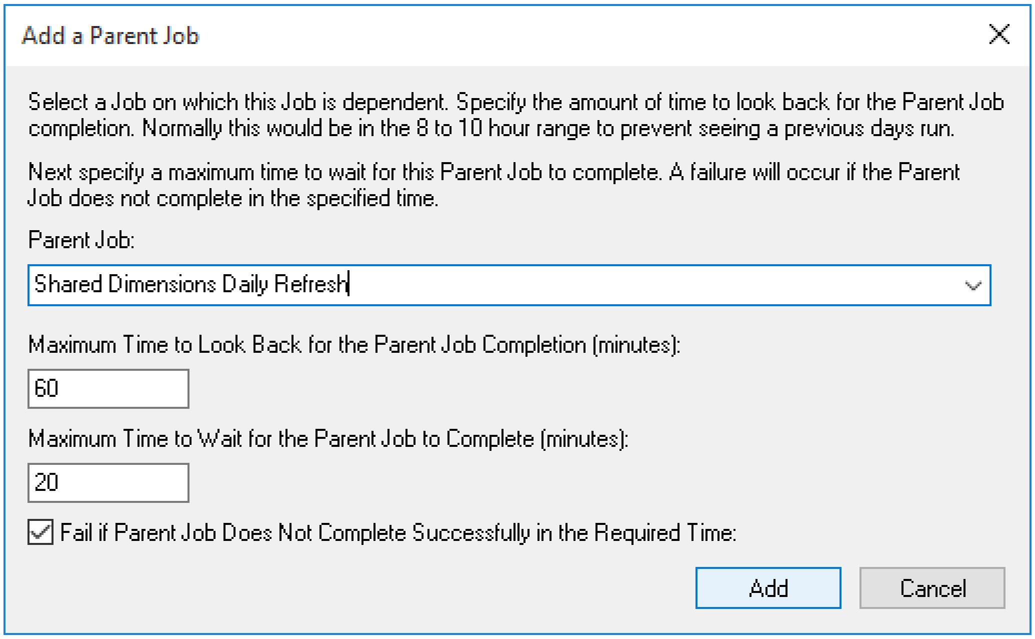
The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

Success Command:

Failure Command:

In the window that appears, select the **Parent Job** from the drop-down list. In our case we choose the job **Shared Dimensions Daily Refresh**.



Add a Parent Job

Select a Job on which this Job is dependent. Specify the amount of time to look back for the Parent Job completion. Normally this would be in the 8 to 10 hour range to prevent seeing a previous days run.

Next specify a maximum time to wait for this Parent Job to complete. A failure will occur if the Parent Job does not complete in the specified time.

Parent Job: Shared Dimensions Daily Refresh

Maximum Time to Look Back for the Parent Job Completion (minutes): 60

Maximum Time to Wait for the Parent Job to Complete (minutes): 20

☑ Fail if Parent Job Does Not Complete Successfully in the Required Time:

The **Maximum Time to Look Back for the Parent Job Completion** field prevents older iterations of the parent job as being identified as a completion. In our example, we are starting both jobs at 3am, so we don't need to look too far back to ensure that the dimension refresh has run. We have therefore set the look back minutes to 60 to allow for any delays in starting this job.

The **Maximum Time to Wait for the Parent Job to Complete** specifies how long to await a successful completion of the parent job. In our example, we know that the dimension refresh only takes a few minutes, but we should allow for the occasional slow network or resource drains making the dimension refresh take longer; so we have set the maximum wait to 20 minutes. This means that our job will wait 20 minutes from its own scheduled start time for the parent job to complete.

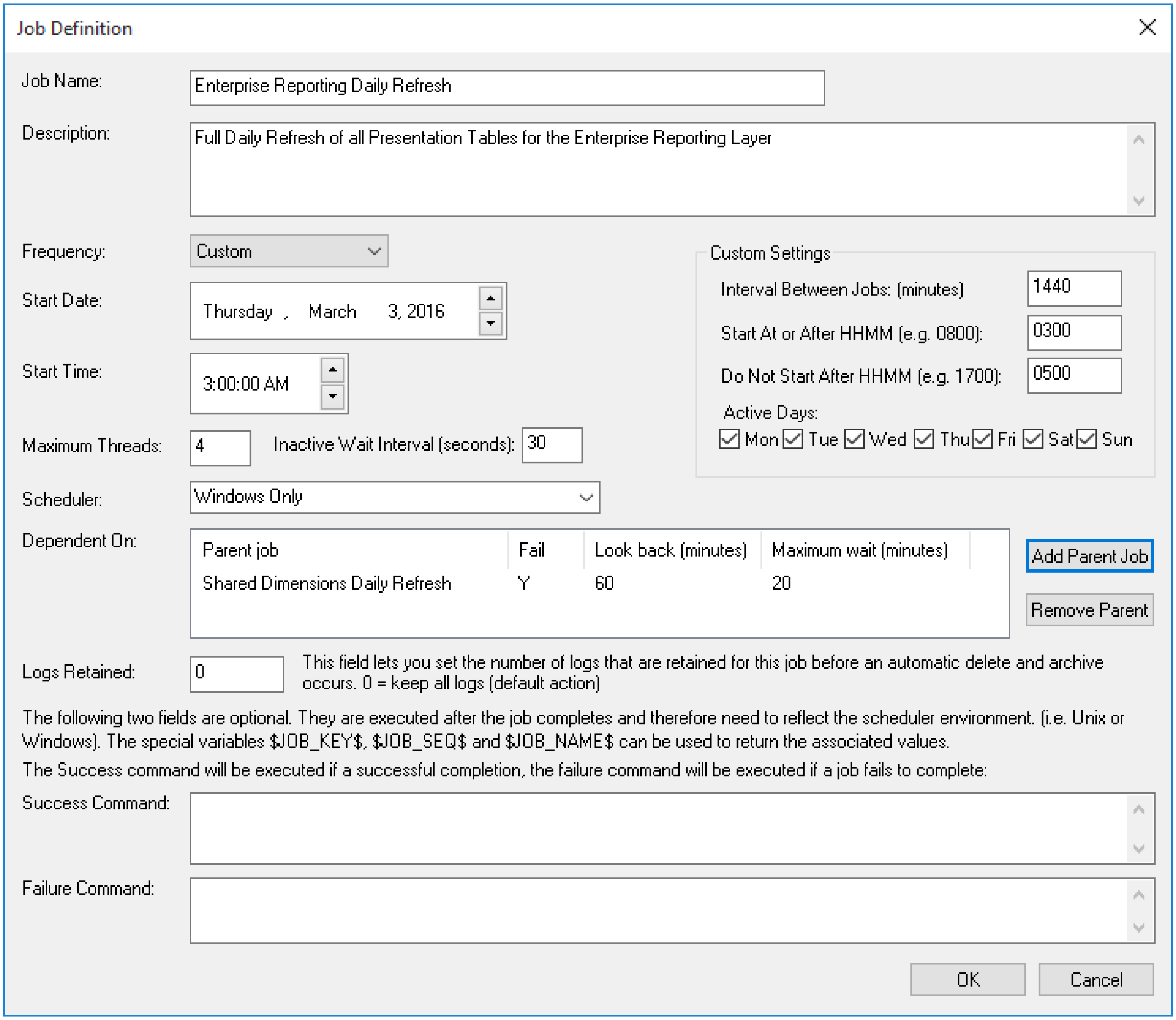
The check box to fail if the parent job does not complete in time prevents this job from running if the parent job (dimension refresh) does not complete successfully. We do not want the transactional data in our fact deliveries to be flagged with ‘Unknown’ dimensional item(s), so we can leave this check box selected to ensure that this job does not run.

Click **Add**.

Note:

Clearing the check box to fail if the parent fails will simply ensure that this job waits for the completion of the dimension refresh, and irrespective of the dimensions refresh success or failure, it starts.

Click **OK** to link this data job to the parent dimensional job. In this way, the job **Enterprise Reporting Daily Refresh** cannot run until the parent job **Shared Dimensions Daily Refresh** has completed successfully; thus the facts will have the latest dimensional keys associated with them.



Job Name: Enterprise Reporting Daily Refresh

Description: Full Daily Refresh of all Presentation Tables for the Enterprise Reporting Layer

Frequency: Custom

Start Date: Thursday , March 3, 2016

Start Time: 3:00:00 AM

Maximum Threads: 4

Inactive Wait Interval (seconds): 30

Scheduler: Windows Only

Custom Settings

Interval Between Jobs: (minutes) 1440

Start At or After HHMM (e.g. 0800): 0300

Do Not Start After HHMM (e.g. 1700): 0500

Active Days:

☑ Mon ☑ Tue ☑ Wed ☑ Thu ☑ Fri ☑ Sat ☑ Sun

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)
Shared Dimensions Daily Refresh	Y	60	20

Logs Retained: 0

This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

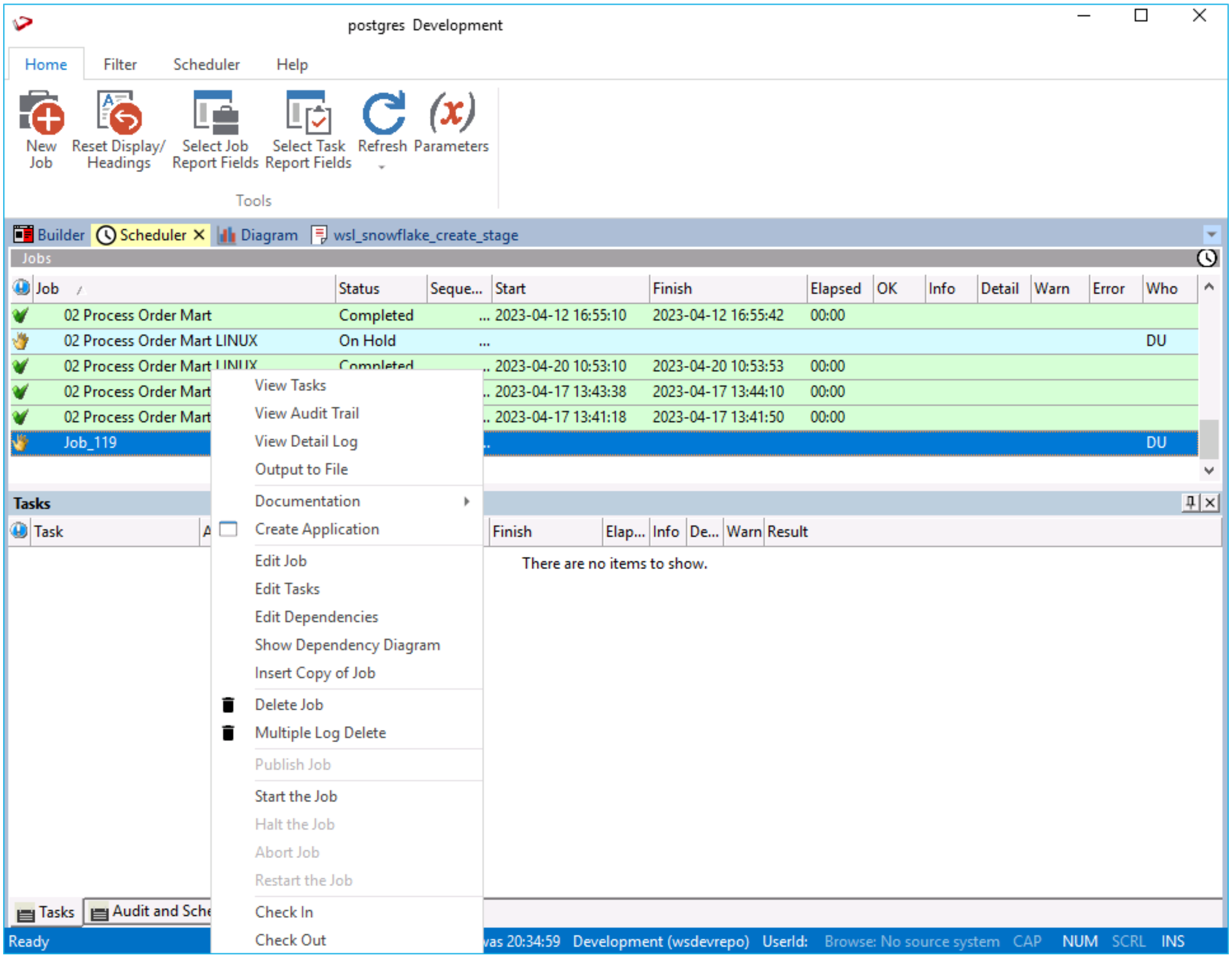
The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

Success Command:

Failure Command:

Working with Jobs

When you right click a Job in the Scheduler window, the context menu provides several options for working with the job.



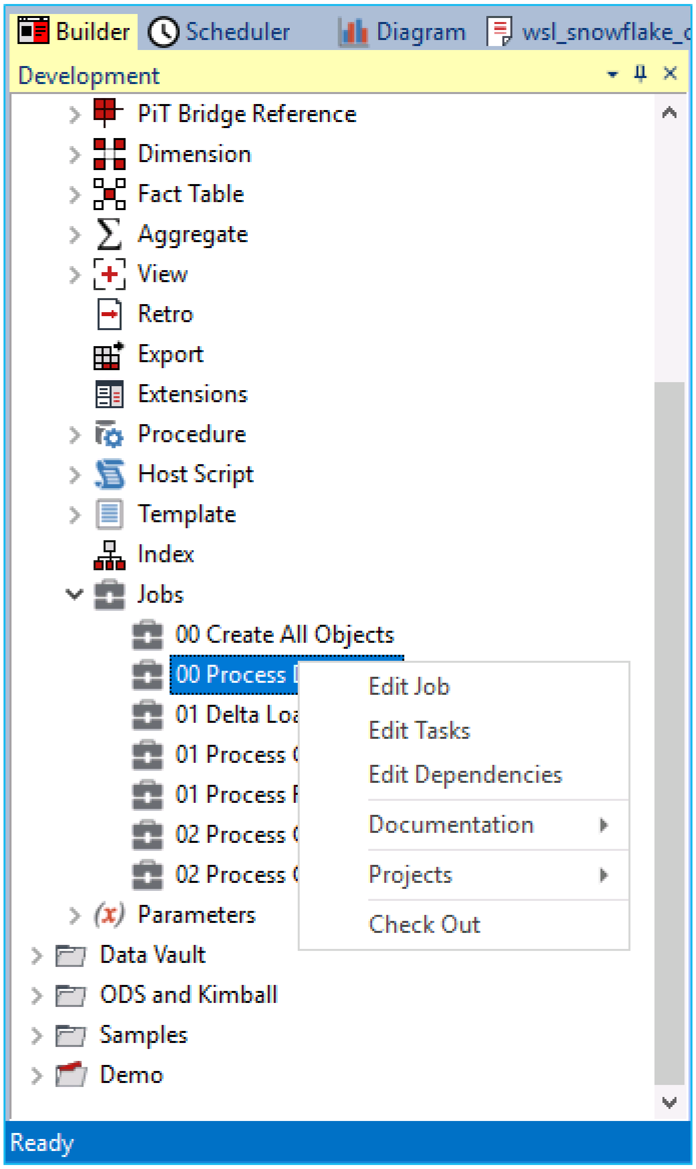
Job	Status	Seque...	Start	Finish	Elapsed	OK	Info	Detail	Warn	Error	Who
02 Process Order Mart	Completed	...	2023-04-12 16:55:10	2023-04-12 16:55:42	00:00						
02 Process Order Mart LINUX	On Hold	...									DU
02 Process Order Mart LINUX	Completed	..	2023-04-20 10:53:10	2023-04-20 10:53:53	00:00						
02 Process Order Mart		..	2023-04-17 13:43:38	2023-04-17 13:44:10	00:00						
02 Process Order Mart		..	2023-04-17 13:41:18	2023-04-17 13:41:50	00:00						
Job_119		.									DU

Tasks

Task	A	Finish	Elap...	Info	De...	Warn	Result

There are no items to show.

You can also access and manage the scheduler jobs from the Objects list pane of the **Builder** window.



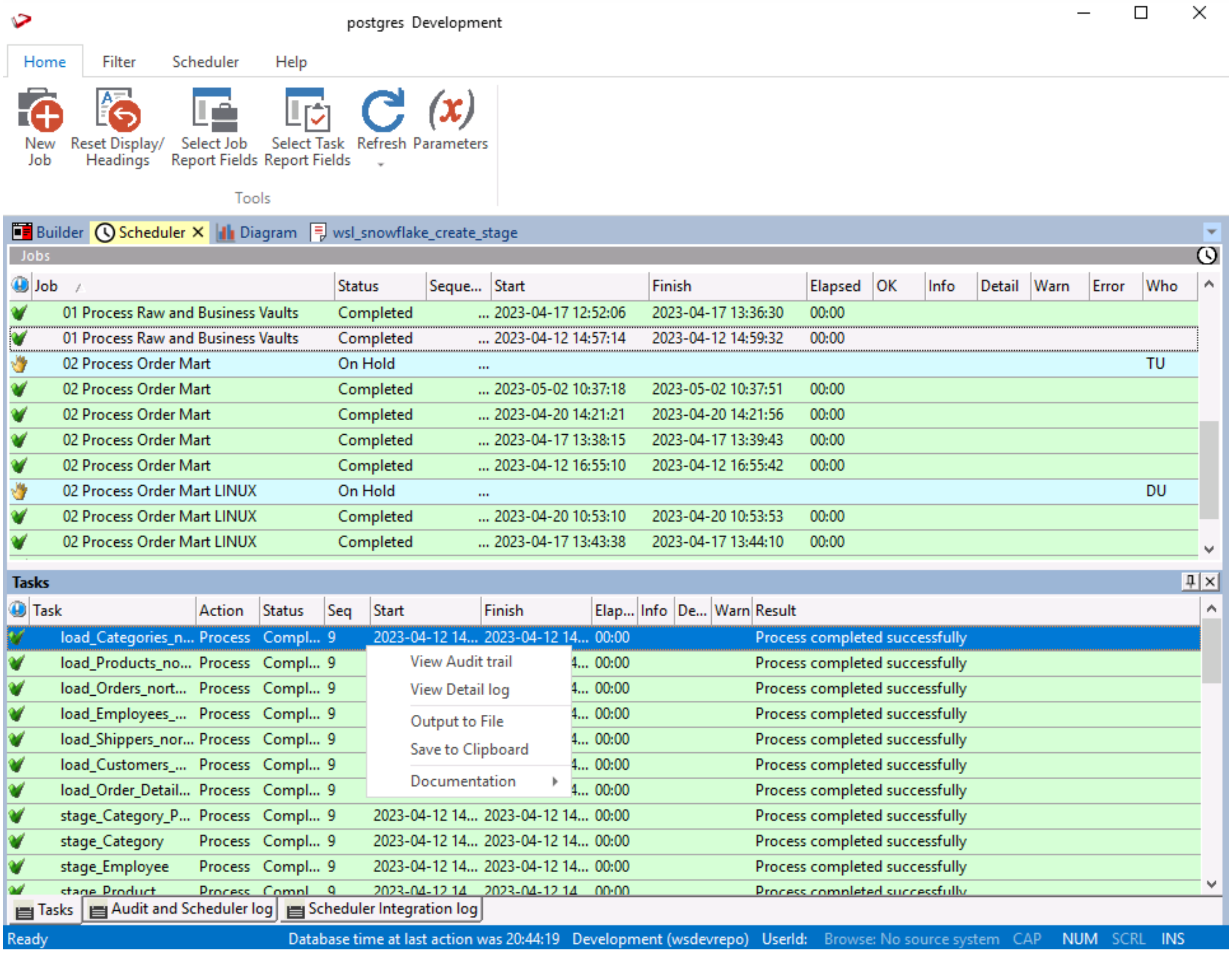
Some of the options are discussed in more detail in the succeeding sections, however a brief overview of the menu options follows:

Option	Description
View Tasks	Enables you to view the tasks of a job.
View Audit Trail	Enables you to view the audit trail of a job.
View Detail Log	Enables you to view the detail log of a job.
Output to File	Enables you to export the jobs to a CSV file.
Documentation	Enables you to display or create documentation for a job.
Edit Job	Enables you to edit a job. Refer to **Editing a Job** for details.
Edit Tasks	Enables you to edit the tasks of a job. Refer to **Editing Tasks in a Job** for details.
Edit Dependencies	Enables you to edit the task dependencies of a job. Refer to **Editing Task Dependencies** for details.
Insert Copy of Job	Enables you to insert a copy of a job. Refer to **Inserting a Copy of a Job** for details.
Delete Job	Enables you to delete a job. Refer to **Deleting a Job** for details.
Multiple Log Delete	Enables you to delete multiple logs of a job. Refer to **Deleting Job Logs** for details.

Option	Description
Start the Job	Enables you to start a job. Refer to **Starting a Job** for details.
Halt the Job	Enables you to halt a job. Refer to **Halting a Job** for details.
Abort the Job	Enables you to abort a job. Refer to **Aborting a Job** for details.
Restart the Job	Enables you to restart a job. Refer to **Restarting a Job** for details.

Job Tasks

You can display the tasks associated with a job by right-clicking the job and selecting **View Tasks** from the context menu. The tasks are listed below the Jobs pane and provides the following context menu options:



Job	Status	Seque...	Start	Finish	Elapsed	OK	Info	Detail	Warn	Error	Who
01 Process Raw and Business Vaults	Completed	...	2023-04-17 12:52:06	2023-04-17 13:36:30	00:00						
01 Process Raw and Business Vaults	Completed	...	2023-04-12 14:57:14	2023-04-12 14:59:32	00:00						
02 Process Order Mart	On Hold	...									TU
02 Process Order Mart	Completed	...	2023-05-02 10:37:18	2023-05-02 10:37:51	00:00						
02 Process Order Mart	Completed	...	2023-04-20 14:21:21	2023-04-20 14:21:56	00:00						
02 Process Order Mart	Completed	...	2023-04-17 13:38:15	2023-04-17 13:39:43	00:00						
02 Process Order Mart	Completed	...	2023-04-12 16:55:10	2023-04-12 16:55:42	00:00						
02 Process Order Mart LINUX	On Hold	...									DU
02 Process Order Mart LINUX	Completed	...	2023-04-20 10:53:10	2023-04-20 10:53:53	00:00						
02 Process Order Mart LINUX	Completed	...	2023-04-17 13:43:38	2023-04-17 13:44:10	00:00						

Task	Action	Status	Seq	Start	Finish	Elap...	Info	De...	Warn	Result
load_Categories_n...	Process	Compl...	9	2023-04-12 14...	2023-04-12 14...	00:00				Process completed successfully
load_Products_no...	Process	Compl...	9		4...	00:00				Process completed successfully
load_Orders_nort...	Process	Compl...	9		4...	00:00				Process completed successfully
load_Employees_...	Process	Compl...	9		4...	00:00				Process completed successfully
load_Shippers_nor...	Process	Compl...	9		4...	00:00				Process completed successfully
load_Customers_...	Process	Compl...	9		4...	00:00				Process completed successfully
load_Order_Detail...	Process	Compl...	9		4...	00:00				Process completed successfully
stage_Category_P...	Process	Compl...	9	2023-04-12 14...	2023-04-12 14...	00:00				Process completed successfully
stage_Category	Process	Compl...	9	2023-04-12 14...	2023-04-12 14...	00:00				Process completed successfully
stage_Employee	Process	Compl...	9	2023-04-12 14...	2023-04-12 14...	00:00				Process completed successfully
stage_Product	Process	Compl...	9	2023-04-12 14...	2023-04-12 14...	00:00				Process completed successfully

Ready Database time at last action was 20:44:19 Development (wsdevrepo) UserId: Browse: No source system CAP NUM SCRL INS

Option	Description
View Audit Trail	Enables you to view the audit trail of a task.
View Detail Log	Enables you to view the detail log of a task.
Output to File	Enables you to export the task details to a CSV file.
Save to Clipboard	• **Tasks**- Enables you to copy the contents of the **Result** column(s) of the selected line(s) from the **Tasks** list to the Windows clipboard.

Option	Description
	- **Audit and Scheduler log** - Enables you to copy the contents of the **Message** column(s) of the selected line(s) from the **Audit and Scheduler log** to the Windows clipboard.
Documentation	Enables you to display or create documentation for a task.

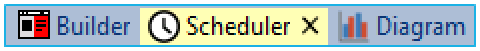
Creating a Job

Tip:

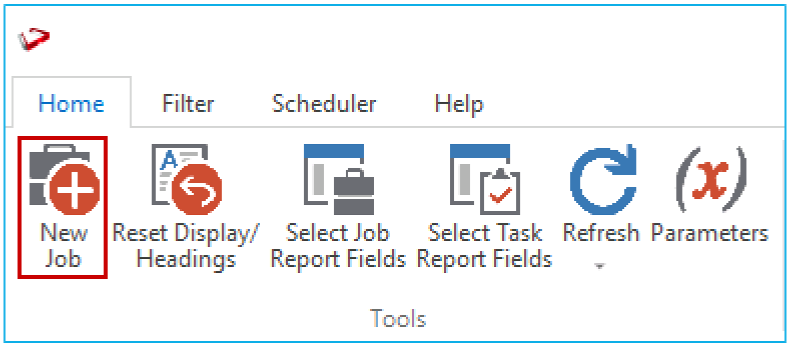
You can create a Scheduler job from:

- Projects or Project Groups—refer to **Building Scheduler Jobs from Projects or Groups** for details.
- Object Groups—refer to **Building Scheduler Jobs from Object Groups** for details.
- Diagrams—refer to **Creating a Job from a Diagram** for details.

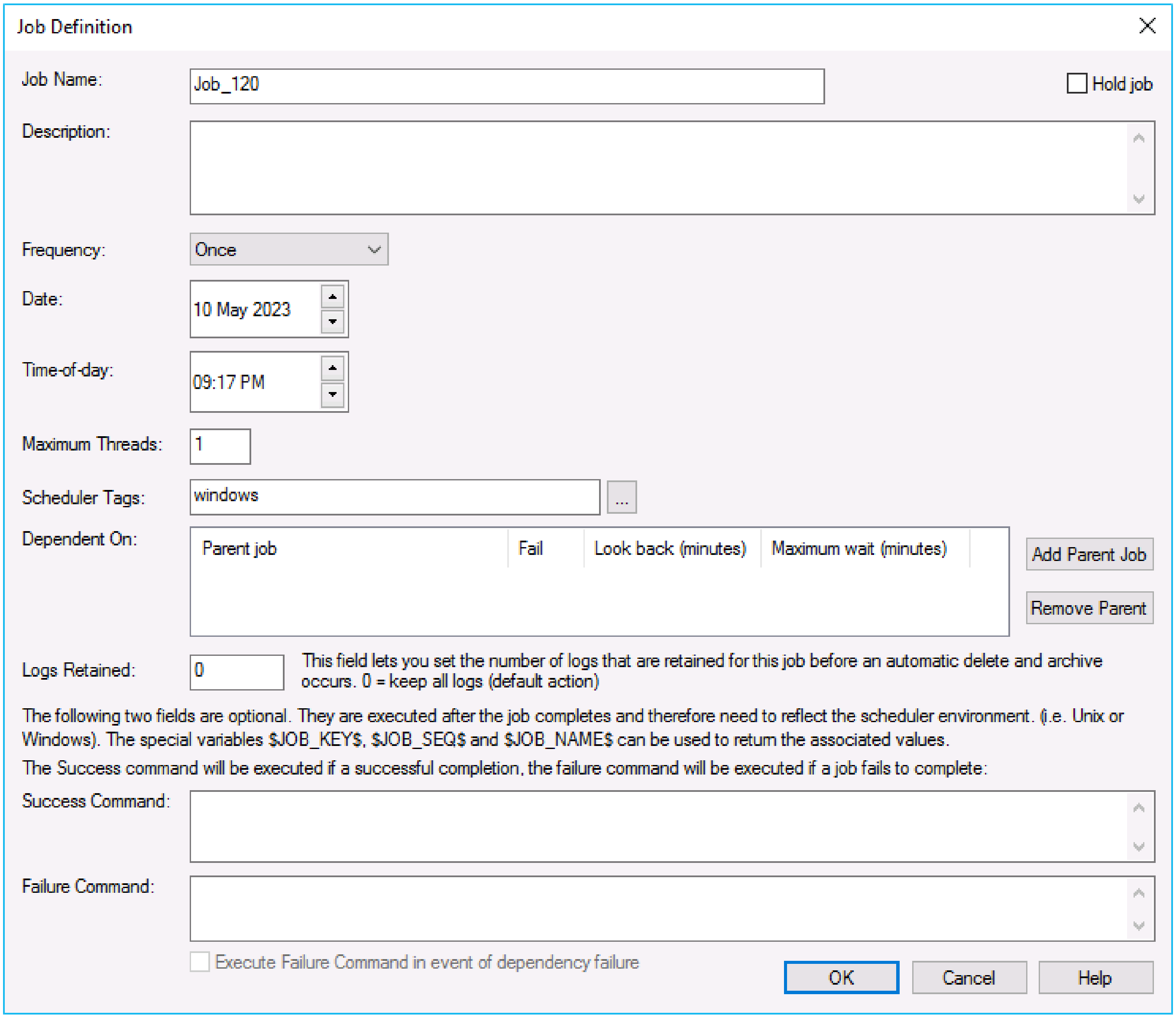
Click the **Scheduler** tab to open the **Scheduler** window.



Click the **New Job** button to create a new job.



A **Job Definition** window is displayed.



Job Definition

Job Name: Job_120 □ Hold job

Description:

Frequency: Once

Date: 10 May 2023

Time-of-day: 09:17 PM

Maximum Threads: 1

Scheduler Tags: windows

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)

Logs Retained: 0 This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

Success Command:

Failure Command:

□ Execute Failure Command in event of dependency failure

Complete the fields and then click **OK**. The main fields are described in the following table:

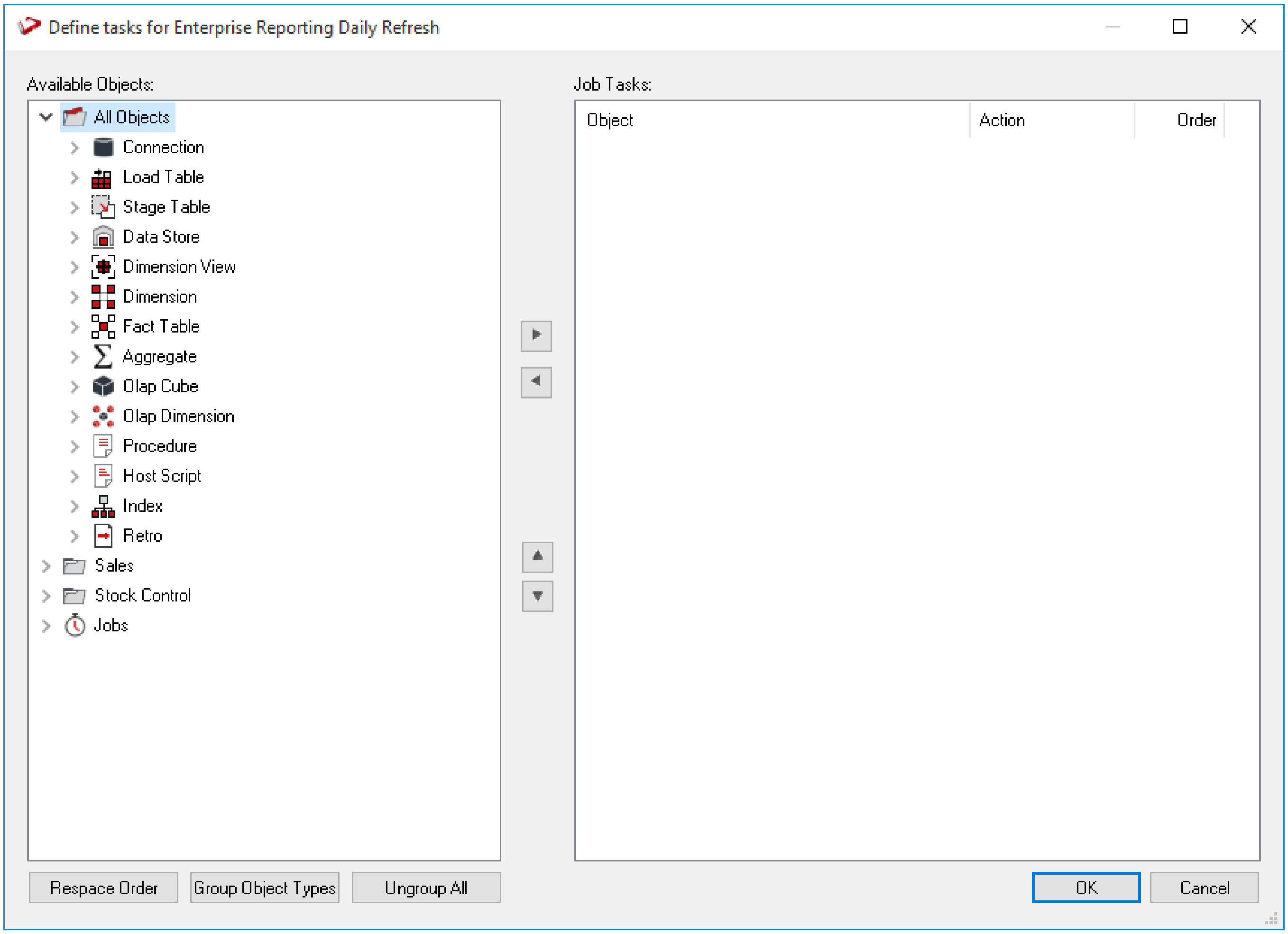
Field	Description
Job Name	The Scheduler defaults to the next job number in the sequence. You can alter this to any alphanumeric. **Tip** Only alphanumerics, spaces and the underscore are supported in the name. **Warning** On some UNIX systems, long job names can cause jobs to be canceled (see Knowledge Base article 67), so where possible keep the name short.
Description	A description of the job.
Frequency	When the job runs. The options available in the drop-down list box are: • Once - job is deleted on completion • Custom - enables custom definition • Weekly - runs the job weekly • Monthly - runs the job monthly

Field	Description
	• Annually - runs the job annually
Start Date and Start Time	The date and time for the job to start.
Max Threads	The maximum number of threads allocated to run the job, e.g, if some tasks can run in parallel then if more than one thread is allocated then they will run in parallel.
Scheduler	Certain types of jobs will only run in a specific environment. For example ODBC based loads can only be handled by the Windows scheduler. It is possible to have multiple schedulers running. Select the desired scheduler from this drop-down. The valid options are: UNIX Preferred, UNIX Only, Windows Preferred, Windows Only, or the name of a specific scheduler can be entered (e.g. WIN0002)
Dependent On	A job can be dependent on the successful completion of one or more other jobs. Click the **Add Parent Job** button to select a job that this job will be dependent on. The maximum time to look back for parent job completion field prevents older iterations of the parent job as being identified as a completion. The maximum time to wait specifies how long to await a successful completion of the parent job. The action if that wait expires can also be set. Refer to the Job Dependency example in **Scheduling a Job** for details.
Logs Retained	Specify the number of logs to retain for the job. By default, all logs are retained. This field can be used to reduce the build up of scheduler logs by specifying a number of logs to retain.
Success command and Failure command	These are either UNIX or Windows shell commands depending on which scheduler is used. They are executed if the condition is met. Typically, these commands would mail or page on success or failure. **Notes:** • The RED scheduler does not check return codes from called commands, scripts and programs. • It is recommended that all output from commands, scripts and programs is re-directed to a log file. For example, add this to the end of any SUCCESS/FAILURE commands: >> c:\scheduler\success_failure_prod.log 2>&1

The following fields are available if a frequency of **Custom** is chosen:

Field	Description
Interval between jobs (Minutes)	Specify the number of minutes between iterations of the job. For example, to run a job every 30 minutes set this value to 30. If a job is to run only once but on selected days set this value to 1440 (daily)
Start at or after HHMM	The time that the job may run from. To run anytime set to 0000.
Do not start after HHMM	If multiple iterations are being done then this is the time after which a new iteration will not be started. For example, if a job is running every 10 minutes it will continue until this time is reached. To run till the end of day set to 2400.
Active on the days	Select each day of the week that the custom job is to be active on.

Once the job itself has been defined, tasks then need to be added to the job. The **Define tasks** window is shown below.



The screen has two main areas. The right pane shows the tasks to be run for this job and the left pane lists all the objects.

Adding a task

Double-click an object in the left pane to add it to the task list in the right pane. Normally objects, such as Load or Fact tables are scheduled rather than procedures.

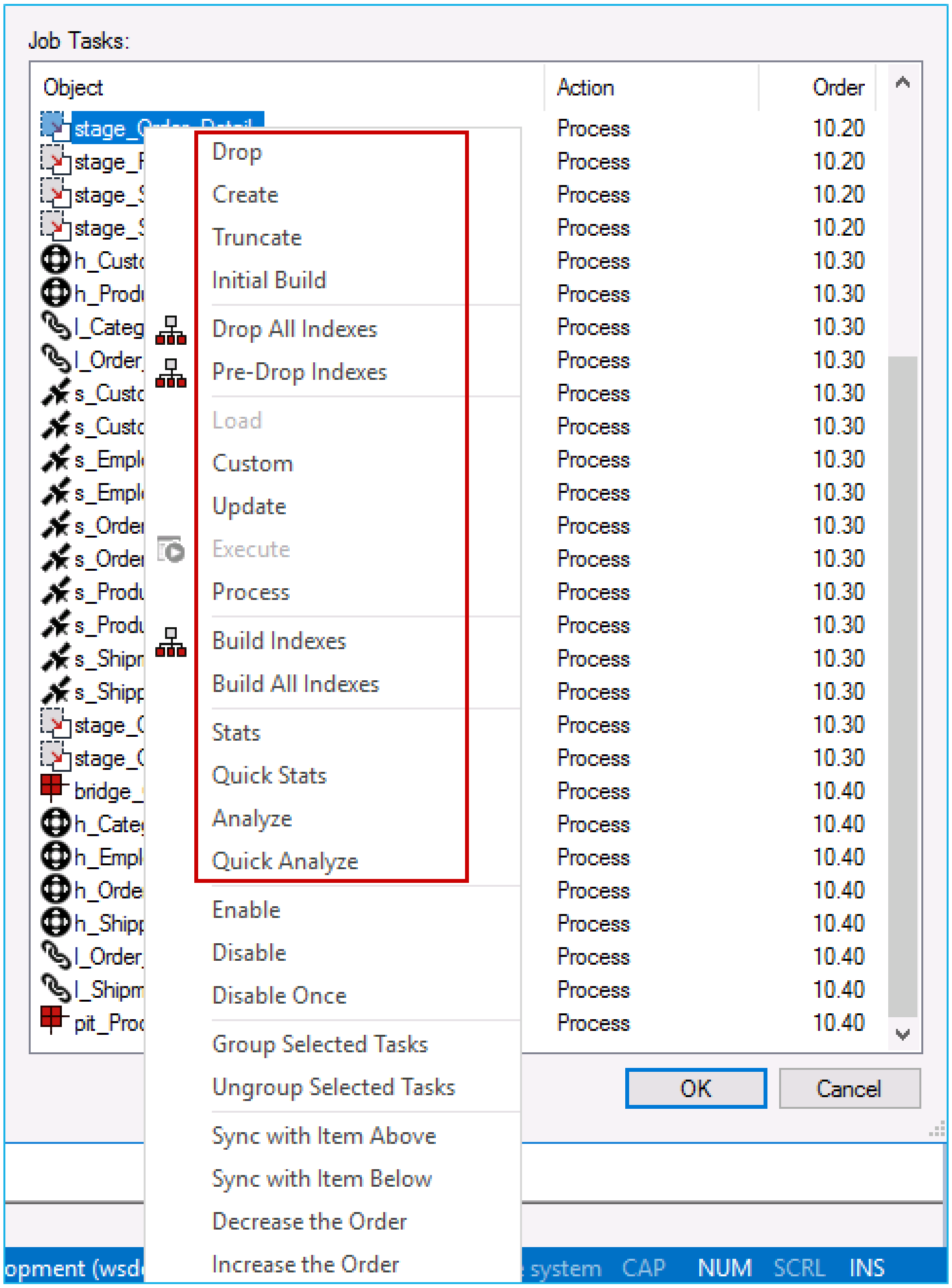
Setting the action on a task

Each task can have a specific action that is to be performed on its object.

The default action for **load tables** is **process**. This means that when the task is actioned, it will drop any indexes that are due to be dropped, or have **pre-drop** set, then load the table and perform any post-load procedures or transformations and then re-create any dropped indexes.

The default action for all other tables is the same as above, except it will execute the **update** procedure rather than loading the table.

You can change the action on a task by right-clicking the task in the right pane. The menu options are shown below.



The following task actions are available:

Action	Description
Drop	Drop table, view or index.
Create	Create table, view or index.
Truncate	Delete all rows from the table.
Initial Build	**Drop All Indexes** then **Custom** then **Build All Indexes**.
Drop All Indexes	Drop all indexes on the table.
Pre Drop Indexes	Drop all indexes on the table marked as "Pre Drop".
Load	Load the table (Load tables only).
Custom	Run the custom procedure on the table.

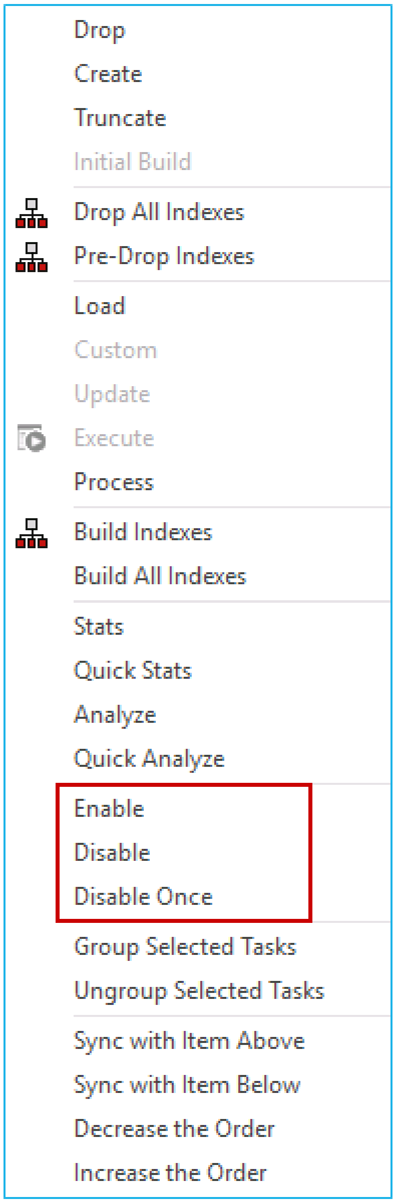
Action	Description
Update	Run the update procedure on the table.
Execute	Execute the procedure or host script.
Process	**Pre Drop Indexes** then **Update** and then **Build Indexes**.
Process and Statistics	**Process** then Default Stats as defined in the Table Properties > Statistics > Process and statistics method (DB2 only).
Build Indexes	Build the indexes on the table marked as "Pre Drop".
Build All Indexes	Build all indexes on the table.
SQL Server: Analyze	Performs: • **UPDATE STATISTICS name WITH FULLSCAN**
SQL Server: Quick Analyze	Performs: • **UPDATE STATISTICS name WITH SAMPLE 3 PERCENT**
SQL Server: Stats	Same as Analyze.
SQL Server: Quick Stats	Same as Quick Analyze.

Note

Not all actions are available on all object types.

Setting the state of a task

Each task can be set to a state:

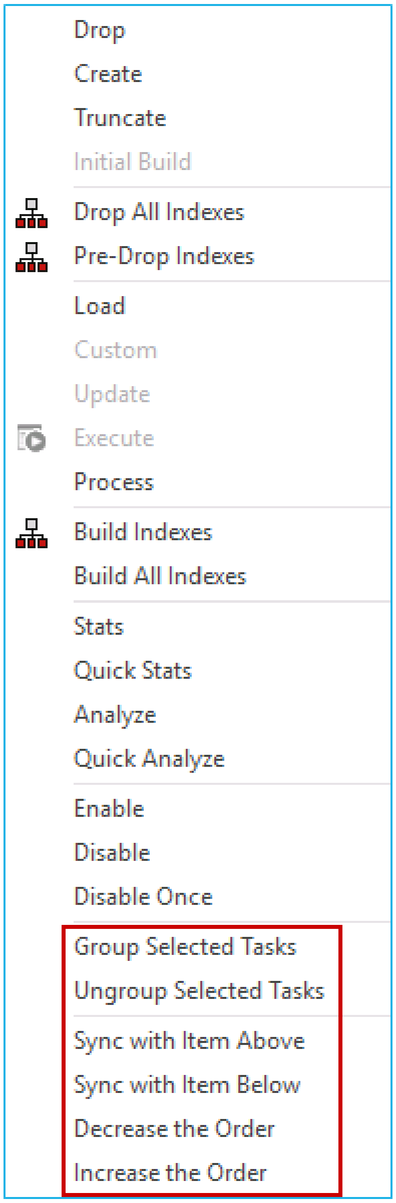


The following states are available:

State	Description
Enable	Job Task is enabled.
Disable	Job Task is disabled.
Disable Once	Job Task is disabled once and then reverts to enabled next time the Job is released by the Scheduler.

Creating dependencies between tasks

You can create dependencies between tasks in the list by selecting one or more tasks and right-clicking to bring up the dependency options.



The following task dependency options are available from the menu:

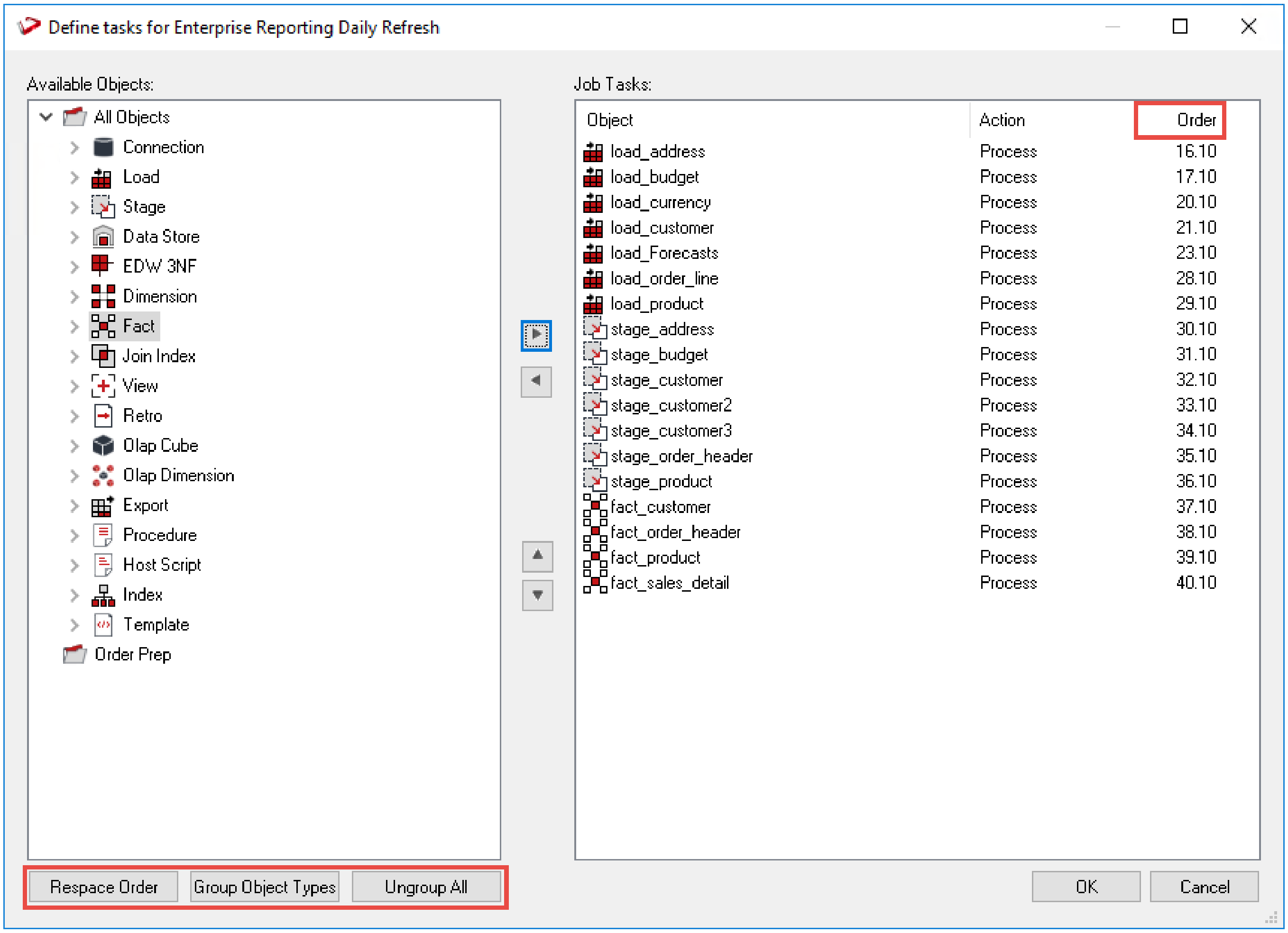
Task Option	Description
Group Selected Tasks	Groups two or more selected tasks to have the same order value, allowing them to run in parallel if the maximum threads setting allows.
Ungroup Selected Tasks	Un-group selected tasks.
Sync with Item Above	Changes a selected task to have the same order value as the task above it, allowing them to run in parallel if the maximum threads setting allows.
Sync with Item Below	Changes a selected task to have the same order value as the task below it, allowing them to run in parallel if the maximum threads setting allows.
Decrease the Order	Changes a selected task to an order number one less than its current value. The task will now run immediately before it would have previously.
Increase the Order	Changes a selected task to an order number one more than its current value. The task will now run immediately after it would have previously.

Ordering or Grouping the tasks

The **Order** column shows the order in which the tasks are to be run, e.g. 20.20 If the two numbers are the same as another task then those tasks can run in parallel. If the two numbers are different then those tasks run sequentially. This is an initial definition of dependencies. These dependencies can be altered specifically once the job has been created.

Tasks can be moved up or down by selecting the task and clicking the **Move Up** ▲ or **Move Down** ▼ buttons.

To respace the order of the tasks; to group or ungroup object types, use the **buttons** at the bottom of the **Define tasks** window.



Object	Action	Order
load_address	Process	16.10
load_budget	Process	17.10
load_currency	Process	20.10
load_customer	Process	21.10
load_Forecasts	Process	23.10
load_order_line	Process	28.10
load_product	Process	29.10
stage_address	Process	30.10
stage_budget	Process	31.10
stage_customer	Process	32.10
stage_customer2	Process	33.10
stage_customer3	Process	34.10
stage_order_header	Process	35.10
stage_product	Process	36.10
fact_customer	Process	37.10
fact_order_header	Process	38.10
fact_product	Process	39.10
fact_sales_detail	Process	40.10

Button	Description
Respace Order	Click to respace the order numbers. The existing dependency structure and groupings are retained. The purpose of this button is simply to allow room between tasks to fit new tasks. For example, if we have two tasks that have an order of 20.19.5 and 20.20.6 and we want to add a task between these two tasks, we can click the **Respace Order** button to open up a gap between the two tasks.
Group Object Types	Click to put all objects of the same type into groups. For example, all load tables will be able to run in parallel, all dimensions, etc.
Ungroup All	Click to remove all groupings and make all tasks sequential. New groupings can be made by selecting a range of sequentially listed tasks in the left pane and using the right-click menu option **Group Selected Tasks**. Tasks that are grouped have the same first two numbers in the order and can execute at the same time if the job has multiple threads.

Upon completion of adding tasks, click **OK** to exit.

Editing a Job

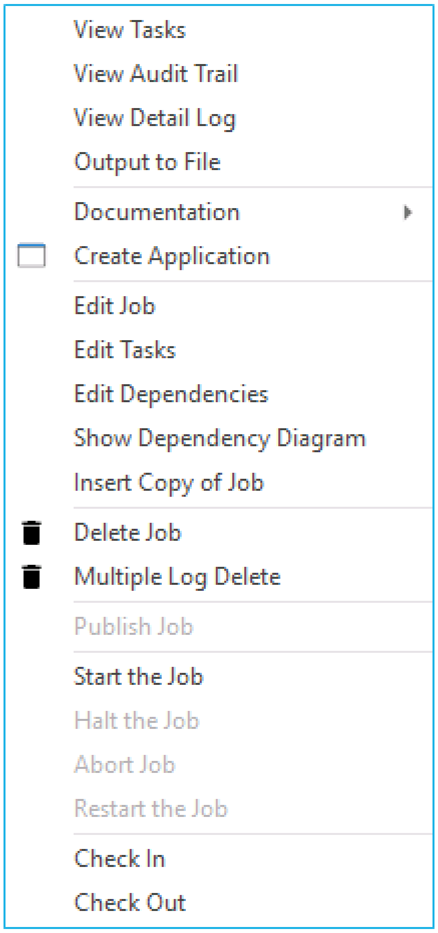
Once jobs have been created they can be edited.

Note

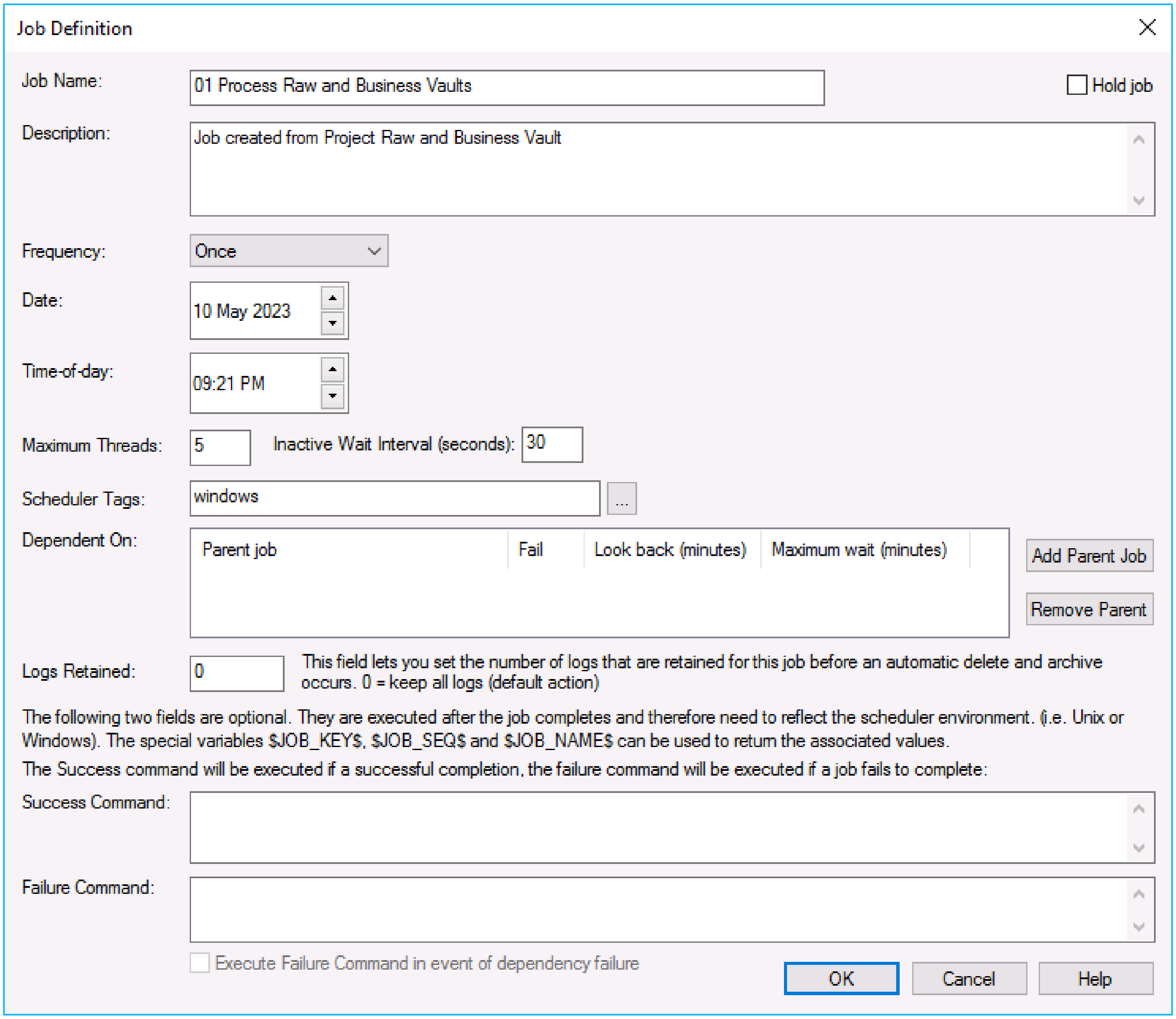
A job can only be edited when it is not in a running state and only if the job is a scheduled job. Completed jobs remain in the list but only logs remain.

To edit a job

Select the job from the middle pane and then right-click to select **Edit Job** from the context menu.



The **Job Definition** is displayed.



Job Definition

Job Name:	01 Process Raw and Business Vaults
□ Hold job	
Description:	Job created from Project Raw and Business Vault
Frequency:	Once
Date:	10 May 2023
Time-of-day:	09:21 PM
Maximum Threads:	5
Inactive Wait Interval (seconds):	30
Scheduler Tags:	windows

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)

Logs Retained: 0

This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

Success Command:

Failure Command:

□ Execute Failure Command in event of dependency failure

Edit the fields as required and click **OK**. The main fields are described in the following table:

Field	Description
Job Name	The Scheduler defaults to the next job number in the sequence. You can alter this to any alphanumeric. **Tip** Only alphanumerics, spaces and the underscore are supported in the name. **Warning** On some UNIX systems, long job names can cause jobs to be canceled (see Knowledge Base article 67), so where possible keep the name short.
Description	A description of the job
Frequency	When the job runs. The options available in the drop-down list box are: • Once Only - job is deleted on completion • Once and Hold - runs and puts another copy of the job on hold • Hold - puts the job on hold for manual release • Daily - runs the job daily

Field	Description
	- Custom - enables custom definition - Weekly - runs the job weekly - Monthly - runs the job monthly - Annually - runs the job annually
Start Date and Start Time	The date and time for the job to start.
Max Threads	The maximum number of threads allocated to run the job, e.g, if some tasks can run in parallel then if more than one thread is allocated then they will run in parallel. **Caution** Users must be aware that adding additional threads to a job increases the communication overhead with the DB server containing the Data Warehouse repository. Each job thread holds a dedicated connection to the server for its lifetime.
Scheduler	Certain types of jobs will only run in a specific environment. For example, ODBC based loads can only be handled by the Windows' scheduler. It is possible to have multiple schedulers running. Select the desired scheduler from this drop-down. The valid options are:UNIX Preferred, UNIX Only, Windows Preferred, Windows Only, or the name of a specific scheduler can be entered (e.g. WIN0002)
Dependent On	A job can be dependent on the successful completion of one or more other jobs. Click the **Add Parent Job** button to select a job that this job will be dependent on. The maximum time to look back for parent job completion field prevents older iterations of the parent job as being identified as a completion. The maximum time to wait specifies how long to await a successful completion of the parent job. The action if that wait expires can also be set. Refer to the Job Dependency example in **Scheduling a Job** for details.
Logs Retained	Specify the number of logs to retain for the job. By default, all logs are retained. This field can be used to reduce the build up of scheduler logs by specifying a number of logs to retain.
Success command and Failure command	These are either UNIX or Windows shell commands depending on which scheduler is used. They are executed if the condition is met. Typically, these commands would mail or page on success or failure. **Notes** - The RED scheduler does not check return codes from called commands, scripts and programs. - It is recommended that all output from commands, scripts and programs is redirected to a log file. For example, add this to the end of any SUCCESS/FAILURE commands: >> c:\scheduler\success_failure_prod.log 2>&1

The following fields are available if a frequency of **Custom** is chosen:

Field	Description
Interval between jobs (Minutes)	Specify the number of minutes between iterations of the job. For example, to run a job every 30 minutes set this value to 30. If a job is to run only once but on selected days set this value to 1440 (daily)
Start at or after HHMM	The time that the job may run from. To run anytime set to 0000.
Do not start after	If multiple iterations are being done then this is the time after which a new iteration will not

Field	Description
HHMM	be started. For example, if a job is running every 10 minutes it will continue until this time is reached. To run till the end of day set to 2400.
Active on the days	Select each day of the week that the custom job is to be active on.

Editing Tasks in a Job

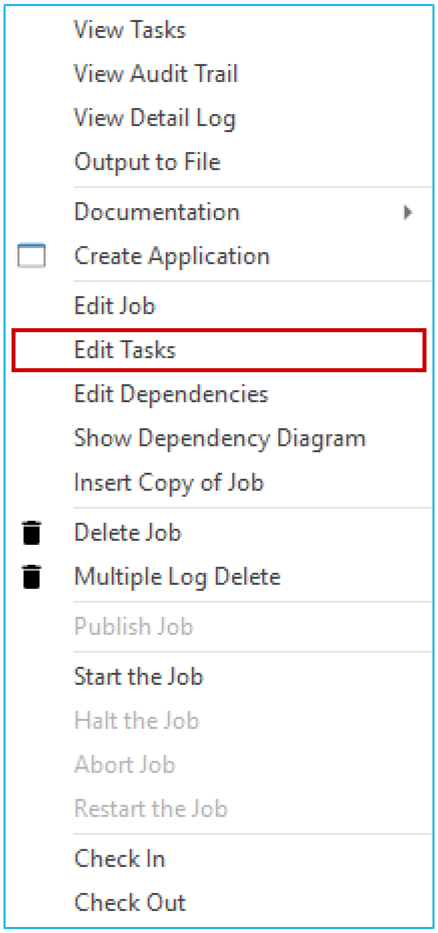
Once jobs have been created, you can edit their tasks.

Note

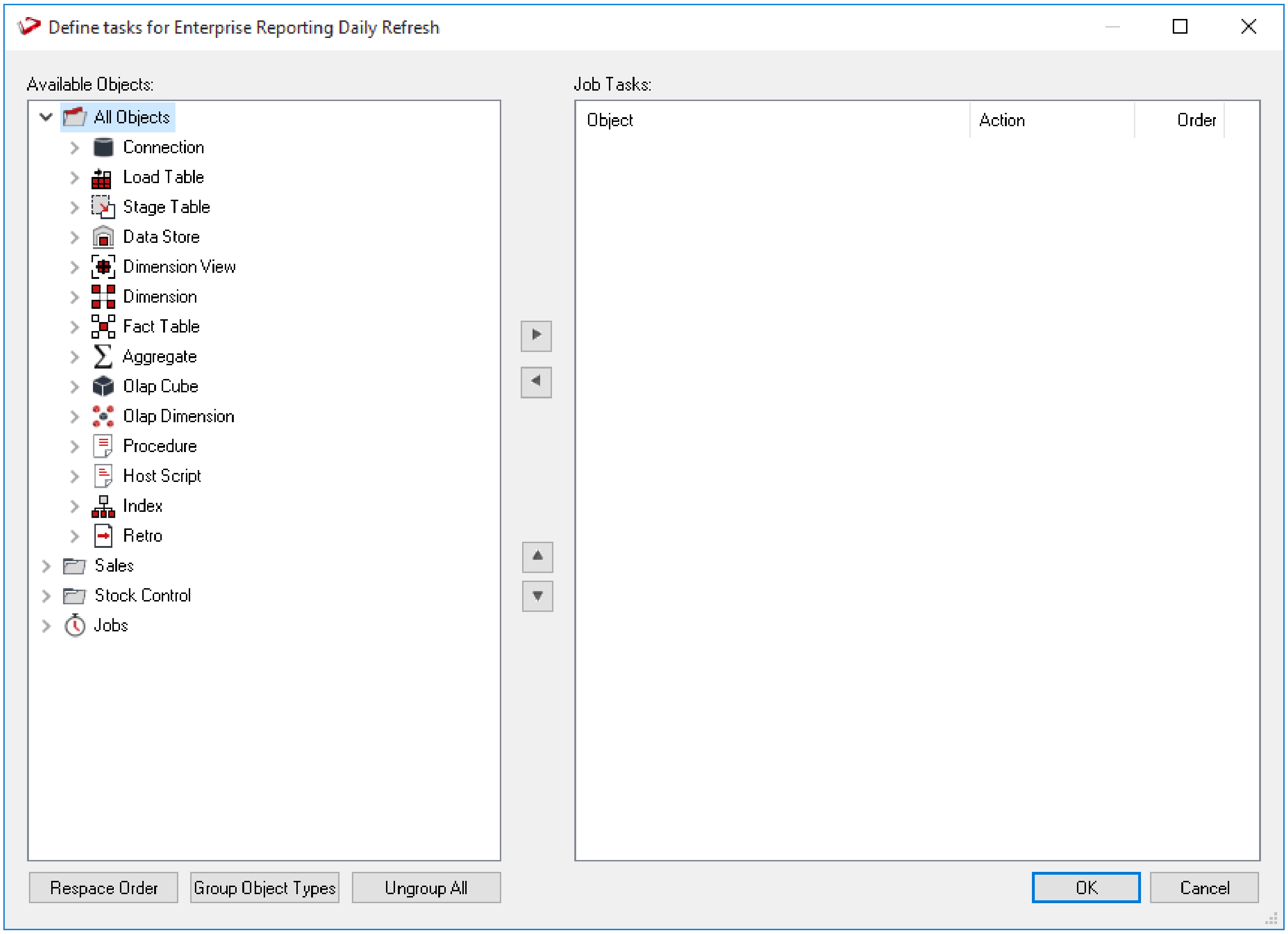
- A job can only be edited when it is not in a running state and only if the job is a scheduled job. Completed jobs remain in the list but only logs remain.
- **JOB TASK LIMIT -** There is maximum number of 999 tasks that can be added to a job.

To edit the tasks of a job

Select the job from the middle pane and then right-click the job to select **Edit Tasks** from the context menu.



The **Define tasks** window is displayed.



The screen has two main areas. The right pane shows the tasks to be run for this job and the left pane lists all the objects.

Adding a task

Double-click an object in the left pane to add it to the task list in the right pane. Normally objects, such as Load or Fact tables are scheduled rather than procedures.

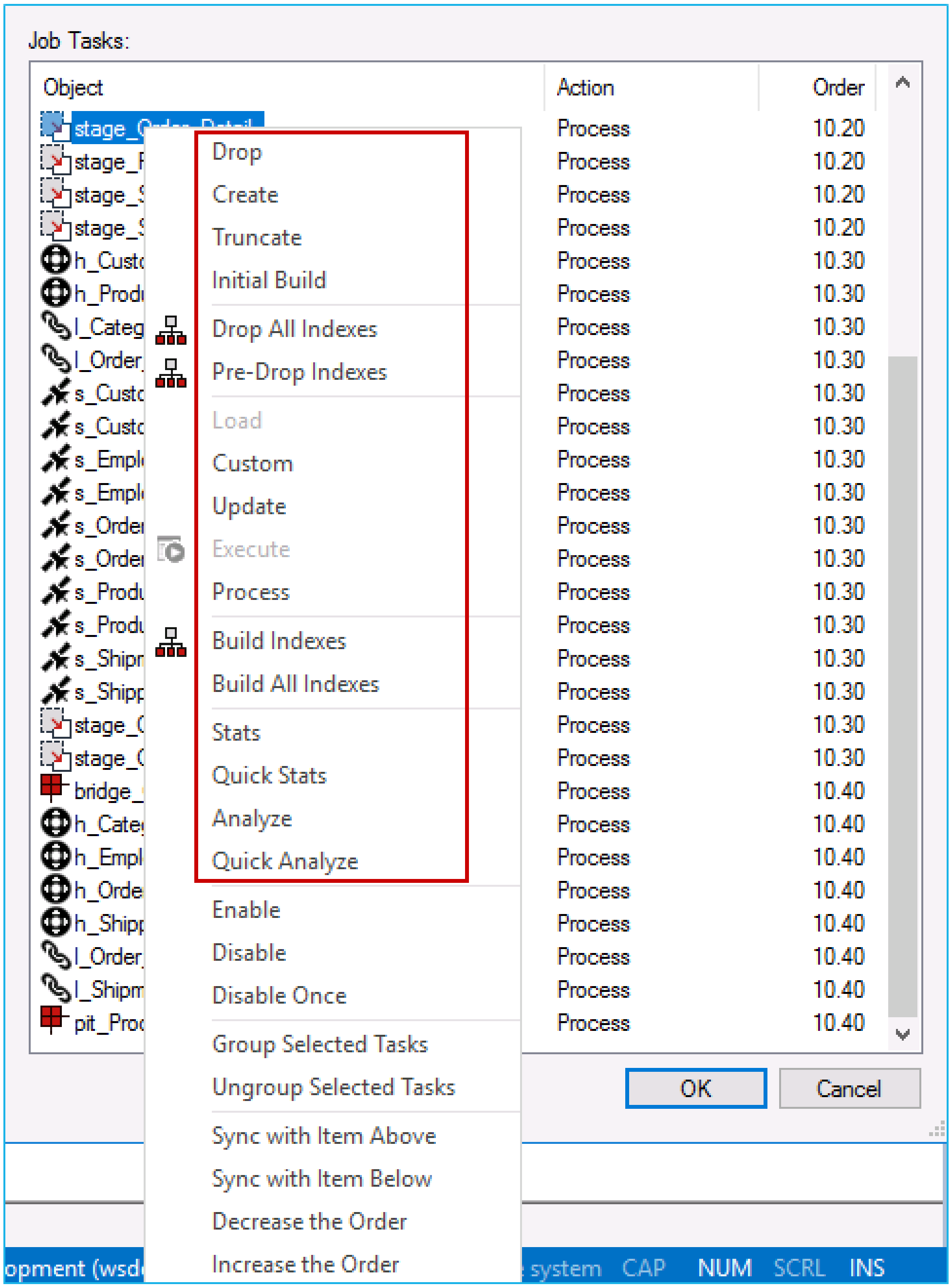
Setting the action on a task

Each task can have a specific action that is to be performed on its object.

The default action for **load tables** is **process**. This means that when the task is actioned, it will drop any indexes that are due to be dropped, or have **pre-drop** set, then load the table and perform any post-load procedures or transformations and then re-create any dropped indexes.

The default action for all other tables is the same as above, except it will execute the **update** procedure rather than loading the table.

You can change the action on a task by right-clicking the task in the right pane. The menu options are shown below.


Job Tasks:

Object	Action	Order
stage_Order_Detail	Process	10.20
stage_F	Process	10.20
stage_S	Process	10.20
stage_S	Process	10.20
h_Custo	Process	10.30
h_Prod	Process	10.30
l_Categ	Process	10.30
l_Order	Process	10.30
s_Custo	Process	10.30
s_Custo	Process	10.30
s_Empl	Process	10.30
s_Empl	Process	10.30
s_Order	Process	10.30
s_Order	Process	10.30
s_Produ	Process	10.30
s_Produ	Process	10.30
s_Shipr	Process	10.30
s_Shipp	Process	10.30
stage_(	Process	10.30
stage_(	Process	10.30
bridge_	Process	10.40
h_Cate	Process	10.40
h_Empl	Process	10.40
h_Orde	Process	10.40
h_Shipp	Process	10.40
l_Order	Process	10.40
l_Shipm	Process	10.40
pit_Proc	Process	10.40

The following task actions are available:

Action	Description
Drop	Drop table, view or index.
Create	Create table, view or index.
Truncate	Delete all rows from the table.
Initial Build	**Drop All Indexes** then **Custom** then **Build All Indexes**.
Drop All Indexes	Drop all indexes on the table.
Pre Drop Indexes	Drop all indexes on the table marked as "Pre Drop".
Load	Load the table (Load tables only).
Custom	Run the custom procedure on the table.

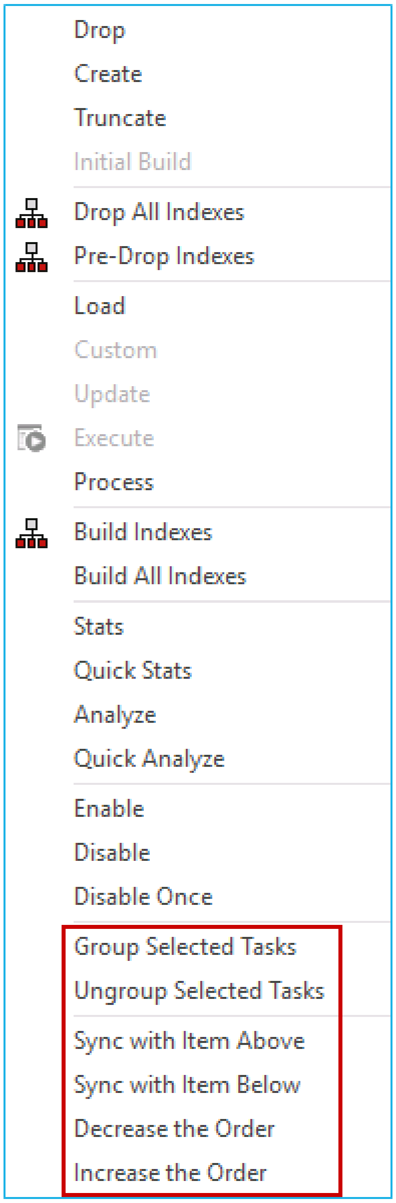
Action	Description
Update	Run the update procedure on the table.
Execute	Execute the procedure or host script.
Process	**Pre Drop Indexes** then **Update** and then **Build Indexes**.
Process and Statistics	**Process** then Default Stats as defined in the Table Properties > Statistics > Process and statistics method (DB2 only).
Build Indexes	Build the indexes on the table marked as "Pre Drop".
Build All Indexes	Build all indexes on the table.
SQL Server: Analyze	Performs: • **UPDATE STATISTICS name WITH FULLSCAN**
SQL Server: Quick Analyze	Performs: • **UPDATE STATISTICS name WITH SAMPLE 3 PERCENT**
SQL Server: Stats	Same as Analyze.
SQL Server: Quick Stats	Same as Quick Analyze.

Note

Not all actions are available on all object types.

Creating dependencies between tasks

You can create dependencies between tasks in the list by selecting one or more tasks and right-clicking to bring up the dependency options.



The following task dependency options are available from the menu:

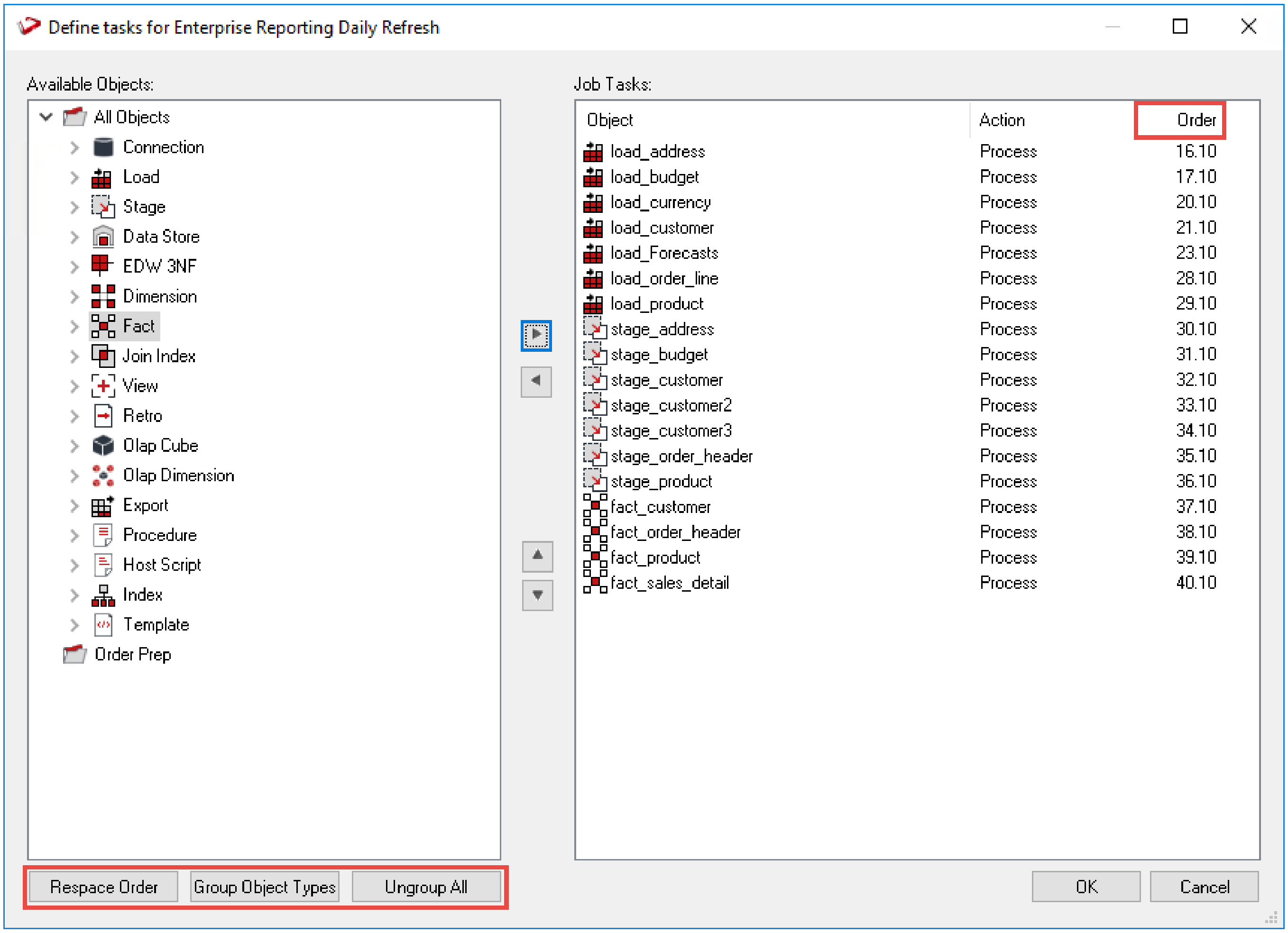
Task Option	Description
Group Selected Tasks	Groups two or more selected tasks to have the same order value, allowing them to run in parallel if the maximum threads setting allows.
Ungroup Selected Tasks	Un-group selected tasks.
Sync with Item Above	Changes a selected task to have the same order value as the task above it, allowing them to run in parallel if the maximum threads setting allows.
Sync with Item Below	Changes a selected task to have the same order value as the task below it, allowing them to run in parallel if the maximum threads setting allows.
Decrease the Order	Changes a selected task to an order number one less than its current value. The task will now run immediately before it would have previously.
Increase the Order	Changes a selected task to an order number one more than its current value. The task will now run immediately after it would have previously.

Ordering or Grouping the tasks

The **Order** column shows the order in which the tasks are to be run, e.g. 20.20 If the two numbers are the same as another task then those tasks can run in parallel. If the two numbers are different then those tasks run sequentially. This is an initial definition of dependencies. These dependencies can be altered specifically once the job has been created.

Tasks can be moved up or down by selecting the task and clicking the **Move Up** ▲ or **Move Down** ▼ buttons.

To respace the order of the tasks; to group or ungroup object types, use the **buttons** at the bottom of the **Define tasks** window.



Object	Action	Order
load_address	Process	16.10
load_budget	Process	17.10
load_currency	Process	20.10
load_customer	Process	21.10
load_Forecasts	Process	23.10
load_order_line	Process	28.10
load_product	Process	29.10
stage_address	Process	30.10
stage_budget	Process	31.10
stage_customer	Process	32.10
stage_customer2	Process	33.10
stage_customer3	Process	34.10
stage_order_header	Process	35.10
stage_product	Process	36.10
fact_customer	Process	37.10
fact_order_header	Process	38.10
fact_product	Process	39.10
fact_sales_detail	Process	40.10

Button	Description
Respace Order	Click to respace the order numbers. The existing dependency structure and groupings are retained. The purpose of this button is simply to allow room between tasks to fit new tasks. For example, if we have two tasks that have an order of 20.19.5 and 20.20.6 and we want to add a task between these two tasks, we can click the **Respace Order** button to open up a gap between the two tasks.
Group Object Types	Click to put all objects of the same type into groups. For example, all load tables will be able to run in parallel, all dimensions, etc.
Ungroup All	Click to remove all groupings and make all tasks sequential. New groupings can be made by selecting a range of sequentially listed tasks in the left pane and using the right-click menu option **Group Selected Tasks**. Tasks that are grouped have the same first two numbers in the order and can execute at the same time if the job has multiple threads.

Upon completion of adding tasks, click **OK** to exit.

Editing Task Dependencies

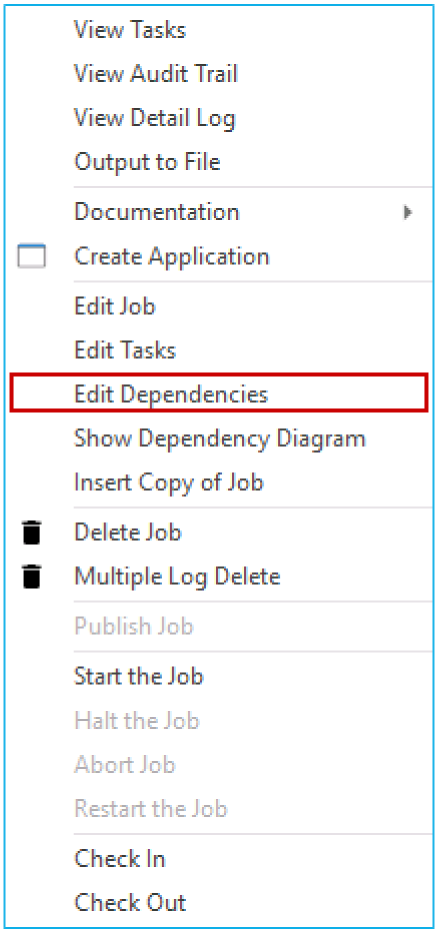
Once jobs have been created they can be edited.

Note

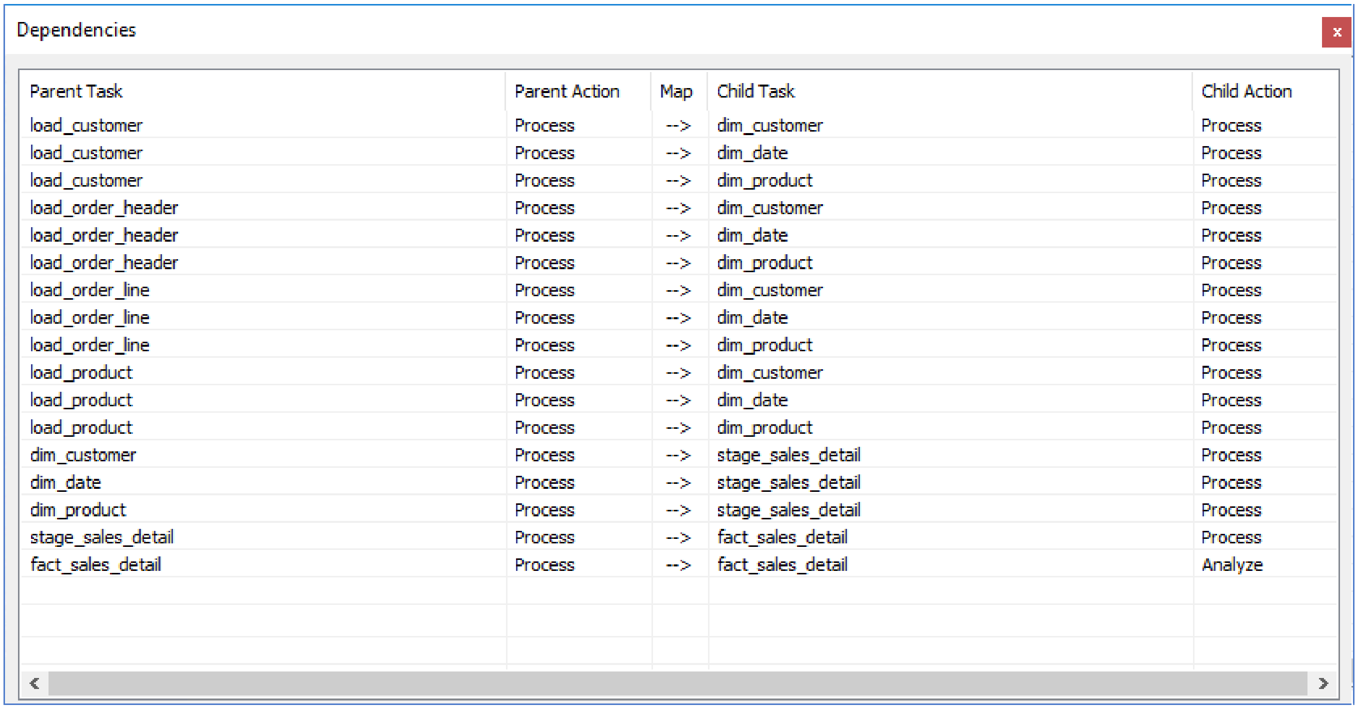
A job can only be edited when it is not in a running state and only if the job is a scheduled job. Completed jobs remain in the list but only logs remain.

To edit task dependencies

Select the job from the middle pane and then right-click to select **Edit Dependencies** from the context menu.



The **Dependencies** window is displayed, showing the dependencies between the tasks of the job. The list consists of **Parent Tasks** on the left and **Child Tasks** on the right. A child task is thus dependent on its parent task in that it cannot run until its parent has run.

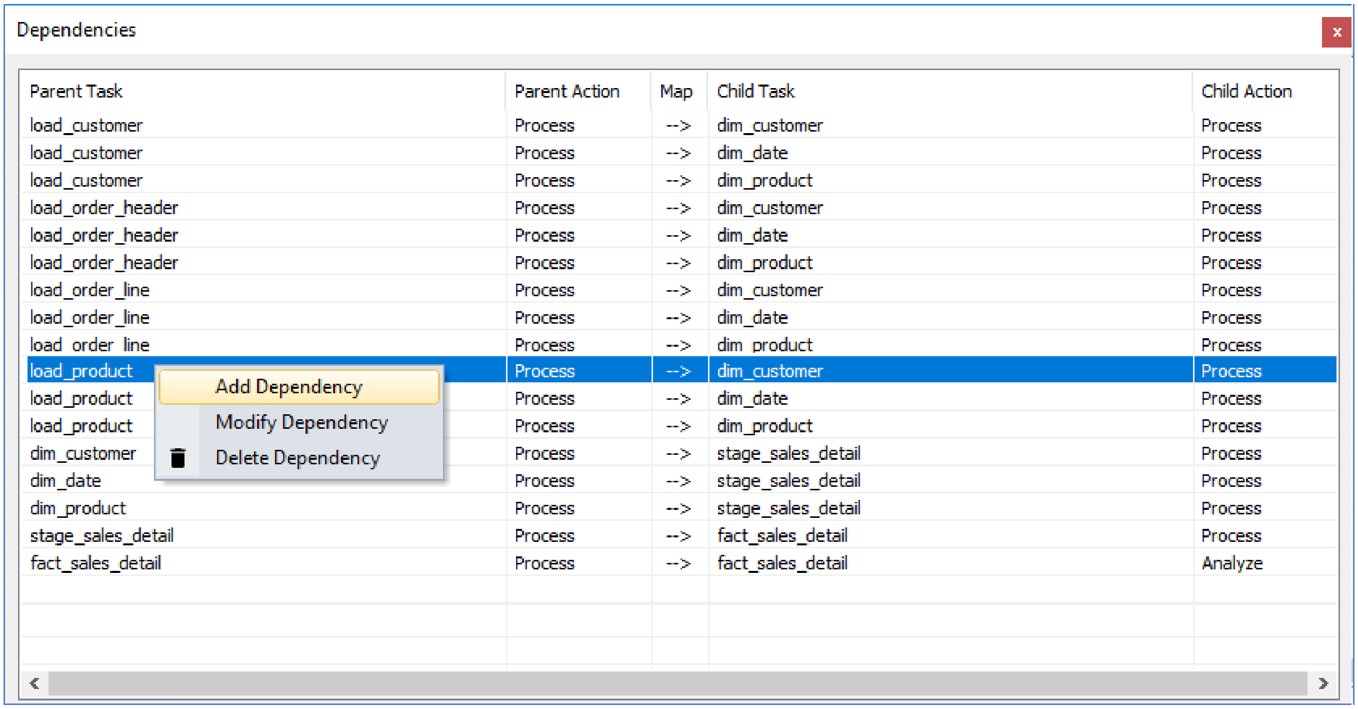


Parent Task	Parent Action	Map	Child Task	Child Action
load_customer	Process	-->	dim_customer	Process
load_customer	Process	-->	dim_date	Process
load_customer	Process	-->	dim_product	Process
load_order_header	Process	-->	dim_customer	Process
load_order_header	Process	-->	dim_date	Process
load_order_header	Process	-->	dim_product	Process
load_order_line	Process	-->	dim_customer	Process
load_order_line	Process	-->	dim_date	Process
load_order_line	Process	-->	dim_product	Process
load_product	Process	-->	dim_customer	Process
load_product	Process	-->	dim_date	Process
load_product	Process	-->	dim_product	Process
dim_customer	Process	-->	stage_sales_detail	Process
dim_date	Process	-->	stage_sales_detail	Process
dim_product	Process	-->	stage_sales_detail	Process
stage_sales_detail	Process	-->	fact_sales_detail	Process
fact_sales_detail	Process	-->	fact_sales_detail	Analyze

Edit the dependencies and close the window.

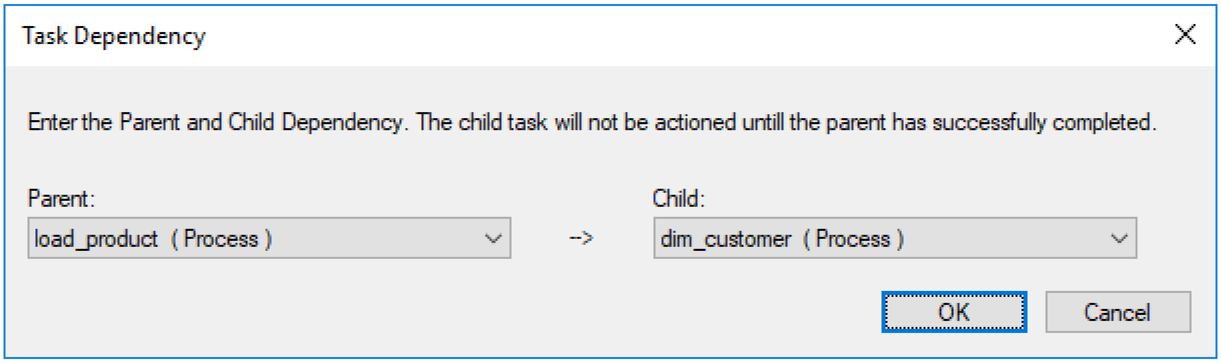
To add a task dependency

To add a task dependency, right-click anywhere in the **Dependencies** window and select **Add Dependency**.



Parent Task	Parent Action	Map	Child Task	Child Action
load_customer	Process	-->	dim_customer	Process
load_customer	Process	-->	dim_date	Process
load_customer	Process	-->	dim_product	Process
load_order_header	Process	-->	dim_customer	Process
load_order_header	Process	-->	dim_date	Process
load_order_header	Process	-->	dim_product	Process
load_order_line	Process	-->	dim_customer	Process
load_order_line	Process	-->	dim_date	Process
load_order_line	Process	-->	dim_product	Process
load_product	Process	-->	dim_customer	Process
load_product	Process	-->	dim_date	Process
load_product	Process	-->	dim_product	Process
dim_customer	Process	-->	stage_sales_detail	Process
dim_date	Process	-->	stage_sales_detail	Process
dim_product	Process	-->	stage_sales_detail	Process
stage_sales_detail	Process	-->	fact_sales_detail	Process
fact_sales_detail	Process	-->	fact_sales_detail	Analyze

Select the **Parent** and **Child** tasks from the drop-down lists to create the dependency and then click **OK**.

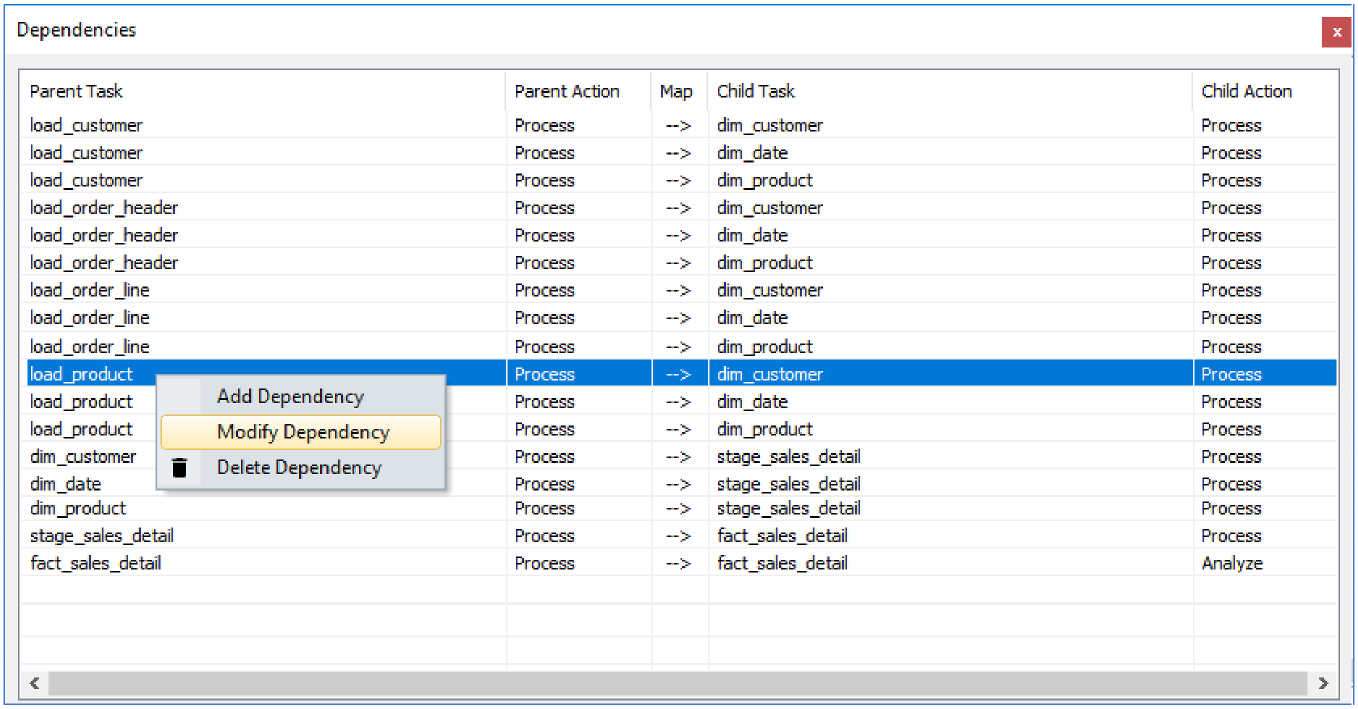


Enter the Parent and Child Dependency. The child task will not be actioned untill the parent has successfully completed.

Parent: load_product (Process) --> Child: dim_customer (Process)

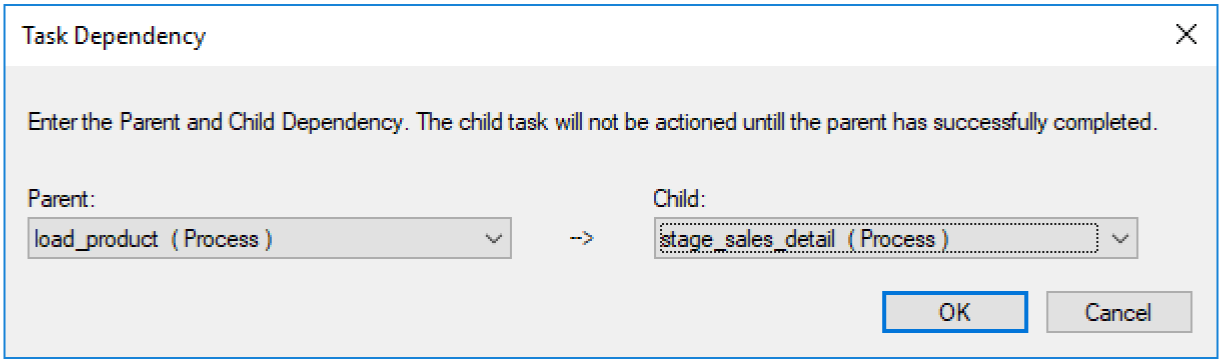
To modify a task dependency

To modify a task dependency, right-click the dependency in the **Dependencies** window and select **Modify Dependency**.



Parent Task	Parent Action	Map	Child Task	Child Action
load_customer	Process	-->	dim_customer	Process
load_customer	Process	-->	dim_date	Process
load_customer	Process	-->	dim_product	Process
load_order_header	Process	-->	dim_customer	Process
load_order_header	Process	-->	dim_date	Process
load_order_header	Process	-->	dim_product	Process
load_order_line	Process	-->	dim_customer	Process
load_order_line	Process	-->	dim_date	Process
load_order_line	Process	-->	dim_product	Process
load_product	Process	-->	dim_customer	Process
load_product	Process	-->	dim_date	Process
load_product	Process	-->	dim_product	Process
dim_customer	Process	-->	stage_sales_detail	Process
dim_date	Process	-->	stage_sales_detail	Process
dim_product	Process	-->	stage_sales_detail	Process
stage_sales_detail	Process	-->	fact_sales_detail	Process
fact_sales_detail	Process	-->	fact_sales_detail	Analyze

Change the **Parent** and **Child** tasks to modify the dependency and then click **OK**.

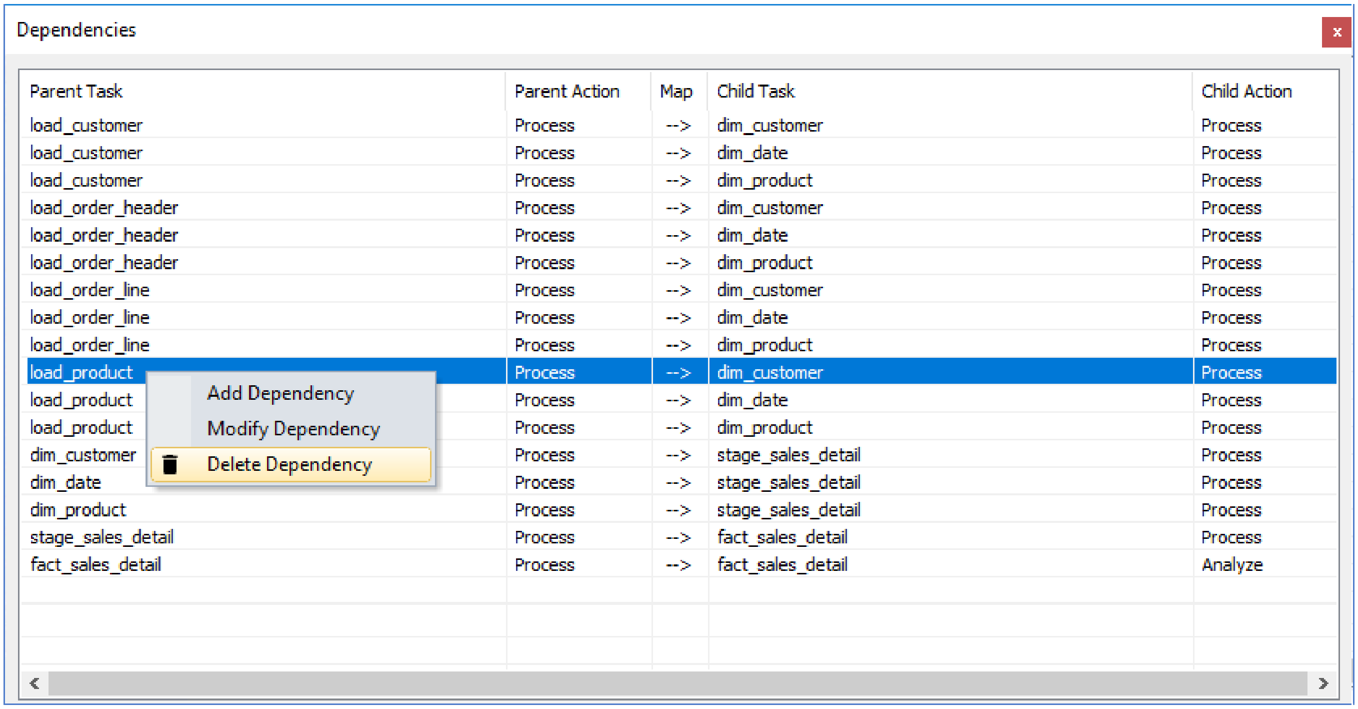


Enter the Parent and Child Dependency. The child task will not be actioned untill the parent has successfully completed.

Parent: load_product (Process) --> Child: stage_sales_detail (Process)

To delete a task dependency

To delete a task dependency, right-click the dependency in the **Dependencies** window and select **Delete Dependency**.

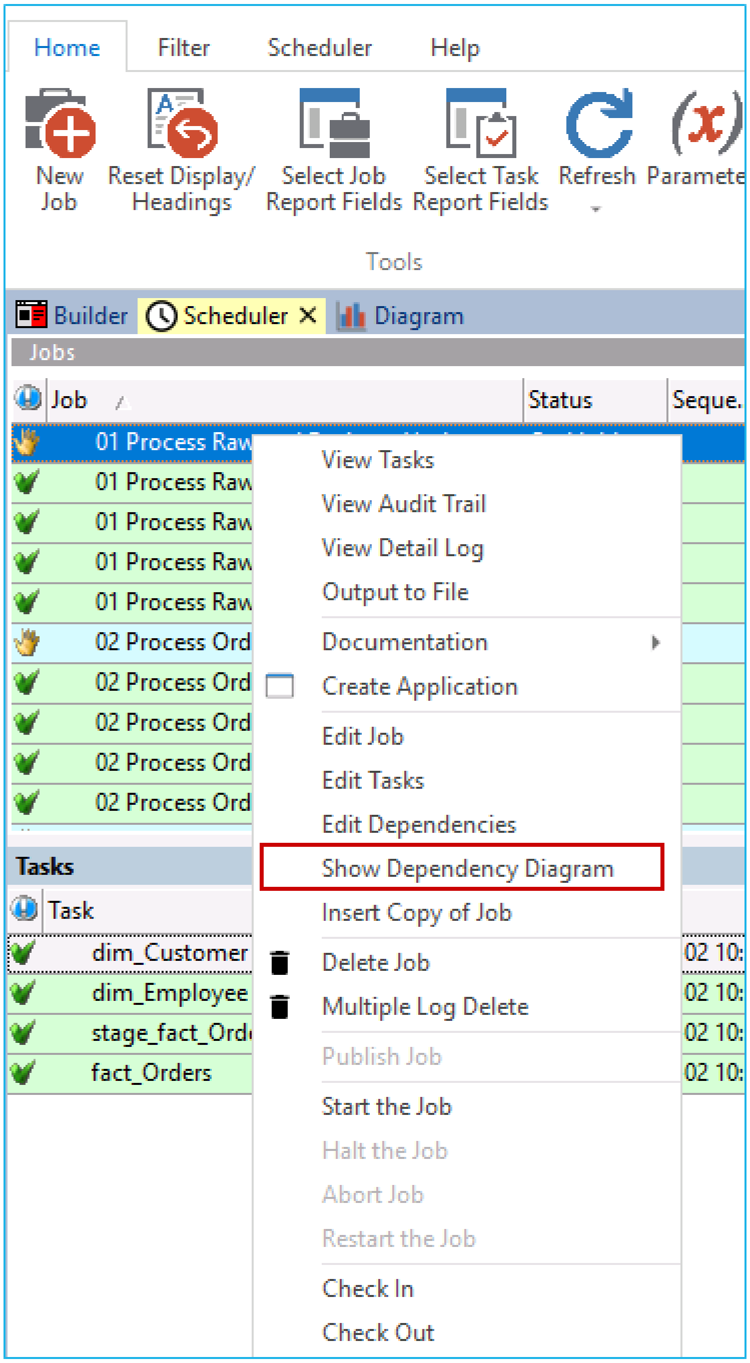


Parent Task	Parent Action	Map	Child Task	Child Action
load_customer	Process	-->	dim_customer	Process
load_customer	Process	-->	dim_date	Process
load_customer	Process	-->	dim_product	Process
load_order_header	Process	-->	dim_customer	Process
load_order_header	Process	-->	dim_date	Process
load_order_header	Process	-->	dim_product	Process
load_order_line	Process	-->	dim_customer	Process
load_order_line	Process	-->	dim_date	Process
load_order_line	Process	-->	dim_product	Process
load_product	Process	-->	dim_customer	Process
load_product	Process	-->	dim_date	Process
load_product	Process	-->	dim_product	Process
dim_customer	Process	-->	stage_sales_detail	Process
dim_date	Process	-->	stage_sales_detail	Process
dim_product	Process	-->	stage_sales_detail	Process
stage_sales_detail	Process	-->	fact_sales_detail	Process
fact_sales_detail	Process	-->	fact_sales_detail	Analyze

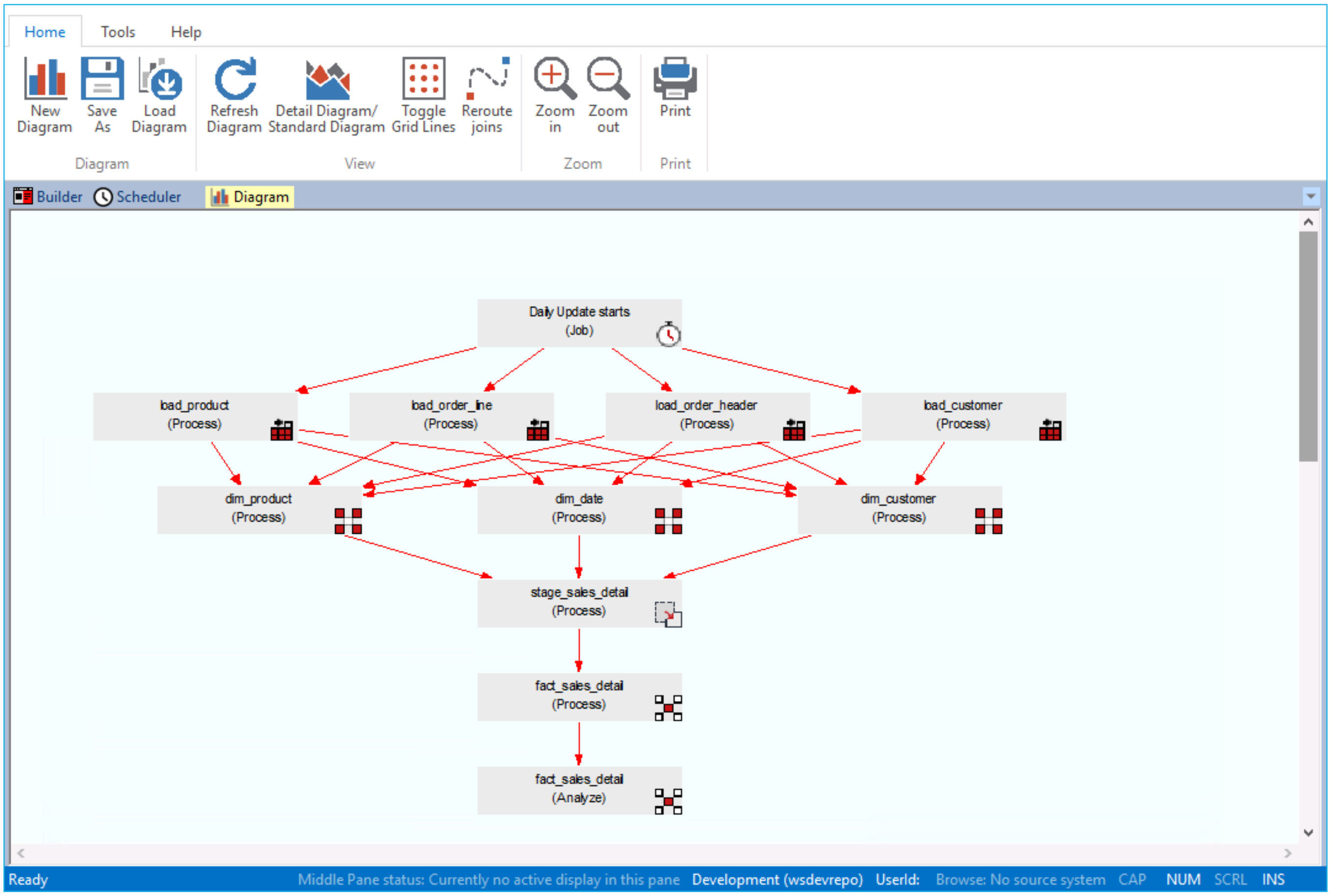
The dependency is deleted without warning.

Show Dependencies Diagram

Select the **Show Dependency Diagram** option from the right-click menu of any job, to see all job dependencies displayed as a Diagram from RED's **Diagram** view tab.

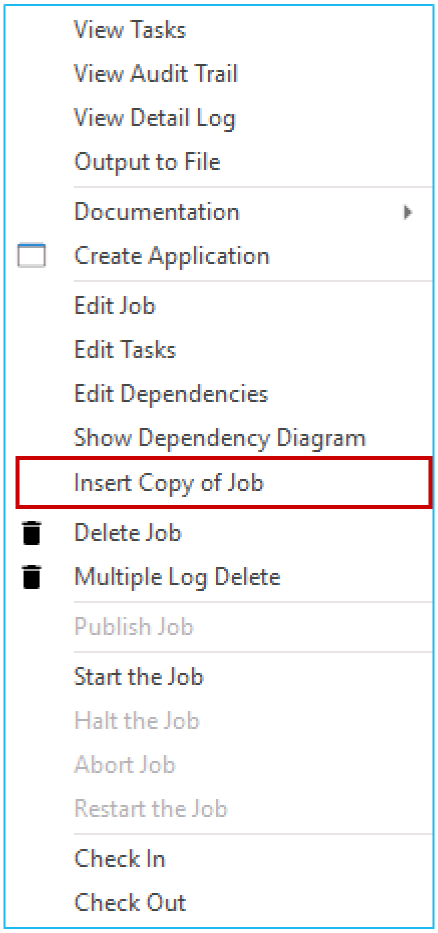


Job Dependency Diagram view

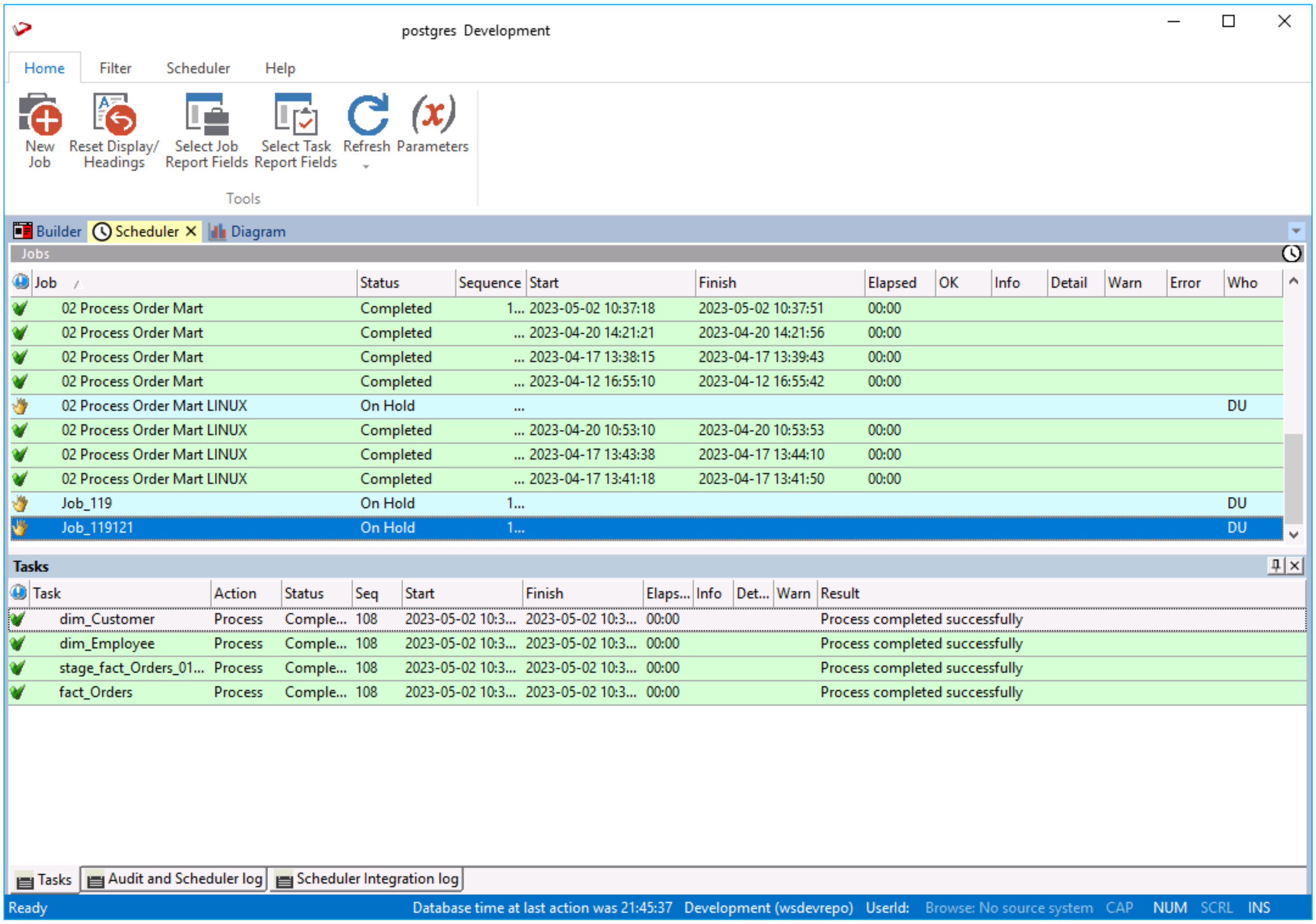


Inserting a Copy of a Job

A copy of a job can be inserted by right-clicking the job and choosing **Insert Copy of Job** from the context menu.



The new job is immediately visible and the Status is **On Hold**.



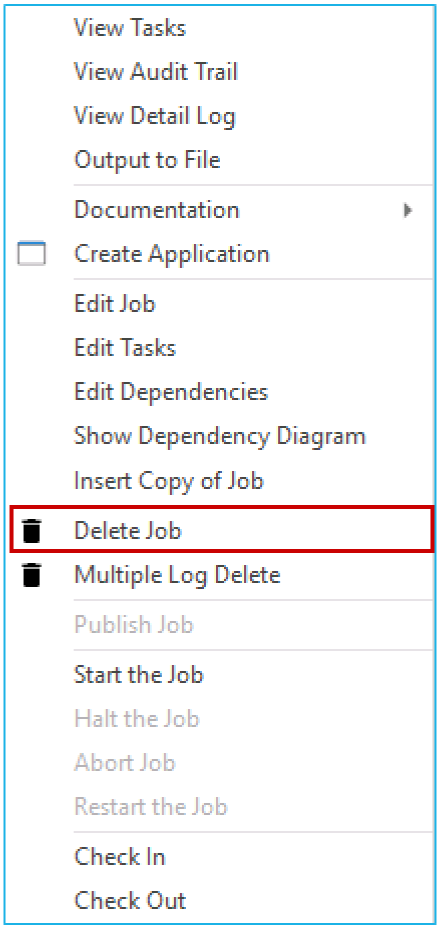
	Job	Status	Sequence	Start	Finish	Elapsed	OK	Info	Detail	Warn	Error	Who
✓	02 Process Order Mart	Completed	1...	2023-05-02 10:37:18	2023-05-02 10:37:51	00:00						
✓	02 Process Order Mart	Completed	...	2023-04-20 14:21:21	2023-04-20 14:21:56	00:00						
✓	02 Process Order Mart	Completed	...	2023-04-17 13:38:15	2023-04-17 13:39:43	00:00						
✓	02 Process Order Mart	Completed	...	2023-04-12 16:55:10	2023-04-12 16:55:42	00:00						
	02 Process Order Mart LINUX	On Hold	...									DU
✓	02 Process Order Mart LINUX	Completed	...	2023-04-20 10:53:10	2023-04-20 10:53:53	00:00						
✓	02 Process Order Mart LINUX	Completed	...	2023-04-17 13:43:38	2023-04-17 13:44:10	00:00						
✓	02 Process Order Mart LINUX	Completed	...	2023-04-17 13:41:18	2023-04-17 13:41:50	00:00						
	Job_119	On Hold	1...									DU
	Job_119121	On Hold	1...									DU

	Task	Action	Status	Seq	Start	Finish	Elaps...	Info	Det...	Warn	Result
✓	dim_Customer	Process	Comple...	108	2023-05-02 10:3...	2023-05-02 10:3...	00:00				Process completed successfully
✓	dim_Employee	Process	Comple...	108	2023-05-02 10:3...	2023-05-02 10:3...	00:00				Process completed successfully
✓	stage_fact_Orders_01...	Process	Comple...	108	2023-05-02 10:3...	2023-05-02 10:3...	00:00				Process completed successfully
✓	fact_Orders	Process	Comple...	108	2023-05-02 10:3...	2023-05-02 10:3...	00:00				Process completed successfully

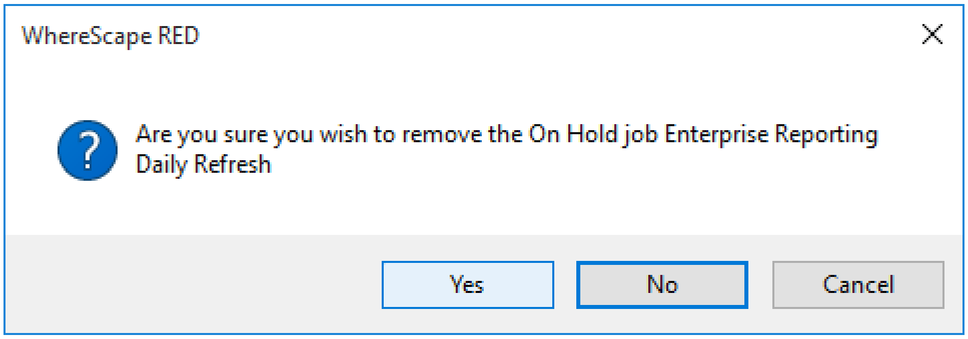
Ready Database time at last action was 21:45:37 Development (wsdevrepo) UserId: Browse: No source system CAP NUM SCRL INS

Deleting a Job

A job can be deleted by right-clicking the job in the Scheduler window and then choosing **Delete Job** from the context menu.



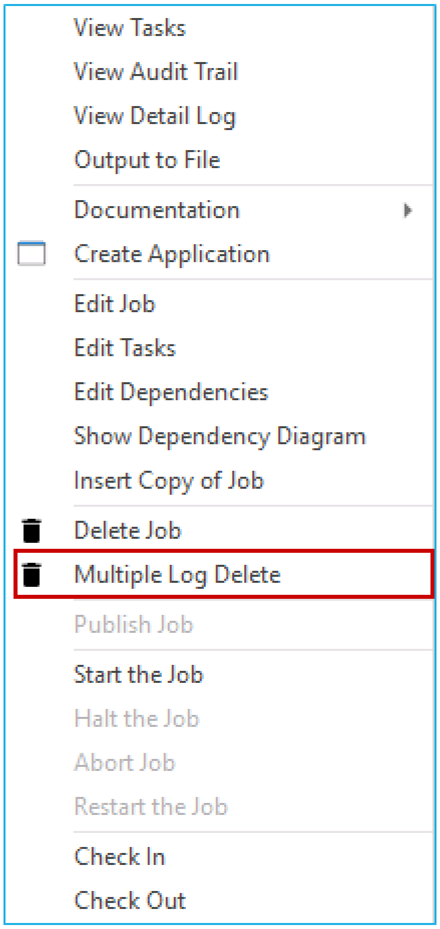
A confirmation prompt is displayed; click **Yes** to delete.


WhereScape RED

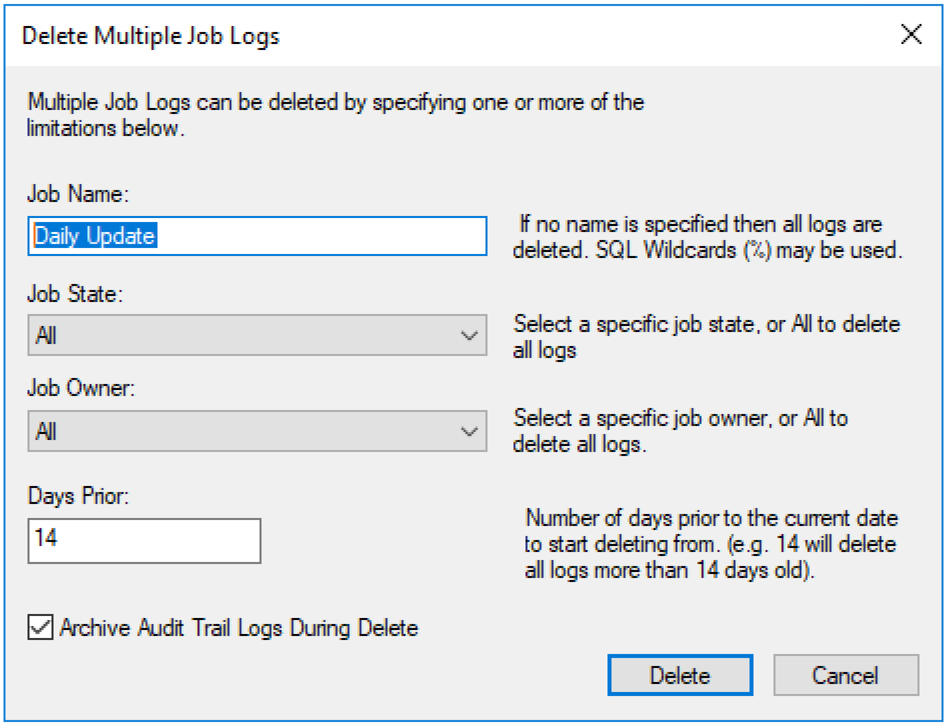
Are you sure you wish to remove the On Hold job Enterprise Reporting Daily Refresh

Deleting Job Logs

Multiple job logs can be deleted by right-clicking a job in the Scheduler window and then choosing **Multiple Log Delete** from the context menu.



The **Delete Multiple Job Logs** window is displayed. Select or enter the appropriate options to delete the range of job logs required.



Delete Multiple Job Logs

Multiple Job Logs can be deleted by specifying one or more of the limitations below.

Job Name: Daily Update

If no name is specified then all logs are deleted. SQL Wildcards (%) may be used.

Job State: All

Select a specific job state, or All to delete all logs

Job Owner: All

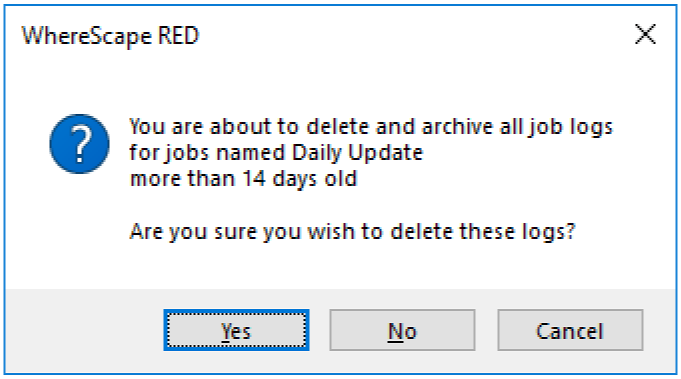
Select a specific job owner, or All to delete all logs.

Days Prior: 14

Number of days prior to the current date to start deleting from. (e.g. 14 will delete all logs more than 14 days old).

☑ Archive Audit Trail Logs During Delete

A confirmation prompt is displayed. Click **Yes** to delete.



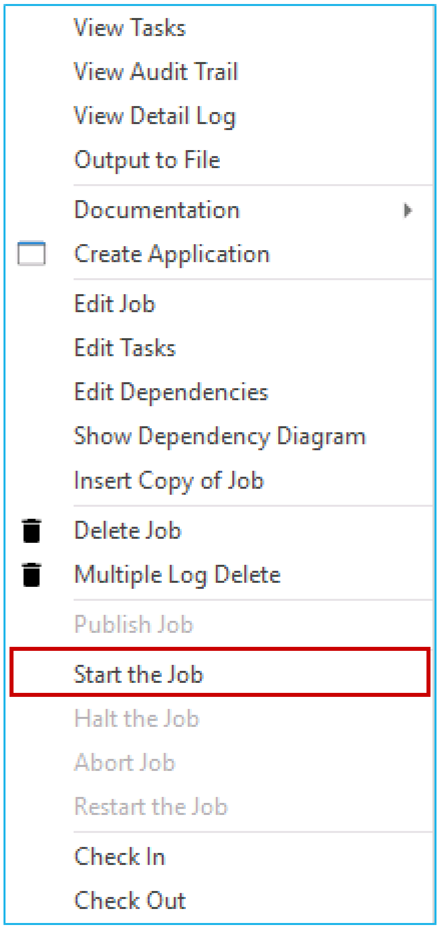
WhereScape RED

You are about to delete and archive all job logs
for jobs named Daily Update
more than 14 days old

Are you sure you wish to delete these logs?

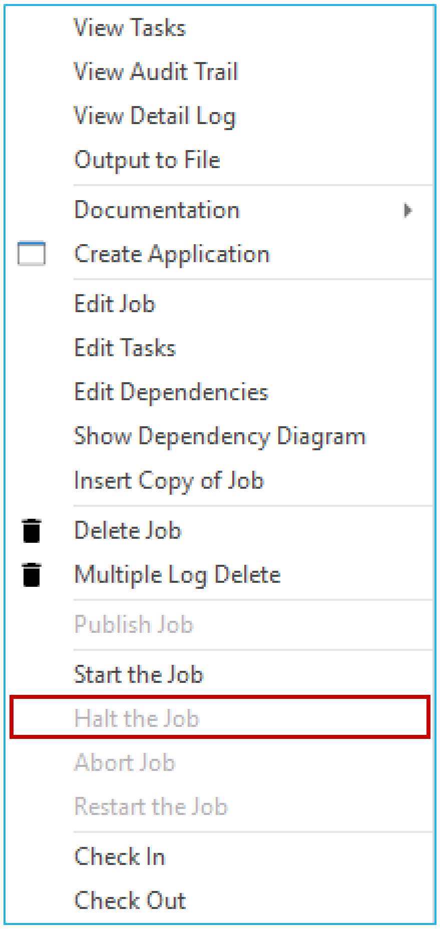
Starting a Job

A job can be started by right-clicking the job in the Scheduler window and then choosing **Start the Job** from the context menu.



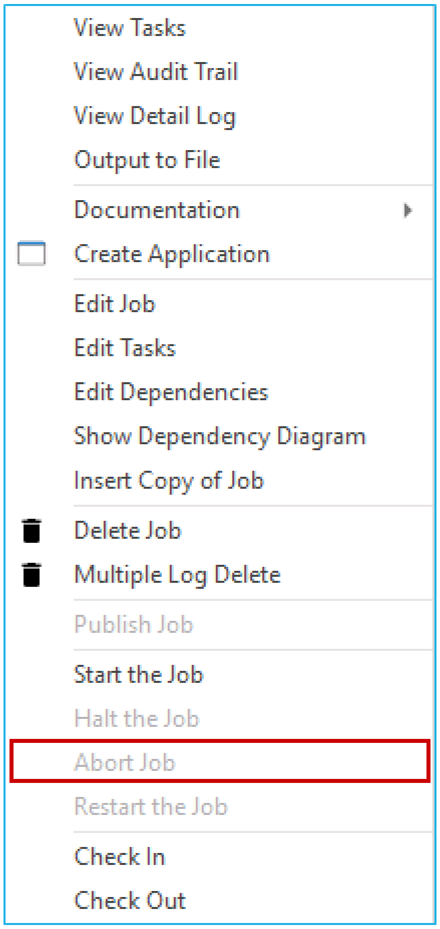
Halting a Job

A job can be halted by right-clicking the job in the Scheduler window and then choosing **Halt the Job** from the context menu.



Aborting a Job

A job can be aborted by right-clicking the job in the Scheduler window and choosing **Abort Job** from the context menu.



Once in this state, a job cannot be restarted. The job now only exists as a log of what occurred and is no longer regarded as a job.

Effects of Aborting a Job

The effects of aborting a job depend on the Operating System the Scheduler is running on, the target platform and the action being run.

Note

Rolling back the current transaction may take a considerable amount of time.

Windows Scheduler

Greenplum (target)

Load and update processes are not stopped for all objects.

Hive (target)

Load and update processes are not stopped for all objects.

Netezza (target)

Load and update processes are not stopped for all objects.

PDW (target)

Object	Action	Effect
Load	Native Load, SSIS Load	Does not stop process
Load	ODBC Load, Database Link Load, File/Script	Stops process

Object	Action	Effect
	Windows Load	
All objects using Procedures (Local targets)	Update	Stops process and rolls back current transaction
All objects using Procedures (remote targets)	Update	Does not stop process
Cube	Update	Does not stop process

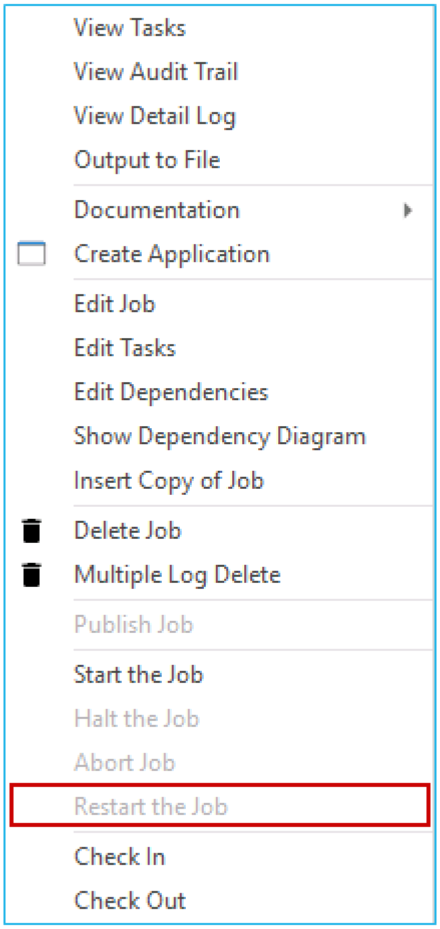
Unix Scheduler

Hive (target)

File/Hadoop Load processes are not stopped on Hive target platforms.

Restarting a Job

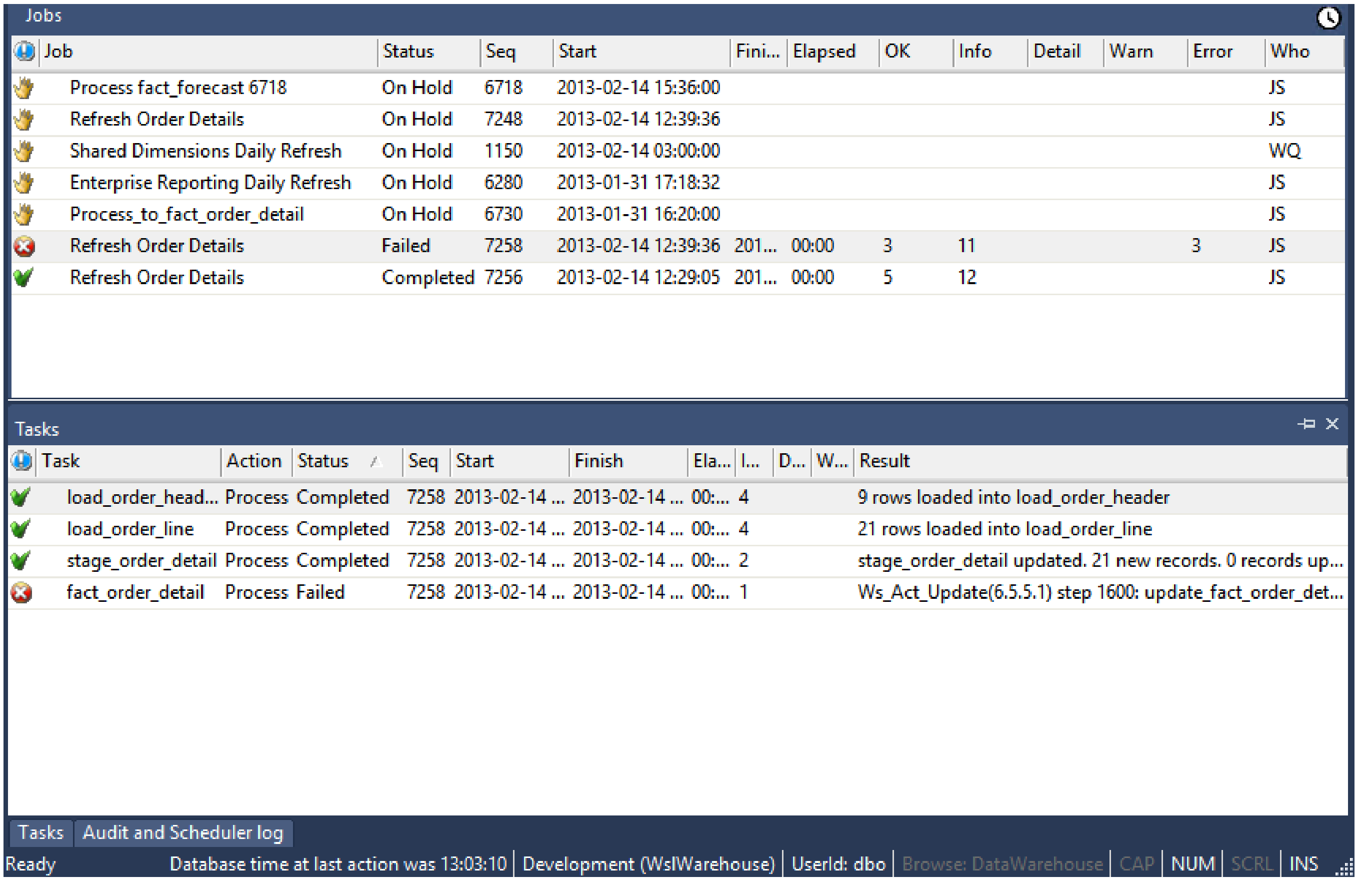
A job can be restarted by right-clicking the job in the Scheduler window and choosing **Restart the Job** from the context menu.



Before restarting a job, it is possible to edit the **status** of the job tasks so that only certain tasks will be **run again** or be **skipped over**.

To run a task again

View the job tasks by double-clicking on the failed job. The tasks will be displayed in the bottom pane.

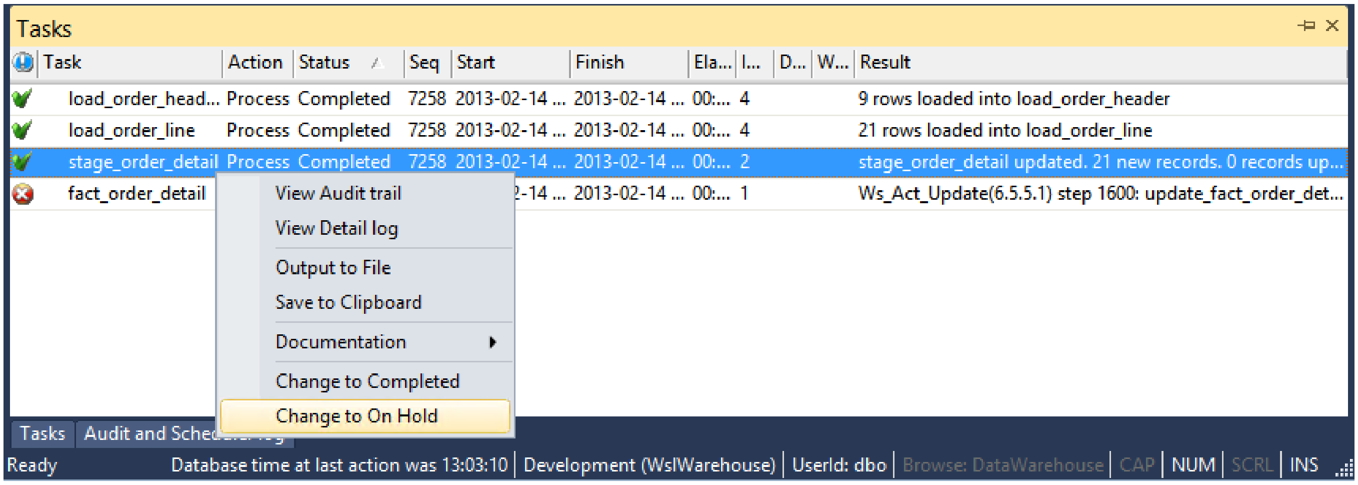


Job	Status	Seq	Start	Fini...	Elapsed	OK	Info	Detail	Warn	Error	Who
Process fact_forecast 6718	On Hold	6718	2013-02-14 15:36:00								JS
Refresh Order Details	On Hold	7248	2013-02-14 12:39:36								JS
Shared Dimensions Daily Refresh	On Hold	1150	2013-02-14 03:00:00								WQ
Enterprise Reporting Daily Refresh	On Hold	6280	2013-01-31 17:18:32								JS
Process_to_fact_order_detail	On Hold	6730	2013-01-31 16:20:00								JS
Refresh Order Details	Failed	7258	2013-02-14 12:39:36	201...	00:00	3	11			3	JS
Refresh Order Details	Completed	7256	2013-02-14 12:29:05	201...	00:00	5	12				JS

Task	Action	Status	Seq	Start	Finish	Ela...	I...	D...	W...	Result
load_order_head...	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	4			9 rows loaded into load_order_header
load_order_line	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	4			21 rows loaded into load_order_line
stage_order_detail	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	2			stage_order_detail updated. 21 new records. 0 records up...
fact_order_detail	Process	Failed	7258	2013-02-14 ...	2013-02-14 ...	00:...	1			Ws_Act_Update(6.5.5.1) step 1600: update_fact_order_det...

Ready | Database time at last action was 13:03:10 | Development (WslWarehouse) | UserId: dbo | Browse: DataWarehouse | CAP | NUM | SCRL | INS

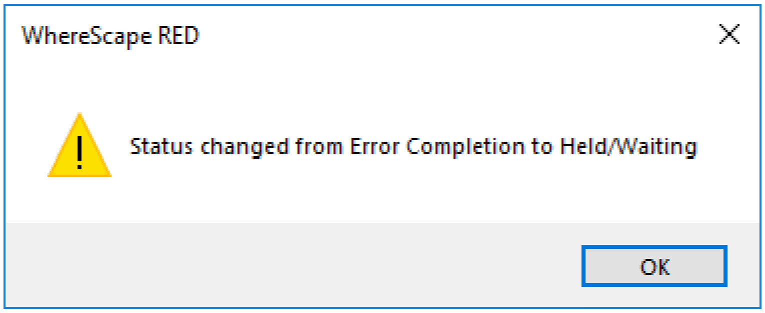
To rerun a task, right-click the completed task and select **Change to On Hold**



Task	Action	Status	Seq	Start	Finish	Ela...	I...	D...	W...	Result
load_order_head...	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	4			9 rows loaded into load_order_header
load_order_line	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	4			21 rows loaded into load_order_line
stage_order_detail	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	2			stage_order_detail updated. 21 new records. 0 records up...
fact_order_detail				-14 ...	2013-02-14 ...	00:...	1			Ws_Act_Update(6.5.5.1) step 1600: update_fact_order_det...

Ready | Database time at last action was 13:03:10 | Development (WslWarehouse) | UserId: dbo | Browse: DataWarehouse | CAP | NUM | SCRL | INS

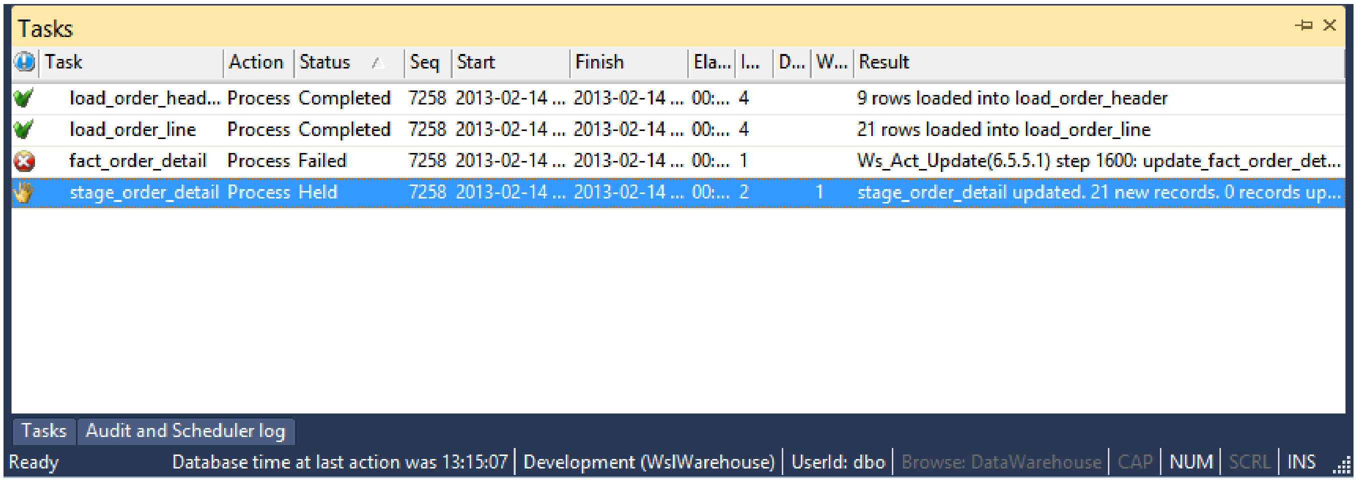
Click **OK** on the message dialog.



WhereScape RED

Status changed from Error Completion to Held/Waiting

Double-click on the job again to display the tasks. You will see that the selected task now has a status of **Held** and will thus be rerun when you restart the job.



Tasks

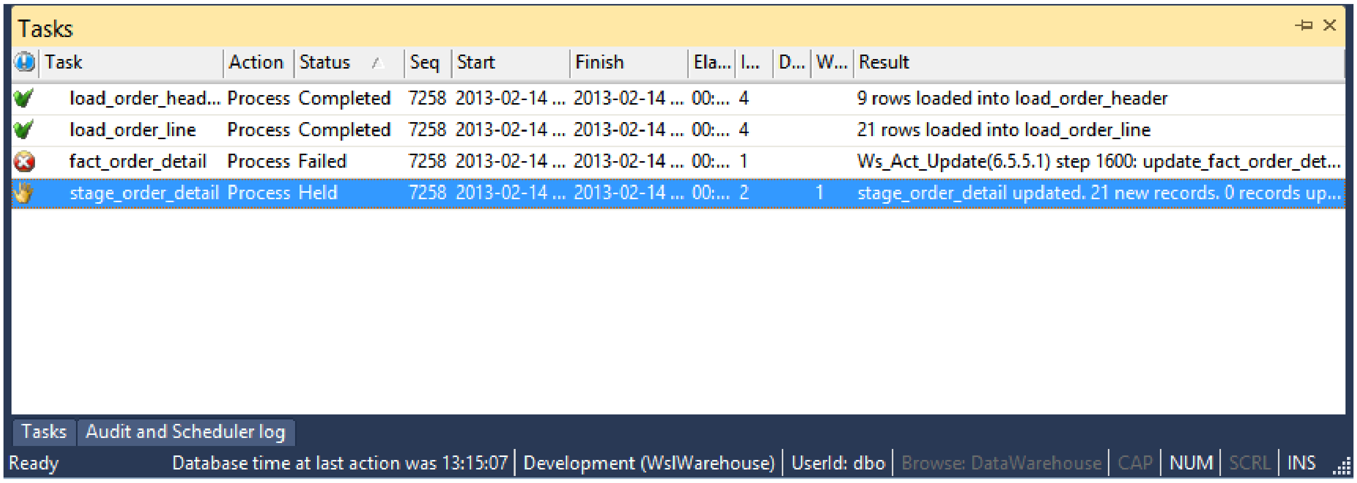
	Task	Action	Status	Seq	Start	Finish	Ela...	I...	D...	W...	Result
✓	load_order_head...	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	4			9 rows loaded into load_order_header
✓	load_order_line	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	4			21 rows loaded into load_order_line
✗	fact_order_detail	Process	Failed	7258	2013-02-14 ...	2013-02-14 ...	00:...	1			Ws_Act_Update(6.5.5.1) step 1600: update_fact_order_det...
✋	stage_order_detail	Process	Held	7258	2013-02-14 ...	2013-02-14 ...	00:...	2		1	stage_order_detail updated. 21 new records. 0 records up...

Tasks | Audit and Scheduler log

Ready | Database time at last action was 13:15:07 | Development (WslWarehouse) | UserId: dbo | Browse: DataWarehouse | CAP | NUM | SCRL | INS

To skip over a task

View the job tasks by double-clicking on the failed job. The tasks will be displayed in the bottom pane.



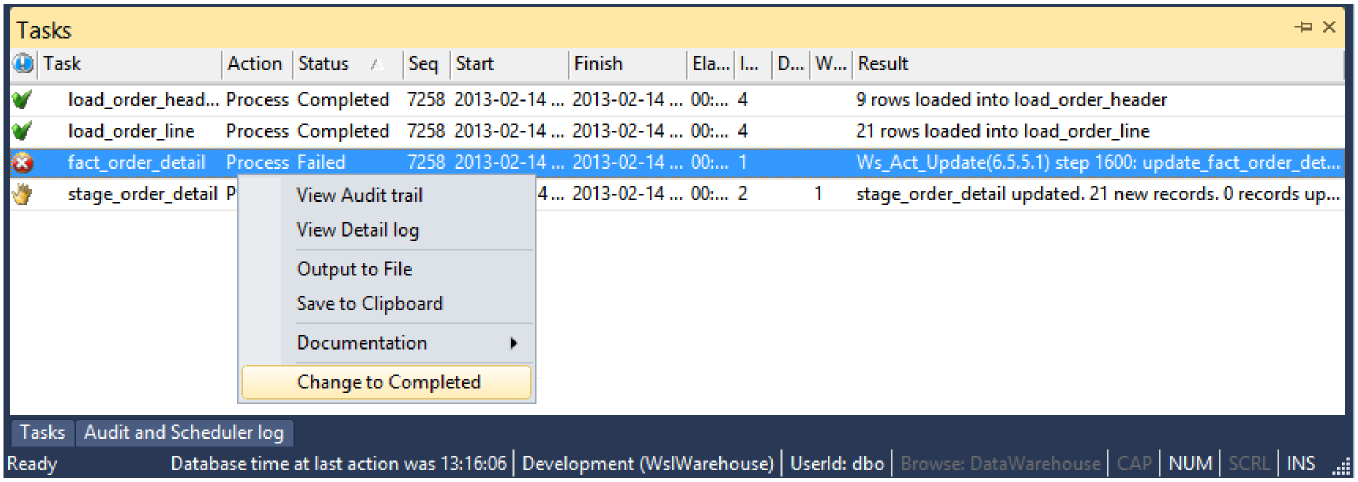
Tasks

	Task	Action	Status	Seq	Start	Finish	Ela...	I...	D...	W...	Result
✓	load_order_head...	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	4			9 rows loaded into load_order_header
✓	load_order_line	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	4			21 rows loaded into load_order_line
✗	fact_order_detail	Process	Failed	7258	2013-02-14 ...	2013-02-14 ...	00:...	1			Ws_Act_Update(6.5.5.1) step 1600: update_fact_order_det...
✋	stage_order_detail	Process	Held	7258	2013-02-14 ...	2013-02-14 ...	00:...	2		1	stage_order_detail updated. 21 new records. 0 records up...

Tasks | Audit and Scheduler log

Ready | Database time at last action was 13:15:07 | Development (WslWarehouse) | UserId: dbo | Browse: DataWarehouse | CAP | NUM | SCRL | INS

To skip over a task, right-click the task and select **Change to Completed**.



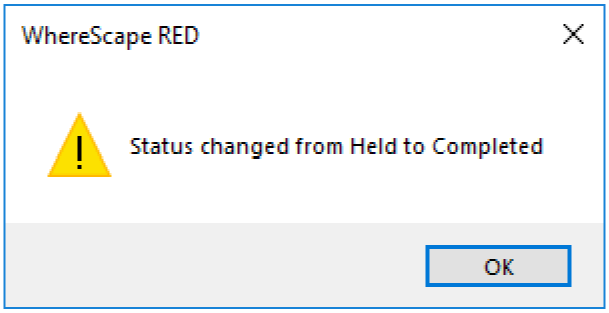
Tasks

	Task	Action	Status	Seq	Start	Finish	Ela...	I...	D...	W...	Result
✓	load_order_head...	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	4			9 rows loaded into load_order_header
✓	load_order_line	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	4			21 rows loaded into load_order_line
✗	fact_order_detail	Process	Failed	7258	2013-02-14 ...	2013-02-14 ...	00:...	1			Ws_Act_Update(6.5.5.1) step 1600: update_fact_order_det...
✋	stage_order_detail	P			4...	2013-02-14 ...	00:...	2		1	stage_order_detail updated. 21 new records. 0 records up...

Tasks | Audit and Scheduler log

Ready | Database time at last action was 13:16:06 | Development (WslWarehouse) | UserId: dbo | Browse: DataWarehouse | CAP | NUM | SCRL | INS

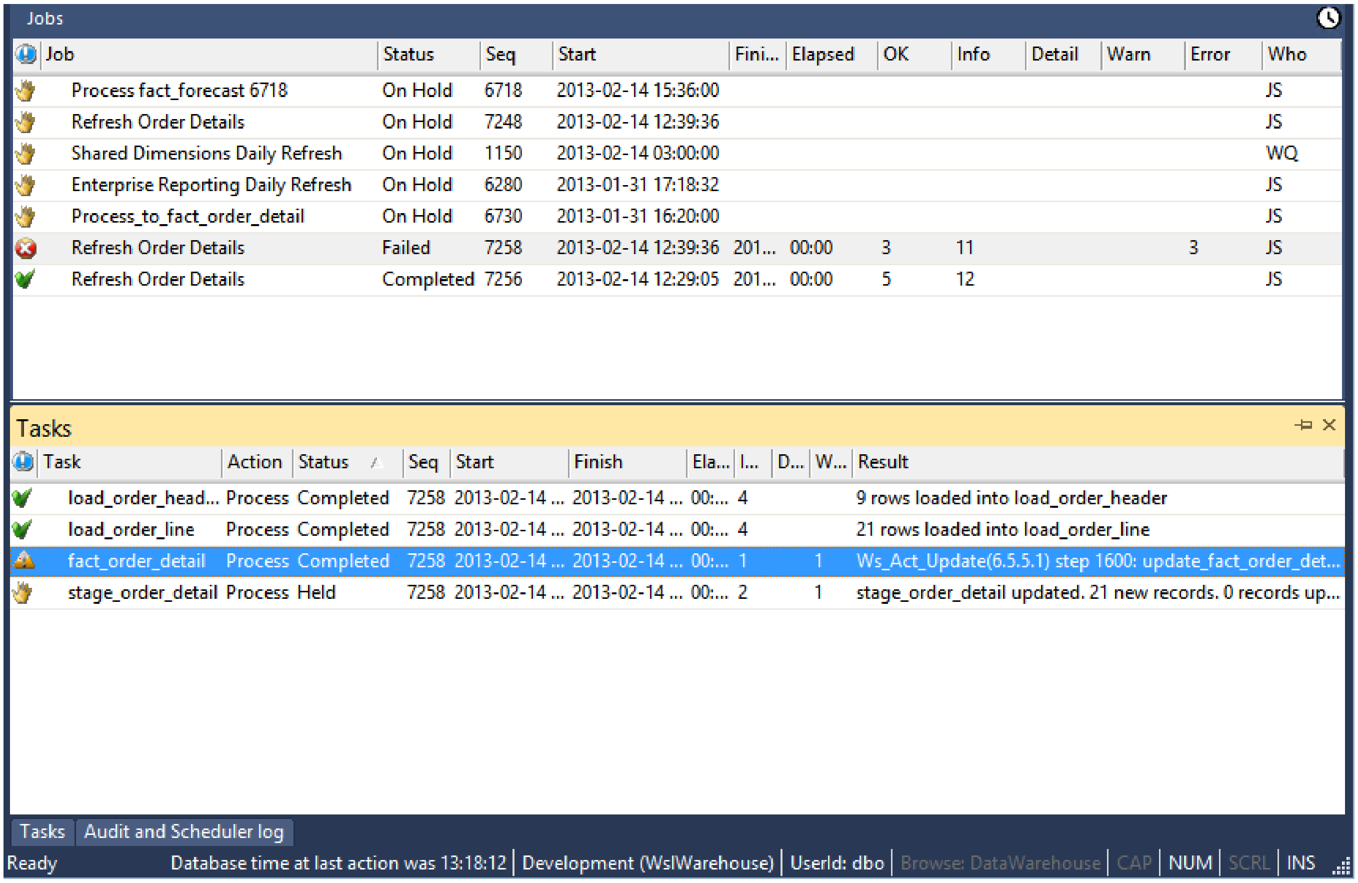
Click **OK** on the message dialog.



WhereScape RED

Status changed from Held to Completed

Double-click the job again to display the tasks. The selected task now has a status of **Completed** and will thus be skipped when you restart the job.



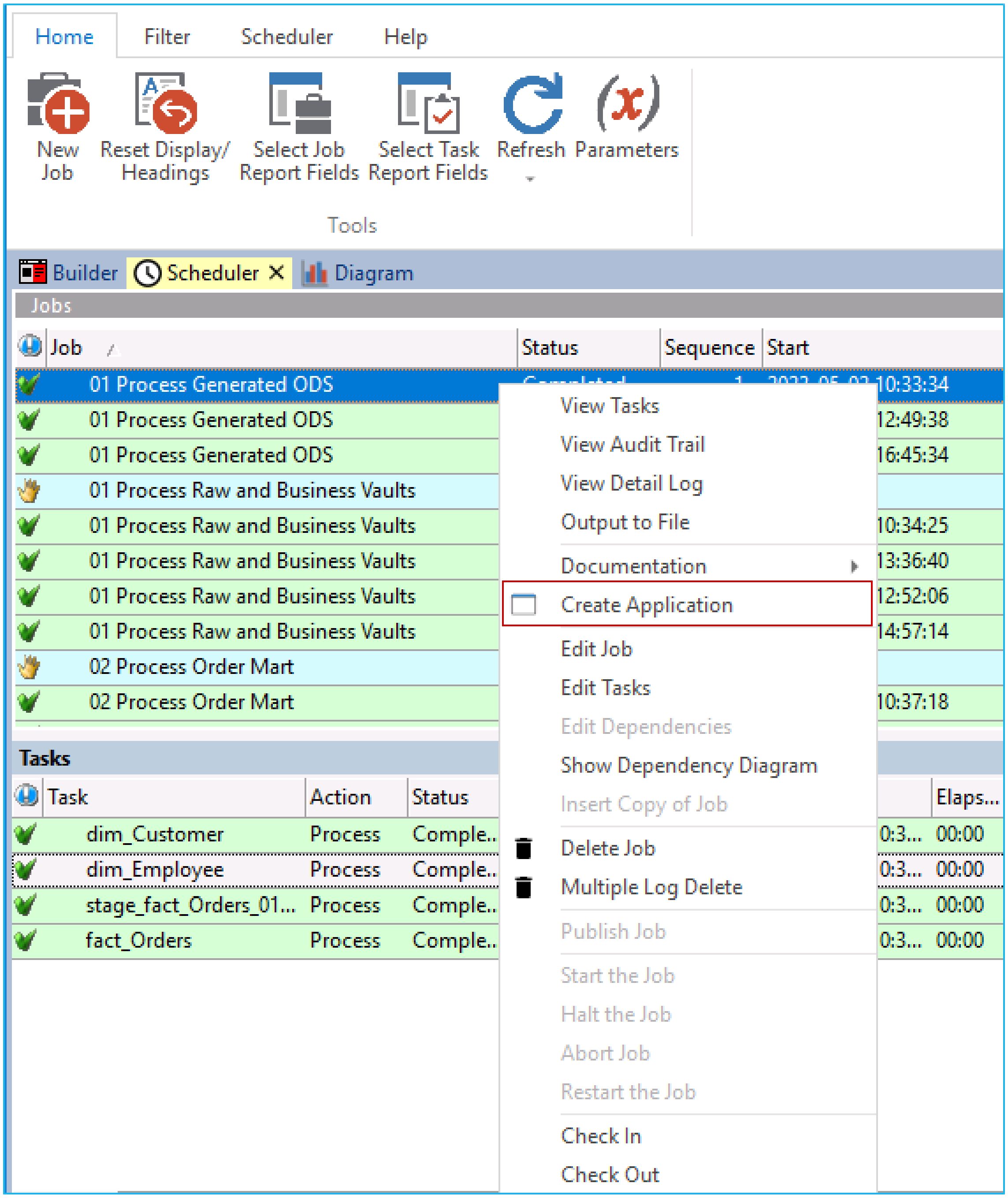
	Job	Status	Seq	Start	Fini...	Elapsed	OK	Info	Detail	Warn	Error	Who
✋	Process fact_forecast 6718	On Hold	6718	2013-02-14 15:36:00								JS
✋	Refresh Order Details	On Hold	7248	2013-02-14 12:39:36								JS
✋	Shared Dimensions Daily Refresh	On Hold	1150	2013-02-14 03:00:00								WQ
✋	Enterprise Reporting Daily Refresh	On Hold	6280	2013-01-31 17:18:32								JS
✋	Process_to_fact_order_detail	On Hold	6730	2013-01-31 16:20:00								JS
✗	Refresh Order Details	Failed	7258	2013-02-14 12:39:36	201...	00:00	3	11			3	JS
✓	Refresh Order Details	Completed	7256	2013-02-14 12:29:05	201...	00:00	5	12				JS

	Task	Action	Status	Seq	Start	Finish	Ela...	I...	D...	W...	Result
✓	load_order_head...	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	4			9 rows loaded into load_order_header
✓	load_order_line	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	4			21 rows loaded into load_order_line
⚠	fact_order_detail	Process	Completed	7258	2013-02-14 ...	2013-02-14 ...	00:...	1		1	Ws_Act_Update(6.5.5.1) step 1600: update_fact_order_det...
✋	stage_order_detail	Process	Held	7258	2013-02-14 ...	2013-02-14 ...	00:...	2		1	stage_order_detail updated. 21 new records. 0 records up...

Ready | Database time at last action was 13:18:12 | Development (WslWarehouse) | UserId: dbo | Browse: DataWarehouse | CAP | NUM | SCRL | INS

Creating an Application from a Job

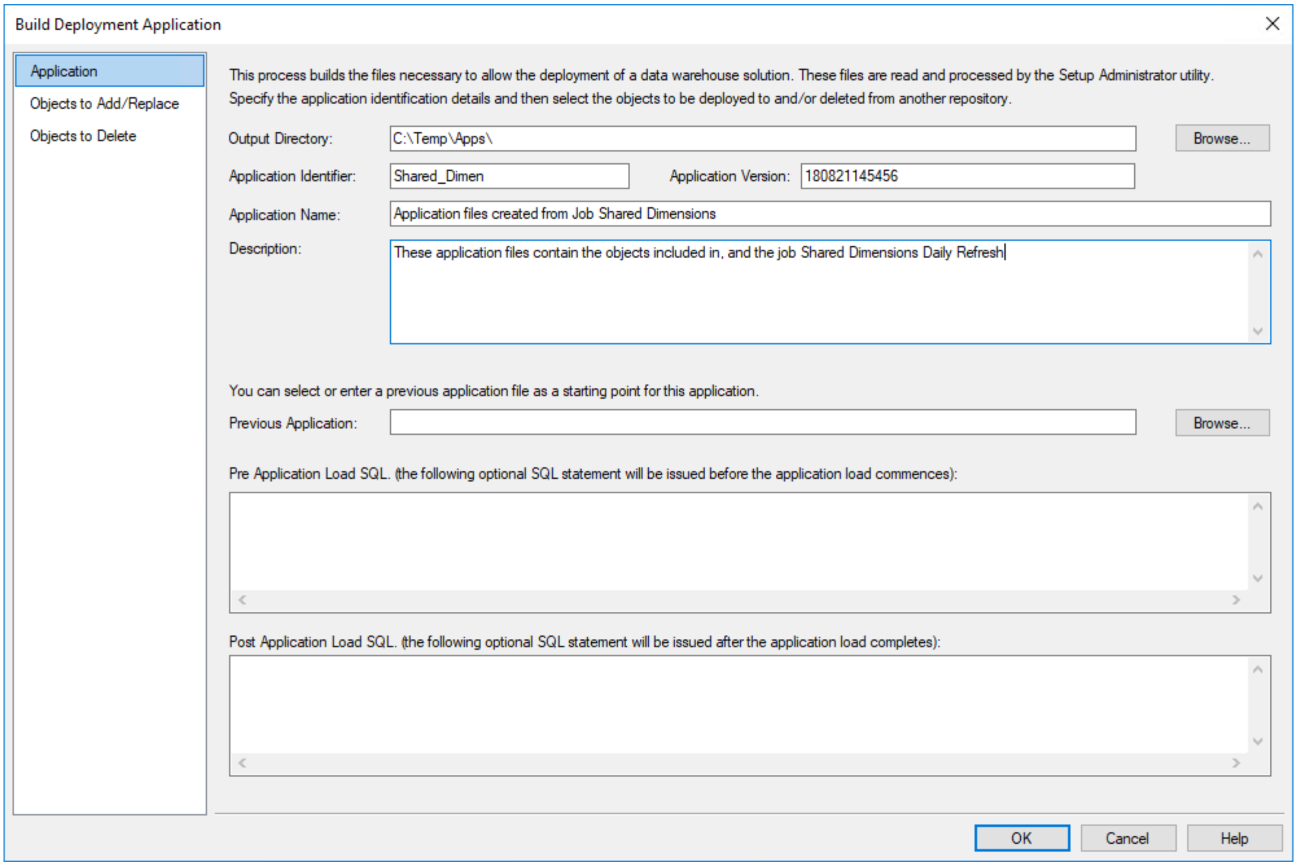
1. Right-click the job in the Scheduler window and then select **Create Application** from the context menu.



Job	Status	Sequence	Start
01 Process Generated ODS	Completed	1	2022-05-02 10:33:34
01 Process Generated ODS			12:49:38
01 Process Generated ODS			16:45:34
01 Process Raw and Business Vaults			
01 Process Raw and Business Vaults			10:34:25
01 Process Raw and Business Vaults			13:36:40
01 Process Raw and Business Vaults			12:52:06
01 Process Raw and Business Vaults			14:57:14
02 Process Order Mart			
02 Process Order Mart			10:37:18

Task	Action	Status		Elaps...
dim_Customer	Process	Comple..	0:3...	00:00
dim_Employee	Process	Comple..	0:3...	00:00
stage_fact_Orders_01...	Process	Comple..	0:3...	00:00
fact_Orders	Process	Comple..	0:3...	00:00

2. Edit the application details as required.



Build Deployment Application

This process builds the files necessary to allow the deployment of a data warehouse solution. These files are read and processed by the Setup Administrator utility. Specify the application identification details and then select the objects to be deployed to and/or deleted from another repository.

Output Directory: C:\Temp\Apps\

Application Identifier: Shared_Dimen

Application Version: 180821145456

Application Name: Application files created from Job Shared Dimensions

Description: These application files contain the objects included in, and the job Shared Dimensions Daily Refresh

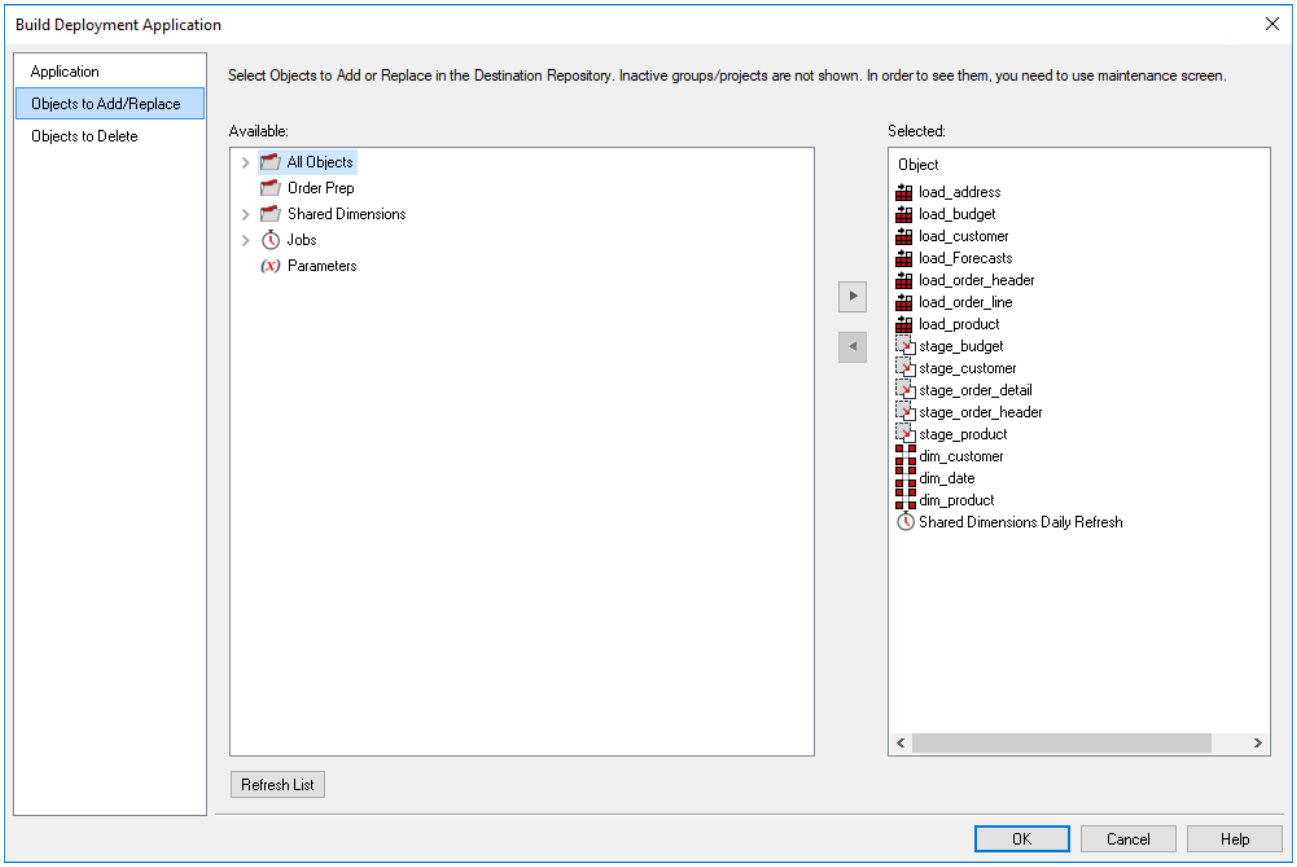
You can select or enter a previous application file as a starting point for this application.

Previous Application:

Pre Application Load SQL. (the following optional SQL statement will be issued before the application load commences):

Post Application Load SQL. (the following optional SQL statement will be issued after the application load completes):

3. Edit the objects to add or replace as required.



Build Deployment Application

Select Objects to Add or Replace in the Destination Repository. Inactive groups/projects are not shown. In order to see them, you need to use maintenance screen.

Available:

- All Objects
- Order Prep
- Shared Dimensions
- Jobs
- Parameters

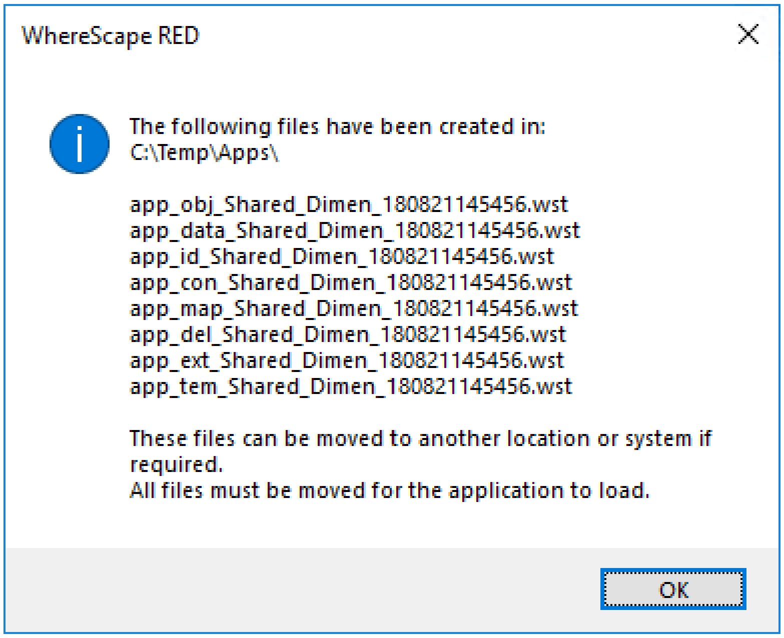
Selected:

Object
- load_address
- load_budget
- load_customer
- load_Forecasts
- load_order_header
- load_order_line
- load_product
- stage_budget
- stage_customer
- stage_order_detail
- stage_order_header
- stage_product
- dim_customer
- dim_date
- dim_product
- Shared Dimensions Daily Refresh

Note

Creating an application from a job will save the objects in the job and the job, but not the associated objects.

4. Click **OK** when finished.
5. A message window appears, confirming the creation of the application files. Click **OK**.



WhereScape RED

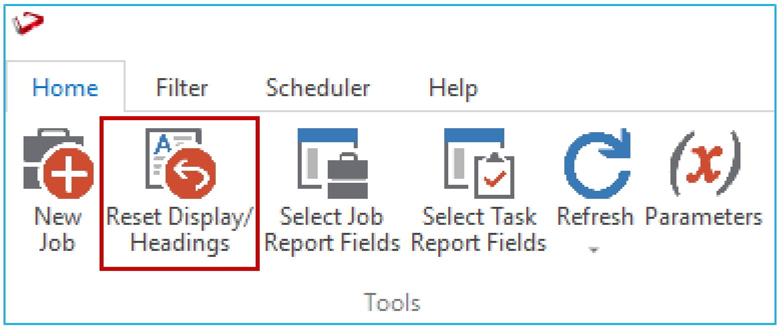
The following files have been created in:
C:\Temp\Apps\

app_obj_Shared_Dimen_180821145456.wst
app_data_Shared_Dimen_180821145456.wst
app_id_Shared_Dimen_180821145456.wst
app_con_Shared_Dimen_180821145456.wst
app_map_Shared_Dimen_180821145456.wst
app_del_Shared_Dimen_180821145456.wst
app_ext_Shared_Dimen_180821145456.wst
app_tem_Shared_Dimen_180821145456.wst

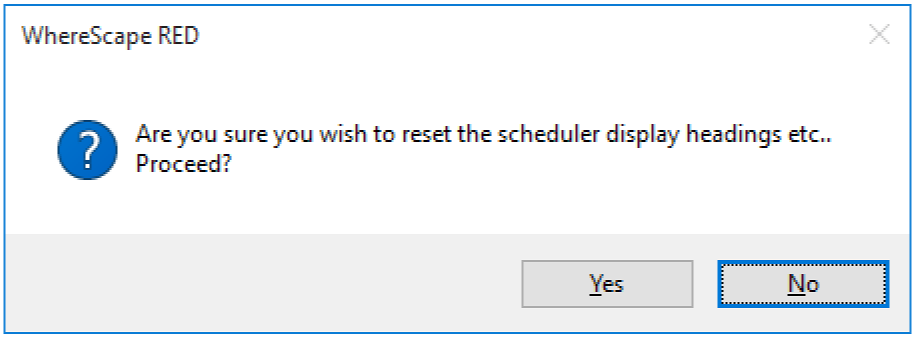
These files can be moved to another location or system if required.
All files must be moved for the application to load.

Reset Columns in Job and Task View

Job and Task Report headings can be reset by selecting the **View > Reset Display Headings** menu option from the Scheduler window. The short-cut keys are Alt+V-R.



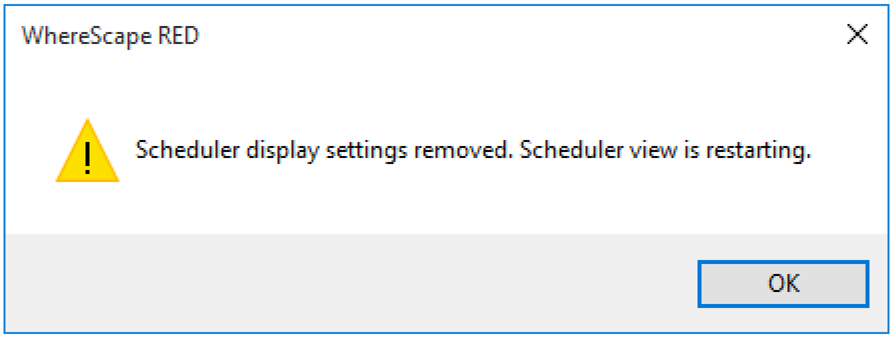
A message prompt asks you to confirm the request.



WhereScape RED

Are you sure you wish to reset the scheduler display headings etc.. Proceed?

Select **Yes** to reset the display settings, a message is displayed to confirm the reset has occurred.



Scheduler display settings removed. Scheduler view is restarting.

Indexes

Indexes can be defined on any table object in WhereScape RED but RED will only auto-generate indexes for permanent tables that have specific column types such as business keys.

To work with indexes in RED you will require the following items which are usually provided by your Enablement Pack if your target platform requires them:

- DDL Templates for both Create Index and Drop Index operations
- A Validate Index SQL Block - for index validation statements

RED will auto-generate indexes when building the Primary routine for an object based on the column key types specified and the object type. This feature can be toggled on or off from the connection properties of the associated target connection.

The **Enable Automatic Creation of Indexes** option in the **Connection Properties** window enables you to turn ON/OFF the automatic creation of indexes when generating routines for objects. Neither the metadata nor the physical index is created, if this setting is not enabled.

The generated indexes can be altered or deleted. New indexes can be created as required.

Note

The maintenance of the indexes is performed as part of the normal scheduler processing.

In the left pane, right-click a table to:

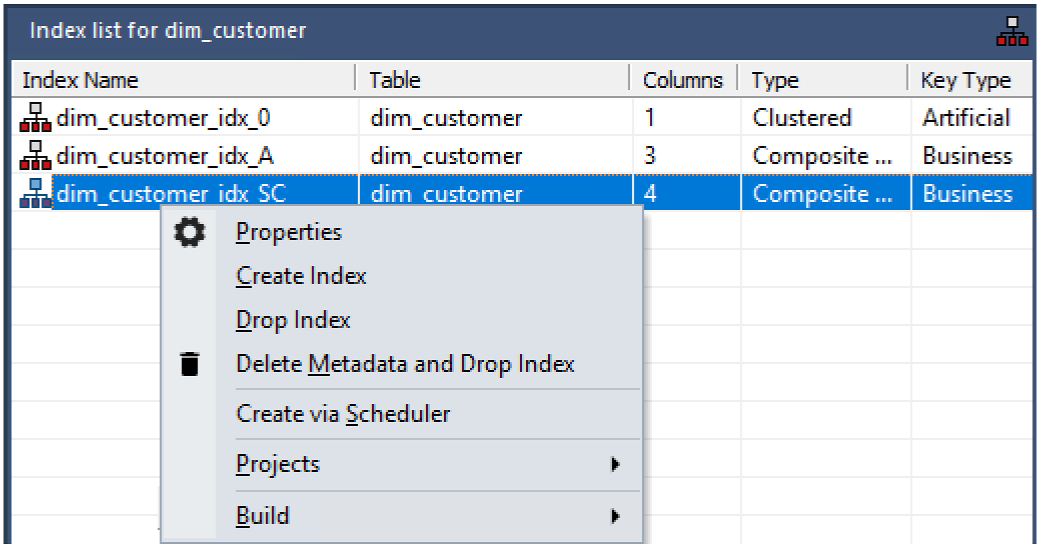
- Display indexes
- Add indexes

Index Definition

Right-clicking a table in the left pane and selecting **Display Indexes** from the context menu lists the indexes for the table in the middle pane. Alternatively, you can double-click the Index object type in the left pane, to display all indexes in the repository or a specific group or project.

In the middle pane, right-click an index and the following options are displayed:

- Properties
- Create Index
- Drop Index
- Delete Metadata and Drop Index
- Create via Scheduler
- Projects
- Build



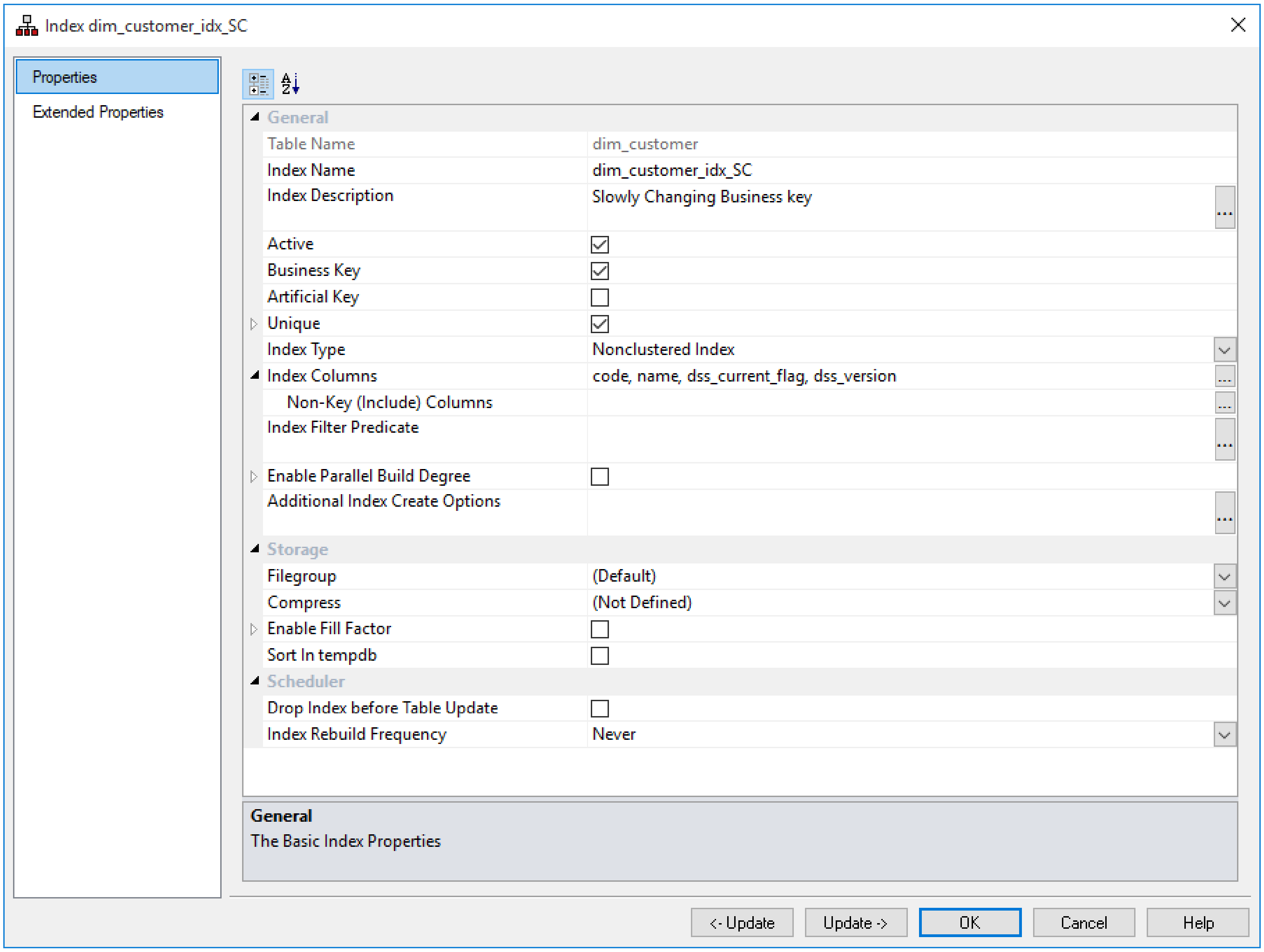
Index list for dim_customer

Index Name	Table	Columns	Type	Key Type
dim_customer_idx_0	dim_customer	1	Clustered	Artificial
dim_customer_idx_A	dim_customer	3	Composite ...	Business
dim_customer_idx_SC	dim_customer	4	Composite ...	Business

Properties

The **Properties** screen (see example below) can be selected via the right-click context menu when positioned on an index name in the middle pane. The Update Buttons: **Update <-** and **Update ->** are used to move to the previous and next index respectively. The Update Buttons are not available when browsing all indexes in a group, project or repository.

SQL Server example:



Index dim_customer_idx_SC

General

Table Name	dim_customer
Index Name	dim_customer_idx_SC
Index Description	Slowly Changing Business key
Active	☑
Business Key	☑
Artificial Key	□
Unique	☑
Index Type	Nonclustered Index
Index Columns	code, name, dss_current_flag, dss_version
Non-Key (Include) Columns	
Index Filter Predicate	
Enable Parallel Build Degree	□
Additional Index Create Options	

Storage

Filegroup	(Default)
Compress	(Not Defined)
Enable Fill Factor	□
Sort In tempdb	□

Scheduler

Drop Index before Table Update	□
Index Rebuild Frequency	Never

General
The Basic Index Properties

The fields in the **General** section are described below:

Field	Description
Index Name	Typically the table short name followed by: • **_idx_0** indicating primary key • **_idx_n** where n = any number from 1 indicating dimensional keys • **_idx_x** where x = any letter a thru z indicating business keys.
Index Description	Free flow description of the index.
Active	• When enabled means the index is in use. • When disabled means the index is not managed by the scheduler.
Business Key	Denotes a business key.
Artificial Key	When enabled indicates that this is the surrogate (artificial) key generated by the system.
Unique	Specifies that the index is a unique index. **Note** If both **unique and artificial** are set it is assumed to be a primary key constraint and it is added as such.
Index Type	Database-specific type of index On **SQL Server** the options are: • Nonclustered Index

Field	Description
	- Clustered Index - ColumnStore Index available for SQL2012
Index Columns	Shows the columns in the order they will be applied to the index. The order can be changed using the up/down buttons at the left.
Non-Key (Include) Columns	Extend a Non-clustered Index to include Non-Key columns in addition to the key columns.
Index Filter Predicate	A filtered index is an optimized Non-clustered index, which uses a filter predicate to
Enable Parallel Build Degree	Activates parallel creation /build of the index. For unsupported database versions (Pre SQL2005) leave disabled.
Additional Index Create Options	Database-specific-and-compliant options to include in the generated CREATE INDEX statement.

The fields in the **Storage** section are described below:

Field	Description
Filegroup/Tablespace	Default Filegroup/Tablespace of the index that determines the storage location it is created. Select (Default) to use the default.
Compress	Compress index to reduce disk and memory use. For unsupported database versions (Pre SQL2008) select (Not Defined).
Enable Fill Factor	Setting this will enable setting of the Fill Factor. For unsupported database versions (Pre SQL2005) leave disabled.
Sort In tempdb	Use the tempdb to store the intermediate sort results that are used to build the index, which may reduce the index build time if tempdb is on different disks to the index.

The fields in the **Scheduler** section are described below:

Field	Description
Drop Index before Table Update	Drop the index before running the table update procedure and recreate the index after the update procedure has completed.
Index Rebuild Frequency	Optional frequency that the index is automatically rebuilt by the Scheduler.

Indexes are normally managed by the scheduler as part of the normal processing of a table.

Extended Properties

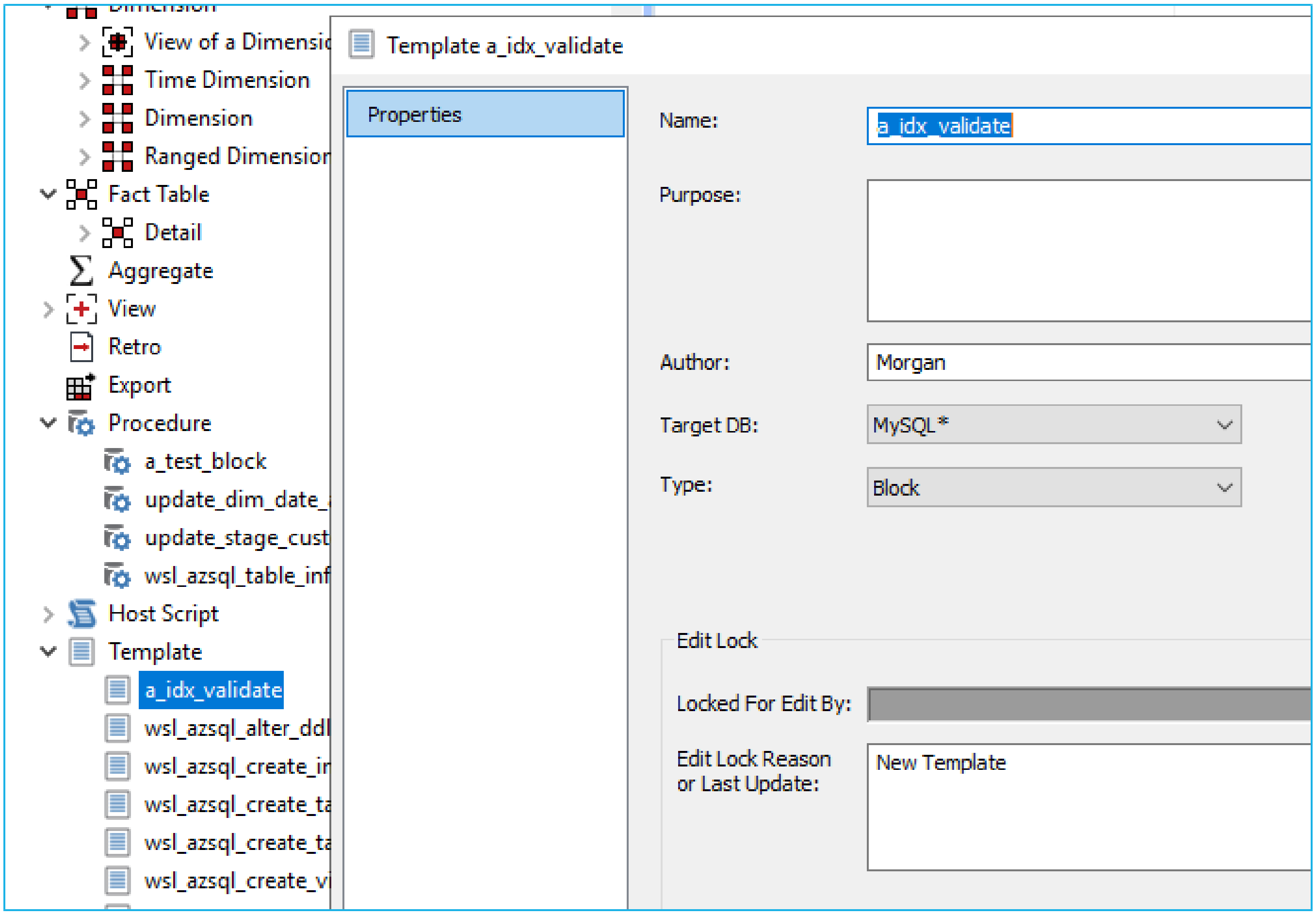
Extended property variables can be set for index objects. An extended property's scope is defined by object type but the value for the variable can be set for the table and each index independently. Refer to **Creating an Extended Property Definition** for details.

Automatically created indexes are recreated each time the table's procedure is rebuilt. Therefore, the values set for extended properties in automatically created indexes are lost when they are recreated. Manually created indexes and their extended properties are persisted.

Similar to the extended properties defined for tables and connections, extended properties defined in indexes can be accessed from templates, including create index DDL, drop index DDL and update procedure/script.

Index Validate Process

Setting up the index validate process starts by creating a template of type **Block**.



Name: a_idx_validate

Purpose:

Author: Morgan

Target DB: MySQL*

Type: Block

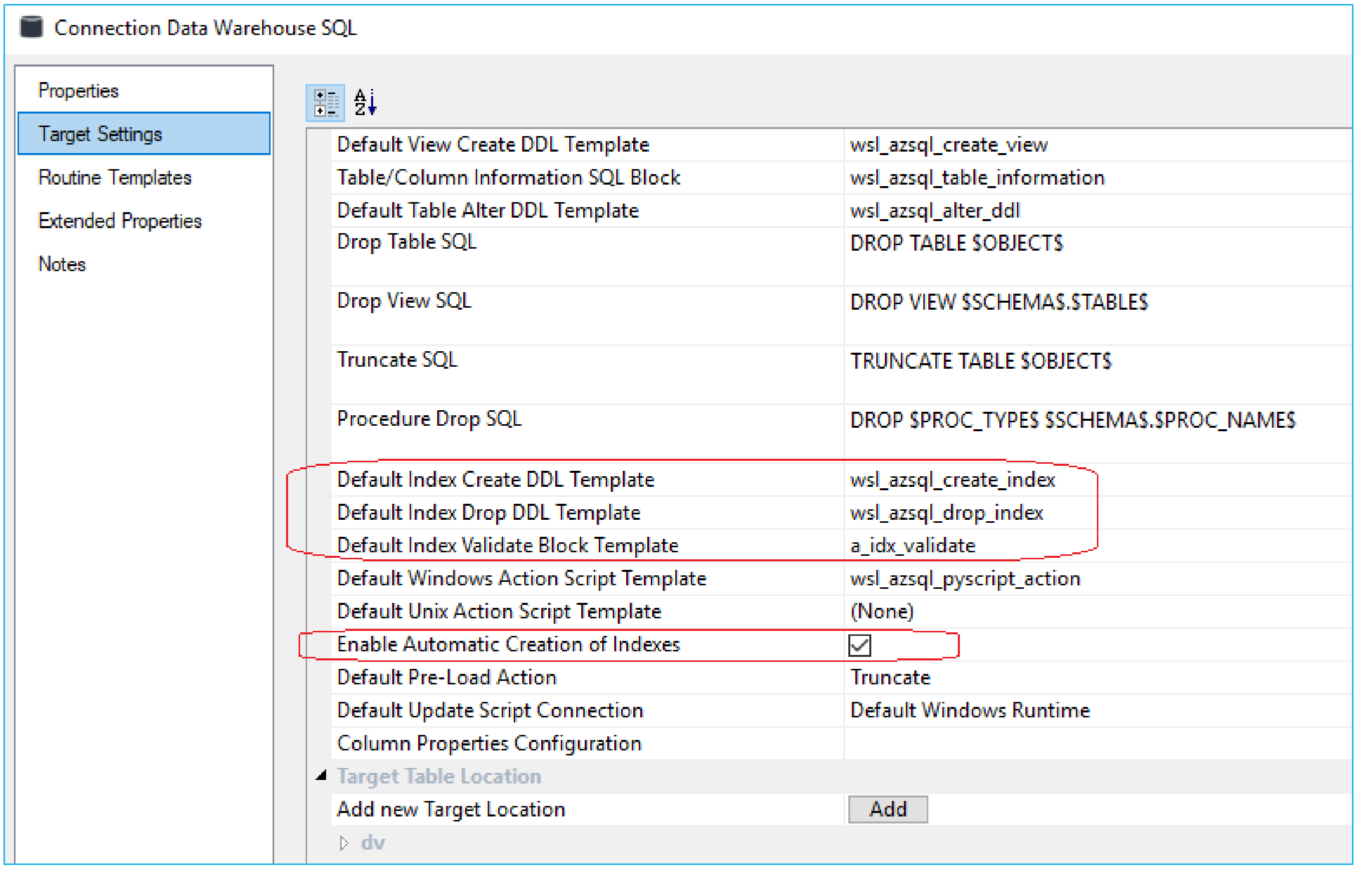
Locked For Edit By:

Edit Lock Reason or Last Update: New Template

Following is an example of a SQL Server template:

```
SELECT '{{index.name}} does not exist'
EXCEPT
SELECT '{{index.name}} does not exist'
FROM sys.indexes
WHERE name = N'{{ index.name }}'
AND object_id = object_id(N'{{ index.table.schema }}.{{ index.table.name }}')
```

Select the template in the Target Connection and make sure to check **Enable Automatic Creation of Indexes** option:

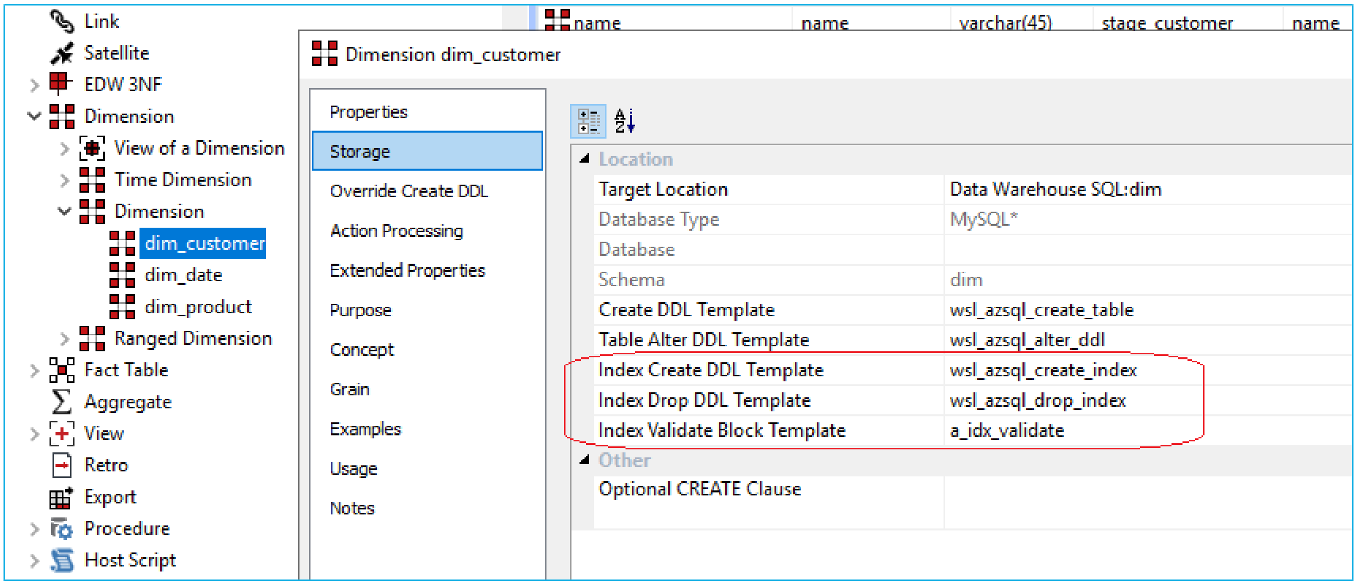


Connection Data Warehouse SQL

- Properties
- Target Settings
- Routine Templates
- Extended Properties
- Notes

Default View Create DDL Template	wsl_azsql_create_view
Table/Column Information SQL Block	wsl_azsql_table_information
Default Table Alter DDL Template	wsl_azsql_alter_ddl
Drop Table SQL	DROP TABLE $OBJECT$
Drop View SQL	DROP VIEW $SCHEMA$.$TABLE$
Truncate SQL	TRUNCATE TABLE $OBJECT$
Procedure Drop SQL	DROP $PROC_TYPE$ $SCHEMA$.$PROC_NAME$
Default Index Create DDL Template	wsl_azsql_create_index
Default Index Drop DDL Template	wsl_azsql_drop_index
Default Index Validate Block Template	a_idx_validate
Default Windows Action Script Template	wsl_azsql_pyscript_action
Default Unix Action Script Template	(None)
Enable Automatic Creation of Indexes	☑
Default Pre-Load Action	Truncate
Default Update Script Connection	Default Windows Runtime
Column Properties Configuration	
Target Table Location	
Add new Target Location	Add
dv	

Select the index templates at the table level for existing tables, new tables pick these up automatically:



Dimension dim_customer

- Properties
- Storage
- Override Create DDL
- Action Processing
- Extended Properties
- Purpose
- Concept
- Grain
- Examples
- Usage
- Notes

Location	
Target Location	Data Warehouse SQL:dim
Database Type	MySQL*
Database	
Schema	dim
Create DDL Template	wsl_azsql_create_table
Table Alter DDL Template	wsl_azsql_alter_ddl
Index Create DDL Template	wsl_azsql_create_index
Index Drop DDL Template	wsl_azsql_drop_index
Index Validate Block Template	a_idx_validate
Other	
Optional CREATE Clause	

Next, an example of an index validate template for PostgreSQL:

```
{# -- TemplateVersion:001 MinVersion:21010 MaxVersion:* TargetType:Postgres ModelType:*
TemplateType:Block       -- #}
SELECT '{{index.name | lower}} does not exist'
EXCEPT
SELECT CONCAT(i.relname,' does not exist')
FROM
  pg_catalog.pg_class t,
  pg_catalog.pg_class i,
  pg_catalog.pg_index ix
WHERE
  t.oid = ix.indrelid
  AND i.oid = ix.indexrelid
  AND t.relkind = 'r'
  AND t.relname = '{{ index.table.name | lower }}'
  AND i.relname =  '{{ index.name | lower }}'
;

{%- for col in index.columns %}
SELECT '{{ index.name | lower }} requires updating'
EXCEPT
SELECT CONCAT(i.relname,' requires updating')
FROM
    pg_class t,
    pg_class i,
    pg_index ix,
    pg_attribute a
WHERE
    t.oid = ix.indrelid
    AND i.oid = ix.indexrelid
    AND a.attrelid = t.oid
    AND a.attnum = ANY(ix.indkey)
    AND t.relkind = 'r'
    AND t.relname = '{{ index.table.name | lower }}'
    AND i.relname =  '{{ index.name | lower }}'
        AND a.attname = '{{ col.name | lower }}'
;
{% endfor %}
```

Notes

Notes about validating indexes:

- Validate Index block is executed during Procedure/Script Rebuild/Regenerate workflows
- If there is no configured Index Validate Block Template in a table object that an index is associated with, then the index validate fails.
- The SQL Block which is generated from the configured Index Validate Block Template can consist of zero of more Select queries that fetch a single column of type Varchar(256).
- Each Select query is separated by the SQL Block statement separator which is configured in Tools > Options (for example, <EOS>).
- Each Select query is executed in turn, in the order in the generated SQL Block.
- Any null values or empty strings are ignored from the result set.
- If there are any non-null non-empty strings returned, these indicate a validation failure.
- As soon as a query in the template returns any validation errors, the process halts (so subsequent queries of the generated SQL Block are not executed).
- If the index validate fails for any reason, then the drop and create index operations will be performed when the index is regenerated.

Documentation and Diagrams

WhereScape RED includes the ability to document the data warehouse based on the information stored against the metadata for all the tables and columns in the data warehouse.

The documentation will only be meaningful if information is stored in the metadata. The business definition and a description should be stored against all columns that will be visible to end users.

The following sections describe how to **generate** and **read** the documentation.

Also included is a section on information that is available to assist in the connection of end user **query tools**.

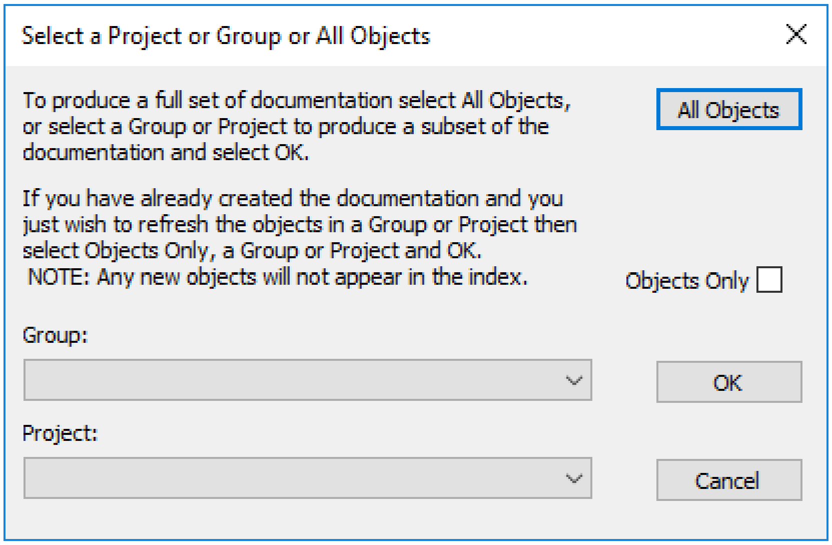
Creating Documentation

To create the documentation for the components of the data warehouse, click **Doc** from the Builder window menu, then click **Create Documentation**.

If the repository has Projects or Groups in use, then the following window appears to allow the selection of a specific group or project. The default is all objects in the repository.

Note

Projects or Groups that have been flagged as inactive are not displayed/available from the Group and Project drop-down lists. Refer to **Maintaining Group/Project Active Flag** for details.



Select a Project or Group or All Objects

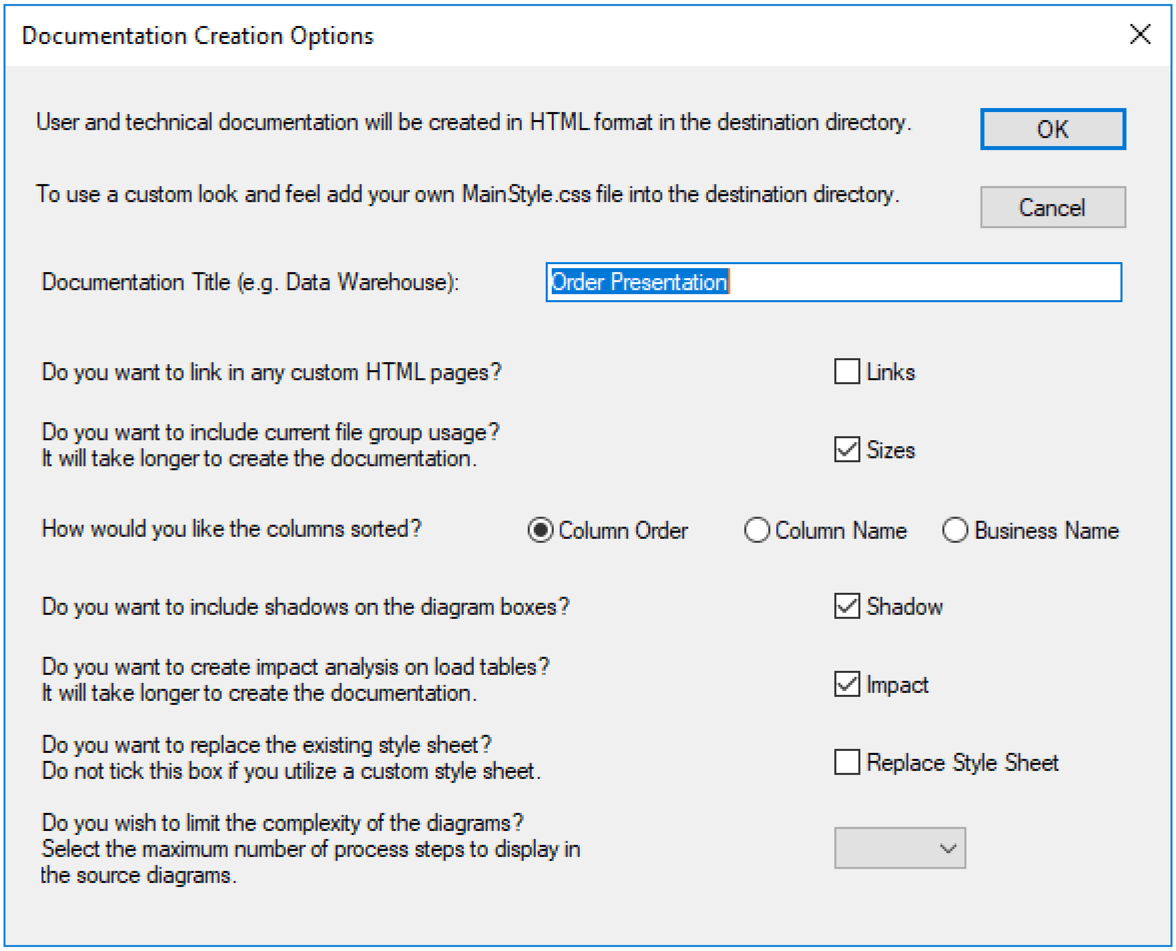
To produce a full set of documentation select All Objects, or select a Group or Project to produce a subset of the documentation and select OK.

If you have already created the documentation and you just wish to refresh the objects in a Group or Project then select Objects Only, a Group or Project and OK.
NOTE: Any new objects will not appear in the index.

A file dialog appears which enables you to select a file path (directory) where the generated HTML files are saved.

A style sheet called **mainstyle.css** is created if it does not exist. If this style sheet exists then it is not overwritten. The style sheet can therefore be modified to match an existing intranet standard.

The next screen enables the inclusion of a banner and user defined links. It also provides some build options:


Documentation Creation Options

User and technical documentation will be created in HTML format in the destination directory.

To use a custom look and feel add your own MainStyle.css file into the destination directory.

Documentation Title (e.g. Data Warehouse): Order Presentation

Do you want to link in any custom HTML pages? □ Links

Do you want to include current file group usage?
It will take longer to create the documentation. ☑ Sizes

How would you like the columns sorted? ● Column Order ○ Column Name ○ Business Name

Do you want to include shadows on the diagram boxes? ☑ Shadow

Do you want to create impact analysis on load tables?
It will take longer to create the documentation. ☑ Impact

Do you want to replace the existing style sheet?
Do not tick this box if you utilize a custom style sheet. □ Replace Style Sheet

Do you wish to limit the complexity of the diagrams?
Select the maximum number of process steps to display in
the source diagrams.

The size checkbox instructs the documentation generator to examine the size of all tables and indexes in the database. This process may be slow in some situations, thus should normally be used only for final documentation.

The sorted checkbox sorts the columns within the tables into alphabetical order. By default, the columns are in the order in which they appear within the tables.

Creating a header

If you check the banner frame option then a banner (heading) will be created at the top of each page in the documentation. You will be prompted for height of the banner frame in pixels (default is 60), an image file (jpg or gif) and any banner text. It is recommended that any image be relatively small (60 pixels high or approximately 1/2 an inch) as it will appear on every page.

Adding Links

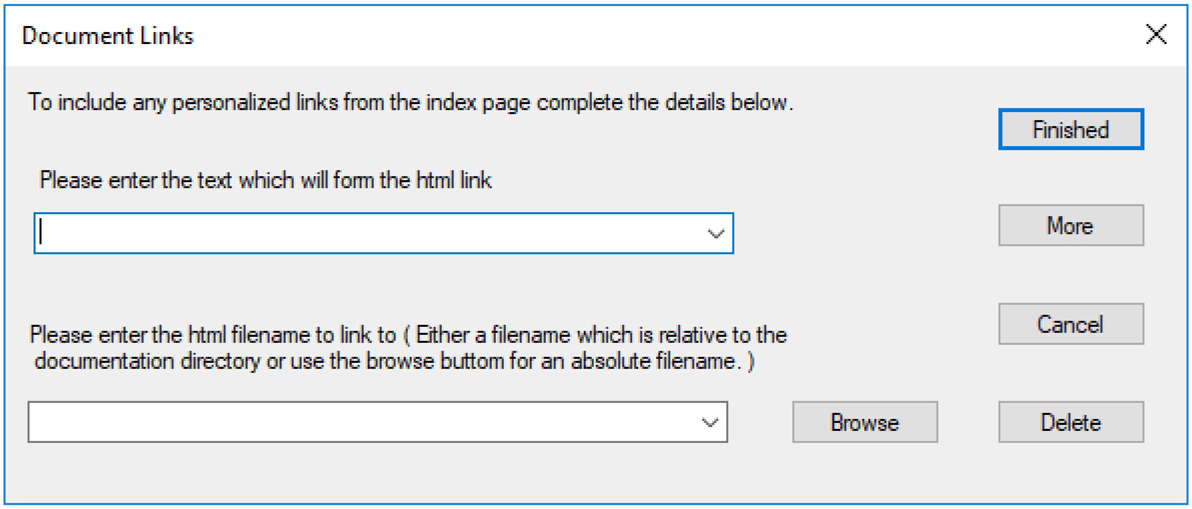
Custom information can be linked into the generated documentation.

This means that every time the documentation is regenerated, custom information will be included. In this way the complete documentation for the data warehouse can be viewed in one location.

If you select the add links option then you are prompted to include personalized links from index pages. These links must be to previously created HTML files.

Index pages (linkage points) are available at three points:

- index - initial page
- techindex - technical documentation initial page
- indexuser - user documentation initial page



Document Links

To include any personalized links from the index page complete the details below.

Please enter the text which will form the html link

Please enter the html filename to link to (Either a filename which is relative to the documentation directory or use the browse buttom for an absolute filename.)

Multiple links can be added to each index page by using the **More** option.

Adding glossary elements

As part of the user documentation a glossary is produced defining all the business terms and column names used in the data warehouse. This glossary is primarily based on the columns in the dimension and fact tables. Additional information can however, be added via the 'Ws_Api_Glossary' procedure defined in the procedures chapter. This procedure allows the manual inclusion of glossary elements that are stored in the metadata repository and added to the glossary, whenever the documentation is recreated.

Batch Documentation Creation

WhereScape RED includes the ability to document the data warehouse, based on the information stored against the metadata for all the tables and columns in the data warehouse. In a larger environment, it may be a good idea to generate documentation in batch mode.

The following syntax chart illustrates the options available:

```
med.exe /BD { /U UserName { /P Password } } /C OdbcSource { /M Schema } { /N
FullName } /D Directory { /G GroupName | ProjectName } /S NumHops { /I- }
```

Note:

{ } indicates an optional parameter and | indicates alternative values.

Parameter Descriptions

The following parameters are available:

Parameter	Specify Value?	Mandatory?	Description
BD	No	Yes	Indicates batch documentation mode.
U	Yes	Sometimes *1	ODBC username parameter.
P	Yes	Sometimes *1	ODBC password parameter.
C	Yes	Yes	ODBC data source parameter.
A	Yes	No	ODBC DSN architecture.
M	Yes	Sometimes *2	Metadata database parameter.

Parameter	Specify Value?	Mandatory?	Description
N	Yes	No	RED user name parameter, only for logging purposes.
D	Yes	Yes	Directory name where documentation is created.
G	Yes	No	Group or Project name if specified. All Objects if not included.
S	Yes	Yes	Number of processes/hops in the source diagrams.
I-	No	No	Exclude Impact Analysis on Load tables (reduces the time required to generate documentation)

Notes

- *1. User Name and Password are not required when using a trusted connection or operating system authentication.

Example

The following example connects to a SQL Server repository called WslWarehouse, using a trusted connection into the C:\Temp\my_doco directory with 4 hops in diagrams:

```
med /BD /C "WslWarehouse" /D "C:\Temp\my_doco" /S "4"
```

Reading the Documentation

To read the documentation you have created, select **Doc** from the builder menu bar, then **Display Documentation**. This will launch a browser and display the contents of index.html. Alternatively, you can access the HTML pages directly from their saved location.

Diagrams

Types of Diagrams

Six types of diagrams are provided to give visual representation of what has been created. These are

- The **Schema Diagram**
- The **Source Diagram**
- The **Joins Diagram**
- The **Links Diagram**
- The **Impact Diagram**
- The **Dependency Diagram**

1. To display the **Diagram Selection** window, click the New Diagram button 📊 from the toolbar.

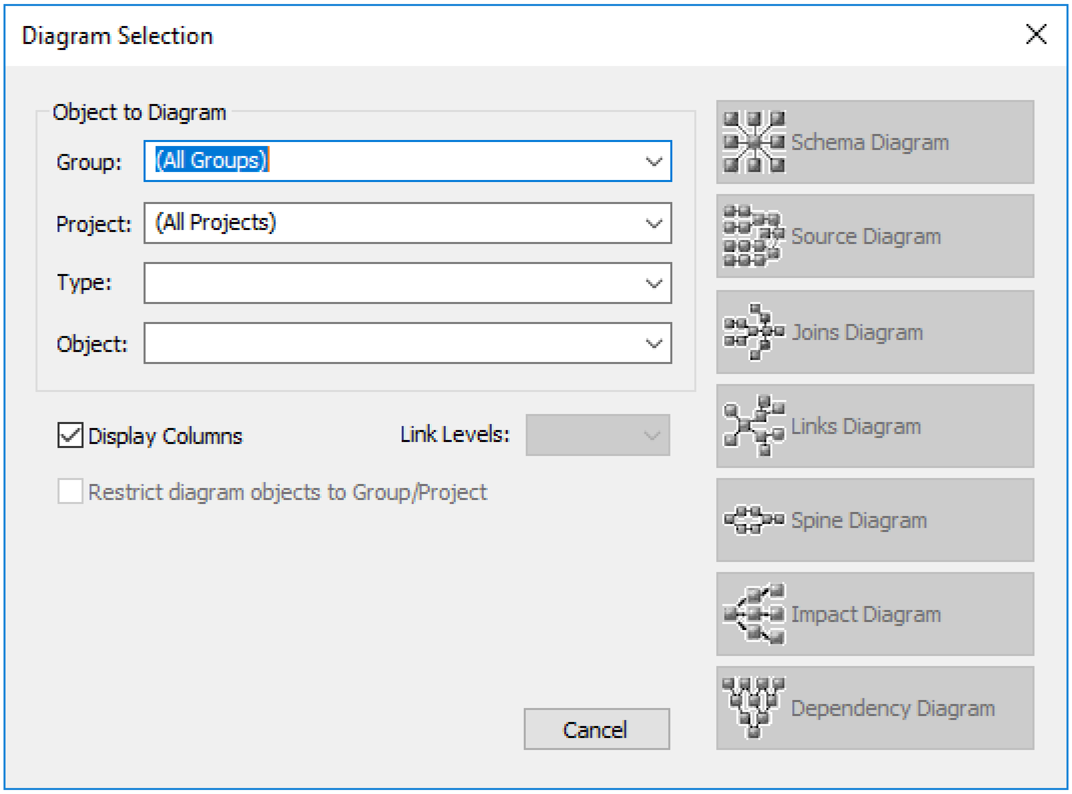


Diagram Selection

Object to Diagram

Group: (All Groups)

Project: (All Projects)

Type:

Object:

☑ Display Columns

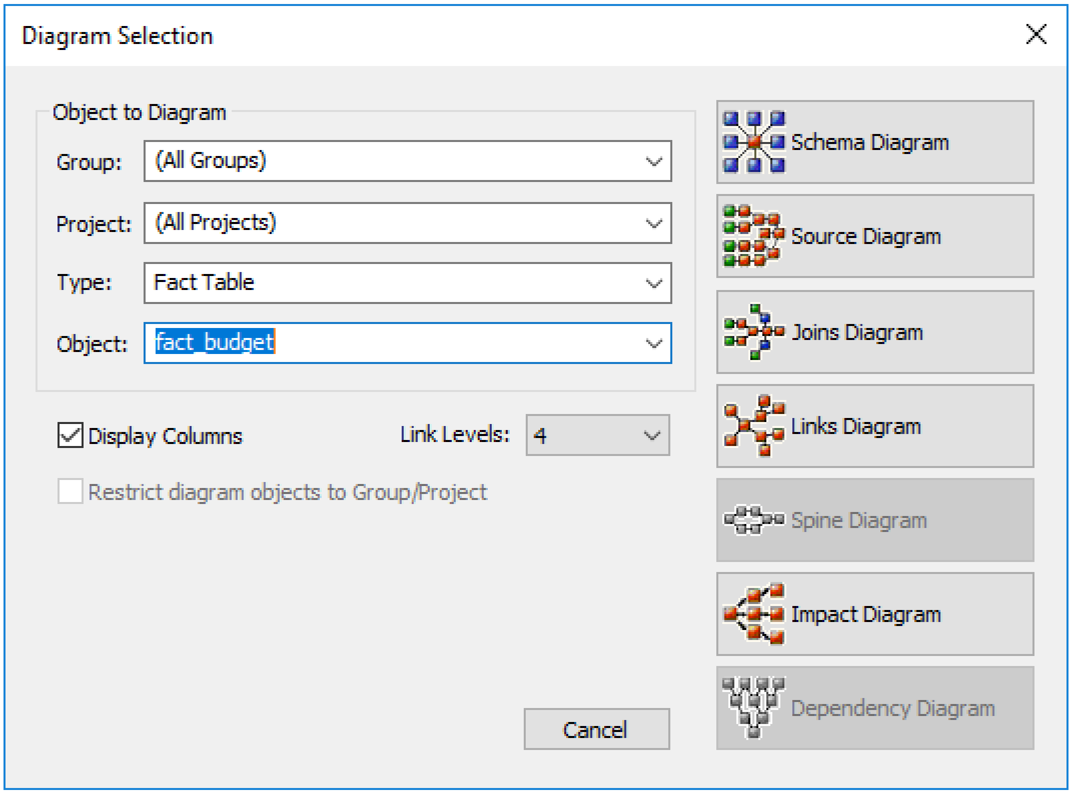
Link Levels:

□ Restrict diagram objects to Group/Project

2. Choose the object to diagram by optionally choosing the **Type** to limit the selection list; and then selecting the **Object**. The diagram type buttons on the right becomes active and you can choose the type of diagram to display.

Note

Projects or Groups that have been flagged as inactive are not displayed/available from the Group and Project drop-down lists. Refer to **Maintaining Group/Project Active Flag** for details.



Group: (All Groups)

Project: (All Projects)

Type: Fact Table

Object: fact_budget

☑ Display Columns

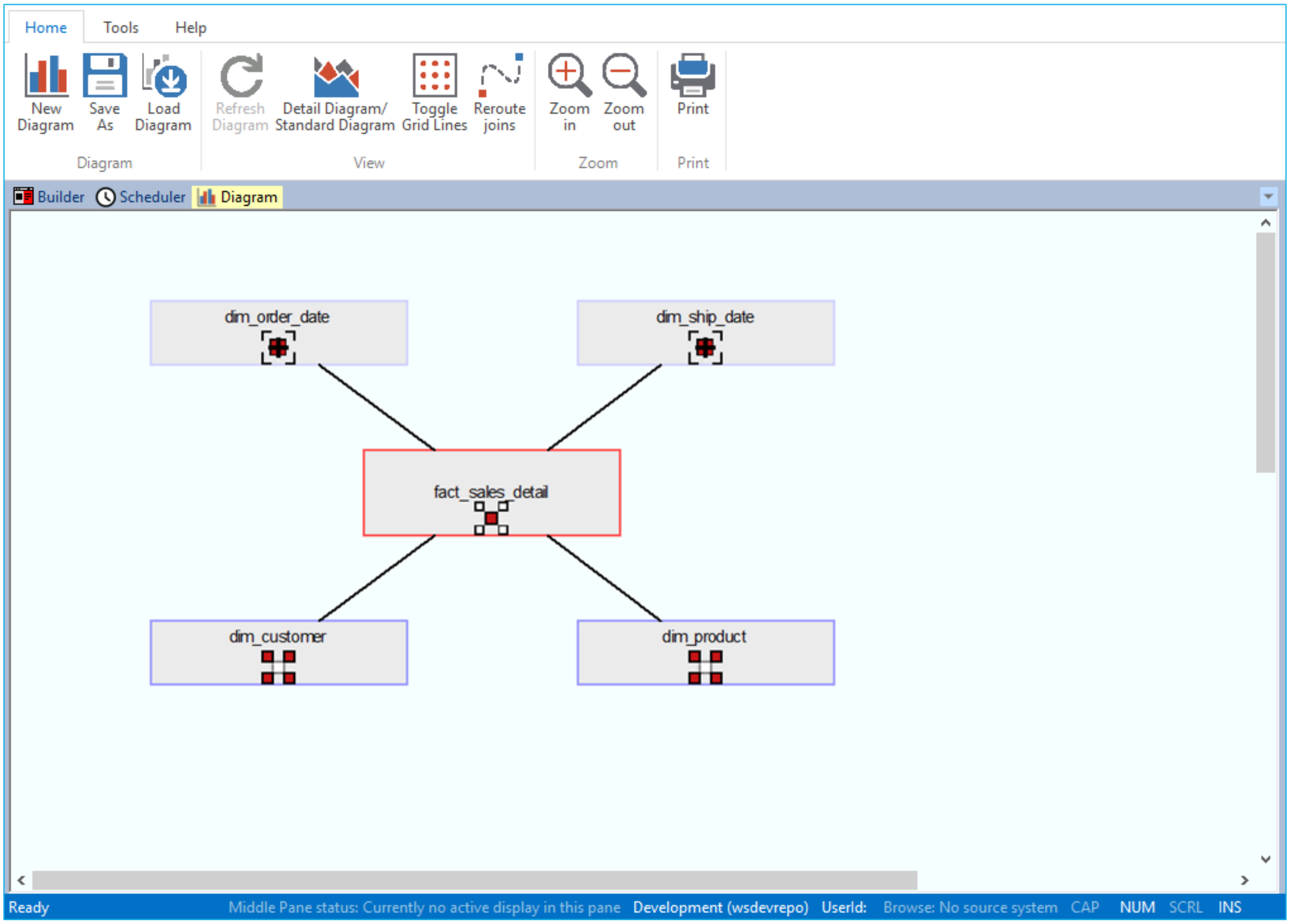
Link Levels: 4

□ Restrict diagram objects to Group/Project

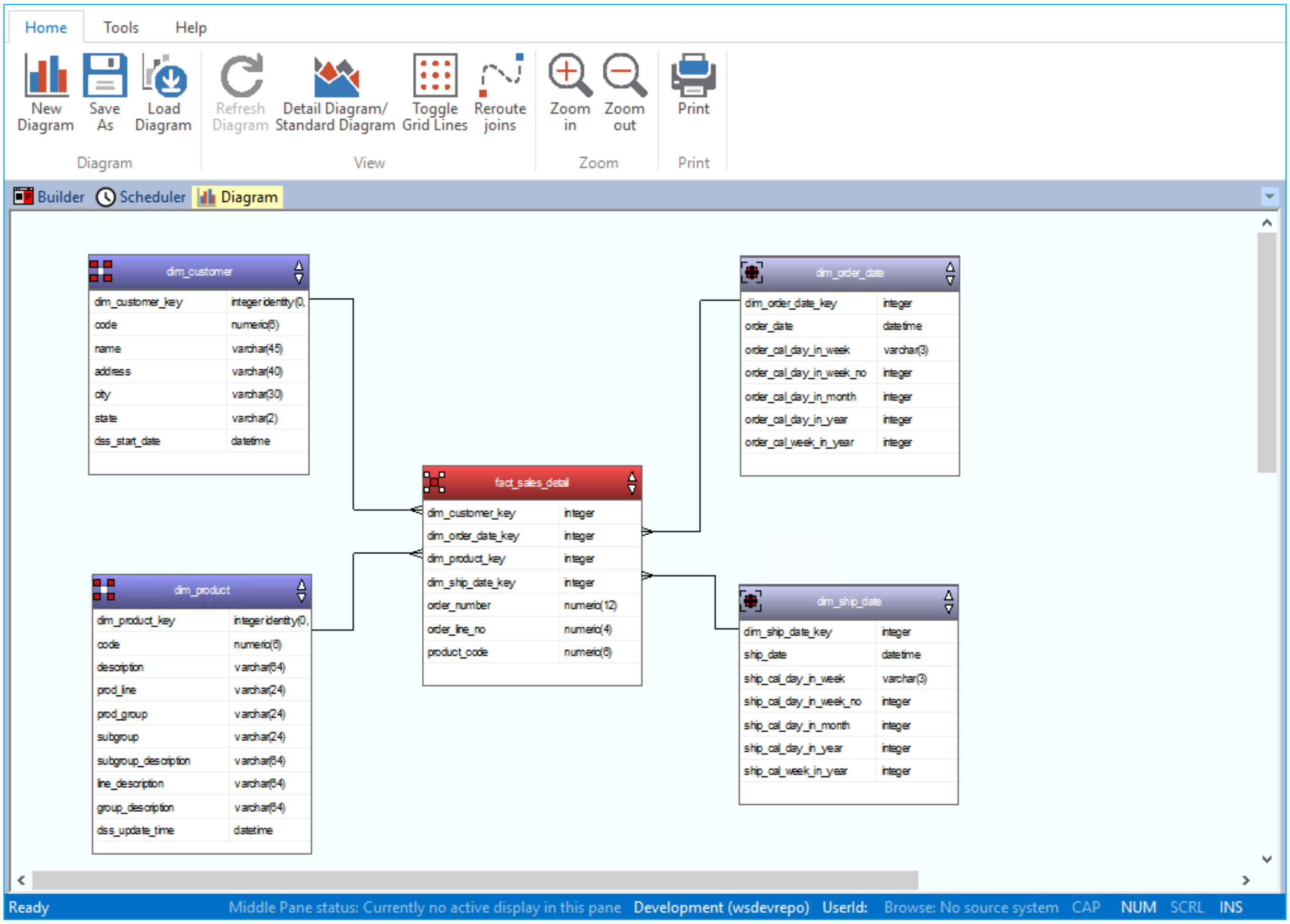
Schema Diagram

A star schema diagram can be displayed for a fact table, an aggregate table and an OLAP cube. It shows the central table with the outlying dimensions.

An example of a **Schema** diagram in **Standard Diagram** format is displayed below.



An example of a **Schema** diagram in **Detail Diagram** format is displayed below.



dim_customer	
dim_customer_key	integer identity(0,
code	numeric(6)
name	varchar(45)
address	varchar(40)
city	varchar(30)
state	varchar(2)
dss_start_date	datetime

dim_order_date	
dim_order_date_key	integer
order_date	datetime
order_cal_day_in_week	varchar(3)
order_cal_day_in_week_no	integer
order_cal_day_in_month	integer
order_cal_day_in_year	integer
order_cal_week_in_year	integer

fact_sales_detail	
dim_customer_key	integer
dim_order_date_key	integer
dim_product_key	integer
dim_ship_date_key	integer
order_number	numeric(12)
order_line_no	numeric(4)
product_code	numeric(6)

dim_product	
dim_product_key	integer identity(0,
code	numeric(6)
description	varchar(64)
prod_line	varchar(24)
prod_group	varchar(24)
subgroup	varchar(24)
subgroup_description	varchar(64)
line_description	varchar(64)
group_description	varchar(64)
dss_update_time	datetime

dim_ship_date	
dim_ship_date_key	integer
ship_date	datetime
ship_cal_day_in_week	varchar(3)
ship_cal_day_in_week_no	integer
ship_cal_day_in_month	integer
ship_cal_day_in_year	integer
ship_cal_week_in_year	integer

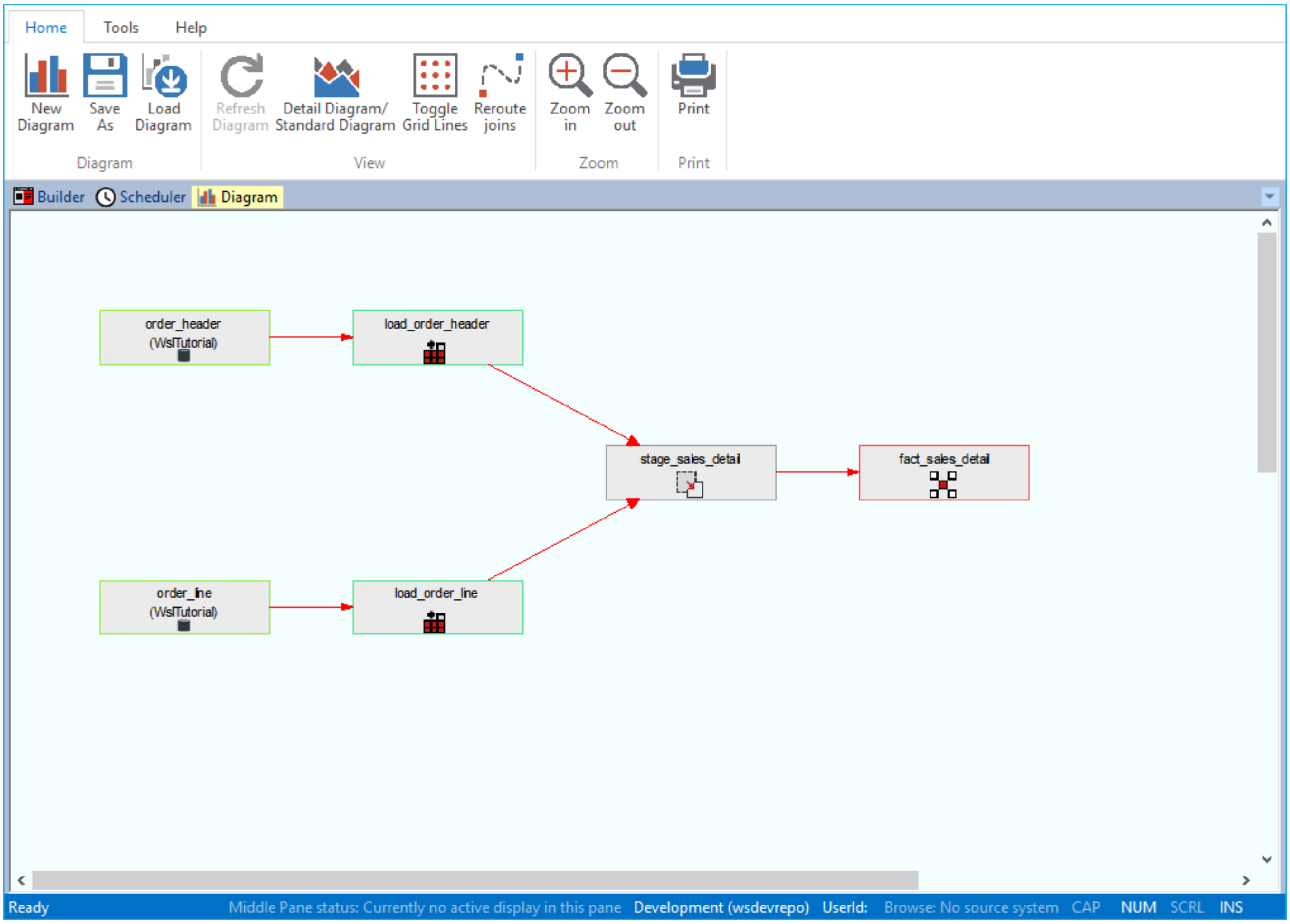
Ready   Middle Pane status: Currently no active display in this pane   Development (wsdevrepo)   UserId: Browse: No source system   CAP   NUM   SCRL   INS

The star schema diagrams are produced in **Standard Diagram** format as part of the user and technical documentation when you select **Doc > Create Documentation** from the main **Builder** window.

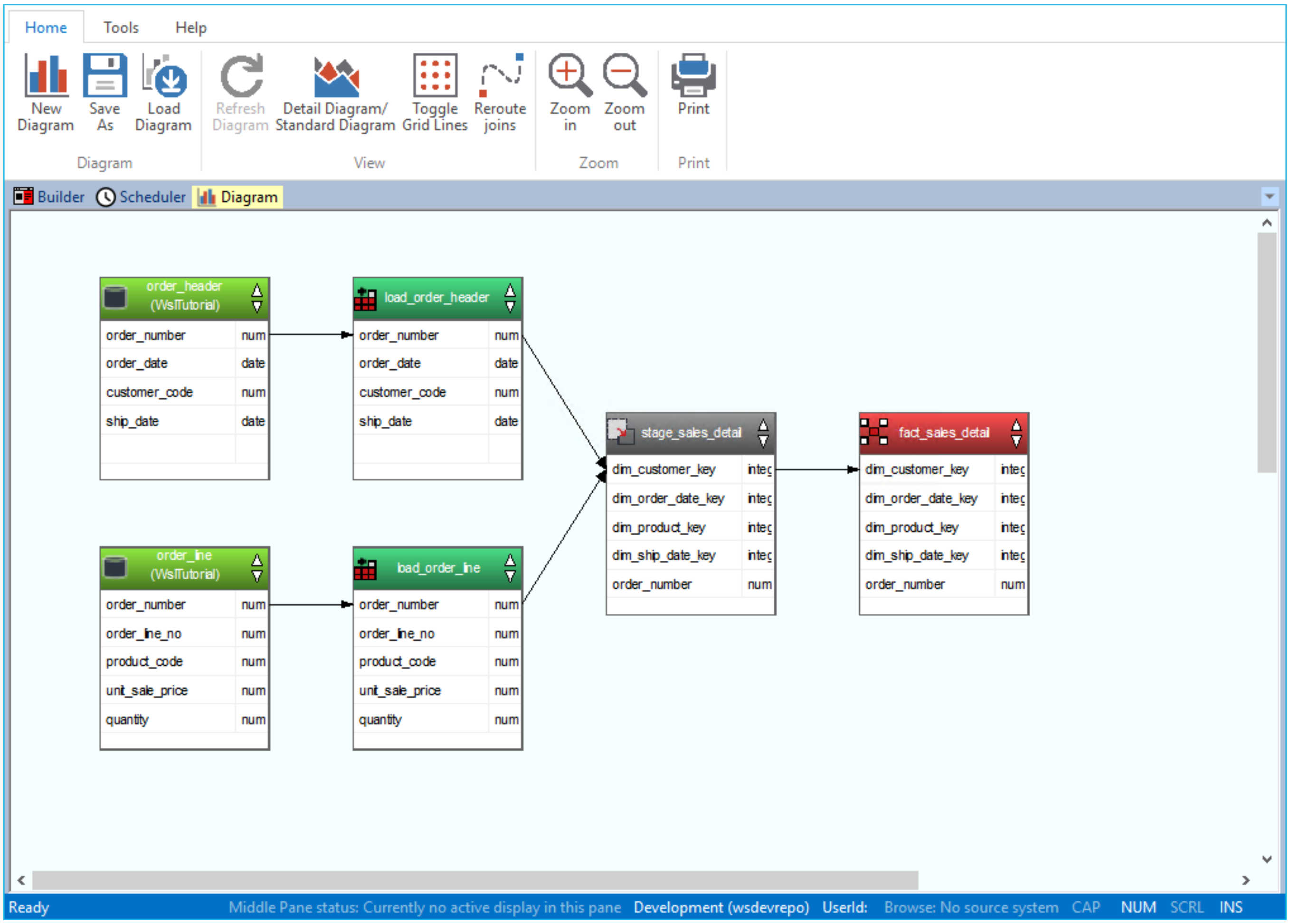
Source Diagram

A source tracking diagram can be displayed for any table. It shows connections back from the chosen table to the source tables from which information was derived. Hovering the cursor over a line shows additional information. For lines going into Load tables, the source of the data is displayed; while for other lines in the diagram, the procedure used to move data between two tables is displayed.

An example of a **Source** diagram in **Standard Diagram** format is displayed below.



An example of a **Source** diagram in **Detail Diagram** format is displayed below.



order_header (WsITutorial)	
order_number	num
order_date	date
customer_code	num
ship_date	date

load_order_header	
order_number	num
order_date	date
customer_code	num
ship_date	date

order_line (WsITutorial)	
order_number	num
order_line_no	num
product_code	num
unit_sale_price	num
quantity	num

load_order_line	
order_number	num
order_line_no	num
product_code	num
unit_sale_price	num
quantity	num

stage_sales_detail	
dim_customer_key	integ
dim_order_date_key	integ
dim_product_key	integ
dim_ship_date_key	integ
order_number	num

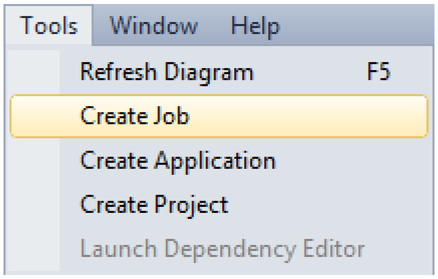
fact_sales_detail	
dim_customer_key	integ
dim_order_date_key	integ
dim_product_key	integ
dim_ship_date_key	integ
order_number	num

The Source diagrams are produced in **Standard Diagram** format as part of the technical documentation, when you select **Doc > Create Documentation** from the main Builder window.

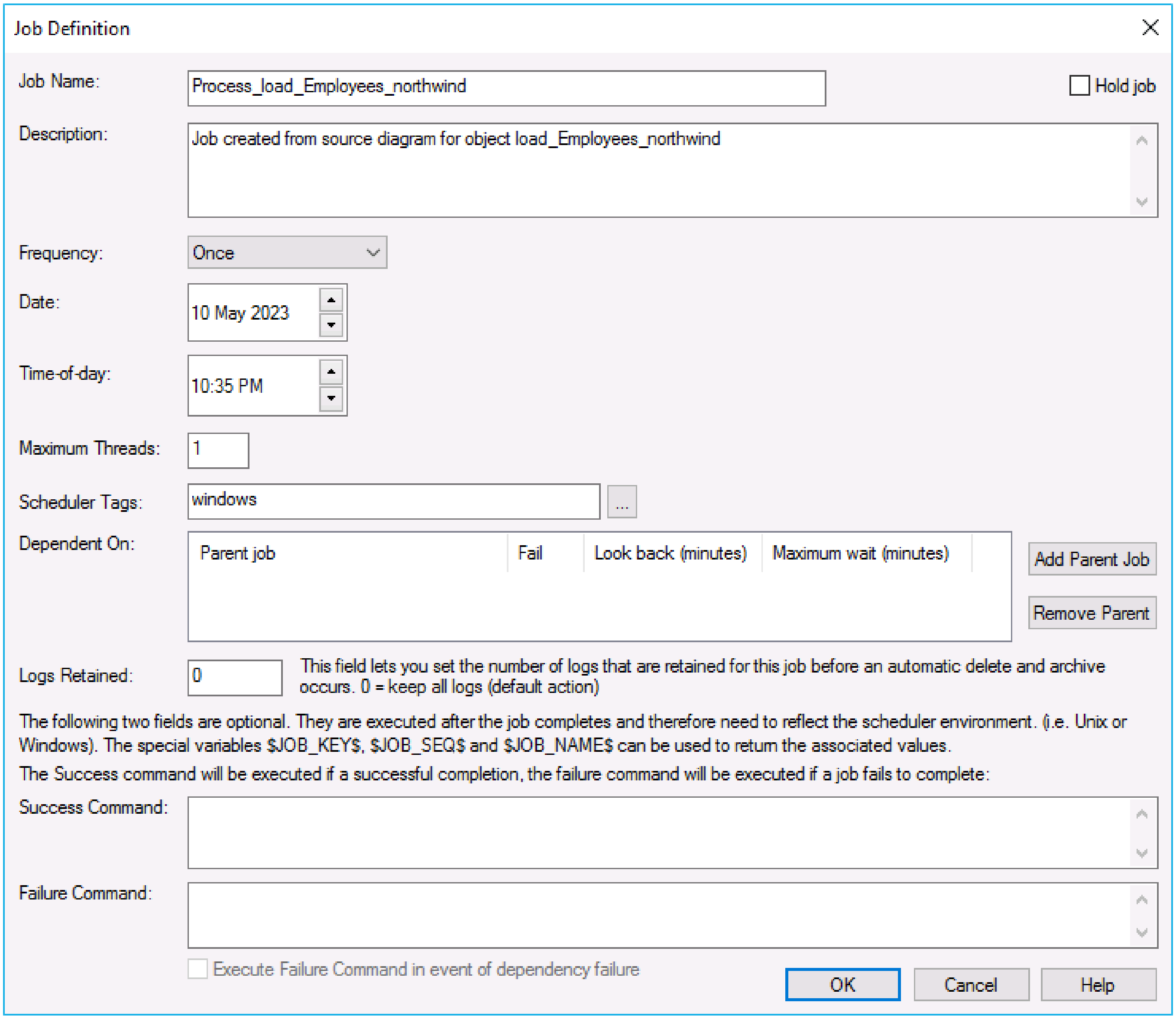
Creating a Job from a Source Diagram

Once a source tracking diagram has been created for a table, a scheduler job can be generated from the diagram. This job will be called **Process_to_table_name**, where table_name is the name of the table the track back diagram was run for.

To create a Job, select **Create Job** from the **Tools** menu after the diagram is displayed:



The Job Definition is then displayed:



Job Definition

Job Name: Process_load_Employees_northwind

☐ Hold job

Description: Job created from source diagram for object load_Employees_northwind

Frequency: Once

Date: 10 May 2023

Time-of-day: 10:35 PM

Maximum Threads: 1

Scheduler Tags: windows

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)

Logs Retained: 0

This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

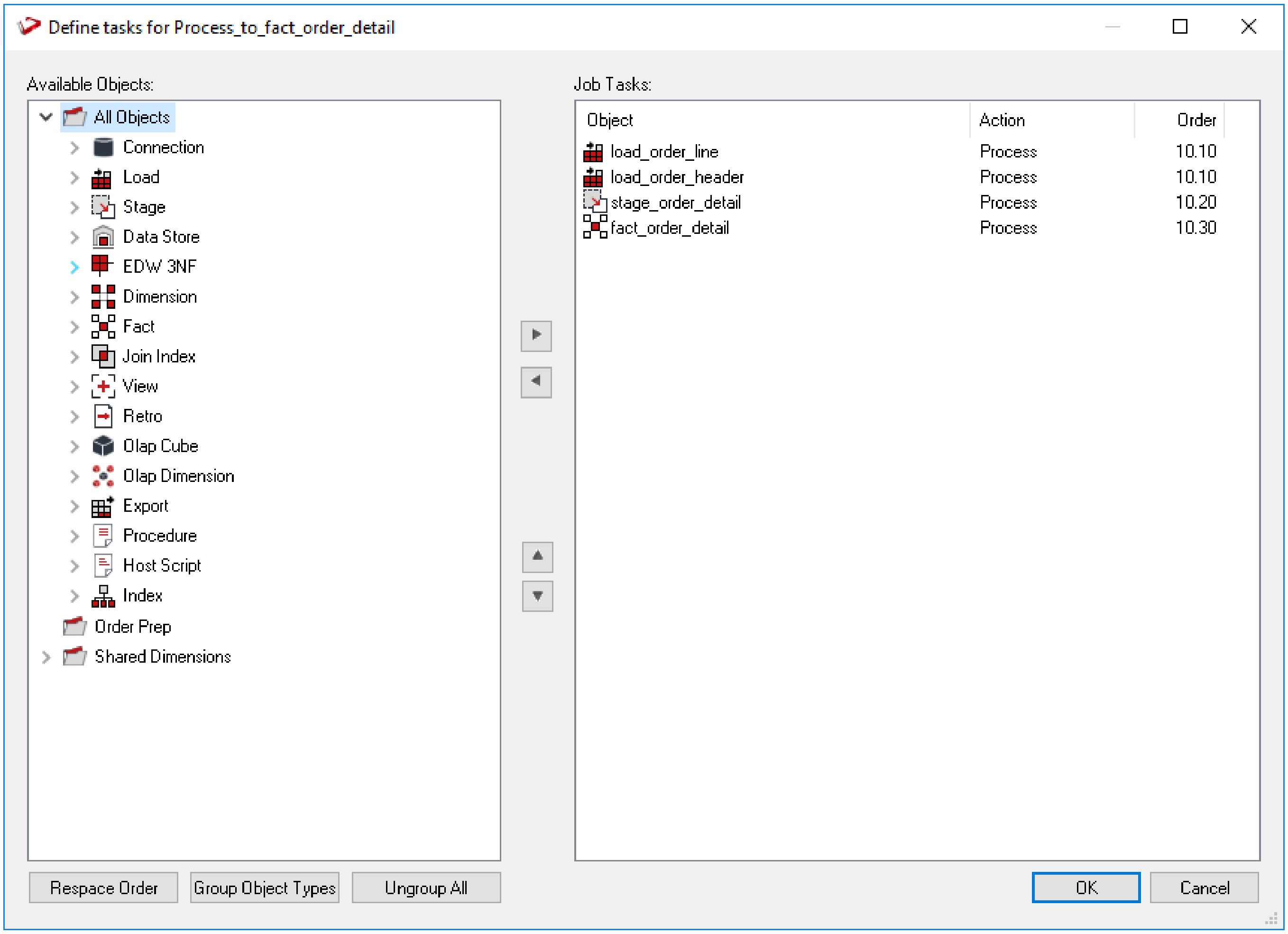
Success Command:

Failure Command:

☐ Execute Failure Command in event of dependency failure

Make any changes here that are required and click **OK**.

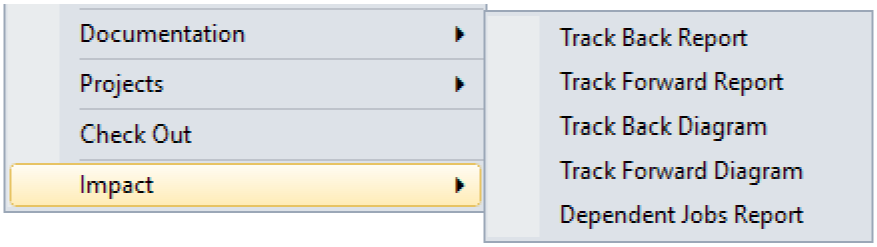
For the diagram above, a job is created with the following **tasks**:



Object	Action	Order
load_order_line	Process	10.10
load_order_header	Process	10.10
stage_order_detail	Process	10.20
fact_order_detail	Process	10.30

Note:

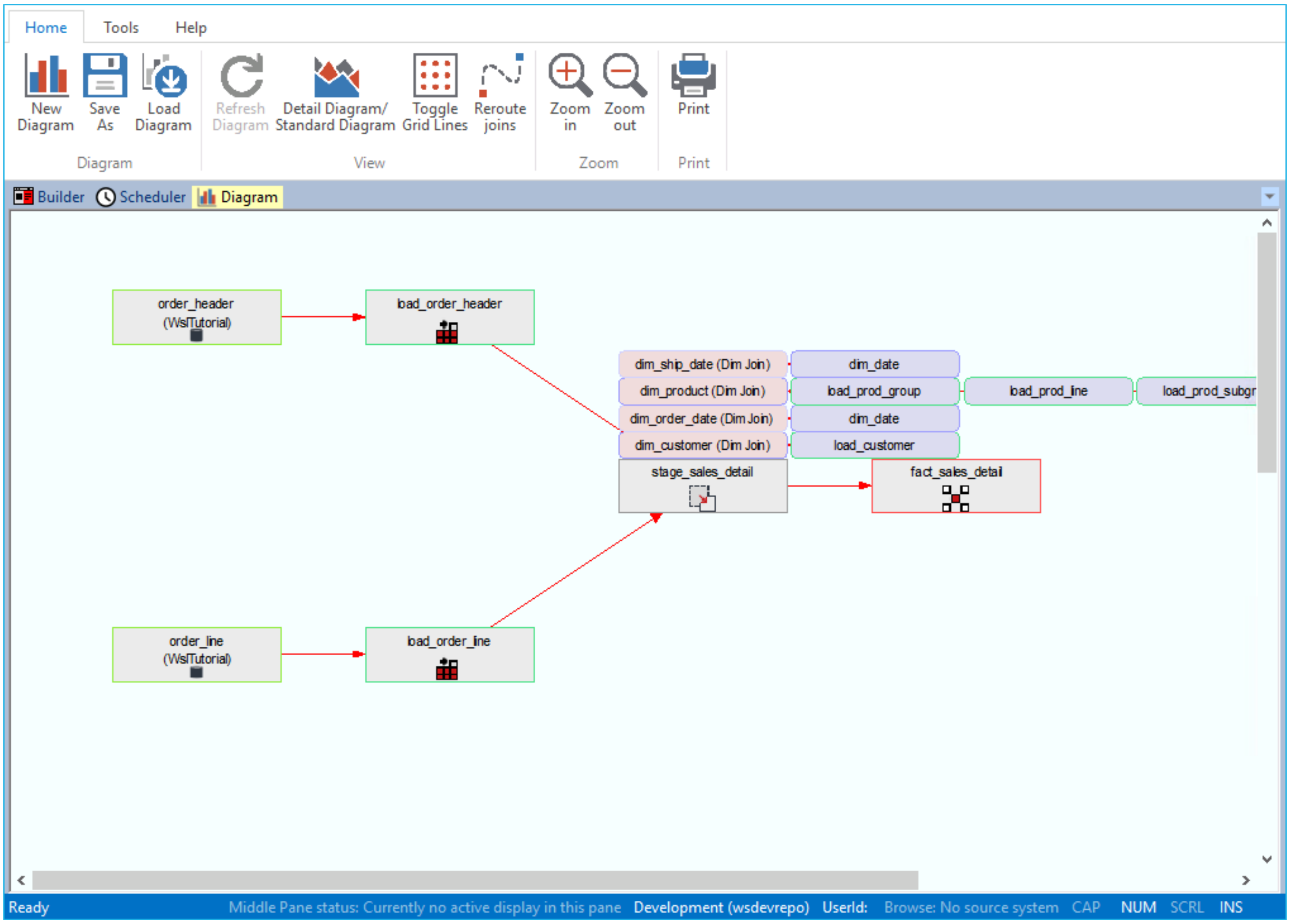
It is also possible to display the source diagram by right-clicking a table and choosing **Impact >Track Back Diagram:**



Joins Diagram

A data join track back diagram can be displayed for any table. It shows connections back from the chosen table to the source tables from which information was derived and includes Dimension table joins. Hovering the cursor over a line shows additional information. For lines going into Load tables, the source of the data is displayed; while for other lines in the diagram, the procedure used to move data between two tables is displayed.

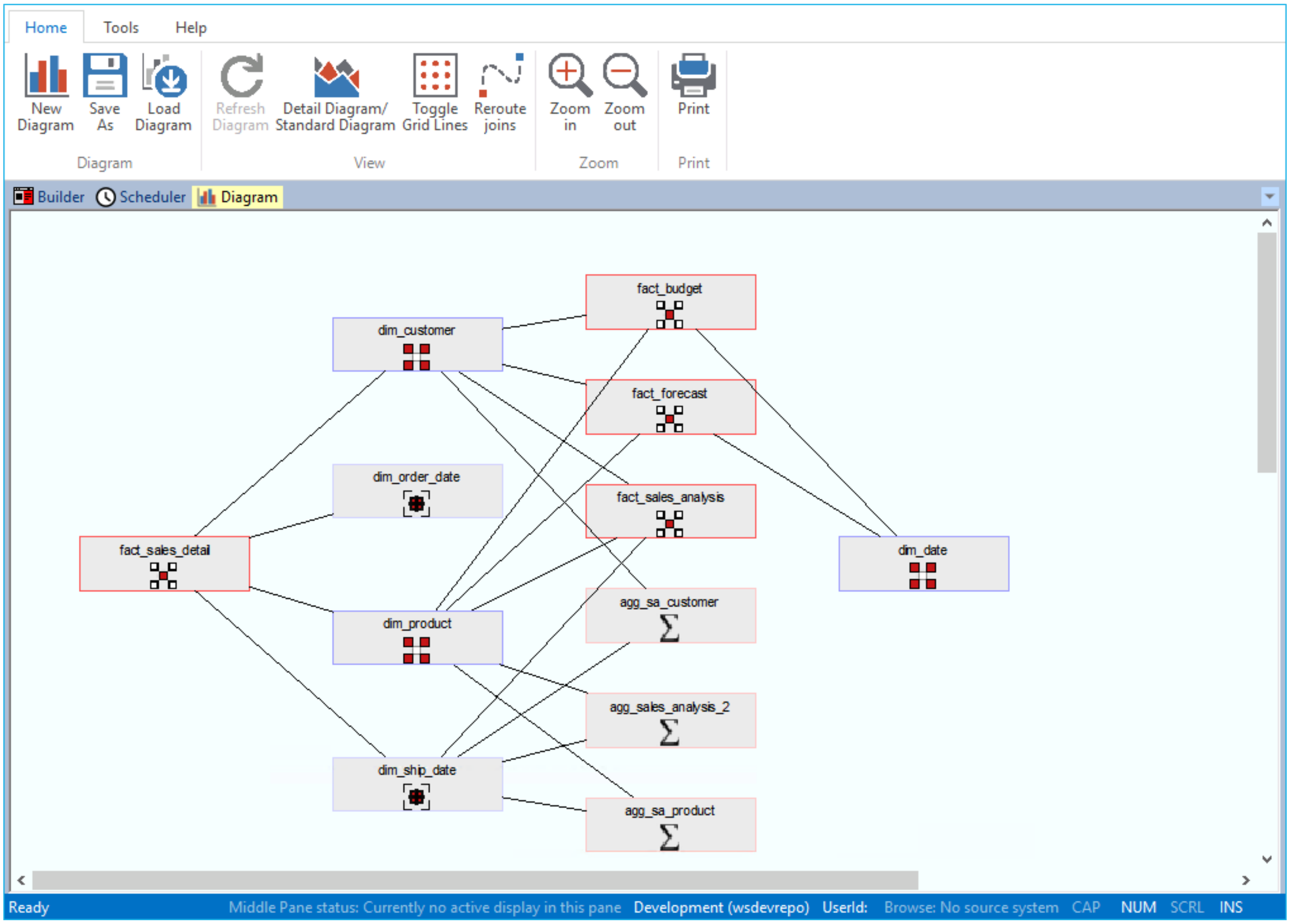
An example of a **Joins** diagram in **Standard Diagram** format is displayed below.



Links Diagram

A linked tables diagram can be shown for any table. It shows relationships between tables, looking out from the chosen table a selected number of hops. The number of hops is determined by table relationships and source and target relationships.

An example of a **linked tables** diagram in **Standard Diagram** format is displayed below.

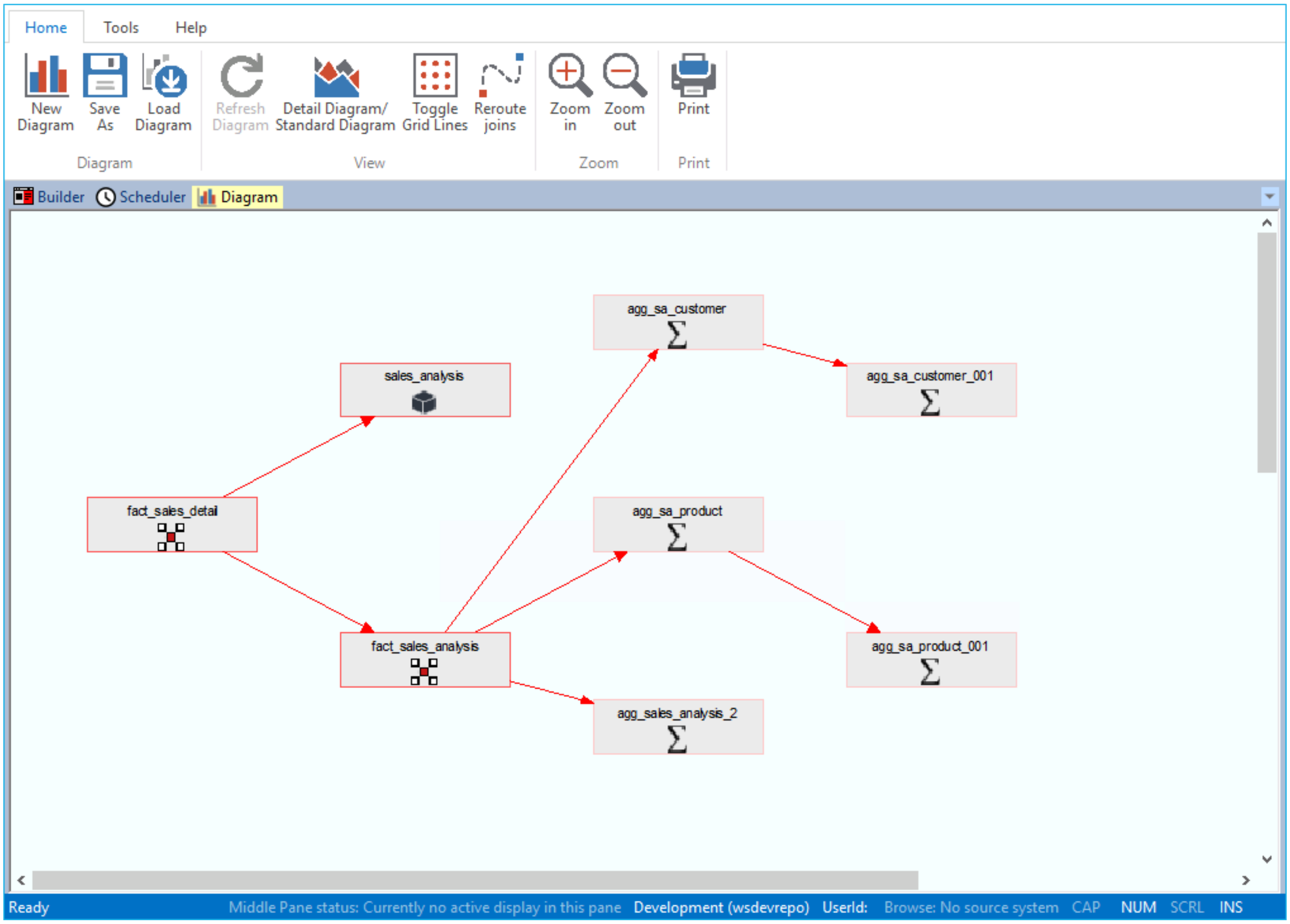


Impact Diagram

A track forward impact diagram can be displayed for any table. It shows connections forward from the chosen table to the subsequent tables built with columns from this table.

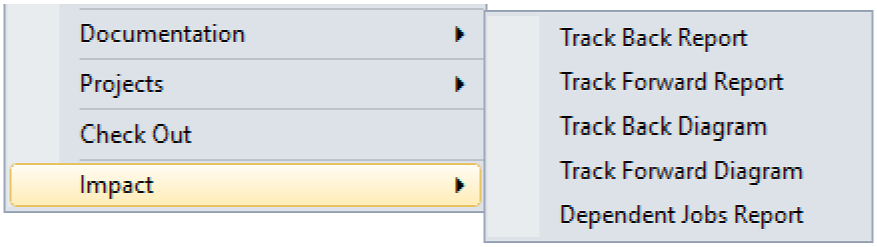
A track back impact diagram can be displayed for any table. It shows connections backwards from the chosen table to the previous tables.

An example of an **Impact** diagram in **Standard Diagram** format is displayed below.



Note

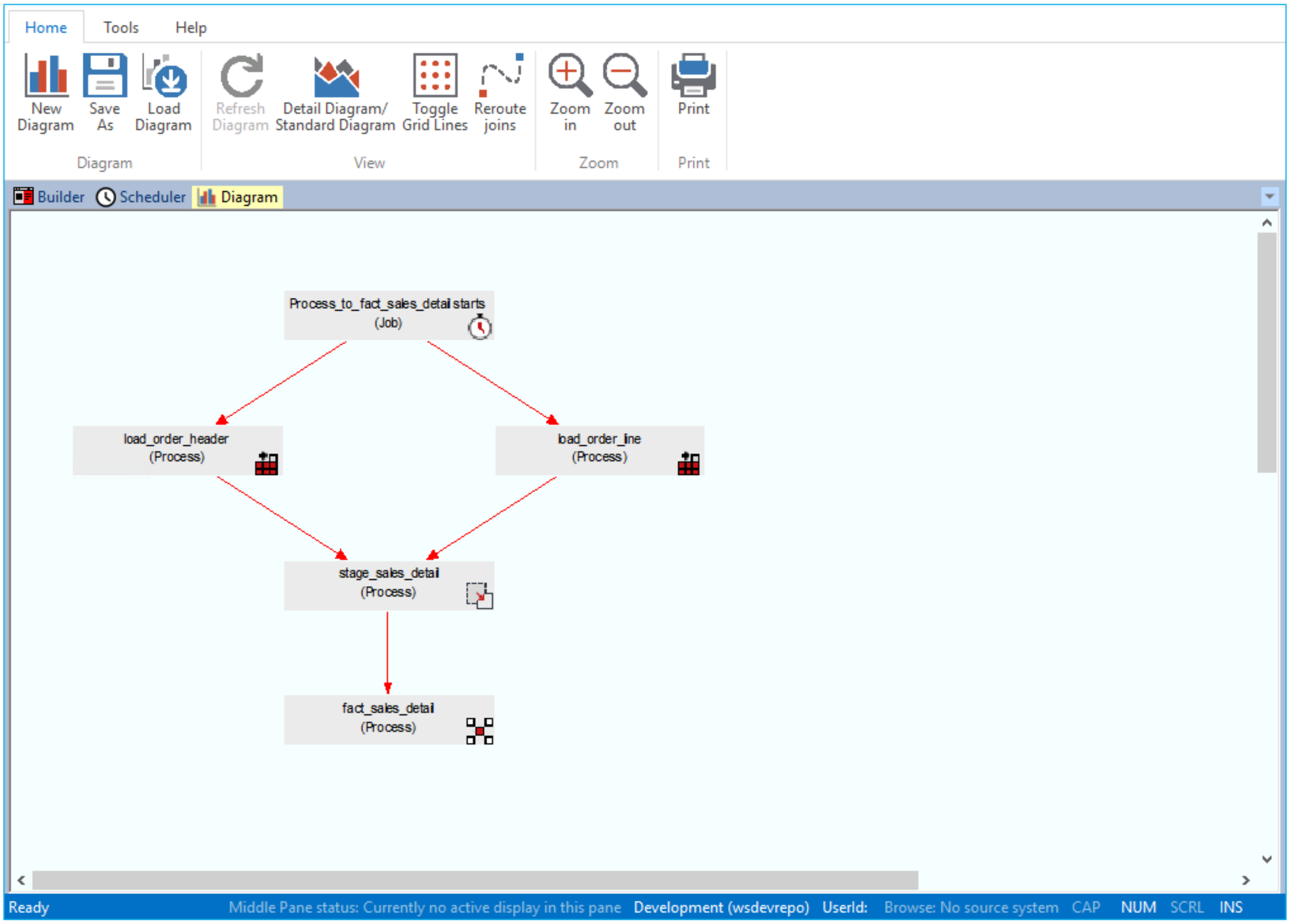
It is also possible to display the track back / forward diagram by right-clicking a table and choosing **Impact > Track Back Diagram or Impact > Track Forward Diagram:**



Dependency Diagram

A job dependency diagram can be displayed for any job defined in the WhereScape RED Scheduler. It shows the parent and child relationships between tasks within a job.

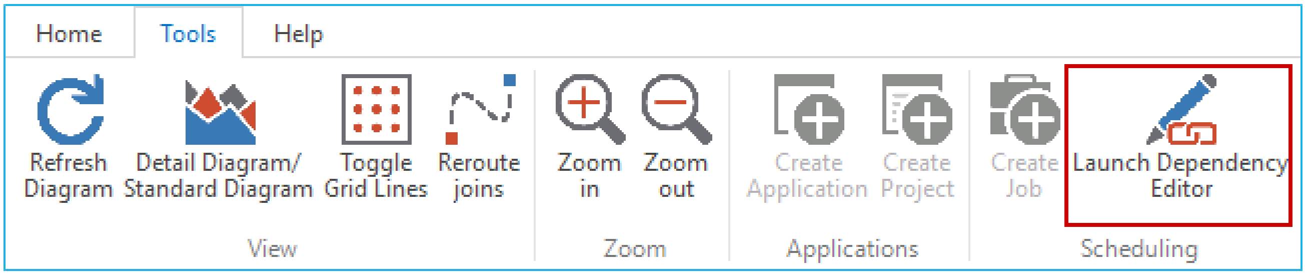
An example of a **Dependency** diagram in **Standard Diagram** format is displayed below.


Home
Tools
Help
New Diagram
Save As
Load Diagram
Refresh Diagram
Detail Diagram/ Standard Diagram
Toggle Grid Lines
Reroute joins
Zoom in
Zoom out
Print
Diagram
View
Zoom
Print
Builder
Scheduler
Diagram
Process_to_fact_sales_detail starts (Job)
load_order_header (Process)
load_order_line (Process)
stage_sales_detail (Process)
fact_sales_detail (Process)

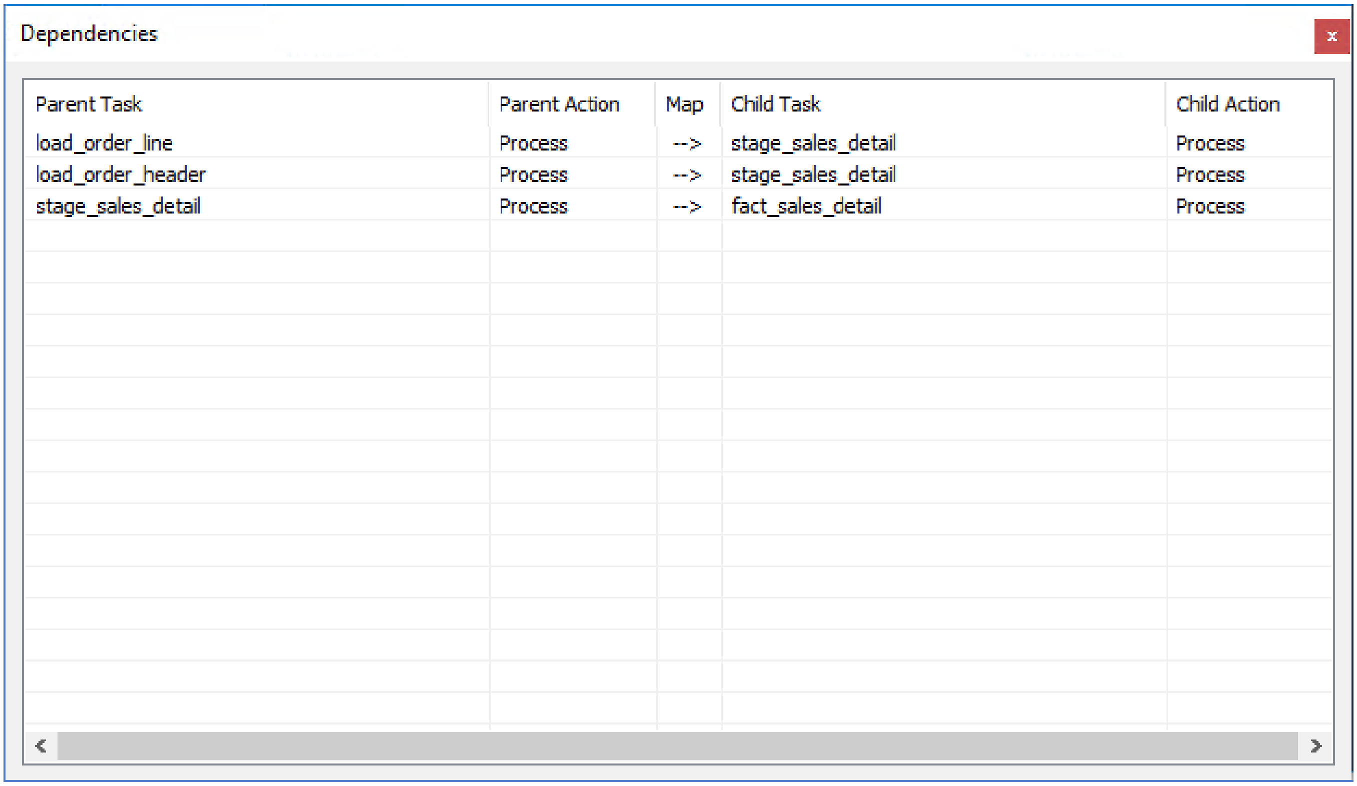
Ready Middle Pane status: Currently no active display in this pane Development (wsdevrepo) UserId: Browse: No source system CAP NUM SCRL INS

Editing a Job's Dependencies from a Job Dependency diagram

Once a Job Dependency diagram has been created for a job, its dependencies can be edited by selecting **Launch Dependency Editor** from the **Tools** menu:



The **Dependencies** window enables you to edit the dependencies within a job:



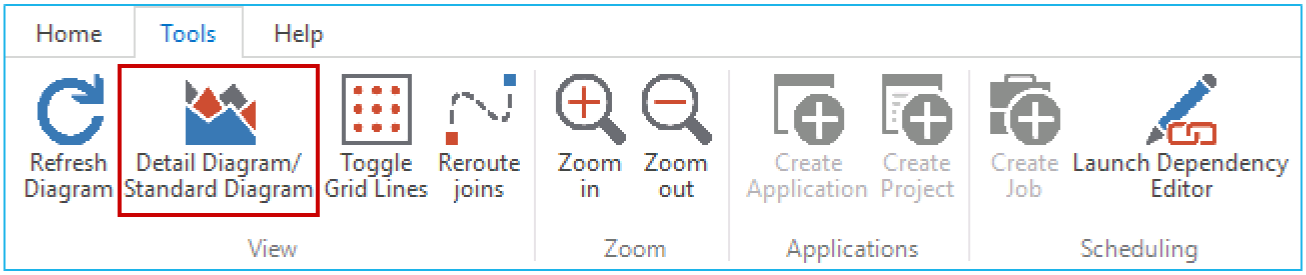
Dependencies

Parent Task	Parent Action	Map	Child Task	Child Action
load_order_line	Process	-->	stage_sales_detail	Process
load_order_header	Process	-->	stage_sales_detail	Process
stage_sales_detail	Process	-->	fact_sales_detail	Process

Working with Diagrams

Diagram Display

Once displayed, there are two modes for diagram display; standard and detailed. To switch between displays, select **File > Detail Diagram / Standard Diagram**.



or use the **toggle** button:

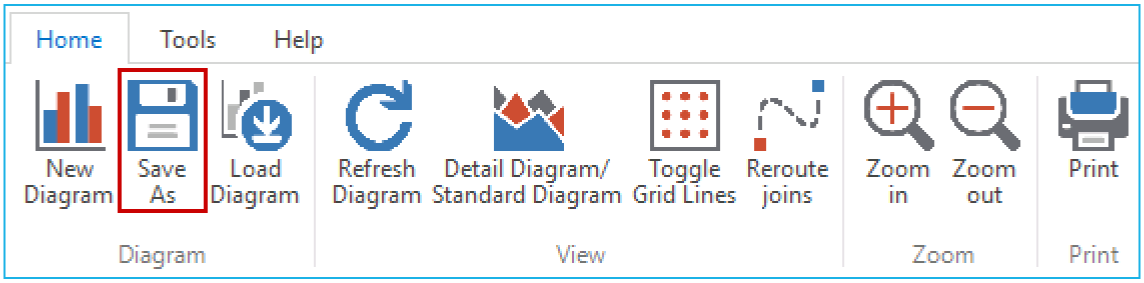
Diagram Save

A diagram can be saved either as a meta file or as a jpeg image. If saved as a meta file, it can be subsequently reloaded and re-edited.

Note

A jpeg cannot be reloaded into WhereScape RED.

The diagram can be saved by selecting **File > Save As ...**



Note

By default, the diagram is saved as a meta file.

Diagram Load

If a diagram has been saved as a meta file, it can be reloaded. To reload a saved meta file, switch to diagrammatic view and then select the menu option **File > Load Diagram**. A window is displayed which enables you to choose a Windows meta file (*.wmf). If the meta file had previously been saved from WhereScape RED then the diagram is loaded.

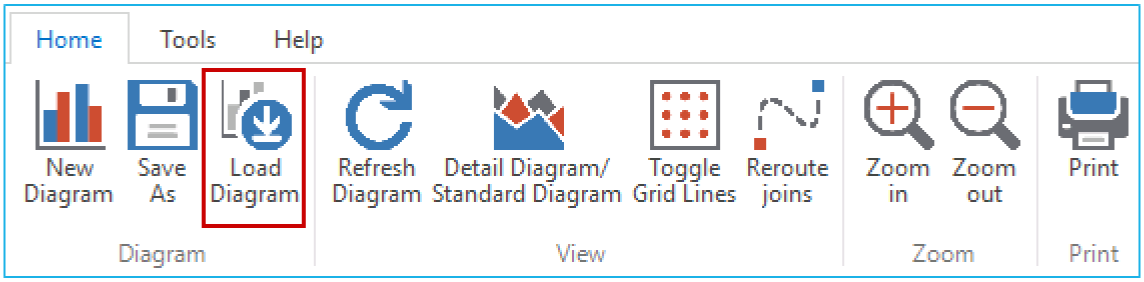


Diagram Print

A diagram can be printed by selecting **File > Print ...**

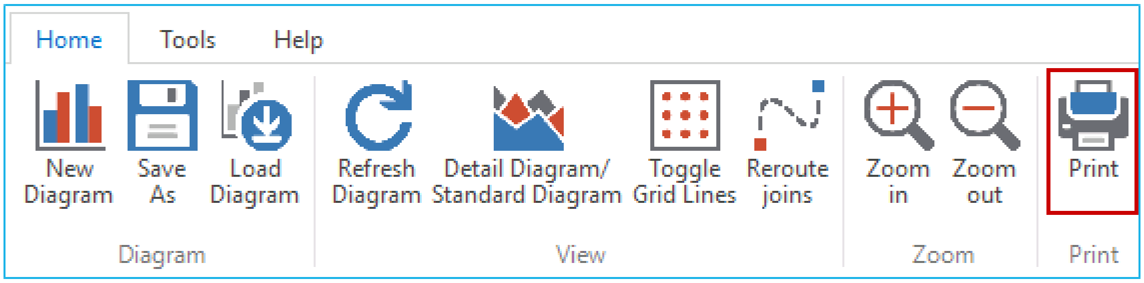
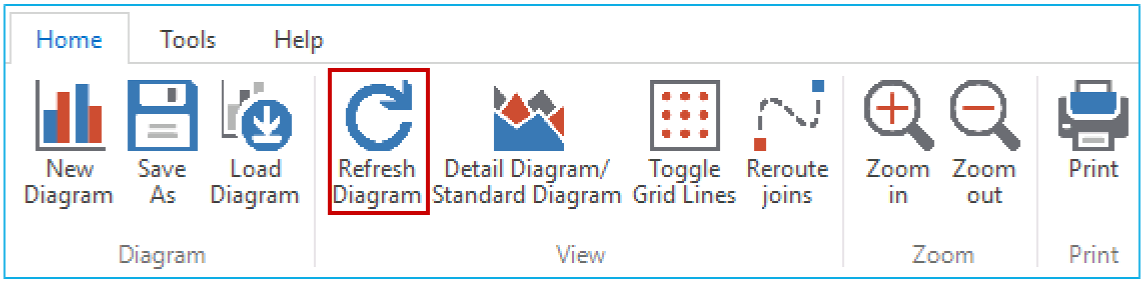


Diagram Refresh

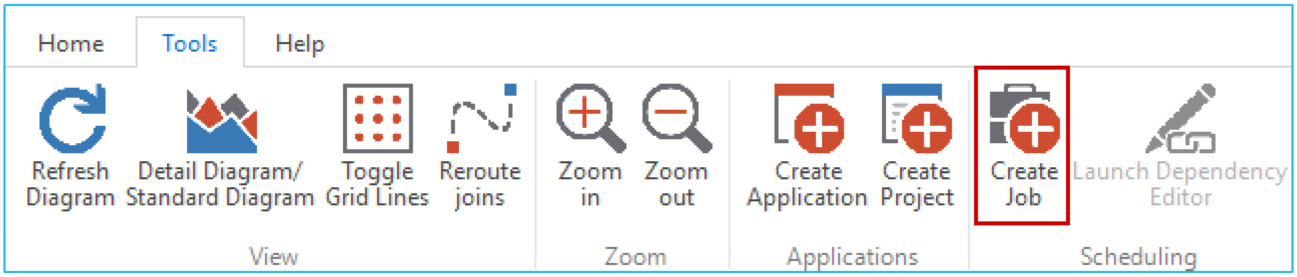
Once a diagram has been displayed, it can be refreshed by choosing **Tools > Refresh Diagram** or by pressing **F5**.



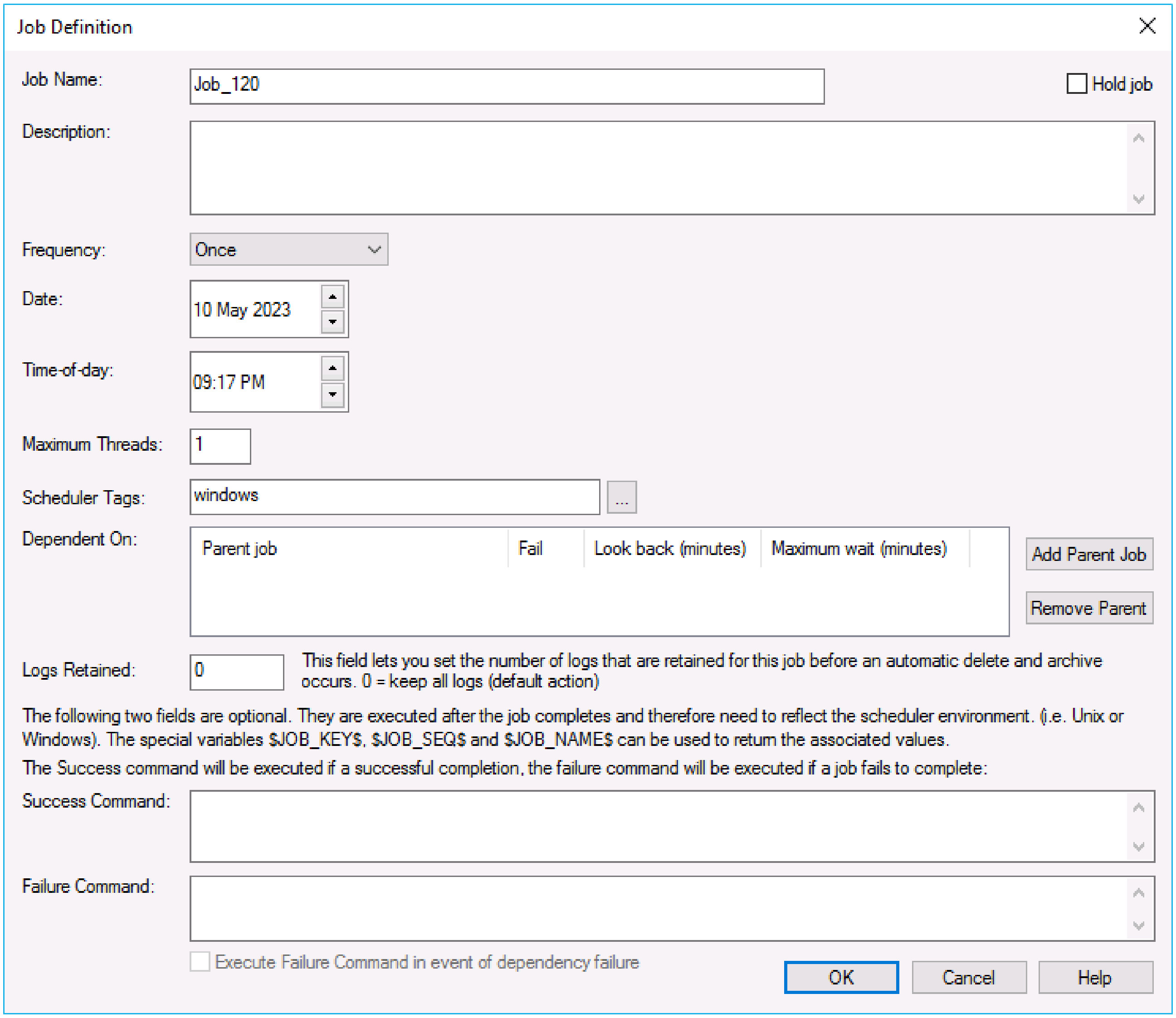
Creating a Job from a Diagram

A job can be created from a **Source** diagram or a **Joins** diagram.

1. Once the diagram is displayed, select **Tools > Create Job**.



2. Edit the job as required and then click **OK**.


Job Definition

Job Name: Job_120 □ Hold job

Description:

Frequency: Once

Date: 10 May 2023

Time-of-day: 09:17 PM

Maximum Threads: 1

Scheduler Tags: windows

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)

Logs Retained: 0 This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

Success Command:

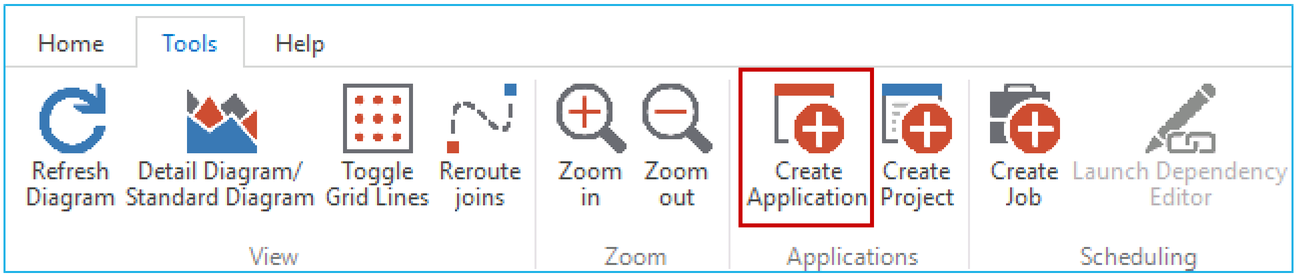
Failure Command:

□ Execute Failure Command in event of dependency failure

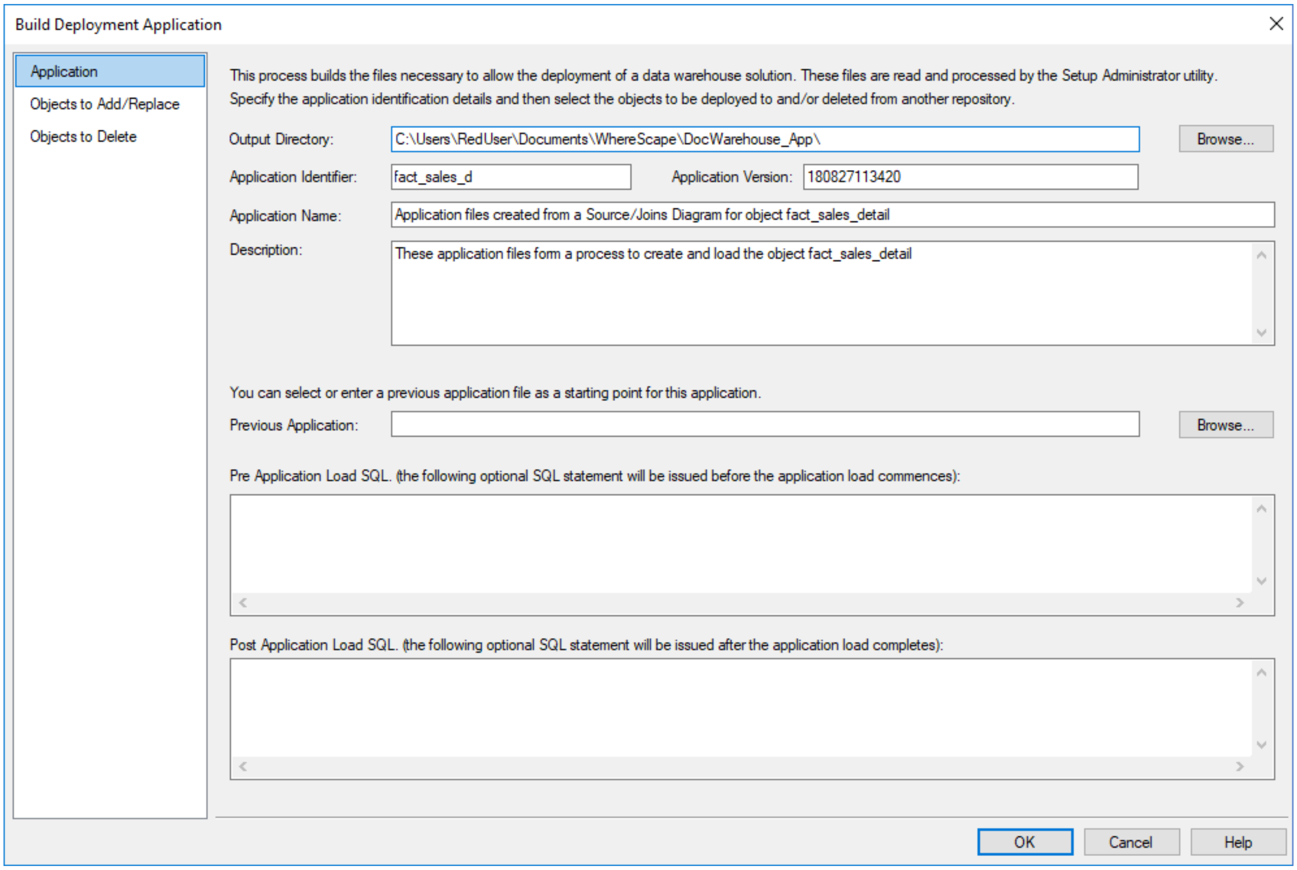
Creating an Application from a Diagram

An application can be created from a **Source** diagram or a **Joins** diagram.

1. Once the diagram is displayed, select **Tools > Create Application**.



2. Edit the application as required and then click **OK**.



This process builds the files necessary to allow the deployment of a data warehouse solution. These files are read and processed by the Setup Administrator utility.
Specify the application identification details and then select the objects to be deployed to and/or deleted from another repository.

Output Directory: C:\Users\RedUser\Documents\WhereScape\DocWarehouse_App\

Application Identifier: fact_sales_d

Application Version: 180827113420

Application Name: Application files created from a Source/Joins Diagram for object fact_sales_detail

Description: These application files form a process to create and load the object fact_sales_detail

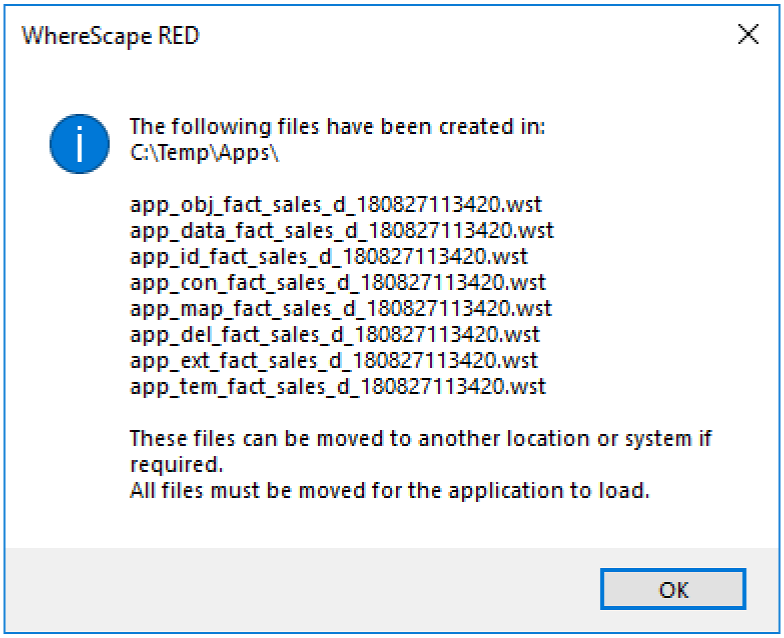
You can select or enter a previous application file as a starting point for this application.

Previous Application:

Pre Application Load SQL. (the following optional SQL statement will be issued before the application load commences):

Post Application Load SQL. (the following optional SQL statement will be issued after the application load completes):

3. A message is displayed, confirming the creation of the application files. Click **OK**.



WhereScape RED

The following files have been created in:
C:\Temp\Apps\

app_obj_fact_sales_d_180827113420.wst
app_data_fact_sales_d_180827113420.wst
app_id_fact_sales_d_180827113420.wst
app_con_fact_sales_d_180827113420.wst
app_map_fact_sales_d_180827113420.wst
app_del_fact_sales_d_180827113420.wst
app_ext_fact_sales_d_180827113420.wst
app_tem_fact_sales_d_180827113420.wst

These files can be moved to another location or system if required.
All files must be moved for the application to load.

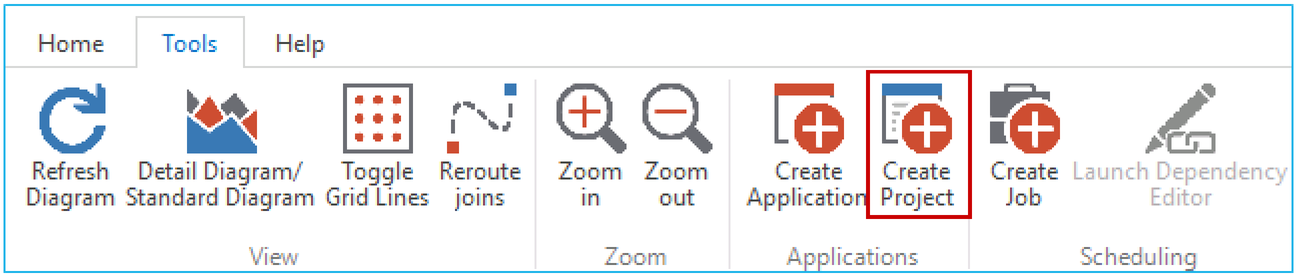
Note

Creating an application from a diagram saves the objects in the diagram and the associated objects, including indexes.

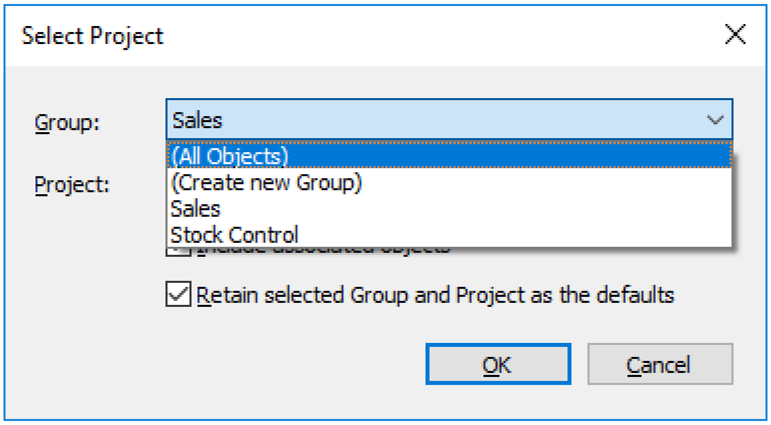
Creating a Project from a Diagram

A project can be created from a **Source** diagram or a **Joins** diagram.

1. Once the diagram is displayed, select **Tools > Create Project**.



2. Select an existing project group or create a new group.



Select Project

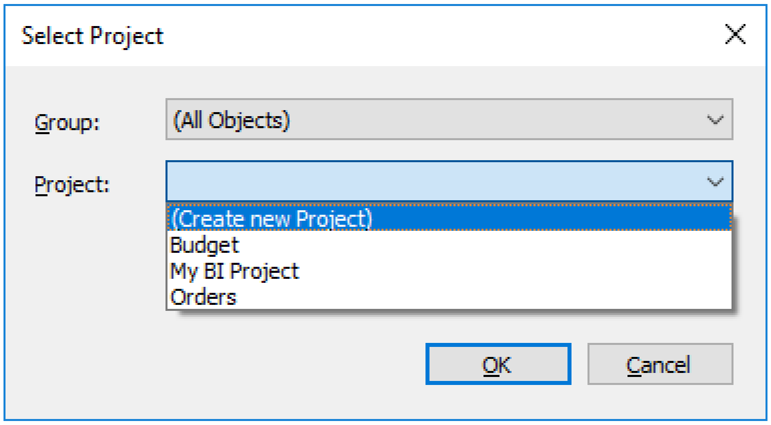
Group: Sales

(All Objects)
(Create new Group)
Sales
Stock Control

Project:

☑ Retain selected Group and Project as the defaults

3. Select **Create new Project**.

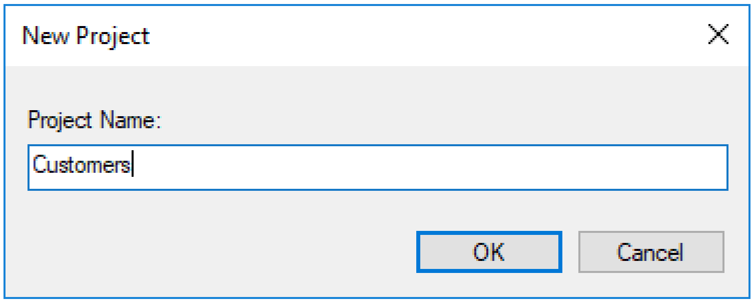

Select Project

Group: (All Objects)

Project:

(Create new Project)
Budget
My BI Project
Orders

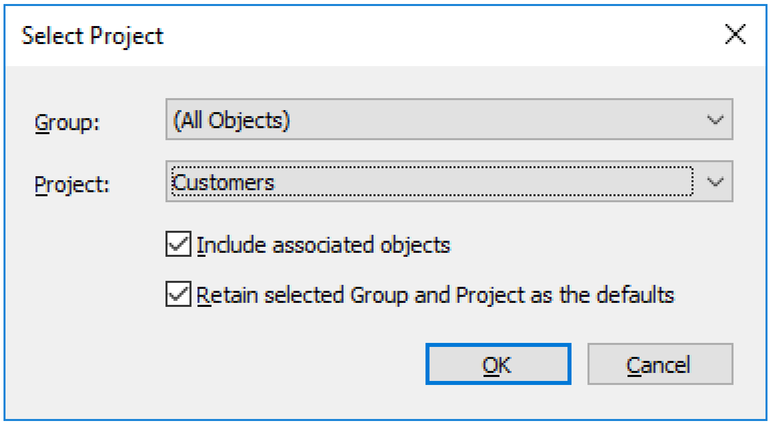
4. Type in the name of the new project and then click **OK**.


New Project

Project Name:

Customers

5. Click **OK** in the **Select Project** window.


Select Project

Group: (All Objects)

Project: Customers

☑ Include associated objects

☑ Retain selected Group and Project as the defaults

The objects in the diagram are moved into the selected project.

If the **Include Associated Objects** check box is selected, all associated procedures, scripts and indexes are included. This option is selected by default.

If the **Retain selected Group and Project as the defaults** check box is selected, the selected Group and Project are retained as the defaults. This option is selected by default.

Query Tool Access

Where the business definition has been set in the properties of either a fact or dimension column, WhereScape RED will attempt to record this information in the database.

For the SQL Server database, the business definition is also stored as a column comment via the sp_addextendedproperty procedure with a property name of **Comment**.

In addition, a number of metadata views exist to make the task of connecting an end user query tool to the data warehouse simpler. Many end user tools can make use of external data to populate their end-user-layer (eul) definitions. Using these views, end user query tools can utilize WhereScape RED's business metadata.

The views supplied are:

Dimension tables and columns

The view **ws_admin_v_dim_col** provides a listing of all columns in all dimensions that have the 'Eul visible' flag set. Included in the information provided is the business definition and description of the column as defined within WhereScape RED.

Fact tables and columns

The view **ws_admin_v_fact_col** provides a listing of all columns in all fact tables that have the 'Eul visible' flag set. Included in the information provided is the business definition and description of the column as defined within WhereScape RED.

Fact - Dimension Joins

The view **ws_admin_v_fact_join** provides a listing of all fact tables and all the dimensions that each fact table joins to. Both the Fact table dimension key column and the dimension key column are included in the view.

Reports

WhereScape RED includes reports for analyzing columns, tables, procedures, indexes, objects and jobs.

When these reports are run, the results are displayed in a separate tab in the bottom pane of the RED screen.

The following sections describe the purpose, parameters and results for each report.

Dimension-Fact Matrix

This report shows dimension tables used by each fact and aggregate table in the metadata as a matrix.

Objects Included

The following WhereScape RED object types are included in this report:

- Dimension Tables
- Fact Tables
- Aggregate Tables

Parameters

There are no parameters for this report.

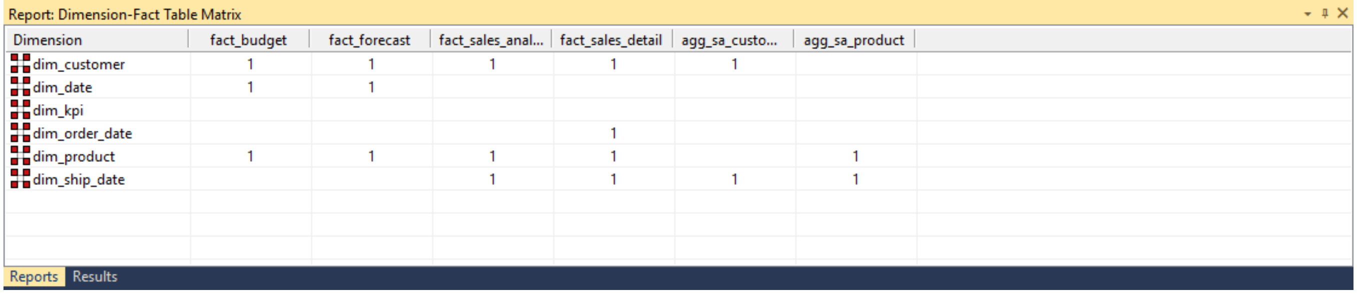
Results

The results of this report are displayed as a list of refreshed objects in the metadata repository with the following columns:

- Dimensions *(the dimension name)*
- Fact/Aggregate Table 1
- Fact/Aggregate Table 2
- Fact/Aggregate Table n

The cells in the cross tab have a 1 to indicate the dimension is used by the fact or aggregate table and blank otherwise. The result set is not sortable.

Report Example



Dimension	fact_budget	fact_forecast	fact_sales_anal...	fact_sales_detail	agg_sa_custo...	agg_sa_product
dim_customer	1	1	1	1	1	
dim_date	1	1				
dim_kpi						
dim_order_date				1		
dim_product	1	1	1	1		1
dim_ship_date			1	1	1	1

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

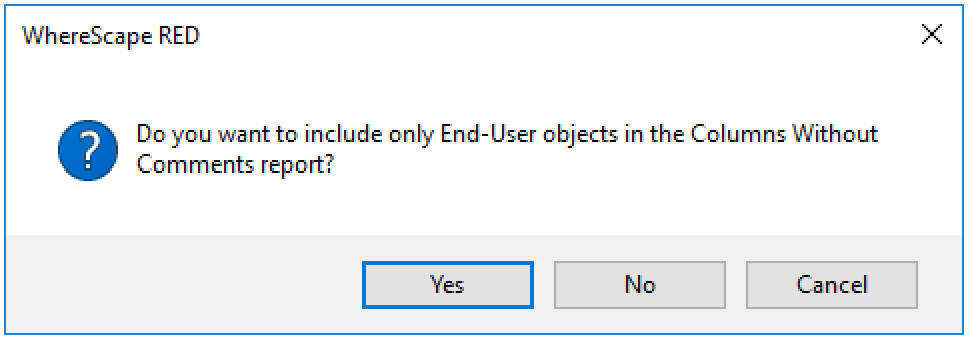
Column Reports

There are five reports for analyzing Columns:

- Columns Without Comments
- All Column Transformations
- Re-Usable Column Transformations
- Column Track-Back
- Column Track-Forward

Columns without Comments

This report shows **table** object columns in the metadata that don't have descriptions. The report prompts the user with the option to only display user facing table object columns or all table object columns.



WhereScape RED

Do you want to include only End-User objects in the Columns Without Comments report?

Objects Included

The following WhereScape RED object types are included in this report:

- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Hubs, Satellites and Links

Note

The EUL object are set in **Tools > Options > Object Types > Object Type End User Setting.**

Parameters

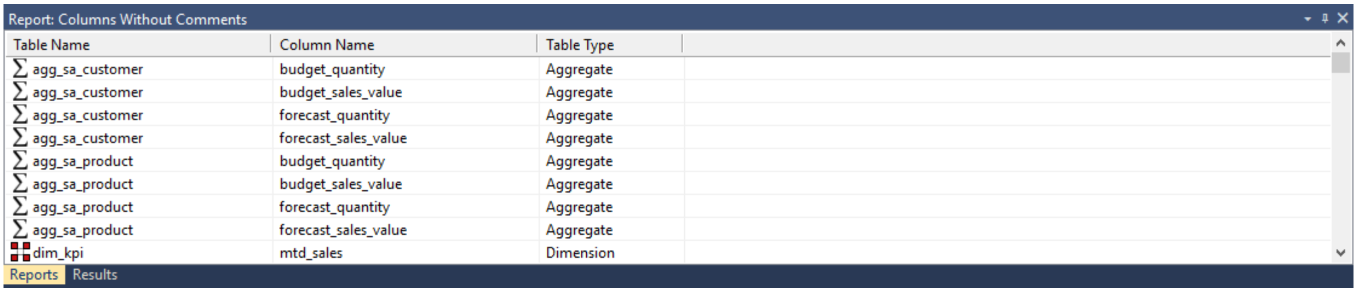
There are no parameters for this report.

Results

The results of this report are displayed as a list of objects and columns in the metadata that are missing comments with the following columns:

- Table name *(the name of the table)*
- Column Name
- Table type

Report Example



Report: Columns Without Comments

Table Name	Column Name	Table Type
∑ agg_sa_customer	budget_quantity	Aggregate
∑ agg_sa_customer	budget_sales_value	Aggregate
∑ agg_sa_customer	forecast_quantity	Aggregate
∑ agg_sa_customer	forecast_sales_value	Aggregate
∑ agg_sa_product	budget_quantity	Aggregate
∑ agg_sa_product	budget_sales_value	Aggregate
∑ agg_sa_product	forecast_quantity	Aggregate
∑ agg_sa_product	forecast_sales_value	Aggregate
dim_kpi	mtd_sales	Dimension

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

All Column Transformations

This report shows all columns that have a column transformation and the details of the transformation.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Exports
- Hubs, Satellites and Links

Parameters

There are no parameters for this report.

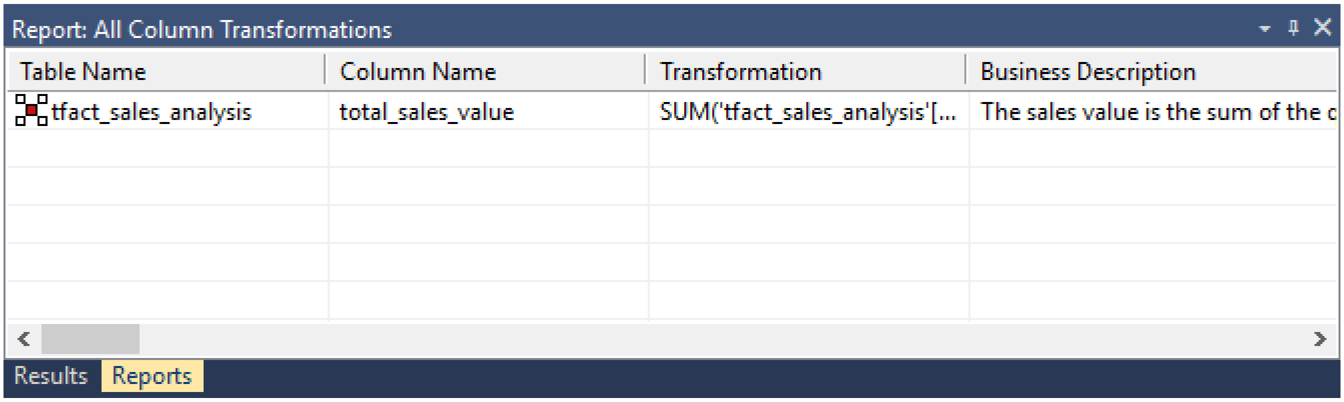
Results

The results of this report are displayed as a list of objects in the metadata repository with the following columns:

- Table name *(the name of the table)*
- Column name
- Transformation

The result set is sortable by clicking on the appropriate column heading.

Report Example



Report: All Column Transformations

Table Name	Column Name	Transformation	Business Description
tfact_sales_analysis	total_sales_value	SUM('tfact_sales_analysis'[...	The sales value is the sum of the c

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Re-Usable Column Transformations

This report shows all reusable transformations as defined via **Tools > Define Re-usable Transformations**.

Parameters

There are no parameters for this report.

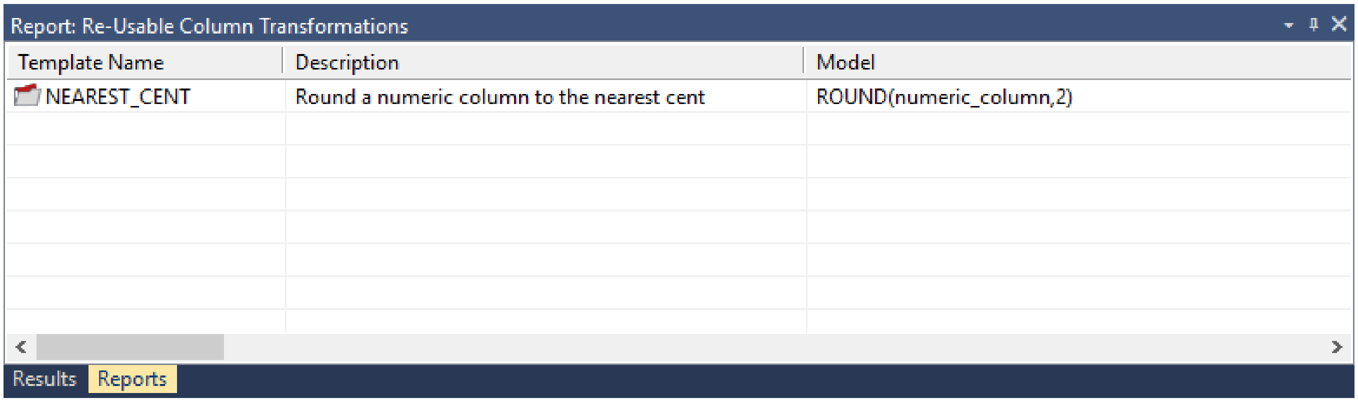
Results

The results of this report are displayed as a list of Re-Usable transformations with the following columns :

- Template Name
- Description

The result set is not sortable.

Report Example



Report: Re-Usable Column Transformations

Template Name	Description	Model
NEAREST_CENT	Round a numeric column to the nearest cent	ROUND(numeric_column,2)

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Column Track-Back

This report lists the origins of the selected objects.

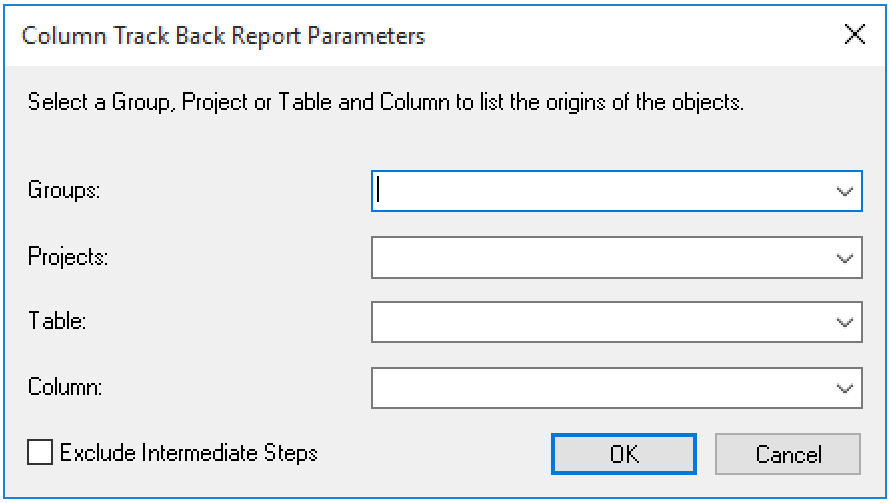
Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Exports
- Hubs, Satellites and Links

Parameters

- Groups
- Projects
- Table
- Column



Column Track Back Report Parameters

Select a Group, Project or Table and Column to list the origins of the objects.

Groups:

Projects:

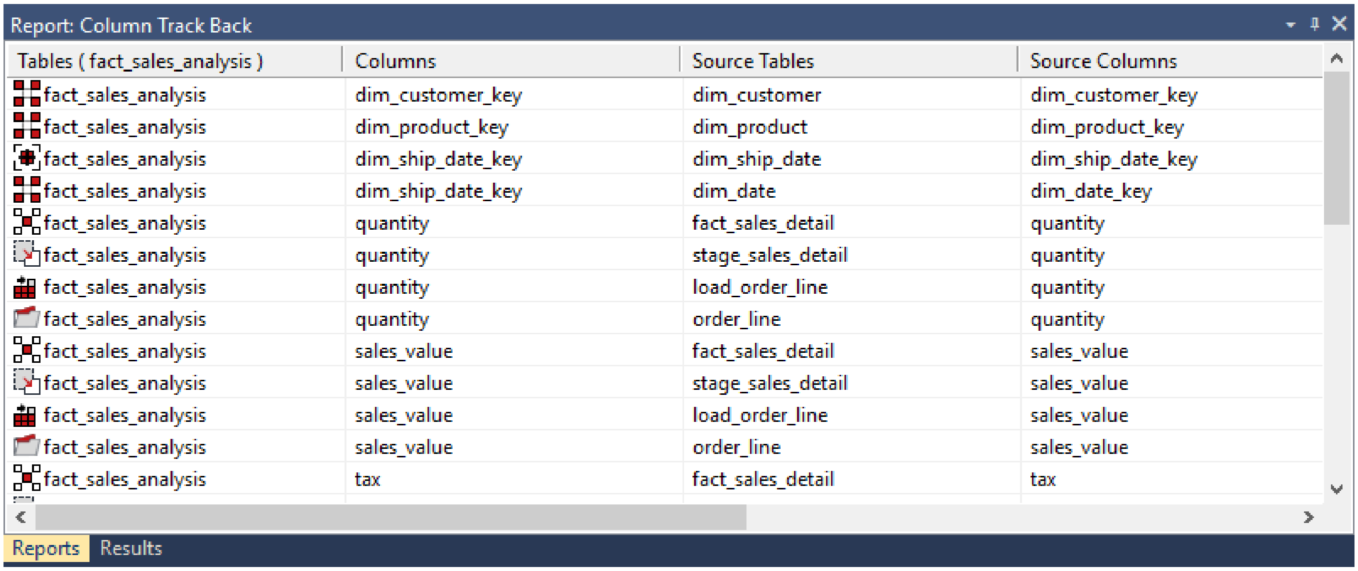
Table:

Column:

□ Exclude Intermediate Steps

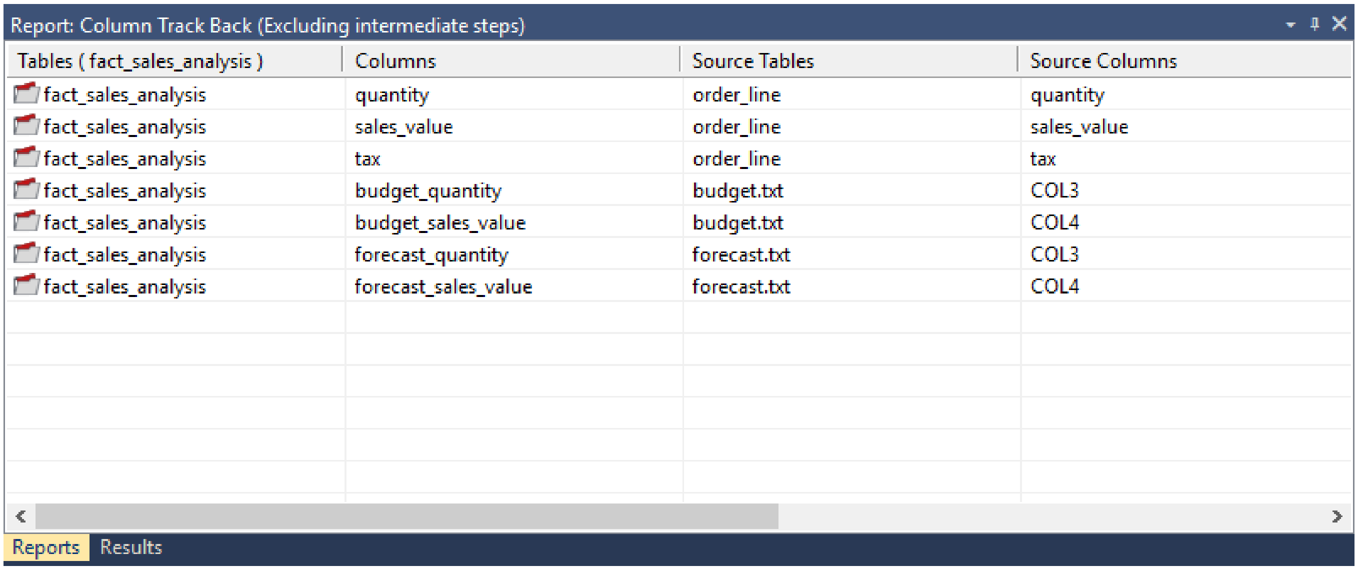
Results

If you left the **Exclude Intermediate Steps** check box **unchecked**, then the results screen is as follows, showing the line of origins for the selected tables:



Tables (fact_sales_analysis)	Columns	Source Tables	Source Columns
fact_sales_analysis	dim_customer_key	dim_customer	dim_customer_key
fact_sales_analysis	dim_product_key	dim_product	dim_product_key
fact_sales_analysis	dim_ship_date_key	dim_ship_date	dim_ship_date_key
fact_sales_analysis	dim_ship_date_key	dim_date	dim_date_key
fact_sales_analysis	quantity	fact_sales_detail	quantity
fact_sales_analysis	quantity	stage_sales_detail	quantity
fact_sales_analysis	quantity	load_order_line	quantity
fact_sales_analysis	quantity	order_line	quantity
fact_sales_analysis	sales_value	fact_sales_detail	sales_value
fact_sales_analysis	sales_value	stage_sales_detail	sales_value
fact_sales_analysis	sales_value	load_order_line	sales_value
fact_sales_analysis	sales_value	order_line	sales_value
fact_sales_analysis	tax	fact_sales_detail	tax

If however, you selected **Exclude Intermediate Steps**, then the results screen is as follows, showing only the original source table for the selected tables:



Tables (fact_sales_analysis)	Columns	Source Tables	Source Columns
fact_sales_analysis	quantity	order_line	quantity
fact_sales_analysis	sales_value	order_line	sales_value
fact_sales_analysis	tax	order_line	tax
fact_sales_analysis	budget_quantity	budget.txt	COL3
fact_sales_analysis	budget_sales_value	budget.txt	COL4
fact_sales_analysis	forecast_quantity	forecast.txt	COL3
fact_sales_analysis	forecast_sales_value	forecast.txt	COL4

The results of this report are displayed as a list of source tables and columns, the order of the result set determining the immediate lineage. The report includes the following columns:

- Tables *(the name of a selected table)*
- Columns *(the name of a selected column)*
- Source Tables
- Source Columns

The result set is **not** sortable, as the order of the result set determines the immediate lineage.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Column Track-Forward

This report lists the columns derived from the selected objects.

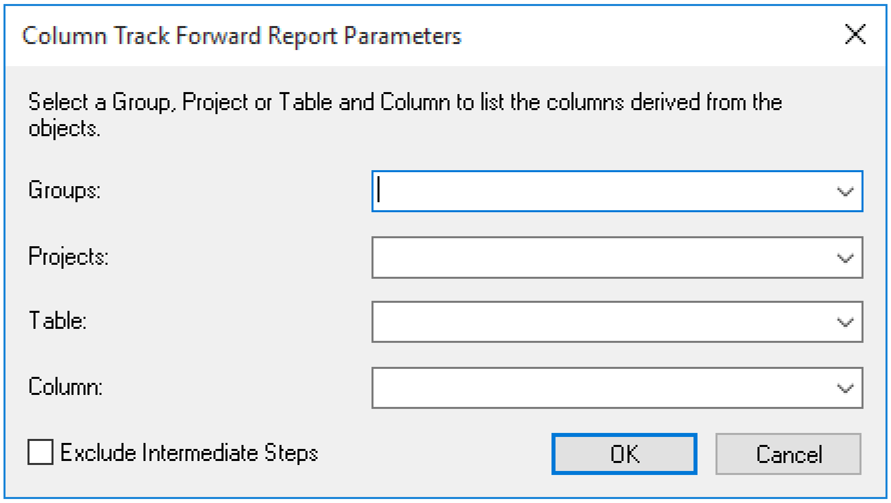
Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Exports
- Hubs, Satellites and Links

Parameters

- Groups
- Projects
- Table
- Column



Column Track Forward Report Parameters

Select a Group, Project or Table and Column to list the columns derived from the objects.

Groups:

Projects:

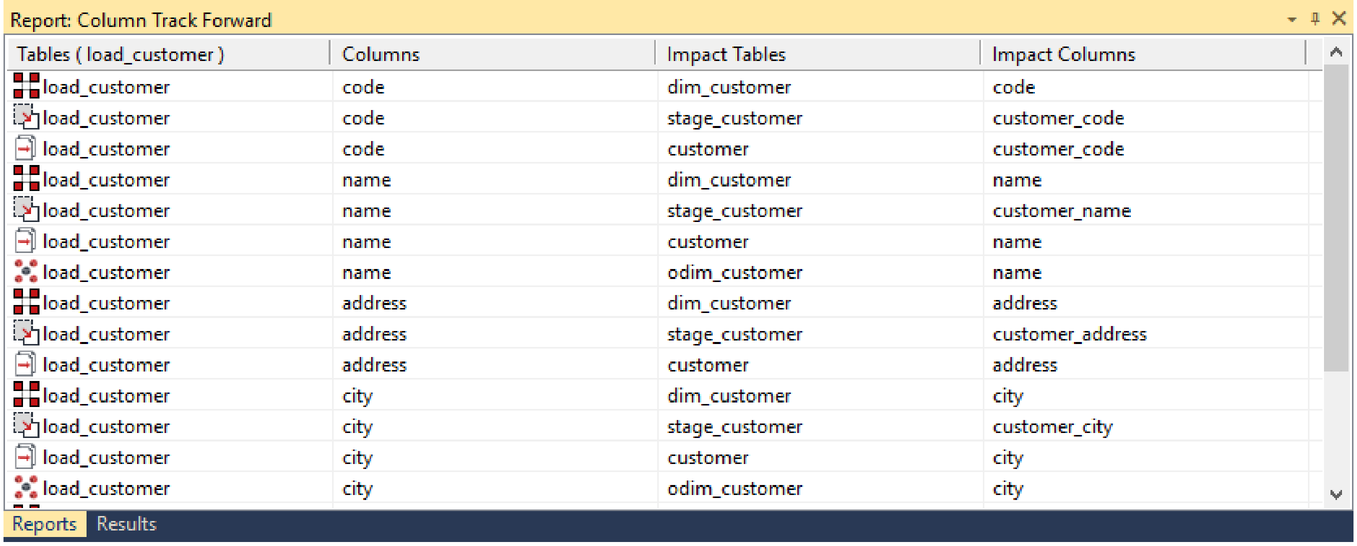
Table:

Column:

□ Exclude Intermediate Steps

Results

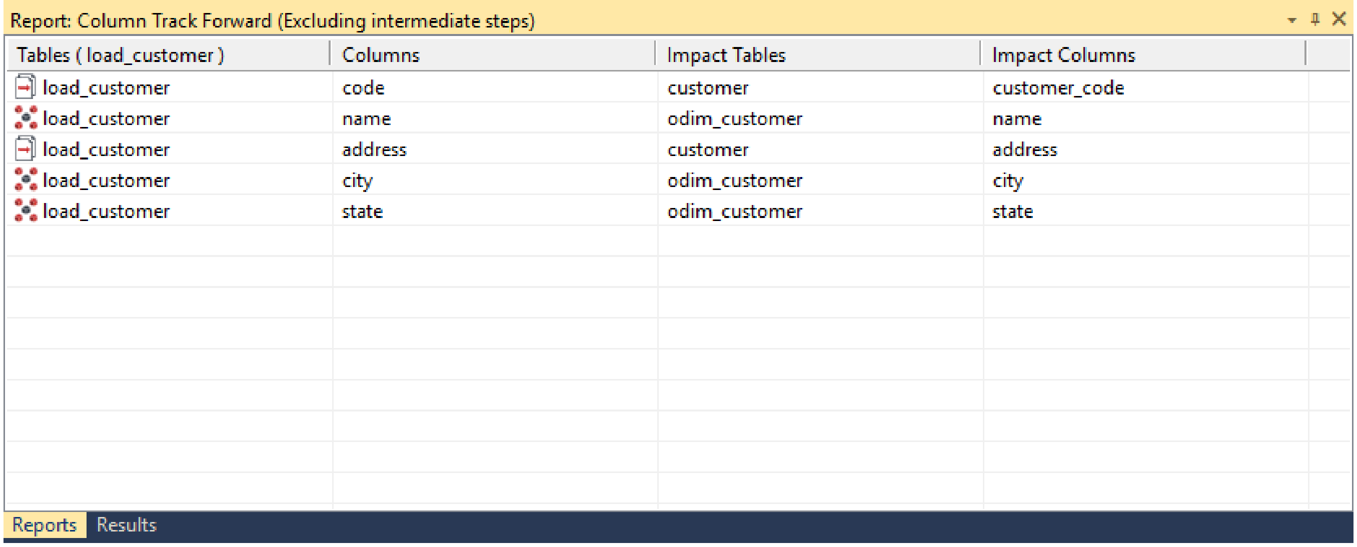
If you left the **Exclude Intermediate Steps** checkbox **unchecked** then the results screen is as follows, showing the impacted tables for the selected tables:



Report: Column Track Forward

Tables (load_customer)	Columns	Impact Tables	Impact Columns
load_customer	code	dim_customer	code
load_customer	code	stage_customer	customer_code
load_customer	code	customer	customer_code
load_customer	name	dim_customer	name
load_customer	name	stage_customer	customer_name
load_customer	name	customer	name
load_customer	name	odim_customer	name
load_customer	address	dim_customer	address
load_customer	address	stage_customer	customer_address
load_customer	address	customer	address
load_customer	city	dim_customer	city
load_customer	city	stage_customer	customer_city
load_customer	city	customer	city
load_customer	city	odim_customer	city

If however, you selected **Exclude Intermediate Steps** then the results screen is as follows, showing only the final impacted table for the selected tables:


Report: Column Track Forward (Excluding intermediate steps)

Tables (load_customer)	Columns	Impact Tables	Impact Columns
load_customer	code	customer	customer_code
load_customer	name	odim_customer	name
load_customer	address	customer	address
load_customer	city	odim_customer	city
load_customer	state	odim_customer	state

The results of this report are displayed as a list of impacted tables and columns, the order of the result set determining the immediate impact. The report includes the following columns:

- Tables *(the names of selected tables)*
- Columns *(the names of selected columns)*
- Impact Tables
- Impact Columns

The result set is **not** sortable, as the order of the result set determines the immediate lineage.

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

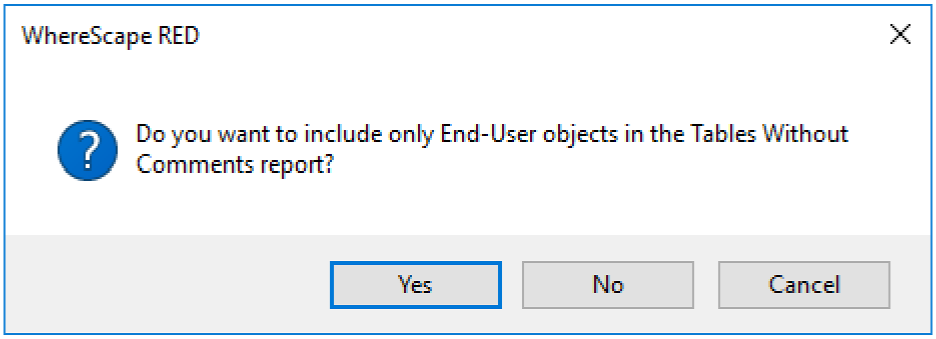
Table Reports

There are five reports for analyzing Tables:

- Tables Without Comments
- Load Tables by Connection
- Export Objects by Connection
- Records that Failed a Dimension Join
- External source Table/Files

Tables without Comments

This report shows **table** objects in the metadata that don't have descriptions. The report prompts the user with the option to only display user facing table objects or all table objects.


WhereScape RED

Do you want to include only End-User objects in the Tables Without Comments report?

Objects Included

The following WhereScape RED object types are included in this report:

- All Tables objects or
- Only End User Layer (EUL) objects

Note

The EUL object are set in **Tools > Options > Object Types > Object Type End User Setting.**

Parameters

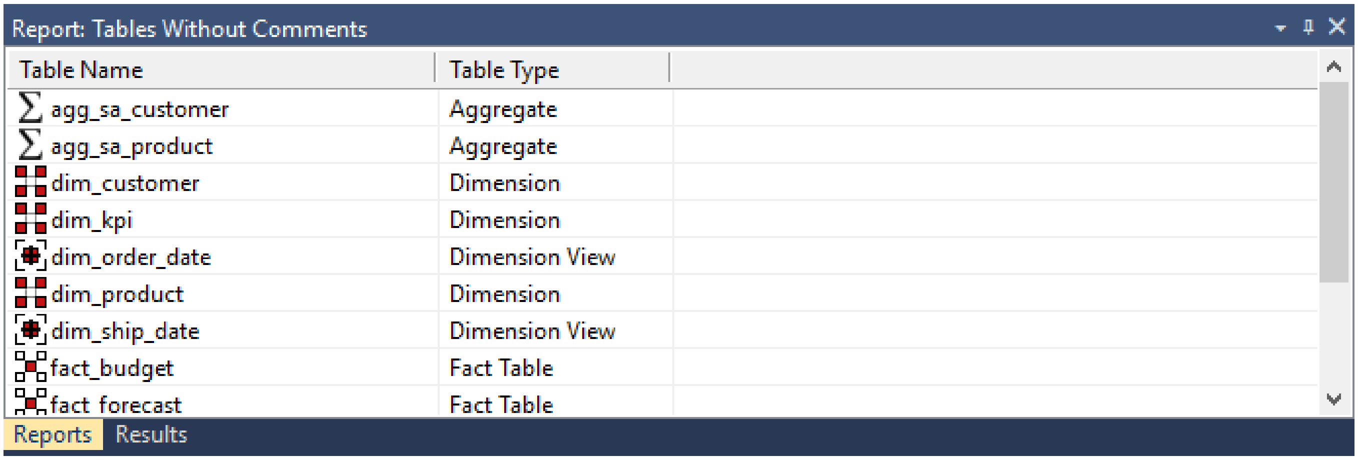
There are no parameters for this report.

Results

The results of this report are displayed as a list of objects in the metadata missing comments with the following columns:

- Table name *(the name of the table)*
- Table type

Report Example



Report: Tables Without Comments

Table Name	Table Type
agg_sa_customer	Aggregate
agg_sa_product	Aggregate
dim_customer	Dimension
dim_kpi	Dimension
dim_order_date	Dimension View
dim_product	Dimension
dim_ship_date	Dimension View
fact_budget	Fact Table
fact_forecast	Fact Table

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Load Tables by Connection

This report shows load tables in the metadata repository with their Connection and Source schema or database.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables

Parameters

There are no parameters for this report.

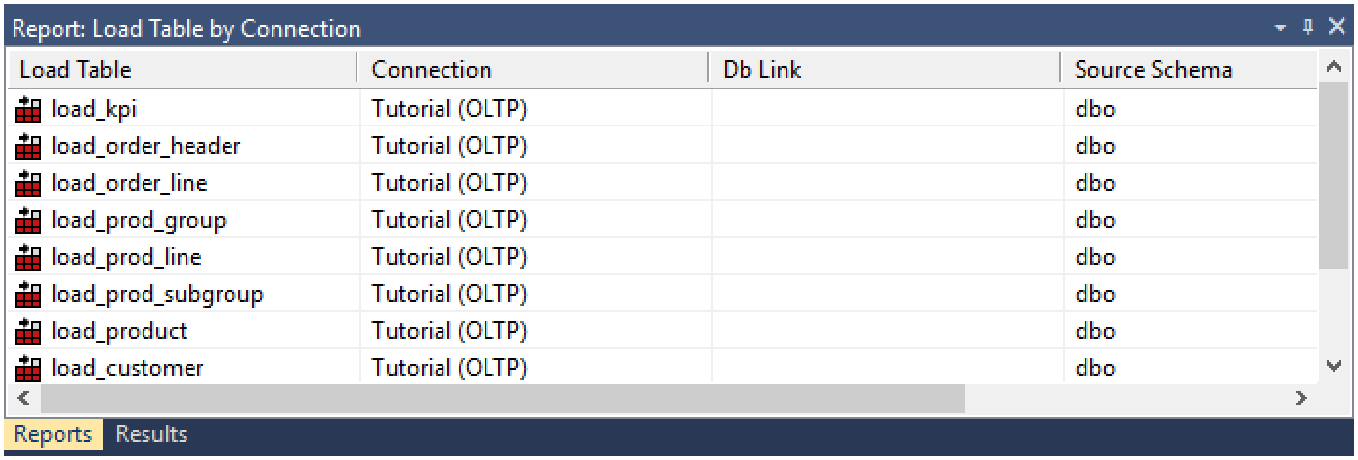
Results

The results of this report are displayed as a list of objects in the metadata repository with the following columns:

- Load table *(the name of the load table)*
- Connections
- Source Schema *(the name the database or schema the load table is source from - blank for files)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Report: Load Table by Connection

Load Table	Connection	Db Link	Source Schema
load_kpi	Tutorial (OLTP)		dbo
load_order_header	Tutorial (OLTP)		dbo
load_order_line	Tutorial (OLTP)		dbo
load_prod_group	Tutorial (OLTP)		dbo
load_prod_line	Tutorial (OLTP)		dbo
load_prod_subgroup	Tutorial (OLTP)		dbo
load_product	Tutorial (OLTP)		dbo
load_customer	Tutorial (OLTP)		dbo

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Export Objects by Connection

This report shows export tables in the metadata repository with their Connection and Source schema or database.

Objects Included

The following WhereScape RED object types are included in this report:

- Export Tables

Parameters

There are no parameters for this report.

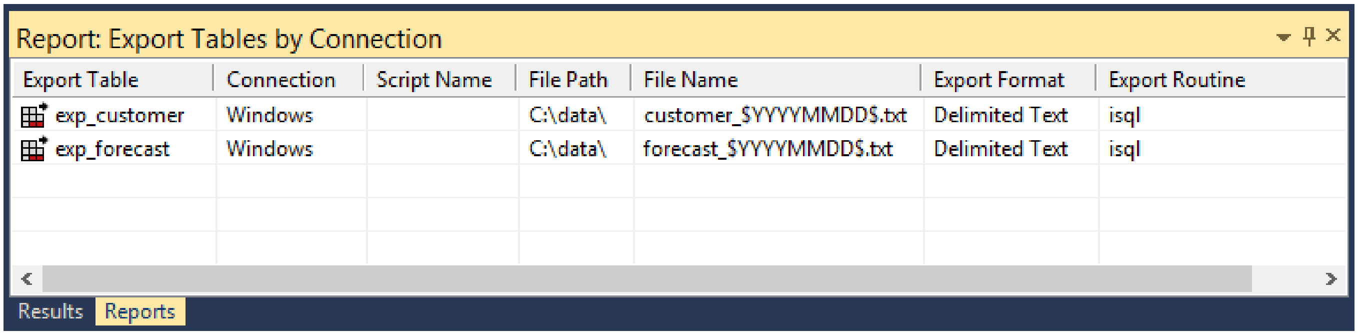
Results

The results of this report are displayed as a list of objects in the metadata repository with the following columns:

- Export Table (the name of the export table)
- Connection
- Script Name
- File Path
- File Name
- Export Format
- Export Routine

The result set is sortable by clicking on the appropriate column heading.

Report Example



Report: Export Tables by Connection

Export Table	Connection	Script Name	File Path	File Name	Export Format	Export Routine
exp_customer	Windows		C:\data\	customer_$YYYYMMDD$.txt	Delimited Text	isql
exp_forecast	Windows		C:\data\	forecast_$YYYYMMDD$.txt	Delimited Text	isql

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Records that failed a Dimension Join

This report shows the dimension business key(s) that could not be found in a specified dimension when a specified staging table was last updated. This report will show null values, blank values and business keys not found in the dimension.

Objects Included

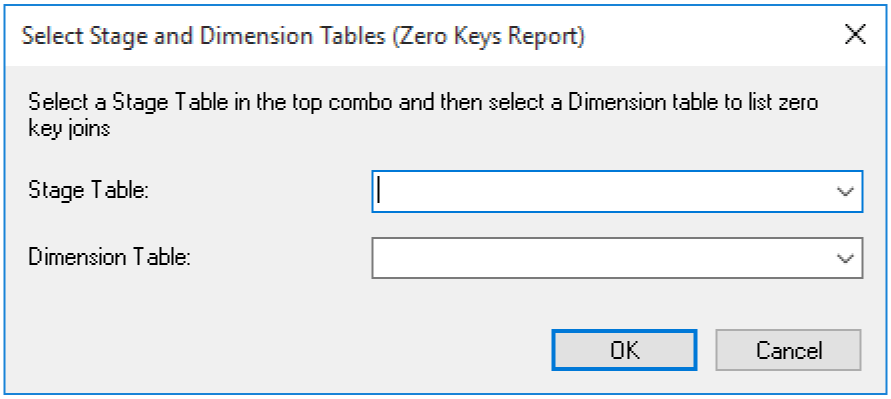
The following WhereScape RED object types are included in this report:

- Stage Tables

Parameters

This report requires two parameters to be specified:

- Stage Table Name *(the staging table to be checked)*
- Dimension Table Name *(the dimension table of the dimension key in the staging table selected first)*



Select Stage and Dimension Tables (Zero Keys Report)

Select a Stage Table in the top combo and then select a Dimension table to list zero key joins

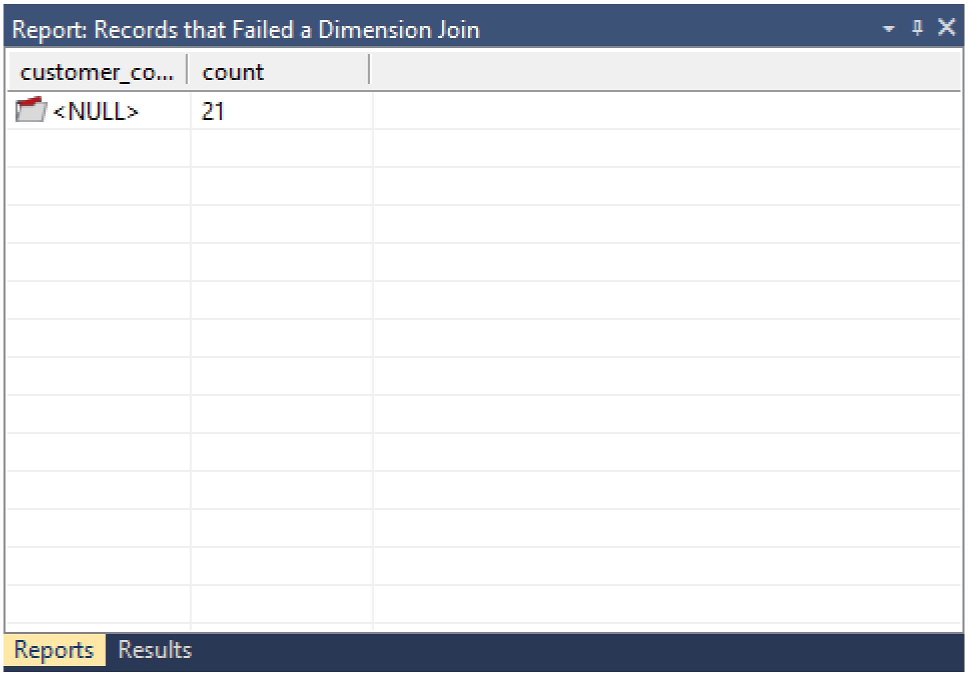
Stage Table:

Dimension Table:

Results

The report contains a list of values not found in the dimension and a count for each value. Each column is sortable by clicking on the appropriate column heading.

Report Example



customer_co...	count
<NULL>	21

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

External Source Tables/Files

This report shows external sources for load tables in the metadata repository.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables

Parameters

There are no parameters for this report.

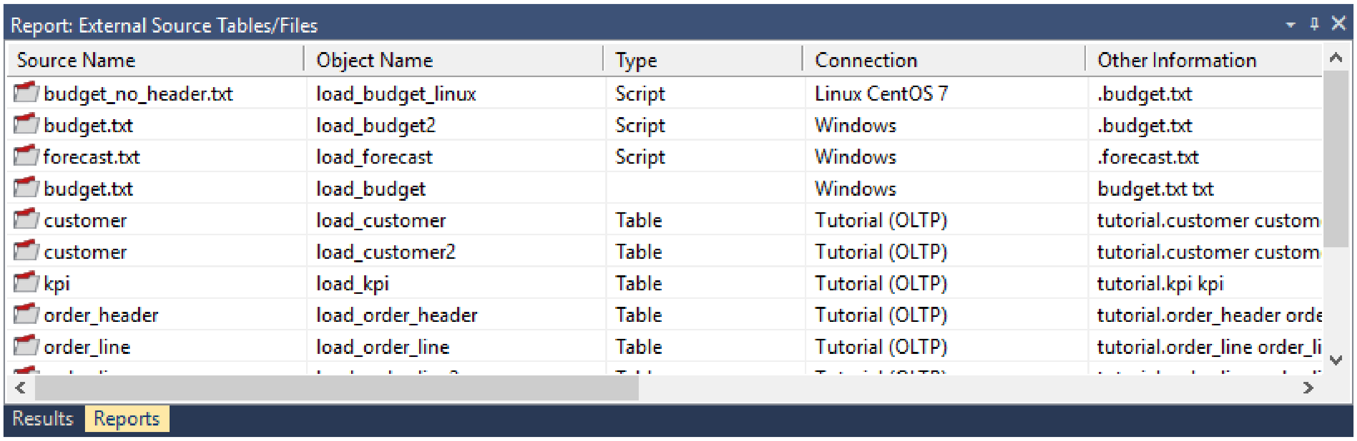
Results

The results of this report are displayed as a list of objects in the metadata repository with the following columns:

- Source name *(the name of the object's source)*
- Object name *(the name of the object)*
- Type *(the type of object the source is: Table or File)*
- Connection
- Other information *(for tables, the source schema/database.source table; for files, the file name)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Source Name	Object Name	Type	Connection	Other Information
budget_no_header.txt	load_budget_linux	Script	Linux CentOS 7	.budget.txt
budget.txt	load_budget2	Script	Windows	.budget.txt
forecast.txt	load_forecast	Script	Windows	.forecast.txt
budget.txt	load_budget		Windows	budget.txt txt
customer	load_customer	Table	Tutorial (OLTP)	tutorial.customer custom
customer	load_customer2	Table	Tutorial (OLTP)	tutorial.customer custom
kpi	load_kpi	Table	Tutorial (OLTP)	tutorial.kpi kpi
order_header	load_order_header	Table	Tutorial (OLTP)	tutorial.order_header orde
order_line	load_order_line	Table	Tutorial (OLTP)	tutorial.order_line order_li

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Procedure Reports

There are two reports for analyzing Procedures:

- Modified Procedures
- Custom Procedures

Modified Procedures

This report shows **modified** procedures in the metadata repository with their creation and modification dates.

Objects Included

The following WhereScape RED object types are included in this report:

- Procedures

Parameters

There are no parameters for this report.

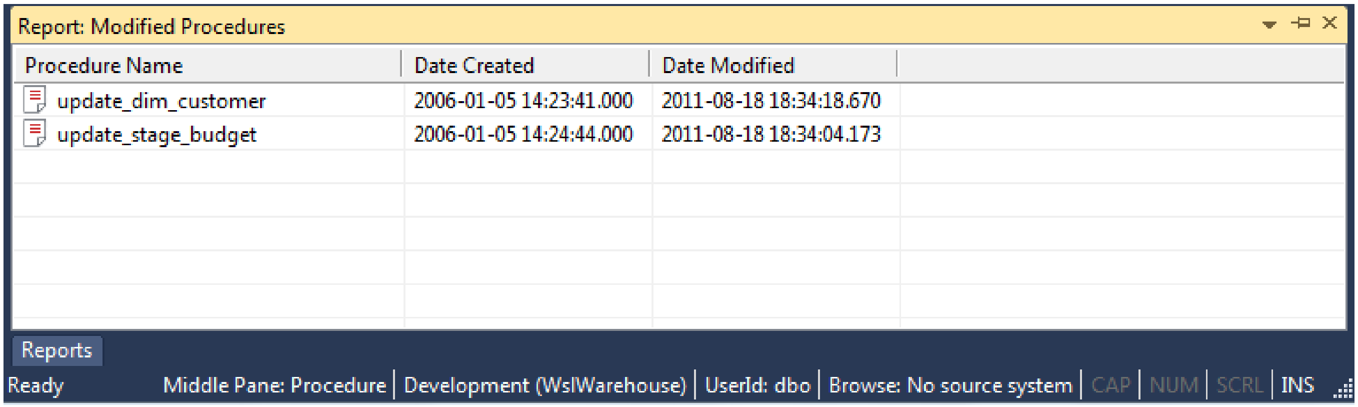
Results

The results of this report are displayed as a list of modified procedures in the metadata repository with the following columns:

- Procedure Name *(the name of the object)*
- Dated Created
- Date Modified *(last modification date)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Procedure Name	Date Created	Date Modified
update_dim_customer	2006-01-05 14:23:41.000	2011-08-18 18:34:18.670
update_stage_budget	2006-01-05 14:24:44.000	2011-08-18 18:34:04.173

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Custom Procedures

This report shows **custom** procedures in the metadata repository with their creation and modification dates.

Note

Custom procedures are procedures generated for any table object as a **Custom Procedure**. Custom procedures are generated via a template.

Objects Included

The following WhereScape RED object types are included in this report:

- Procedures

Parameters

There are no parameters for this report.

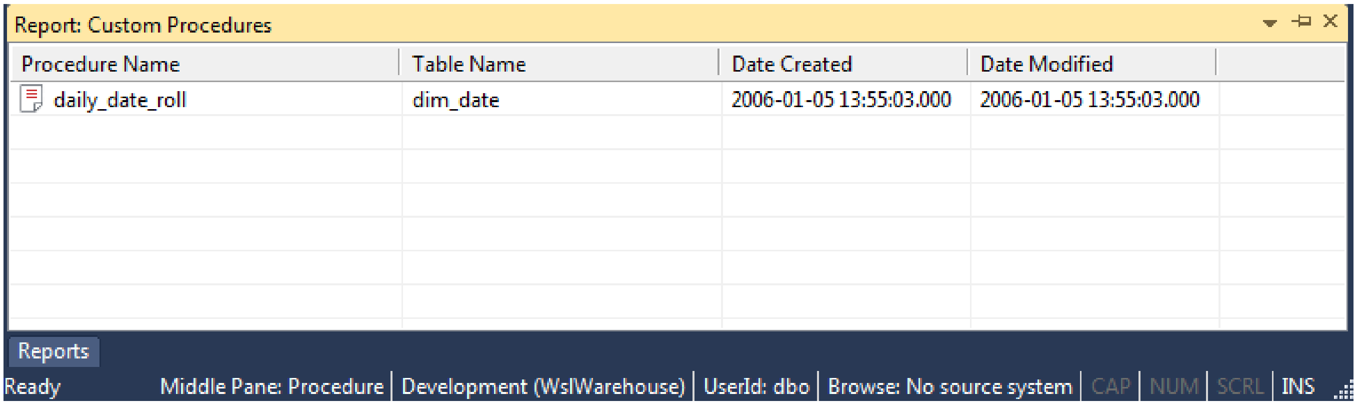
Results

The results of this report are displayed as a list of custom procedures in the metadata repository with the following columns:

- Procedure Name *(the name of the object)*
- Table Name *(the table object the procedure is attached to)*
- Dated Created
- Date Modified *(last modification date)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Report: Custom Procedures

Procedure Name	Table Name	Date Created	Date Modified
daily_date_roll	dim_date	2006-01-05 13:55:03.000	2006-01-05 13:55:03.000

Sending Results to Microsoft Excel

Right-click the result set and select **Output to File** to send the results of this report to Microsoft Excel.

Index Reports

There is one report for analyzing Indexes:

- Modified Indexes

Modified Indexes

This report shows indexes in the metadata repository with their creation and modification dates.

Objects Included

The following WhereScape RED object types are included in this report:

- Indexes

Parameters

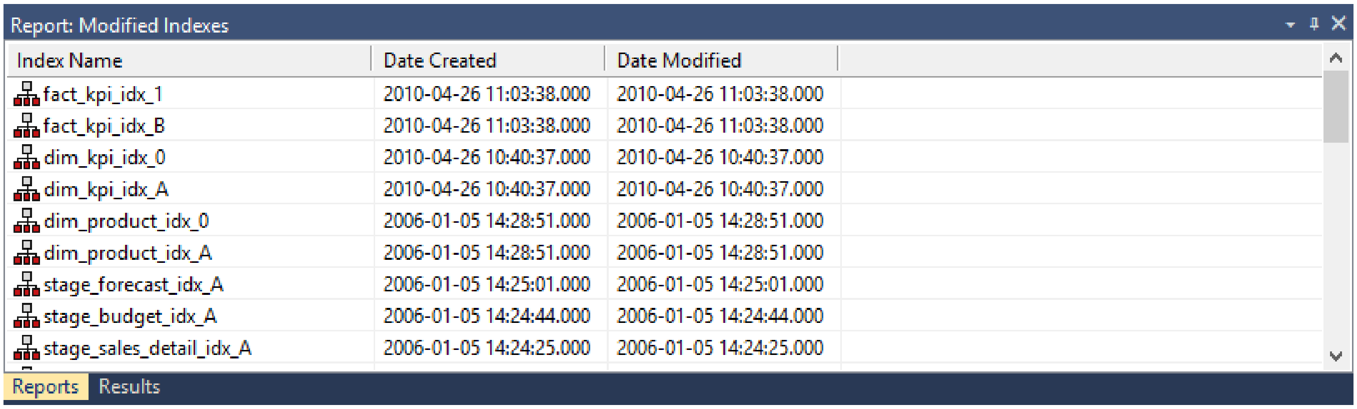
There are no parameters for this report.

Results

- The results of this report are displayed as a list of indexes in the metadata repository with the following columns:
- Index Name *(the name of the index)*
- Dated Created
- Date Modified *(last modification date)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Report: Modified Indexes

Index Name	Date Created	Date Modified
fact_kpi_idx_1	2010-04-26 11:03:38.000	2010-04-26 11:03:38.000
fact_kpi_idx_B	2010-04-26 11:03:38.000	2010-04-26 11:03:38.000
dim_kpi_idx_0	2010-04-26 10:40:37.000	2010-04-26 10:40:37.000
dim_kpi_idx_A	2010-04-26 10:40:37.000	2010-04-26 10:40:37.000
dim_product_idx_0	2006-01-05 14:28:51.000	2006-01-05 14:28:51.000
dim_product_idx_A	2006-01-05 14:28:51.000	2006-01-05 14:28:51.000
stage_forecast_idx_A	2006-01-05 14:25:01.000	2006-01-05 14:25:01.000
stage_budget_idx_A	2006-01-05 14:24:44.000	2006-01-05 14:24:44.000
stage_sales_detail_idx_A	2006-01-05 14:24:25.000	2006-01-05 14:24:25.000

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Object Reports

There are five reports for analyzing Objects:

- Objects-Project Matrix
- Modified Objects (excluding indexes)
- Objects Checked-out
- Loaded or Imported Objects
- Objects with Extended Properties

Objects-Project Matrix

This report lists the objects that exist in projects other than **All Objects**.

Objects Included

All object types are included in this report.

Parameters

There are no parameters for this report.

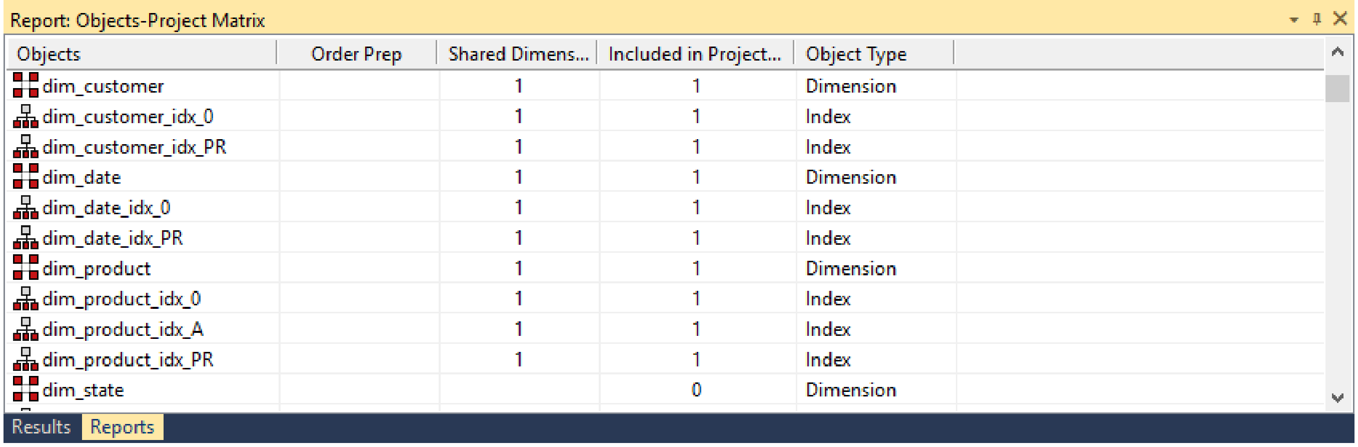
Results

The results of this report are displayed as a list of objects in the metadata repository that are in one or more projects (other than All Objects) with the following columns as a grid:

- Objects *(the name of the object)*
- Project Name 1 *(heading is the name of the first project, value is a 1 to indicate the object is in this project, blank otherwise)*
- Project Name 2 *(heading is the name of the second project, value is a 1 to indicate the object is in this project, blank otherwise)*
- ...
- Project Name n *(heading is the name of the nth project, value is a 1 to indicate the object is in this project, blank otherwise)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Report: Objects-Project Matrix

Objects	Order Prep	Shared Dimens...	Included in Project...	Object Type
dim_customer		1	1	Dimension
dim_customer_idx_0		1	1	Index
dim_customer_idx_PR		1	1	Index
dim_date		1	1	Dimension
dim_date_idx_0		1	1	Index
dim_date_idx_PR		1	1	Index
dim_product		1	1	Dimension
dim_product_idx_0		1	1	Index
dim_product_idx_A		1	1	Index
dim_product_idx_PR		1	1	Index
dim_state			0	Dimension

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Modified Objects (excluding indexes)

This report shows objects in the metadata repository with their creation and modification dates.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Exports
- Procedures
- Host Scripts

Parameters

There are no parameters for this report.

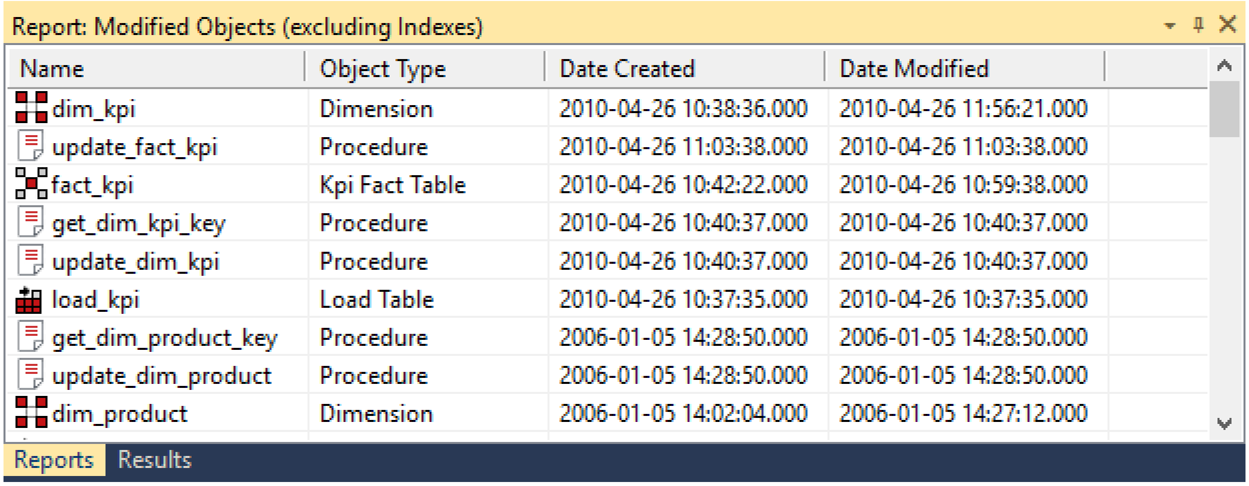
Results

The results of this report are displayed as a list of objects in the metadata repository with the following columns:

- Name *(the name of the object)*
- Object Type
- Dated Created
- Date Modified *(last modification date)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Report: Modified Objects (excluding Indexes)

Name	Object Type	Date Created	Date Modified
dim_kpi	Dimension	2010-04-26 10:38:36.000	2010-04-26 11:56:21.000
update_fact_kpi	Procedure	2010-04-26 11:03:38.000	2010-04-26 11:03:38.000
fact_kpi	Kpi Fact Table	2010-04-26 10:42:22.000	2010-04-26 10:59:38.000
get_dim_kpi_key	Procedure	2010-04-26 10:40:37.000	2010-04-26 10:40:37.000
update_dim_kpi	Procedure	2010-04-26 10:40:37.000	2010-04-26 10:40:37.000
load_kpi	Load Table	2010-04-26 10:37:35.000	2010-04-26 10:37:35.000
get_dim_product_key	Procedure	2006-01-05 14:28:50.000	2006-01-05 14:28:50.000
update_dim_product	Procedure	2006-01-05 14:28:50.000	2006-01-05 14:28:50.000
dim_product	Dimension	2006-01-05 14:02:04.000	2006-01-05 14:27:12.000

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Objects Checked-out

This report lists all objects currently checked out.

Objects Included

All object types and data warehouse tables can be included in this report.

Parameters

There are no parameters for this report.

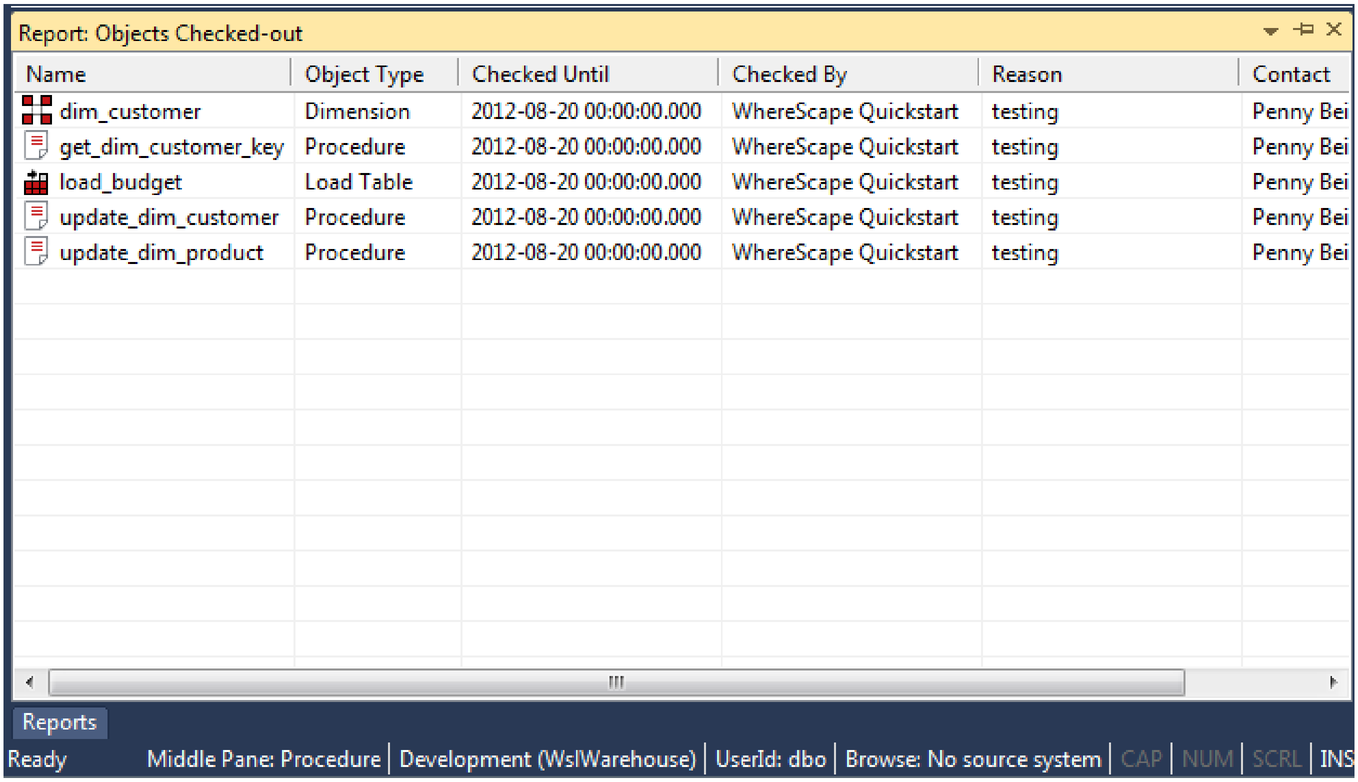
Results

The results of this report are displayed with the following columns:

- Name *(The name of the object)*
- Object Type *(The type of the object, e.g. Fact, Dimension, etc.)*
- Checked Until *(The date the object will be automatically checked back in)*
- Checked By *(The name of the WhereScape RED user who checked out the object)*
- Reason *(The reason provided for checking out the object)*
- Contact *(The contact details provided when the object was checked out)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Report: Objects Checked-out

Name	Object Type	Checked Until	Checked By	Reason	Contact
dim_customer	Dimension	2012-08-20 00:00:00.000	WhereScape Quickstart	testing	Penny Bei
get_dim_customer_key	Procedure	2012-08-20 00:00:00.000	WhereScape Quickstart	testing	Penny Bei
load_budget	Load Table	2012-08-20 00:00:00.000	WhereScape Quickstart	testing	Penny Bei
update_dim_customer	Procedure	2012-08-20 00:00:00.000	WhereScape Quickstart	testing	Penny Bei
update_dim_product	Procedure	2012-08-20 00:00:00.000	WhereScape Quickstart	testing	Penny Bei

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Loaded or Imported Objects

This report lists objects in the metadata repository that have been refreshed or imported from another repository.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Join Indexes
- Views
- Cubes
- Exports

- Procedures
- Host Scripts

Notes

- Indexes are not included.
- Objects created via Application Load must have Auto Version enabled to be included in the report.

Parameters

There are no parameters for this report.

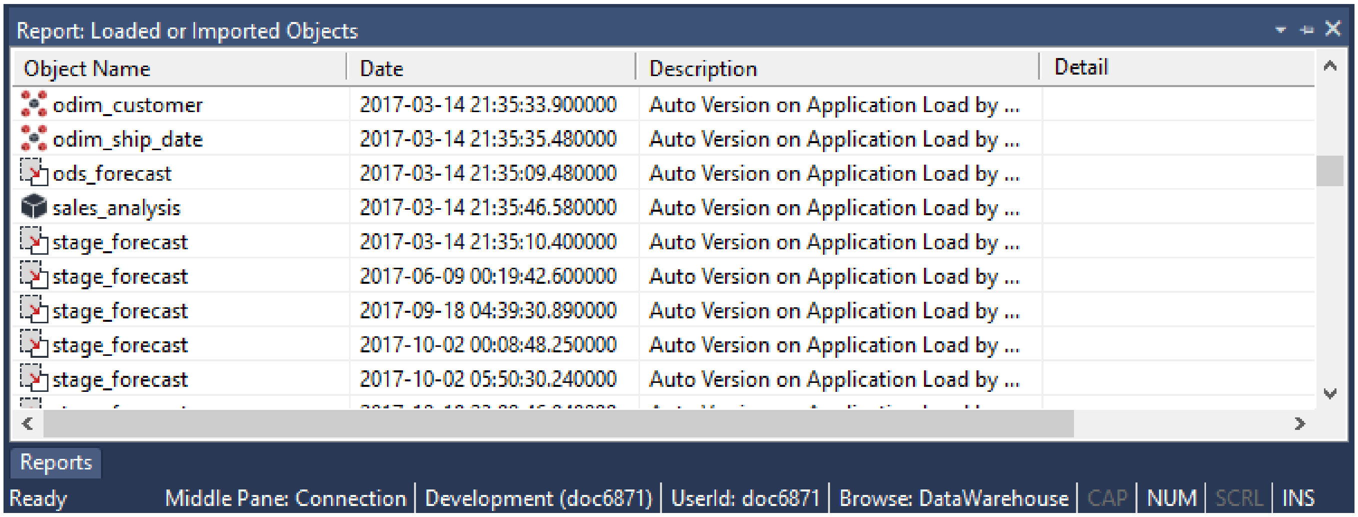
Results

The results of this report are displayed as a list of refreshed objects in the metadata repository with the following columns:

- Object Name
- Date *(of last refresh or import)*
- Description *(the kind of import or refresh)*
- Detail *(not currently used)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Report: Loaded or Imported Objects

Object Name	Date	Description	Detail
odim_customer	2017-03-14 21:35:33.900000	Auto Version on Application Load by ...	
odim_ship_date	2017-03-14 21:35:35.480000	Auto Version on Application Load by ...	
ods_forecast	2017-03-14 21:35:09.480000	Auto Version on Application Load by ...	
sales_analysis	2017-03-14 21:35:46.580000	Auto Version on Application Load by ...	
stage_forecast	2017-03-14 21:35:10.400000	Auto Version on Application Load by ...	
stage_forecast	2017-06-09 00:19:42.600000	Auto Version on Application Load by ...	
stage_forecast	2017-09-18 04:39:30.890000	Auto Version on Application Load by ...	
stage_forecast	2017-10-02 00:08:48.250000	Auto Version on Application Load by ...	
stage_forecast	2017-10-02 05:50:30.240000	Auto Version on Application Load by ...	

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Objects with Extended Properties

This report lists objects in the metadata repository with Extended Properties defined.

Objects Included

The following WhereScape RED object types are included in this report:

- Connections
- Load Tables
- Stage Tables
- Data Store Tables
- Hub Tables
- Link Tables

- Satellite Tables
- EDW 3NF Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Join Indexes
- Views
- Exports

Parameters

There are no parameters for this report.

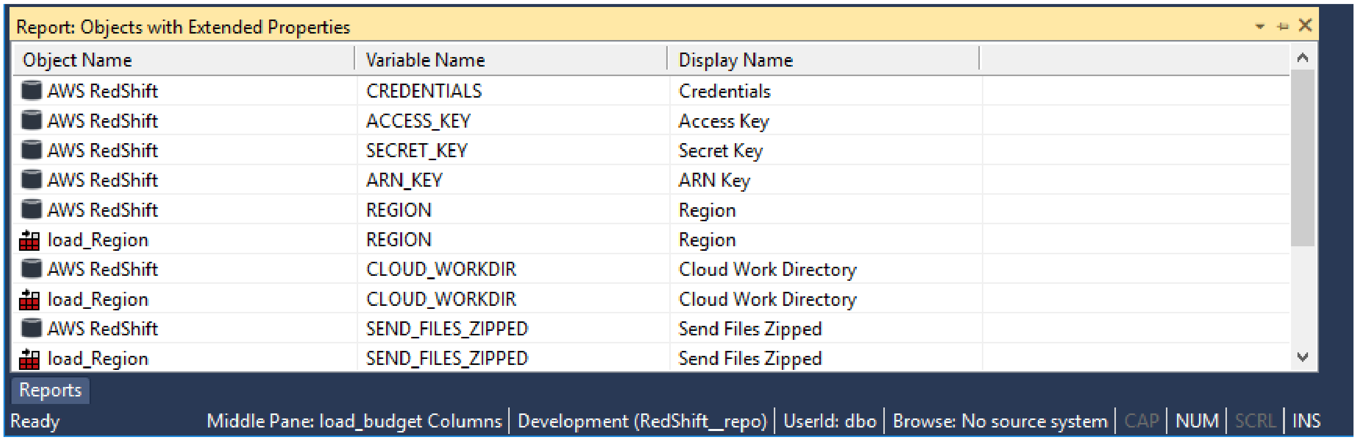
Results

The results of this report are displayed as a list of refreshed objects in the metadata repository with the following columns:

- Object Name
- Variable Name *(The unique variable name in the format it is accessed from the RED metadata service and included into scripts)*
- Display Name *(The extended property name displayed in the Connection or Table properties)*

The result set is sortable by clicking on the appropriate column heading.

Report Example



Report: Objects with Extended Properties

Object Name	Variable Name	Display Name
AWS RedShift	CREDENTIALS	Credentials
AWS RedShift	ACCESS_KEY	Access Key
AWS RedShift	SECRET_KEY	Secret Key
AWS RedShift	ARN_KEY	ARN Key
AWS RedShift	REGION	Region
load_Region	REGION	Region
AWS RedShift	CLOUD_WORKDIR	Cloud Work Directory
load_Region	CLOUD_WORKDIR	Cloud Work Directory
AWS RedShift	SEND_FILES_ZIPPED	Send Files Zipped
load_Region	SEND_FILES_ZIPPED	Send Files Zipped

Reports
Ready
Middle Pane: load_budget Columns
Development (RedShift__repo)
UserId: dbo
Browse: No source system
CAP
NUM
SCRL
INS

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Job Reports

There are three reports for analyzing Jobs:

- Object by Scheduler Job
- Jobs with an Object
- Tasks of a Job

Objects by Scheduler Job

This report shows all jobs and objects as well as the object actions. Table and Cube objects not in any jobs are displayed with a job name of **No Job**.

Objects Included

All object types are included in this report.

Parameters

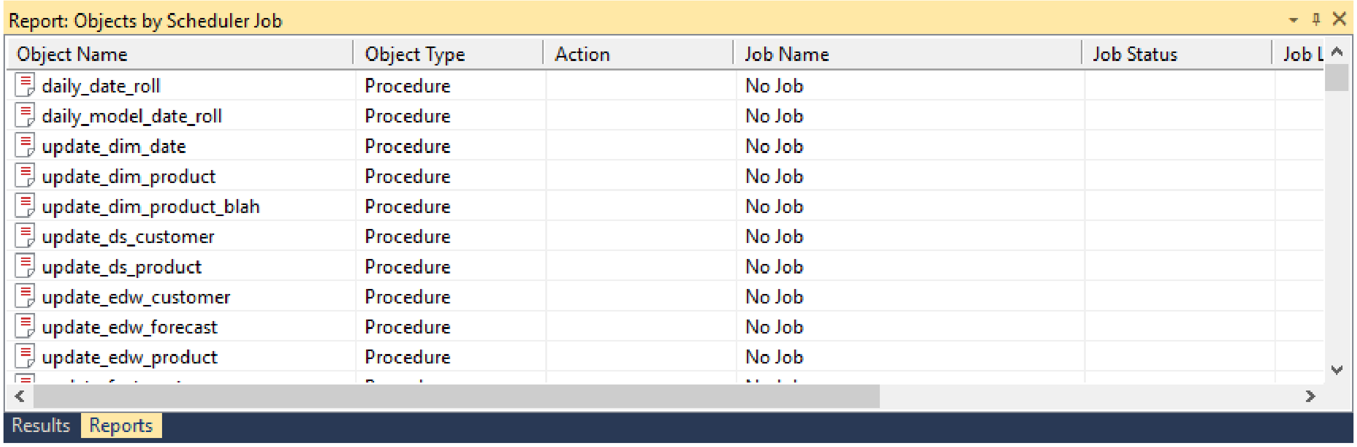
There are no parameters for this report.

Results

The results of this report are displayed as a list of jobs with the following columns:

- Table Name *(the name of the table or cube)*
- Action
- Job Name *(the name of the job)*
- Job Status
- Job Last Run
- Job Next Run

Report Example



Report: Objects by Scheduler Job

Object Name	Object Type	Action	Job Name	Job Status	Job L
daily_date_roll	Procedure		No Job		
daily_model_date_roll	Procedure		No Job		
update_dim_date	Procedure		No Job		
update_dim_product	Procedure		No Job		
update_dim_product_blah	Procedure		No Job		
update_ds_customer	Procedure		No Job		
update_ds_product	Procedure		No Job		
update_edw_customer	Procedure		No Job		
update_edw_forecast	Procedure		No Job		
update_edw_product	Procedure		No Job		

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Jobs with an Object

This report shows all jobs containing a selected object and lists the job action.

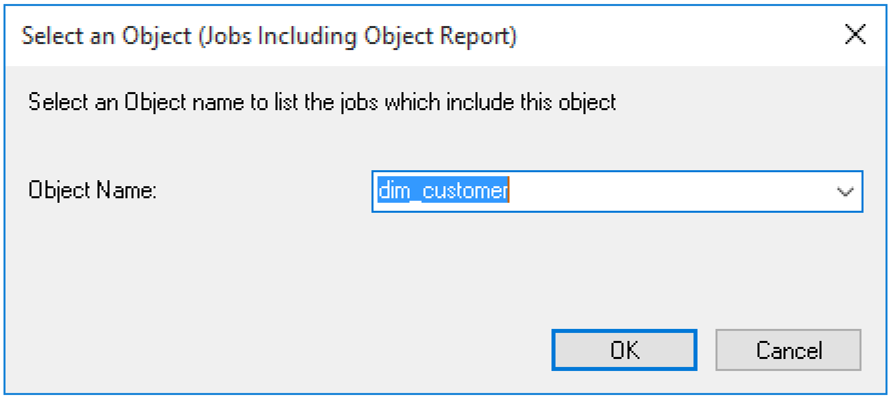
Objects Included

All object types are included in this report.

Parameters

This report has one parameter:

- Object Name



Select an Object (Jobs Including Object Report)

Select an Object name to list the jobs which include this object

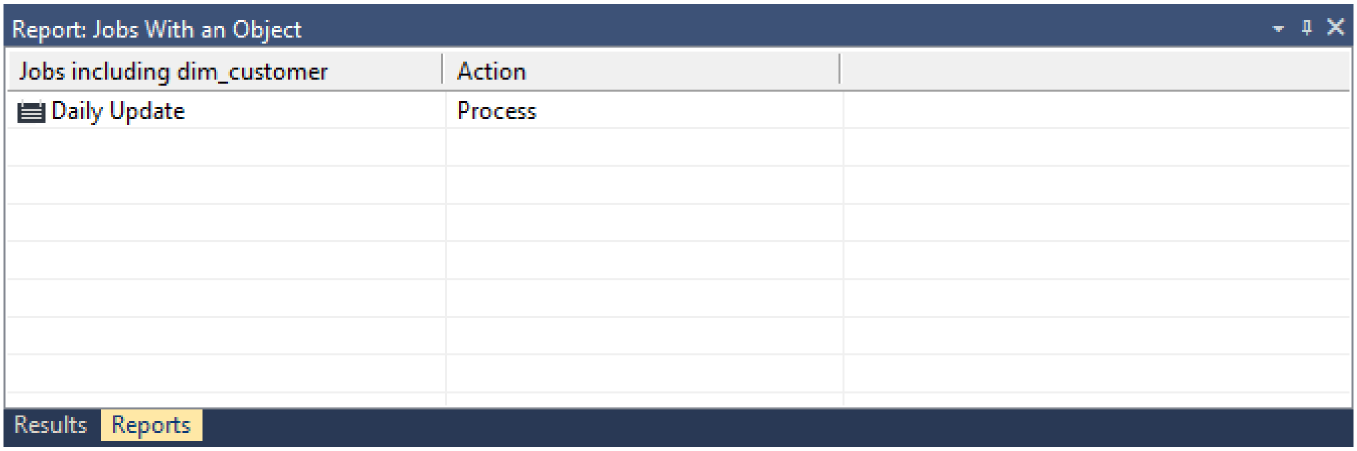
Object Name: dim_customer

Results

The results of this report are displayed as a list of jobs with the following columns:

- Jobs including **object_name**
- Action

Report Example



Jobs including dim_customer	Action
Daily Update	Process

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Tasks of a Job

This report shows all tasks for a selected job including dependencies.

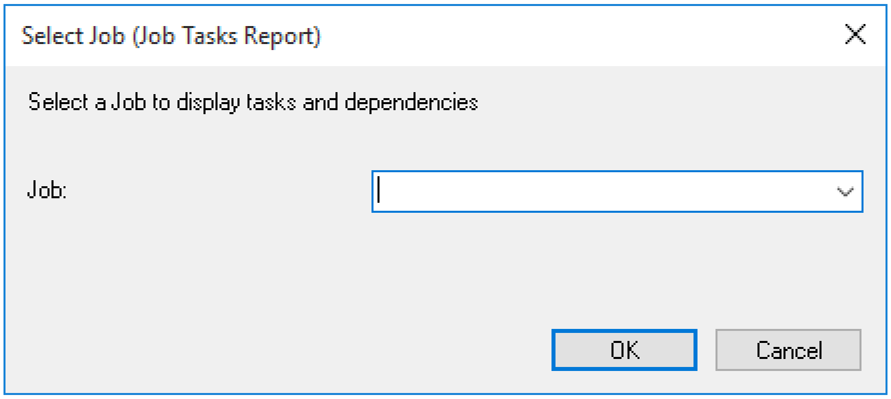
Objects Included

All object types are included in this report.

Parameters

This report has one parameter:

- Job Name



Select a Job to display tasks and dependencies

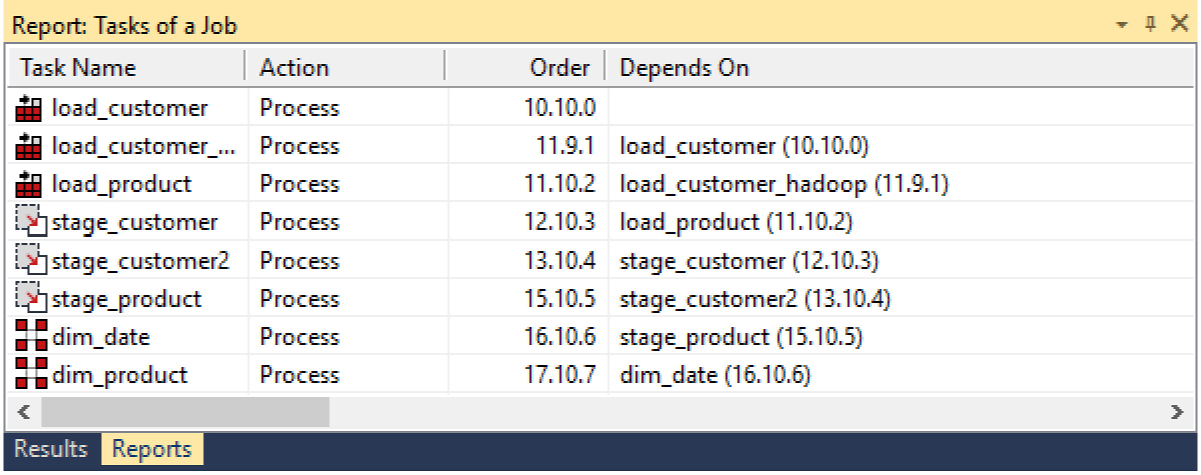
Job:

Results

The results of this report are displayed as a list of task dependencies with the following columns:

- Task name *(the table, Index, procedure or script name)*
- Action
- Order *(the order number as shown in the edit tasks dialog in the scheduler)*
- Depends On *(the task(s) and order number this task depends on)*

Report Example



Report: Tasks of a Job

Task Name	Action	Order	Depends On
load_customer	Process	10.10.0	
load_customer_...	Process	11.9.1	load_customer (10.10.0)
load_product	Process	11.10.2	load_customer_hadoop (11.9.1)
stage_customer	Process	12.10.3	load_product (11.10.2)
stage_customer2	Process	13.10.4	stage_customer (12.10.3)
stage_product	Process	15.10.5	stage_customer2 (13.10.4)
dim_date	Process	16.10.6	stage_product (15.10.5)
dim_product	Process	17.10.7	dim_date (16.10.6)

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Operational Reports

There are four Operational Reports:

- Object Performance History
- Job Performance History
- Task Performance History
- Fragmentation

Object Performance History

This report shows the audit trail from the scheduler logs for a selected object.

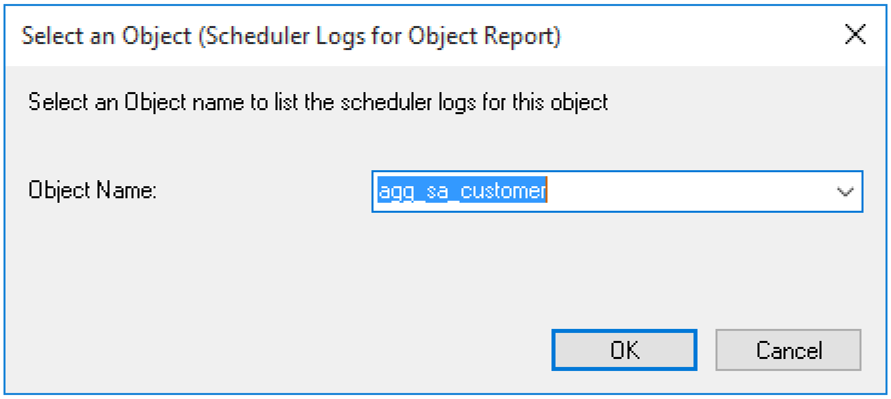
Objects Included

All object types are included in this report.

Parameters

This report has one parameter:

- Object Name



Select an Object (Scheduler Logs for Object Report)

Select an Object name to list the scheduler logs for this object

Object Name: agg_sa_customer

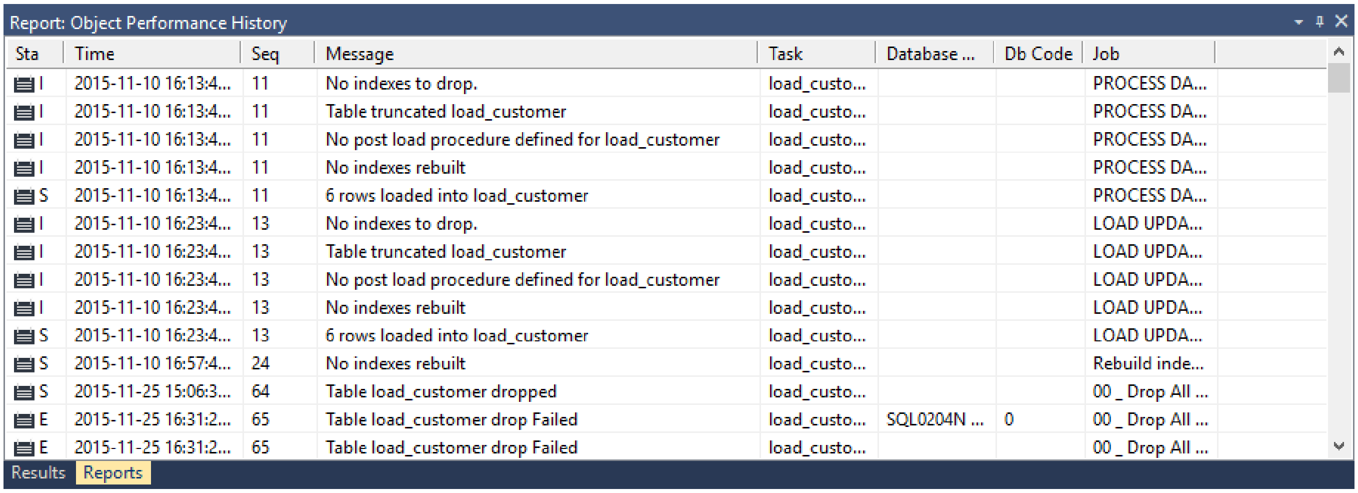
Results

The results of this report are displayed as a list of audit log entries with the following columns:

- Sta *(the type of audit log entry)*
- Time *(the date and time the audit log entry was written)*

- Seq *(the job sequence of the job writing the audit log entry)*
- Message *(the message in the Audit log)*
- Task *(the object name)*
- Job *(the name of the job that ran the task)*

Report Example



Report: Object Performance History

Sta	Time	Seq	Message	Task	Database ...	Db Code	Job
▤ I	2015-11-10 16:13:4...	11	No indexes to drop.	load_custo...			PROCESS DA...
▤ I	2015-11-10 16:13:4...	11	Table truncated load_customer	load_custo...			PROCESS DA...
▤ I	2015-11-10 16:13:4...	11	No post load procedure defined for load_customer	load_custo...			PROCESS DA...
▤ I	2015-11-10 16:13:4...	11	No indexes rebuilt	load_custo...			PROCESS DA...
▤ S	2015-11-10 16:13:4...	11	6 rows loaded into load_customer	load_custo...			PROCESS DA...
▤ I	2015-11-10 16:23:4...	13	No indexes to drop.	load_custo...			LOAD UPDA...
▤ I	2015-11-10 16:23:4...	13	Table truncated load_customer	load_custo...			LOAD UPDA...
▤ I	2015-11-10 16:23:4...	13	No post load procedure defined for load_customer	load_custo...			LOAD UPDA...
▤ I	2015-11-10 16:23:4...	13	No indexes rebuilt	load_custo...			LOAD UPDA...
▤ S	2015-11-10 16:23:4...	13	6 rows loaded into load_customer	load_custo...			LOAD UPDA...
▤ S	2015-11-10 16:57:4...	24	No indexes rebuilt	load_custo...			Rebuild inde...
▤ S	2015-11-25 15:06:3...	64	Table load_customer dropped	load_custo...			00 _ Drop All ...
▤ E	2015-11-25 16:31:2...	65	Table load_customer drop Failed	load_custo...	SQL0204N ...	0	00 _ Drop All ...
▤ E	2015-11-25 16:31:2...	65	Table load_customer drop Failed	load_custo...			00 _ Drop All ...

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Job Performance History

This report shows the performance (duration) over time of a selected job.

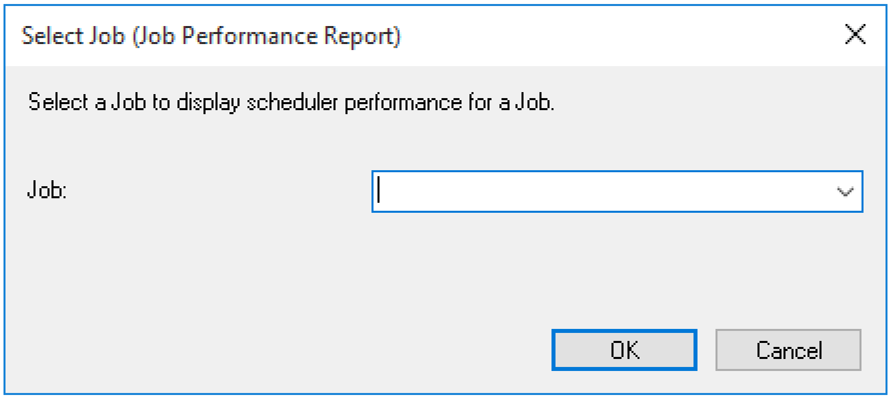
Objects Included

All object types are included in this report.

Parameters

This report has one parameter:

- Job Name



Select Job (Job Performance Report)

Select a Job to display scheduler performance for a Job.

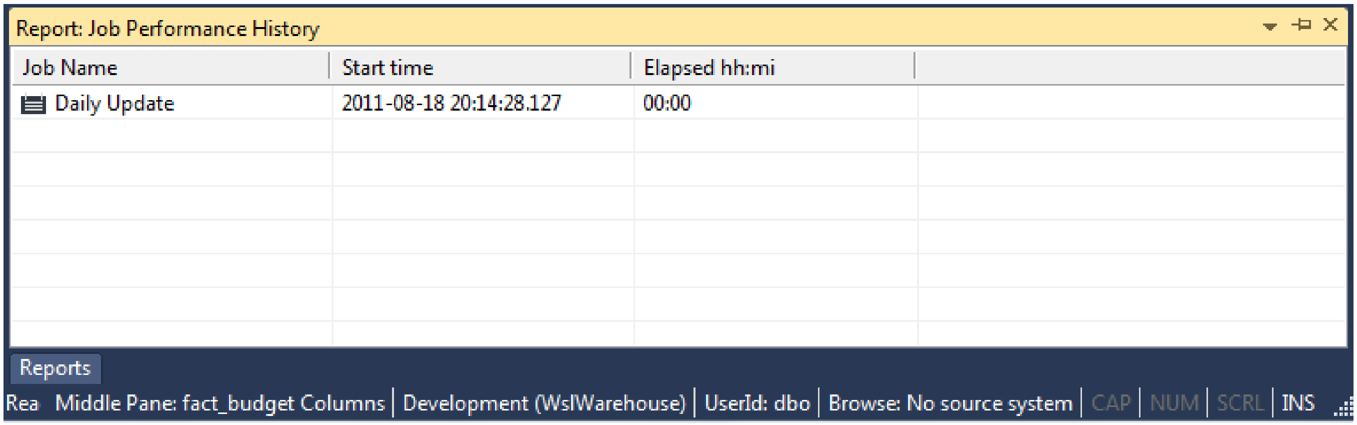
Job:

Results

The results of this report are displayed as a list of job instances with the following columns:

- Job name *(the name of the job)*
- Start time *(the date and time the job started)*
- Elapsed hh:mm *(the duration of the job)*

Report Example



Report: Job Performance History

Job Name	Start time	Elapsed hh:mi
Daily Update	2011-08-18 20:14:28.127	00:00

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Task Performance History

This report shows the performance (duration) over time of a selected task within a selected job.

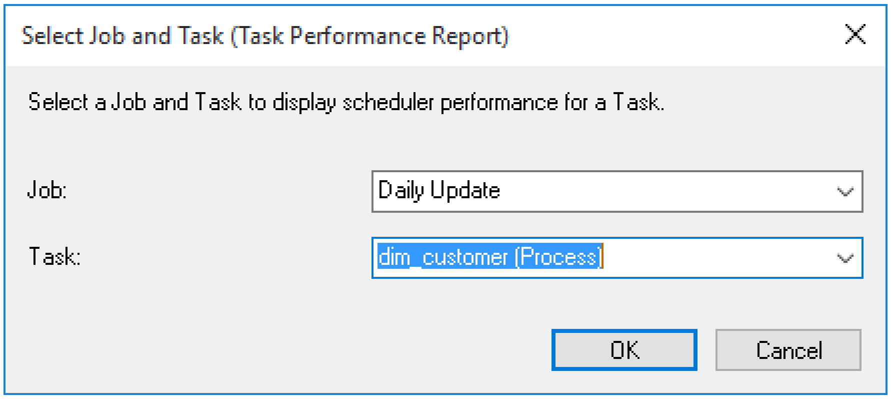
Objects Included

All object types are included in this report.

Parameters

This report has two parameters:

- Job Name
- Task Name (including action)



Select Job and Task (Task Performance Report)

Select a Job and Task to display scheduler performance for a Task.

Job: Daily Update

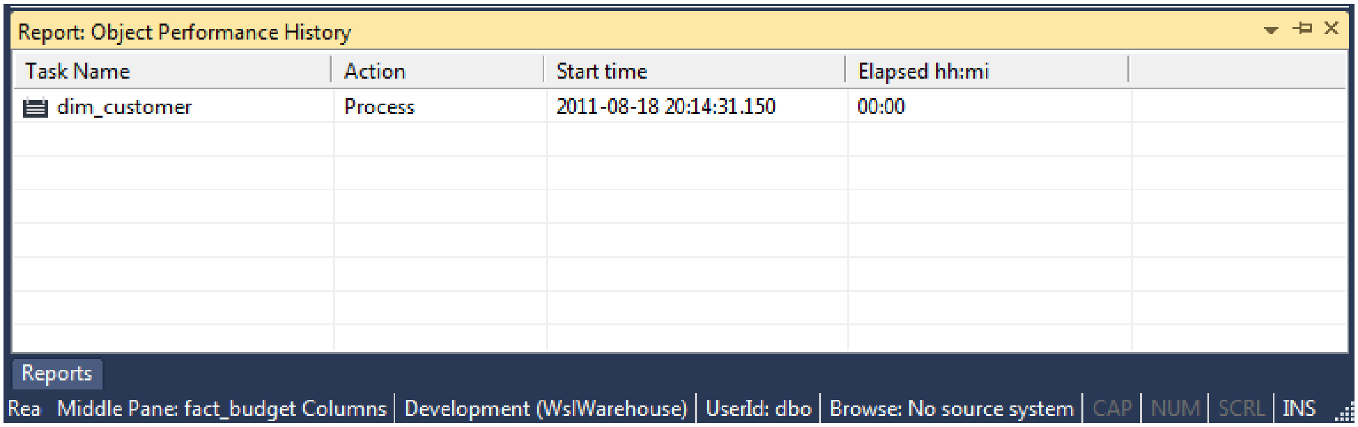
Task: dim_customer (Process)

Results

The results of this report are displayed as a list of task instances for the selected job with the following columns:

- Task name *(the table, Index, procedure or script name)*
- Action
- Start time *(the date and time the task started)*
- Elapsed hh:mm *(the duration of the task)*

Report Example



Report: Object Performance History

Task Name	Action	Start time	Elapsed hh:mi
dim_customer	Process	2011-08-18 20:14:31.150	00:00

Reports
Rea | Middle Pane: fact_budget Columns | Development (WslWarehouse) | UserId: dbo | Browse: No source system | CAP | NUM | SCRL | INS

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Fragmentation

This report is only available on a SQL Server database.

Objects Included

All object types are included in this report.

Parameters

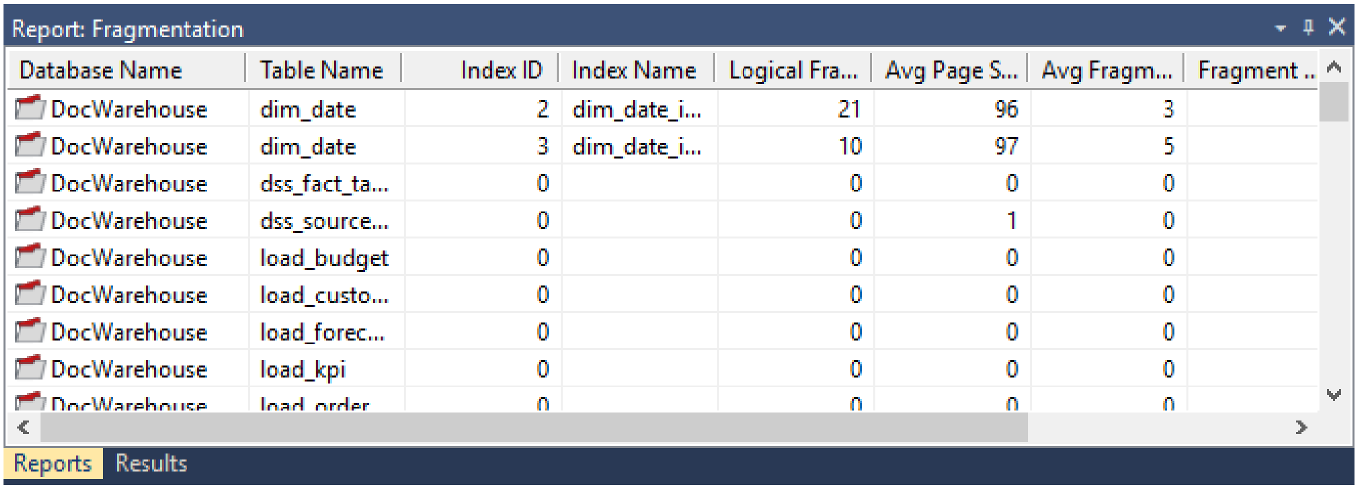
This report has no parameters.

Results

The results of this report are displayed as a list of tables within the database with the following columns:

- Database
- Table Name
- Index ID
- Index Name
- Logical Fragmentation
- Avg Page Space Used
- Avg Fragment Size in Pages
- Fragment Count
- Page Count
- Size in GB

Report Example



Report: Fragmentation

Database Name	Table Name	Index ID	Index Name	Logical Fra...	Avg Page S...	Avg Fragm...	Fragment ..
DocWarehouse	dim_date	2	dim_date_i...	21	96	3	
DocWarehouse	dim_date	3	dim_date_i...	10	97	5	
DocWarehouse	dss_fact_ta...	0		0	0	0	
DocWarehouse	dss_source...	0		0	1	0	
DocWarehouse	load_budget	0		0	0	0	
DocWarehouse	load_custo...	0		0	0	0	
DocWarehouse	load_forec...	0		0	0	0	
DocWarehouse	load_kpi	0		0	0	0	
DocWarehouse	load_order	0		0	0	0	

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Validate

When these Validate processes are run, the results are displayed in the middle pane of the RED screen; the results of the other reports are displayed in a separate tab in the bottom pane of the RED screen.

Validate Meta-data

This process validates the Meta data. If problems are encountered, the results are displayed in the middle pane.

Use the right-click option against each identified issue to apply a repair.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Validate Workflow Data

This process validates the Workflow data. If problems are encountered, the results are displayed in the middle pane.

Use the right-click option against each identified issue to apply a repair.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Validate Table Create Status

This process validates a table structure in the meta data against the table in the database.

Select one or more tables and click the **Validate Selected** button or click the **Validate All** button to validate all the tables.

1. If a table is found to be different then it can be altered by using the right-click menu option when positioned over the table name.
2. If the update date and the modified in database date imply a change that is not apparent then these dates can be re-synced in the same way.

Sync Column order with database

Right click on the result set and select **Sync Column order with database** to reorder the metadata columns to match the column order in the database table.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Validate Load Table Status

This process compares a load table in the meta data with the table in the source system. It compares the columns and column data types.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Validate Index Status

This process validates an index structure in the meta data against the index in the database. Select one or more indexes and click the **Validate Selected** button or click the **Validate All** button to validate all indexes.

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Validate Procedure Status

This process compares a procedure in the meta data with the compiled version of the same procedure stored within the database. The subsequent display will report either a match or a difference.

If a procedure is found to differ then you can use the procedure editor to examine the exact differences by selecting the **Tools > Compare to Compiled Source** option.

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

List Metadata Tables not in the Database

This report shows database **table** objects in the metadata that don't exist in the data warehouse database.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Retro Copies (but not Retro Definitions)

Parameters

There are no parameters for this report.

Results

The results of this report are displayed as a list of objects in the metadata not in the data warehouse database.

Report Example

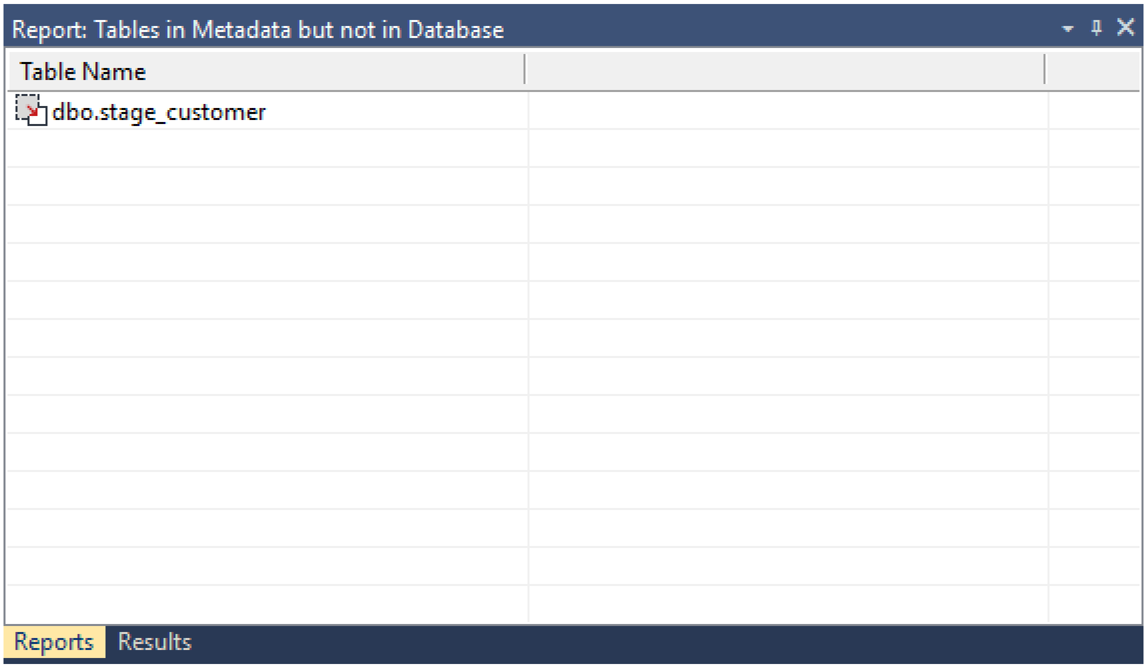


Table Name		
dbo.stage_customer		

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

List Database Tables not in the Meta-data

This report shows database **table** objects that exist in the data warehouse database but not in the metadata.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Retro Copies (but not Retro Definitions)

Parameters

There are no parameters for this report.

Results

The results of this report are displayed as a list of objects in the data warehouse database not in the metadata.

Report Example

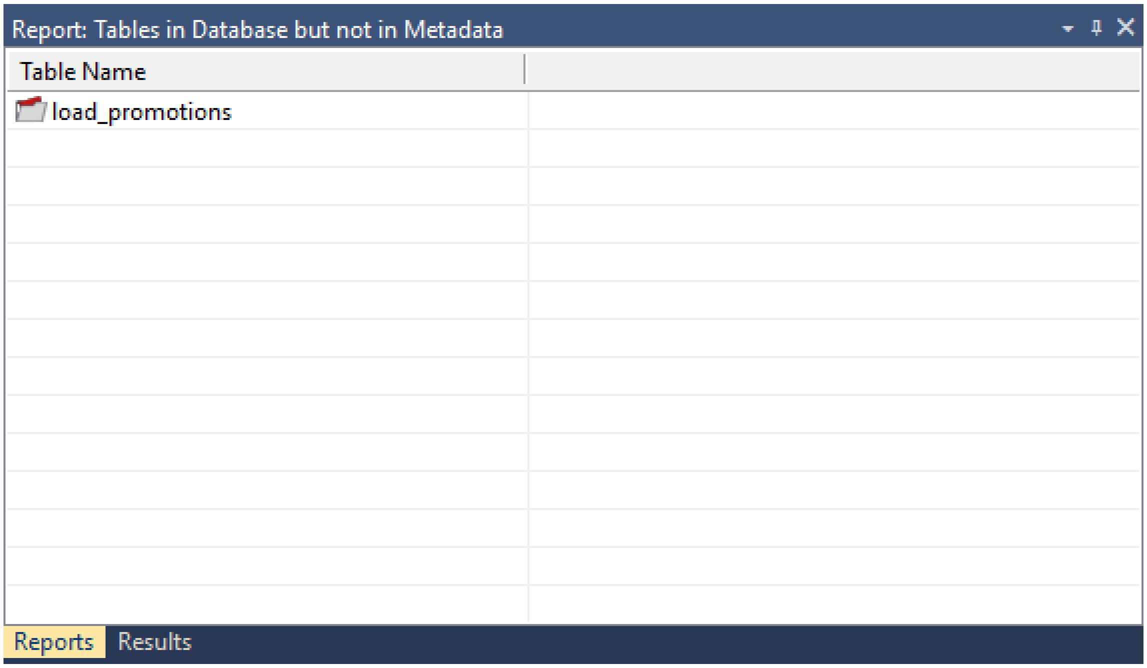


Table Name
load_promotions

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

List Tables with no related Procedures or Scripts

This report shows all table objects (certain types of objects only - see below) in the metadata repository that don't have an associated update procedure.

Objects Included

The following WhereScape RED object types are included in this report:

- Load Tables **(script based loads only)**
- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables
- Data Store Tables
- EDW 3NF Tables

Parameters

There are no parameters for this report.

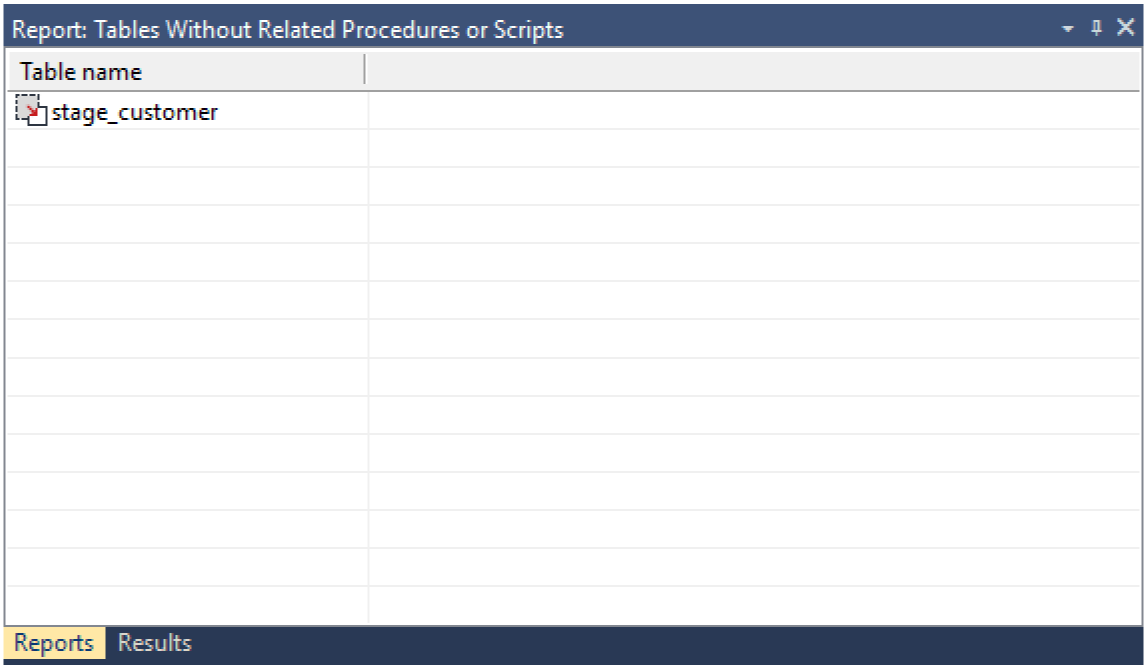
Results

The results of this report are displayed as a list of table objects in the metadata repository with the following columns:

- Table name

The result set is sortable by clicking on the appropriate column heading.

Report Example



Report: Tables Without Related Procedures or Scripts

Table name
stage_customer

Sending Results to Microsoft Excel

Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

List Procedures not related to a Table

This report shows all procedures and host scripts in the metadata repository that aren't associated with a table object.

Objects Included

The following WhereScape RED object types are included in this report:

- Procedures
- Host Scripts

Parameters

There are no parameters for this report.

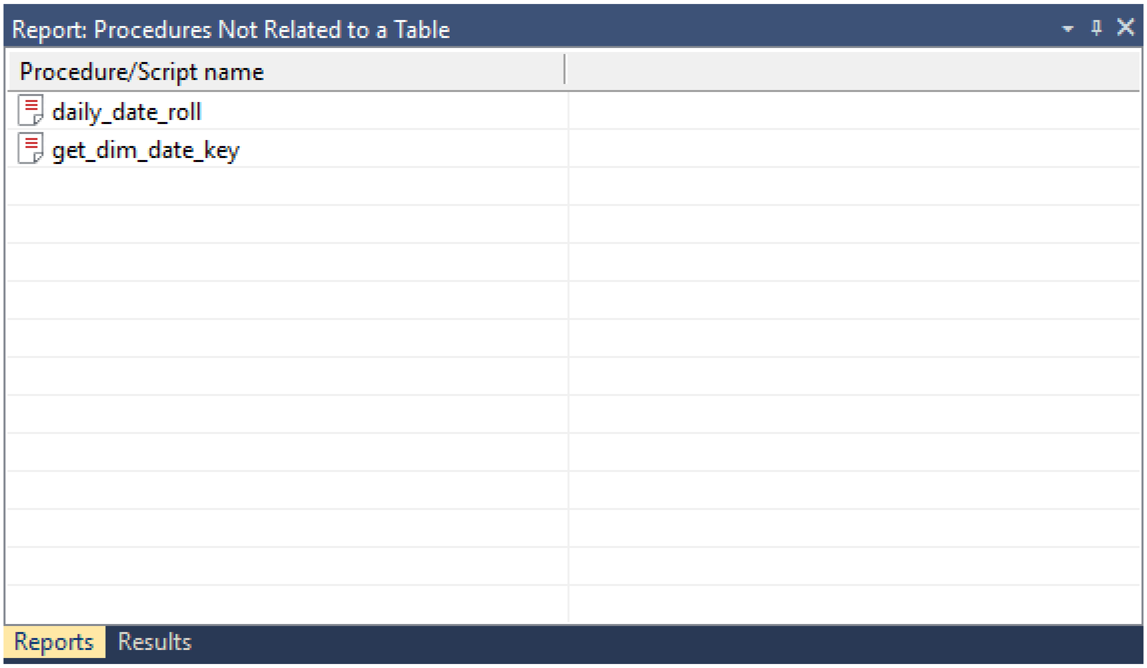
Results

The results of this report are displayed as a list of code objects in the metadata repository with the following columns:

- Procedure/Script name

The result set is sortable by clicking on the appropriate column heading.

Report Example



Procedure/Script name	
daily_date_roll	
get_dim_date_key	

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Compare Meta-data Repository to Another

This report compares the current metadata repository to a remote repository.

Objects Included

All object types are included in this report.

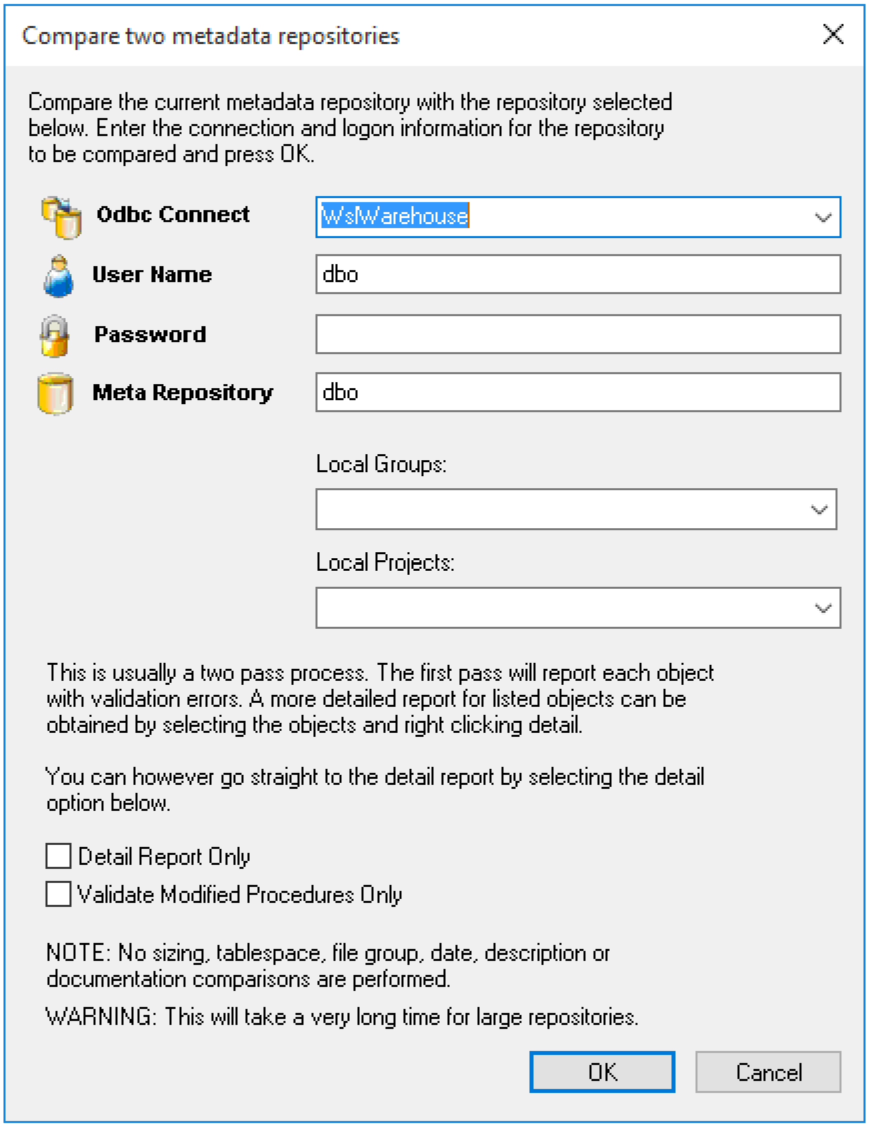
Parameters

This report requires connection information for the remote repository to be entered. Specifically:

- ODBC Connect **(the ODBC source for the other repository)**
- User Name **(user name for the remote repository)**
- Password **(password for the remote repository)**
- Meta Repository **(database/user the remote repository metadata is stored in)**

Four additional parameters can be specified:

- Optional filter on Local Groups
- Optional filter on Local Projects
- Do detail report
- Only validate procedures that have been modified



Compare two metadata repositories

Compare the current metadata repository with the repository selected below. Enter the connection and logon information for the repository to be compared and press OK.

Odbc Connect: WslWarehouse

User Name: dbo

Password:

Meta Repository: dbo

Local Groups:

Local Projects:

This is usually a two pass process. The first pass will report each object with validation errors. A more detailed report for listed objects can be obtained by selecting the objects and right clicking detail.

You can however go straight to the detail report by selecting the detail option below.

☐ Detail Report Only

☐ Validate Modified Procedures Only

NOTE: No sizing, tablespace, file group, date, description or documentation comparisons are performed.

WARNING: This will take a very long time for large repositories.

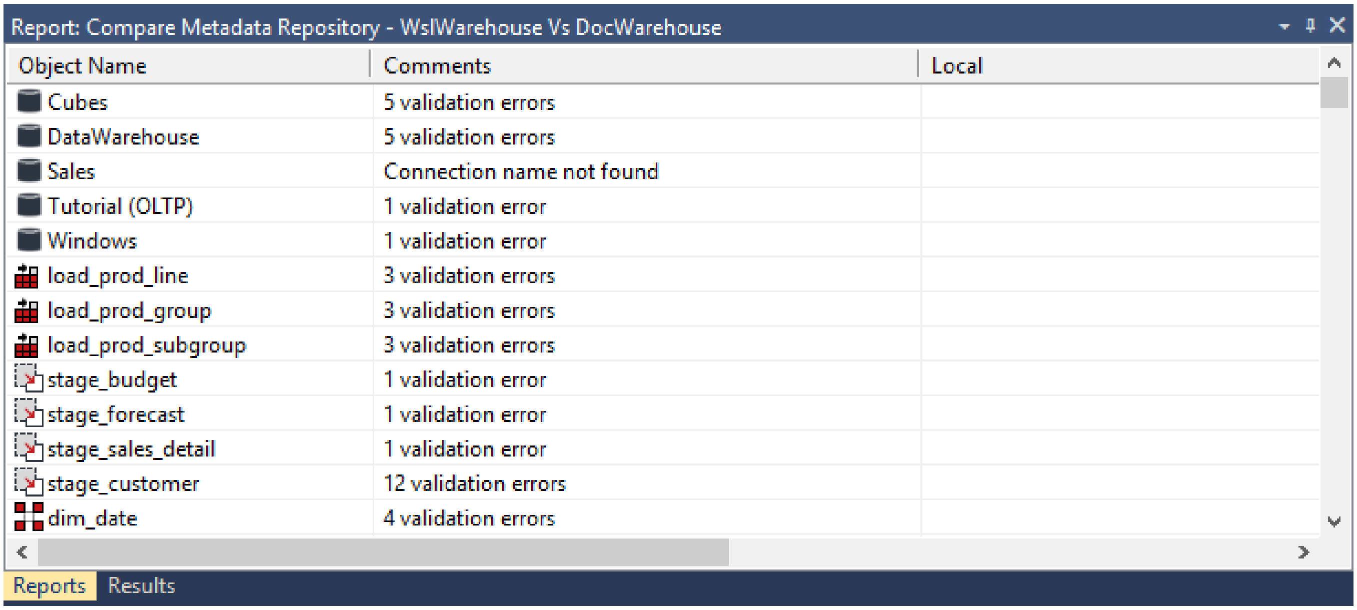
Results

The results of this report are displayed as a list of differences with the following columns:

- Object Name
- Comments **(Summary difference between current and selected repository)**
- Local
- Remote

The result set is sortable by clicking the appropriate column heading.

Report Example



Report: Compare Metadata Repository - WslWarehouse Vs DocWarehouse

Object Name	Comments	Local
Cubes	5 validation errors	
DataWarehouse	5 validation errors	
Sales	Connection name not found	
Tutorial (OLTP)	1 validation error	
Windows	1 validation error	
load_prod_line	3 validation errors	
load_prod_group	3 validation errors	
load_prod_subgroup	3 validation errors	
stage_budget	1 validation error	
stage_forecast	1 validation error	
stage_sales_detail	1 validation error	
stage_customer	12 validation errors	
dim_date	4 validation errors	

Checking result details:

- Right-click the object name with validation errors.
- Select **Detail**.
- This will rerun the report just for the selected object and will display more details about the errors in the Comments, Local and/or Remote Column(s).

Sending Results to Microsoft Excel

- Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Compare Meta-data Indexes to Database

This report shows database indexes in the data warehouse database that don't exist in the metadata.

Objects Included

The following WhereScape RED object types are included in this report:

- Indexes

Parameters

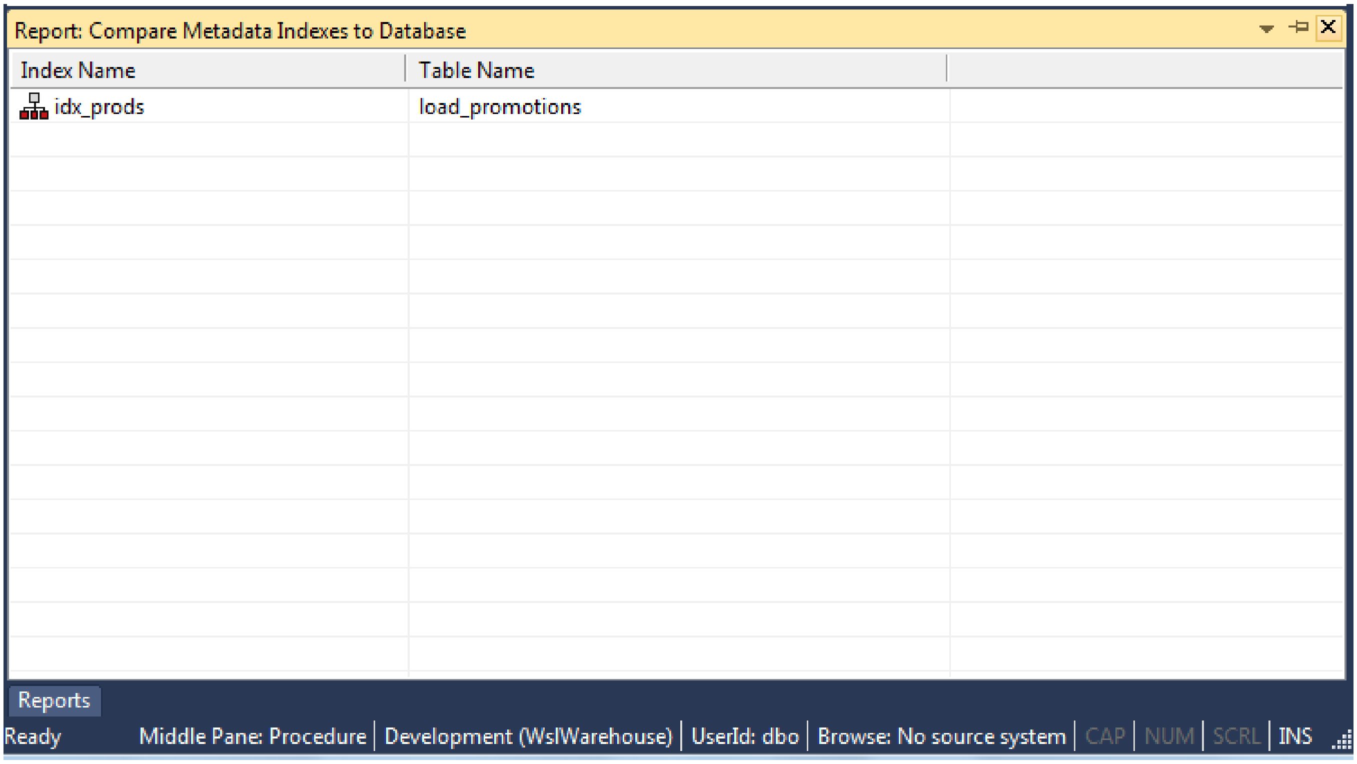
There are no parameters for this report.

Results

The results of this report are displayed as a list of indexes in the data warehouse database not in the metadata with the following columns:

- Index Name **(the name of the index)**
- Table Name

Report Example



Index Name	Table Name
idx_prods	load_promotions

Sending Results to Microsoft Excel

- Right-click on the result set and click **Output to File** to send the results of this report to Microsoft Excel.

List Duplicate Business Key Columns

This report shows any columns that are the business (natural) key of more than one table.

Objects Included

The following WhereScape RED object types are included in this report:

- Stage Tables
- Dimension Tables and Views
- Fact Tables
- Aggregate Tables

Parameters

There are no parameters for this report.

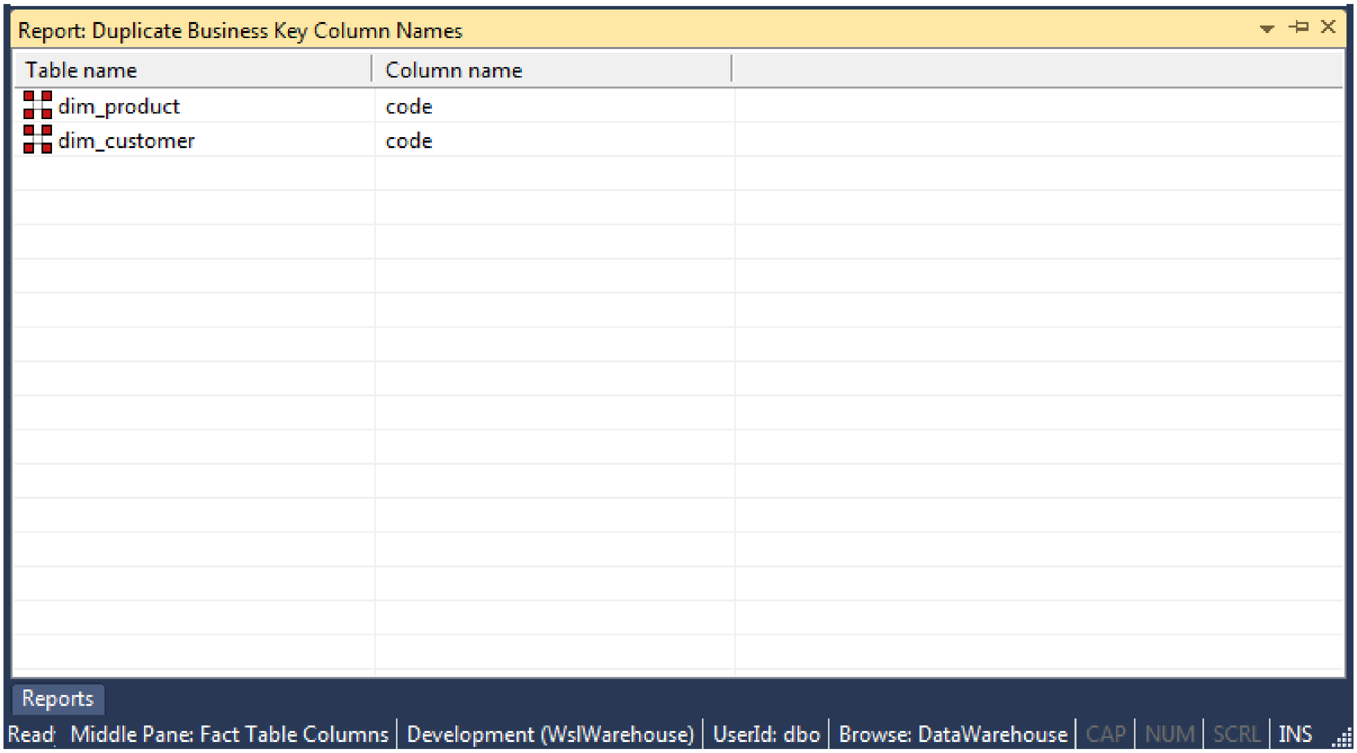
Results

The results of this report are displayed as a list of table columns with the following columns:

- Table name
- Column name

The result set is sortable by clicking the appropriate column heading.

Report Example



Report: Duplicate Business Key Column Names

Table name	Column name
dim_product	code
dim_customer	code

Sending Results to Microsoft Excel

- Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Query Data Warehouse Objects

This report allows SQL queries to be run as the user signed into the repository.

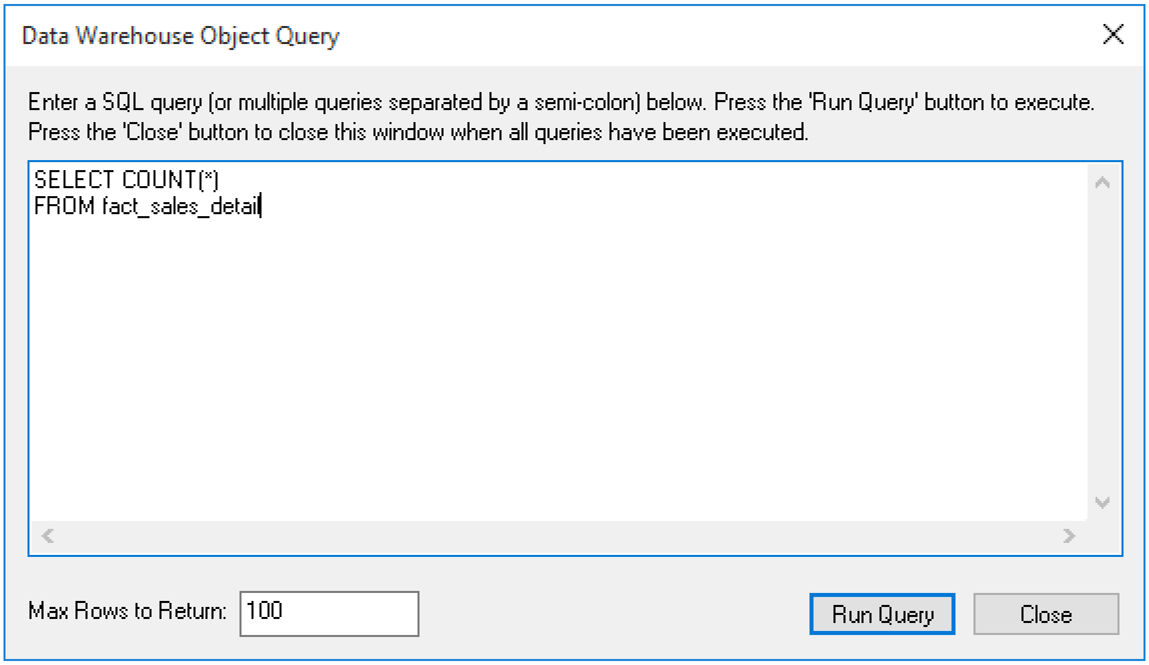
Objects Included

All object types and data warehouse tables can be included in this report.

Parameters

This report has one parameter:

- The SQL Query



Enter a SQL query (or multiple queries separated by a semi-colon) below. Press the 'Run Query' button to execute. Press the 'Close' button to close this window when all queries have been executed.

```
SELECT COUNT(*)
FROM fact_sales_detail
```

Max Rows to Return: 100

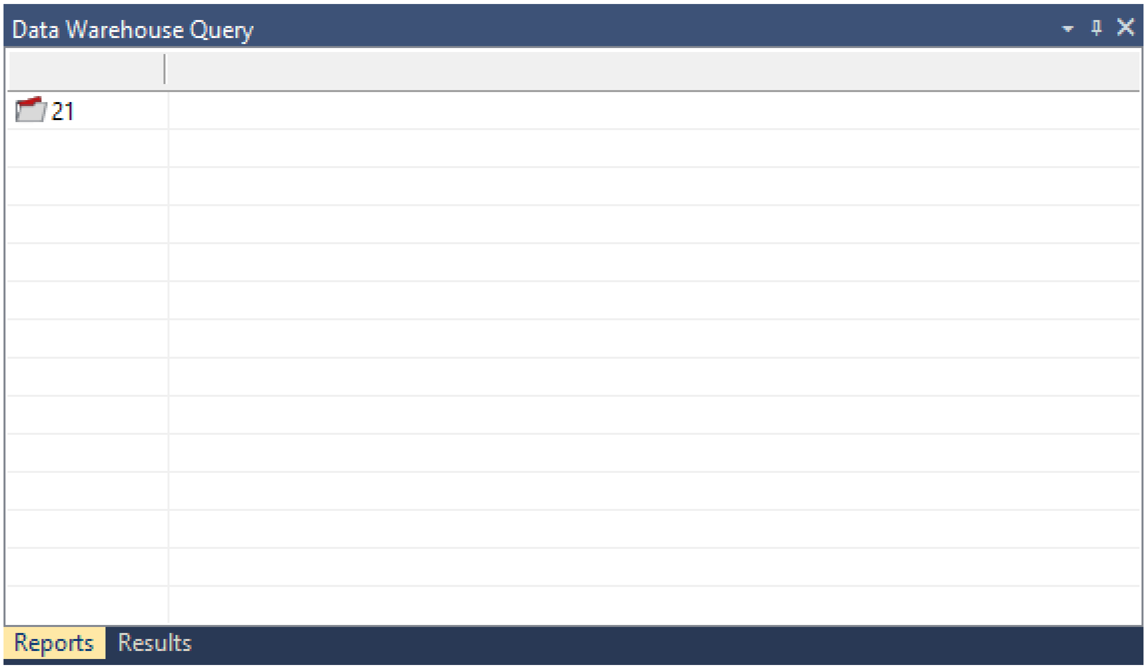
Results

The results of this report are displayed with the following columns:

- First SQL column SELECTed
- Second SQL column SELECTed
- ...
- nth SQL column SELECTed

The result set is sortable by clicking on the appropriate column heading.

Report Example



```
21
```

Sending Results to Microsoft Excel

Right-click the result set and click **Output to File** to send the results of this report to Microsoft Excel.

Promoting Between Environments

This chapter covers the promotion of metadata objects between environments. Various methods exist for getting new or changed metadata from one repository into another repository.

It is of course possible and in fact desirable to have multiple metadata repositories. At the very least we would normally have a development and a production environment.

In some situations, it may also be desirable to have multiple child development repositories with one master repository where all elements are brought together. WhereScape RED supports this type of structure but does not include source control or co-ordination of the repositories. It is up to the data warehouse manager to manually ensure that the various objects are kept in sync and coordinated.

As with any software system there are issues around how code is moved from a development environment into a testing or production environment.

This promotion of objects can be achieved via a number of different methods. Each is discussed below.

In summary, they are:

1. Updating a repository with an application or application patch.
2. Importing objects from another repository.
3. Restoring a full metadata set into a repository. Refer to **Backing Up and Restoring Metadata** for details.

Applications

The definition of an application is discussed in this section and the process of loading and updating applications is discussed in more detail in the RED Installation Guide. Only the concepts of the use of applications are covered here.

An application is defined for our purposes as a group of objects. An application is a method of loading objects into a metadata repository. It can be used to upgrade or provide patches to an existing metadata repository. As such, an application can be used to distribute and remotely maintain a specific data warehousing solution.

An application consists of a series of Windows files, which can be distributed to remote sites.

A list of the applications that have been applied to the metadata repository can be displayed via the **Tools > List Loaded Deployment Applications** menu option.

An application is created through the **Tools > Build Deployment Application** menu option. The created application can then update a metadata repository through the WhereScape RED Setup Administrator utility. In this manner, the application model can be used to update a metadata repository in an ordered and controlled fashion. Loading an application inserts various objects into the chosen metadata repository. An application is best defined as a set of objects that are shipped to enable inclusion of those objects in a remote repository.

Notes:

An application can only be loaded into a metadata repository running on the same database type as that of the application creator .

WhereScape RED also enable you to create deployment applications via the **Projects/Groups and Object Groups** context menu, as well as from a **Diagram** generated in RED.

Application Creation

Tip:

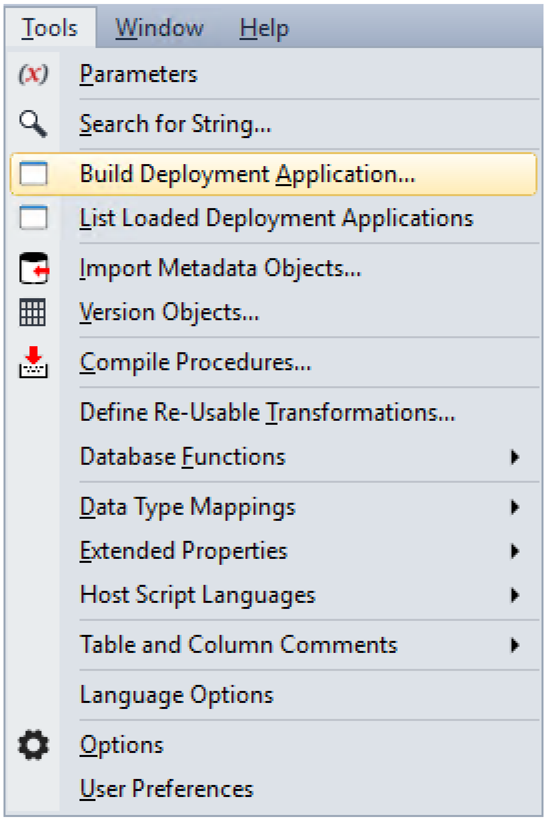
You can automatically create an Application from:

- Projects or Project Groups—refer to **Building an Application from Projects or Groups** for details.
- Object Groups—refer to **Building an Application from Object Groups** for details.

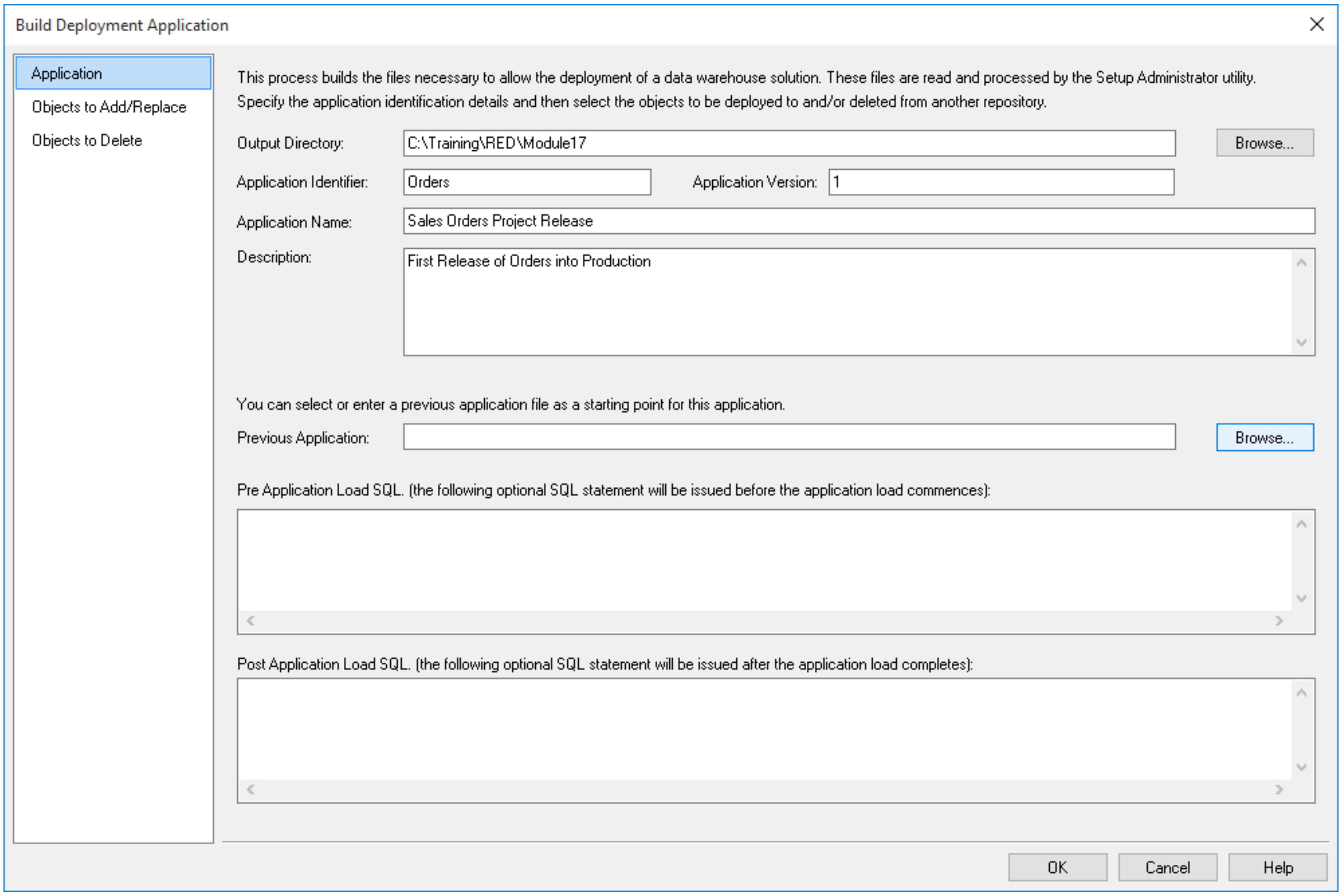
Tip:

- Diagrams—refer to Creating an Application from a Diagram for details.

An application is created by selecting the **Tools > Build Deployment Application** menu option.



The following screen is displayed:



Build Deployment Application

This process builds the files necessary to allow the deployment of a data warehouse solution. These files are read and processed by the Setup Administrator utility. Specify the application identification details and then select the objects to be deployed to and/or deleted from another repository.

Output Directory: C:\Training\RED\Module17

Application Identifier: Orders

Application Version: 1

Application Name: Sales Orders Project Release

Description: First Release of Orders into Production

You can select or enter a previous application file as a starting point for this application.

Previous Application:

Pre Application Load SQL. (the following optional SQL statement will be issued before the application load commences):

Post Application Load SQL. (the following optional SQL statement will be issued after the application load completes):

Once the application is defined and the objects selected, the application files are generated when you click the **OK** button.

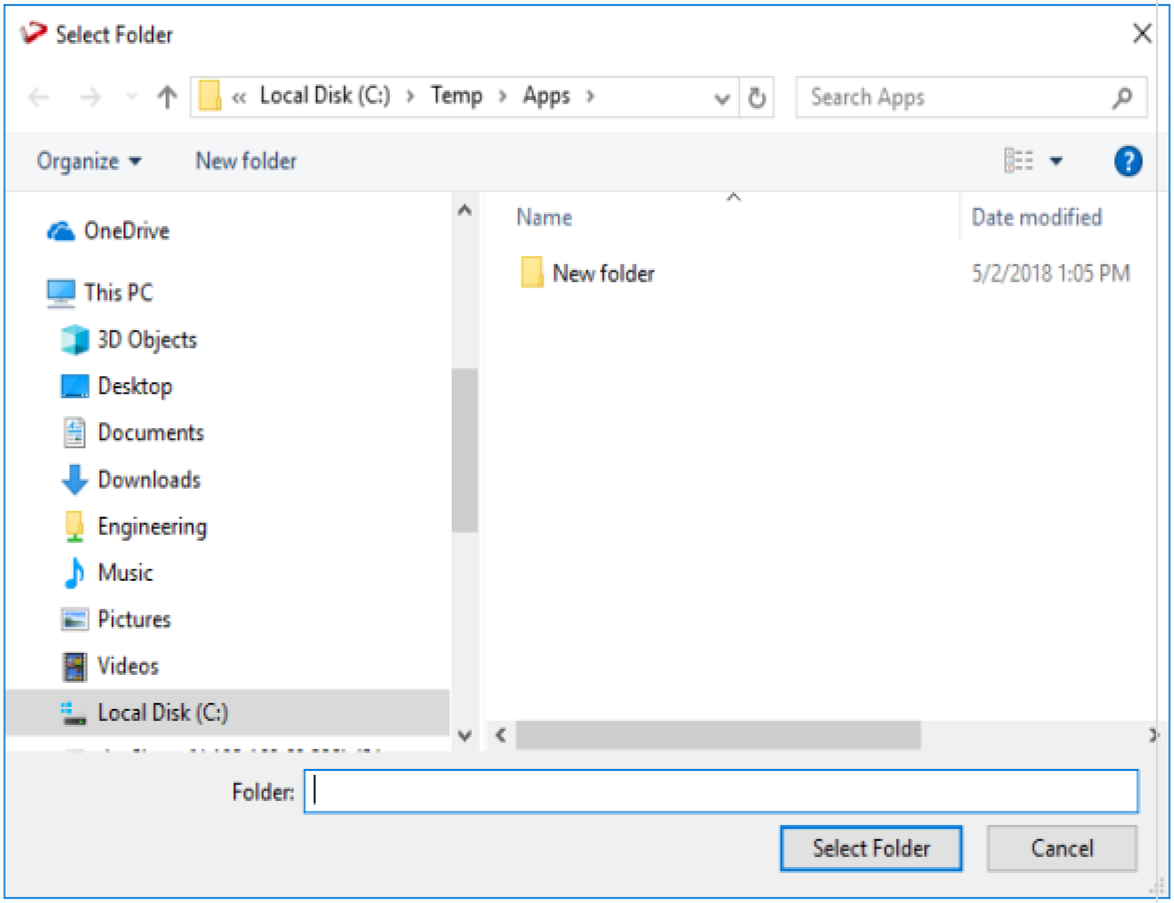
If procedures are compiled as part of the subsequent application load, the compiles occur in the order in which they are listed in the application. If procedure dependencies exist, ensure that their ordering in the application object list is correct.
There are three tabs in the **Build Deployment Application** screen.

- The first tab defines the application.
- The second tab lists the objects to add or replace in the destination repository.
- The third tab lists the objects to delete in the destination repository.

Define an Application distribution

Application

Fields	Description
Output Directory	The directory to which the application files will be written. By default, this is the WhereScape program directory. **Note** To browse for the required folder, click the **Browse...** button. Click **New Folder** to create a new folder in the selected directory.

<table>
<tr><th>Fields</th><th>Description</th></tr>
<tr><td></td><td>
</td></tr>
<tr><td><b>Application Identifier</b></td><td>The application identifier is a four character code used to uniquely identify the application. This identifier is used in the naming of the files that are created to contain the application data.</td></tr>
<tr><td><b>Application Version</b></td><td>The version is a character string that provides a version number for reference purposes. This version number is displayed when applications are being loaded and is used in the naming of the files that are created to contain the application data. As such, it must contain characters that are valid in a Windows file name</td></tr>
<tr><td><b>Application Name</b></td><td>The name by which the application is known. This name is displayed during the selection of an application to load and is recorded in the metadata of the repository into which an application is loaded. It is not used, apart from documentation purposes.</td></tr>
<tr><td><b>Description</b></td><td>This description is displayed during the selection of an application to load. It is not used at any other point, apart from documentation purposes.</td></tr>
<tr><td><b>Application Files</b></td><td>When an application is created the following files are built, where XXXX is the application identifier and NNNNN is the application version.
<table>
<tr><th>File</th><th>Purpose</th></tr>
<tr><td><b>App_data_XXXX_NNNNN.wst</b></td><td>This file contains the scripts and data required to rebuild the objects in the new metadata repository.</td></tr>
<tr><td><b>App_id_XXXX_NNNNN.wst</b></td><td>This control file identifies the application and its version.</td></tr>
<tr><td><b>App_obj_XXXX_NNNNN.wst</b></td><td>This file contains control information and each object in the application.</td></tr>
<tr><td><b>App_con_XXX</b></td><td>A list of all the connections either in the application or used by objects in the</td></tr>
</table></td></tr>
</table>

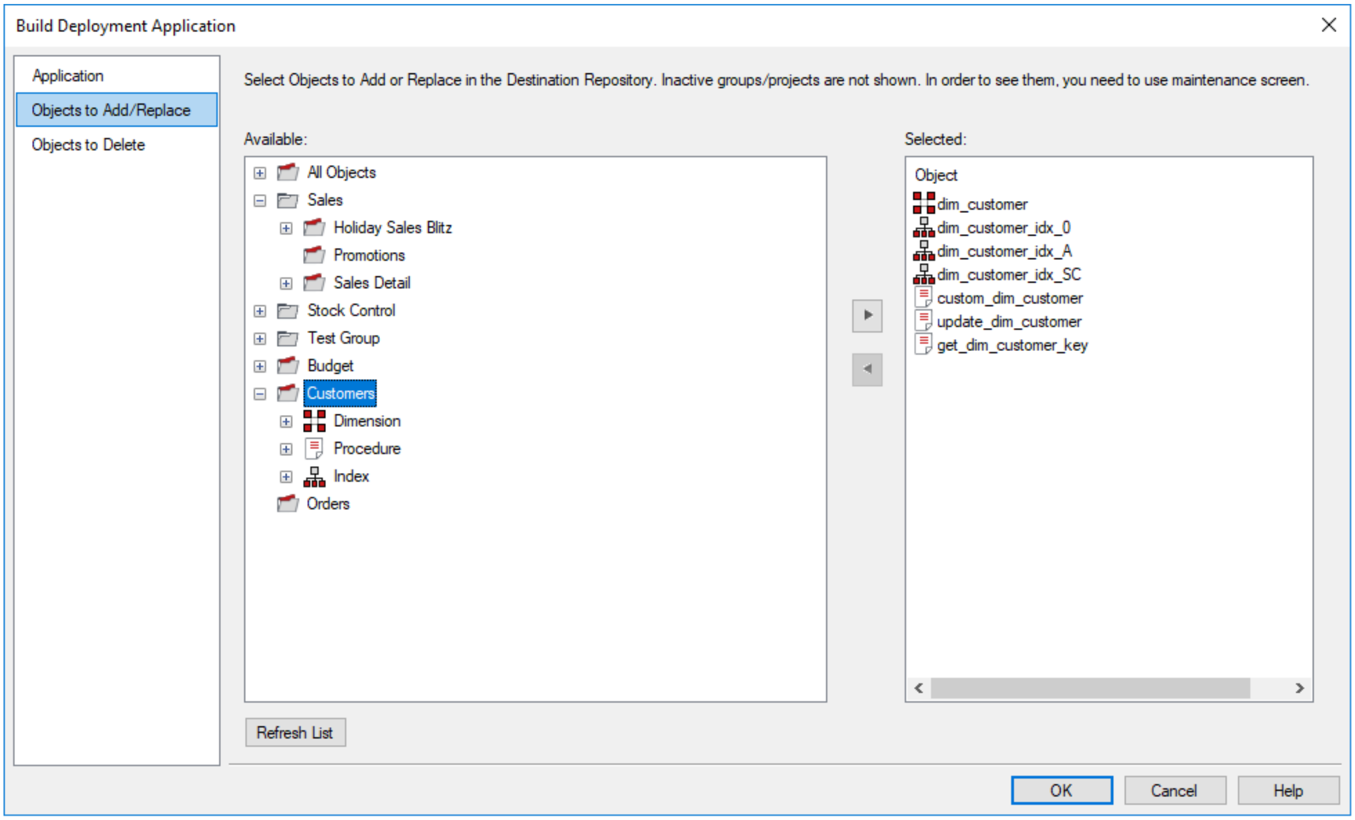
<table>
<tr><th>Fields</th><th>Description</th></tr>
<tr><td></td><td><table><tr><td><b>X_NNNNN.wst</b></td><td>application.</td></tr><tr><td><b>App_map_XXXX_NNNNN.wst</b></td><td>A list of all the project and group mappings for the objects in the application.</td></tr></table></td></tr>
<tr><td><b>Previous Application</b></td><td>Click the <b>Browse</b> button next to <b>Previous Application</b> to choose a previously built application to use as a list of objects to include in the new application. After using a previous application as a starting point for this application, additional objects can be added or removed from the application.</td></tr>
<tr><td><b>Pre application load SQL</b></td><td>This text box enables you to enter an SQL statement that is executed before the application is loaded. For example, you may wish to drop the date dimension before loading the application because we have changed the primary key constraint. In such a case, you would enter 'drop table dim_date' in this field to have the table dropped before the application is loaded.</td></tr>
<tr><td><b>Post application load SQL</b></td><td>This text box enables you to enter an SQL statement that is executed after the application is loaded. For example, you could execute a function to populate a table.</td></tr>
</table>

Objects to Add/Replace

This tab enables you to select the objects to add or replace in the destination repository. Double click a Project/Group folder to display the objects that are associated with them. Objects can be moved from the left object tree pane to the right pane by double-clicking an object or by using the **>** button.

Note:

Maximum number of objects in an application
5000 objects (including jobs)
2000 source views of views
1000 jobs.



Select Objects to Add or Replace in the Destination Repository. Inactive groups/projects are not shown. In order to see them, you need to use maintenance screen.

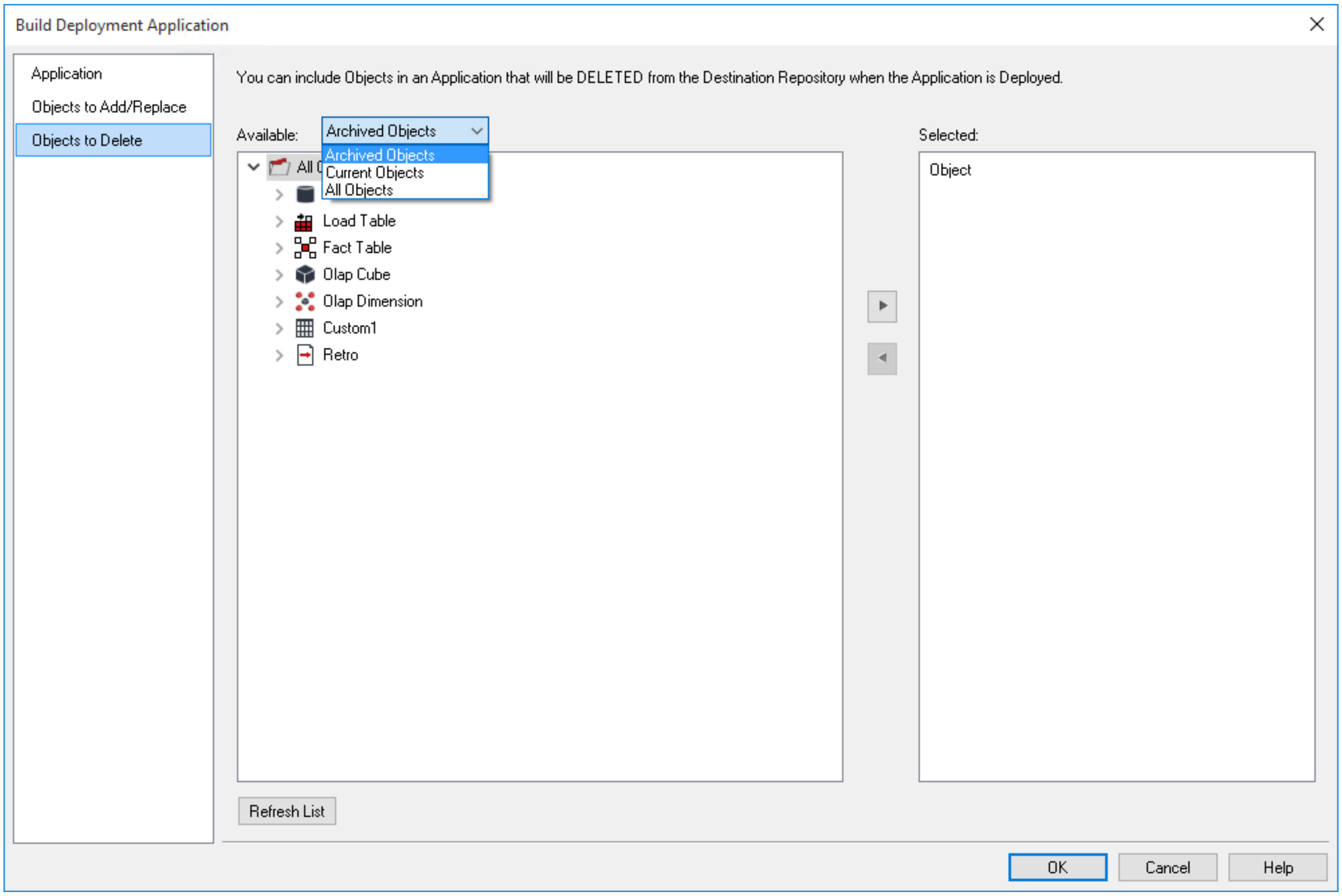
Object
dim_customer
dim_customer_idx_0
dim_customer_idx_A
dim_customer_idx_SC
custom_dim_customer
update_dim_customer
get_dim_customer_key

Notes:

- Projects or Groups that have been flagged as inactive are not displayed in the **Available** pane. Refer to **Maintaining Group/Project Active Flag** for details.
- You can select multiple objects using the Windows standard keyboard keys (CTRL or SHIFT) for multi selection.

Objects to Delete

This tab enables you to select the objects to delete in the destination repository. Click the **Available** drop-down field to select an option for displaying objects in left object tree pane, each option is described in the table below. Double click a Project/Group folder to display the objects that are associated with them. Objects can be moved from the left object tree pane to the right pane by double-clicking an object or by using the **>** button.



You can include Objects in an Application that will be DELETED from the Destination Repository when the Application is Deployed.

Options	Description
Archived Objects	This option includes all objects that have been versioned and deleted. This is the case in a development repository, where the object was removed after versioning. This is the default option.
Current Objects	This option includes all the currently available objects and their associated Projects or Groups.
All Objects	This option includes all the **Current Objects** plus the **Archived Objects** (deleted / versioned). Objects that were versioned and deleted will appear in **All Objects** but will not appear under any Projects or Groups.

Application Loading

Note

Applications can only be loaded into the same relational database type from which they where created.

Applications are loaded via the RED **Setup Administrator** utility. The normal process for implementing an application are as follows:

1. Run the **RED Setup Wizard** utility.
2. Change the application directory to the application's location.
3. Choose the level of metadata application. There are several levels, from load metadata only through to load metadata and apply changes to all tables.
4. Resolve any connections and tablespaces/filegroups to those present in the target environment.
5. Create/Re-create or Alter database tables, if selected in (5).

6. Compile database procedures, if selected in (5).
7. Turn OFF logging.
8. Review the output in the Setup Administrator utility.
9. Review the log file.

Note

Some database operations, such as converting an existing non-partitioned table to a partitioned table, cannot be done using a deployment application. In these cases, some manual intervention may be required to update the target databases to match the new metadata.

Refer to the RED Installation manual for more information about loading an application.

Creating and Loading Applications from the Command Line

It is possible to create and load applications from the command line by running a bat file. The WhereScape RED Application directory contains an example bat file WSL_Application_Create_Restore_Point_and_Load.bat for creating and loading.

Right-click this file and select **Edit** to see the steps outlined, as well as the details about the options available.

The first step **creates** a restore point application (R) based on objects about to be loaded. This process calls the command line functionality of RED and creates the application file.

The second step **loads** an application (A) into a test WhereScape RED repository. It uses an xml file to specify various options, calls the command line automation functionality of Setup Administrator and loads the application (A).

The WhereScape RED Application directory contains example xml files:

- WSL_Application_Load_SQL.xml for SQL Server

The tags in the xml file must be edited as the login/connection details, etc. must be correctly defined.

The Batch Application Create options and the Batch Application Load options are listed at the end of the batch file. The values for these variables must also be customized before running the file.

The process typically involves the following steps:

1. Create an application (A) in RED containing your data warehouse changes.

2. RUN WSL_Application_Create_Restore_Point_and_Load.bat from the command line; which creates a restore point application (R) and applies application (A) to a test WhereScape RED repository.

3. If the changes are incorrect, they can be undone by loading the restore point application (R).

Batch Application CREATE Options

The following options are available in RED to create applications.

Option	Description
/BA	Selects batch application create.
/U	ODBC user name.
/P	ODBC password.
/C	ODBC DSN name.
/A	DSN architecture
/N	RED User name.

Option	Description
/D	Directory to save application files.
/I	New application files identifier.
/V	New application files version.
/AP	Project name - all objects in a Project; and associated jobs.
/AG	Group name - all objects in a Group; and associated jobs.
/ALL	All objects - all objects, jobs and parameters.
/RC	Remove connections - drop all connections from the application.
/RJ	Remove jobs - drop all jobs from the application.
/RP	Remove parameters - drop all parameters from the application.
/AF	Absolute application id file name which restore point is being created for.

If using a trusted connection, the user and password are not necessary.

Batch Application LOAD Options

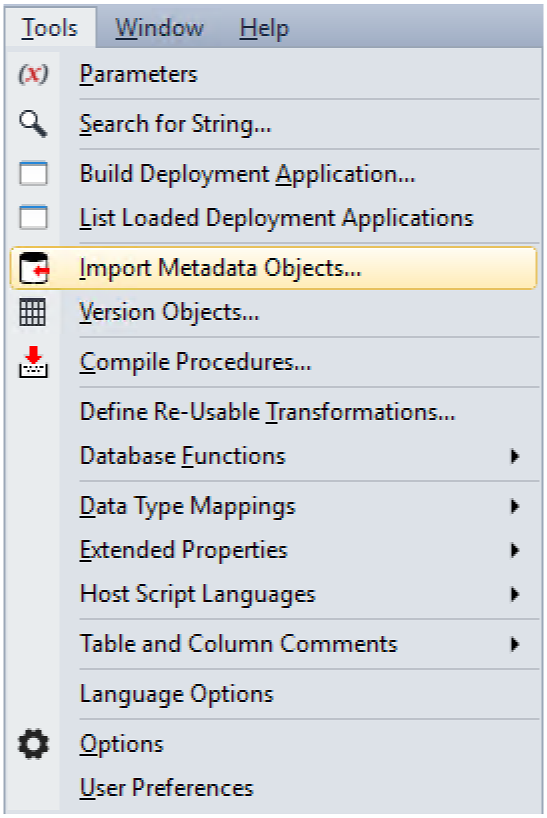
The following options are available in Setup Administrator to load applications.

Option	Description
/AL	Select application load.
/AF	Absolute application ID file name.
/LF	Absolute log file name.
/PF	Absolute xml parameter file name.
/SO	Stay open - Setup Administrator does not close after the application has been created. Relevant for /AL LOAD option. Relevant for -QA, -QR, -WS, -QL and -QV application switch options. Not relevant for -QS switch option. For switch options, refer to **SQL Server Quick Application** for details.

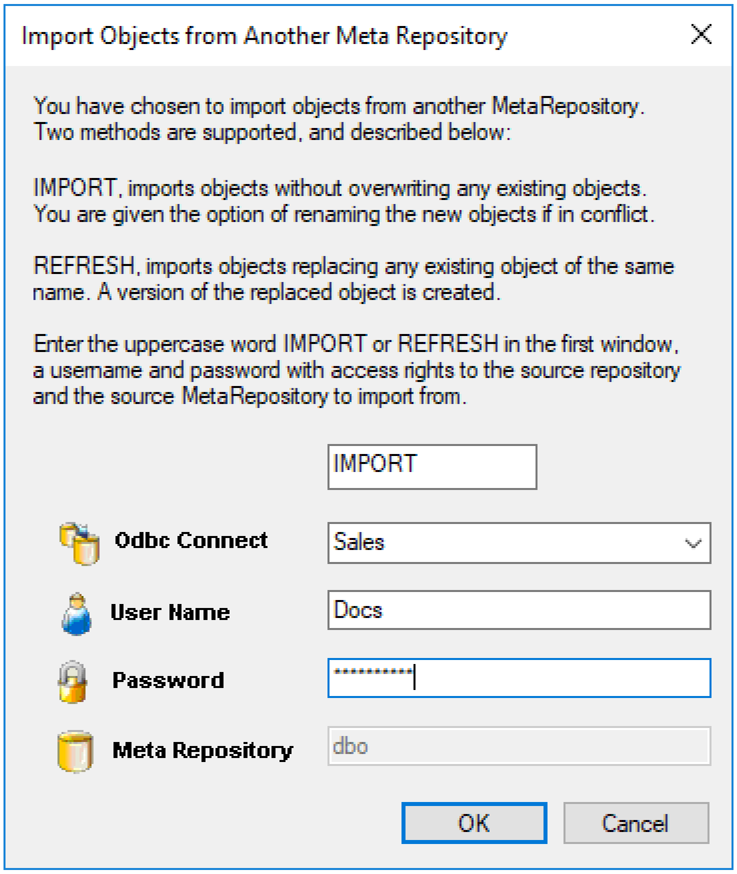
Importing Object Metadata

Any group of objects can be imported into the current metadata repository from another repository. If an object already exists in the target repository, then it is either skipped or replaced depending on the type of import undertaken. If an object is to be replaced as part of an import, a version of the object is created prior to its replacement.

To import an object or group of objects select the **Tools > Import Metadata Objects** menu option.



A window appears which provides two options, **IMPORT** or **REFRESH**. An import will not replace an existing object of the same name. A refresh will version and replace any existing object of the same name.



Import Objects from Another Meta Repository

You have chosen to import objects from another MetaRepository.
Two methods are supported, and described below:

IMPORT, imports objects without overwriting any existing objects.
You are given the option of renaming the new objects if in conflict.

REFRESH, imports objects replacing any existing object of the same name. A version of the replaced object is created.

Enter the uppercase word IMPORT or REFRESH in the first window, a username and password with access rights to the source repository and the source MetaRepository to import from.

IMPORT

Odbc Connect: Sales
User Name: Docs
Password: ***********
Meta Repository: dbo

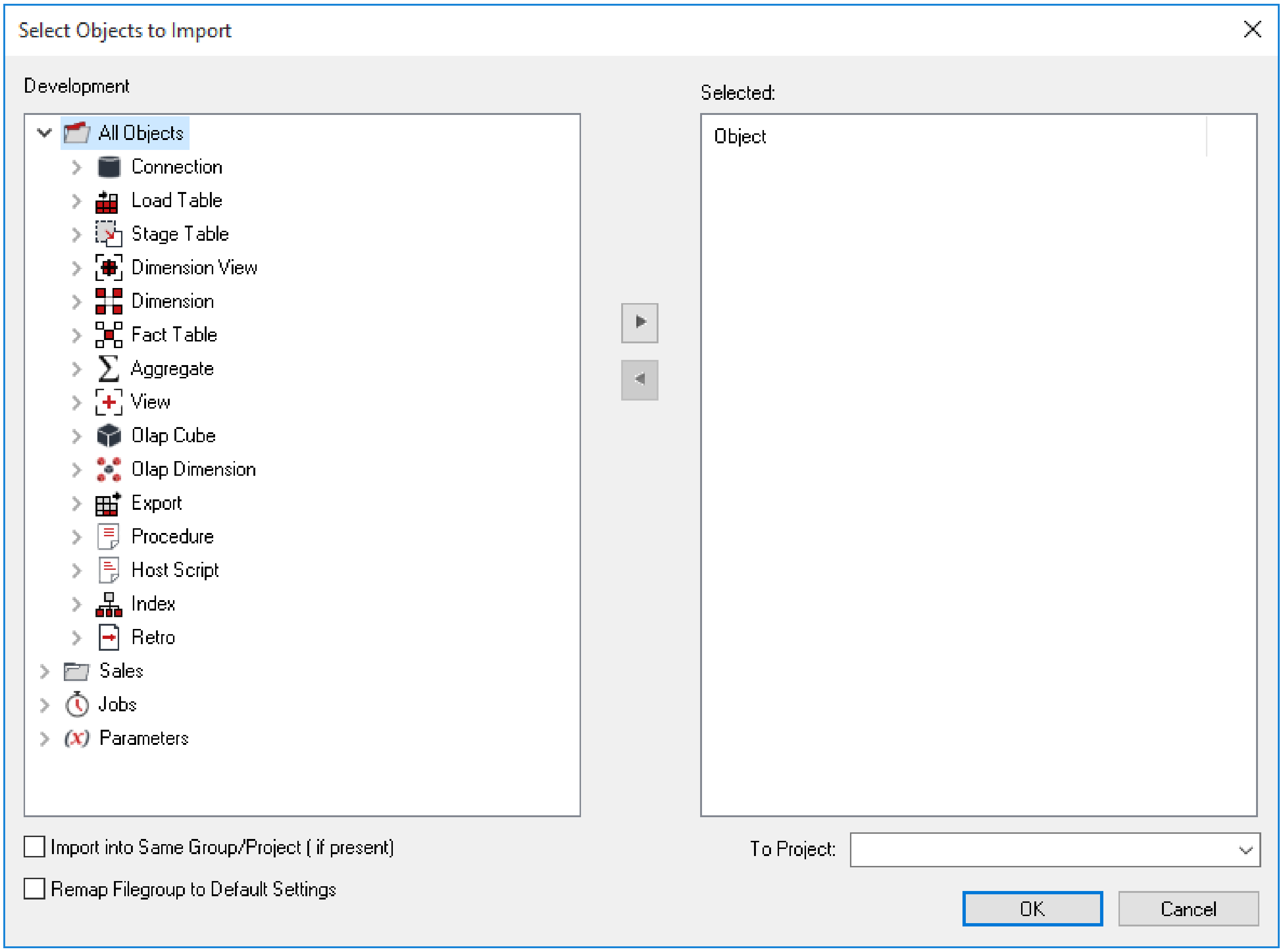
Enter the connection, and a database user name and password that has access to the source metadata repository.

Finally, enter the user name of the metadata repository you want to import from. In most situations, the **User Name** and **Meta Repository** would be the same.

However, if you only have read access to a meta repository, then it may be necessary to log in to the database under a different user name from that of the repository you are trying to import from.

You are not permitted to select the current meta repository in the **Meta Repository** field, but are permitted to log in using the existing repository username. Once a successful log on is completed, the contents of the source repository are loaded and the following window appears.

Select an object by double-clicking it or by selecting and using the **>** button. If an object such as a table is chosen, then any related scripts, procedures and indexes are also selected.
They can be removed if required. A target project can be selected.



Once all required objects are selected, click **OK** to start the import process.

On completion, a window appears which displays the number of each type of object imported, and skipped.

Note:

The repository from which you are importing must be the same metadata version as the target repository.

Importing Language Files

Note

Applications can only be loaded into the same relational database type from which they were created.

Language Files are loaded via the **Setup Administrator** utility. The normal process for implementing a Language file would be as follows:

1. Run the **Setup Administrator** utility.
2. Go to the **Languages** menu item in the top command bar and select **Load Languages**.
3. Right-click the Language file to be loaded and select **Install Language**.
4. Select the **ODBC** data source and Log on to the target meta repository.
5. Select the language to be updated.
6. Review the output in the **Setup Administrator** utility.

Refer to the RED Installation Guide for more information about loading a Language File.

Data Warehouse Testing

Note

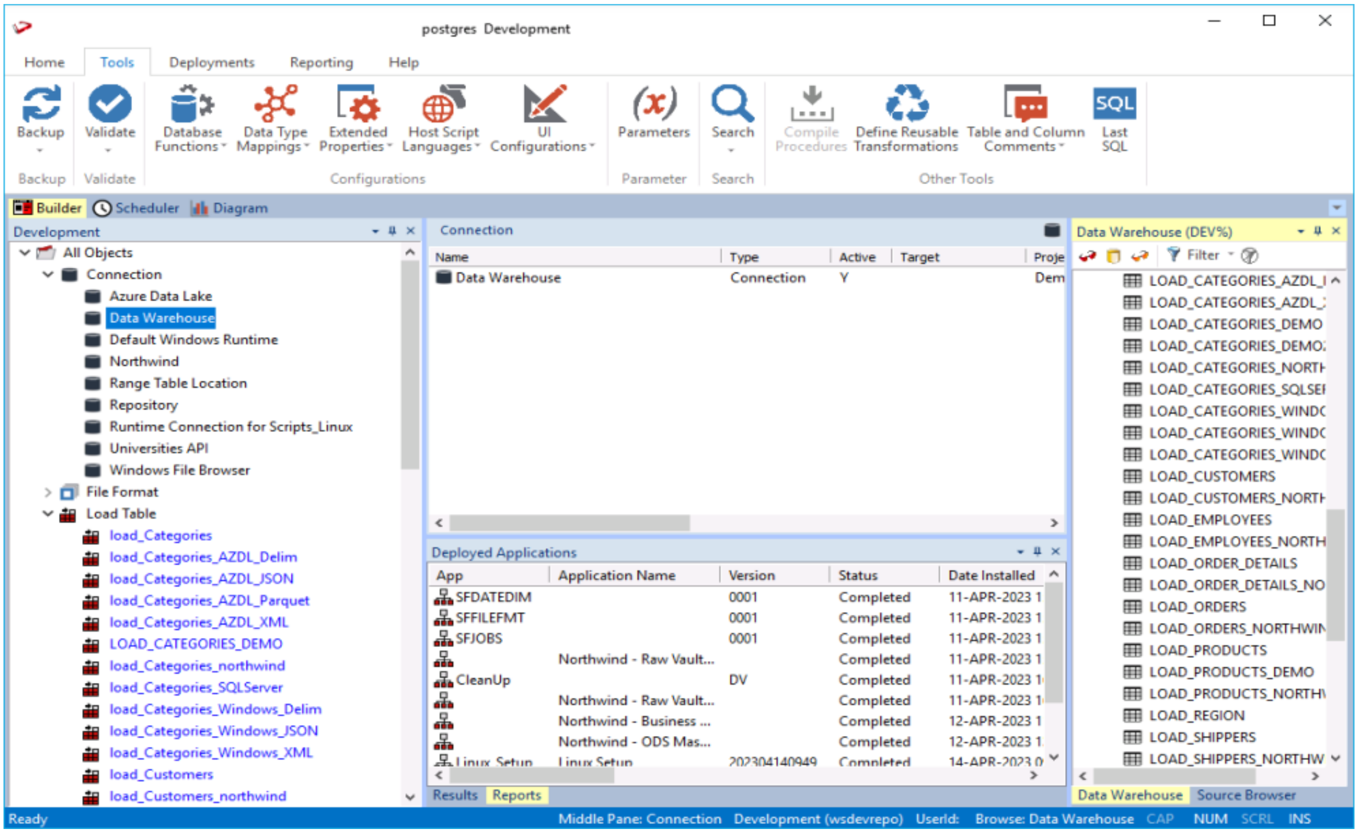
Applications can only be loaded into the same relational database type from which they where created.

Testing applications are loaded via the **Setup Administrator** utility. Refer to the RED Installation Guide for more information on how to load an application.

A testing application set consists of a Procedure and an XML script and provides the ability to define a series of tests against data warehouse objects; either comparing them to an expected value or to the results of a query.

Once the application set has been loaded, the Procedure and the XML script is visible in the left pane.

Procedure Objects List:



Development

- All Objects
  - Connection
    - Azure Data Lake
    - Data Warehouse
    - Default Windows Runtime
    - Northwind
    - Range Table Location
    - Repository
    - Runtime Connection for Scripts_Linux
    - Universities API
    - Windows File Browser
  - File Format
  - Load Table
    - load_Categories
    - load_Categories_AZDL_Delim
    - load_Categories_AZDL_JSON
    - load_Categories_AZDL_Parquet
    - load_Categories_AZDL_XML
    - LOAD_CATEGORIES_DEMO
    - load_Categories_northwind
    - load_Categories_SQLServer
    - load_Categories_Windows_Delim
    - load_Categories_Windows_JSON
    - load_Categories_Windows_XML
    - load_Customers
    - load_Customers_northwind

Connection

Name	Type	Active	Target	Proje
Data Warehouse	Connection	Y		Dem

Deployed Applications

App	Application Name	Version	Status	Date Installed
SFDATEDIM		0001	Completed	11-APR-2023 1
SFFILEFMT		0001	Completed	11-APR-2023 1
SFJOBS		0001	Completed	11-APR-2023 1
	Northwind - Raw Vault...		Completed	11-APR-2023 1
CleanUp		DV	Completed	11-APR-2023 1
	Northwind - Raw Vault...		Completed	11-APR-2023 1
	Northwind - Business ...		Completed	12-APR-2023 1
	Northwind - ODS Mas...		Completed	12-APR-2023 1
Linux_Setup	Linux Setup	202304140949	Completed	14-APR-2023 0

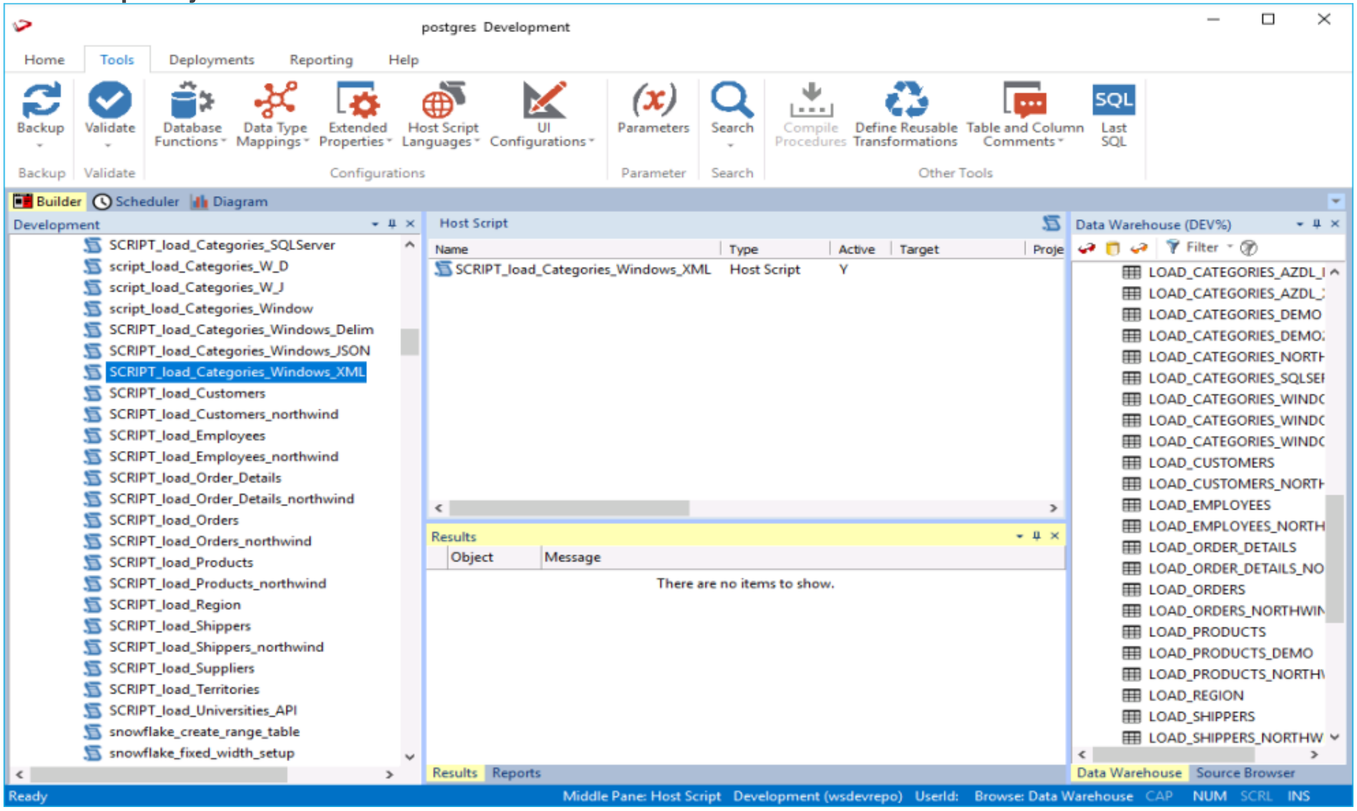
Data Warehouse (DEV%)

Filter

- LOAD_CATEGORIES_AZDL_I
- LOAD_CATEGORIES_AZDL_
- LOAD_CATEGORIES_DEMO
- LOAD_CATEGORIES_DEMO:
- LOAD_CATEGORIES_NORTH
- LOAD_CATEGORIES_SQLSEI
- LOAD_CATEGORIES_WINDC
- LOAD_CATEGORIES_WINDC
- LOAD_CATEGORIES_WINDC
- LOAD_CUSTOMERS
- LOAD_CUSTOMERS_NORTH
- LOAD_EMPLOYEES
- LOAD_EMPLOYEES_NORTH
- LOAD_ORDER_DETAILS
- LOAD_ORDER_DETAILS_NO
- LOAD_ORDERS
- LOAD_ORDERS_NORTHWIN
- LOAD_PRODUCTS
- LOAD_PRODUCTS_DEMO
- LOAD_PRODUCTS_NORTH\
- LOAD_REGION
- LOAD_SHIPPERS
- LOAD_SHIPPERS_NORTHW

Ready Middle Pane: Connection Development (wsdevrepo) UserId: Browse: Data Warehouse CAP NUM SCRL INS

Host Script Objects List:



Development

- SCRIPT_load_Categories_SQLServer
- script_load_Categories_W_D
- script_load_Categories_W_J
- script_load_Categories_Window
- SCRIPT_load_Categories_Windows_Delim
- SCRIPT_load_Categories_Windows_JSON
- SCRIPT_load_Categories_Windows_XML
- SCRIPT_load_Customers
- SCRIPT_load_Customers_northwind
- SCRIPT_load_Employees
- SCRIPT_load_Employees_northwind
- SCRIPT_load_Order_Details
- SCRIPT_load_Order_Details_northwind
- SCRIPT_load_Orders
- SCRIPT_load_Orders_northwind
- SCRIPT_load_Products
- SCRIPT_load_Products_northwind
- SCRIPT_load_Region
- SCRIPT_load_Shippers
- SCRIPT_load_Shippers_northwind
- SCRIPT_load_Suppliers
- SCRIPT_load_Territories
- SCRIPT_load_Universities_API
- snowflake_create_range_table
- snowflake_fixed_width_setup

Host Script

Name	Type	Active	Target	Proje
SCRIPT_load_Categories_Windows_XML	Host Script	Y		

Results

Object	Message

There are no items to show.

Data Warehouse (DEV%)

Filter

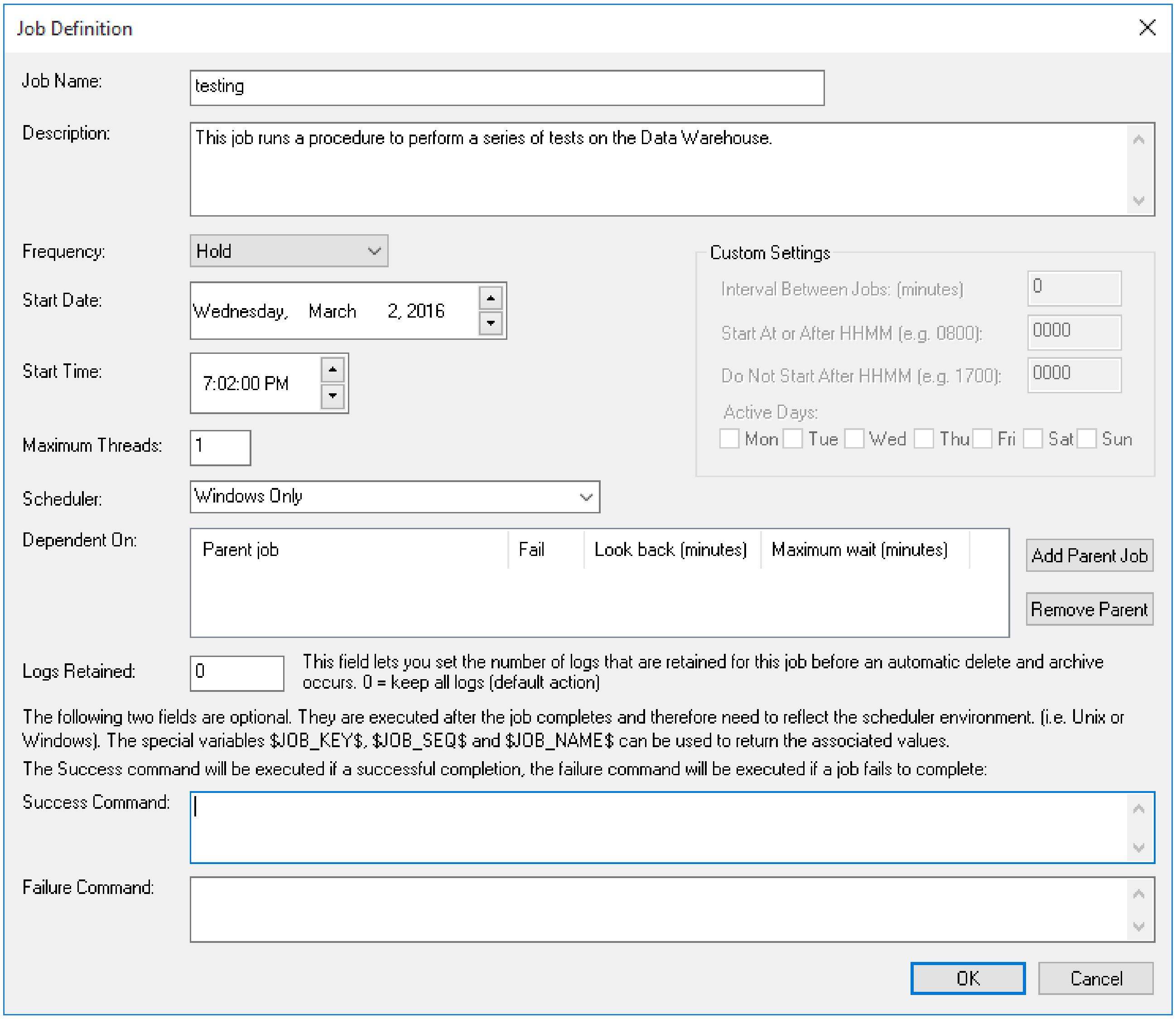
- LOAD_CATEGORIES_AZDL_I
- LOAD_CATEGORIES_AZDL_
- LOAD_CATEGORIES_DEMO
- LOAD_CATEGORIES_DEMO:
- LOAD_CATEGORIES_NORTH
- LOAD_CATEGORIES_SQLSEI
- LOAD_CATEGORIES_WINDC
- LOAD_CATEGORIES_WINDC
- LOAD_CATEGORIES_WINDC
- LOAD_CUSTOMERS
- LOAD_CUSTOMERS_NORTH
- LOAD_EMPLOYEES
- LOAD_EMPLOYEES_NORTH
- LOAD_ORDER_DETAILS
- LOAD_ORDER_DETAILS_NO
- LOAD_ORDERS
- LOAD_ORDERS_NORTHWIN
- LOAD_PRODUCTS
- LOAD_PRODUCTS_DEMO
- LOAD_PRODUCTS_NORTH\
- LOAD_REGION
- LOAD_SHIPPERS
- LOAD_SHIPPERS_NORTHW

Ready Middle Pane: Host Script Development (wsdevrepo) UserId: Browse: Data Warehouse CAP NUM SCRL INS

The XML script contains the test definitions. Each test is a new XML node in the comparison query. The procedure simply runs the test and determines whether the tests are passed or not. This is most likely to be run as a scheduled job within WhereScape RED.

To create a job:

1. Click the **Scheduler** button.
2. Choose **File** and then **New Job**.
3. Enter the definition of the job.



Job Definition

Job Name: testing

Description: This job runs a procedure to perform a series of tests on the Data Warehouse.

Frequency: Hold

Start Date: Wednesday, March 2, 2016

Start Time: 7:02:00 PM

Maximum Threads: 1

Scheduler: Windows Only

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)

Custom Settings

Interval Between Jobs: (minutes) 0

Start At or After HHMM (e.g. 0800): 0000

Do Not Start After HHMM (e.g. 1700): 0000

Active Days: □ Mon □ Tue □ Wed □ Thu □ Fri □ Sat □ Sun

Logs Retained: 0

This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

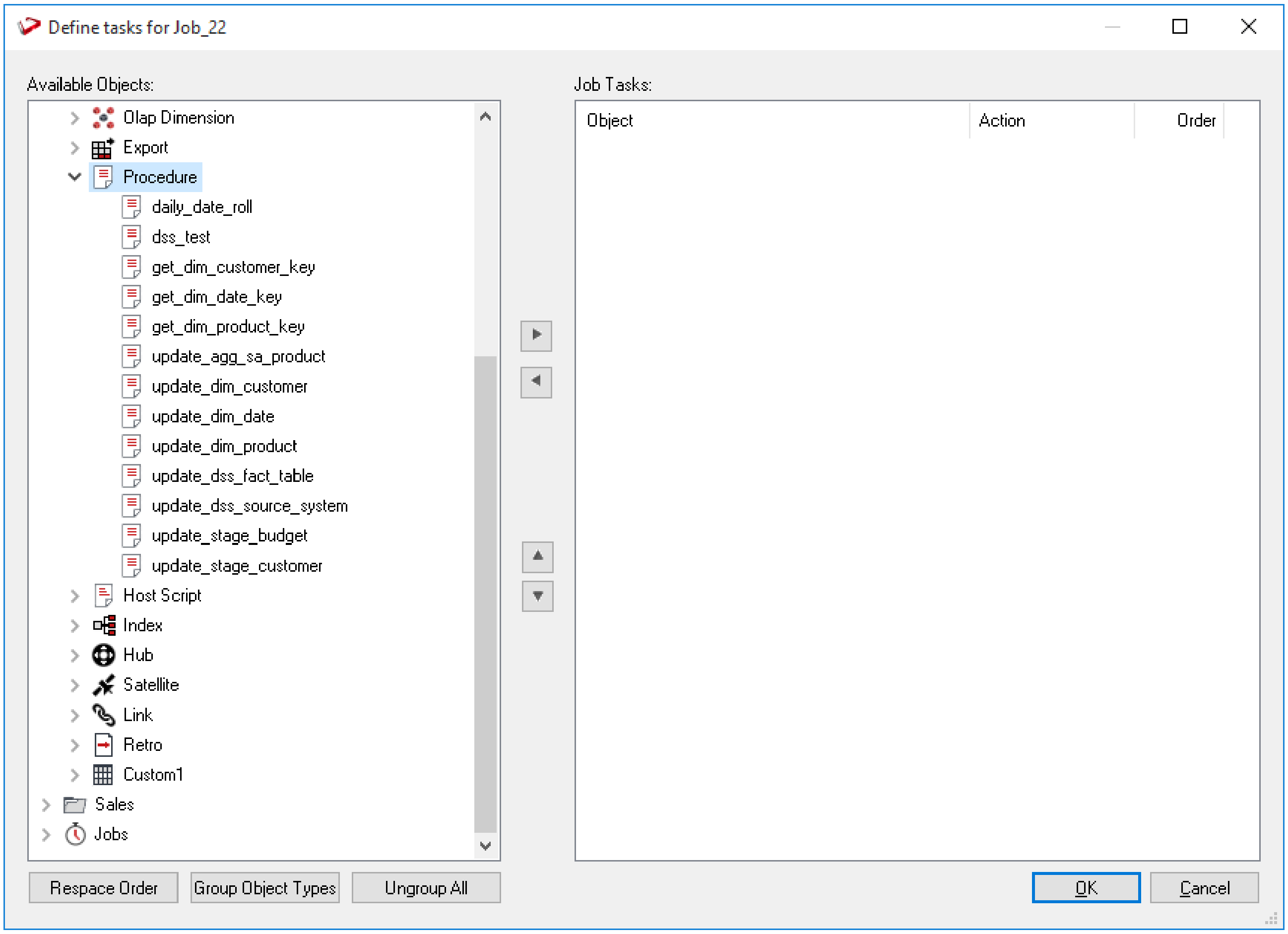
The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

Success Command:

Failure Command:

4. To select the test procedure as a task, open the Procedure object heading in the left pane. Choose **dss_test** and the **>** button. Click **OK**.



5. To run the job, click the **All Jobs** button and then right-click the job to select **Start the Job**.

Backing Up and Restoring Metadata

This chapter covers the saving and reloading of metadata repository objects. The backup section describes the methods for backing up the metadata repository. It can also be backed up via normal database backup procedures. The restore section covers the metadata restoration functions available.

Backup using DB Routines

The backup of the metadata repository can be undertaken as a separate exercise from the general backup of the data warehouse.

The Backup and Restore process uses PostgreSQL pg_dump and pg_restore respectively. The version of these tools installed with RED are at 12.4. If needed, edit the backup command line from the **Options** menu in RED to override the path to the executables or add parameters. To backup the metadata database from RED select menu item **Backup>Export** the Metadata (PostgreSQL pg_dump) and select a destination directory.

Restoring DB Backups

WhereScape RED metadata can be restored from a prior backup. For SQL Server data warehouses, the restore must take place in a Windows environment either through the WhereScape RED tool or manually.

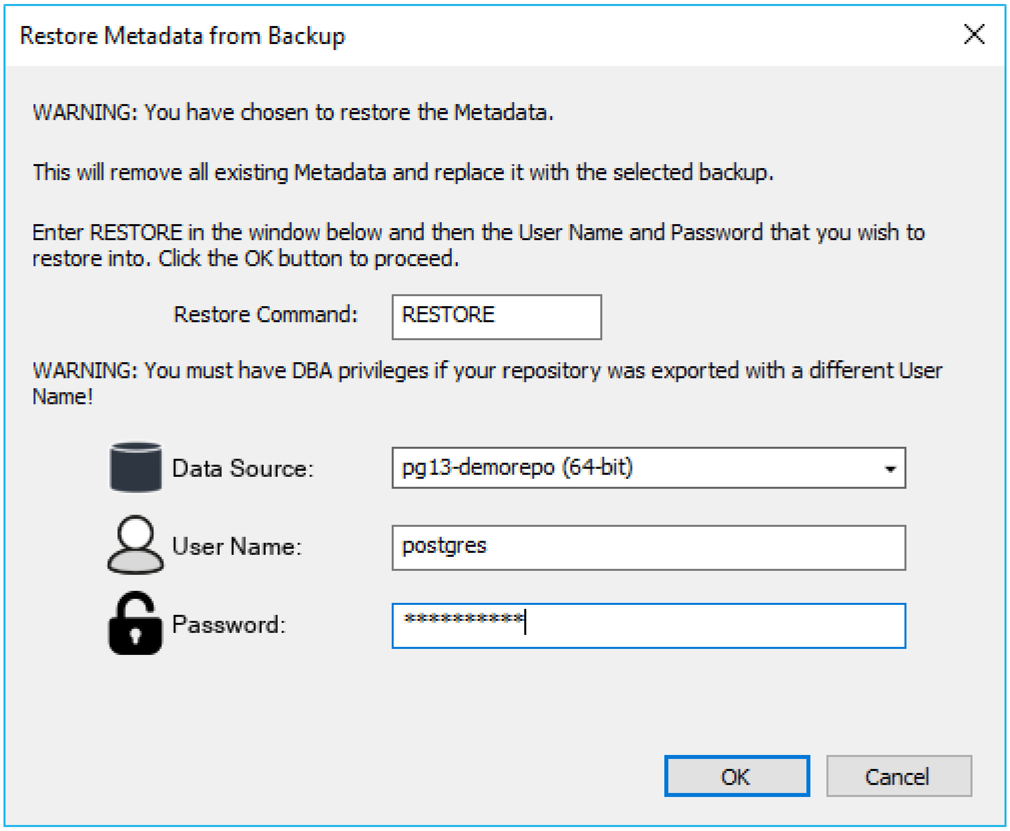
Restoring metadata

A backup of a metadata repository can be restored over the top of an existing repository. This action replaces the existing repository in its entirety. The restore of repositories is covered in an earlier section, but its uses in the promoting of code are discussed here. This restore process does not affect any existing database tables or any compiled procedures, so it can be used as a means of updating a metadata repository, albeit by replacing that repository. It is often a good method to choose when first establishing a new repository. For example, if we have a development repository and want to create a production environment the steps may be:

1. Backup the development repository.
2. Create a new repository using the **RED Setup Wizard**.
3. Log on to the newly created repository.
4. Restore the backup of the development repository into the new repository.
5. Modify any connections, set any default values, alter any table sizes or extents.
6. Create the tables.
7. Compile all procedures.

Restore from RED

To restore the metadata from a backup use the menu item **Backup>Restore the Metadata** (PostgreSQL pg_dump) and follow the prompts.



Restore Metadata from Backup

WARNING: You have chosen to restore the Metadata.

This will remove all existing Metadata and replace it with the selected backup.

Enter RESTORE in the window below and then the User Name and Password that you wish to restore into. Click the OK button to proceed.

Restore Command: RESTORE

WARNING: You must have DBA privileges if your repository was exported with a different User Name!

Data Source: pg13-demorepo (64-bit)

User Name: postgres

Password: ***********

Once the **OK** button is clicked, a new dialog box will appear asking for a selection of the export directory. Browse to the contents of the directory where the export is located. Once the export directory is selected, the import will begin. A dialog box will appear to show the results of the import

Altering Metadata

This chapter provides information on how to change and manipulate the data warehouse, once it has been established.

New source columns or changes to the source systems from which the data warehouse is built will require modifications to both the metadata and the data warehouse tables and procedures.

Validating Tables

The metadata as stored and maintained by WhereScape RED does not necessarily reflect the actual tables and procedures in use in the data warehouse. For example, if a new column is added to the metadata for a table, then that change is not automatically made in the actual physical table residing in the data warehouse. Likewise, if a column is deleted from the metadata, then that column may still exist in the physical database table.

This situation may be particularly apparent after an application patch or upgrade. The menu option **Validate > Validate Table Create Status**, and the right-click menu options in either the left or middle panes all provide a means of comparing the metadata to the physical tables in the database. A table, range of tables or all tables can be chosen. Each chosen table is a table in the metadata, and it is compared against the physical database table, if it exists.

The following example is the output from a validation.

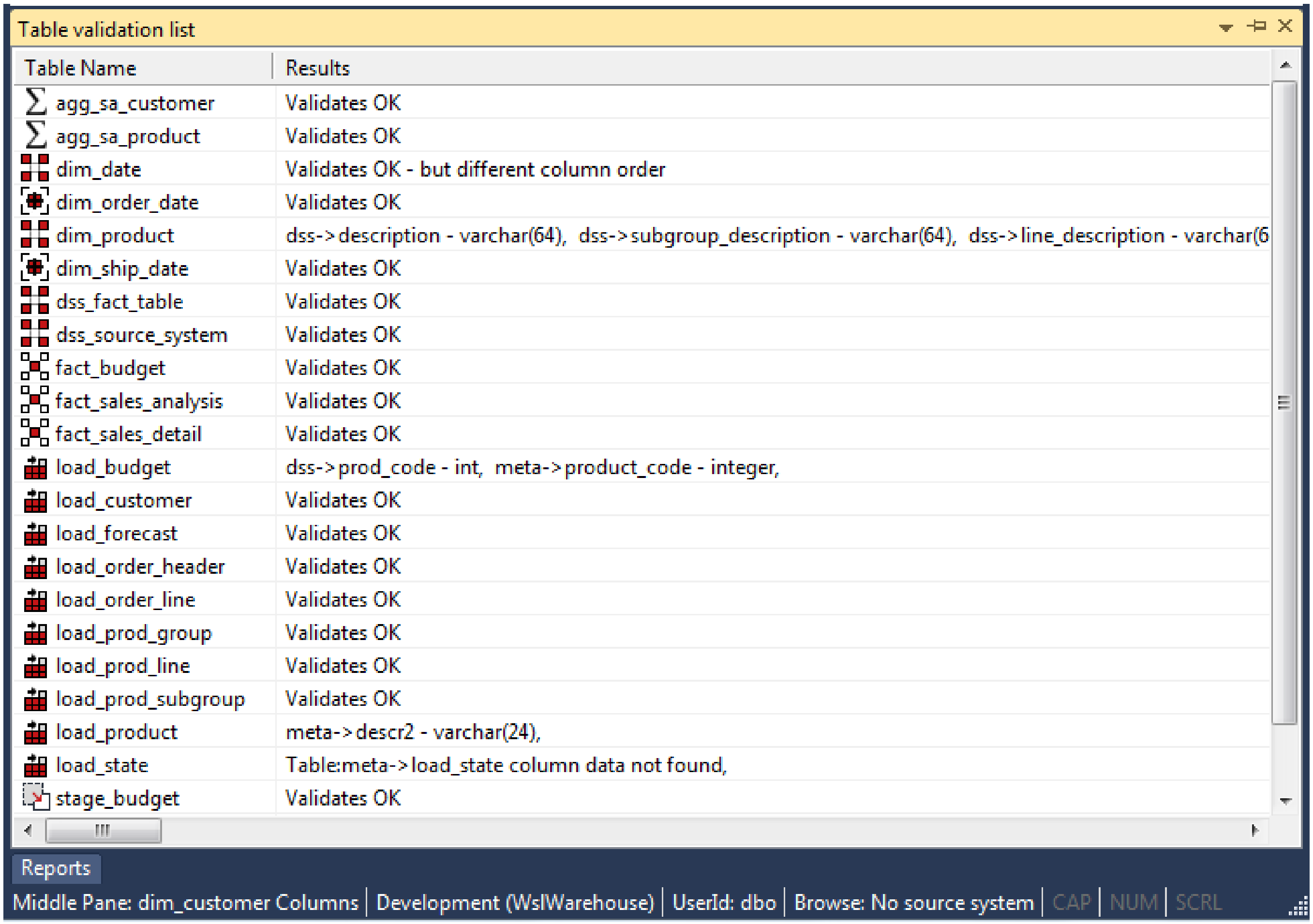


Table validation list

Table Name	Results
agg_sa_customer	Validates OK
agg_sa_product	Validates OK
dim_date	Validates OK - but different column order
dim_order_date	Validates OK
dim_product	dss->description - varchar(64), dss->subgroup_description - varchar(64), dss->line_description - varchar(6
dim_ship_date	Validates OK
dss_fact_table	Validates OK
dss_source_system	Validates OK
fact_budget	Validates OK
fact_sales_analysis	Validates OK
fact_sales_detail	Validates OK
load_budget	dss->prod_code - int, meta->product_code - integer,
load_customer	Validates OK
load_forecast	Validates OK
load_order_header	Validates OK
load_order_line	Validates OK
load_prod_group	Validates OK
load_prod_line	Validates OK
load_prod_subgroup	Validates OK
load_product	meta->descr2 - varchar(24),
load_state	Table:meta->load_state column data not found,
stage_budget	Validates OK

Middle Pane: dim_customer Columns | Development (WslWarehouse) | UserId: dbo | Browse: No source system | CAP | NUM | SCRL

In this example, we see five different scenarios.

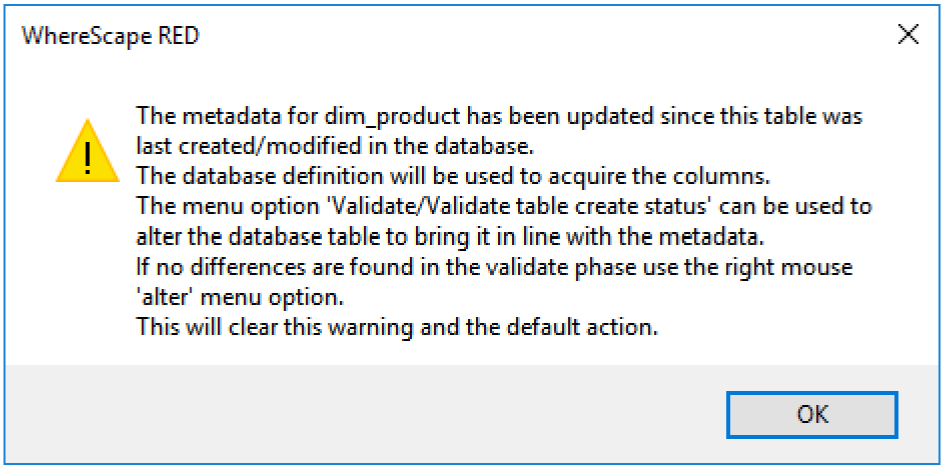
1. The metadata for table **agg_sa_customer** matches the physical table in the database.

2. The table **dim_date** has the same columns in both the metadata and the physical table, but the column order is different. This is probably not an issue for most tables but may be a problem for some type of load tables, where the column order is important. This could be the result of a previous altering of the table. The table must be re-created if the order is important.
3. The physical database table **dim_product** have additional columns not found in the metadata. The columns are 'subgroup_description' and 'line_description'. The table can be altered if desired. See the next section on Altering Tables.
4. The metadata for the table **load_product** does not match the physical table. The metadata has an additional column called 'state'. This column was not found in the physical table. The table can be altered if desired. See **Altering Tables** for details.
5. The table **load_state** is defined in the metadata but has not been physically created in the database. The table can be created in the normal manner.

Using outdated metadata in drag and drop

When dragging from a data warehouse table to create another data warehouse table (e.g. Load table to create Dimension table) a check is made to ensure that the metadata matches the database table.

If the two are found to be out of sync, the following message appears:



The metadata for dim_product has been updated since this table was last created/modified in the database.
The database definition will be used to acquire the columns.
The menu option 'Validate/Validate table create status' can be used to alter the database table to bring it in line with the metadata.
If no differences are found in the validate phase use the right mouse 'alter' menu option.
This will clear this warning and the default action.

If a subsequent validate of the table in question shows that it validates, this message means that the dates are somehow out of sync. This can occur for example, after an import where the metadata has been replaced, but the underlying table still matches the metadata. Another common occurrence is where a new column is added and then deleted. To prevent the message from reoccurring in such a situation, proceed as follows.

Use the right-click menu and select **Alter Table** when positioned on the table name in the validate results screen (event though the table validates OK). The metadata update time is set back to that of the last database table create.

Setup Validate against DB for Custom Database Targets

The new Validate Against DB functionality works in the following stages:

Stage	Operations	Configuration
Validate	1. Create the expected table, under a different name, based on the metadata. 2. Fetch information about the expected and actual tables. 3. Compare the actual and	Table/column information SQL Block Procedure. Configured on the Target Connection.

Stage	Operations	Configuration
	expected information.	
Alter	1. Generate a SQL Block procedure using a template. 2. Execute the SQL Block.	Alter DDL Template. Configured the Table. Default is configured on the Target connection.

Create DDL

The Create DDL for the table to be validated/altered MUST use any of the following tokens to refer to the name of the table: `$OBJECT$, $TABLE$ or [TABLEOWNER].[table_name]`.

Without these tokens the temporary "expected" table cannot created reliably as only these tokens can be replaced with temporary table names for the temporary table creation.

Note

- The Create DDL is usually generated by a template and so these tokens need to be specified in the template itself unless using the Override Create DDL.
- The generated Create DDL for a temporary table is validated to make sure it contains the unique temporary table name. If not for interactive validation the user is warned and given the chance to continue but forapplication deployment, the validation is skipped to avoid the potential dropping of a data warehouse table

Drop Table DDL

The Drop Table DDL statement MUST use any of the following tokens to refer to the name of the table: `$OBJECT$, $TABLE$`. Otherwise, the dropping of the temporary table after the validate process may drop the actual data warehouse table.

Table/Column information SQL Block

These are some of the characteristics that this procedure needs to be created:

- This is a SQL Block Procedure object that is configured on the target connection.
- SQL Blocks are a sub-type of the Procedure object in RED.
- The SQL Block must contain exactly two (2) separated statements by the configured end-of the statement string (<EOS>, by default).
- Each statement must return one result set.
- Each result set must contain the specified columns below in the correct order.
- The data types shown below are the maximum sizes to be processed.
- Additional information is returned in triples of extra columns at the end of the result set. This additional information is compared during validation and is provided to the template used to generate alter DDL.

Statement 1 - Table information

Columm	Type	Description
1	VARCHAR(128)	catalog
2	VARCHAR(128)	schema
3	VARCHAR(128)	table name
4 + (*i* * 3)	VARCHAR(128)	additional information key
5 + (*i* * 3)	VARCHAR(128)	additional information description

Columm	Type	Description
6 + (*i* * 3)	VARCHAR(128)	additional information value

Where *i* ranges from 0 to the number of additional information keys are exclusive.

Statement 2 - Column information

Columm	Type	Description
1	VARCHAR(128)	catalog
2	VARCHAR(128)	schema
3	VARCHAR(128)	table name
4	INTEGER	ordinal position
5	VARCHAR(128)	column name
6	VARCHAR(128)	data type
7 + (*i* * 3)	VARCHAR(128)	additional information key
8 + (*i* * 3)	VARCHAR(4000)	additional information description
9+ (*i* * 3)	VARCHAR(128)	additional information value

Where *i* ranges from 0 to the number of additional information keys are exclusive.

The following example uses the standard `information_schema` views. It is also the default if nothing is configured

```
SELECT table_catalog, table_schema, table_name
        FROM information_schema.tables
       WHERE UPPER(table_catalog) = UPPER('$DATABASE$')
         AND UPPER(table_schema) = UPPER('$SCHEMA$')
         AND UPPER(table_name) = UPPER('$TABLE$')
       ORDER BY table_catalog, table_schema, table_name
<EOS>
      SELECT table_catalog, table_schema, table_name, ordinal_position, column_name,
             CONCAT(data_type, CASE WHEN COALESCE(character_maximum_length, numeric_precision,
datetime_precision) IS NOT NULL
                                    THEN CONCAT('(',
                                                CONCAT(CAST(COALESCE(character_maximum_length,
                                                                     numeric_precision,
                                                                     datetime_precision) AS
VARCHAR(20)),
                                                       CONCAT(CASE WHEN numeric_scale IS NOT NULL
                                                                   THEN CONCAT(',
', CAST(numeric_scale AS VARCHAR(20)))
                                                                   ELSE ''
                                                              END,
                                                              ')')))
                                    ELSE ''
                               END) data_type,
             'COLUMN_DEFAULT', 'Default Value', column_default,
             'NULLABLE', 'Nullable', is_nullable
        FROM information_schema.columns
       WHERE UPPER(table_catalog) = UPPER('$DATABASE$')
         AND UPPER(table_schema) = UPPER('$SCHEMA$')
         AND UPPER(table_name) = UPPER('$TABLE$')
       ORDER BY table_catalog, table_schema, table_name, ordinal_position
```

Alter DDL Template

- This is a template that is configured on each table.
- The default for new tables is taken from the default template configured on the target connection.
- The template must be set as Type: 'Alter'
- The result of the template is interpreted as a sequence of SQL Blocks separated by two newlines.
  - Each SQL Block is executed independently; the failure of one does not prevent the execution of the following.
  - Within each SQL Block there can be multiple statements separated by "<EOS>" or the configured separator.
    - A failure of a statement within an SQL Block does prevent the execution of the following statements in that block.
  - The usual variables are available, e.g. table, options, env.

In addition, the following variables are available:

Variable	Description
changes	An array of objects describing the ways in which the actual table is different to the expected table. Each element has type Change.
actualTable	An object containing the table/column information obtained for the actual table. Has type TableInformation.

'Change' object type

Field	Description
changeType	One of the enum values in Types. ChangeType; determines the type of the object under data.
data	An object containing the details of the specific change; the type depends on the value of changeType.

Change types

Change type	Description	Fields in data
Types.ChangeType.TableCreation	Indicates that the actual table does not exist.	(none)
Types.ChangeType.TableAdditionalInformationChange	Indicates that one of the additional information fields returned by the Table Information statement differs.	`changedKey` The "key" of the additional information. `changedDescription` The "description" of the additional information. `actualValue` The "value" of the additional information in the actual table. `expectedValue` The "value" of the additional information in the expected table.
Types.ChangeType.ColumnOrderChange	Indicates that the columns differ only in their order.	(none)
Types.ChangeType.ColumnAddition	Indicates that one of the columns in the expected table is not present in the actual table.	`columnName` The name of the column in the expected table.
Types.ChangeType.ColumnDeletion	Indicates that one of the columns in the actual table is not present in the expected table.	`columnName` The name of the column in the actual table.

Change type	Description	Fields in data
Types.ChangeType.ColumnNameChange	Indicates that the name of the one of the columns differs.	`columnName` The name of the column in the expected table. `actualColumnName` The name of the corresponding column in the actual table.
Types.ChangeType.ColumnDataTypeChange	Indicates that the data type of one of the columns differs.	`columnName` The name of the column (in the expected table). `dataType` The data type of the column in the expected table.
Types.ChangeType.ColumnAdditionalInformationChange	Indicates that one of additional information fields returned by the Column Information statement differs for one column.	`columnName` The name of the column whose additional information differs. `changedKey` The "key" of the additional information. `changedDescription` The "description" of the additional information. `actualValue` The "value" of the additional information in the actual table. `expectedValue` The "value" of the additional information in the expected table.

'TableInformation' object type

Variable	Description
catalog	name of the catalog containing the table
schema	name of the schema containing the table
name	name of the table
additionalInformation	array of AdditionalInformation objects
columns	array of ColumnInformation objects

'ColumnInformation' object type

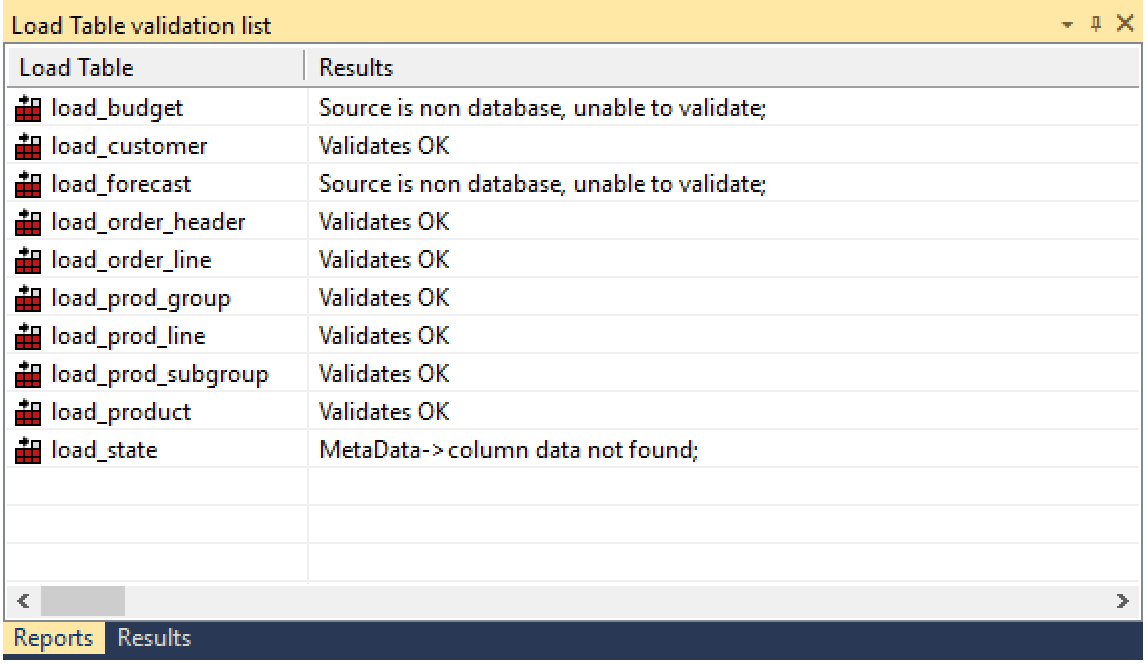
Variable	Description
name	name of the column
dataType	data type of the column
additionalInformation	array of AdditionalInformation objects

'AdditionalInformation' object type

Variable	Description
key	additional information "key"
description	additional information "description"
value	additional information "value"

Validating Source (Load) Tables

Changes to the source systems from which the data warehouse is built can be detected to a limited degree. The menu option **Validate > Validate Load Table Status** enables a comparison between load tables and the source tables from which they were built. This comparison is not available for flat file or script-based loads. A Load table or group of load tables are selected, and the results are displayed in the middle pane. An example screen from a Load table validate is as follows:



Load Table	Results
load_budget	Source is non database, unable to validate;
load_customer	Validates OK
load_forecast	Source is non database, unable to validate;
load_order_header	Validates OK
load_order_line	Validates OK
load_prod_group	Validates OK
load_prod_line	Validates OK
load_prod_subgroup	Validates OK
load_product	Validates OK
load_state	MetaData->column data not found;

The tables **load_budget** and **load_forecast** are Windows file loads and as such, cannot be validated.

If a table shows additional columns in the source table; such a scenario will not cause problems for the continued operation of the data warehouse. It simply means that more columns are present in the source table than have been loaded into the data warehouse. This may have been the result of an initial partial selection or as a result of new columns. Further investigation of the source table would be required to ascertain if there was new information available.

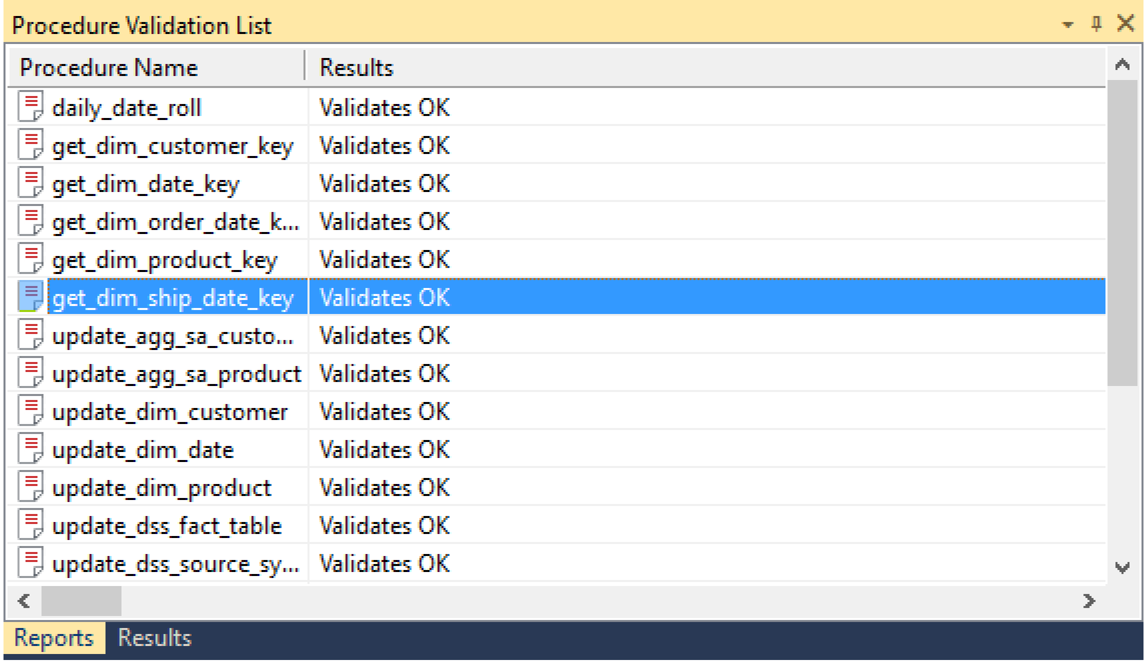
The table **load_state** reflects a problem for the continued operation of the data warehouse. The source table does not have a column that was previously identified as having come from that table. This will probably cause the load of that table to fail. This scenario would also require an investigation into the source table. The resolution may be to delete the column. The potential impact on later tables (Dimension, Stage and Fact) and procedures in the data warehouse can be ascertained by using the right-click menu of a Load table.

Validating Procedures

The menu option **Validate > Validate Procedure Status** compares procedures as stored in the metadata with those compiled and running in the data warehouse. This option provides a listing in the middle pane of each selected procedure and its status. The status will be **Validates OK**, **Not compiled**, or **Compare failed**.

Where a procedure is marked as **Not compiled**, this means that the procedure exists in the metadata but has not been compiled into the database.

Where a procedure fails to compare, the Procedure Editor must be used to find the actual differences. Start the editor by double-clicking the procedure object from the Objects list in the left pane. Use the **Tools > Compare to Compiled Source** menu option, to display the differences between the procedure in the metadata and that compiled in the database.



Procedure Validation List

Procedure Name	Results
daily_date_roll	Validates OK
get_dim_customer_key	Validates OK
get_dim_date_key	Validates OK
get_dim_order_date_k...	Validates OK
get_dim_product_key	Validates OK
get_dim_ship_date_key	Validates OK
update_agg_sa_custo...	Validates OK
update_agg_sa_product	Validates OK
update_dim_customer	Validates OK
update_dim_date	Validates OK
update_dim_product	Validates OK
update_dss_fact_table	Validates OK
update_dss_source_sy...	Validates OK

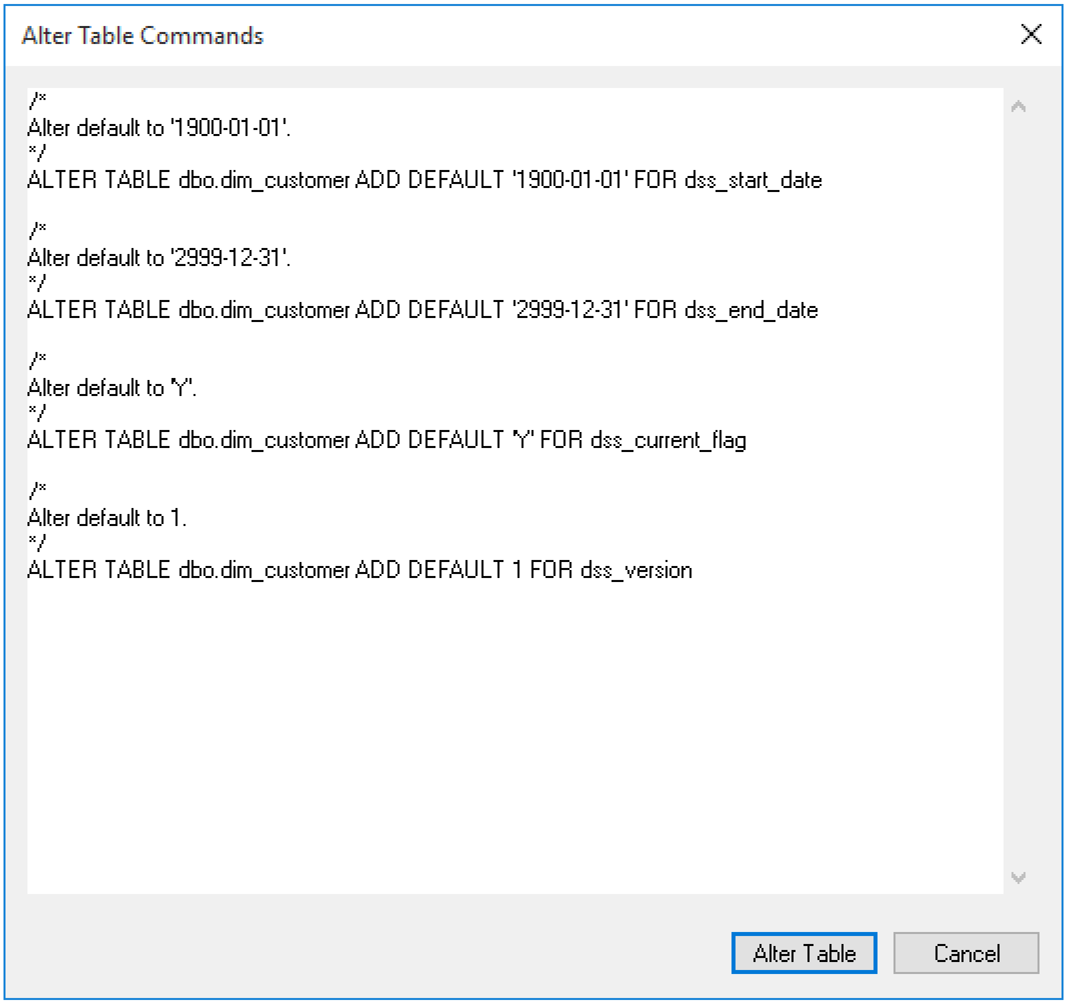
Altering Tables

The previous section covered the process of validating a table as defined in the metadata with the physical table as defined in the database. If a table is found to be different it is possible to alter the table.

Note

Caution should be taken when altering large data warehouse tables. A number of factors such as the time required to perform the alter, access to the table and the optimum storage of the table come into play.

To alter a table, first validate the table through the **Validate > Validate Table Create Status** menu option, or use the right-click menu option from the object name. Then in the middle pane (the validation listing) right-click the table that has not validated and select **Alter table**. A screen similar to the one below appears advising of the planned changes.



```
/*
Alter default to '1900-01-01'.
*/
ALTER TABLE dbo.dim_customer ADD DEFAULT '1900-01-01' FOR dss_start_date

/*
Alter default to '2999-12-31'.
*/
ALTER TABLE dbo.dim_customer ADD DEFAULT '2999-12-31' FOR dss_end_date

/*
Alter default to 'Y'.
*/
ALTER TABLE dbo.dim_customer ADD DEFAULT 'Y' FOR dss_current_flag

/*
Alter default to 1.
*/
ALTER TABLE dbo.dim_customer ADD DEFAULT 1 FOR dss_version
```

In this example, the **dim_product** table is to be altered. Comments at the top of the screen, show the reason(s) for the alteration and the actual alter command(s) follow.

The alter table window is an edit window. The command to be executed can be changed or additional commands entered. The command may also be cut to be executed in some other environment or at some later stage.

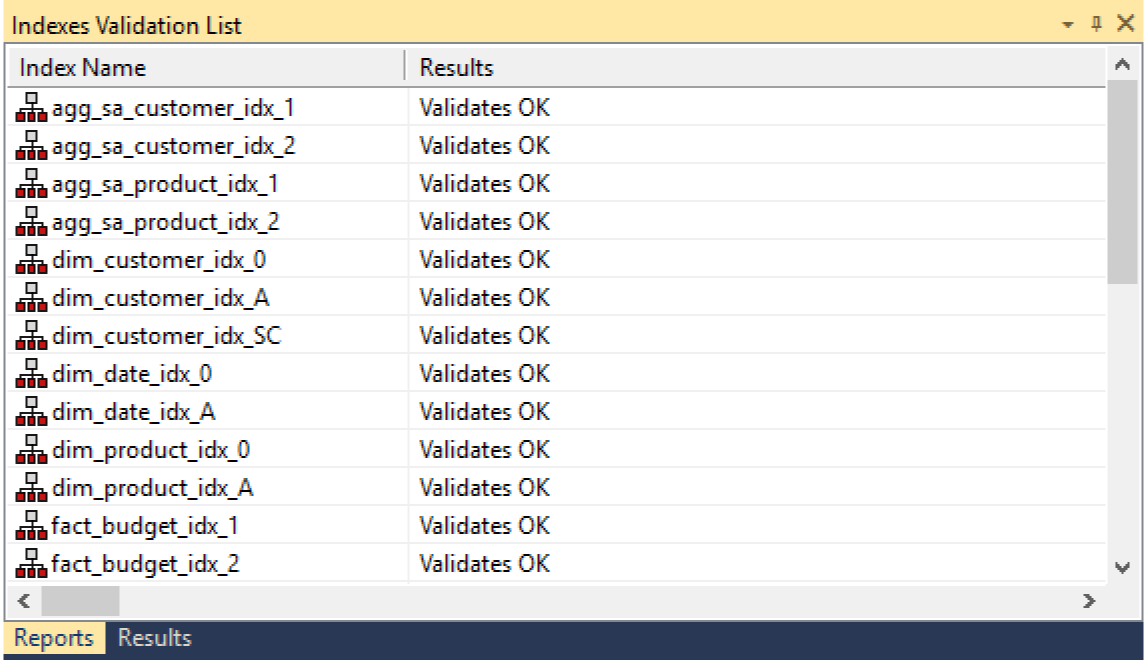
Clicking the **Alter Table** button alters the table. In effect, it will execute any command in the window.

Validating Indexes

The menu option **Validate > Validate Index Status** checks the metadata definition of an index against that in use in the database. It checks to ensure that the index exists and that the index type and columns are the same.

The results are listed in the middle pane with the status of each selected index shown.

If an index differs, then the only option is to drop and create the index either via the scheduler or through the right-click menu options for the index.

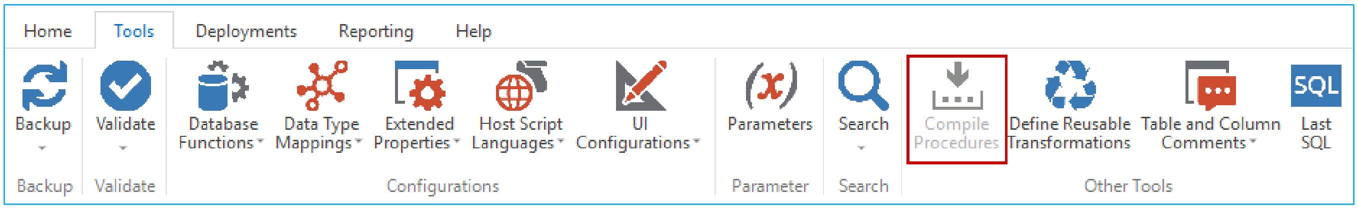


Indexes Validation List

Index Name	Results
agg_sa_customer_idx_1	Validates OK
agg_sa_customer_idx_2	Validates OK
agg_sa_product_idx_1	Validates OK
agg_sa_product_idx_2	Validates OK
dim_customer_idx_0	Validates OK
dim_customer_idx_A	Validates OK
dim_customer_idx_SC	Validates OK
dim_date_idx_0	Validates OK
dim_date_idx_A	Validates OK
dim_product_idx_0	Validates OK
dim_product_idx_A	Validates OK
fact_budget_idx_1	Validates OK
fact_budget_idx_2	Validates OK

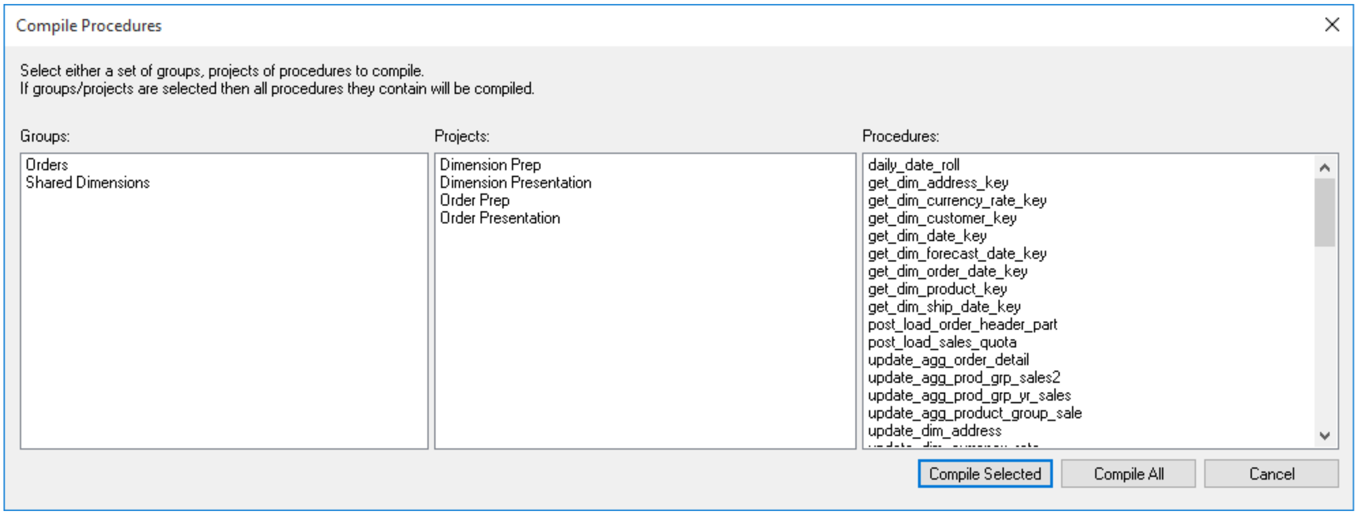
Recompiling Procedures

Procedures can be invalidated as a result of changes to underlying tables, or child procedures. A procedure can be recompiled through the procedure editor or via the menu option **Tools > Compile Procedures**:



.

The **Compile Procedures** window is displayed :



Compile Procedures

Select either a set of groups, projects of procedures to compile.
If groups/projects are selected then all procedures they contain will be compiled.

Groups:

Orders
Shared Dimensions

Projects:

Dimension Prep
Dimension Presentation
Order Prep
Order Presentation

Procedures:

daily_date_roll
get_dim_address_key
get_dim_currency_rate_key
get_dim_customer_key
get_dim_date_key
get_dim_forecast_date_key
get_dim_order_date_key
get_dim_product_key
get_dim_ship_date_key
post_load_order_header_part
post_load_sales_quota
update_agg_order_detail
update_agg_prod_grp_sales2
update_agg_prod_grp_yr_sales
update_agg_product_group_sale
update_dim_address

All procedures in the metadata can be compiled, or a selected group of procedures may be compiled. Selection is done via standard Windows multi selection, using the shift and CTRL keys. By selecting a project or group it is possible to compile all procedures associated with the project or group.

As each procedure is compiled, it is displayed in the middle pane of the Builder window. If a procedure fails to compile, a failure message is displayed along side the procedure name. Procedures that fail to compile will need to be investigated through the procedure editor, as no specific error information is provided by this bulk compiler.

Callable Routines

Introduction to Callable Routines

Callable Routines API

WhereScape RED callable routines provide an Application Program Interface (API) to the WhereScape RED metadata using the following SQL-invoked routines:

Routine Name	Description
Ws_Api_Glossary	Adds an entry to the documentation glossary.
Ws_Connect_Replace	Replaces the contents of a connection with details from another connection.
Ws_Load_Change	Changes the Connection or Schema of a load table.
Ws_Version_Clear	Purges metadata versions for all objects that do not meet the specified retention criteria.
WsParameterRead	Returns the value and comment (for most RDBMS) of a WhereScape RED metadata Parameter.
WsParameterReadF	Returns the value of a WhereScape RED metadata Parameter.
WsParameterReadG	Returns the value of a "global" WhereScape RED metadata Parameter that relates to a load table.
WsParameterWrite	Updates the value and comment of a WhereScape RED metadata Parameter or creates it.
WsWrkAudit	Records a message in the Audit Log.
WsWrkAuditBulk	Records multiple messages in the Audit Log.
WsWrkError	Records a message in the Error/Detail Log.
WsWrkErrorBulk	Records multiple messages in the Error/Detail Log.
WsWrkTask	Updates row counts for a task in the Task Log.

Callable Routines Names Qualifier

For PostgreSQL it is necessary to qualify the routine name with the schema name 'red' of the WhereScape RED metadata repository. All RED-generated procedures in a PostgreSQL repository that invoke a WhereScape RED Callable Routine do so by qualifying the routine name with **[METABASE]** e.g. **[METABASE]**.routine_name.

Callable Routines Common Input

The following input parameters are common to most of the WhereScape RED Callable Routines, which are primarily used for integration with the WhereScape RED Scheduler.

Input	Parameter Name	Description
Job Instance Identifier	p_sequence	Unique identifier of the **running job** (i.e. the running instance of a held or scheduled job) that executed the routine. • When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. • When invoked manually or externally to the WhereScape RED Scheduler, then any integer value can be used.
Job Name	p_job_name	Name of the **running job** that executed the routine.

Input	Parameter Name	Description
		- When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. - When invoked manually or externally to the WhereScape RED Scheduler, then any name can be used.
Task Name	p_task_name	Name of the **running task** (of a running job) that executed the routine. - When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. - When invoked manually or externally to the WhereScape RED Scheduler, then any name can be used.
Job Identifier	p_job_id	Unique identifier of the **held or scheduled job** that the running job is a specific instance of. - When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. - When invoked manually or externally to the WhereScape RED Scheduler, it is recommended to use 0 (zero).
Task Identifier	p_task_id	Unique identifier of the **running task** (of a running job) that executed the routine. - When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. - When invoked manually or externally to the WhereScape RED Scheduler, it is recommended to use 0 (zero).

Note

Typically, the **parameter names** of the WhereScape RED Callable Routines use a p_ prefix as indicated but in some routines a v_ prefix is used instead.

Ws_Api_Glossary

Synopsis

Adds an entry to the documentation glossary.

Description

Adds the specified entry to the documentation glossary, which is included in documentation that is subsequently generated by WhereScape RED.

Input

Input	Description
Object Name	Item that appears in the left column of the glossary, which normally represents the business name of a column in a dimension or fact table (although it can be used for other purposes).
Glossary Term	Item that appears under the analysis area heading of the glossary, which normally represents a dimension ot fact table name (although it can be used for other purposes).
Glossary Description	Description of the term being defined.
Action	Either **ADD** or **DELETE** the specified glossary entry.

Output

Output	Description
Result Text	A message to indicate whether or not the glossary term was successfully added/deleted.

PostgreSQL

PostgreSQL Parameters: Ws_Api_Glossary

Callable Routine Type:FUNCTION.

Parameter Name	Datatype	Mode
p_object_name	VARCHAR(64)	IN
p_term	VARCHAR(256)	IN
p_comments	VARCHAR(4000)	IN
p_option	VARCHAR(12)	IN
p_result	VARCHAR(256)	OUT

Ws_Connect_Replace

Synopsis

Replaces the contents of a connection with the details from another connection.

Description

Copies the details of the specified Source connection to the specified Target connection.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Action	**REPLACE** the details of the specified Target connection.
Source Connection	The name of the Source connection whose details will be copied.
Target Connection	The name of the Target connection whose details will be changed.

Output

Output	Description
Return Code	Output Return Code: • S Success. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • -2 Error. • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: Ws_Connect_Replace

Callable Routine Type:FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_action	VARCHAR(64)	IN
p_source	VARCHAR(64)	IN
p_target	VARCHAR(64)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Ws_Job_Abort

Synopsis

Aborts a job if it is in a running state.

Description

Aborts the specified job if it is in a running state, which changes it to a failed state, fails all running tasks, and holds all waiting tasks.

Input

Input	Description
Abort Job Name	The name of the job to be aborted. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler. The specified job must be in a running or failed state in order to be aborted.
Job Instance Identifier	Unique identifier of the running job (that may be in a failed state).
Abort Job Message Text	Custom message text to be recorded in the WhereScape RED Audit Log for the aborted job and the WhereScape RED Task Log for each aborted task.

PostgreSQL

PostgreSQL Parameters: Ws_Job_Abort

Callable Routine Type:FUNCTION.

Parameter Name	Datatype	Mode
p_job_name	VARCHAR(64)	IN
p_job_sequence	INTEGER	IN
p_job_msg	VARCHAR(256)	IN

Ws_Job_Clear_Archive

Synopsis

Purges archived job logs that are older than the specified age in days.

Description

Deletes job-related logs that were previously archived (into the WX_WRK_AUDIT_ARCHIVE and WX_WRK_ERROR_ARCHIVE tables via a RED Scheduler and/or RED callable routines, such as Ws_Job_Clear_Logs and Ws_Job_Clear_Logs_By_Date) depending on their age in days.

When the maximum age of the archived logs to retain is exceeded all the older logs are deleted. For example, if 90 days are retained then all the archived logs that are older than 90 days are deleted. If a maximum age of 0 days is specified then all the archived logs are deleted. Alternatively, the **TRUNCATE** option can be used to remove all the archived logs, which overrides all other criteria.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Maximum Days to Retain	The maximum age (in days) of the archived logs to retain. If 90 days are retained then all the archived logs that are older than 90 days are **purged/deleted**. If 0 days are retained then all the archived logs are **purged/deleted**.
Job Name to Purge	The name of the job whose **archived** logs are to be **purged**. Wild cards are supported. Specifying % will match ALL jobs.
Options	The **TRUNCATE** option can be used to remove ALL the archived logs, which overrides all other criteria i.e. irrespective of the days to retain or job name.

Output

Output	Description
Return Code	Output Return Code: • S Success. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • -2 Error. e.g. Due to invalid job name or job not running. • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: Ws_Job_Clear_Archive

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_day_count	INTEGER	IN
p_job	VARCHAR(64)	IN
p_options	VARCHAR(256)	IN

Parameter Name	Datatype	Mode
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Ws_Job_Clear_Logs

Synopsis

Archives job logs when the maximum number of current logs to retain is exceeded.

Description

Moves job-related logs from the current log tables (such as WS_WRK_AUDIT_LOG and WS_WRK_ERROR_LOG) to the corresponding archive log tables (such as WX_WRK_AUDIT_ARCHIVE and WX_WRK_ERROR_ARCHIVE) depending on the number of logs to retain.

When the maximum number of current logs to retain is exceeded the oldest logs are archived for the specified job(s) to reduce the number of current logs to the specified retention limit. For example, if 10 is specified then only the latest 10 logs are retained. If a retained count of 0 is specified then all the current logs are archived for the specified job(s).

Note

Equivalent functionality is available via a WhereScape RED Scheduler and the "Logs Retained" property of a job.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Job Name(s) to Archive	The name of the job(s) whose current logs are to be archived. Wild cards are supported. Specifying % will match ALL jobs.
Maximum Logs to Retain	The maximum number of logs to retain. When the maximum is exceeded, the oldest logs are **archived** to reduce the number of current logs to the specified retention limit.

Output

Output	Description
Return Code	Output Return Code: • S Success. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • -2 Error. e.g. Due to invalid job name or job not running. • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: Ws_Job_Clear_Logs

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_job_to_clean	VARCHAR(64)	IN
p_keep_count	INTEGER	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Ws_Job_Clear_Logs_By_Date

Synopsis

Archives job logs that are older than the specified age in days.

Description

Moves job-related logs from the current log tables (such as WS_WRK_AUDIT_LOG and WS_WRK_ERROR_LOG) to the corresponding archive log tables (such as WX_WRK_AUDIT_ARCHIVE and WX_WRK_ERROR_ARCHIVE) depending on their age in days.

When the maximum age of the current logs to retain is exceeded all the older logs are archived for the specified job(s). For example, if 90 days are retained then all the current logs that are older than 90 days are archived.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Job Name(s) to Archive	The name of the job(s) whose current logs are to be **archived**. Wild cards are supported. Specifying % will match ALL jobs.
Maximum Days to Retain	The maximum age (in days) of the current logs to retain. If 90 days are retained, then all the current logs that are older than 90 days are archived.

Output

Output	Description
Return Code	Output Return Code: • S Success. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • -2 Error. e.g. Due to invalid job name or job not running. • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: Ws_Job_Clear_Logs_By_Date

Callable Routine Type:FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_job_to_clean	VARCHAR(64)	IN
p_day_count	INTEGER	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Ws_Job_Create

Synopsis

Creates a job based on an existing job and optionally starts it immediately.

Description

Creates a job from the specified existing job, if it is in either a holding or waiting state. The new job can be started immediately. Typically, this routine is used to create & start a job from within another job. Only jobs that are in a holding or waiting state can be used as a template for the new job.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Template Job Name	The name of the job to be used as a template for the new job. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler. The specified job must be in a holding or waiting state in order to be used as a template.
New Job Name	The name of the job to be created. The new job name cannot already exist.
Job Description	A description of the new job. When not specified the setting of the Template job is copied.
New Job Status	The initial status/frequency of the new job: • **HOLD** - The status of the new job will show as 'On Hold'. The job will not run until it is subsequently released. • **ONCE** - The new job will start immediately and upon successful completion the new job will be deleted. • **ONCE+HOLD -** The new job will start immediately and upon successful completion the status will show as 'On Hold'.
Thread Count	The number of threads for the new job. When not specified the setting of the Template job is copied.
Scheduler Preference	A scheduler type or a specific scheduler name that is allowed to run the job. When not specified, the setting of the Template job is copied. **Note** Some jobs/tasks can only run in a specific environment, such as Windows or UNIX/Linux.
Maximum Logs to Retain	The maximum number of logs to retain. When not specified the setting of the Template job is copied.
Success Command	A command-line action to execute upon successful completion of the new job. When not

Input	Description
	specified, the setting of the Template job is copied. The command must be executable within the context of the scheduler that runs the job—it must be a valid Windows/UNIX/Linux command that is appropriate to the scheduler environment.
Failure Command	A command-line action to execute upon failure of the new job. When not specified the setting of the Template job is copied. The command must be executable within the context of the scheduler that runs the job—it must be a valid Windows/UNIX/Linux command that is appropriate to the scheduler environment.

Output

Output	Description
Return Code	Output Return Code: • S Success. • **N No action because the Template Job is not in a holding or waiting state.** • **P No action because the New Job name already exists.** • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • **-1 Template Job is not in a holding/waiting state or the New Job name already exists.** • -2 Error. e.g. Due to invalid job name or job not running. • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: Ws_Job_Create

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_template_job	VARCHAR(64)	IN
p_new_job	VARCHAR(64)	IN
p_description	VARCHAR(256)	IN
p_state	VARCHAR(64)	IN
p_threads	INTEGER	IN
p_scheduler	VARCHAR(64)	IN
p_logs	INTEGER	IN
p_okay	VARCHAR(256)	IN
p_fail	VARCHAR(256)	IN

Parameter Name	Datatype	Mode
p_att1	VARCHAR(64)	IN
p_att2	VARCHAR(64)	IN
p_att3	VARCHAR(64)	IN
p_att4	VARCHAR(64)	IN
p_att5	VARCHAR(64)	IN
p_att6	VARCHAR(64)	IN
p_att7	VARCHAR(64)	IN
p_att8	VARCHAR(64)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Ws_Job_CreateWait

Synopsis

Creates a job based on an existing job and schedules it to start later.

Description

Creates a job from the specified existing job, if it is in either a holding or waiting state. The new job is scheduled to start later at the specified release time. Typically, this routine is used to create & schedule a job from within another job. Only jobs that are in a holding or waiting state can be used as a template for the new job.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Template Job Name	The name of the job to be used as a template for the new job. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler. The specified job must be in a holding or waiting state in order to be used as a template.
New Job Name	The name of the job to be created. The new job name cannot already exist.
Job Description	A description of the new job. When not specified the setting of the Template job is copied.
New Job Status	The initial status/frequency of the new job: • **HOLD** - The status of the new job will show as 'On Hold'. The job will not run until it is subsequently released. • **ONCE** - The new job will start immediately and upon successful completion the new job will be deleted. • **ONCE+HOLD** - The new job will start immediately and upon successful completion the status will show as 'On Hold'.
Scheduled Release Date/Time	The date/time when the new job is scheduled to be run.
Thread Count	The number of threads for the new job. When not specified, the setting of the Template job is copied.
Scheduler Preference	A scheduler type or a specific scheduler name that is allowed to run the job. When not specified, the setting of the Template job is copied. **Note** Some jobs/tasks can only run in a specific environment, such as Windows or UNIX/Linux.

Input	Description
Maximum Logs to Retain	The maximum number of logs to retain. When not specified, the setting of the Template job is copied.
Success Command	A command-line action to execute upon successful completion of the new job. When not specified, the setting of the Template job is copied. The command must be executable within the context of the scheduler that runs the job—it must be a valid Windows/UNIX/Linux command that is appropriate to the scheduler environment.
Failure Command	A command-line action to execute upon failure of the new job. When not specified the setting of the Template job is copied. The command must be executable within the context of the scheduler that runs the job—it must be a valid Windows/UNIX/Linux command that is appropriate to the scheduler environment.

Output

Output	Description
Return Code	Output Return Code: • S Success. • **N No action because the Template Job is not in a holding or waiting state.** • **P No action because the New Job name already exists.** • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • **-1 Template Job is not in a holding/waiting state or the New Job name already exists.** • -2 Error. e.g. Due to invalid job name or job not running. • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: Ws_Job_CreateWait

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_template_job	VARCHAR(64)	IN
p_new_job	VARCHAR(64)	IN
p_description	VARCHAR(256)	IN
p_state	VARCHAR(64)	IN
p_release_time	DATETIME	IN
p_threads	INTEGER	IN

Parameter Name	Datatype	Mode
p_scheduler	VARCHAR(64)	IN
p_logs	INTEGER	IN
p_okay	VARCHAR(256)	IN
p_fail	VARCHAR(256)	IN
p_att1	VARCHAR(64)	IN
p_att2	VARCHAR(64)	IN
p_att3	VARCHAR(64)	IN
p_att4	VARCHAR(64)	IN
p_att5	VARCHAR(64)	IN
p_att6	VARCHAR(64)	IN
p_att7	VARCHAR(64)	IN
p_att8	VARCHAR(64)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Ws_Job_Dependency

Synopsis

Adds or removes a child-to-parent dependency between two jobs to control the child job.

Description

Adds or removes a child-to-parent dependency between two jobs to control the child job. The dependent child job can be defined to fail (if necessary) when the parent job does not complete successfully in the required timeframe. The acceptable timeframe can be defined in terms of the maximum minutes in the past, to look back and the maximum minutes in the future to wait for successful completion of the parent job.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Action	Either '**ADD**' or '**DELETE**' the job dependency.
Parent Job Name	The name of Parent Job that the Child Job will depend on.
Child Job Name	The name of the Child Job that will be dependent on the Parent Job.
Parent Success Required	Indicates whether or not the Child Job will fail when the Parent Job does not complete successfully in the required time frame.
Maximum Look Back Minutes	The Maximum minutes in the past to look back for successful completion of the Parent Job.
Maximum Wait Minutes	The Maximum minutes in the future to wait for successful completion of the Parent Job.

Output

Output	Description
Return Code	Output Return Code: • S Success. • **W Warning. Dependency already exists ('ADD' action) or does not exist ('DELETE' action).**

Output	Description
	- E Error. - F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: - 1 Success. - **-1 Warning. Dependency already exists ('ADD' action) or does not exist ('DELETE' action).** - -2 Error. e.g. Due to invalid job name or job not running. - -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: Ws_Job_Dependency

Callable Routine Type:FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_action	VARCHAR(64)	IN
p_parent	VARCHAR(64)	IN
p_child	VARCHAR(64)	IN
p_required	VARCHAR(1)	IN
p_look_back	INTEGER	IN
p_max_wait	INTEGER	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Ws_Job_Release

Synopsis

Starts a job if it is in a holding or waiting state.

Description

Releases the specified job, if it is in a holding or waiting state, which sets the start time to the current time so that it starts immediately. Typically, this routine is used to start a job from within another job or via a third-party scheduler (rather than a WhereScape RED Scheduler).

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.

Input	Description
Release Job Name	The name of the job to be started/released. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler. The specified job must be in a holding or waiting state in order to be released.

Output

Output	Description
Return Code	Output Return Code: • S Success. • **N No action because the Template Job is not in a holding or waiting state**. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • **-1 No action because the job is not in a holding or waiting state**. • -2 Error. • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: Ws_Job_Release

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_release_job	VARCHAR(64)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Ws_Job_Restart

Synopsis

Starts a job if it is in a failed state.

Description

Releases the specified job if it is in a failed state, which sets the start time to the current time so that it starts immediately. This routine can be executed as part of a database start-up sequence to restart each failed job that may have stopped, due to an earlier database shutdown.

Input

Input	Description

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Restart Job Name	The name of the job to be restarted/released. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler. The specified job must be in a failed state in order to be restarted.

Output

Output	Description
Return Code	Output Return Code: • S Success. • **N No action because the job is not in a failed state.** • **R No action because the job is currently running.** • **U No action because the job is in an unusual state due to an error (result number -2). The job is classified as running but it is NOT actually running or failed so it cannot be restarted. This may occur if the scheduler has failed and the job is in a pending state.** • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • **-1 No action because the job is not in a failed state or it is currently running.** • -2 Error. • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: Ws_Job_Restart

Callable Routine Type:FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_restart_job	VARCHAR(64)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Ws_Job_Schedule

Synopsis

Schedules a job if it is in a holding or waiting state.

Description

Schedules the specified job if it is in a holding or waiting state, which will start at the specified time. Typically, this routine is used to schedule a job from within another job.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Schedule Job Name	The name of the job to be scheduled. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler. The specified job must be in a holding or waiting state in order to be scheduled.
Scheduled Release Time	The date/time that the job is to be scheduled to be released/started.

Output

Output	Description
Return Code	Output Return Code: • S Success. • **N No action because the job is not in a holding or waiting state.** • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • **-1 No action because the job is not in a holding or waiting state.** • -2 Error. • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: Ws_Job_Schedule

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_release_job	VARCHAR(64)	IN
p_release_time	DATETIME	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Ws_Job_Status

Synopsis

Returns the current status of a job.

Description

Returns the current status of the specified job, as recorded by a WhereScape RED Scheduler. Typically, this routine is used by a third-party scheduler or a user-defined procedure/script to check on a job.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Job Sequence	The unique integer identifier of the job to return the status of. This input is optional but when it is specified, the started **within** and started **after** inputs should not be specified.
Job Name	The name of the job to return the status of. The specified name must exactly match the job name as displayed by the WhereScape RED Scheduler.
Started Within Last Minutes	The maximum minutes [0-148599] (up to ~103.1 days) in the past to look back for the job to have started. This input is optional but when it is specified, the job name must be specified and the job sequence and started **after** inputs should not be specified. Note: If multiple instances of the job have started in the specified time frame then the last job to start is returned (i.e. the job with the highest sequence number).
Started After Time	The date/time after which to look for the job to have started. This input is optional but when it is specified, the job name must be specified and the job sequence and started **within** inputs should not be specified. **Note** If multiple instances of the job have started in the specified time frame then the last job to start is returned (i.e. the job with the highest sequence number).

Output

Output	Description
Return Code	Output Return Code: • S - **Success**. • N - **The job exists but it was NOT started within the specified time frame.** • E - **Error**. • F - **Fatal/Unexpected Error**.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 - **Success**. • -1 - **The job exists but it was NOT started within the specified time frame.** • -2 - **Error**. • -3 **-Fatal/Unexpected Error**. • 0 - see note below.
Simplified Job Status Code	Simplified Job Status Code: • N - **Not Running**. • R - **Running**. • F - **Failed**. • C - **Completed**. • 0 - see note below.
Standard Job Status Code	Standard Job Status Code: • H - **On Hold**. The job is on hold. A held job can be edited and/or started.

Output	Description
	• W - **Waiting**. The job is waiting to start (it is either waiting for the scheduled time to arrive or is waiting for an available scheduler). • B - **Blocked**. The job is blocked because a previous instance of the same job is still running. • P - **Pending**. This is the initial interim status of an "about to start running" job. The scheduler has identified that the job is ready to start and is preparing to run it. A job should only be pending for a brief period so if it remains pending for a prolonged period then an unexpected error has occurred. • R - **Running**. The job is currently running. • F - **Failed**. The job failed due to an error. • C - **Completed**. The job completed successfully (but it may have warnings). A completed job cannot be restarted. • G - **Failed - Aborted**. The job failed and was subsequently aborted. An aborted job cannot be restarted. • E - **Error Completion**. • 0 - see note below.
Enhanced Job Status Number	Enhanced Job Status Number that returns an **integer** rather than the standard alphabetic code. The **running** and **completed statuses are enhanced** to distinguish errors or warnings. 1. **On Hold.** The job is on hold. A held job can be edited and/or started. 2. **Waiting**. The job is waiting to start (it is either waiting for the scheduled time to arrive or is waiting for an available scheduler). 3. **Blocked**. The job is blocked because a previous instance of the same job is still running. 4. **Pending**. This is the initial interim status of an "about to start running" job. The scheduler has identified that the job is ready to start and is preparing to run it. A job should only be pending for a brief period so if it remains pending for a prolonged period then an unexpected error has occurred. 5. **Running**. The job is currently running and no tasks have failed or produced warnings. 6. **Running with Errors**. The job is currently running but some tasks have failed. The job will ultimately fail when all the tasks that are NOT dependent on the failed tasks have finished. 7. **Running with Warnings**. The job is currently running and some tasks have produced warnings. 8. **Failed**. The job failed due to an error. 9. **Completed**. The job completed without warnings. A completed job cannot be restarted. 10. **Completed with Warnings**. The job completed with warnings. A completed job cannot be restarted. 11. **Failed - Aborted**. The job failed and it was subsequently aborted. An aborted job cannot be restarted. 12. **Error Completion**. 0 - see note below.

Notes

All three returned status values can also return '**0**' in any of the following situations:

- Illegal combination of parameters specified.
- Unable to locate specified job sequence.
- Unable to locate specified job name.
- Job Not Found having started in the last **SpecifiedMinutes** minutes.

Notes

- Job Not Found having started after **SpecifiedDateTime**.

PostgreSQL

PostgreSQL Parameters: Ws_Job_Status

Callable Routine Type:FUNCTION

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_check_sequence	INTEGER	IN
p_check_job	VARCHAR(64)	IN
p_started_in_last_mi	INTEGER	IN
p_started_after_dt	DATETIME	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT
p_job_status_simple	VARCHAR(1)	OUT
p_job_status_standard	VARCHAR(1)	OUT
p_job_status_enhanced	VARCHAR(2)	OUT

Ws_Load_Change

Synopsis

Changes the Connection or Schema of a load table.

Description

Changes either the Connection or the Schema of the specified load table. Only the Connection or Schema can be changed so two calls are required to change both.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.

Input	Description
Change Property	Change either the '**SCHEMA**' or the '**CONNECTION**' of the specified load table. Separate calls must be made if both the schema and connection need to be changed.
Load Table Name	The name of the load table to be changed.
New Property Value	Either the new schema name or the new connection name.

Output

Output	Description
Return Code	Output Return Code: • S Success. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • -2 Error. • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQLr Parameters: Ws_Load_Change

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_action	VARCHAR(64)	IN
p_table	VARCHAR(64)	IN
p_new_value	VARCHAR(64)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

Ws_Maintain_Indexes

Synopsis

Drops and/or builds database indexes that are defined in the WhereScape RED metadata.

Description

Drops and/or builds indexes for a specified table or a specified index. Only indexes that are defined in the WhereScape RED metadata are supported. Typically, this routine is used by a WhereScape RED Scheduler and RED-generated procedures to automatically maintain indexes. However, it is also valid for user-defined custom procedures/scripts to execute this routine to control when indexes are dropped and/or created.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Table Name	Table Name to process the relevant indexes of. **Notes** The Table Name is ignored when the optional Index Name is specified.
Index Name	Optional Index Name to only process the specified index. When NOT specified all the relevant indexes of the table are processed. **Notes** • The Table Name is ignored when the Index Name is specified. • Must be specified to use the 'DROP' or 'BUILD' index actions.
Index Action	Action that specifies whether indexes are dropped or built and what types of indexes are applicable: • **DROP** - Drops the specified index (Index Name must be specified). • **DROP ALL** - Drops ALL the indexes of the table. • **PRE DROP** - Drops the indexes of the table that are defined as pre-drop. • **BUILD** - Builds the specified index (Index Name must be specified). Otherwise, build all the indexes of the table that were pre-dropped. • **BUILD ALL** - Builds ALL the indexes of the table.

Output

Output	Description
Result Number	Output Result Number: • 1 Success. • **-1 Warning.** • -2 Error. • -3 Fatal/Unexpected Error.

Note

Ws_Maintain_Indexes does NOT include a Return Code or Return Message like most of the WhereScape RED Callable routines, but it does output a Result Number.

PostgreSQL

PostgreSQL Parameters: Ws_Maintain_Indexes

Callable Routine Type: FUNCTION

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN

Parameter Name	Datatype	Mode
p_task_id	INTEGER	IN
p_table_name	VARCHAR(64)	IN
p_parameter	VARCHAR(64)	IN
p_index_name	VARCHAR(64)	IN
p_option	VARCHAR(64)	IN
p_result	INTEGER	OUT

Ws_Version_Clear

Synopsis

Purges metadata versions for all objects that do not meet the specified retention criteria.

Description

Deletes metadata versions for all objects that do not meet the specified retention criteria, which can be specified as the minimum number of versions to retain per object and/or the maximum age (in days) of versions to retain. For example, it is possible to specify that a minimum of 5 versions are retained for each object and/or that versions are retained for a maximum of 90 days.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Maximum Days to Retain	The maximum age (in days) of the versions to retain. If 90 days are retained, then all the versions that are older than 90 days are **purged/deleted** to reduce the number of versions to the specified maximum number of versions per object. If not specified, then the retention date of each version determines whether it is deleted.
Minimum Versions per Object to Retain	The minimum number of versions to retain for each object. If 5 is specified, then the last 5 versions are retained per object **regardless** of the specified maximum age to retain.
Options	Currently NOT used.

Output

Output	Description
Return Code	Output Return Code: • S Success. • E Error. • F Fatal/Unexpected Error.
Return Message	Output message indicating the action applied or the reason for no action.
Result Number	Output Result Number: • 1 Success. • -2 Error. • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: Ws_Version_Clear

Callable Routine Type:FUNCTION.

Parameter Name	Datatype	Mode
p_sequence	INTEGER	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
p_day_count	INTEGER	IN
p_keep_count	INTEGER	IN
p_options	VARCHAR(256)	IN
p_return_code	VARCHAR(1)	OUT
p_return_msg	VARCHAR(256)	OUT
p_result	INTEGER	OUT

WsParameterRead

Synopsis

Returns the value and comment (for most RDBMS) of a WhereScape RED metadata Parameter.

Description

Returns the value and comment (for most RDBMS) of the specified parameter from the DSS_PARAMETER metadata table. For SQL Server, Teradata, and DB2 this routine is a PROCEDURE that returns both the parameter value and comment.

Typically, this routine is used by procedures to read information that is written by another process (automatically or manually via the RED **Tools > Parameters** menu item), which is external to the procedure.

Input

Input	Description
Parameter Name	The case-sensitive name of the WhereScape RED metadata parameter to be retrieved. The name must exactly match an existing parameter, otherwise a NULL value is returned.

Output

Output	Description
Parameter Value	The retrieved value of the parameter. Corresponds to the "Value" property that is visible and maintainable via **Tools > Parameters**.
Parameter Comments	The maximum number of versions to retain for each object. If 5 is specified then the last 5 versions are retained per object **regardless** of the specified maximum age to retain.

PostgreSQL

PostgreSQL Parameters: WsParameterRead

Callable Routine Type:FUNCTION.

Parameter Name	Datatype	Mode
p_parameter	VARCHAR(64)	IN
p_value	VARCHAR(2000)	OUT
p_comment	VARCHAR(256)	OUT

Note

There is also a corresponding WsParameterReadF FUNCTION.

WsParameterReadF

Synopsis

Returns the value of a WhereScape RED metadata Parameter.

Description

Returns the value of the specified parameter from the DSS_PARAMETER metadata table.

Typically, this routine is used by procedures to read information that is written by another process (automatically or manually via the RED **Tools > Parameters** menu item), which is external to the procedure.

Input

Input	Description
Parameter Name	The name of the WhereScape RED metadata parameter to be retrieved. The case-sensitive name must exactly match an existing parameter, otherwise a NULL value is returned.

Output

Output	Description
Parameter Value	The value of the parameter. Corresponds to the **Value** property that is visible and maintainable via **Tools > Parameters**.

PostgreSQL

PostgreSQL Parameters: WsParameterReadF

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
@p_parameter	VARCHAR(64)	IN
FUNCTION Return Value	VARCHAR(4000)	OUT-Function

Note

There is also a corresponding WsParameterRead PROCEDURE.

WsParameterReadG

Synopsis

Returns the value of a "global" WhereScape RED metadata Parameter that relates to a load table.

Description

Returns the value of an internal parameter that is defined and populated by WhereScape RED, which is available to a procedure that is currently processing a load table.

The supported parameters are $$TABLE_NAME and $$SOURCE_TABLE.

Input

Input	Description
Global Parameter Name	The supported global parameters are: • **$$TABLE_NAME** returns the Load Table Name that the procedure is executing against. Only available for load tables. • **$$SOURCE_TABLE** returns the maximum value of the Source Table property from

Input	Description
	the columns of the Load Table that the procedure is executing against. Only available for load tables. **Note** Typically, a Load Table has a single Source Table but if it has multiple sources then the maximum (alphabetically) Source Table Name will be returned.
Job Identifier	Unique identifier of the **held or scheduled job** that the running job is a specific instance of. When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. When invoked manually or externally to the WhereScape RED Scheduler, it is recommended to use 0 (zero).
Task or Object Identifier	Unique identifier of the **running task** (of a running job) that executed the routine. When invoked from a WhereScape RED Scheduler the routine will be passed the parameter argument. When invoked manually or externally to the WhereScape RED Scheduler, it should be the **object key**.

Output

Output	Description
Result Table Name	The requested Load Table Name or Source Table Name.

PostgreSQL

PostgreSQL Parameters: WsParameterReadG

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_parameter	VARCHAR(64)	IN
p_job_id	INTEGER	IN
p_task_id	INTEGER	IN
FUNCTION Return Value	VARCHAR(4000)	OUT-Function

WsParameterWrite

Synopsis

Updates the value and comment of a WhereScape RED metadata Parameter or creates it.

Description

Updates the value and comment of the specified parameter in the DSS_PARAMETER metadata table. If the specified parameter is not found, then it is added.

Typically, this routine is used by procedures to write information that is read by another process (automatically or manually via the RED **Tools > Parameters** menu item), which is external to the procedure.

Input

Input	Description
Parameter Name	The case-sensitive name of the WhereScape RED metadata parameter to be updated or added.
Parameter Value	The new value of the parameter to be assigned. Corresponds to the "Value" property that

Input	Description
	is visible and maintainable via **Tools > Parameters**.
Parameter Comments	The new comments of the parameter to be assigned. Corresponds to the "Comments" property that is visible and maintainable via **Tools > Parameters**. The parameter comments will not be modified if a NULL value is specified.

Output

Output	Description
Result Number	Output Result Number : • **1Metadata Parameter Updated.** • **2Metadata Parameter Inserted.** • -3Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: WsParameterWrite

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_parameter	VARCHAR(64)	IN
p_value	VARCHAR(2000)	IN
p_comment	VARCHAR(256)	IN

WsWrkAudit

Synopsis

Records a message in the Audit Log.

Description

Adds the specified message to the WS_WRK_AUDIT_LOG workflow metadata table, which is referred to as the Audit Log or Audit Trail. A variety of message types are supported such as Information, Warning, and Error that are included in the corresponding message type counts for the task and job. Audit Log messages are accessible via the "Scheduler" tab/window and/or the WS_WRK_AUDIT_LOG table.

Note

Both the Audit Log and Error/Detail Log support similar information and in user-defined custom procedures, either or both logs can be used. However in RED-generated procedures/scripts, the Audit Log is used for higher-level or summary messages while the Error/Detail Log is used for more detailed supporting information.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Audit Message Type Code	Audit Message Type Code: • B - Beginning of a Job or Task. • I - Information. • S - Success. • W - Warning. • E - Error. • F - Fatal Error.

Input	Description
Audit Message Text	Custom message text to be recorded in the WhereScape RED Audit Log.
RDBMS Code	RDBMS-specific message code.
RDBMS Message	RDBMS-specific message.

Output

Output	Description
Result Number	Output Result Number: • 1 Success. • -3 Error.

PostgreSQL

PostgreSQL Parameters: WsWrkAudit

Callable Routine Type:FUNCTION.

Parameter Name	Datatype	Mode
p_status_code	VARCHAR(1)	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_sequence	INTEGER	IN
p_message	VARCHAR(256)	IN
p_db_code	VARCHAR(10)	IN
p_db_msg	VARCHAR(256)	IN
p_task_key	INTEGER	IN
p_job_key	INTEGER	IN

WsWrkAuditBulk

Synopsis

Records multiple messages in the Audit Log.

Description

Adds the specified multiple messages to the WS_WRK_AUDIT_LOG workflow metadata table, which is referred to as the Audit Log or Audit Trail. A variety of message types are supported, such as Information, Warning, and Error that are included in the corresponding message type counts for the task and job. Audit Log messages are accessible via the "Scheduler" tab/window and/or the WS_WRK_AUDIT_LOG table.

Note

Both the Audit Log and Error/Detail Log support similar information and in user-defined custom procedures, either or both logs can be used. However in RED-generated procedures/scripts, the Audit Log is used for higher-level or summary messages while the Error/Detail Log is used for more detailed supporting information.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.

Input	Description
Audit Message Type Code	Audit Message Type Code: • B - Beginning of a Job or Task. • I - Information. • S - Success. • W - Warning. • E - Error. • F - Fatal Error.
Audit Message(s) Text	Custom message(s) text to be recorded in the WhereScape RED Audit Log. Multiple messages can be specified but each is limited to 256 characters. Each message must be separated by either a new-line (ASCII 10) or tilde (~) character. e.g. Message1~Message2~Message3 will create 3 messages.
RDBMS Code	RDBMS-specific message code. It is optional but recommended to populate this when an error occurs.
RDBMS Message	RDBMS-specific message. It is optional but recommended to populate this when an error occurs.

Output

Output	Description
Result Number	Output Result Number: • 1 Success. • -2 Error • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: WsWrkAuditBulk

Callable Routine Type:FUNCTION

Parameter Name	Datatype	Mode
p_status_code	VARCHAR(1)	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_sequence	INTEGER	IN
p_message	VARCHAR(4000)	IN
p_db_code	VARCHAR(10)	IN
p_db_msg	VARCHAR(256)	IN
p_task_key	INTEGER	IN
p_job_key	INTEGER	IN

WsWrkError

Synopsis

Records a message in the Error/Detail Log.

Description

Adds the specified message to the WS_WRK_ERROR_LOG workflow metadata table, which is referred to as the Error Log or Detail Log. A variety of message types are supported such as Information, Warning, and Error that are included in the "detail" message counts for the task and job (viewable via the "Scheduler" tab/window). Error/Detail Log messages are accessible via the "Scheduler" tab/window and/or the WS_WRK_ERROR_LOG table.

Note

Both the Audit Log and Error/Detail Log support similar information and in user-defined custom procedures, either or both logs can be used. However in RED-generated procedures/scripts, the Audit Log is used for higher-level or summary messages while the Error/Detail Log is used for more detailed supporting information.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Error/Detail Message Type Code	Error/Detail Message Type Code: - E - Error. - I - Information. - W - Warning.
Error/Detail Message Text	Custom message text to be recorded in the WhereScape RED Error/Detail Log.
RDBMS Code	RDBMS-specific message code. It is optional but recommended to populate this when an error occurs.
RDBMS Message	RDBMS-specific message. It is optional but recommended to populate this when an error occurs.
Custom Message Type Code	Custom Message Type Code. For custom usage and has no meaning within the WhereScape RED metadata.

Output

Output	Description
Result Number	Output Result Number: - 1 Success. - -3 Error.

PostgreSQL

PostgreSQL Parameters: WsWrkError

Callable Routine Type: FUNCTION

Parameter Name	Datatype	Mode
p_status_code	VARCHAR(1)	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_sequence	INTEGER	IN
p_message	VARCHAR(256)	IN
p_db_code	VARCHAR(10)	IN
p_db_msg	VARCHAR(256)	IN
p_task_key	INTEGER	IN
p_job_key	INTEGER	IN

Parameter Name	Datatype	Mode
p_msg_type	VARCHAR(10)	IN

WsWrkErrorBulk

Synopsis

Records multiple messages in the Error/Detail Log.

Description

Adds the specified multiple messages to the WS_WRK_ERROR_LOG workflow metadata table, which is referred to as the Error Log or Detail Log. A variety of message types are supported such as Information, Warning, and Error that are included in the "detail" message counts for the task and job (viewable via the "Scheduler" tab/window). Error/Detail Log messages are accessible via the "Scheduler" tab/window and/or the WS_ADMIN_V_ERROR view of the WS_WRK_ERROR_LOG table.

Note

Both the Audit Log and Error/Detail Log support similar information and in user-defined custom procedures either or both logs can be used. However in RED-generated procedures/scripts, the Audit Log is used for higher-level or summary messages while the Error/Detail Log is used for more detailed supporting information.

Input

Input	Description
Common Input	Includes all 5 inputs of the **Callable Routines Common Input**.
Error/Detail Message Type Code	Error/Detail Message Type Code: • E - Error. • I - Information. • W - Warning.
Error/Detail Message(s) Text	Custom message(s) text to be recorded in the WhereScape RED Error/Detail Log. Multiple messages can be specified but each is limited to 256 characters. Each message must be separated by either a new-line (ASCII 10) or tilde (~) character. e.g. Message1~Message2~Message3 will create 3 messages.
RDBMS Code	RDBMS-specific message code. It is optional but recommended to populate this when an error occurs.
RDBMS Message	RDBMS-specific message. It is optional but recommended to populate this when an error occurs.
Custom Message Type Code	Custom Message Type Code. For custom usage and has no meaning within the WhereScape RED metadata.

Output

Output	Description
Result Number	Output Result Number: • 1 Success. • -2 Error. • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: WsWrkErrorBulk

Callable Routine Type:FUNCTION

Parameter Name	Datatype	Mode
p_status_code	VARCHAR(1)	IN
p_job_name	VARCHAR(64)	IN
p_task_name	VARCHAR(64)	IN
p_sequence	INTEGER	IN
p_message	VARCHAR(4000)	IN
p_db_code	VARCHAR(10)	IN
p_db_msg	VARCHAR(256)	IN
p_task_key	INTEGER	IN
p_job_key	INTEGER	IN
p_msg_type	VARCHAR(10)	IN

WsWrkTask

Synopsis

Updates row counts for a task in the Task Log.

Description

Updates row counts for the specified task in the Task Log. Task Log messages (and row counts) are accessible via the "Scheduler" tab/window and/or the WS_WRK_TASK_RUN and WS_WRK_TASK_LOG tables.

This routine is intended to be executed by a task of a job since it requires a valid job, task, and job sequence number that are provided by a WhereScape RED Scheduler.

Input

Input	Description
Common Input	Includes **3 inputs** of the **Callable Routines Common Input**.
Inserted Row Count	The number of rows inserted by the task.
Updated Row Count	The number of rows updated by the task.
Replaced Row Count	The number of rows replaced by the task.
Deleted Row Count	The number of rows deleted by the task.
Discarded Row Count	The number of rows discarded by the task.
Rejected Row Count	The number of rows rejected by the task.
Error Row Count	The number of rows with an error that were failed by the task.

Output

Output	Description
Result Number	Output Result Number: • **0 Success.** • **-1 Warning.** • -3 Fatal/Unexpected Error.

PostgreSQL

PostgreSQL Parameters: WsWrkTask

Callable Routine Type: FUNCTION.

Parameter Name	Datatype	Mode
p_job_key	INTEGER	IN
p_task_key	INTEGER	IN
p_sequence	INTEGER	IN
p_inserted	INTEGER	IN
p_updated	INTEGER	IN
p_replaced	INTEGER	IN
p_deleted	INTEGER	IN
p_discarded	INTEGER	IN
p_rejected	INTEGER	IN
p_errored	INTEGER	IN

Retrofitting

WhereScape RED includes an advanced retrofit capability that can be used to:

1. Migrate an existing data warehouse from one relational database to another (known as fork-lifting).
2. Load a data model from a modeling tool.

Retrofitting is achieved using the **Retro** object type in WhereScape RED and the **Retrofit tables** wizard.

For information on migrating an existing data warehouse, see Migrating the Data Warehouse Database Platform (**OLAP Retrofitting an OLAP Object**, **Migrating the Data Warehouse Database Platform**).

For information on importing a data model, refer to **Importing a Data Model**.

Migrating the Data Warehouse Database Platform

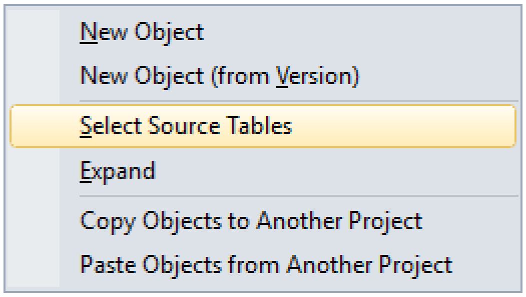
WhereScape RED has an advanced retrofitting wizard for migrating an existing data warehouse from one relational database to another.

The process to migrate an existing data warehouse includes:

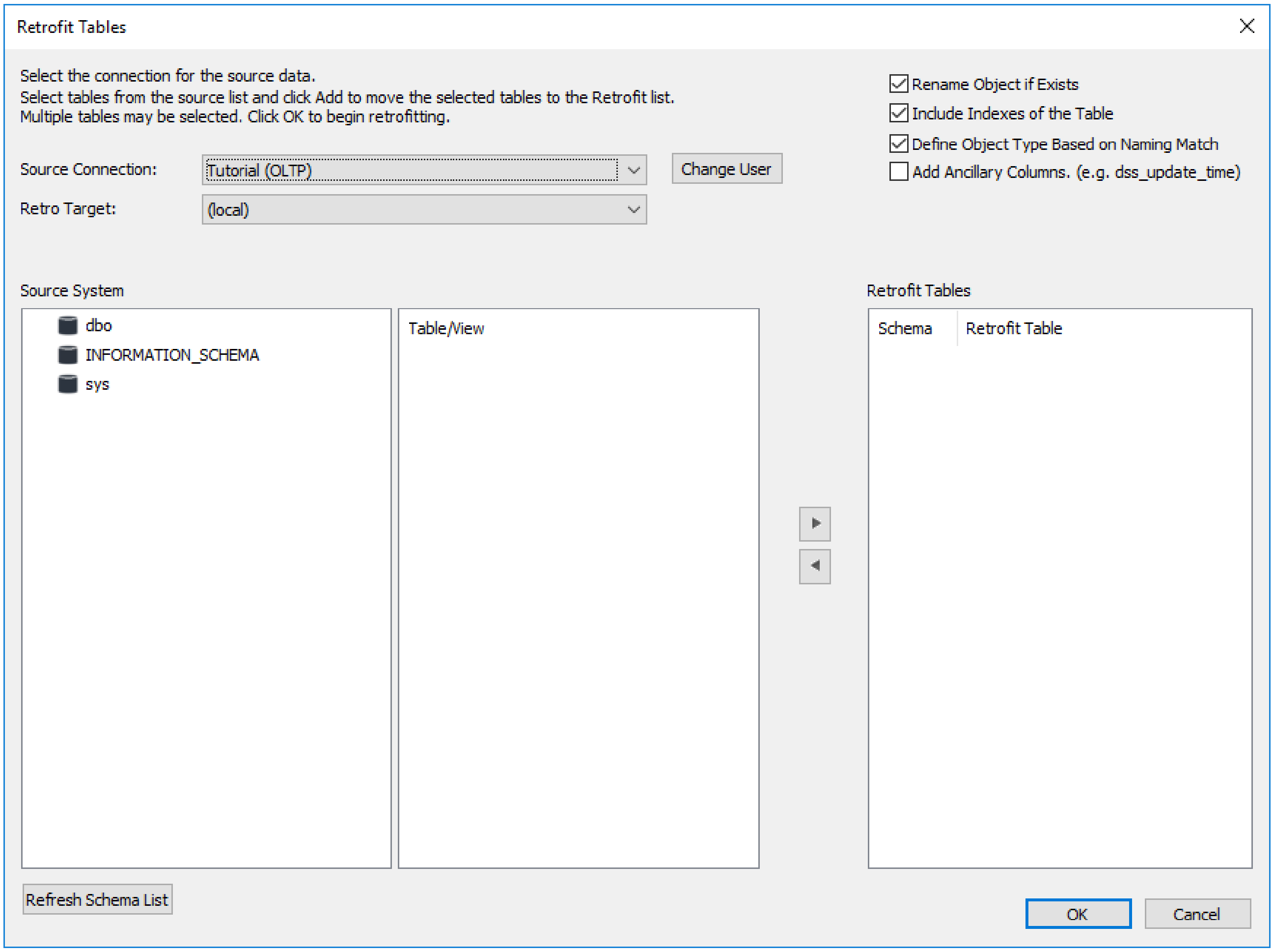
1. Creating a connection object to the existing warehouse database.
2. Creating Retro objects based on the source tables in the existing warehouse database.
3. Setting the Retro objects as Retro Copy type objects.
4. Running a Scheduler task to build the Retro Copy objects from the source tables.
5. Setting the Retro objects back to Retro (Retro Definition) type objects.
6. Converting the Retro objects to the Target Object types.

The steps to use this wizard are as follows:

1. Create a connection object for the old data warehouse database, populating the following fields:
   - Connection Name
   - Connection Type => **ODBC**
   - ODBC Source
   - Work Directory
   - Extract user name
   - Extract password
2. Ensure all naming standards in **Home> Options** are set to match the objects being retrofitted. This saves work later.
3. Ensure **Enable Retro** is selected in the **Home > Options > Object Types** menu.
4. Right-click the **Retro object group** in the object tree in the left pane and select **Select Source Tables**.


New Object
New Object (from Version)
Select Source Tables
Expand
Copy Objects to Another Project
Paste Objects from Another Project

5. The **Retrofit Tables** window appears. In the **Source Connection** drop-down list choose the connection set up in step 1. A list of **databases** appears in the left pane.



Retrofit Tables

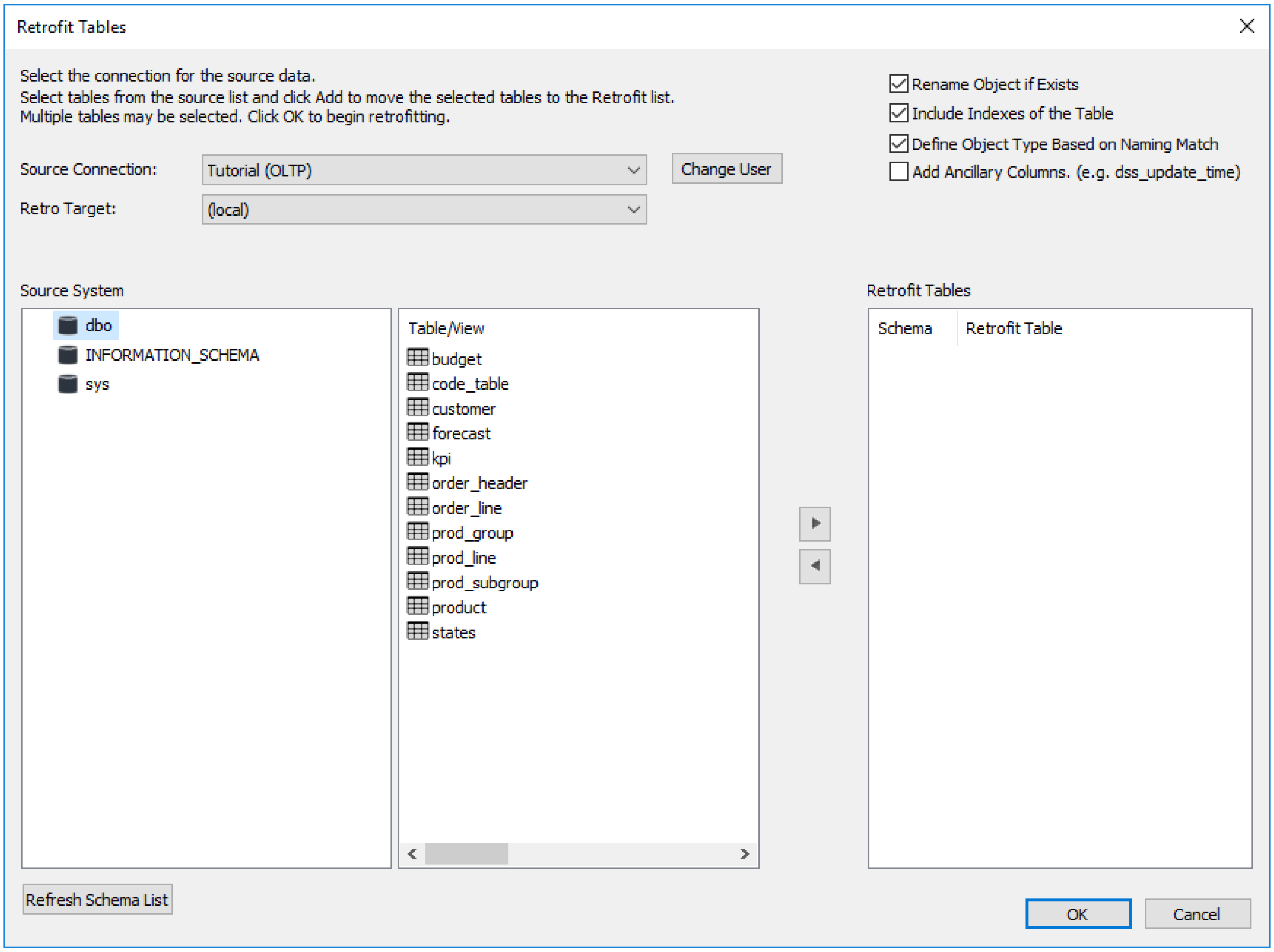
Select the connection for the source data.
Select tables from the source list and click Add to move the selected tables to the Retrofit list.
Multiple tables may be selected. Click OK to begin retrofitting.

Source Connection: Tutorial (OLTP)

Retro Target: (local)

☑ Rename Object if Exists
☑ Include Indexes of the Table
☑ Define Object Type Based on Naming Match
□ Add Ancillary Columns. (e.g. dss_update_time)

6. Double-click the database/user/schema in the left pane list. A list of **tables** in the database is displayed in the middle pane.



Retrofit Tables

Select the connection for the source data.
Select tables from the source list and click Add to move the selected tables to the Retrofit list.
Multiple tables may be selected. Click OK to begin retrofitting.

Source Connection: Tutorial (OLTP)

Retro Target: (local)

☑ Rename Object if Exists
☑ Include Indexes of the Table
☑ Define Object Type Based on Naming Match
☐ Add Ancillary Columns. (e.g. dss_update_time)

Source System

- dbo
- INFORMATION_SCHEMA
- sys

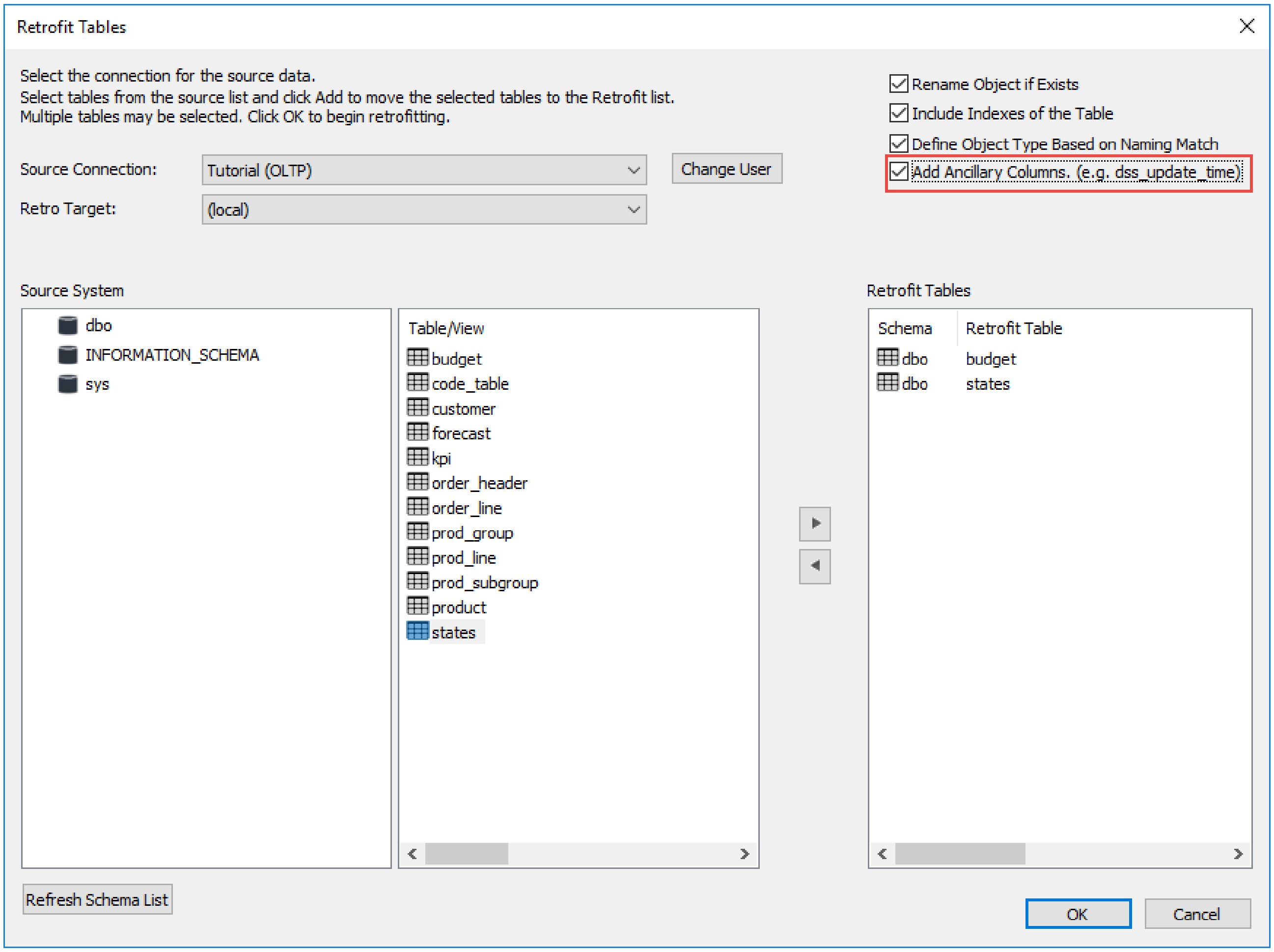
Table/View

- budget
- code_table
- customer
- forecast
- kpi
- order_header
- order_line
- prod_group
- prod_line
- prod_subgroup
- product
- states

Retrofit Tables

Schema	Retrofit Table

7. Select all the required tables from the middle pane list and click > to move them to the right pane. Then click the **Add Ancillary Columns (e.g. dss_update_time)** check-box and click **OK**.



Retrofit Tables

Select the connection for the source data.
Select tables from the source list and click Add to move the selected tables to the Retrofit list.
Multiple tables may be selected. Click OK to begin retrofitting.

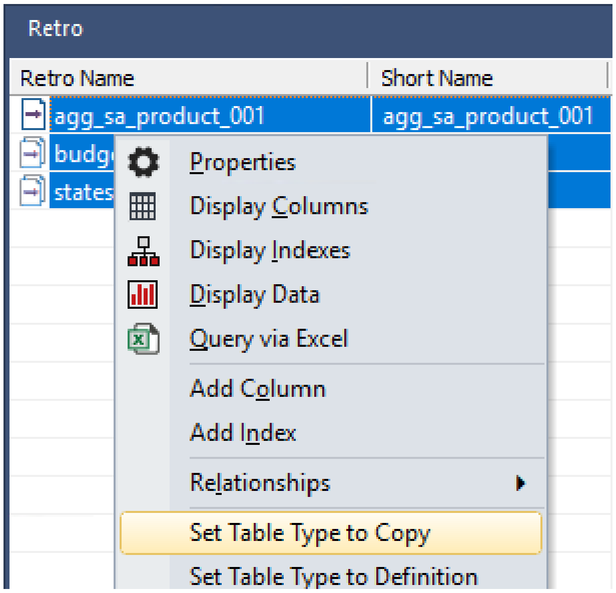
☑ Rename Object if Exists
☑ Include Indexes of the Table
☑ Define Object Type Based on Naming Match
☑ Add Ancillary Columns. (e.g. dss_update_time)

Source Connection: Tutorial (OLTP)

Retro Target: (local)

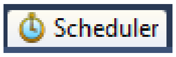
Schema	Retrofit Table
dbo	budget
dbo	states

8. WhereScape RED acquires the metadata for the tables being migrated and creates a new WhereScape RED Retro object for each.
9. Double-click the **Retro object group** in the left pane. Select all Retros in the middle pane. Right-click and select **Set Table Type to Copy**. This enables the data in the legacy data warehouse to be copied across to the new data warehouse.



Retro

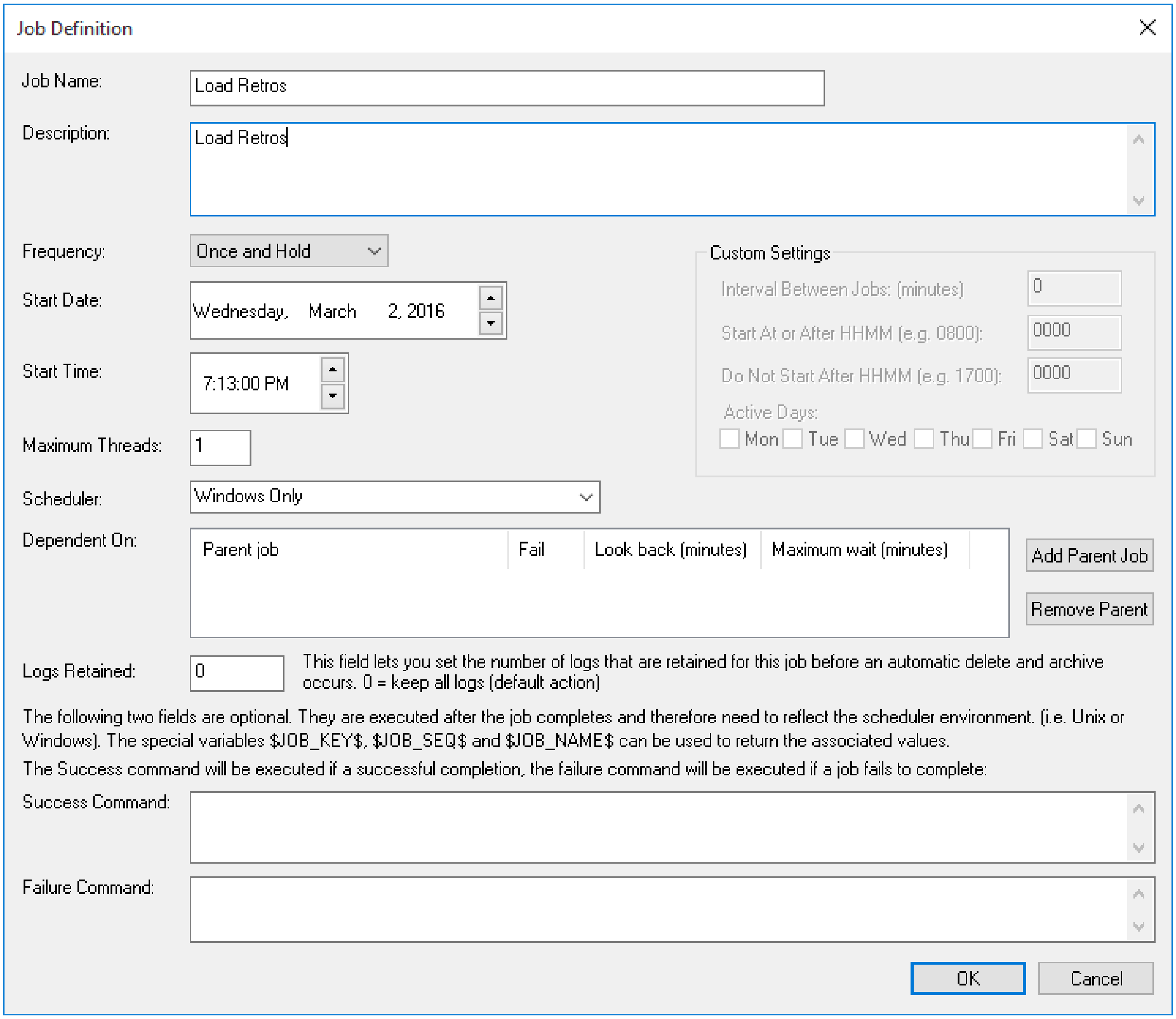
Retro Name	Short Name
agg_sa_product_001	agg_sa_product_001
budg	
states	

10. Click the **Scheduler** button from the toolbar. 

11. Create a new job to run straight away and click **OK**.

Note:

A scheduler must be running on the data warehouse connection for this job to complete, refer to the Scheduler Installation and Configuration section of the RED Installation Guide.



Job Definition

Job Name: Load Retros

Description: Load Retros

Frequency: Once and Hold

Start Date: Wednesday, March 2, 2016

Start Time: 7:13:00 PM

Maximum Threads: 1

Scheduler: Windows Only

Custom Settings

Interval Between Jobs: (minutes) 0

Start At or After HHMM (e.g. 0800): 0000

Do Not Start After HHMM (e.g. 1700): 0000

Active Days:
□ Mon □ Tue □ Wed □ Thu □ Fri □ Sat □ Sun

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)

Logs Retained: 0

This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

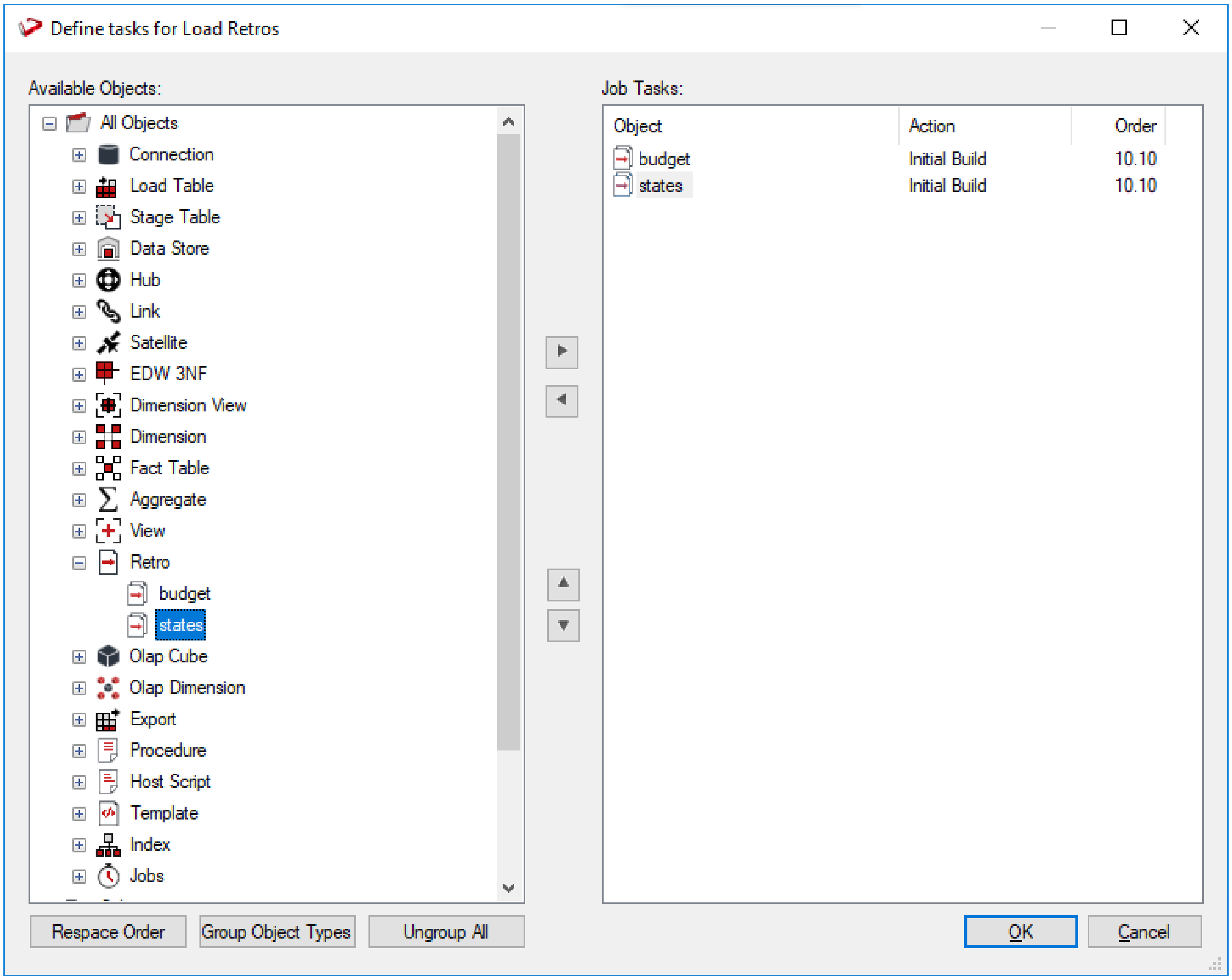
The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

Success Command:

Failure Command:

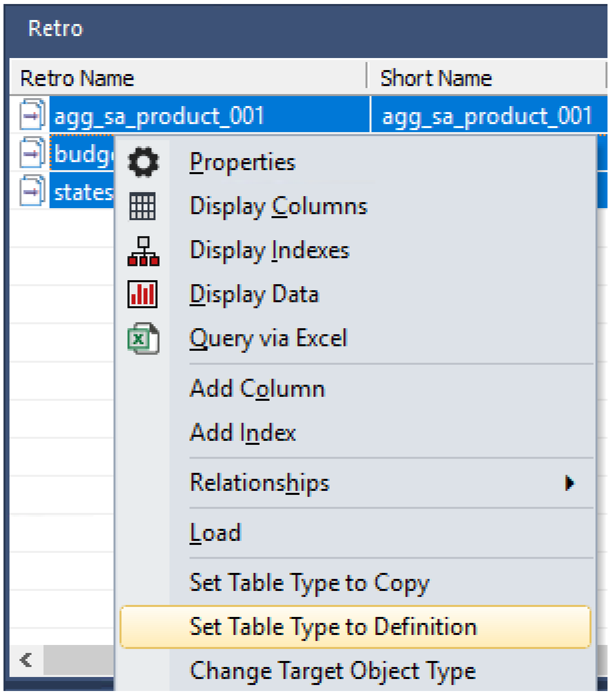
12. Add all Retro objects created in steps (3) to (9) and then click **Group Object Types**.



Job Tasks:

Object	Action	Order
budget	Initial Build	10.10
states	Initial Build	10.10

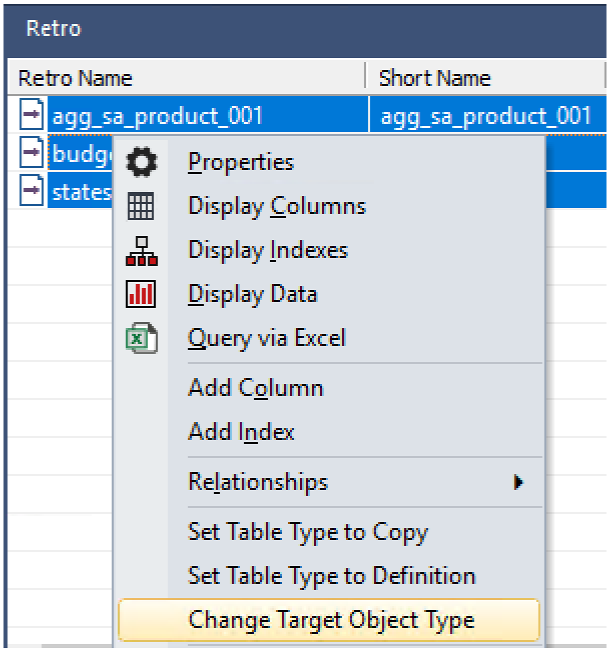
13. Once the job is completed, return to the WhereScape RED Builder window. Double-click the **Retro object group**. Select all objects in the middle pane and then from the right-click context menu select **Set Table Type to Definition**. This indicates the data has been copied into the Retro objects and the Retros can then be converted to the target objects.



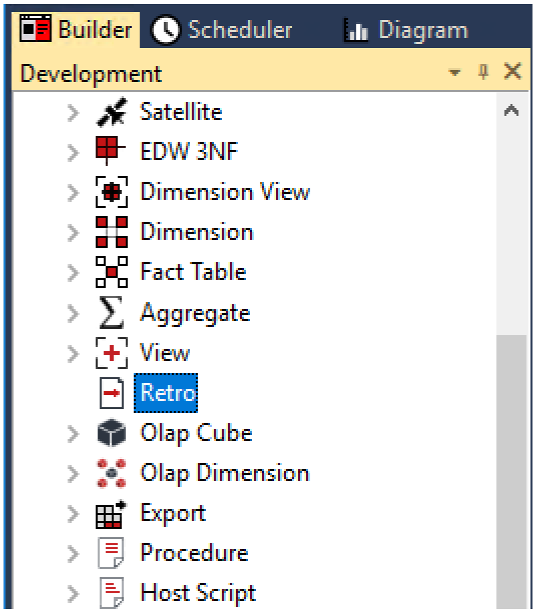
14. In the middle pane, select all objects. Right-click and select **Convert to Target Object**. WhereScape RED converts the **Retro** objects to the appropriate object types.

Note:

If the appropriate Target Object Type has not been set for one or more Retro objects; from the right click context menu select **Change Target Object Type** and select the correct Object Type.



15. There are no longer any Retro objects. They have been converted to Load, Stage, Dimension or Fact objects.



16. Change the source table and source column values on all of the retrofitted objects using either the Re-target source table window, or by editing column properties. Refer to **Re-Targeting Source Tables** for details.
17. Convert the old data warehouses code to WhereScape RED procedure in the new data warehouse database. Refer to **Integrate Procedures** for details.
18. If necessary, create new connections to be used with any migrated Lad tables. Attach a connection to each Load table. Refer to **Loading Data** for details.

Importing a Data Model

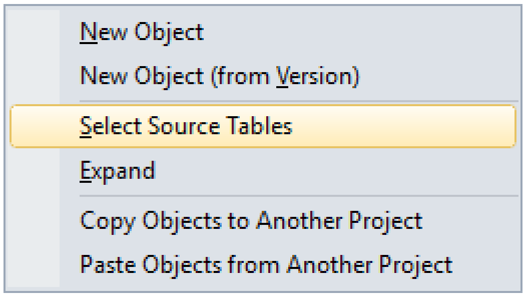
WhereScape RED provides functionality for importing data models from modeling tools.

The process to import a model is:

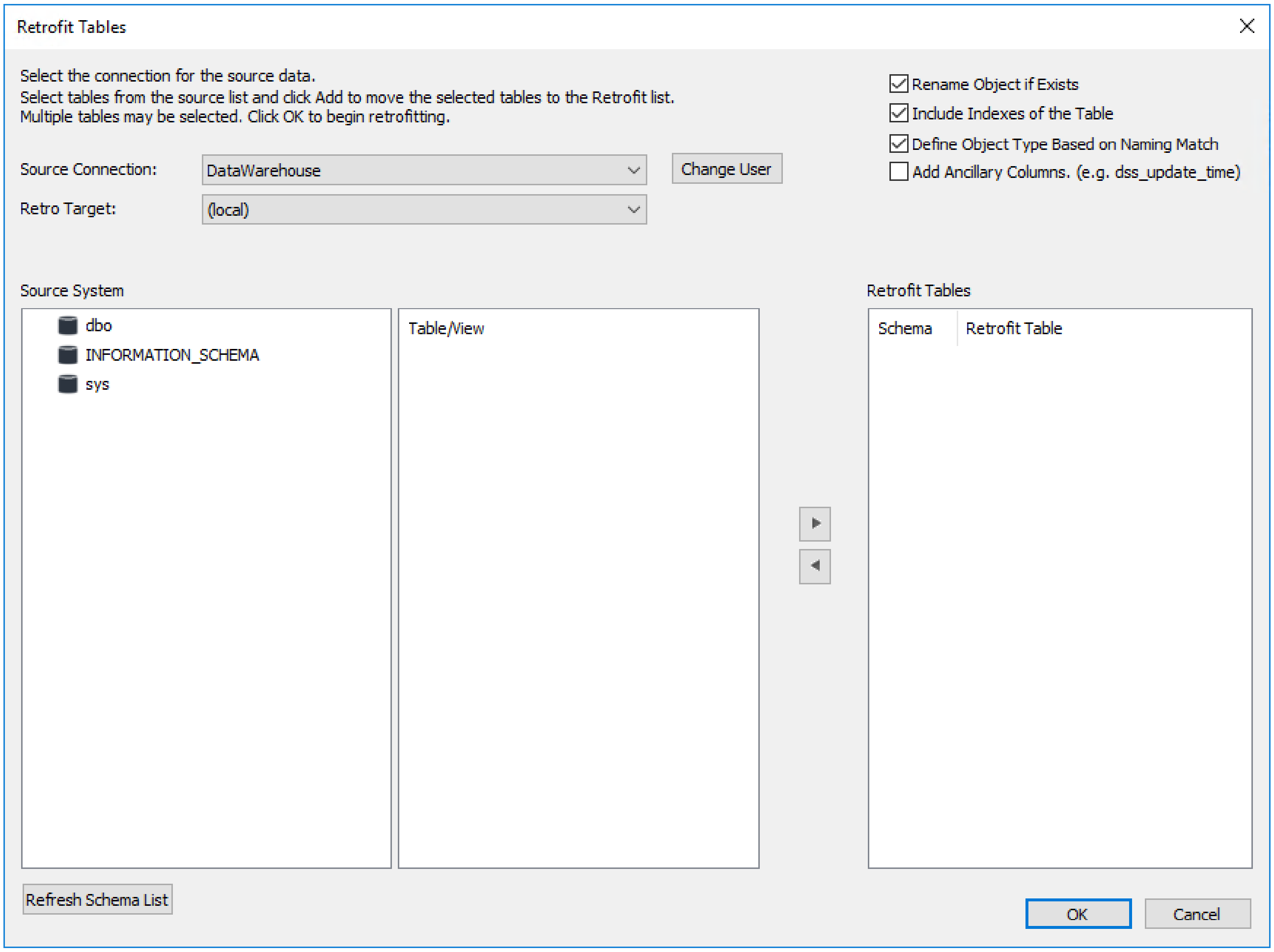
1. Create the physical data model in the modeling tool.
2. Generate DDL for the physical model in the modeling tool.
3. Run the DDL in the data warehouse database to create empty versions of the model tables.
4. Retrofit the tables in the dummy database into the WhereScape RED metadata as Retro objects.
5. Convert the Retro objects to Dimensions and Facts.

The following instructions outline steps 4 and 5 above:

1. Right-click the **Retro object group** in the object tree in the left pane and select **Select Source Tables**.


New Object
New Object (from Version)
Select Source Tables
Expand
Copy Objects to Another Project
Paste Objects from Another Project

2. The **Retrofit tables** dialog is displayed. In the **Source Connection** drop-down list choose the **DataWarehouse** connection. A list of **databases** appears in the left pane (your list will be different).



Retrofit Tables

Select the connection for the source data.
Select tables from the source list and click Add to move the selected tables to the Retrofit list.
Multiple tables may be selected. Click OK to begin retrofitting.

Source Connection: DataWarehouse

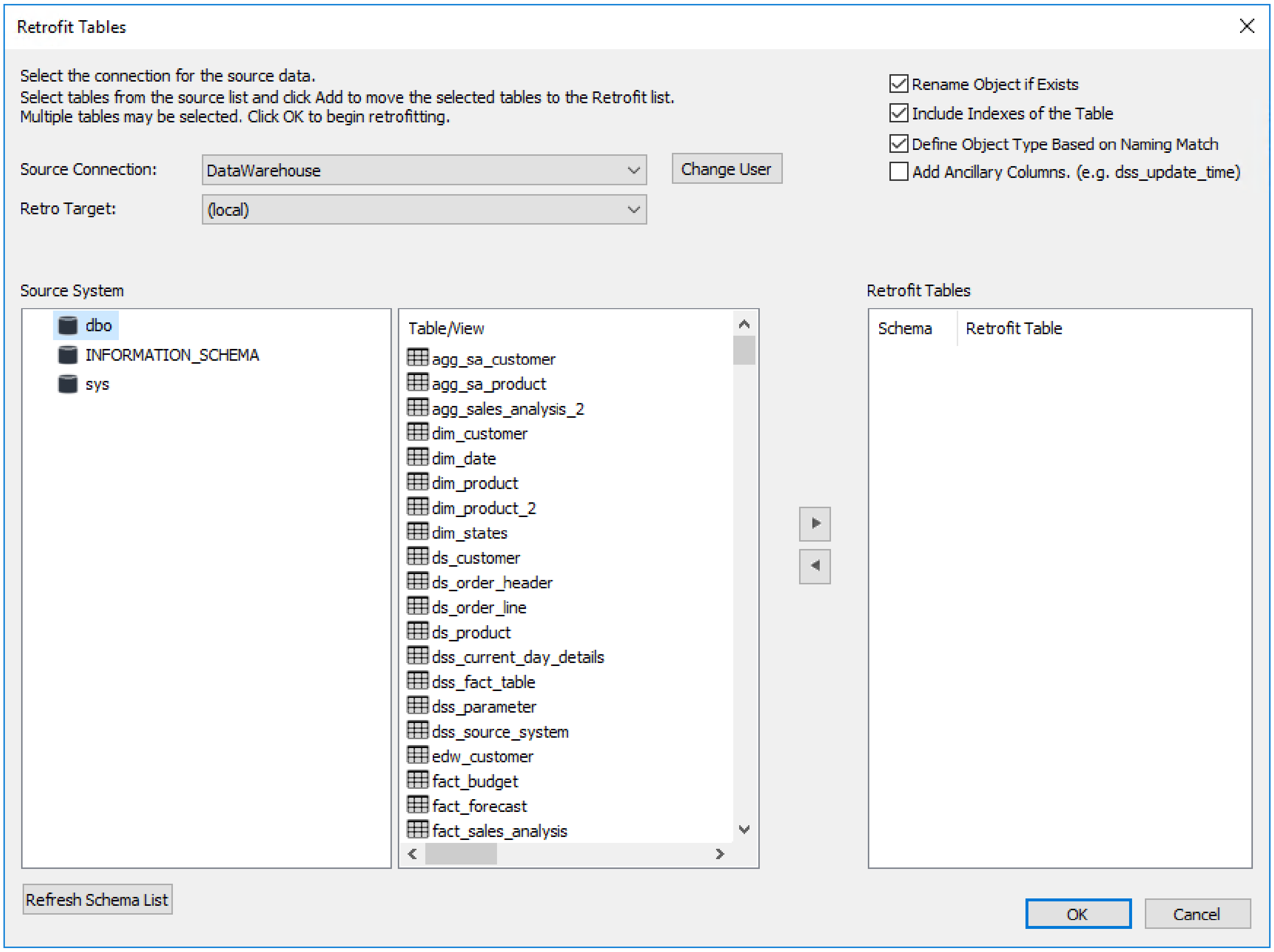
Retro Target: (local)

☑ Rename Object if Exists
☑ Include Indexes of the Table
☑ Define Object Type Based on Naming Match
☐ Add Ancillary Columns. (e.g. dss_update_time)

Source System

- dbo
- INFORMATION_SCHEMA
- sys

3. Double-click the data warehouse database/schema in the left pane list. A list of **tables** in the database is displayed in the middle pane.



Retrofit Tables

Select the connection for the source data.
Select tables from the source list and click Add to move the selected tables to the Retrofit list.
Multiple tables may be selected. Click OK to begin retrofitting.

☑ Rename Object if Exists
☑ Include Indexes of the Table
☑ Define Object Type Based on Naming Match
□ Add Ancillary Columns. (e.g. dss_update_time)

Source Connection: DataWarehouse

Retro Target: (local)

Source System

- dbo
- INFORMATION_SCHEMA
- sys

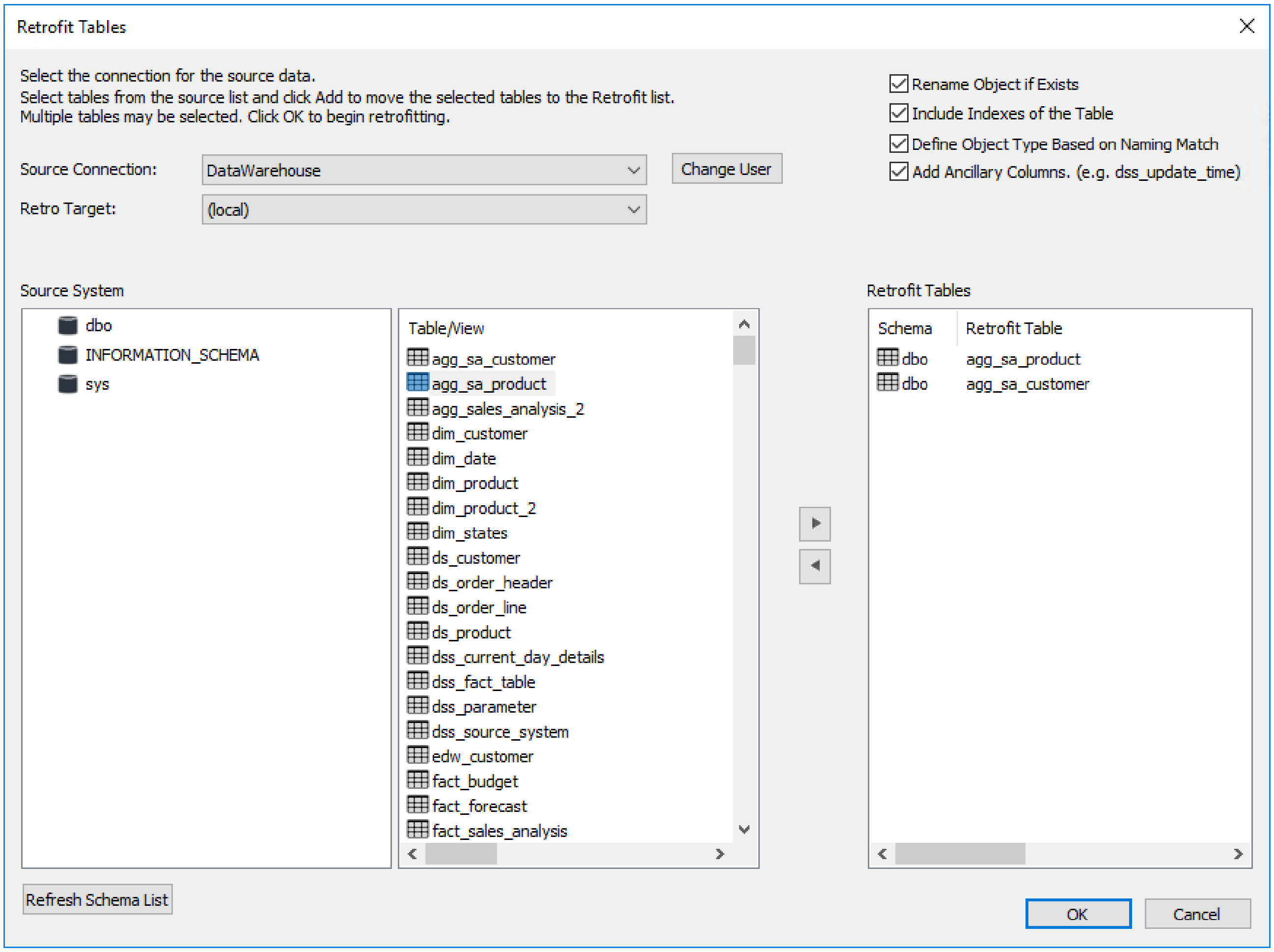
Table/View

- agg_sa_customer
- agg_sa_product
- agg_sales_analysis_2
- dim_customer
- dim_date
- dim_product
- dim_product_2
- dim_states
- ds_customer
- ds_order_header
- ds_order_line
- ds_product
- dss_current_day_details
- dss_fact_table
- dss_parameter
- dss_source_system
- edw_customer
- fact_budget
- fact_forecast
- fact_sales_analysis

Retrofit Tables

Schema	Retrofit Table

4. Click the required tables in the middle pane list and click **>** to move them to the right pane. Then click the **Add Ancillary Columns (e.g. dss_update_time)** check box. Click **OK**.



Retrofit Tables

Select the connection for the source data.
Select tables from the source list and click Add to move the selected tables to the Retrofit list.
Multiple tables may be selected. Click OK to begin retrofitting.

Source Connection: DataWarehouse

Retro Target: (local)

☑ Rename Object if Exists
☑ Include Indexes of the Table
☑ Define Object Type Based on Naming Match
☑ Add Ancillary Columns. (e.g. dss_update_time)

Source System: dbo, INFORMATION_SCHEMA, sys

Table/View: agg_sa_customer, agg_sa_product, agg_sales_analysis_2, dim_customer, dim_date, dim_product, dim_product_2, dim_states, ds_customer, ds_order_header, ds_order_line, ds_product, dss_current_day_details, dss_fact_table, dss_parameter, dss_source_system, edw_customer, fact_budget, fact_forecast, fact_sales_analysis

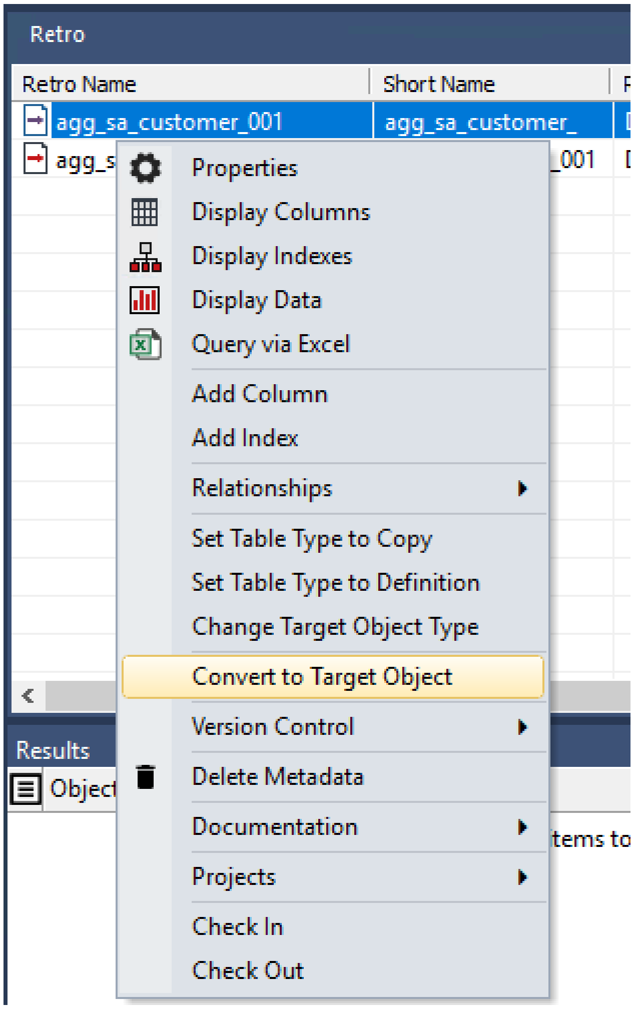
Retrofit Tables

Schema	Retrofit Table
dbo	agg_sa_product
dbo	agg_sa_customer

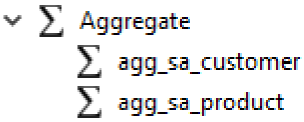
5. Double-click the **Retro object group** in the object tree in the left pane. Select all objects in the middle pane, right-click and select **Convert to Target Object**.

Notes

If the appropriate Target Object Type has not been set for one or more Retro objects; in the right click menu select **Change Target Object Type** and select the correct Object Type.



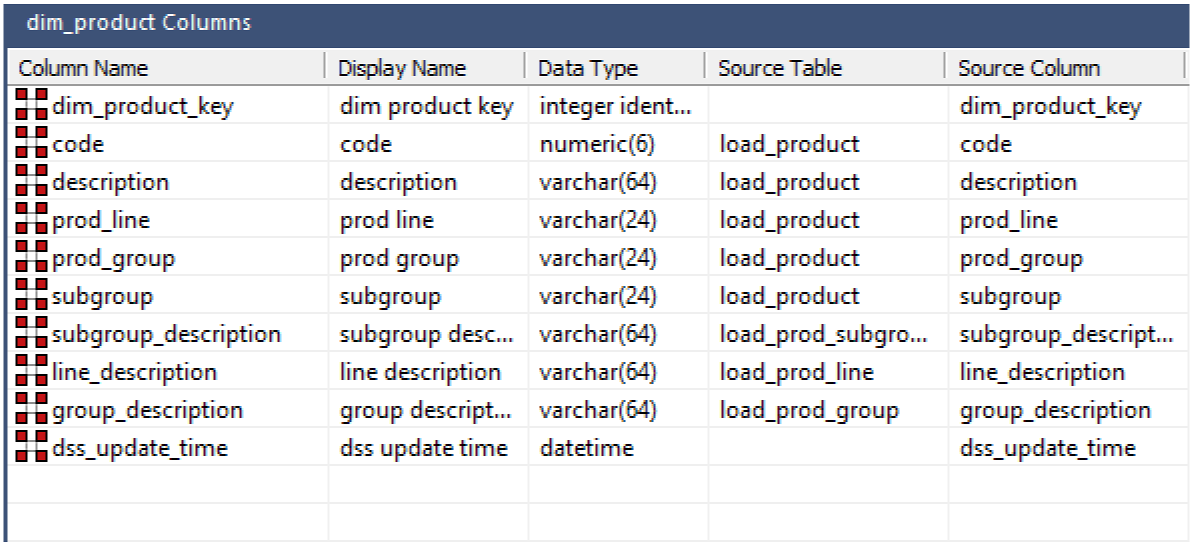
6. The new aggregate tables are imported and listed in the **Aggregate** object group in the left pane.



7. At this stage, you have only created the table metadata. To create the tables in the data warehouse, double click the object group in the left pane. In the middle pane highlight the tables, then right-click and select **Create (ReCreate)**.

Re-Targeting Source Tables

Objects that have been retrofitted into the WhereScape RED meta data have themselves as their source table:

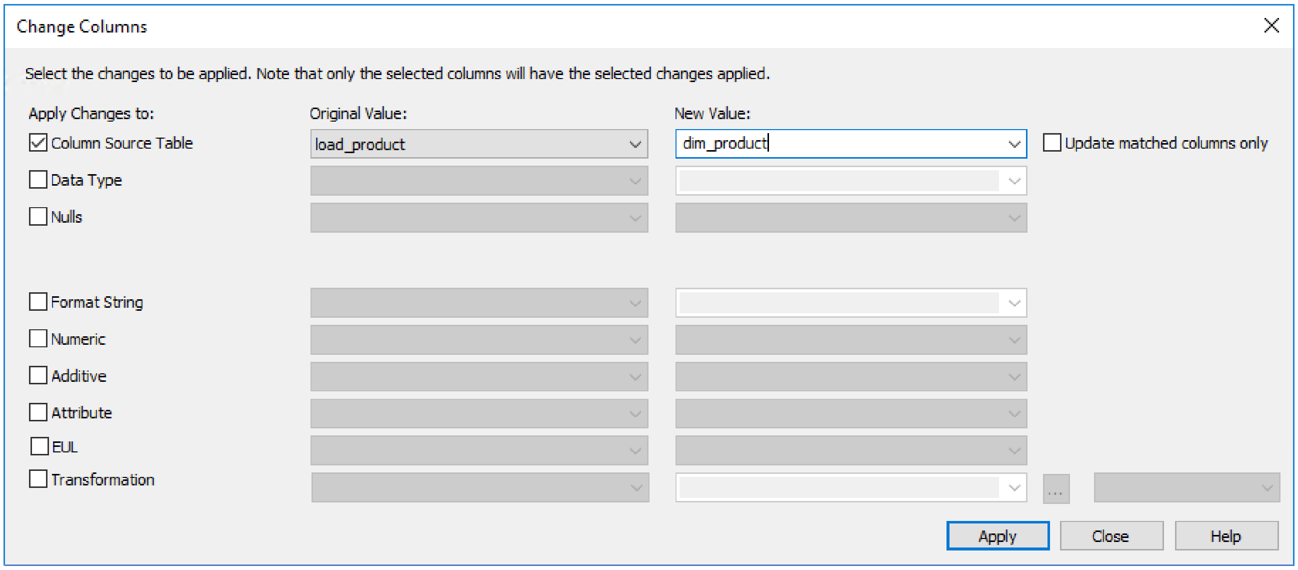


dim_product Columns

Column Name	Display Name	Data Type	Source Table	Source Column
dim_product_key	dim product key	integer ident...		dim_product_key
code	code	numeric(6)	load_product	code
description	description	varchar(64)	load_product	description
prod_line	prod line	varchar(24)	load_product	prod_line
prod_group	prod group	varchar(24)	load_product	prod_group
subgroup	subgroup	varchar(24)	load_product	subgroup
subgroup_description	subgroup desc...	varchar(64)	load_prod_subgro...	subgroup_descript...
line_description	line description	varchar(64)	load_prod_line	line_description
group_description	group descript...	varchar(64)	load_prod_group	group_description
dss_update_time	dss update time	datetime		dss_update_time

They can be re-targeted to the correct source table(s) using the WhereScape RED re-target wizard as follows.

1. Right-click a table object in the left pane and select **Change Column(s)**.
2. Select the **Column Source Table** check box. Select **load_product** from the **Original Value** drop-down list. Select **dim_product** from the **New Value** drop-down list and then click **Apply**.



Change Columns

Select the changes to be applied. Note that only the selected columns will have the selected changes applied.

Apply Changes to: | Original Value: | New Value:

☑ Column Source Table | load_product | dim_product | □ Update matched columns only

□ Data Type

□ Nulls

□ Format String

□ Numeric

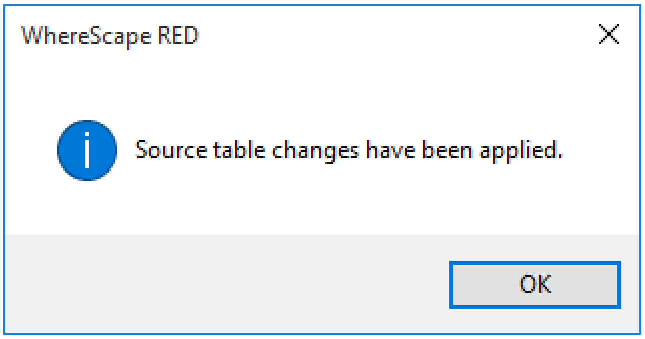
□ Additive

□ Attribute

□ EUL

□ Transformation

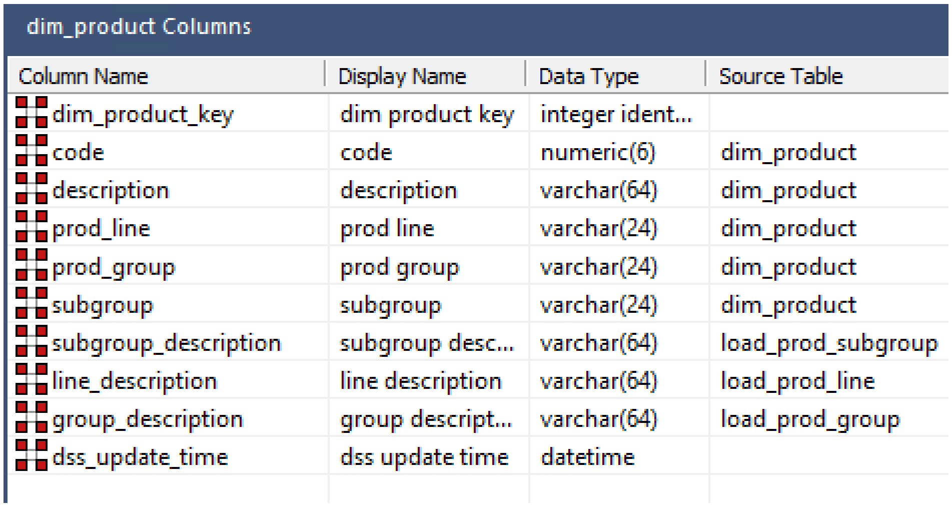
3. A message is displayed to show that the source columns have been changed, click **OK**.



WhereScape RED

Source table changes have been applied.

4. Click **Close**.
5. Confirm the **Source Table** column in the middle pane.



dim_product Columns

Column Name	Display Name	Data Type	Source Table
dim_product_key	dim product key	integer ident...	
code	code	numeric(6)	dim_product
description	description	varchar(64)	dim_product
prod_line	prod line	varchar(24)	dim_product
prod_group	prod group	varchar(24)	dim_product
subgroup	subgroup	varchar(24)	dim_product
subgroup_description	subgroup desc...	varchar(64)	load_prod_subgroup
line_description	line description	varchar(64)	load_prod_line
group_description	group descript...	varchar(64)	load_prod_group
dss_update_time	dss update time	datetime	

Retro Column Properties

Each Retro column has a set of associated properties. The definition of each property is defined below.

If the **Column name** or **Data type** is changed for a column then the metadata will differ from the table as recorded in the database. Use the **Validate > Validate Table Create Status** menu option to compare the metadata to the table in the database. When positioned on the table name after the validate has completed, a right-click menu option **Alter Table** will alter the database table to match the metadata definition.

Tip

If a database table's definition is changed in the metadata then the table will need to be altered in the database. Use the **Validate > Validate Table Create Status** to compare metadata definitions to physical database tables. The option also provides the ability to alter the database table through a pop-up menu option from the validated table name.

General

Fields	Description
Table Name	Database-compliant name of the table that contains the column. [Read-only].
Column Name	Database-compliant name of the column. Typically column-naming standards exclude the use of spaces etc. A good practice is to only use alphanumerics, and the underscore character. Changing this field alters the table's definition. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Business Display Name	Name that the business uses to refer to the column, which is included in the RED-generated documentation and can be used in the end user layer of other tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. **Note** A case conversion button on the right converts the text between different cases: UPPER CASE, Capitalized Case and lower case. The mode cycles to the next case in the sequence each conversion.
Column Description	This field contains the description for the column. In the case of dimension keys, this field

Fields	Description
	is used to show the join between the Retro table and the dimension, once it has been defined as part of the Retro table update procedure generation.

Physical Definition

Fields	Description
Column Order	Numeric value that controls the relative order of columns in the database create statement. The lowest numbered column will appear first in the table. Although this affects the physical table definition no action will be taken unless the table is re-created in the database. The columns can be re-numbered based on the existing order by choosing the **Respace Order Number** pop-up menu option when positioned over any column in the table. This action will number the columns in increments of 10 starting at 10. In addition to a simple change of the order field, the column order can be changed by first displaying the columns in the middle pane and then using drag and drop to move the columns around. This drag and drop process will automatically renumber the columns as required.
Data Type	Database-compliant data type that must be valid for the target database. For SQL Server common types are integer, numeric, varchar() and datetime. Refer to the database documentation for a description of the data types available. Changing this field alters the table's definition.
Null Values Allowed	Determines whether the table column can hold NULL values or whether a value is always mandatory.
Default Value	Initial value that is assigned to the column when a row is inserted into the table but no value is specified for the column.

Meta Definition

Fields	Description
Format	Optional format mask that can be used in end user tools. [Does NOT affect the physical database table]. As such it is a free form entry and any characters are valid. Typically format masks are only used on numeric fields. Example: #,###0.00. It is not worth the effort of populating this field unless it can be utilized by the end user tools in use.
Numeric	Indicates whether the table column holds values that are numeric. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Additive	Indicates whether the table column holds values that are additive. This implies that the column can be summed when performing data grouping in a query. This is normally only relevant for fact tables. It does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the population of an end user tool's end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.
Attribute	Indicates whether the table column holds values that are descriptive, and/or are used for grouping/summing. An attribute is defined as a column that is non factual. For example we may have an order number, or a invoice number stored in the fact table. Such columns are considered attributes, rather than facts. This checkbox is therefore normally only relevant for fact tables. This checkbox does not affect the physical table definition, but rather provides input to the view **ws_admin_v_dim_col** which can be used to assist in the

<table>
<tr><th>Fields</th><th>Description</th></tr>
<tr><td></td><td>population of an end user tools end user layer. The use of this field is not relevant unless it can be utilized by the end user tools.</td></tr>
<tr><td><b>Business Key</b></td><td>Indicates whether the column is part of the BUSINESS key, which is defined during the update procedure build. [Normally maintained automatically]. Multiple columns can form the primary business key.</td></tr>
<tr><td><b>Key Type</b></td><td>Key type that is assigned and used when generating the table's update procedure and indexes. [Normally maintained automatically]. It can be altered here, but this should only be done if the consequences are fully understood and tested.The supported values are:
<table>
<tr><th>Key type</th><th>Meaning</th></tr>
<tr><td><b>0</b></td><td>The artificial key. Set when the key is added during drag and drop table generation.</td></tr>
<tr><td><b>1</b></td><td>Component of all business keys. Indicates that this column is used as part of any business key. For example: By default the dss_source_system_key is added to every dimension table. It is considered part of any lookup on that table and has the key type set to 1. Set when the column is added during drag and drop table generation.</td></tr>
<tr><td><b>2</b></td><td>Indicates that this column is a dimensional join. Used on fact tables to indicate the dimension keys. Results in bitmap indexes being built for the columns. Set during the update procedure generation for a fact table, based on information from the staging table.</td></tr>
<tr><td><b>3</b></td><td>Slowly changing column indicator. Used on dimension tables to indicate that the column is being managed as a slowly changing column within the context of a slowly changing dimension. Set when a column is identified during the dimension update procedure generation.</td></tr>
<tr><td><b>4</b></td><td>Previous value column indicator. Used on dimension tables to indicate that the column is being managed as a previous value column. The source column identifies the parent column. Set during the dimension creation.</td></tr>
<tr><td><b>A</b></td><td>Indicates that the column is part of the primary business key. Set whenever a business key is defined as part of an update procedure generation.</td></tr>
<tr><td><b>B-Z</b></td><td>Indicates that the column is part of a secondary business key. Only used during index generation and not normally set.</td></tr>
</table></td></tr>
</table>

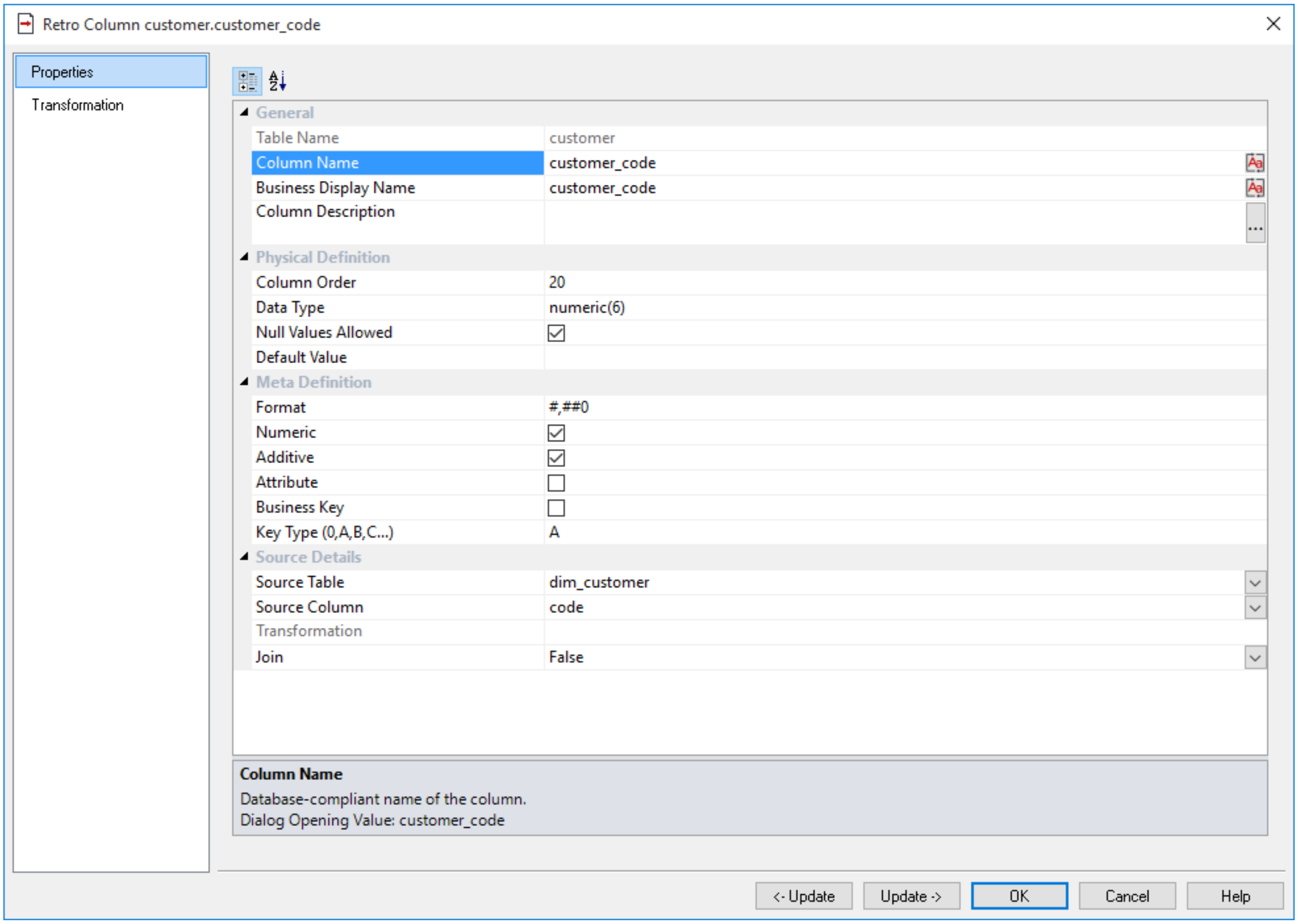
Source Details

Fields	Description
Source Table	Identifies the source table where the column's data comes from.
Source Column	Identifies the source column where the column's data comes from.
Transformation	Transformation. [Read-only].
Join	Indicates whether the table column is used in a table join. [Normally maintained automatically but can be optionally changed to override the default join logic used in the generated update procedure]. The **Source Table** and **Source Column** fields will provide the dimension table's side of the join. The options for this field are: False, True, Manual and Pre Join. • Setting this field to Manual changes the way the dimension table is looked up

Fields	Description
	during the update procedure build. It allows you to join the dimension manually in the Cursor mapping dialog (where the 'Where' clause is built). • Setting this field to Pre Join activates the **Pre Join Source Table** field and allows you to select a table from the drop-down list.
Pre Join Source Table	Indicates the table from which the pre joined column was sourced. When the Join option is set to False, this field becomes inactive. When the Join option is set to True or Manual, this field is set to the current table name. When the Join option is set to Pre Join, then you can select the required table from the drop-down list.

Retro Column Properties Screen

A sample **Properties** screen is as follows:



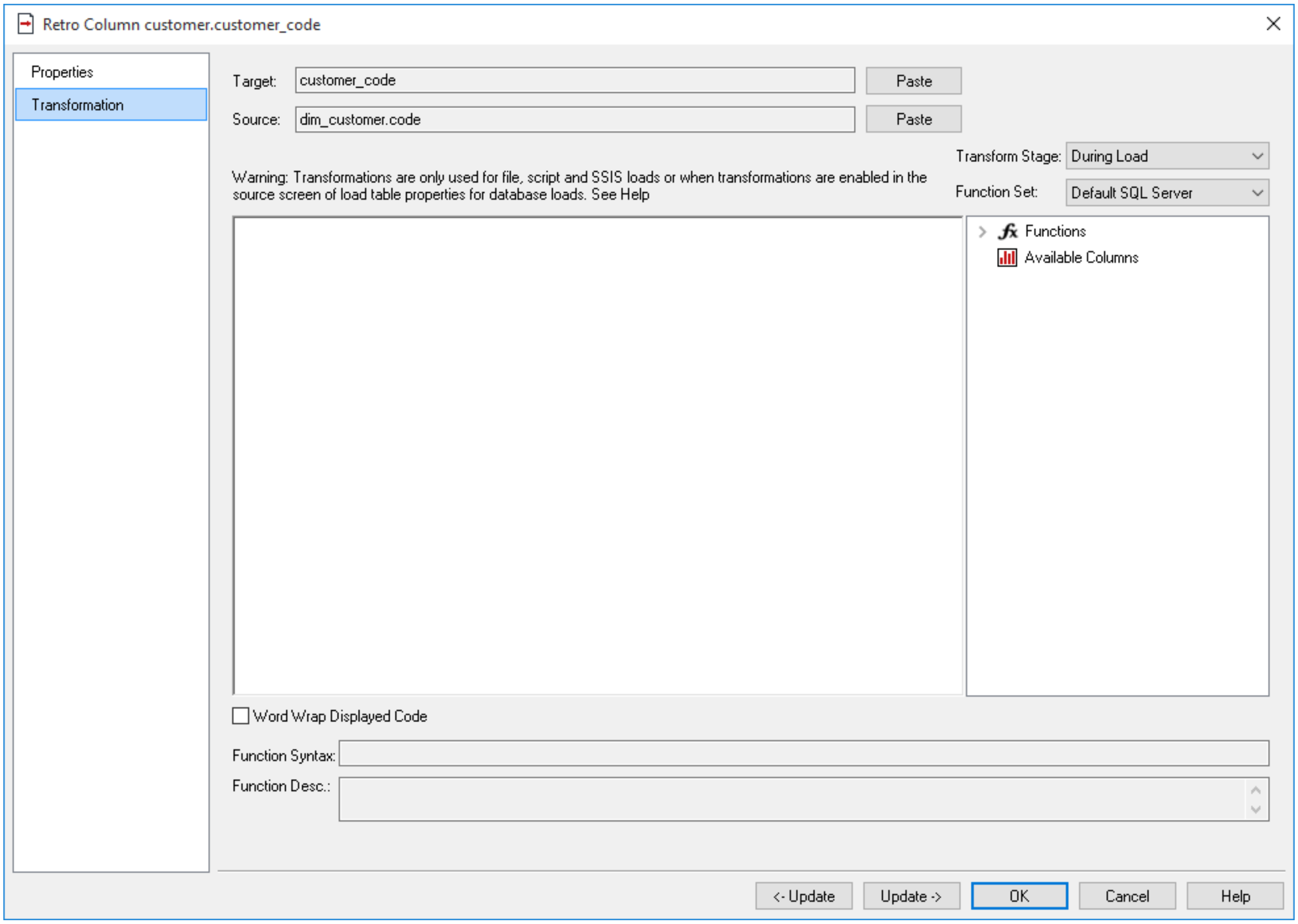
Retro Column customer.customer_code

General	
Table Name	customer
Column Name	customer_code
Business Display Name	customer_code
Column Description	
Physical Definition	
Column Order	20
Data Type	numeric(6)
Null Values Allowed	☑
Default Value	
Meta Definition	
Format	#,##0
Numeric	☑
Additive	☑
Attribute	□
Business Key	□
Key Type (0,A,B,C...)	A
Source Details	
Source Table	dim_customer
Source Column	code
Transformation	
Join	False

Column Name
Database-compliant name of the column.
Dialog Opening Value: customer_code

Retro Column Transformations

It is possible to do transformations on Retro table columns. It is recommended that transformations are not performed on columns that are dimension keys or the business keys for the table. The transformation screen is as follows:



Retro Column customer.customer_code

Target: customer_code

Source: dim_customer.code

Warning: Transformations are only used for file, script and SSIS loads or when transformations are enabled in the source screen of load table properties for database loads. See Help

Transform Stage: During Load

Function Set: Default SQL Server

□ Word Wrap Displayed Code

Function Syntax:

Function Desc.:

Note

Transformations are only put into effect when the procedure is re-generated.

Refer to **Transformations** for details.

Integrating WhereScape RED into an Existing Warehouse

Two main options exist in terms of bringing WhereScape RED into an existing data warehouse environment:

1. Rebuild tables and procedures with WhereScape RED.
2. Integrating existing tables and procedures into WhereScape RED.

Both options require manual coding changes to stored procedures. The main advantages and disadvantages of these two options are discusses below, and in detail in the following sections.

Rebuild

The rebuild option is essentially a redevelopment of the existing data warehouse utilizing the knowledge acquired in the initial development and the rapid development capabilities of WhereScape RED. A rebuild will take more time and effort than just integrating existing tables and procedures but will provide a better platform on which to extend the data warehouse. Refer to **Rebuilding** for details.

Integrate

Existing data warehouse tables can be identified to WhereScape RED. The tables are seen and can be managed to a degree. The main disadvantage is the increased difficulty in utilizing these tables when trying to extend the data warehouse. This option is however significantly quicker and easier than a rebuild. Refer to **Integrating** for details.

The decision as to which option to choose will depend on the size and complexity of the existing data warehouse. Another important factor is the degree to which the existing data warehouse is to be extended. If future enhancements revolve around new analysis areas that have little overlap with the existing environment then a integrate may be the best answer. If the data warehouse is small and relatively simple than a rebuild may be worth considering. In any event the best plan may be to do a test integrate and then re-evaluate the situation.

Rebuilding

The rebuild process essentially is a total re-creation of the data warehouse. One of the major impacts of such an approach is the **end user layer**, or rather the effect on the end user tools and saved queries and reports that are currently in use. The redesign or redeployment of this interface to the end users of the data warehouse may be too large a task to undertake. The problem can be circumvented to some degree though the use of views to make the new data warehouse environment look the same as the previous. But it is this impact and the subsequent re-testing process that must be considered when deciding to undertake a rebuild.

The advantages of a rebuild is the seamless integration of future analysis areas into the data warehouse and the single point of management that is provided. The major steps in the rebuild process will depend very much on the environment being replaced. As a guideline the following steps may be worth considering.

The rebuild process

1. Load a copy of the WhereScape metadata repository into an otherwise empty test environment. Refer to the RED Installation Guide for instructions on how to load a metadata repository.
2. Ensure there are no public synonyms that point to existing table names, if the rebuild process is to use the same names as some or all of the existing tables.

Working within the WhereScape RED tool proceed to:

1. Create connections to the new test data warehouse and to all the source systems.
2. Using the source system knowledge from the existing data warehouse, create the appropriate load tables in the data warehouse based on the existing extract or load methodology.
3. Build up the dimension, stage and fact tables using the same column and table names where possible.

4. Examine the existing procedures or update methodology and include this into the generated stored procedures.
5. Test the new environment.
6. Work out a plan to convert the existing data into the new data warehouse. Where possible it is best to keep existing key values and re-assign sequences to match these existing key values where appropriate.
7. Convert and test the old data warehouse data in the new environment.
8. Redeploy the end user tool access.

Integrating

The integrate process

The steps in the integrate process are:

1. Create a test environment (database user) with the existing data warehouse tables loaded.
2. Load a copy of the WhereScape metadata repository into this test environment. Refer to the RED Installation Guide for instructions on how to load a metadata repository.

Working within the WhereScape RED tool proceed to:

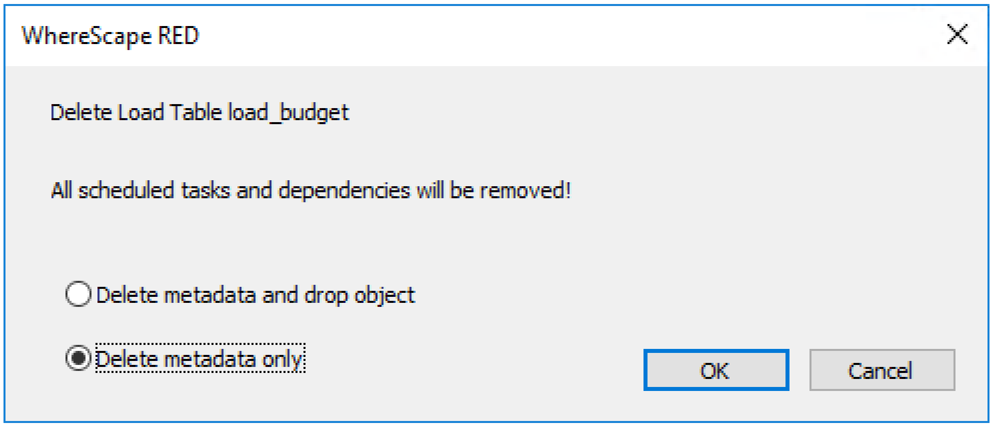
1. Create any connections to UNIX or Windows servers where host scripts are currently executed. Refer to either **Windows connection** or **Unix connection** for details.
2. Create a Data Warehouse connection mapping back to the test environment. Refer to the **data warehouse / metadata repository connection** for details.
3. Incorporate any Host system scripts currently used. Refer to **integrating host scripts** for details.
4. Browse the Data Warehouse connection (**Browse > Source Tables**).
5. Drag and drop each existing data warehouse table into an appropriate object type. Refer to **integrating selecting table type** for details
6. Answer the retrofit questions, and build any required procedures. Refer to **integrating questions** for details.
7. Edit and amend all generated procedures, or create new procedures to handle the existing processing methodology. Refer to **integrating procedures** changes for details.
8. Test the new environment.

Removing the metadata for a table

It is possible to delete the metadata for a table without deleting the table object itself. For example, if the integrate process is incorrectly undertaken, the metadata for the specific table can be removed.

To delete the metadata only:

1. Right click the table and select **Delete Metadata and Drop Table** from the context menu. A prompt is displayed which enables you to select the **Delete metadata only** option.



WhereScape RED

Delete Load Table load_budget

All scheduled tasks and dependencies will be removed!

○ Delete metadata and drop object

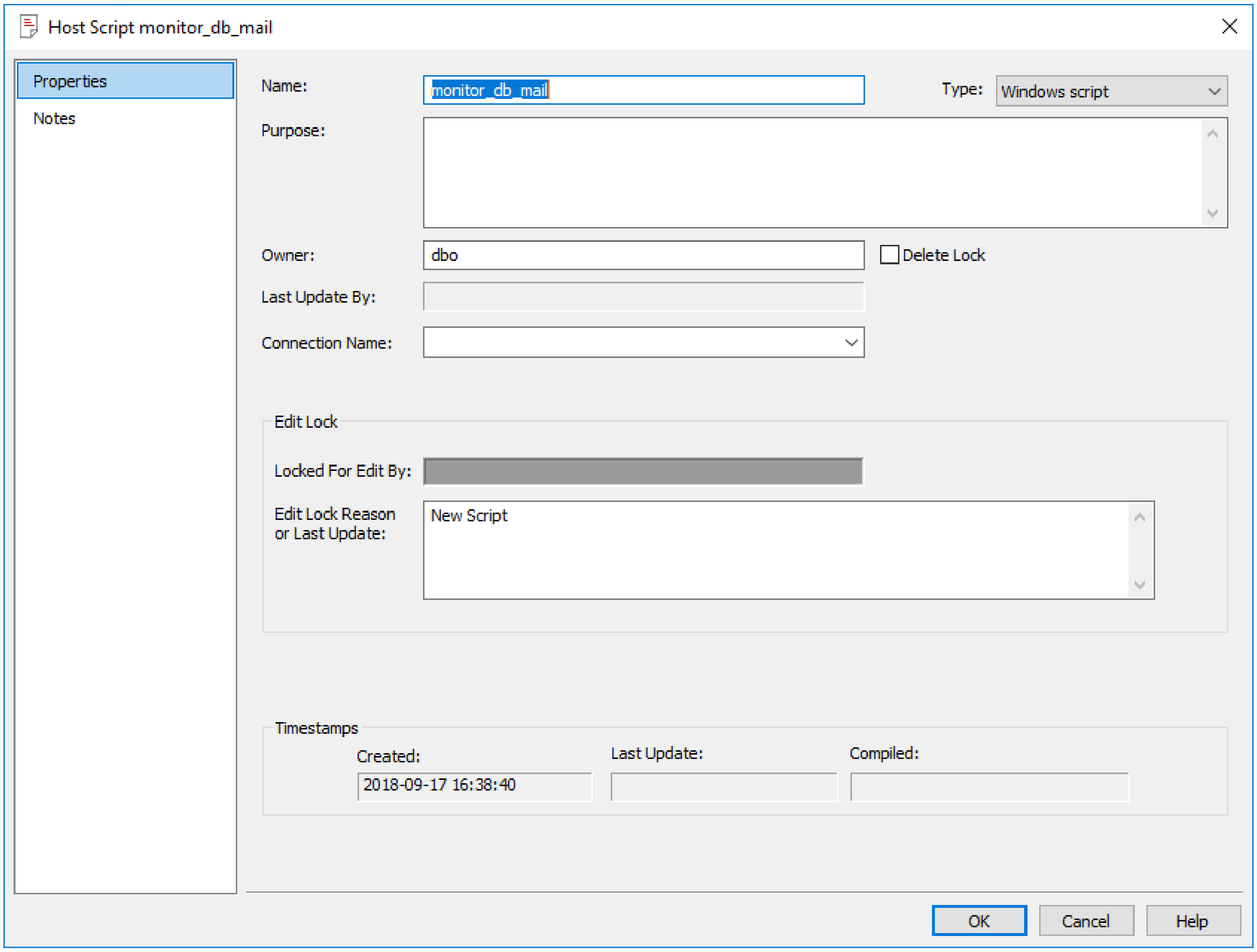
● Delete metadata only

2. Ensure you have selected the **Delete metadata only** option and before clicking **OK**.

Integrating, Host Scripts

Existing host scripts (either UNIX or Windows) can be brought into the WhereScape RED metadata. To incorporate an existing script the process is as follows:

1. Create a **Host Script** object using WhereScape RED. In the left pane, right-click a project or the **All Objects** project and select **New Object**.
2. In the new object window, select **Host Script** from the Object Type drop-down and the type in a name for the new script.
3. Click **Add**, the following **Properties** window appears. Select the script type; either UNIX or Windows script. Select the appropriate connection from the **Connection Name** drop-down. Fill in the **Purpose** field to define the role of the script and then click **OK** to save the changes.



Host Script monitor_db_mail

Name: monitor_db_mail

Type: Windows script

Purpose:

Owner: dbo

☐ Delete Lock

Last Update By:

Connection Name:

Edit Lock

Locked For Edit By:

Edit Lock Reason or Last Update: New Script

Timestamps

Created: 2018-09-17 16:38:40

Last Update:

Compiled:

4. Double-click the new script or right-click the new script and select **Edit the Script**.
5. Within the script editor, either paste the script or if it is available in your PC running RED, select the **File > Insert from File** option and load the file.
6. The script needs to be modified to handle the standards required by the RED scheduler. Refer to **Script based load** for details.

Integrating, Selecting a Table Type

When integrating existing tables there may not be a clear decision as to which table type to use. As a guideline, the following groupings can be considered.

Temporary tables:

- Load tables
- Stage tables

Permanent tables:

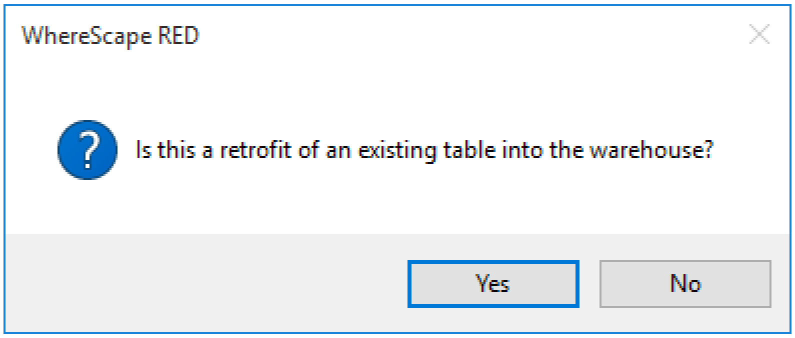
- Dimension tables
- Fact tables
- Aggregate tables

Although these table groups have very distinct names in terms of data warehousing, they do not impose any restrictions on the types of tables they contain. The table groupings are most relevant in the automatic generation of procedures, and in the sequencing for the scheduler.

Typically a mapping table may be stored in the Dimension table group.

Integrating, Questions

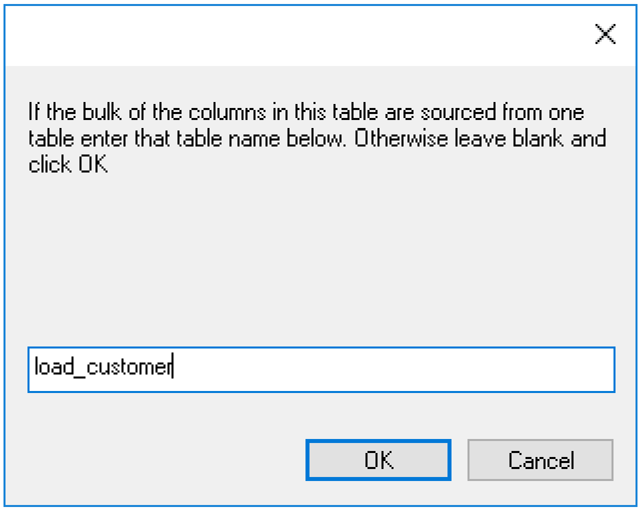
When a table within the data warehouse schema, that is unknown to WhereScape RED, is dropped onto a table target the following dialog appears.


WhereScape RED

Is this a retrofit of an existing table into the warehouse?

If this is an integrate then click **Yes** to proceed with the integrate process. The standard **New Object** dialog appears and it would be advisable to leave the name of the object unchanged so that it matches the existing table.

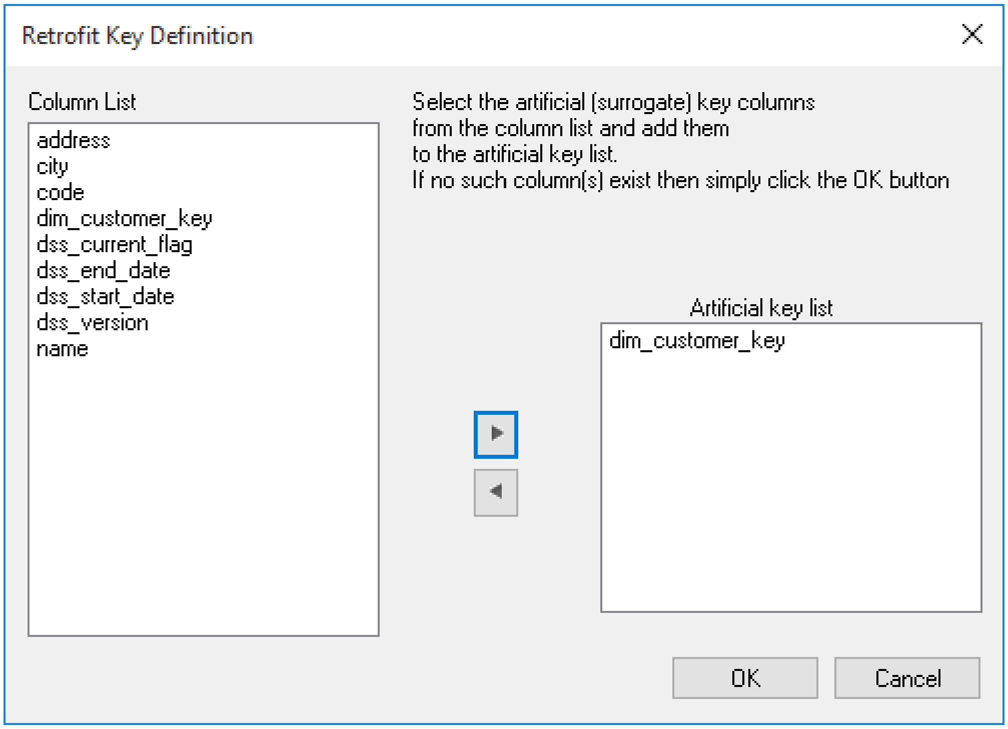
A dialog asks if the bulk of the columns in this table are derived from another table. If they are enter the table from which these columns derive at this stage. The purpose of this dialog is simply to set the **Source Table** field against each of the columns for the table.


If the bulk of the columns in this table are sourced from one table enter that table name below. Otherwise leave blank and click OK

load_customer

Artificial Key definition

If the table target is a dimension or fact table then the following dialog appears to enable the definition of any artificial or surrogate key. If no such key exists then simply proceed to the next question.



Get Key Function

If the target table is a dimension you will be asked if you want to generate a get key function. The get key function is used by WhereScape RED for rapid prototyping. It returns the artificial key when passed the business key. Although this function is possibly not necessary in the existing environment it is useful if the dimension is to be used in any expansion of the data warehouse. If an existing get key type function exists, then this can be used, although changes will probably need to be made to the generated procedures to utilize this existing get key function.

Note

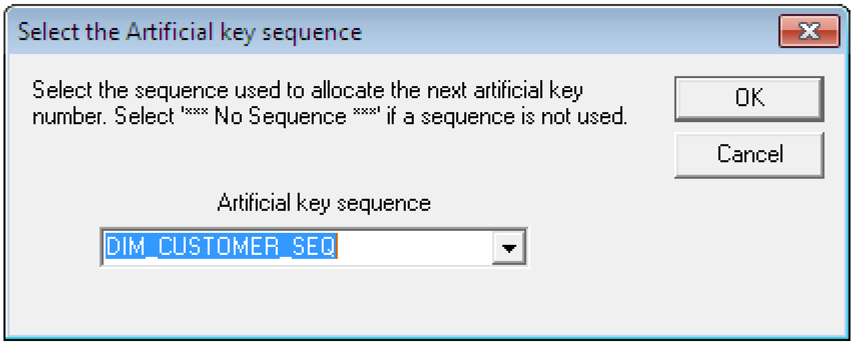
The **Get Key** function is not available for dimension objects that are stored in custom database targets.

Update Procedure

If the target table is a dimension, stage, fact or aggregate a dialog box will ask if an update procedure or update template is required. If selected a subsequent dialog will define the structure and content of this procedure. If an existing procedure is to be modified to include the scheduler imposed procedure standards it is best to select this option and use the existing procedure name to cut down on the amount of work required. The existing procedure can then be loaded into the generated procedure using the procedure editor.

Artificial Key Sequence

If a get key function or update procedure is to be built and an artificial key is in use for the table then a dialog will prompt for the artificial key sequence. A drop-down list provides all the sequences defined for this database user. If no such sequence is used then select the *** no sequence *** option.

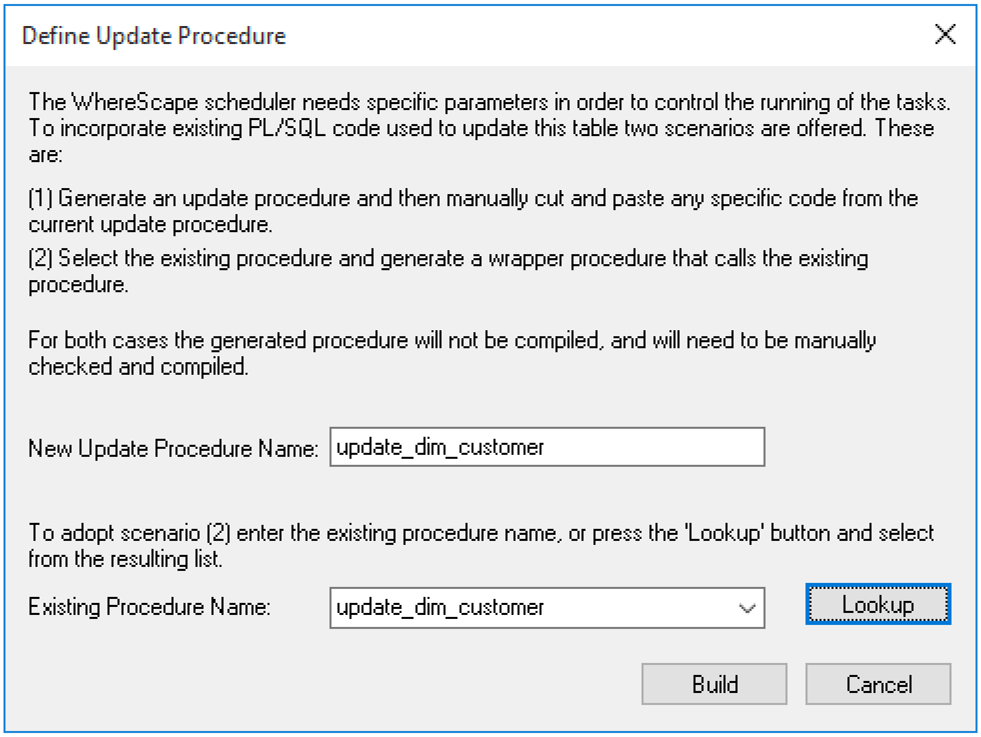


Select the sequence used to allocate the next artificial key number. Select '*** No Sequence ***' if a sequence is not used.

Artificial key sequence: DIM_CUSTOMER_SEQ

Update Procedure Definition

If an update procedure was selected then the following dialog will display to help define how the update procedure is generated.



The WhereScape scheduler needs specific parameters in order to control the running of the tasks. To incorporate existing PL/SQL code used to update this table two scenarios are offered. These are:

(1) Generate an update procedure and then manually cut and paste any specific code from the current update procedure.

(2) Select the existing procedure and generate a wrapper procedure that calls the existing procedure.

For both cases the generated procedure will not be compiled, and will need to be manually checked and compiled.

New Update Procedure Name: update_dim_customer

To adopt scenario (2) enter the existing procedure name, or press the 'Lookup' button and select from the resulting list.

Existing Procedure Name: update_dim_customer

As defined in this dialog box the scheduler needs specific parameters in order to control and run a procedure. An existing procedure will not have these parameters. There are basically two forms of generated procedure. The first option is to generate a procedure that looks much like a standard WhereScape RED procedure and then manually insert the changes from the existing procedure. The second option is to generate a wrapper procedure that calls the existing procedure. This second option will still require manual changes to the procedure to return a valid return code and message.

Business Key definition

If a get key or update procedure has been selected a dialog will now prompt for the selection of the business key from the table. Multiple columns may constitute the business key, but they must uniquely identify each record in the table.

Index definition

Any indexes associated with the table will now be automatically defined and loaded into the metadata. Changes may need to be made in terms of when the indexes are dropped and to set the **Drop Before Update** checkbox if appropriate and the scheduler is to be used to manage these indexes.

Procedure creation

If defined the get key and update procedures will now be generated. They will need to be manually edited and compiled. Refer to **Integrating procedures** for details.

Integrating, Procedures

The procedures managed by the WhereScape scheduler require the following standards.

Parameters

The procedure must have the following parameters in the following order:

Parameter name	Input/Output	SQL Server/DB2 Type
p_sequence	Input	Integer
p_job_name	Input	Varchar(256)
p_task_name	Input	Varchar(256)
p_job_id	Input	Integer
p_task_id	Input	Integer
p_return_msg	Output	Varchar(256)
p_status	Output	Integer

The input parameters are passed to the procedure by the scheduler. If the procedure is called outside the scheduler then the normal practice is to pass zero (0) in the sequence, job_id and task_id. A description of the run can be passed in the job name and the task name is typically the name of the procedure.

The output parameters must be populated by the procedure on completion. The return_msg can be any string up to 256 characters long that describes the result of the procedures execution. The status must be one of the following values:

Status	Meaning	Description
1	Success	Normal completion
-1	Warning	Completion with warnings
-2	Error	Hold subsequent tasks dependent on this task
-3	Fatal Error	Hold all subsequent tasks

The major task in integrating a procedure will be in adapting it to the WhereScape scheduler standards and work flow.

Integrating, Views

When integrating views an additional step is required if you want WhereScape RED to be able to recreate the view.

The view will be mapped correctly and the **Get Key** function can still be built. This step is only required if the view is to be re-created.

Note

The **Get Key** function is not available for dimension objects that are stored in custom database targets.

Change the source column on the artificial key to match the artificial key in the table from which the view was created.

Integrating, WhereScape Tables

When integrating WhereScape generated tables and views a number of additional considerations need to be taken.

The Dimension tables and views make use of secondary business key columns such as dss_source_system_key, dss_current_flag, dss_end_date, dss_version to assist in handling various forms of slowly changing dimensions and non conformance. These secondary business keys need to be flagged as such against the dimension.

Change the properties of all such columns. The key type should be set to 1, and the primary business key checkbox should be set.

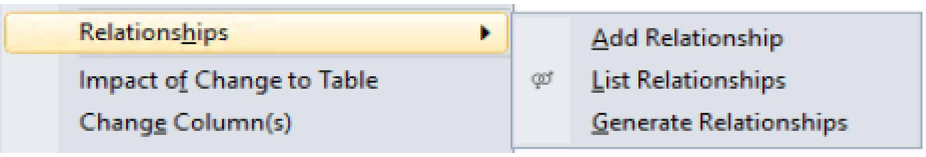
Relationship Maintenance

Relationship Maintenance is available for the maintenance of joins between objects; providing a way to record joins between tables when surrogate keys are not being used. This functionality then enables the generation of **Links Diagrams** for these objects.

Note

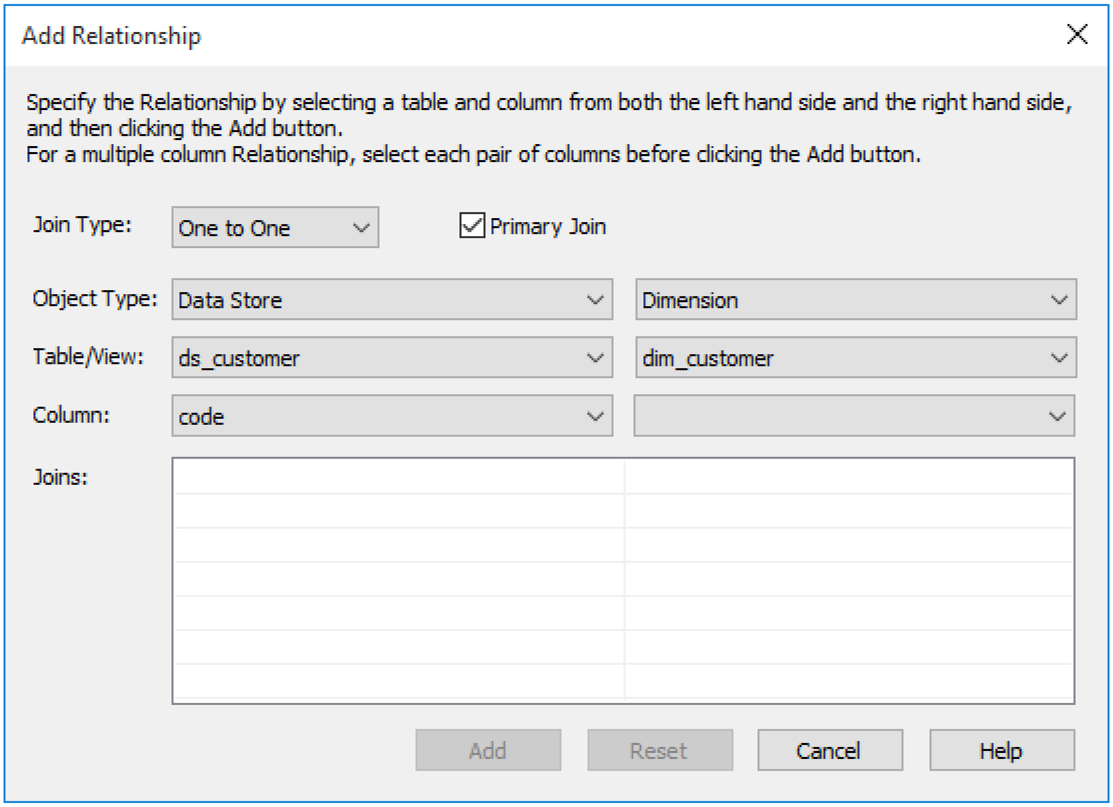
It is necessary to explicitly specify relationships for tables on **Tabular** target databases for the relationships to be created in the Tabular database.

Relationship Maintenance options are available in the Relationships sub-menu when right-clicking on an object in the **Object Pane**.



Add Relationship

To add a relationship, right-click on the object in the **Object Pane** and select **Relationships > Add Relationships**. The following window is displayed.



Specify the Relationship by selecting a table and column from both the left hand side and the right hand side, and then clicking the Add button.
For a multiple column Relationship, select each pair of columns before clicking the Add button.

Join Type: One to One

☑ Primary Join

Object Type: Data Store | Dimension

Table/View: ds_customer | dim_customer

Column: code |

Joins:

For each object in the relationship, type in the following details:

Options	Description
Join Type	For the **Join Type**, choose between the following: - Undefined

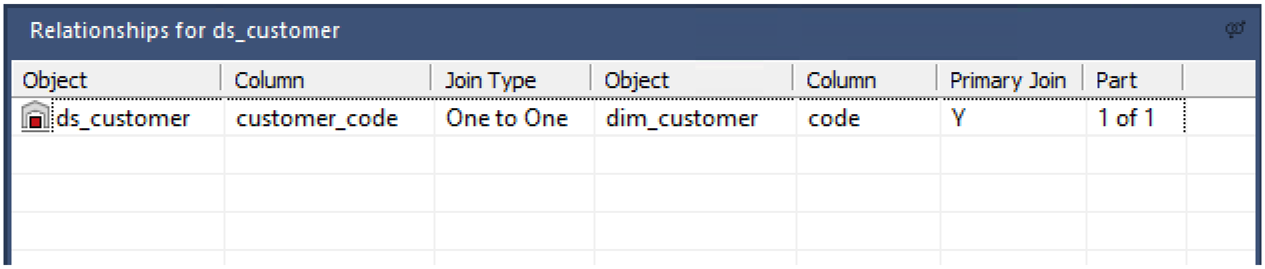
Options	Description
	- Many to One - One to One - One to Many - Many to Many
Primary Join	If this is a primary join, select the **Primary Join** check-box. For MSAS Tabular tables this defines an active relationship. At most, one relationship between two specific tables can have this enabled.
Object Type	Enter the **Object Type** from the drop-down box (Data Store, Load table, Stage table, etc.).
Table/View	Enter the name of the object in the relationship.
Column	Enter the **Column** to join in each object.

Once you have entered the details for the join, the joined columns are displayed in the list of **Joins** at the bottom of the screen. Erroneous joins can be removed by right-clicking the join and selecting **Remove Join**. All joins can be removed by clicking the **Reset** button.

To add all the relationships shown in the list of **Joins**, click the **Add** button.

List Relationships

To view relationship for an object, right-click the object from the **Object Pane** and select **Relationships > List Relationships**. The relationships for the selected object are displayed in the **Drop Target Pane** (middle pane).

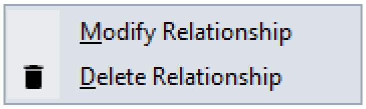


Relationships for ds_customer

Object	Column	Join Type	Object	Column	Primary Join	Part
ds_customer	customer_code	One to One	dim_customer	code	Y	1 of 1

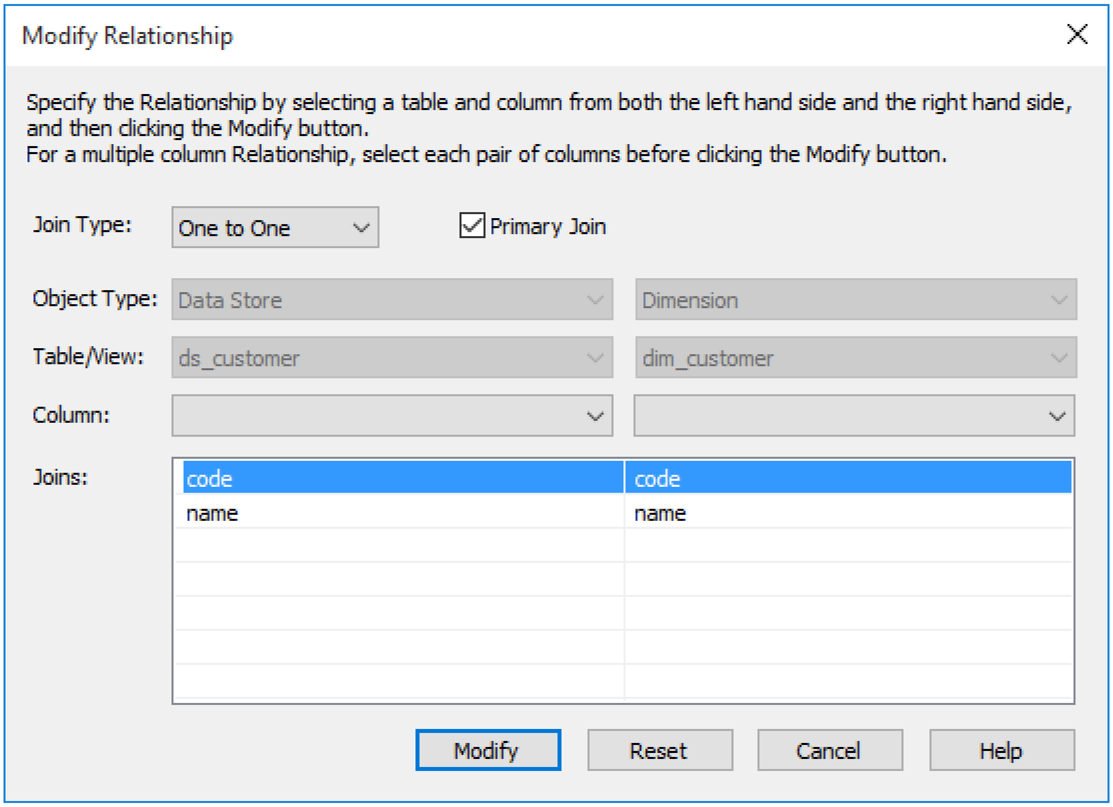
Multi-column joins are shaded when one join is selected.

Right-clicking on a column or join displays the following menu:



Modify Relationship

The Modify Relationship option shows the following window, which enables you to edit joins (including multi-column joins) between the two objects in the selected relationship.



Modify Relationship

Specify the Relationship by selecting a table and column from both the left hand side and the right hand side, and then clicking the Modify button.
For a multiple column Relationship, select each pair of columns before clicking the Modify button.

Join Type: One to One ☑ Primary Join

Object Type: Data Store | Dimension

Table/View: ds_customer | dim_customer

Column:

Joins:

code	code
name	name

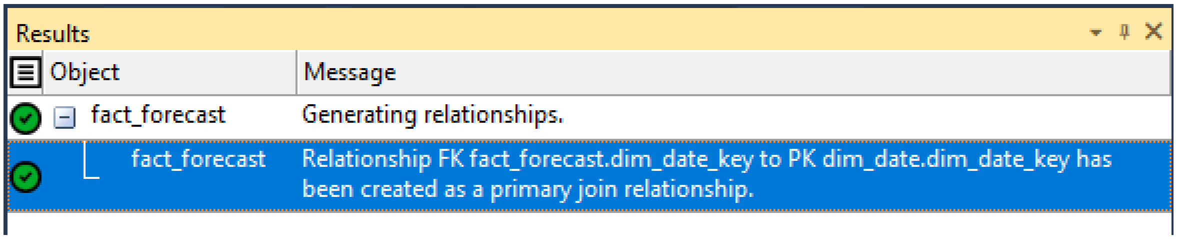
Relationships are edited in this window in the same way as the **Add Relationship** window. **Object Types** and **Table names** cannot be modified.

Delete Relationship

Deletes the selected relationship.

Generate Relationships

To generate relationships in metadata for an object, right-click the object from the **Object Pane** and select **Relationships > Generate Relationships**. Results are shown in the **Results Pane**.



Results

Object	Message
fact_forecast	Generating relationships.
fact_forecast	Relationship FK fact_forecast.dim_date_key to PK dim_date.dim_date_key has been created as a primary join relationship.

Login Checks

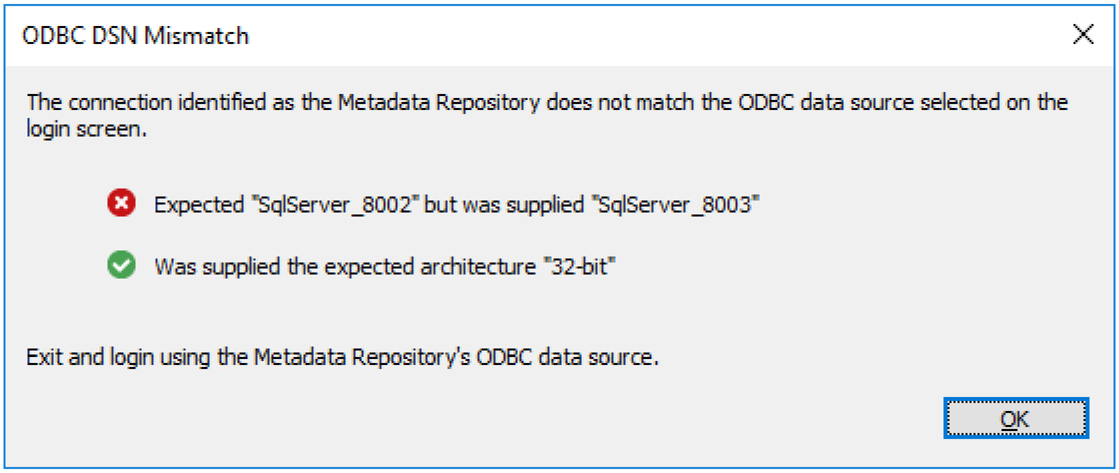
The following checks are performed during login; and if necessary, warning messages are displayed, see more about each specific warning by following the links below:

Login Data Source does not match the data warehouse connection's ODBC DSN

More than one connection has the data warehouse option enabled

Data Source does not match the data warehouse connection's ODBC DSN

Warning about the login Data Source not matching the data warehouse connection's **ODBC DSN:**



ODBC DSN Mismatch

The connection identified as the Metadata Repository does not match the ODBC data source selected on the login screen.

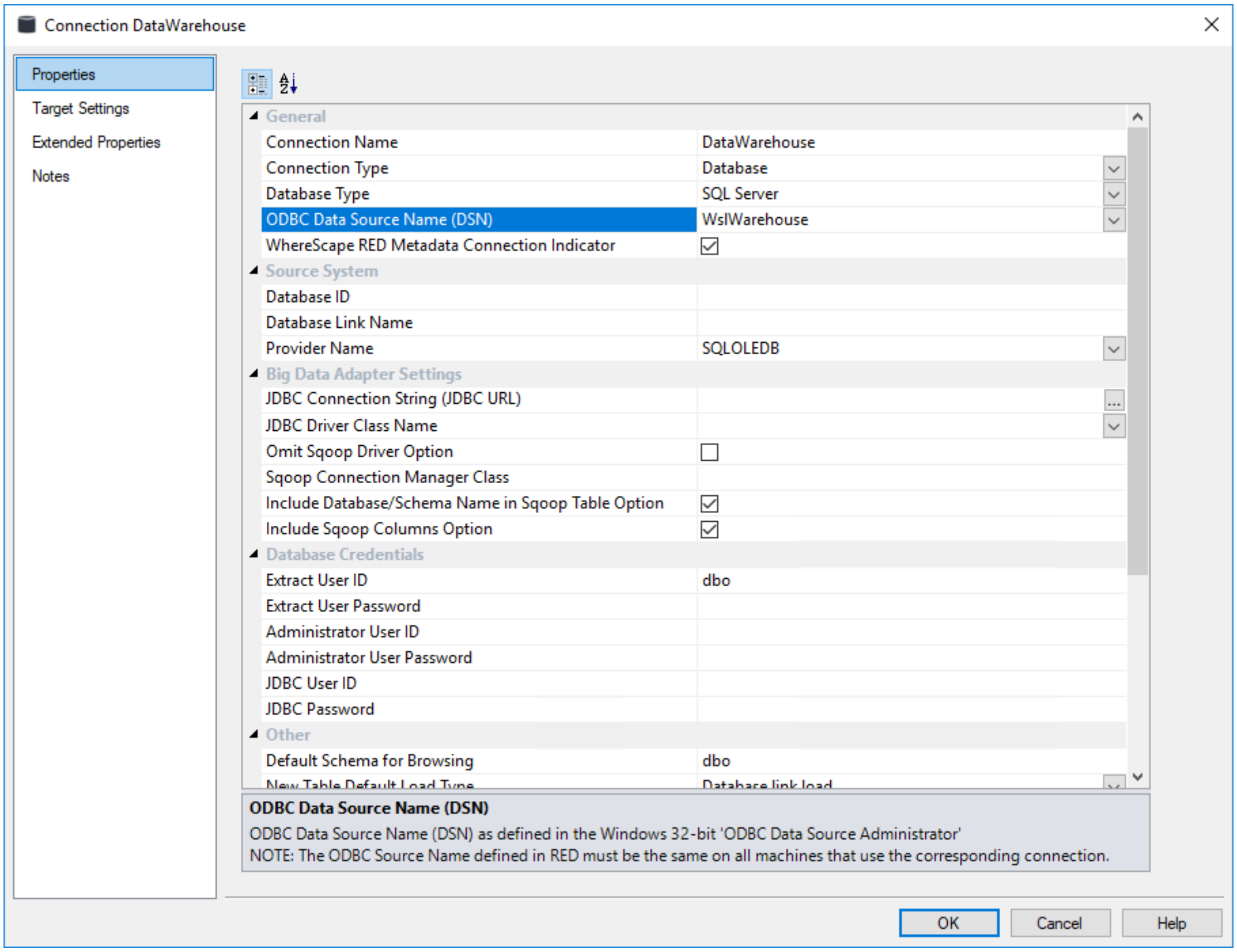
✗ Expected "SqlServer_8002" but was supplied "SqlServer_8003"

✓ Was supplied the expected architecture "32-bit"

Exit and login using the Metadata Repository's ODBC data source.

You can correct this issue by performing one of the following actions:

1. Alter the Data Warehouse Connection **ODBC Data Source Name (DSN)** to match the login **Data Source**.



General	
Connection Name	DataWarehouse
Connection Type	Database
Database Type	SQL Server
ODBC Data Source Name (DSN)	WslWarehouse
WhereScape RED Metadata Connection Indicator	☑
Source System	
Database ID	
Database Link Name	
Provider Name	SQLOLEDB
Big Data Adapter Settings	
JDBC Connection String (JDBC URL)	
JDBC Driver Class Name	
Omit Sqoop Driver Option	□
Sqoop Connection Manager Class	
Include Database/Schema Name in Sqoop Table Option	☑
Include Sqoop Columns Option	☑
Database Credentials	
Extract User ID	dbo
Extract User Password	
Administrator User ID	
Administrator User Password	
JDBC User ID	
JDBC Password	
Other	
Default Schema for Browsing	dbo
New Table Default Load Type	Database link load

ODBC Data Source Name (DSN)

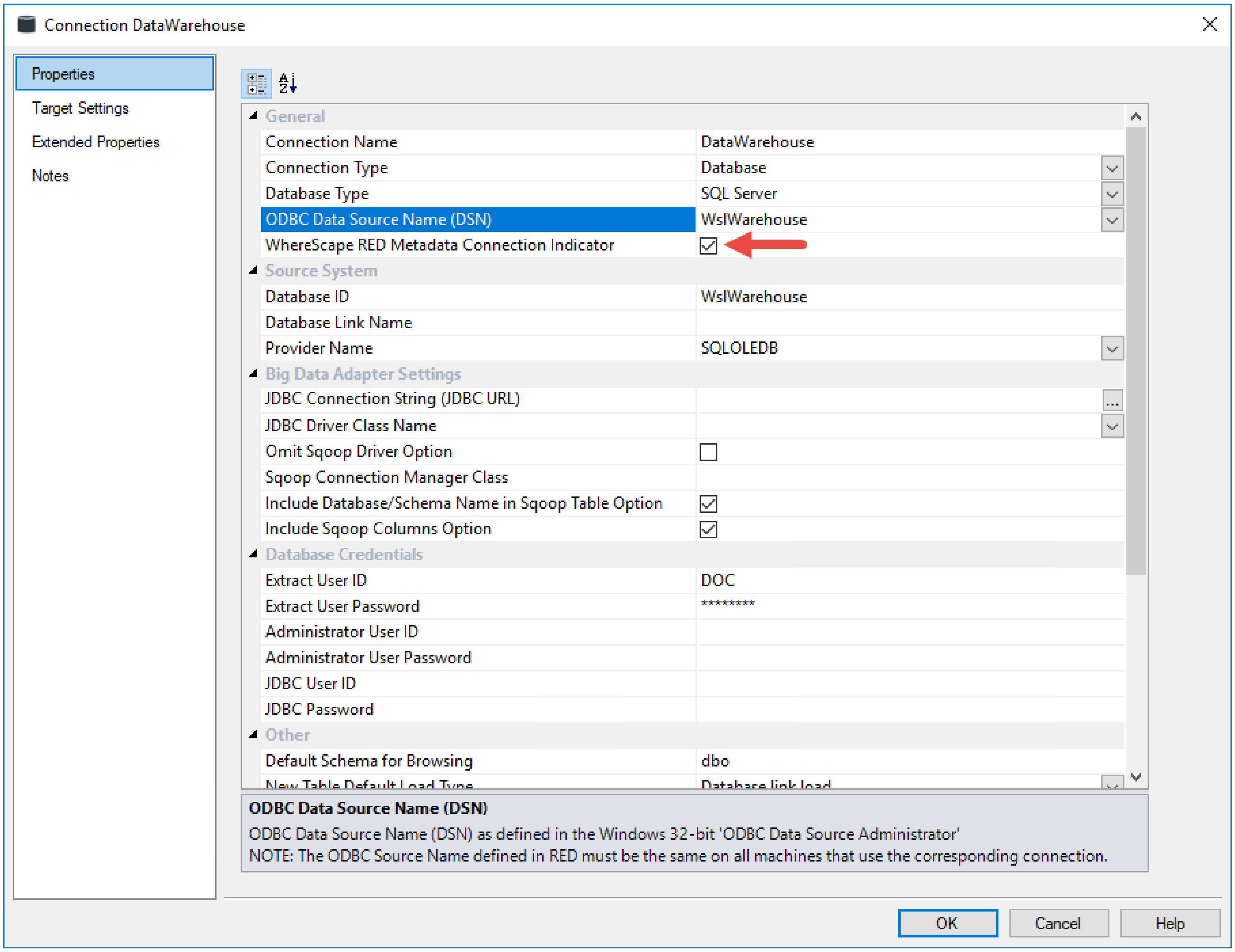
ODBC Data Source Name (DSN) as defined in the Windows 32-bit 'ODBC Data Source Administrator'
NOTE: The ODBC Source Name defined in RED must be the same on all machines that use the corresponding connection.

2. Logoff and log back in using the **Data Source** with the same name as the Data Warehouse Connection **ODBC Source**.

More than one connection has the Data Warehouse Indicator Enabled

Warning about more than one connection having the data warehouse option enabled.

You can correct this issue by editing all the connections and making sure that only one is set to Data Warehouse:



Connection DataWarehouse

General

Connection Name	DataWarehouse
Connection Type	Database
Database Type	SQL Server
ODBC Data Source Name (DSN)	WslWarehouse
WhereScape RED Metadata Connection Indicator	☑

Source System

Database ID	WslWarehouse
Database Link Name	
Provider Name	SQLOLEDB

Big Data Adapter Settings

JDBC Connection String (JDBC URL)	
JDBC Driver Class Name	
Omit Sqoop Driver Option	□
Sqoop Connection Manager Class	
Include Database/Schema Name in Sqoop Table Option	☑
Include Sqoop Columns Option	☑

Database Credentials

Extract User ID	DOC
Extract User Password	********
Administrator User ID	
Administrator User Password	
JDBC User ID	
JDBC Password	

Other

Default Schema for Browsing	dbo
New Table Default Load Type	Database link load

ODBC Data Source Name (DSN)

ODBC Data Source Name (DSN) as defined in the Windows 32-bit 'ODBC Data Source Administrator'
NOTE: The ODBC Source Name defined in RED must be the same on all machines that use the corresponding connection.

Data Type Mappings

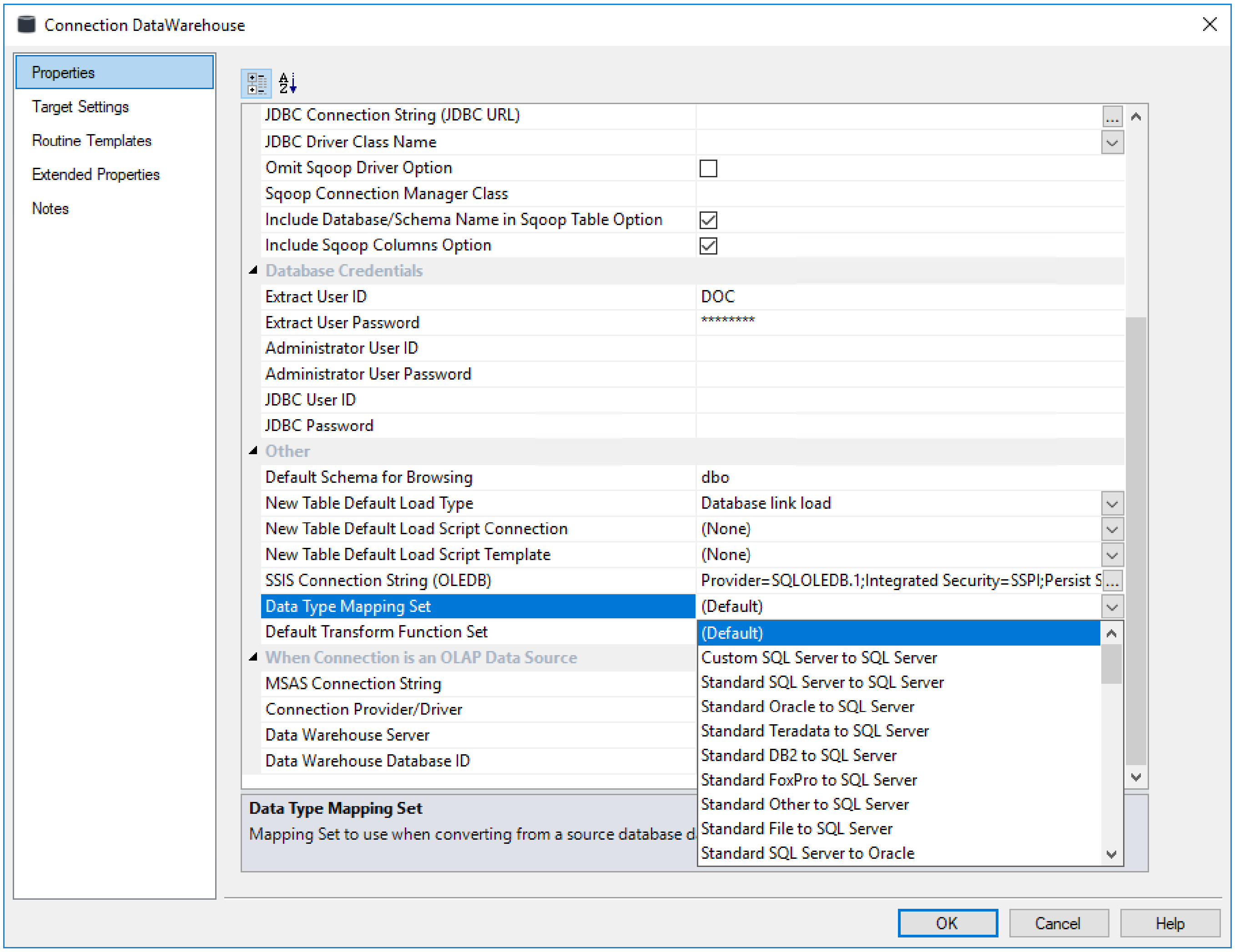
Using Data Type Mapping Sets

Data type mapping sets contain a list of mappings that are used when loading tables into the data warehouse.

Custom data type mapping sets give you the ability to automatically change the data type of any column or to add column transformations when dragging and dropping new Load tables. These mapping sets can be created, edited, deleted, imported and exported using the **Data Type Mappings** options from the **Tools** menu.

In the connection **Properties** screens, a drop-down list **Data Type Mapping Set** is displayed. This enables you to specify the default data type mapping set to use when loading tables into the data warehouse.

Data Warehouse Connection Properties

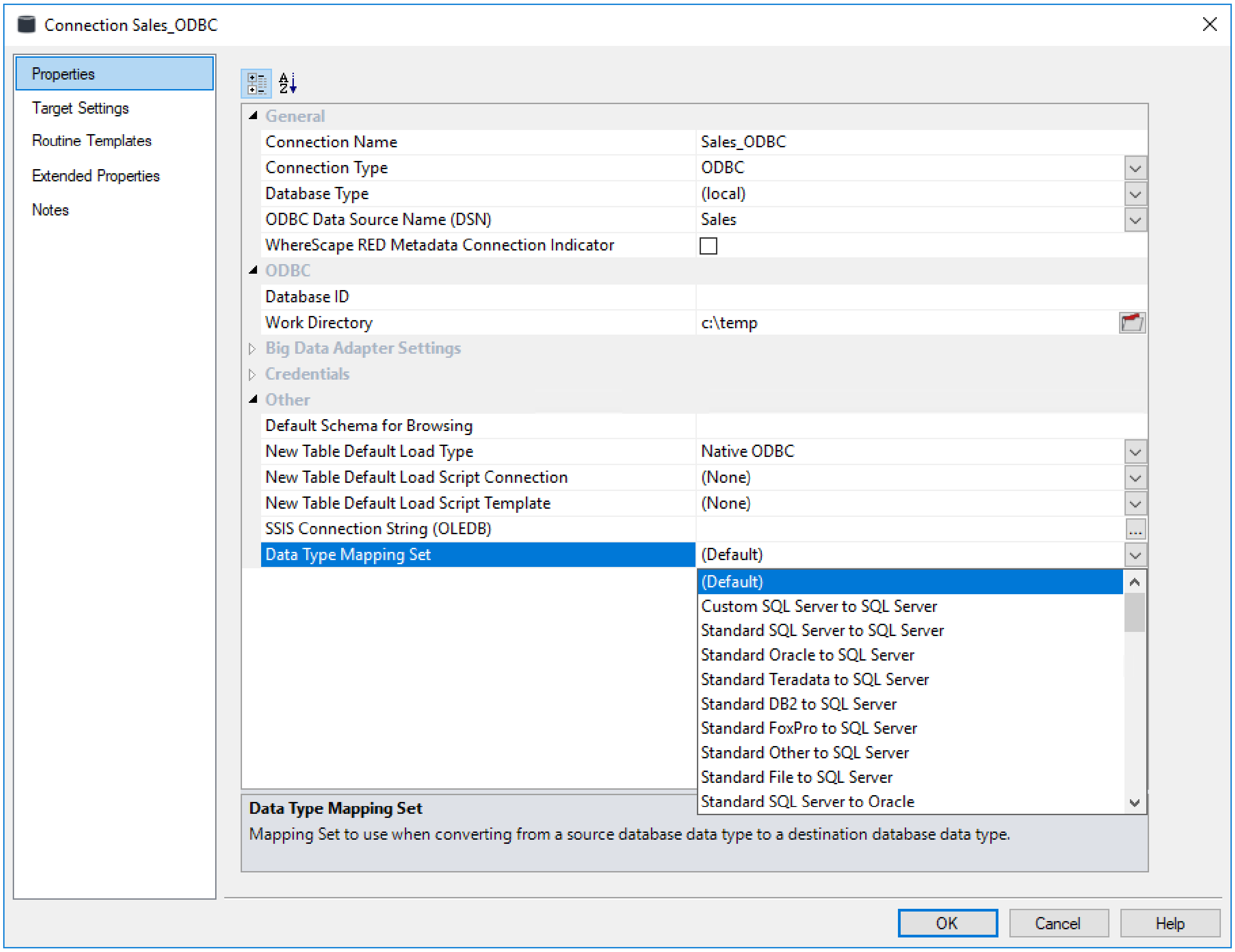


JDBC Connection String (JDBC URL)	
JDBC Driver Class Name	
Omit Sqoop Driver Option	☐
Sqoop Connection Manager Class	
Include Database/Schema Name in Sqoop Table Option	☑
Include Sqoop Columns Option	☑
Database Credentials	
Extract User ID	DOC
Extract User Password	********
Administrator User ID	
Administrator User Password	
JDBC User ID	
JDBC Password	
Other	
Default Schema for Browsing	dbo
New Table Default Load Type	Database link load
New Table Default Load Script Connection	(None)
New Table Default Load Script Template	(None)
SSIS Connection String (OLEDB)	Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist S
Data Type Mapping Set	(Default)
Default Transform Function Set	
When Connection is an OLAP Data Source	
MSAS Connection String	
Connection Provider/Driver	
Data Warehouse Server	
Data Warehouse Database ID	

(Default)
Custom SQL Server to SQL Server
Standard SQL Server to SQL Server
Standard Oracle to SQL Server
Standard Teradata to SQL Server
Standard DB2 to SQL Server
Standard FoxPro to SQL Server
Standard Other to SQL Server
Standard File to SQL Server
Standard SQL Server to Oracle

Data Type Mapping Set
Mapping Set to use when converting from a source database d

Non-Data Warehouse Connection Properties



Connection Sales_ODBC

General	
Connection Name	Sales_ODBC
Connection Type	ODBC
Database Type	(local)
ODBC Data Source Name (DSN)	Sales
WhereScape RED Metadata Connection Indicator	□
ODBC	
Database ID	
Work Directory	c:\temp
Big Data Adapter Settings	
Credentials	
Other	
Default Schema for Browsing	
New Table Default Load Type	Native ODBC
New Table Default Load Script Connection	(None)
New Table Default Load Script Template	(None)
SSIS Connection String (OLEDB)	
Data Type Mapping Set	(Default)

(Default)
Custom SQL Server to SQL Server
Standard SQL Server to SQL Server
Standard Oracle to SQL Server
Standard Teradata to SQL Server
Standard DB2 to SQL Server
Standard FoxPro to SQL Server
Standard Other to SQL Server
Standard File to SQL Server
Standard SQL Server to Oracle

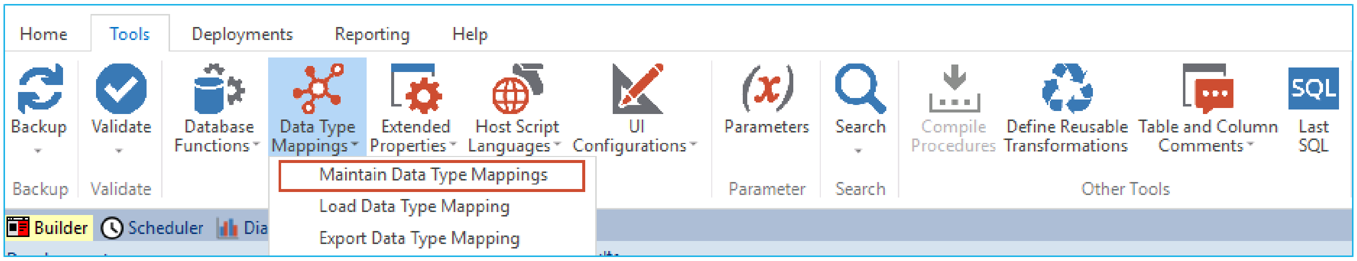
Data Type Mapping Set

Mapping Set to use when converting from a source database data type to a destination database data type.

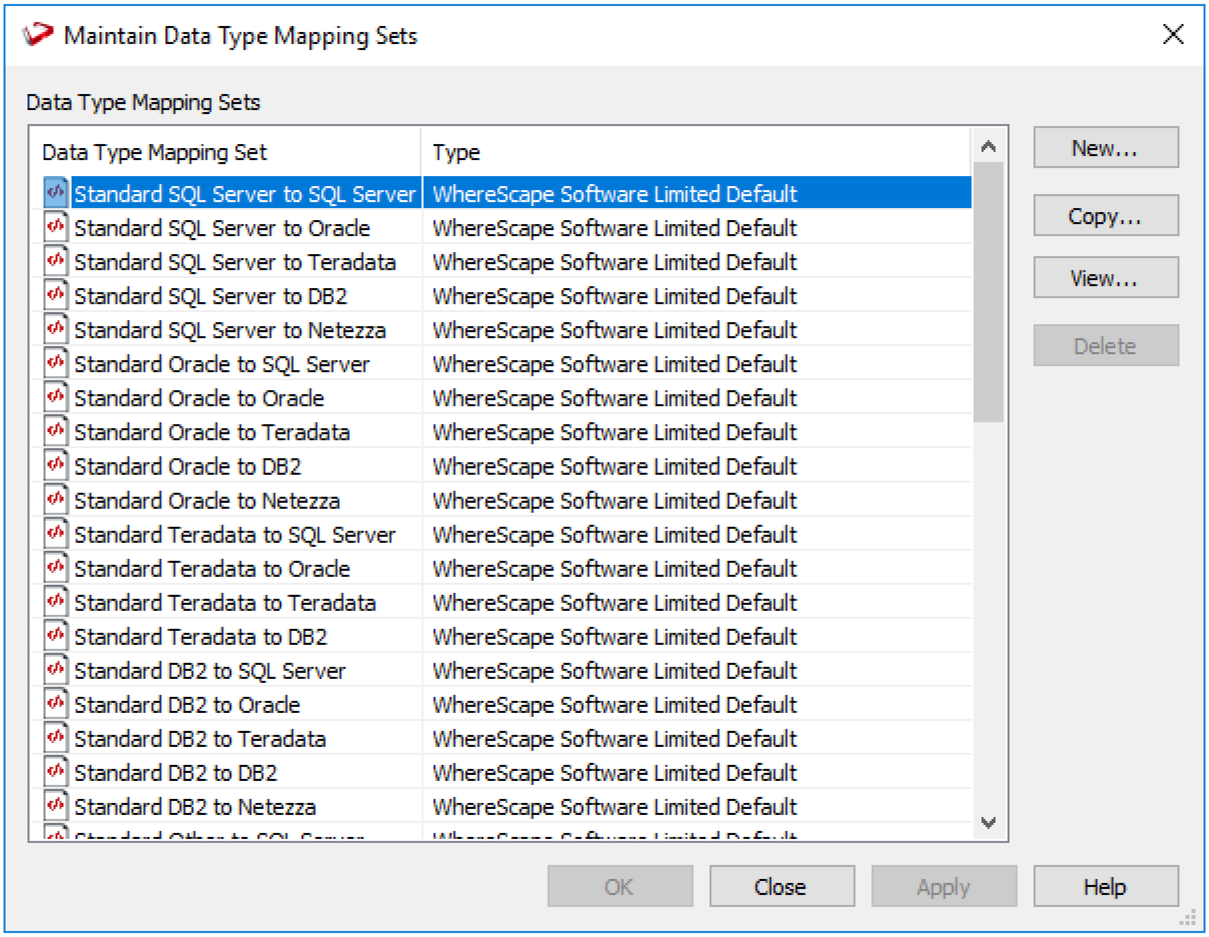
The succeeding sections describe how to manage data type mappings in RED.

Maintaining Data Type Mapping Sets

To maintain the data type mapping sets, select **Tools > Data Type Mappings > Maintain Data Type Mappings**.



The **Maintain Data Type Mapping Sets** window is displayed.



Data Type Mapping Set	Type
Standard SQL Server to SQL Server	WhereScape Software Limited Default
Standard SQL Server to Oracle	WhereScape Software Limited Default
Standard SQL Server to Teradata	WhereScape Software Limited Default
Standard SQL Server to DB2	WhereScape Software Limited Default
Standard SQL Server to Netezza	WhereScape Software Limited Default
Standard Oracle to SQL Server	WhereScape Software Limited Default
Standard Oracle to Oracle	WhereScape Software Limited Default
Standard Oracle to Teradata	WhereScape Software Limited Default
Standard Oracle to DB2	WhereScape Software Limited Default
Standard Oracle to Netezza	WhereScape Software Limited Default
Standard Teradata to SQL Server	WhereScape Software Limited Default
Standard Teradata to Oracle	WhereScape Software Limited Default
Standard Teradata to Teradata	WhereScape Software Limited Default
Standard Teradata to DB2	WhereScape Software Limited Default
Standard DB2 to SQL Server	WhereScape Software Limited Default
Standard DB2 to Oracle	WhereScape Software Limited Default
Standard DB2 to Teradata	WhereScape Software Limited Default
Standard DB2 to DB2	WhereScape Software Limited Default
Standard DB2 to Netezza	WhereScape Software Limited Default
Standard Other to SQL Server	WhereScape Software Limited Default

Select a data type mapping set from the **Data Type Mapping Sets** list.

In the list of standard files, only the files relevant to the database that you are working on is displayed.

To create a data type mapping set, refer to Creating a New Data Type Mapping Set

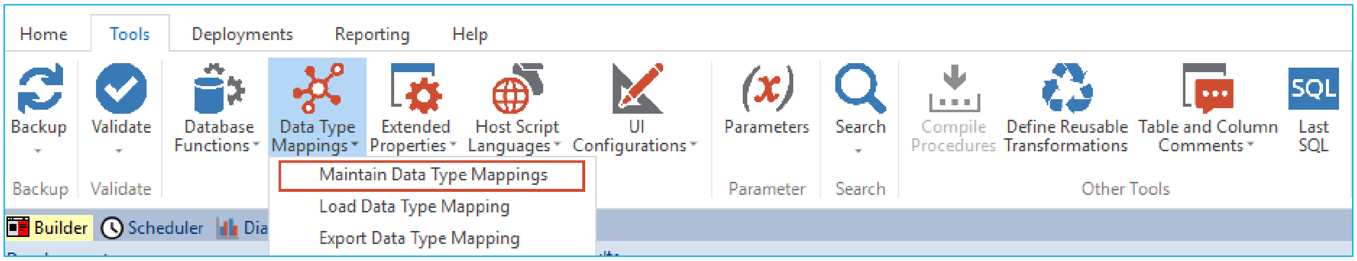
To copy a data type mapping set, refer to Copying a Data Type Mapping Set

To edit a data type mapping set, refer to Editing a Data Type Mapping Set

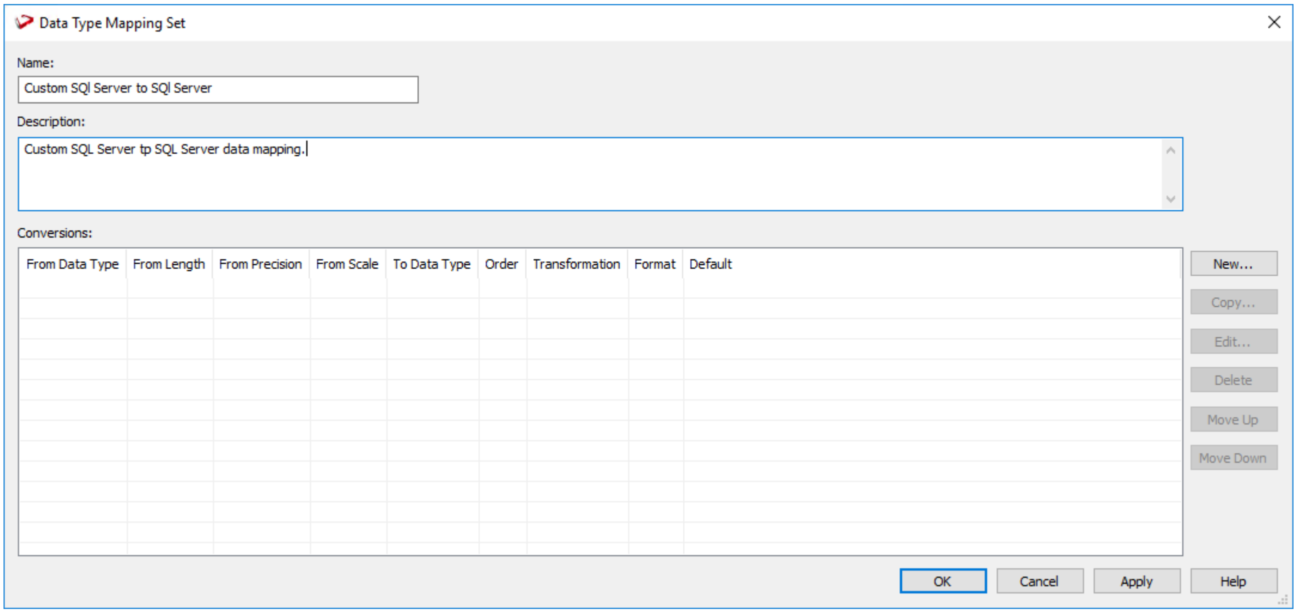
To delete a data type mapping set, refer to Deleting a Data Type Mapping Set

Creating a New Data Type Mapping Set

To create a new data type mapping set, select **Tools > Data Type Mappings > Maintain Data Type Mappings**.



1. Click the **New** button from the **Maintain Data Type Mapping Sets** window to enter the name and description of the new Mapping Set.



Data Type Mapping Set

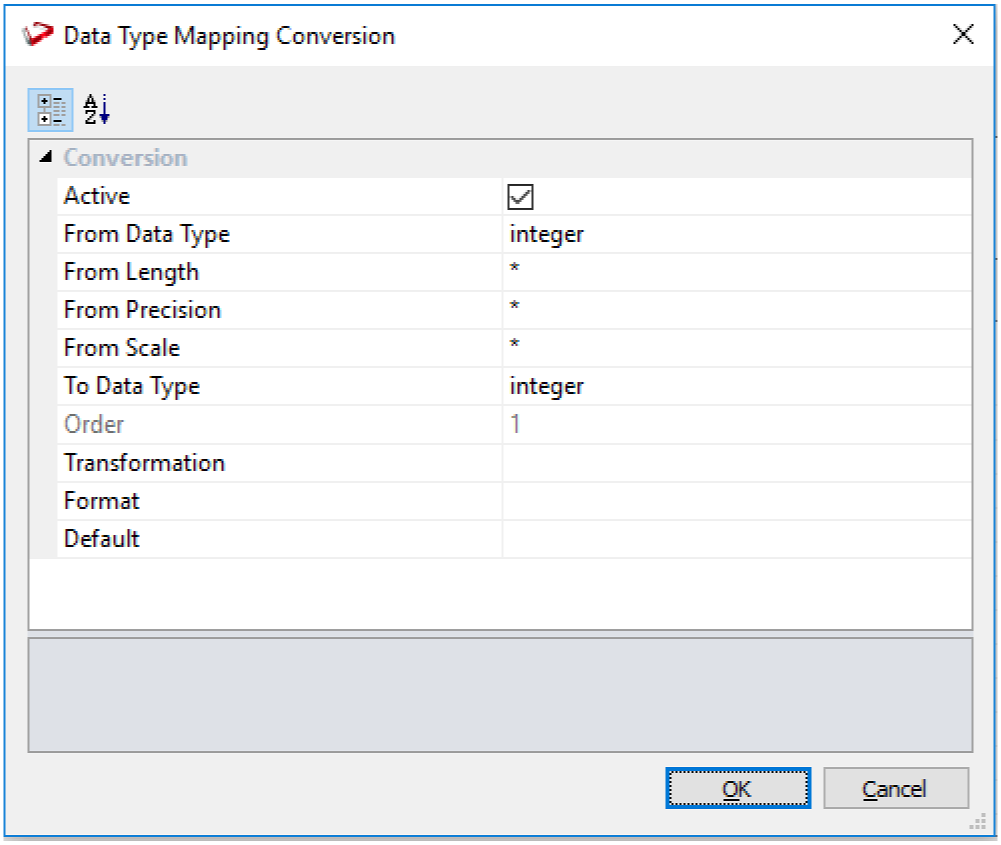
Name: Custom SQl Server to SQl Server

Description: Custom SQL Server tp SQL Server data mapping.

Conversions:

From Data Type	From Length	From Precision	From Scale	To Data Type	Order	Transformation	Format	Default

2. You can then enter the individual data type mapping conversion parameters by clicking the **New** button on the right side of the window.



Data Type Mapping Conversion

Conversion

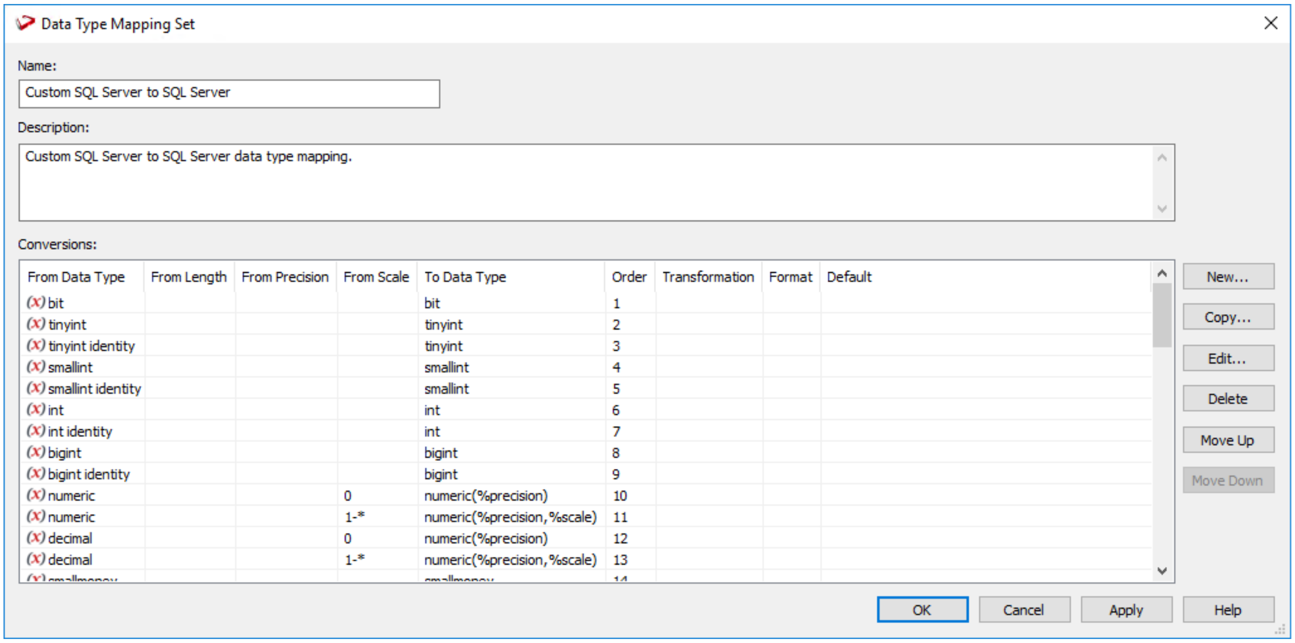
Active	☑
From Data Type	integer
From Length	*
From Precision	*
From Scale	*
To Data Type	integer
Order	1
Transformation	
Format	
Default	

The following describe the conversion parameters:

Parameter	Description
Active	Indicates if the conversion parameter can be used when searching for a match.
From Data Type	The source Data Type to that is used for the conversion to the Target Data Type. It can be any type that source RDBMS uses.
From Length	The length parameter to match. (the number of characters or the number of bytes used

Parameter	Description
	to store the number).
From Precision	The precision parameter to match. (the number of digits in a number).
From Scale	The scale parameter to match. (the number of digits to the right of the decimal point in a number).
To Data Type	The data type, including parameters, which is the output of this mapping. Available tokens are %data_type, %length, %precision and %scale. Refer to **Custom Data Type Mapping Set Examples** for details.
Order	The order of the parameter within the enclosing set of the conversion.
Transformation	The transformation to apply when the conversion is applied. See **Custom Data Type Mapping Set Examples** for details.
Format	The format that values using the data type will use. Typically used for numeric values.
Default	The default value to use for values that use the data type.

3. Click **OK** to save the parameter. Repeat to define more conversion parameters.



Name:
Custom SQL Server to SQL Server

Description:
Custom SQL Server to SQL Server data type mapping.

Conversions:

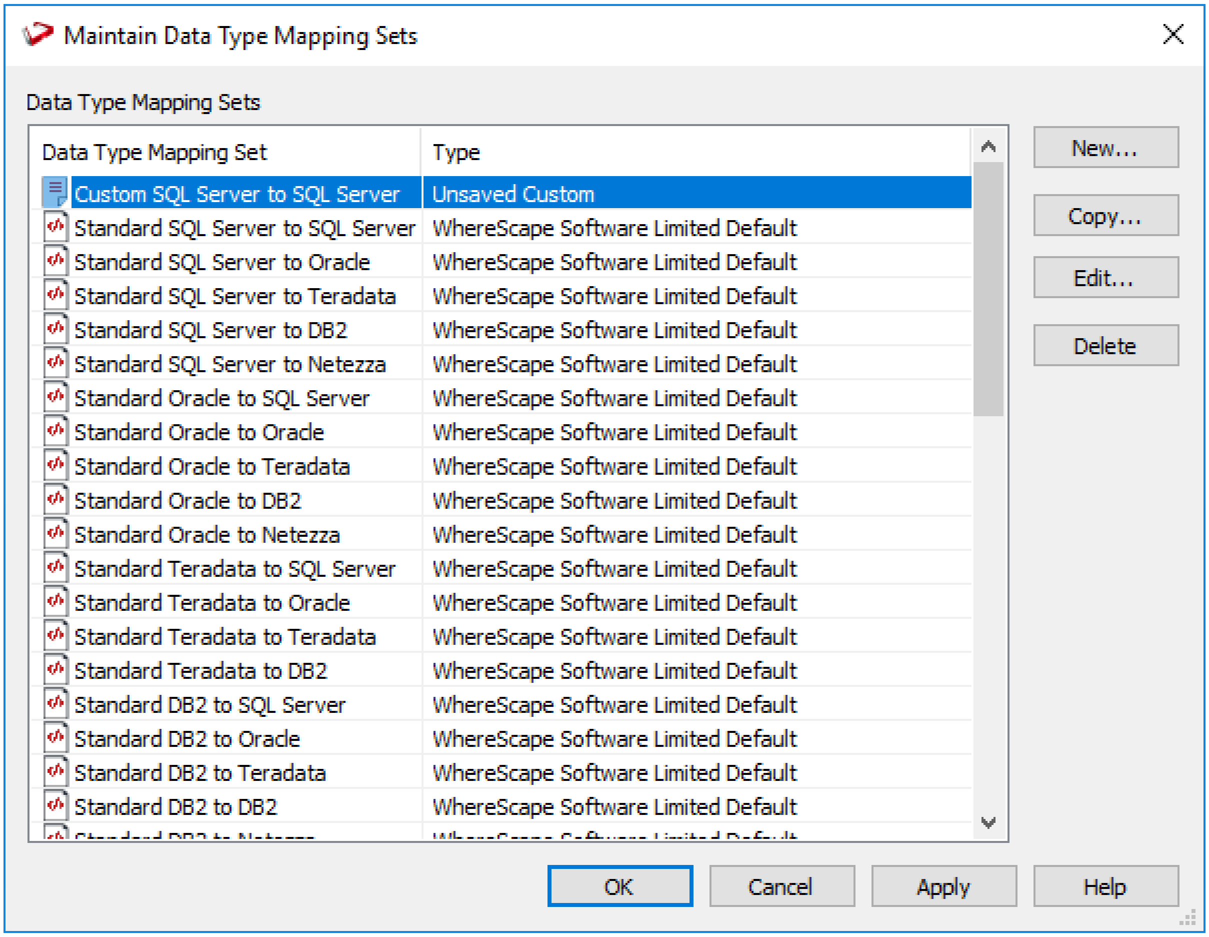
From Data Type	From Length	From Precision	From Scale	To Data Type	Order	Transformation	Format	Default
bit				bit	1			
tinyint				tinyint	2			
tinyint identity				tinyint	3			
smallint				smallint	4			
smallint identity				smallint	5			
int				int	6			
int identity				int	7			
bigint				bigint	8			
bigint identity				bigint	9			
numeric			0	numeric(%precision)	10			
numeric			1-*	numeric(%precision,%scale)	11			
decimal			0	numeric(%precision)	12			
decimal			1-*	numeric(%precision,%scale)	13			
smallmoney				smallmoney	14			

Notes

Use the buttons on the right side of the **Data Type Mapping Set** window to edit, copy or delete a selected parameter.To move a parameter up or down in the list, select it and then click the **Move Up** or **Move Down** button.

The order of the data type mapping conversion parameters is significant because when loading a table, the procedure checks the data type mappings from top to bottom and stops when a data type and its parameters are correctly matched. A blank parameter means that it will match to anything.

4. Click **OK** when you're done. The new data type mapping set is listed in the **Maintain Data Mapping Sets** window.

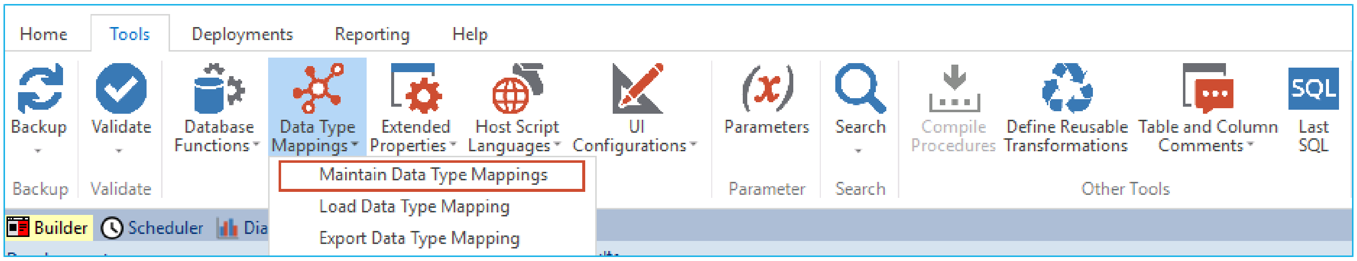


Data Type Mapping Set	Type
Custom SQL Server to SQL Server	Unsaved Custom
Standard SQL Server to SQL Server	WhereScape Software Limited Default
Standard SQL Server to Oracle	WhereScape Software Limited Default
Standard SQL Server to Teradata	WhereScape Software Limited Default
Standard SQL Server to DB2	WhereScape Software Limited Default
Standard SQL Server to Netezza	WhereScape Software Limited Default
Standard Oracle to SQL Server	WhereScape Software Limited Default
Standard Oracle to Oracle	WhereScape Software Limited Default
Standard Oracle to Teradata	WhereScape Software Limited Default
Standard Oracle to DB2	WhereScape Software Limited Default
Standard Oracle to Netezza	WhereScape Software Limited Default
Standard Teradata to SQL Server	WhereScape Software Limited Default
Standard Teradata to Oracle	WhereScape Software Limited Default
Standard Teradata to Teradata	WhereScape Software Limited Default
Standard Teradata to DB2	WhereScape Software Limited Default
Standard DB2 to SQL Server	WhereScape Software Limited Default
Standard DB2 to Oracle	WhereScape Software Limited Default
Standard DB2 to Teradata	WhereScape Software Limited Default
Standard DB2 to DB2	WhereScape Software Limited Default
Standard DB2 to Netezza	WhereScape Software Limited Default

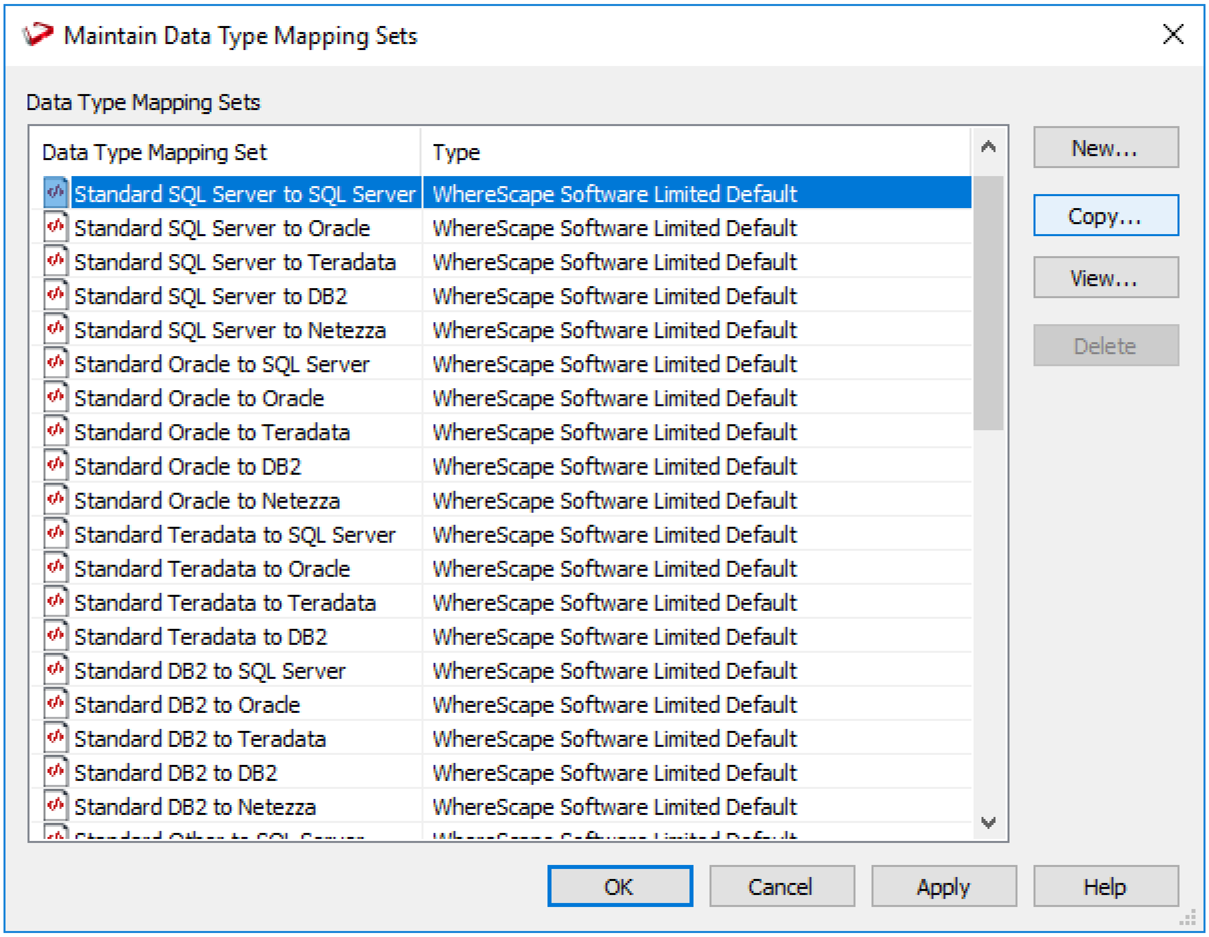
5. Click **OK** or **Apply** to save the new data type mapping set.

Copying a Data Type Mapping Set

To copy an existing data type mapping set, select **Tools > Data Type Mappings > Maintain Data Type Mappings**.



1. Select the Mapping Set to be copied from the list in the **Maintain Data Type Mapping Sets** window and then click the **Copy** button.

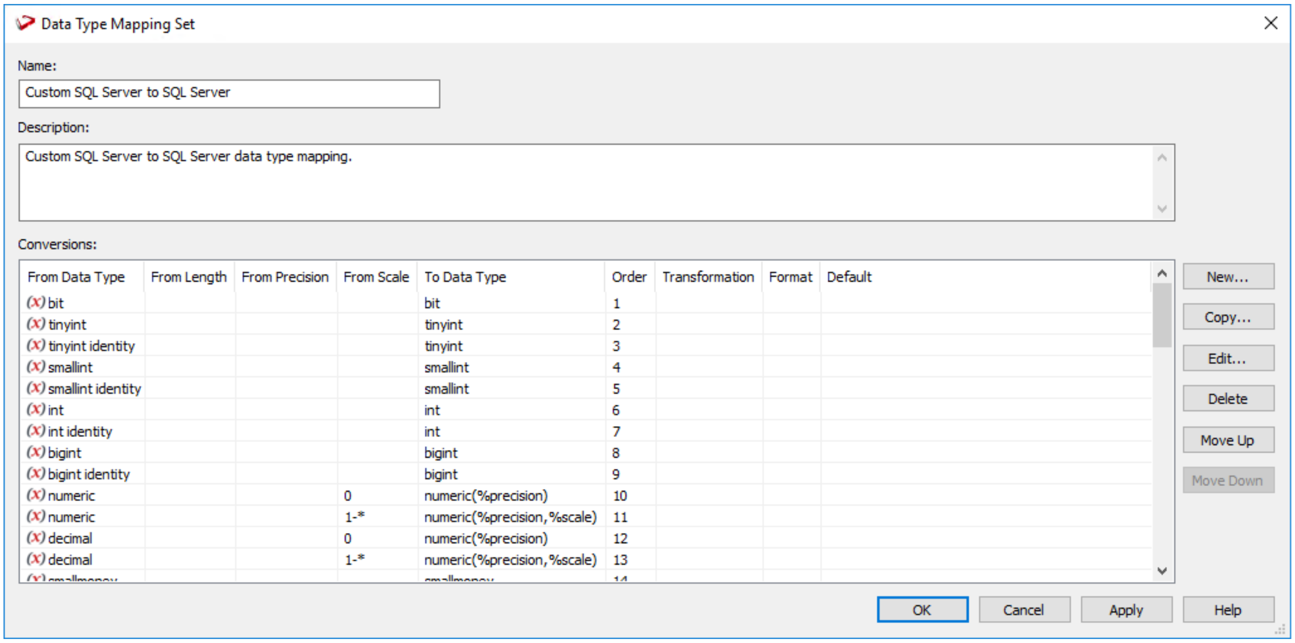


Maintain Data Type Mapping Sets

Data Type Mapping Sets

Data Type Mapping Set	Type
Standard SQL Server to SQL Server	WhereScape Software Limited Default
Standard SQL Server to Oracle	WhereScape Software Limited Default
Standard SQL Server to Teradata	WhereScape Software Limited Default
Standard SQL Server to DB2	WhereScape Software Limited Default
Standard SQL Server to Netezza	WhereScape Software Limited Default
Standard Oracle to SQL Server	WhereScape Software Limited Default
Standard Oracle to Oracle	WhereScape Software Limited Default
Standard Oracle to Teradata	WhereScape Software Limited Default
Standard Oracle to DB2	WhereScape Software Limited Default
Standard Oracle to Netezza	WhereScape Software Limited Default
Standard Teradata to SQL Server	WhereScape Software Limited Default
Standard Teradata to Oracle	WhereScape Software Limited Default
Standard Teradata to Teradata	WhereScape Software Limited Default
Standard Teradata to DB2	WhereScape Software Limited Default
Standard DB2 to SQL Server	WhereScape Software Limited Default
Standard DB2 to Oracle	WhereScape Software Limited Default
Standard DB2 to Teradata	WhereScape Software Limited Default
Standard DB2 to DB2	WhereScape Software Limited Default
Standard DB2 to Netezza	WhereScape Software Limited Default

2. Change the name and description of the new Mapping Set as required.



Data Type Mapping Set

Name: Custom SQL Server to SQL Server

Description: Custom SQL Server to SQL Server data type mapping.

Conversions:

From Data Type	From Length	From Precision	From Scale	To Data Type	Order	Transformation	Format	Default
(x) bit				bit	1			
(x) tinyint				tinyint	2			
(x) tinyint identity				tinyint	3			
(x) smallint				smallint	4			
(x) smallint identity				smallint	5			
(x) int				int	6			
(x) int identity				int	7			
(x) bigint				bigint	8			
(x) bigint identity				bigint	9			
(x) numeric			0	numeric(%precision)	10			
(x) numeric			1-*	numeric(%precision,%scale)	11			
(x) decimal			0	numeric(%precision)	12			
(x) decimal			1-*	numeric(%precision,%scale)	13			

3. Use the buttons on the right side of the **Data Type Mapping Set** window to add new conversion parameters, edit, copy or delete a selected parameter.

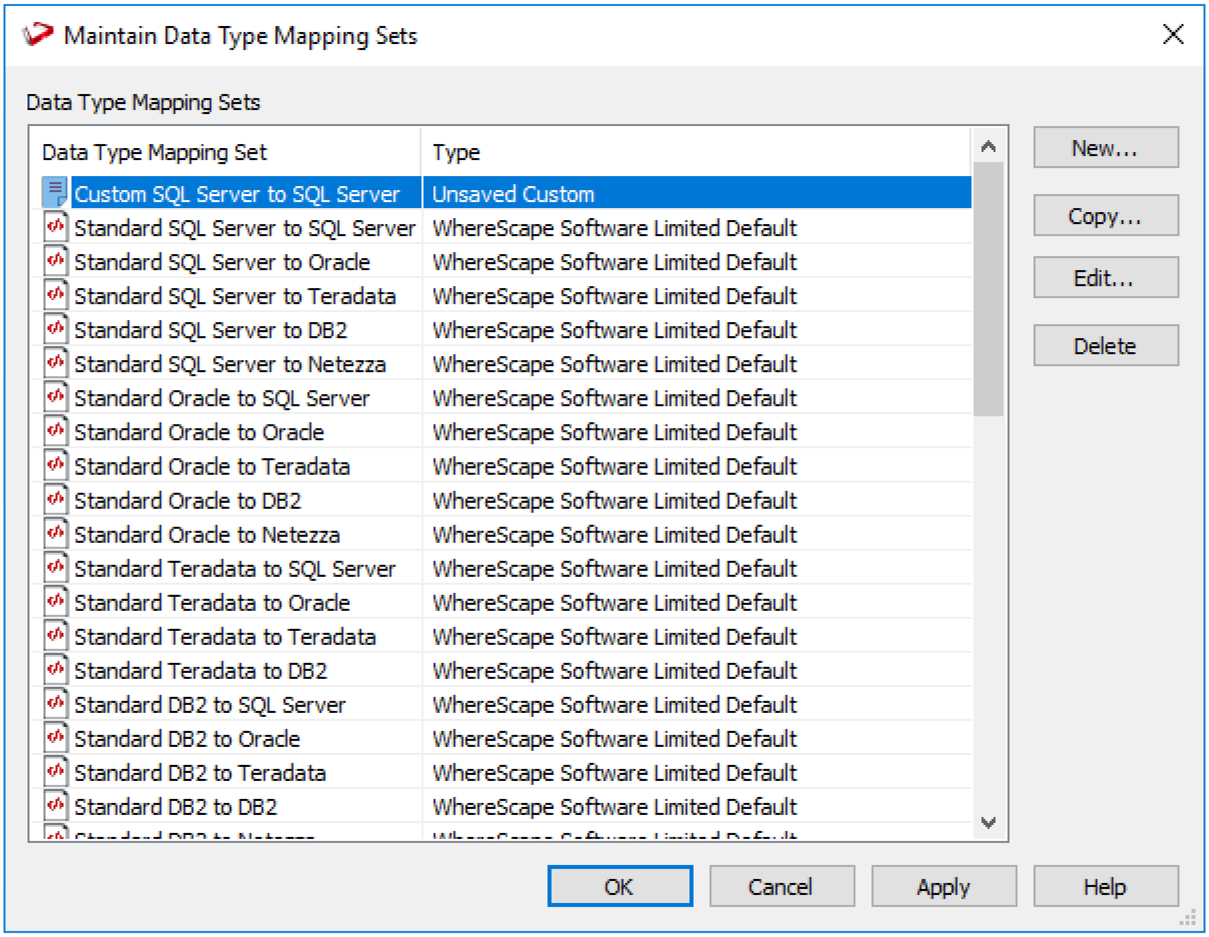
Notes

To move a parameter up or down in the list, select it and then click the **Move Up** or **Move Down** button. The order

Notes

of the data type mapping conversion parameters is significant because when loading a table, the procedure checks the data type mappings from top to bottom and stops when a data type and its parameters are correctly matched. A blank parameter means that it will match to anything.

4. Click **OK** when you're done. The new data type mapping set is listed in the **Maintain Data Mapping Sets** window.

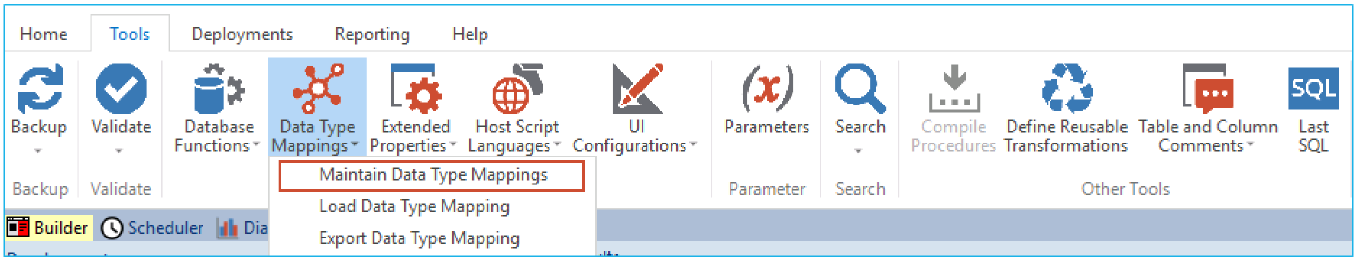


Data Type Mapping Set	Type
Custom SQL Server to SQL Server	Unsaved Custom
Standard SQL Server to SQL Server	WhereScape Software Limited Default
Standard SQL Server to Oracle	WhereScape Software Limited Default
Standard SQL Server to Teradata	WhereScape Software Limited Default
Standard SQL Server to DB2	WhereScape Software Limited Default
Standard SQL Server to Netezza	WhereScape Software Limited Default
Standard Oracle to SQL Server	WhereScape Software Limited Default
Standard Oracle to Oracle	WhereScape Software Limited Default
Standard Oracle to Teradata	WhereScape Software Limited Default
Standard Oracle to DB2	WhereScape Software Limited Default
Standard Oracle to Netezza	WhereScape Software Limited Default
Standard Teradata to SQL Server	WhereScape Software Limited Default
Standard Teradata to Oracle	WhereScape Software Limited Default
Standard Teradata to Teradata	WhereScape Software Limited Default
Standard Teradata to DB2	WhereScape Software Limited Default
Standard DB2 to SQL Server	WhereScape Software Limited Default
Standard DB2 to Oracle	WhereScape Software Limited Default
Standard DB2 to Teradata	WhereScape Software Limited Default
Standard DB2 to DB2	WhereScape Software Limited Default
Standard DB2 to Netezza	WhereScape Software Limited Default

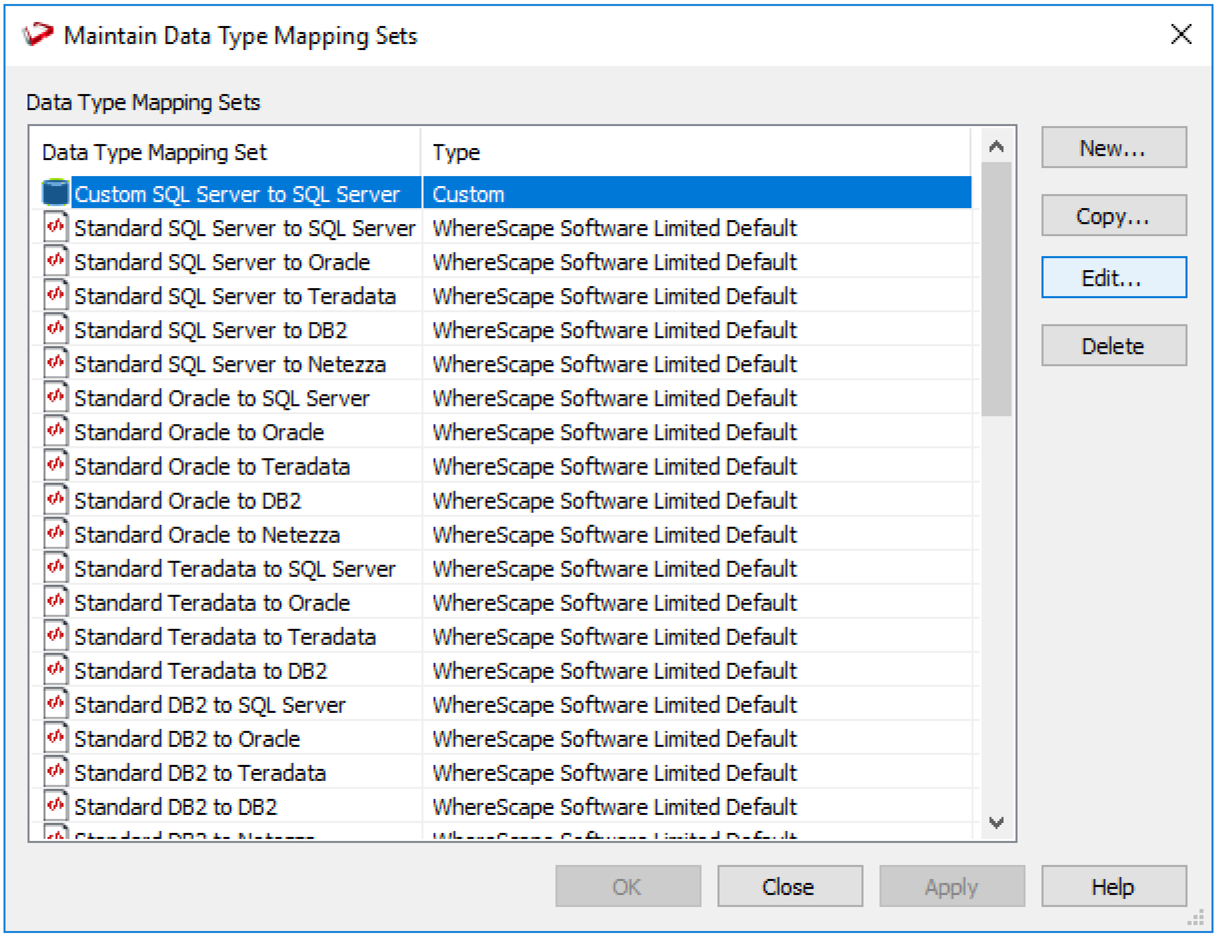
5. Click **OK** or **Apply** to save the new data type mapping set.

Editing a Data Type Mapping Set

To edit a data type mapping set, select **Tools > Data Type Mappings > Maintain Data Type Mappings**.



1. Select the Mapping Set to be modified from the list in the **Maintain Data Type Mapping Sets** window and then click the **Edit** button.



Maintain Data Type Mapping Sets

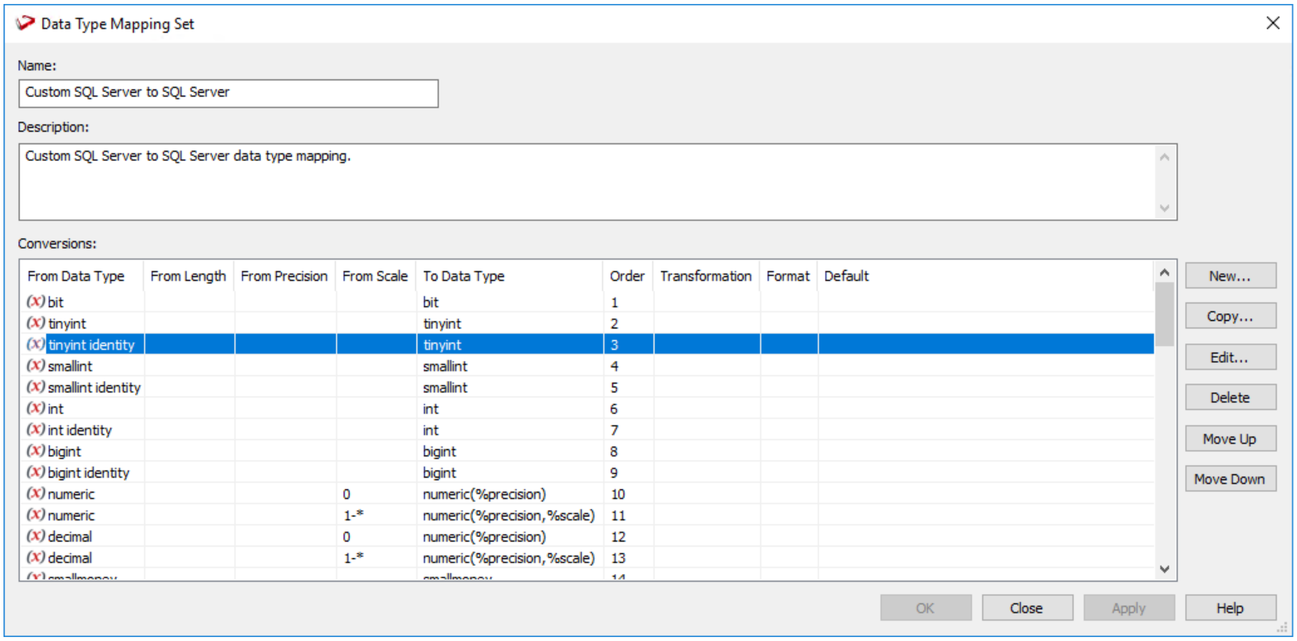
Data Type Mapping Sets

Data Type Mapping Set	Type
Custom SQL Server to SQL Server	Custom
Standard SQL Server to SQL Server	WhereScape Software Limited Default
Standard SQL Server to Oracle	WhereScape Software Limited Default
Standard SQL Server to Teradata	WhereScape Software Limited Default
Standard SQL Server to DB2	WhereScape Software Limited Default
Standard SQL Server to Netezza	WhereScape Software Limited Default
Standard Oracle to SQL Server	WhereScape Software Limited Default
Standard Oracle to Oracle	WhereScape Software Limited Default
Standard Oracle to Teradata	WhereScape Software Limited Default
Standard Oracle to DB2	WhereScape Software Limited Default
Standard Oracle to Netezza	WhereScape Software Limited Default
Standard Teradata to SQL Server	WhereScape Software Limited Default
Standard Teradata to Oracle	WhereScape Software Limited Default
Standard Teradata to Teradata	WhereScape Software Limited Default
Standard Teradata to DB2	WhereScape Software Limited Default
Standard DB2 to SQL Server	WhereScape Software Limited Default
Standard DB2 to Oracle	WhereScape Software Limited Default
Standard DB2 to Teradata	WhereScape Software Limited Default
Standard DB2 to DB2	WhereScape Software Limited Default
Standard DB2 to Netezza	WhereScape Software Limited Default

Notes

Standard Data Type Mapping sets are not editable, they can only be viewed.

2. In the **Data Type Mapping Set** window, select the parameter you want to change from the **Conversions** pane. Use the scroll bar to see all the defined parameters.



Data Type Mapping Set

Name:
Custom SQL Server to SQL Server

Description:
Custom SQL Server to SQL Server data type mapping.

Conversions:

From Data Type	From Length	From Precision	From Scale	To Data Type	Order	Transformation	Format	Default
(x) bit				bit	1			
(x) tinyint				tinyint	2			
(x) tinyint identity				tinyint	3			
(x) smallint				smallint	4			
(x) smallint identity				smallint	5			
(x) int				int	6			
(x) int identity				int	7			
(x) bigint				bigint	8			
(x) bigint identity				bigint	9			
(x) numeric			0	numeric(%precision)	10			
(x) numeric			1-*	numeric(%precision,%scale)	11			
(x) decimal			0	numeric(%precision)	12			
(x) decimal			1-*	numeric(%precision,%scale)	13			
(x) smallmoney				smallmoney	14			

3. Use the buttons on the right side of the **Data Type Mapping Set** window to add new conversion parameters, edit, copy or delete a selected parameter. These buttons are not available for standard mapping sets. Only **user defined** mapping sets are editable.

Notes

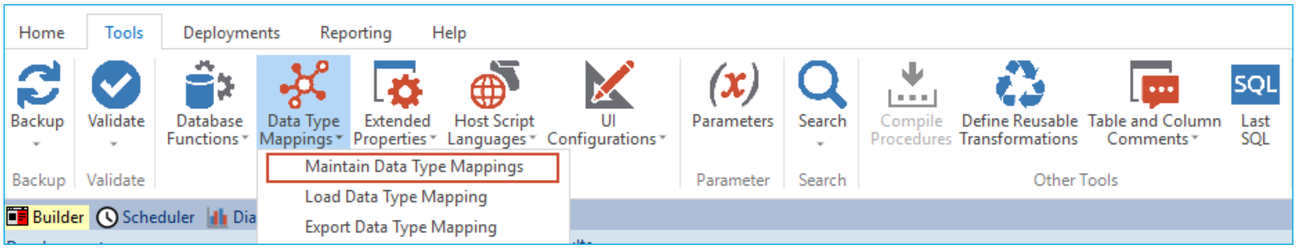
To move a parameter up or down in the list, select it and then click the **Move Up** or **Move Down** button. The order of the data type mapping conversion parameters is significant because when loading a table, the procedure checks the data type mappings from top to bottom and stops when a data type and its parameters are correctly matched. A blank parameter means that it will match to anything.

4. Click **OK** when you're done.

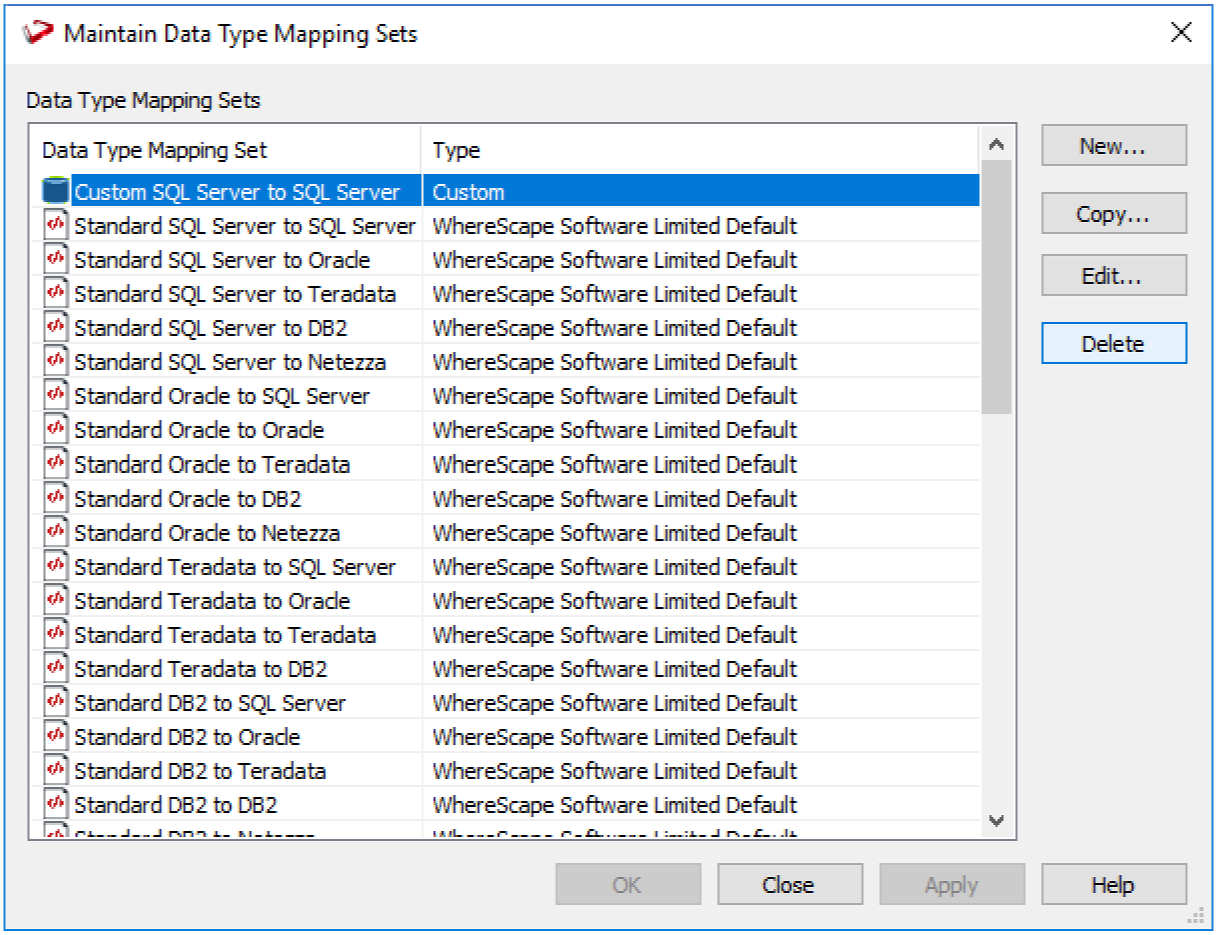
See previous sections for details on adding or copying a new data type mapping set.

Deleting a Data Type Mapping Set

1. To delete a data type mapping set, select **Tools > Data Type Mappings > Maintain Data Type Mappings**.



2. Select the Mapping Set to be deleted and then click the **Delete** button.



Maintain Data Type Mapping Sets

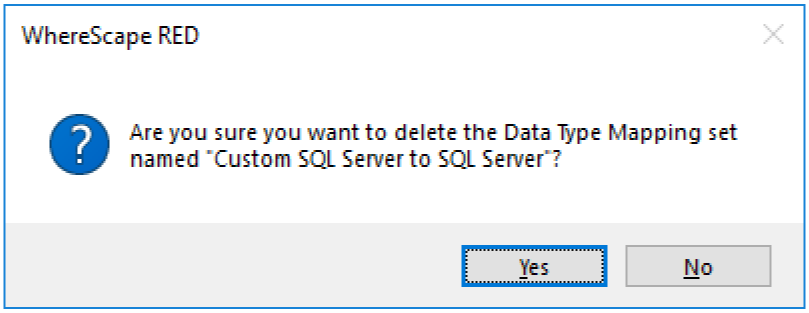
Data Type Mapping Sets

Data Type Mapping Set	Type
Custom SQL Server to SQL Server	Custom
Standard SQL Server to SQL Server	WhereScape Software Limited Default
Standard SQL Server to Oracle	WhereScape Software Limited Default
Standard SQL Server to Teradata	WhereScape Software Limited Default
Standard SQL Server to DB2	WhereScape Software Limited Default
Standard SQL Server to Netezza	WhereScape Software Limited Default
Standard Oracle to SQL Server	WhereScape Software Limited Default
Standard Oracle to Oracle	WhereScape Software Limited Default
Standard Oracle to Teradata	WhereScape Software Limited Default
Standard Oracle to DB2	WhereScape Software Limited Default
Standard Oracle to Netezza	WhereScape Software Limited Default
Standard Teradata to SQL Server	WhereScape Software Limited Default
Standard Teradata to Oracle	WhereScape Software Limited Default
Standard Teradata to Teradata	WhereScape Software Limited Default
Standard Teradata to DB2	WhereScape Software Limited Default
Standard DB2 to SQL Server	WhereScape Software Limited Default
Standard DB2 to Oracle	WhereScape Software Limited Default
Standard DB2 to Teradata	WhereScape Software Limited Default
Standard DB2 to DB2	WhereScape Software Limited Default
Standard DB2 to Netezza	WhereScape Software Limited Default

Notes

Standard data type mapping sets cannot be deleted. The **Delete** button is only enabled for non standard data type mapping sets.

3. A prompt is displayed asking you to confirm the action. Click **Yes** to proceed or **No** to cancel.



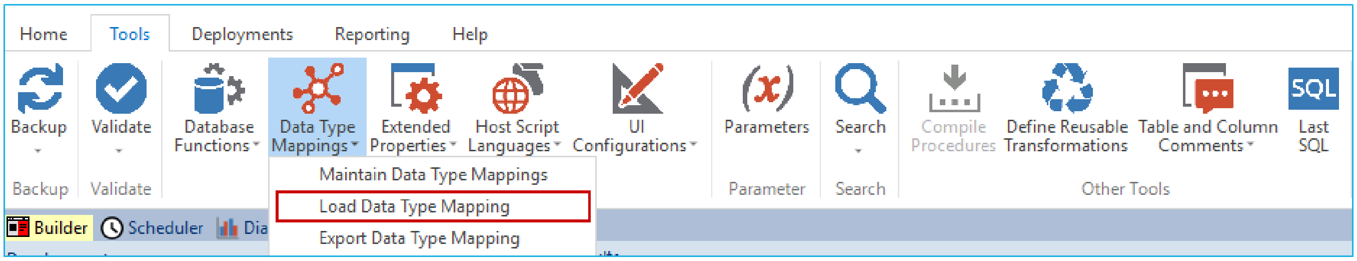
WhereScape RED

Are you sure you want to delete the Data Type Mapping set named "Custom SQL Server to SQL Server"?

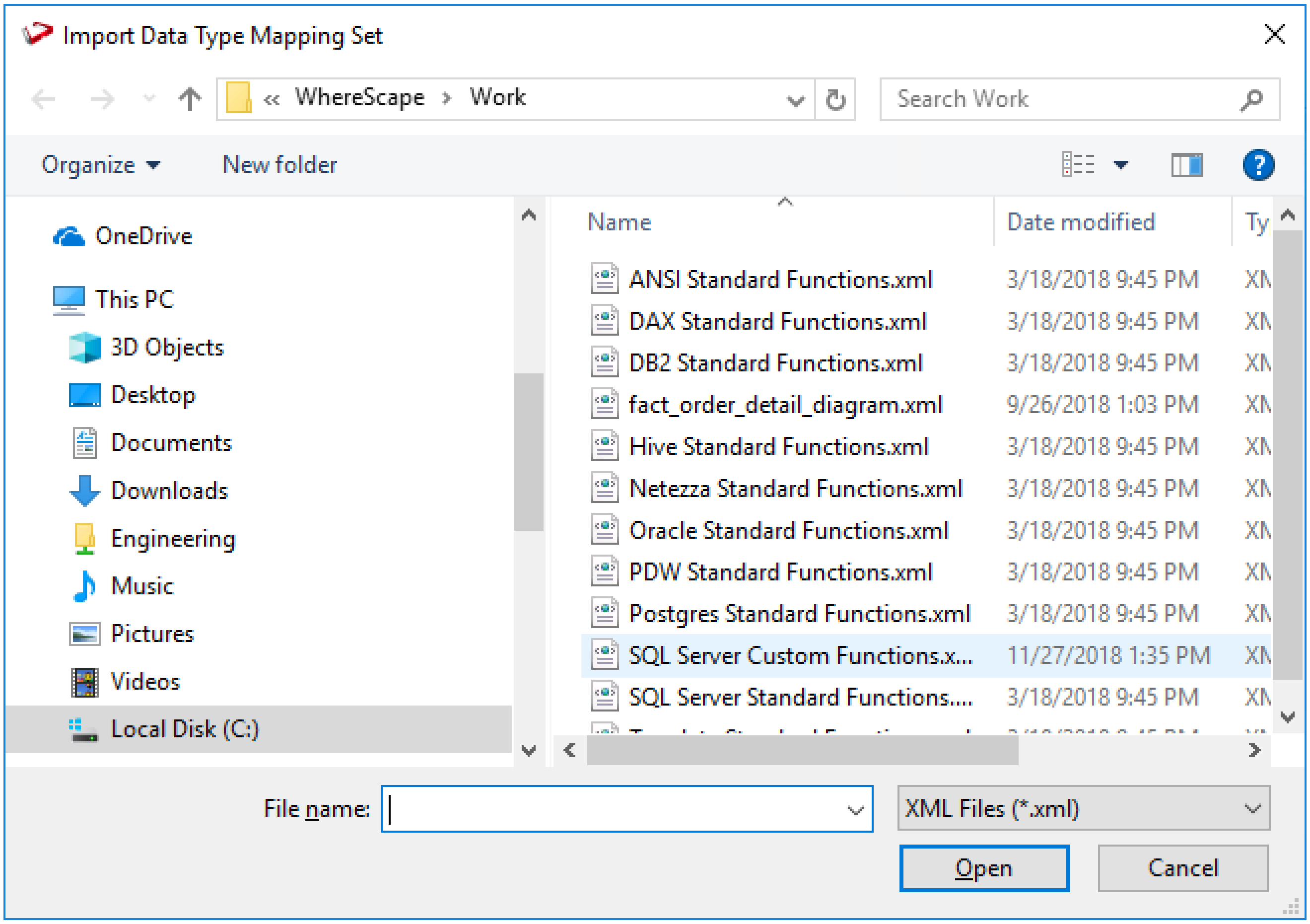
Loading Custom Data Type Mapping Sets

The **Load Custom Data Type Mapping Set** menu option enables you to load a custom data type mapping set from an XML file into the RED metadata repository.

To load a data type mapping set, select **Tools > Data Type Mappings > Load Custom Data Type Mapping Set**.



The following window is displayed. Select the xml file to load the data type mapping set. By default, RED expects the xml files to be in the **ProgramData\WhereScape\Work** directory.

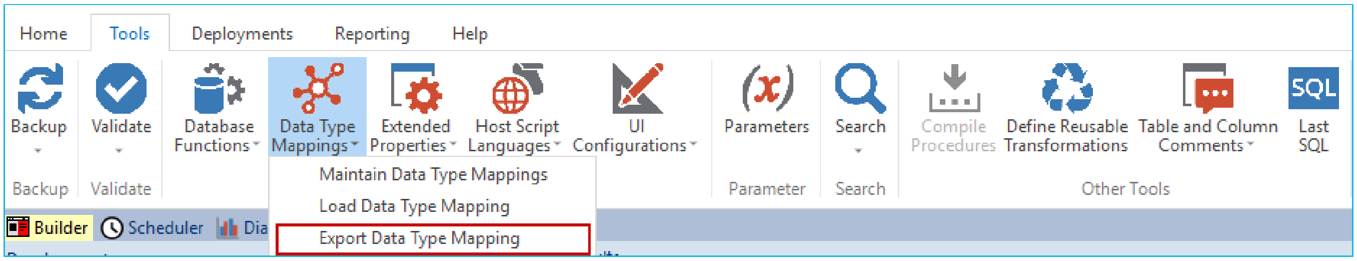


Name	Date modified	Ty
ANSI Standard Functions.xml	3/18/2018 9:45 PM	XN
DAX Standard Functions.xml	3/18/2018 9:45 PM	XN
DB2 Standard Functions.xml	3/18/2018 9:45 PM	XN
fact_order_detail_diagram.xml	9/26/2018 1:03 PM	XN
Hive Standard Functions.xml	3/18/2018 9:45 PM	XN
Netezza Standard Functions.xml	3/18/2018 9:45 PM	XN
Oracle Standard Functions.xml	3/18/2018 9:45 PM	XN
PDW Standard Functions.xml	3/18/2018 9:45 PM	XN
Postgres Standard Functions.xml	3/18/2018 9:45 PM	XN
SQL Server Custom Functions.x...	11/27/2018 1:35 PM	XN
SQL Server Standard Functions....	3/18/2018 9:45 PM	XN

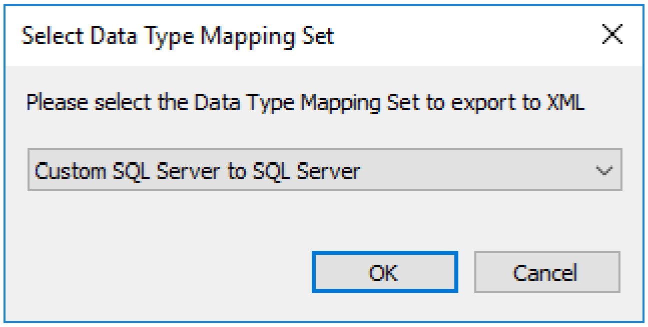
Exporting Custom Data Type Mapping Sets

The **Export Custom Data Type Mapping Set** menu option enables you to export a custom data type mapping set from the RED metadata repository to an XML file.

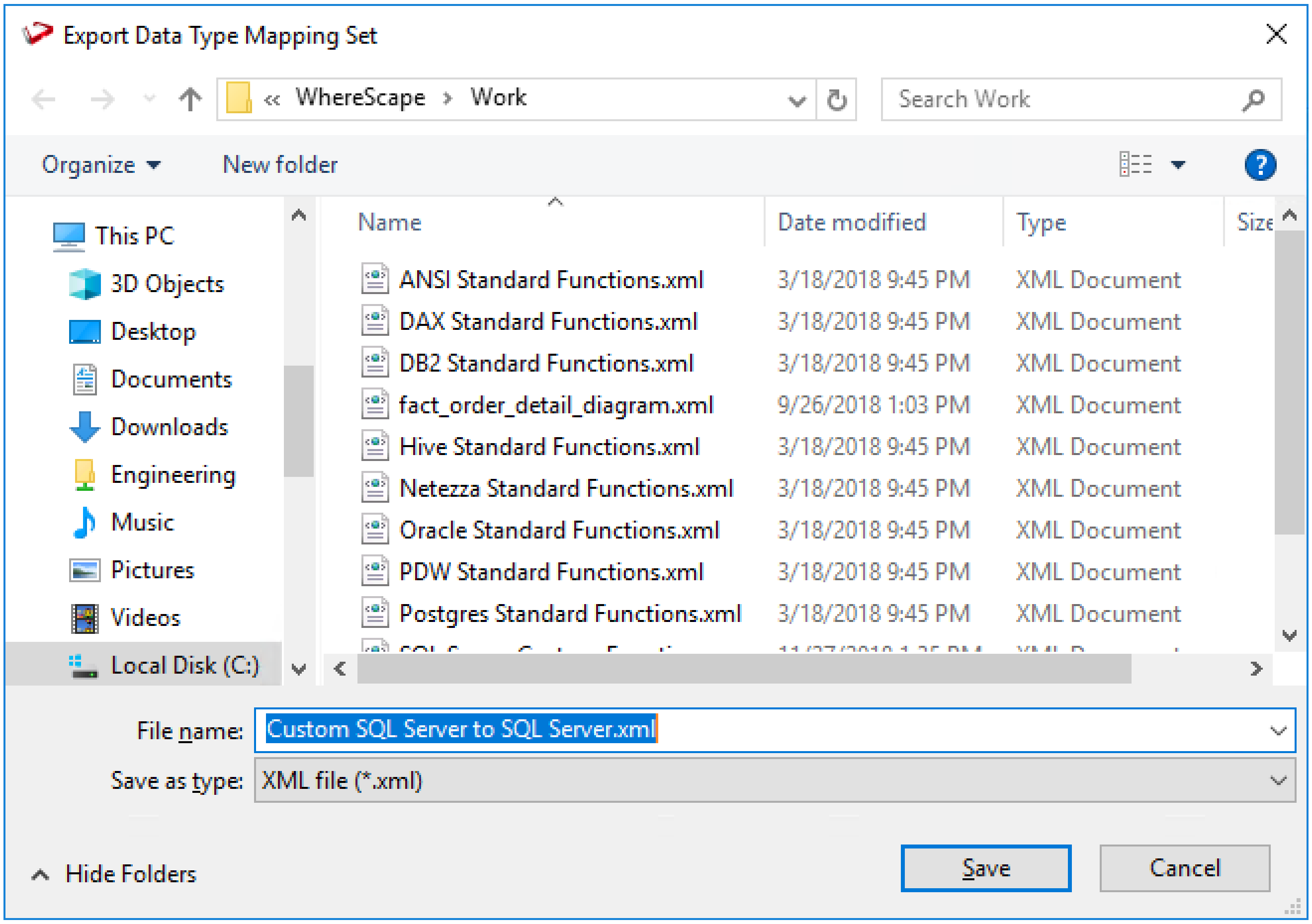
To export a data type mapping set, select **Tools > Data Type Mappings > Export Custom Data Type Mapping Set**.



Select the data type mapping set to export from the drop-down list. Click **OK** to proceed.


Select Data Type Mapping Set
Please select the Data Type Mapping Set to export to XML
Custom SQL Server to SQL Server
OK
Cancel

By default, RED exports the xml file to the **ProgramData\WhereScape\Work** directory, but this can be changed. Change the File name if necessary and click **Save**.


Export Data Type Mapping Set
« WhereScape › Work
Search Work
Organize ▾
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Engineering
Music
Pictures
Videos
Local Disk (C:)
Name
Date modified
Type
Size
ANSI Standard Functions.xml 3/18/2018 9:45 PM XML Document
DAX Standard Functions.xml 3/18/2018 9:45 PM XML Document
DB2 Standard Functions.xml 3/18/2018 9:45 PM XML Document
fact_order_detail_diagram.xml 9/26/2018 1:03 PM XML Document
Hive Standard Functions.xml 3/18/2018 9:45 PM XML Document
Netezza Standard Functions.xml 3/18/2018 9:45 PM XML Document
Oracle Standard Functions.xml 3/18/2018 9:45 PM XML Document
PDW Standard Functions.xml 3/18/2018 9:45 PM XML Document
Postgres Standard Functions.xml 3/18/2018 9:45 PM XML Document
File name: Custom SQL Server to SQL Server.xml
Save as type: XML file (*.xml)
Hide Folders
Save
Cancel

Custom Data Type Mapping Set Examples

WhereScape RED enables you to create **Custom** Data Type Mapping Sets. This gives you the ability to automatically change the data type of any column or to add column transformations, when dragging and dropping new Load tables.

The examples in this topic demonstrate how **Custom** Data Type Mapping Sets can be configured, using the following variables:

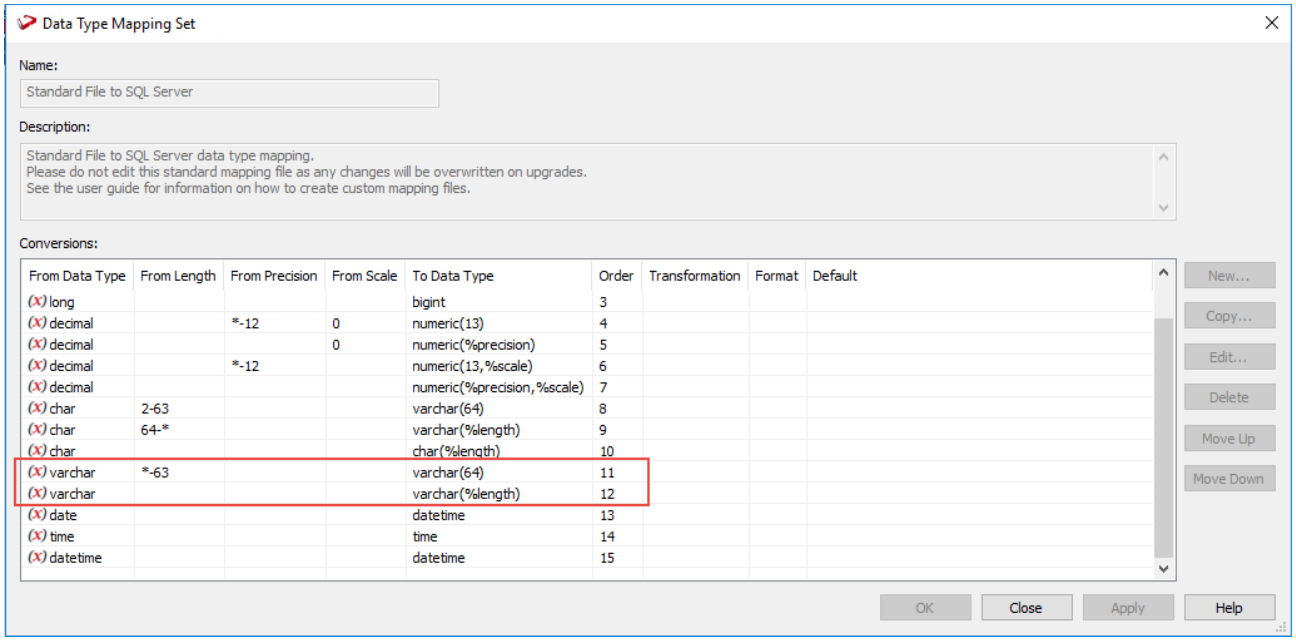
- %length
- %scale
- %precision

- %table_name
- %column_name
- %format
- %prompt

%length

In the example below, when converting a varchar in a standard file to SQL Server format, we follow the following steps in the given order:

- If the varchar is of a length less than or equal to 63, the **data type** will become varchar(64).
- If the first step was NOT applied, i.e. the varchar is of a length greater than 63, then the **data type** will become varchar(%length); where we substitute the length for the variable '%length'. Thus if the varchar is of length 64 then the resulting data type will be varchar(64), but if the varchar is of length 123 then the resulting data type will be varchar(123).



Name:
Standard File to SQL Server

Description:
Standard File to SQL Server data type mapping.
Please do not edit this standard mapping file as any changes will be overwritten on upgrades.
See the user guide for information on how to create custom mapping files.

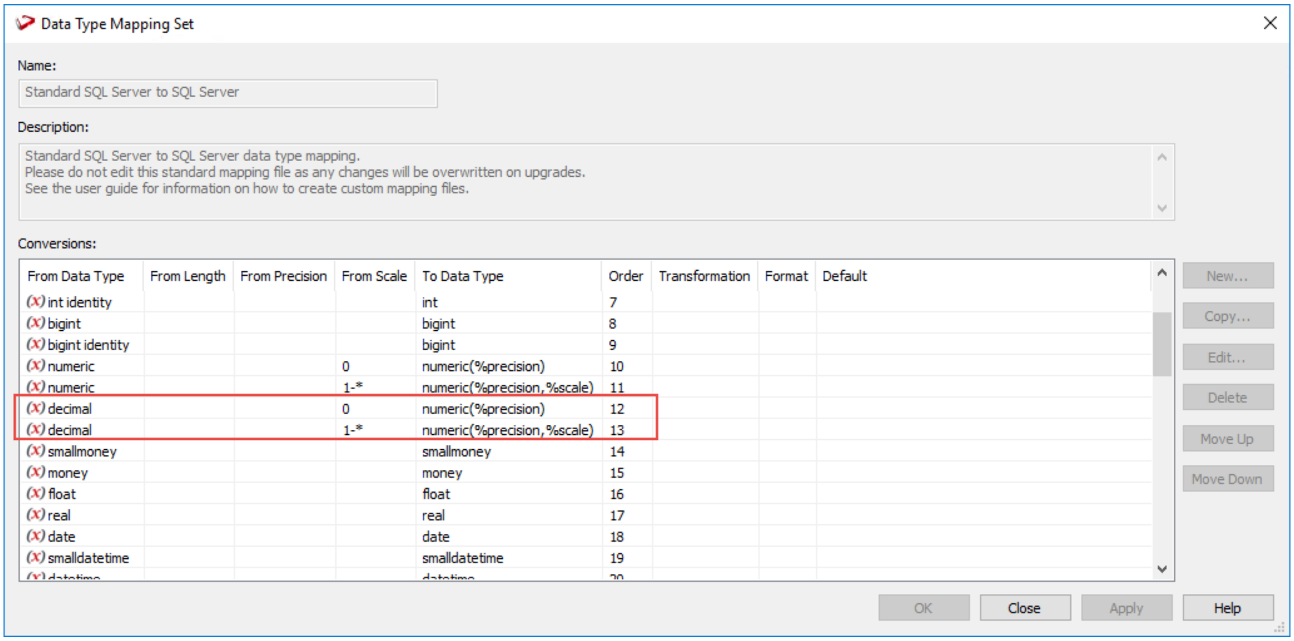
Conversions:

From Data Type	From Length	From Precision	From Scale	To Data Type	Order	Transformation	Format	Default
long				bigint	3			
decimal		*-12	0	numeric(13)	4			
decimal			0	numeric(%precision)	5			
decimal		*-12		numeric(13,%scale)	6			
decimal				numeric(%precision,%scale)	7			
char	2-63			varchar(64)	8			
char	64-*			varchar(%length)	9			
char				char(%length)	10			
varchar	*-63			varchar(64)	11			
varchar				varchar(%length)	12			
date				datetime	13			
time				time	14			
datetime				datetime	15			

%scale

In the example below, when converting a decimal in SQL Server to SQL Server format, the following steps are done in the given order:

- If the decimal has a scale of zero, the **data type** will become numeric(%precision); where we substitute the number of digits in the number for the variable '%precision'. Thus if the decimal has 8 digits then the resulting data type will be numeric(8).
- If the first step was NOT applied, i.e. the decimal has a scale of 1 or greater, then the **data type** will become numeric(%precision,%scale); where we not only substitute the number of digits in the number for the variable '%precision', but we also substitute the scale for the variable '%scale'. Thus, if the decimal is made up of 8 digits and has 3 digits after the decimal point (example 12345,678), the resulting data type will be numeric(8,3).



Name:

Standard SQL Server to SQL Server

Description:

Standard SQL Server to SQL Server data type mapping.
Please do not edit this standard mapping file as any changes will be overwritten on upgrades.
See the user guide for information on how to create custom mapping files.

Conversions:

From Data Type	From Length	From Precision	From Scale	To Data Type	Order	Transformation	Format	Default
(x) int identity				int	7			
(x) bigint				bigint	8			
(x) bigint identity				bigint	9			
(x) numeric			0	numeric(%precision)	10			
(x) numeric			1-*	numeric(%precision,%scale)	11			
(x) decimal			0	numeric(%precision)	12			
(x) decimal			1-*	numeric(%precision,%scale)	13			
(x) smallmoney				smallmoney	14			
(x) money				money	15			
(x) float				float	16			
(x) real				real	17			
(x) date				date	18			
(x) smalldatetime				smalldatetime	19			
(x) datetime				datetime	20			

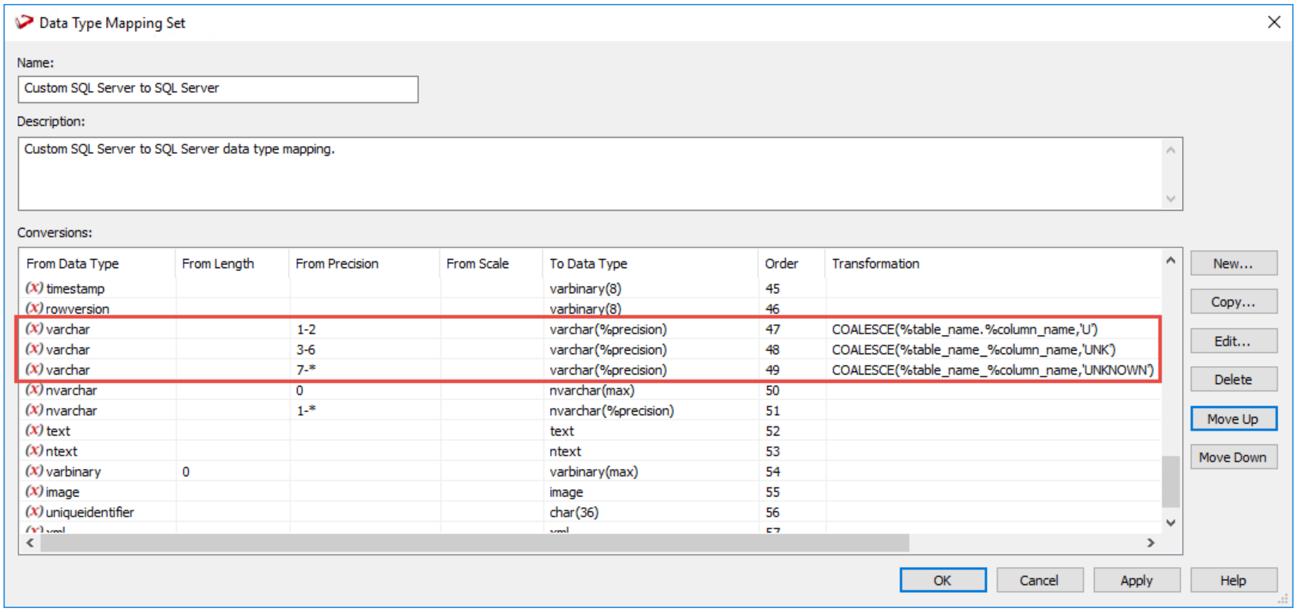
Notes

The **Scale>** is the number of digits to the right of the decimal point in a number.

%table_name and %column_name

In the example below, the following transformations are used to handle NULL for different lengths of varchars:

- If the varchar is 1 or 2 digits/chars long, the **data type** will become varchar(%precision); where we substitute the number of digits/chars in the varchar for the variable '%precision'. Secondly, the **value** of the column will become the column value (if it is not null), else it will become 'U'.
- If the varchar is 3-6 digits/chars long, the **data type** will become varchar(%precision); where we substitute the number of digits/chars in the varchar for the variable '%precision'. Secondly, the **value** of the column will become the column value (if it is not null), else it will become 'UNK'.
- If the varchar is 7 or more digits/chars long, the **data type** will become varchar(%precision); where we substitute the number of digits/chars in the varchar for the variable '%precision'. Secondly, the **value** of the column will become the column value (if it is not null), else it will become 'UNKNOWN'.



Name: Custom SQL Server to SQL Server

Description: Custom SQL Server to SQL Server data type mapping.

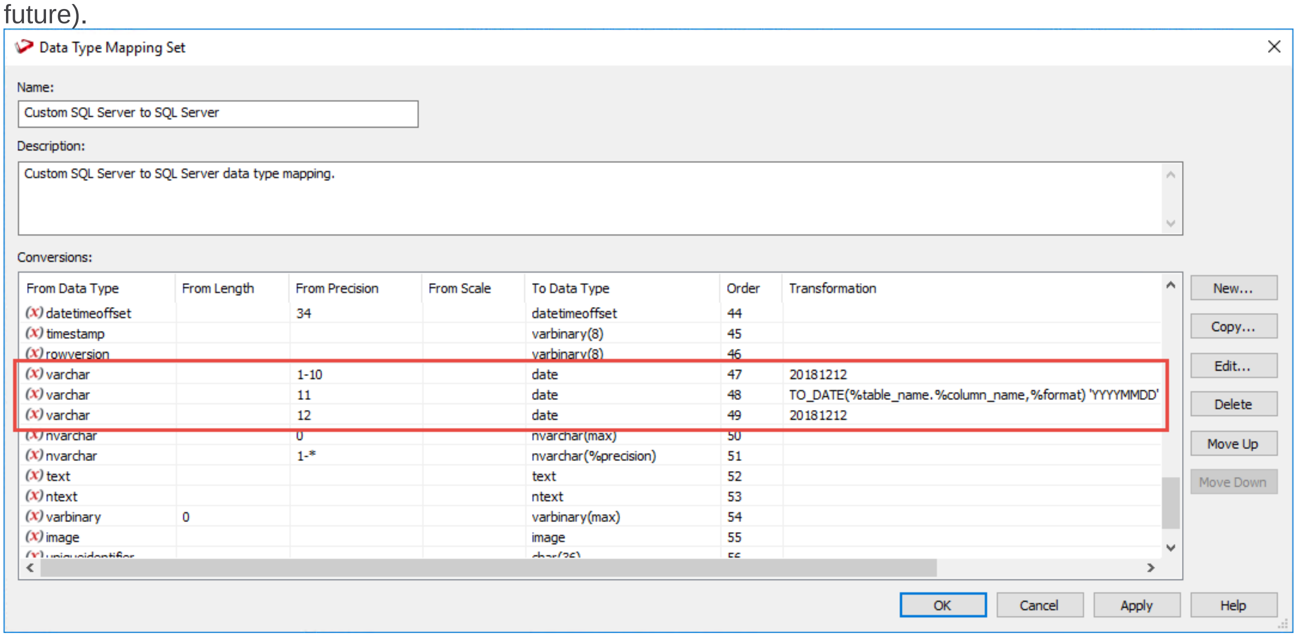
Conversions:

From Data Type	From Length	From Precision	From Scale	To Data Type	Order	Transformation
(x) timestamp				varbinary(8)	45	
(x) rowversion				varbinary(8)	46	
(x) varchar		1-2		varchar(%precision)	47	COALESCE(%table_name.%column_name,'U')
(x) varchar		3-6		varchar(%precision)	48	COALESCE(%table_name_%column_name,'UNK')
(x) varchar		7-*		varchar(%precision)	49	COALESCE(%table_name_%column_name,'UNKNOWN')
(x) nvarchar		0		nvarchar(max)	50	
(x) nvarchar		1-*		nvarchar(%precision)	51	
(x) text				text	52	
(x) ntext				ntext	53	
(x) varbinary	0			varbinary(max)	54	
(x) image				image	55	
(x) uniqueidentifier				char(36)	56	
(x) xml				xml	57	

%format

In the example below, the following transformations are used to convert a certain character field to a date:

- If the varchar has a length of 1-10, the **data type** will become date and the **value** of the column will become the date 20131212 (a chosen date in the future).
- If the varchar has a length of 11, the **data type** will become date and the **value** of the column will use the transformation TO_DATE(%table_name.%column_name,%format); where we substitute 'YYYYMMDD' for the variable '%format'. Thus the value of the column will be converted to a date of format 'YYYYMMDD'.
- If the varchar has a length of 12 or greater, the **data type** will become date and the **value** of the column will become the date 20181212 (a chosen date in the future).



Name: Custom SQL Server to SQL Server

Description: Custom SQL Server to SQL Server data type mapping.

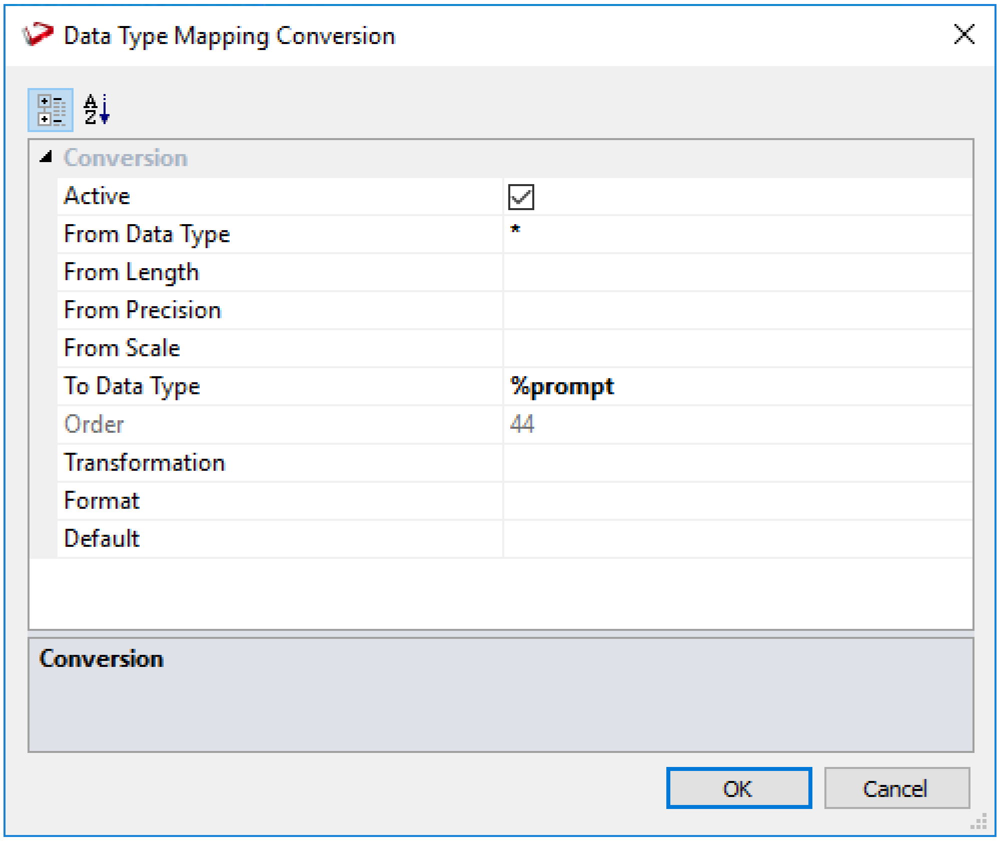
Conversions:

From Data Type	From Length	From Precision	From Scale	To Data Type	Order	Transformation
(x) datetimeoffset		34		datetimeoffset	44	
(x) timestamp				varbinary(8)	45	
(x) rowversion				varbinary(8)	46	
(x) varchar		1-10		date	47	20181212
(x) varchar		11		date	48	TO_DATE(%table_name.%column_name,%format) 'YYYYMMDD'
(x) varchar		12		date	49	20181212
(x) nvarchar		0		nvarchar(max)	50	
(x) nvarchar		1-*		nvarchar(%precision)	51	
(x) text				text	52	
(x) ntext				ntext	53	
(x) varbinary	0			varbinary(max)	54	
(x) image				image	55	
(x) uniqueidentifier				char(36)	56	

%prompt

In the example below, %prompt is used to help the user define a mapping for an unknown data type that is not already mapped in the previous mapping rules.
This variable must be used with a **custom** Data Type mapping set, as described in the following steps:

- Create a new custom set or copy from an existing set.
- Create a new Data Type mapping with a **From Data Type** of star (*****) and a **To Data Type** of **%prompt**. Click **OK** to save the New Data Type Mapping to the Custom set.



Data Type Mapping Conversion

Conversion

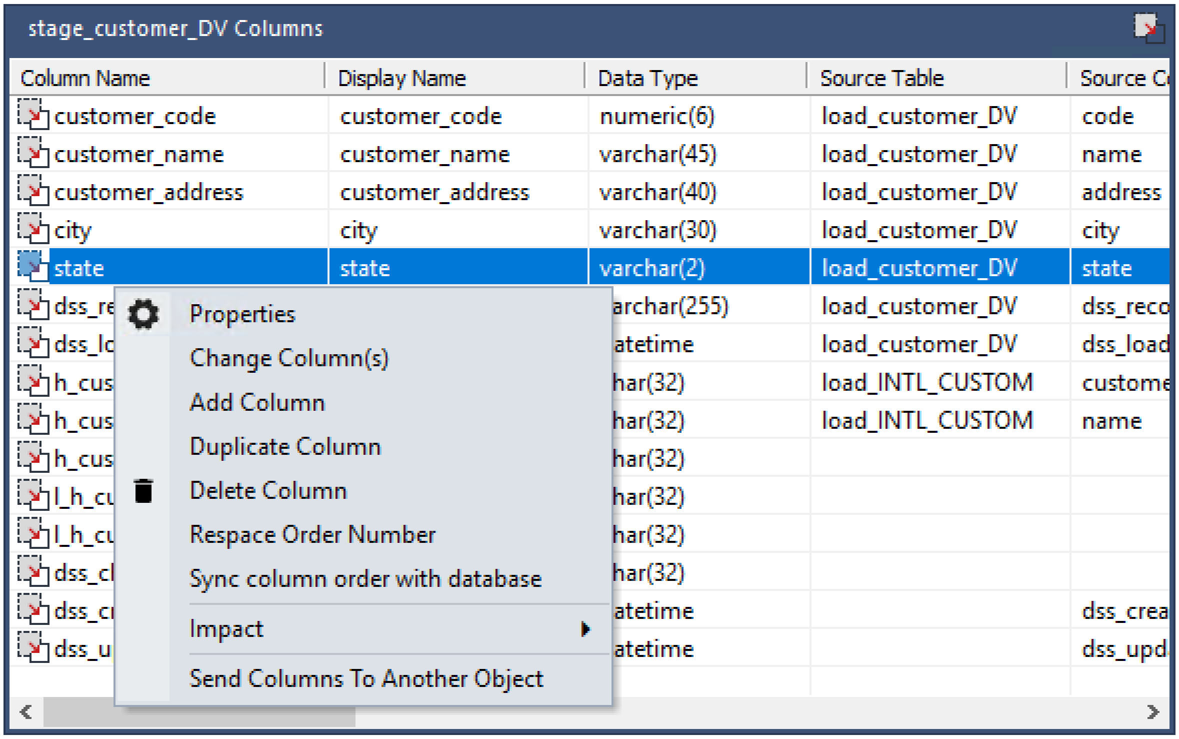
Active	☑
From Data Type	*
From Length	
From Precision	
From Scale	
To Data Type	**%prompt**
Order	44
Transformation	
Format	
Default	

Conversion

- As the table is dragged and dropped to the middle pane, RED prompts to have the new data type mapping defined.

Column Context Menu

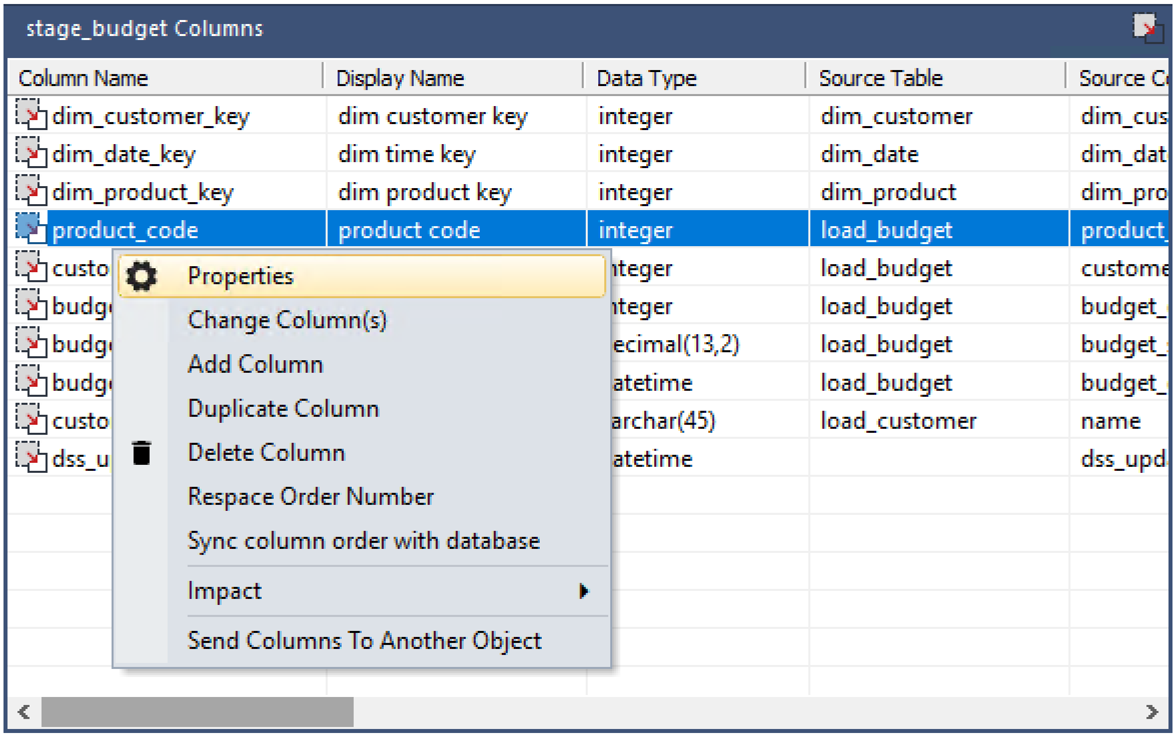
To view the column context menu, click on an object in the left pane to display the columns in the middle pane. When positioned on a column in the middle pane, right-click on the column to bring up the menu.



Column Name	Display Name	Data Type	Source Table	Source C
customer_code	customer_code	numeric(6)	load_customer_DV	code
customer_name	customer_name	varchar(45)	load_customer_DV	name
customer_address	customer_address	varchar(40)	load_customer_DV	address
city	city	varchar(30)	load_customer_DV	city
state	state	varchar(2)	load_customer_DV	state
dss_re		archar(255)	load_customer_DV	dss_reco
dss_lo		atetime	load_customer_DV	dss_load
h_cus		har(32)	load_INTL_CUSTOM	custome
h_cus		har(32)	load_INTL_CUSTOM	name
h_cus		har(32)		
l_h_cu		har(32)		
l_h_cu		har(32)		
dss_cl		har(32)		
dss_cr		atetime		dss_crea
dss_u		atetime		dss_upd

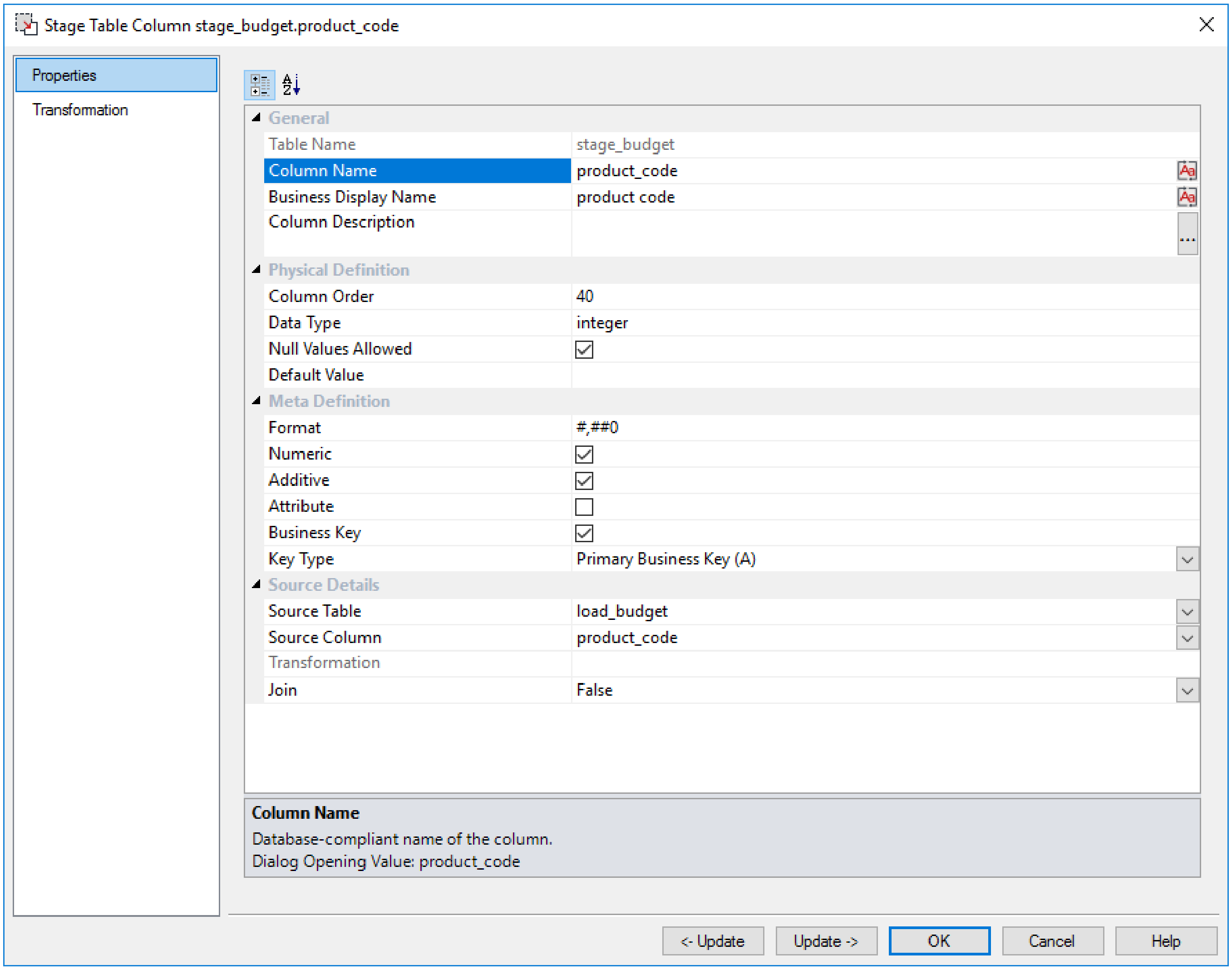
Properties

To display the column Properties, right-click on a column in the middle pane and select **Properties**.



Column Name	Display Name	Data Type	Source Table	Source C
dim_customer_key	dim customer key	integer	dim_customer	dim_cus
dim_date_key	dim time key	integer	dim_date	dim_dat
dim_product_key	dim product key	integer	dim_product	dim_pro
product_code	product code	integer	load_budget	product_
custo		nteger	load_budget	custome
budg		nteger	load_budget	budget_
budg		ecimal(13,2)	load_budget	budget_
budg		atetime	load_budget	budget_
custo		archar(45)	load_customer	name
dss_u		atetime		dss_upd

Edit any field as required and then click **OK** to close.



Stage Table Column stage_budget.product_code

General	
Table Name	stage_budget
Column Name	product_code
Business Display Name	product code
Column Description	
Physical Definition	
Column Order	40
Data Type	integer
Null Values Allowed	☑
Default Value	
Meta Definition	
Format	#,##0
Numeric	☑
Additive	☑
Attribute	☐
Business Key	☑
Key Type	Primary Business Key (A)
Source Details	
Source Table	load_budget
Source Column	product_code
Transformation	
Join	False

Column Name

Database-compliant name of the column.

Dialog Opening Value: product_code

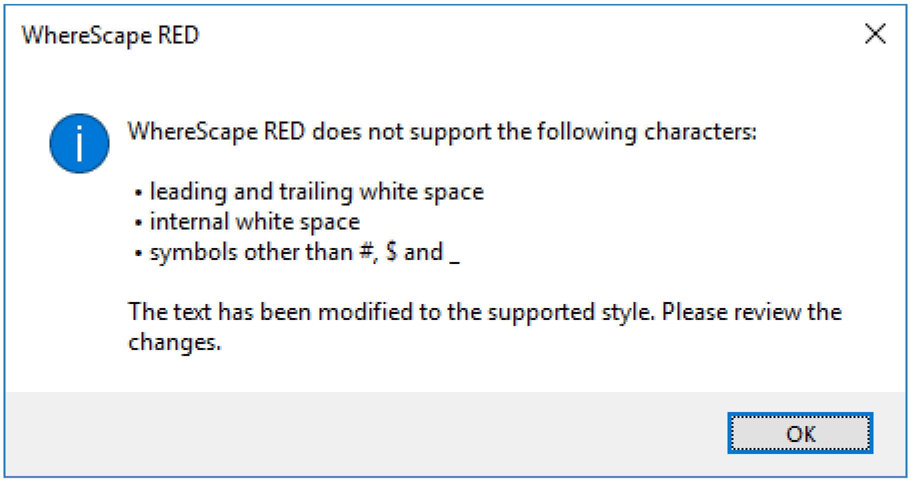
Warning

WhereScape RED does not support the following characters in **Column Names**:

- leading and trailing white spaces
- internal white spaces
- symbols other than #, $ and _

If users attempt to enter any of the above characters in Column Names, the following message is displayed, advising users to review changes made by RED to correct any unsupported column name characters:

Warning



WhereScape RED

WhereScape RED does not support the following characters:

- leading and trailing white space
- internal white space
- symbols other than #, $ and _

The text has been modified to the supported style. Please review the changes.

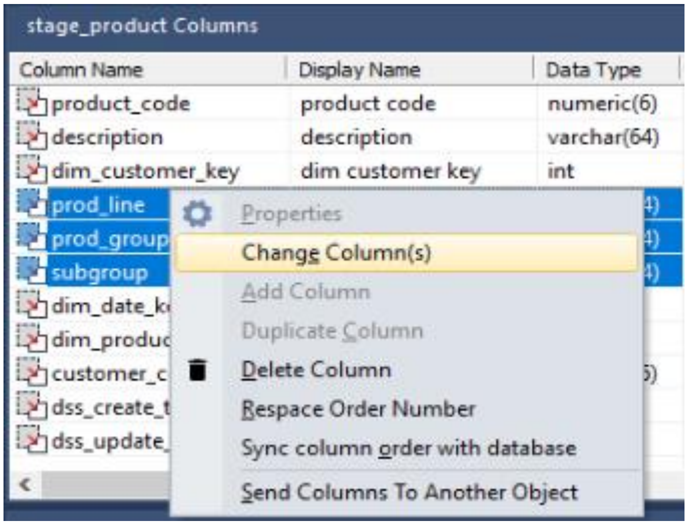
Note

There is a variation in column Properties, depending on the object type.

- For Dimension tables, refer to **Dimension Column Properties**
- For Stage tables, refer to **Stage Table Column Properties**
- For Data Store tables, refer to **Data Store Column Properties**
- For EDW 3NF tables, refer to **EDW 3NF Table Column Properties**
- For Fact tables, refer to **Fact Table Column Properties**

Change Column(s)

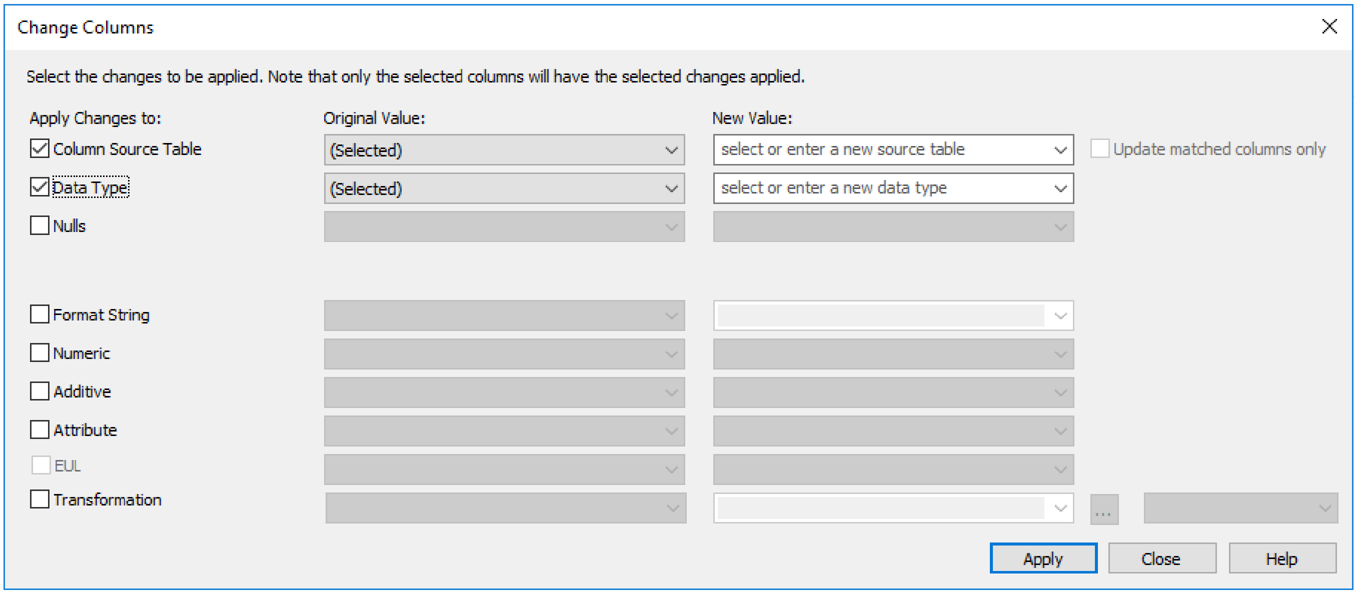
To change the properties of multiple columns, right-click the columns in the middle pane and select **Change Column(s)**.



stage_product Columns

Column Name	Display Name	Data Type
product_code	product code	numeric(6)
description	description	varchar(64)
dim_customer_key	dim customer key	int
prod_line		4)
prod_group		4)
subgroup		4)
dim_date_k		
dim_produc		
customer_c		5)
dss_create_t		
dss_update_		

Select the relevant **check boxes** in the left pane to specify the properties to be changed. Selecting a check box enables you to change the value for that field in the column properties.

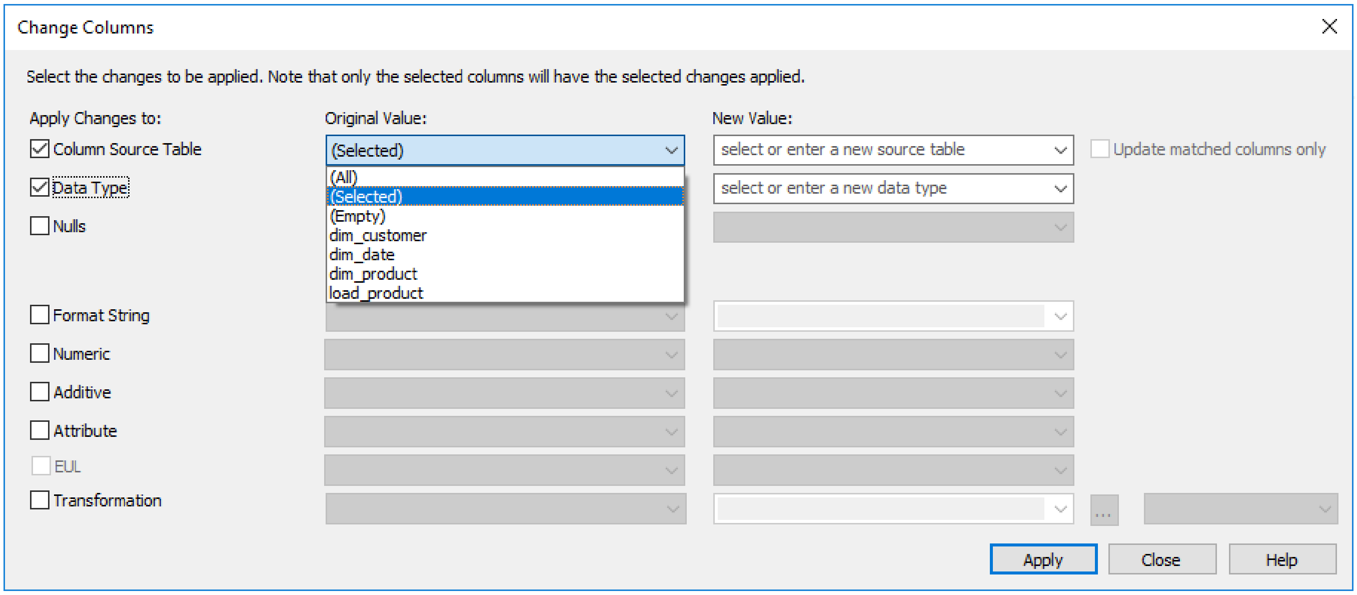


Change Columns

Select the changes to be applied. Note that only the selected columns will have the selected changes applied.

Apply Changes to:	Original Value:	New Value:	
☑ Column Source Table	(Selected)	select or enter a new source table	□ Update matched columns only
☑ Data Type	(Selected)	select or enter a new data type	
□ Nulls			
□ Format String			
□ Numeric			
□ Additive			
□ Attribute			
□ EUL			
□ Transformation			

In the **Original Value** column, select the value(s) to be changed.



Change Columns

Select the changes to be applied. Note that only the selected columns will have the selected changes applied.

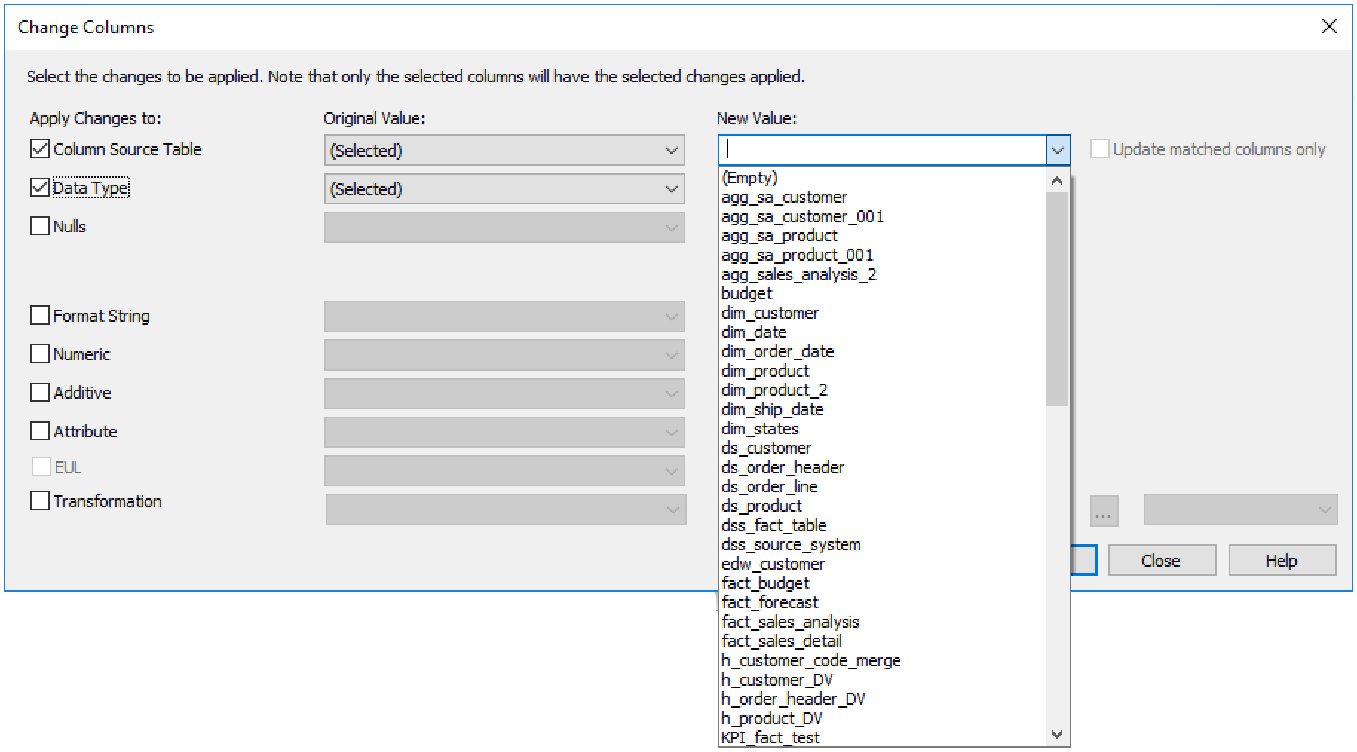
Apply Changes to:	Original Value:	New Value:	
☑ Column Source Table	(Selected)	select or enter a new source table	□ Update matched columns only
☑ Data Type	(All) (Selected) (Empty) dim_customer dim_date dim_product load_product	select or enter a new data type	
□ Nulls			
□ Format String			
□ Numeric			
□ Additive			
□ Attribute			
□ EUL			
□ Transformation			

- Choosing **(All)** changes the selected property for all of the columns in the table.
- Choosing **(Selected)** changes the selected property for the selected column in the table.
- Choosing **(Empty)** changes the selected property for all of the columns where that property field is empty. This option is only available if there is a column where this property is empty.
- Choosing one of the **other** options changes the selected property for all the columns in the table having that value.

Note:

(Selected) is the default for the **Original Value** column.

In the **New Value** column, select the new value to be assigned; or key in the new value.

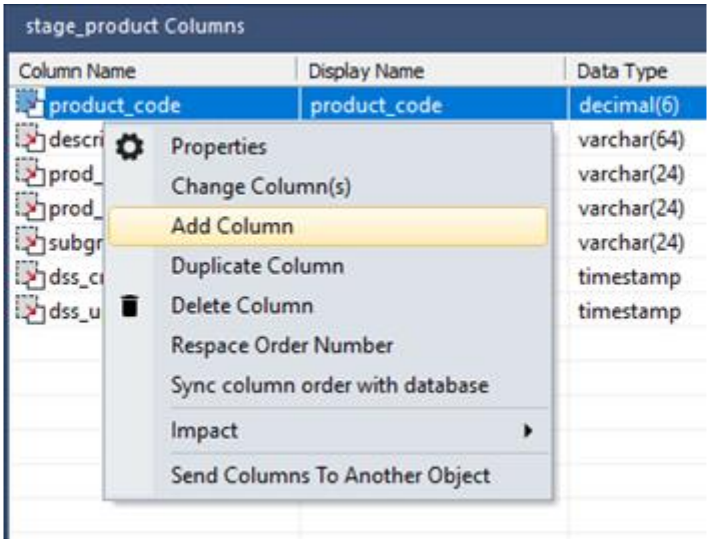


Notes:

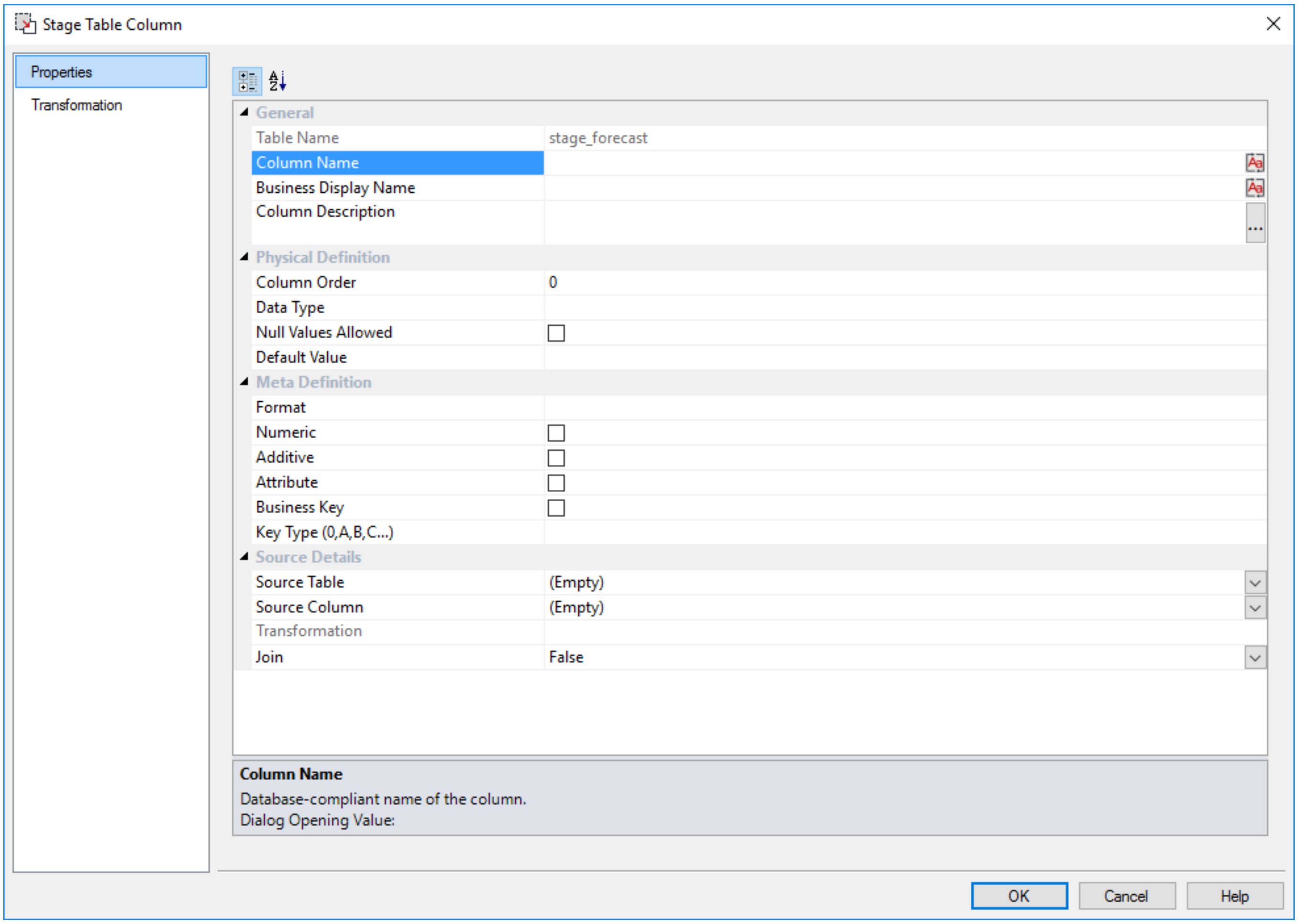
- When editing the **Column Source Table** property, selecting the **Update matched columns only** check-box validates that the selected column name exists in the new source table. If it does not exist, then the update does not take place.
- Selecting the **Transformation** check box enables you to define a transformation for the selected column(s). Refer to **Column Transformations** for details.

Add Column

To add a column, right-click on one of the columns in the middle pane and select **Add Column**.



Enter the details to define a new column and click **OK**.

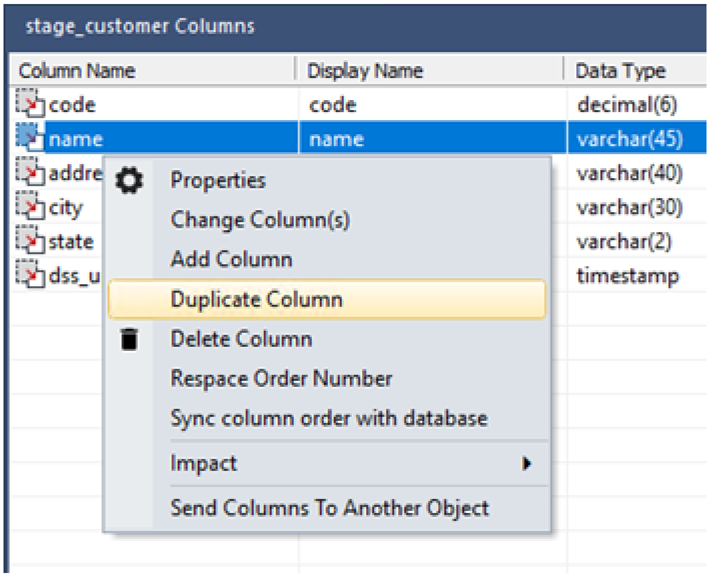


General	
Table Name	stage_forecast
Column Name	
Business Display Name	
Column Description	
Physical Definition	
Column Order	0
Data Type	
Null Values Allowed	□
Default Value	
Meta Definition	
Format	
Numeric	□
Additive	□
Attribute	□
Business Key	□
Key Type (0,A,B,C...)	
Source Details	
Source Table	(Empty)
Source Column	(Empty)
Transformation	
Join	False

Column Name
Database-compliant name of the column.
Dialog Opening Value:

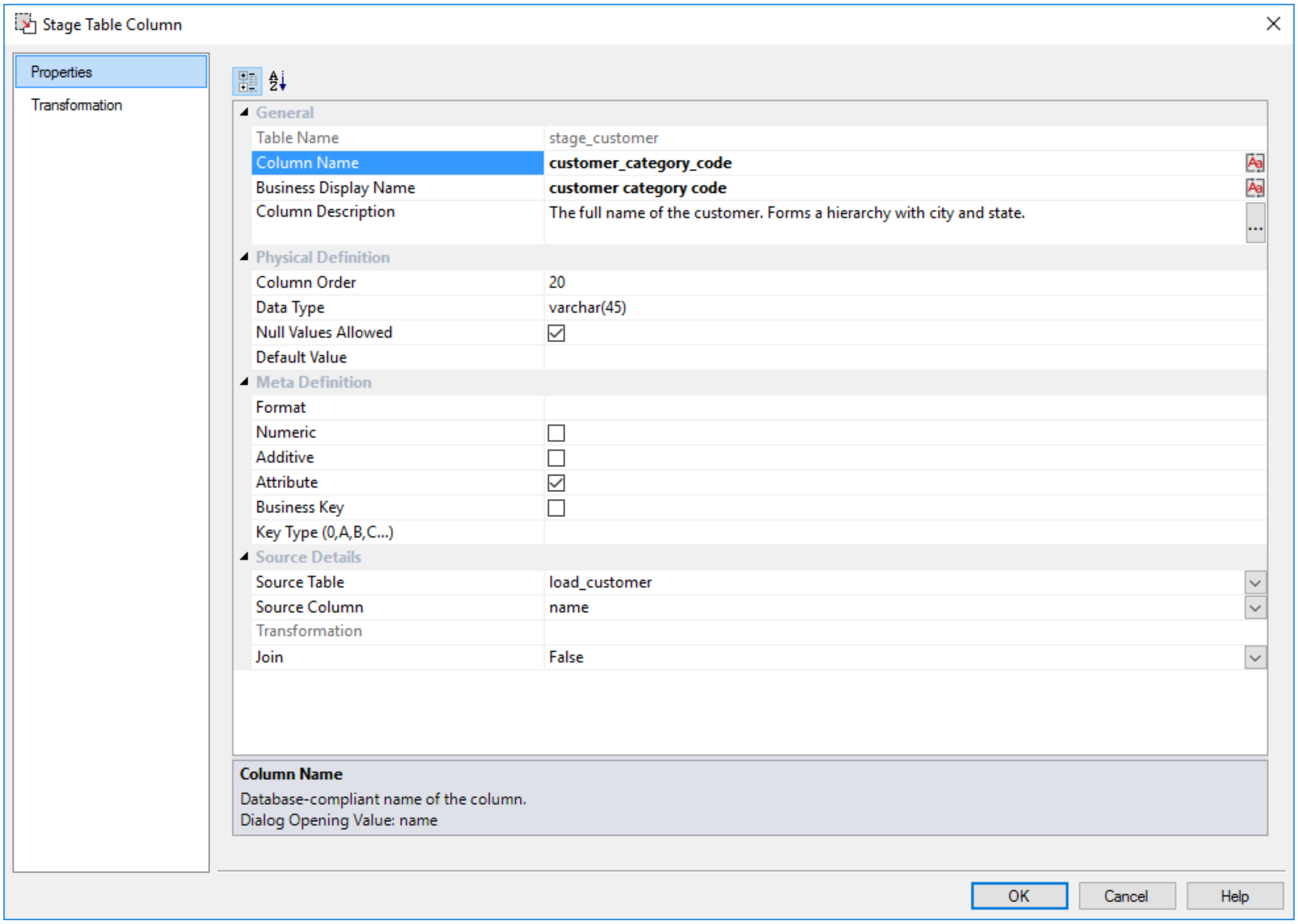
Duplicate Column

To copy a column, right-click on one of the columns in the middle pane and select **Duplicate Column**.


stage_customer Columns

Column Name	Display Name	Data Type
code	code	decimal(6)
name	name	varchar(45)
addre		varchar(40)
city		varchar(30)
state		varchar(2)
dss_u		timestamp

Change the **Column Name** and the **Business Display Name** and any other properties to define a new column and click **OK**.

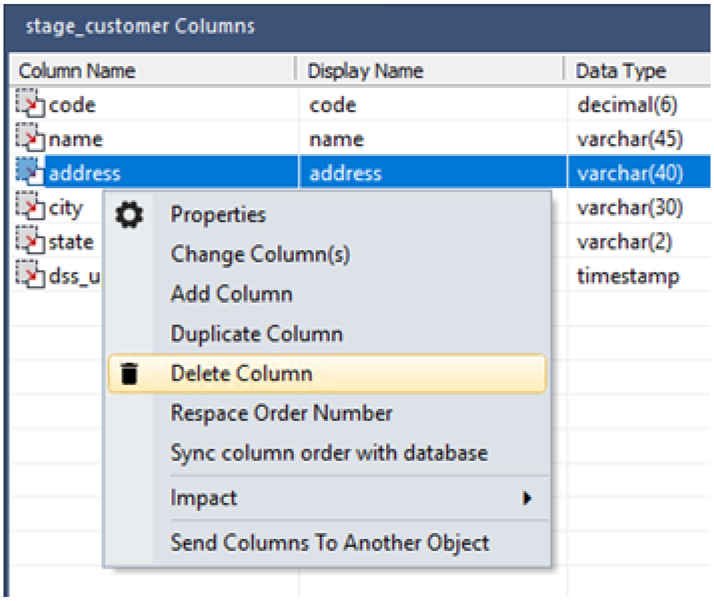


General	
Table Name	stage_customer
Column Name	**customer_category_code**
Business Display Name	**customer category code**
Column Description	The full name of the customer. Forms a hierarchy with city and state.
Physical Definition	
Column Order	20
Data Type	varchar(45)
Null Values Allowed	☑
Default Value	
Meta Definition	
Format	
Numeric	□
Additive	□
Attribute	☑
Business Key	□
Key Type (0,A,B,C...)	
Source Details	
Source Table	load_customer
Source Column	name
Transformation	
Join	False

Column Name
Database-compliant name of the column.
Dialog Opening Value: name

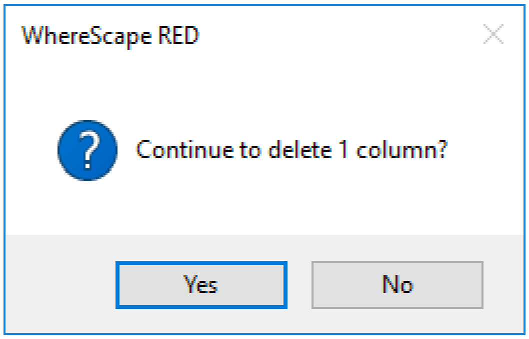
Delete Column

To delete a column, right-click on one of the columns in the middle pane and select **Delete Column**.


stage_customer Columns

Column Name	Display Name	Data Type
code	code	decimal(6)
name	name	varchar(45)
address	address	varchar(40)
city		varchar(30)
state		varchar(2)
dss_u		timestamp

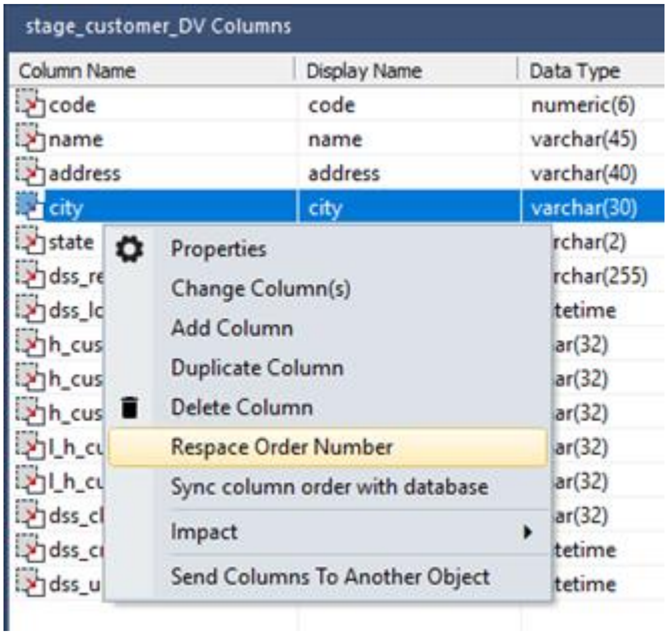
Click **Yes** to continue with the delete.


WhereScape RED

Continue to delete 1 column?

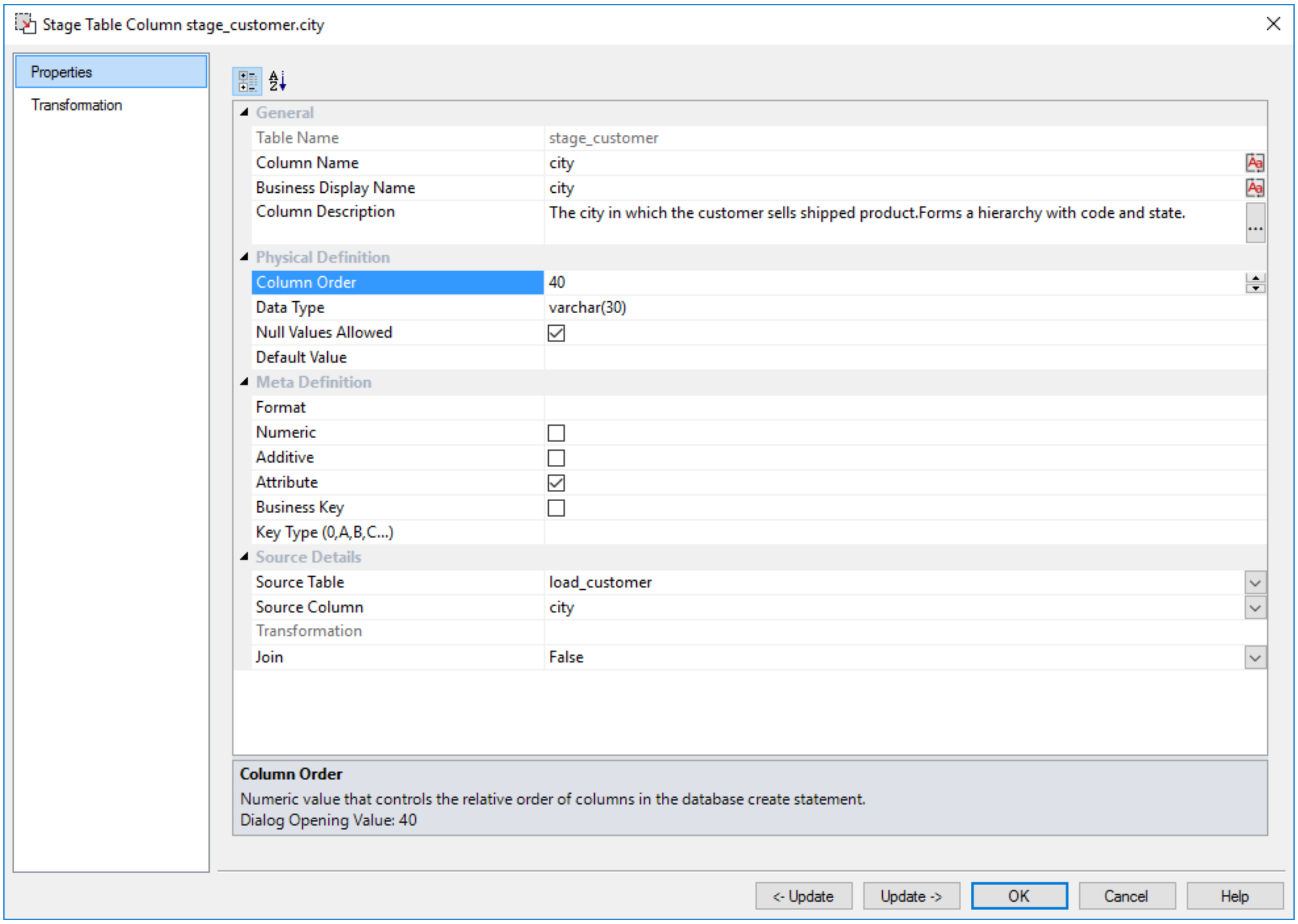
Re-space Order Number

To re-space the column order, right-click on any column in the middle pane and select **Respace Order Number.**



Column Name	Display Name	Data Type
code	code	numeric(6)
name	name	varchar(45)
address	address	varchar(40)
city	city	varchar(30)
state		rchar(2)
dss_re		rchar(255)
dss_lo		tetime
h_cus		ar(32)
h_cus		ar(32)
h_cus		ar(32)
l_h_cu		ar(32)
l_h_cu		ar(32)
dss_cl		ar(32)
dss_ci		tetime
dss_u		tetime

The **Column Order** number for each column will be adjusted so that the column order numbers are evenly spaced.



General

Table Name	stage_customer
Column Name	city
Business Display Name	city
Column Description	The city in which the customer sells shipped product.Forms a hierarchy with code and state.

Physical Definition

Column Order	40
Data Type	varchar(30)
Null Values Allowed	☑
Default Value	

Meta Definition

Format	
Numeric	□
Additive	□
Attribute	☑
Business Key	□
Key Type (0,A,B,C...)	

Source Details

Source Table	load_customer
Source Column	city
Transformation	
Join	False

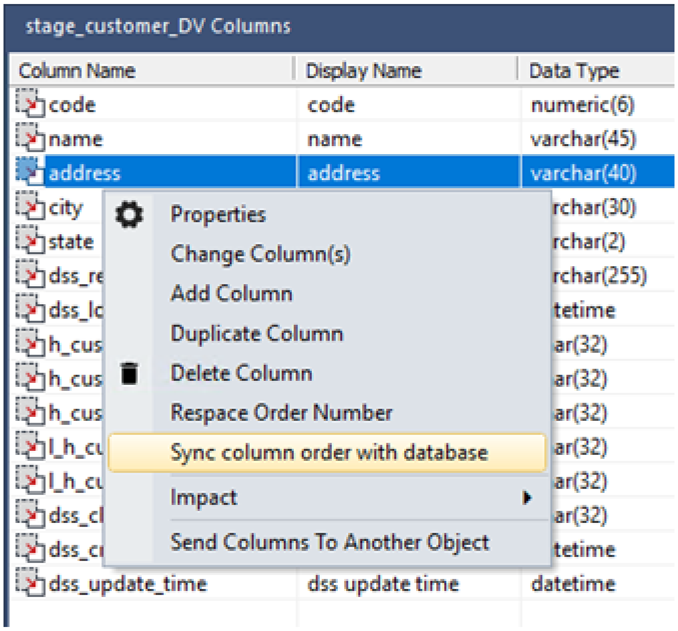
Column Order
Numeric value that controls the relative order of columns in the database create statement.
Dialog Opening Value: 40

Sync Column order with database

To synchronize the metadata's column order to match the same order in the physical table in the database:

Right-click on one of the columns in the middle pane and select **Sync the column order with the database**.

This will reorder the metadata columns to match the column order in the database table.

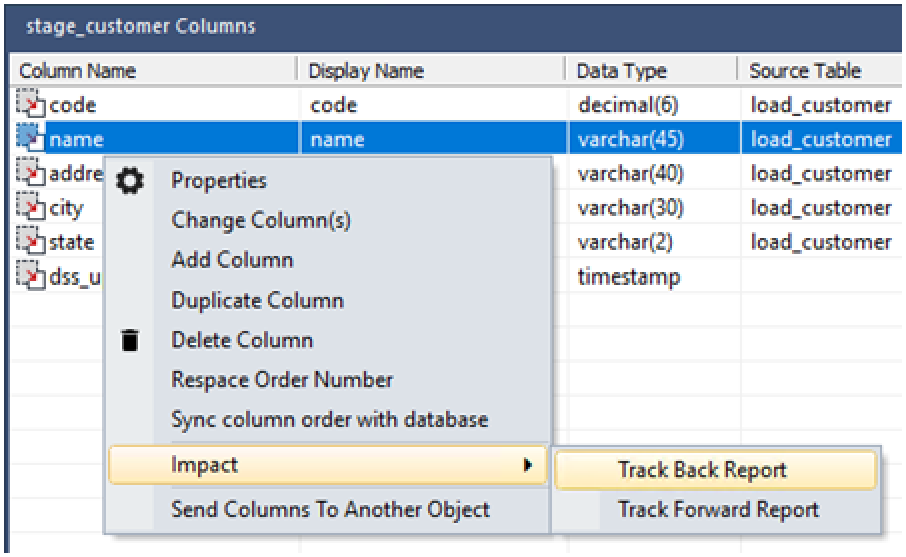


stage_customer_DV Columns

Column Name	Display Name	Data Type
code	code	numeric(6)
name	name	varchar(45)
address	address	varchar(40)
city		rchar(30)
state		rchar(2)
dss_re		rchar(255)
dss_lc		tetime
h_cus		ar(32)
h_cus		ar(32)
h_cus		ar(32)
l_h_cu		ar(32)
l_h_cu		ar(32)
dss_cl		ar(32)
dss_cr		tetime
dss_update_time	dss update time	datetime

Impact

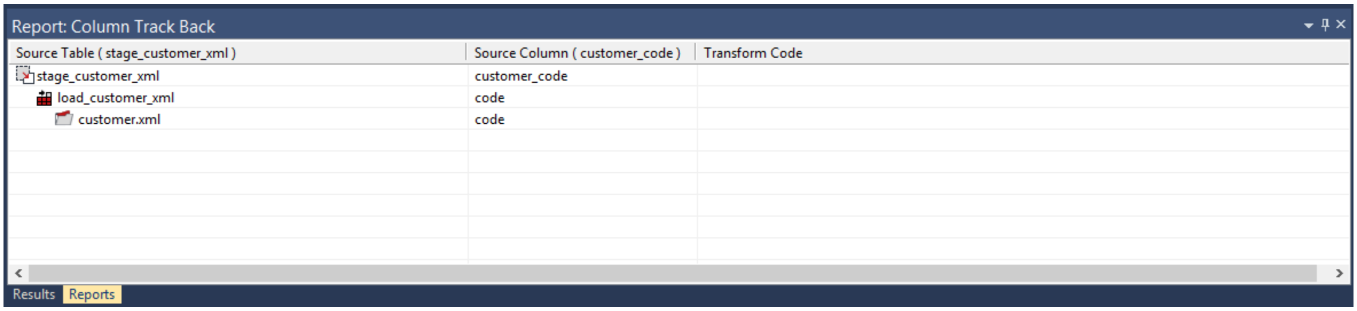
To display a Track Back Report, right-click on a column in the middle pane and select **Impact > Track Back Report**.



stage_customer Columns

Column Name	Display Name	Data Type	Source Table
code	code	decimal(6)	load_customer
name	name	varchar(45)	load_customer
addre		varchar(40)	load_customer
city		varchar(30)	load_customer
state		varchar(2)	load_customer
dss_u		timestamp	

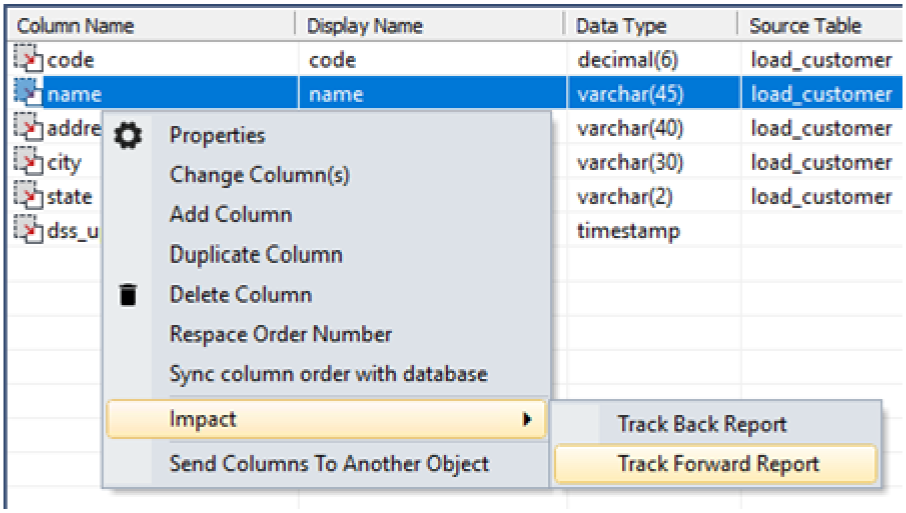
The report will be displayed in the bottom middle pane. This report lists the origins of the selected column. Refer to Track Back Report for details.



Report: Column Track Back

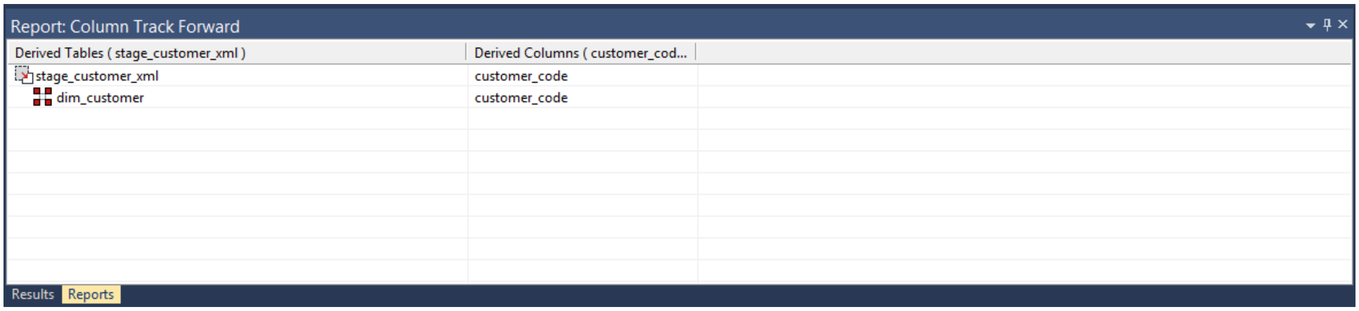
Source Table (stage_customer_xml)	Source Column (customer_code)	Transform Code
stage_customer_xml	customer_code	
load_customer_xml	code	
customer.xml	code	

To display a Track Forward Report, right-click on a column in the middle pane and select **Impact > Track Forward Report**.



Column Name	Display Name	Data Type	Source Table
code	code	decimal(6)	load_customer
name	name	varchar(45)	load_customer
addre		varchar(40)	load_customer
city		varchar(30)	load_customer
state		varchar(2)	load_customer
dss_u		timestamp	

The report will be displayed in the bottom middle pane. This report lists the columns derived from the selected column. Refer to Track Forward Report for details.

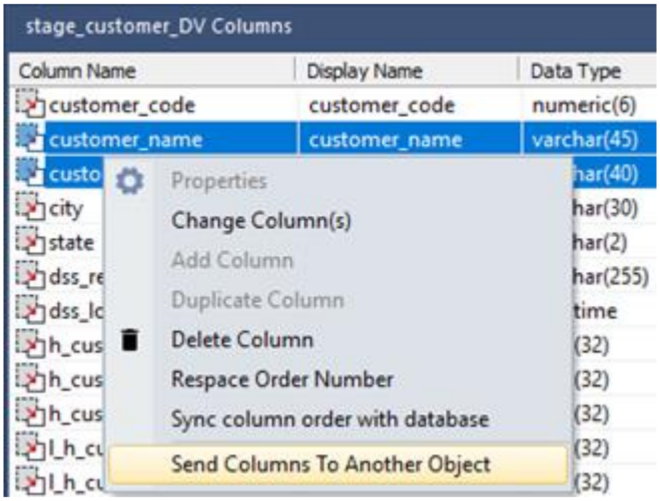


Report: Column Track Forward

Derived Tables (stage_customer_xml)	Derived Columns (customer_cod...
stage_customer_xml	customer_code
dim_customer	customer_code

Send Columns to Another Object

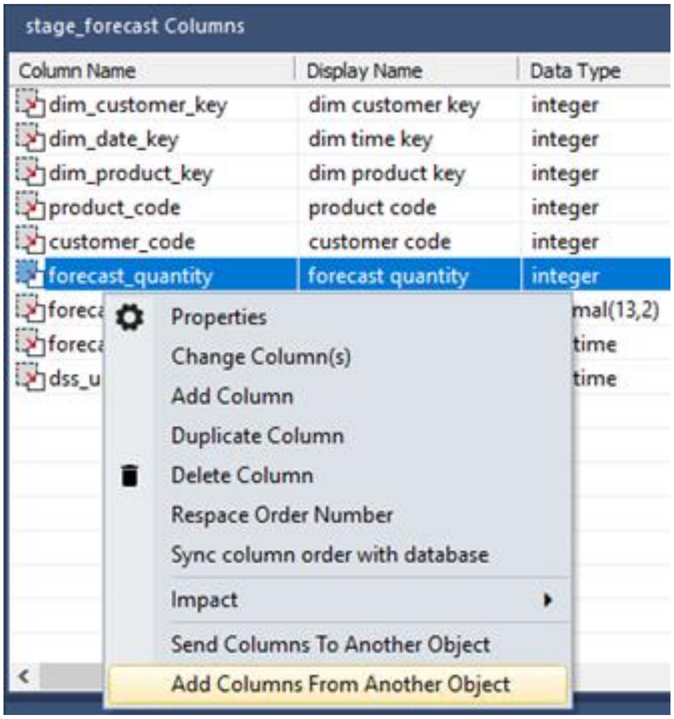
To send/copy columns to another object, right-click on a column in the middle pane and select **Send Columns To Another Object**.



stage_customer_DV Columns

Column Name	Display Name	Data Type
customer_code	customer_code	numeric(6)
customer_name	customer_name	varchar(45)
custo		har(40)
city		har(30)
state		har(2)
dss_re		har(255)
dss_lo		time
h_cus		(32)
h_cus		(32)
h_cus		(32)
l_h_cu		(32)
l_h_cu		(32)

Click on the destination table in the left pane, then right-click in the middle pane and select **Add Columns From Another Object**.

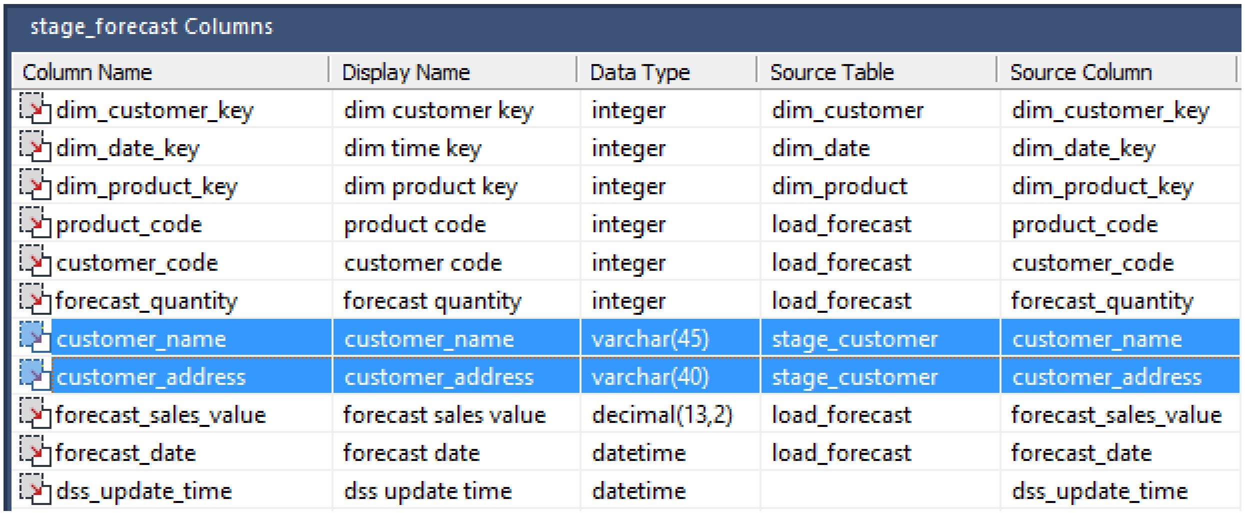


stage_forecast Columns

Column Name	Display Name	Data Type
dim_customer_key	dim customer key	integer
dim_date_key	dim time key	integer
dim_product_key	dim product key	integer
product_code	product code	integer
customer_code	customer code	integer
forecast_quantity	forecast quantity	integer
foreca		mal(13,2)
foreca		time
dss_u		time

Properties
Change Column(s)
Add Column
Duplicate Column
Delete Column
Respace Order Number
Sync column order with database
Impact
Send Columns To Another Object
Add Columns From Another Object

The columns will be added to the destination table using the same functionality and settings as drag and drop.



stage_forecast Columns

Column Name	Display Name	Data Type	Source Table	Source Column
dim_customer_key	dim customer key	integer	dim_customer	dim_customer_key
dim_date_key	dim time key	integer	dim_date	dim_date_key
dim_product_key	dim product key	integer	dim_product	dim_product_key
product_code	product code	integer	load_forecast	product_code
customer_code	customer code	integer	load_forecast	customer_code
forecast_quantity	forecast quantity	integer	load_forecast	forecast_quantity
customer_name	customer_name	varchar(45)	stage_customer	customer_name
customer_address	customer_address	varchar(40)	stage_customer	customer_address
forecast_sales_value	forecast sales value	decimal(13,2)	load_forecast	forecast_sales_value
forecast_date	forecast date	datetime	load_forecast	forecast_date
dss_update_time	dss update time	datetime		dss_update_time

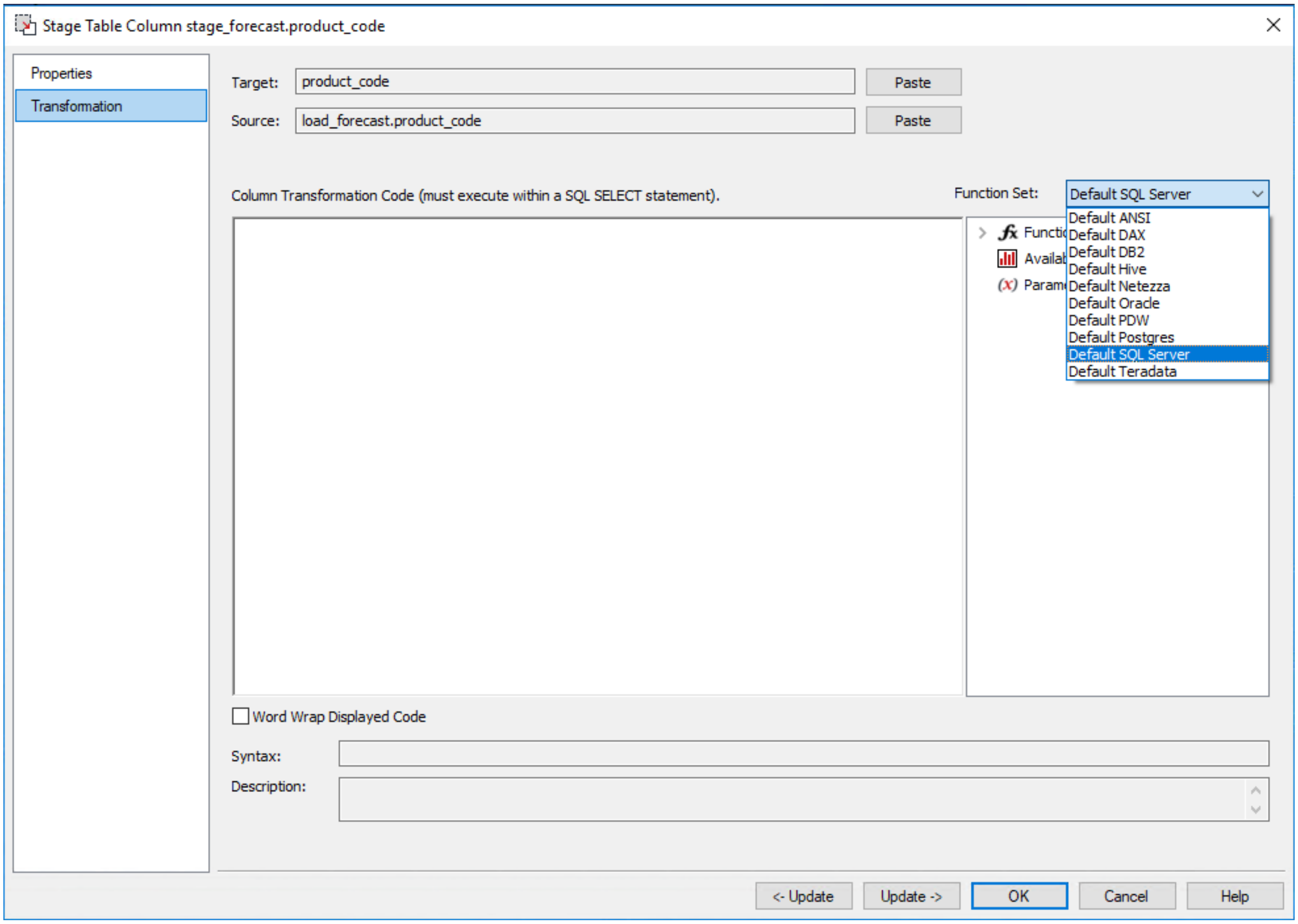
Database Functions

Using Database Function Sets

Database function sets contain a list of functions and operators that can be used for building transformations. These function sets may be created, edited, deleted, imported and exported using the **Database Functions** options from the **Tools** menu.

Column Transformation Properties

A drop-down list enables you to select which set of functions are to be displayed in the tree view when creating a transformation on a column of a table.



Target: product_code

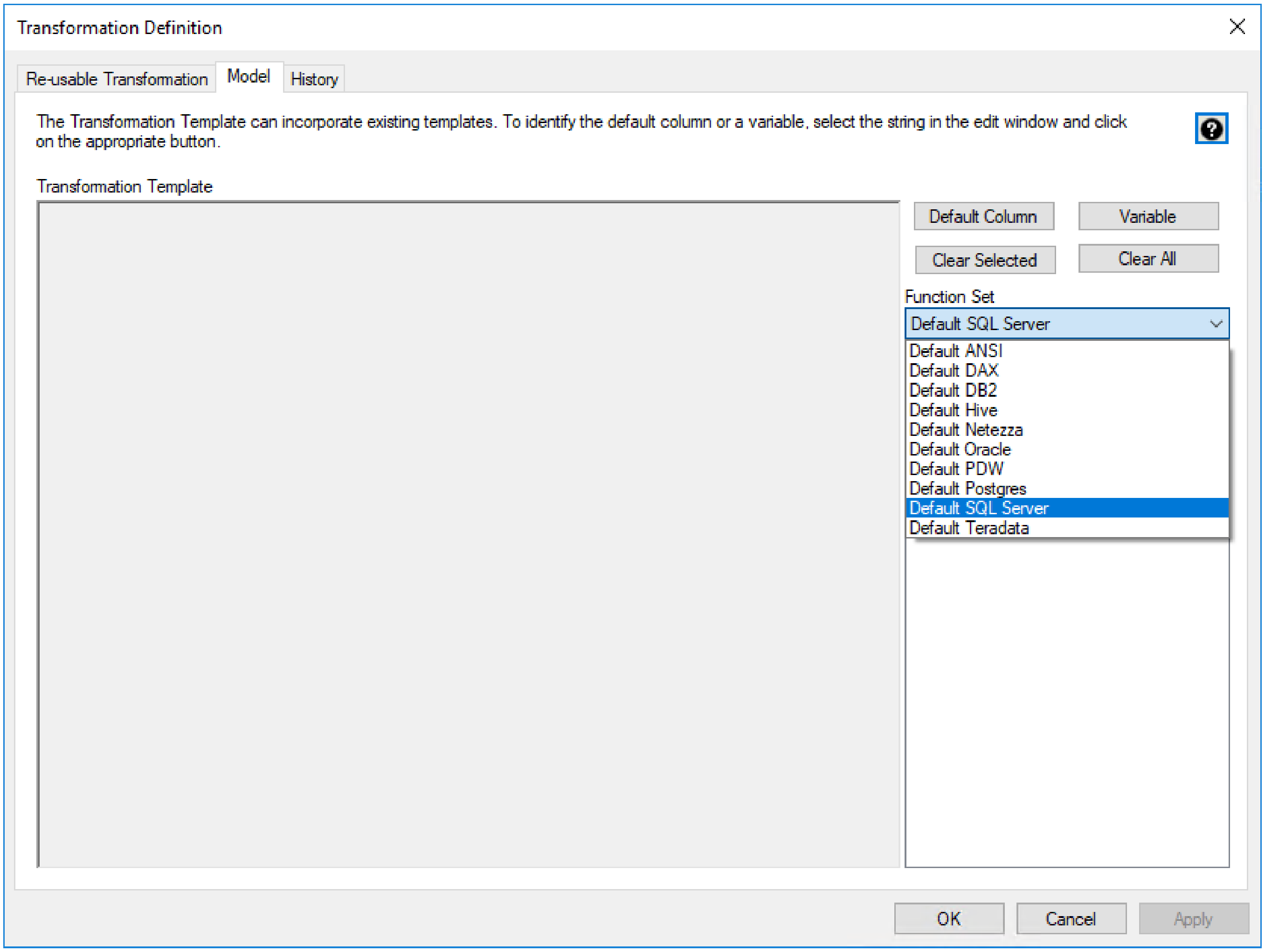
Source: load_forecast.product_code

Function Set: Default SQL Server

□ Word Wrap Displayed Code

Transformation Definition

A drop-down list enables you to select which set of functions are to be displayed in the tree view when creating a re-usable transformation using Tools > Define Re-Usable Transformations.

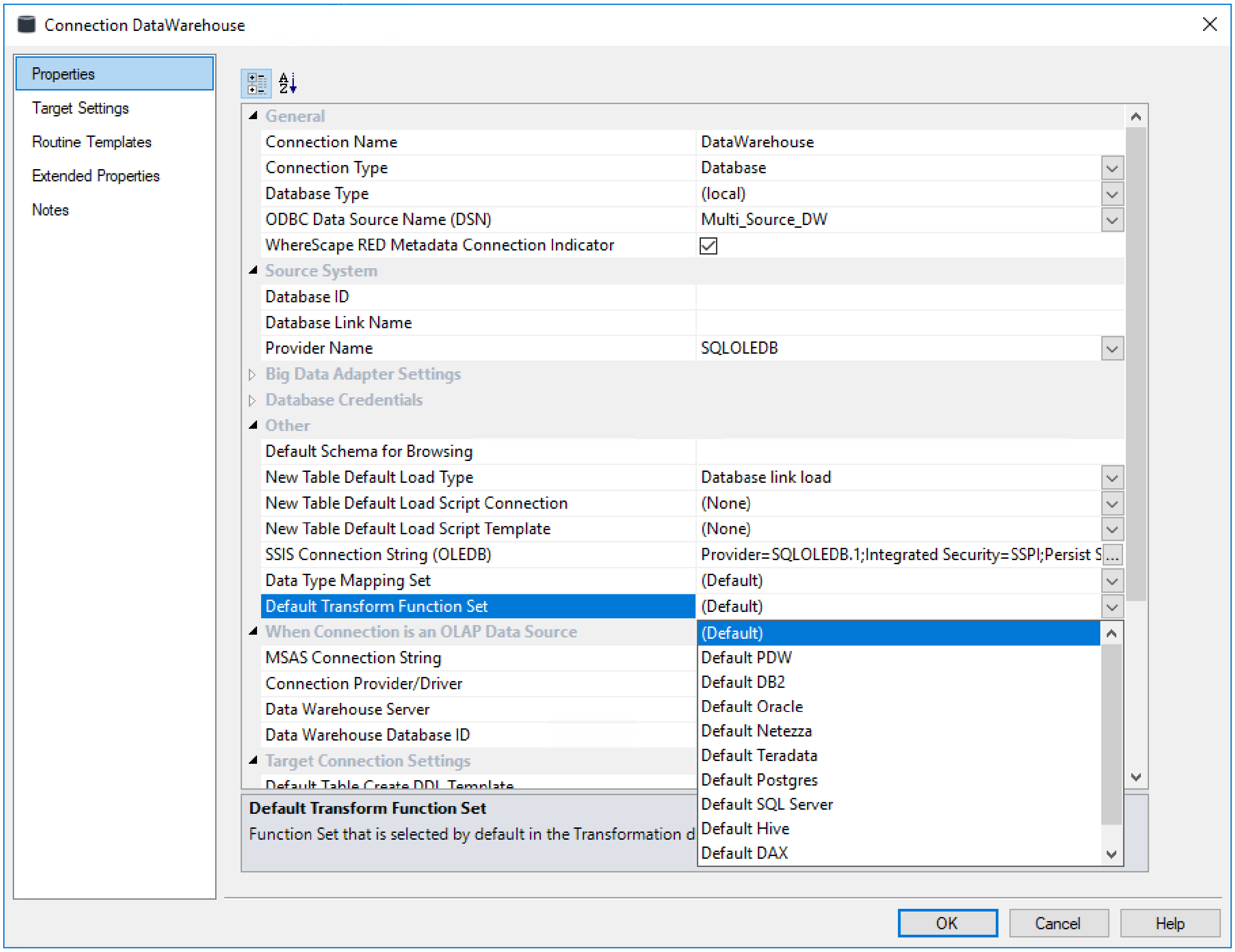


The Transformation Template can incorporate existing templates. To identify the default column or a variable, select the string in the edit window and click on the appropriate button.

Function Set: Default SQL Server

Connection Properties

If the **Data Warehouse Metadata Connection Indicator** check-box is selected In the connection **Properties** window, a drop-down list **Default Transform Function Set** is displayed. This is the default set selected in the transformation screens above.



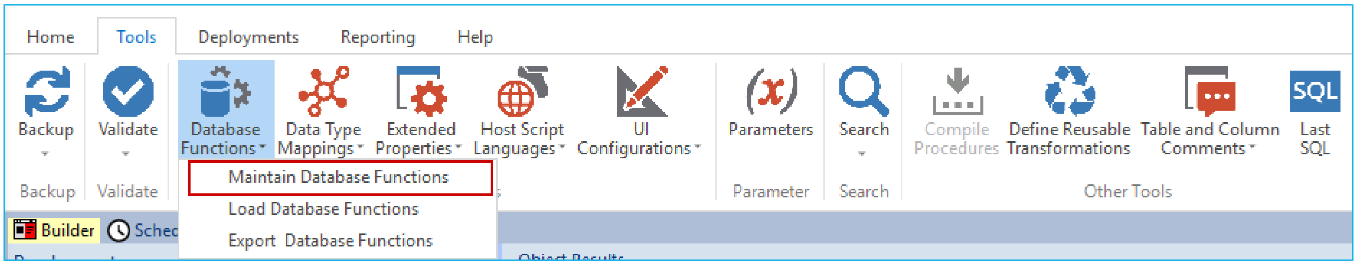
Connection DataWarehouse

General	
Connection Name	DataWarehouse
Connection Type	Database
Database Type	(local)
ODBC Data Source Name (DSN)	Multi_Source_DW
WhereScape RED Metadata Connection Indicator	☑
Source System	
Database ID	
Database Link Name	
Provider Name	SQLOLEDB
Big Data Adapter Settings	
Database Credentials	
Other	
Default Schema for Browsing	
New Table Default Load Type	Database link load
New Table Default Load Script Connection	(None)
New Table Default Load Script Template	(None)
SSIS Connection String (OLEDB)	Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist S
Data Type Mapping Set	(Default)
Default Transform Function Set	(Default)
When Connection is an OLAP Data Source	
MSAS Connection String	
Connection Provider/Driver	
Data Warehouse Server	
Data Warehouse Database ID	
Target Connection Settings	
Default Table Create DDL Template	

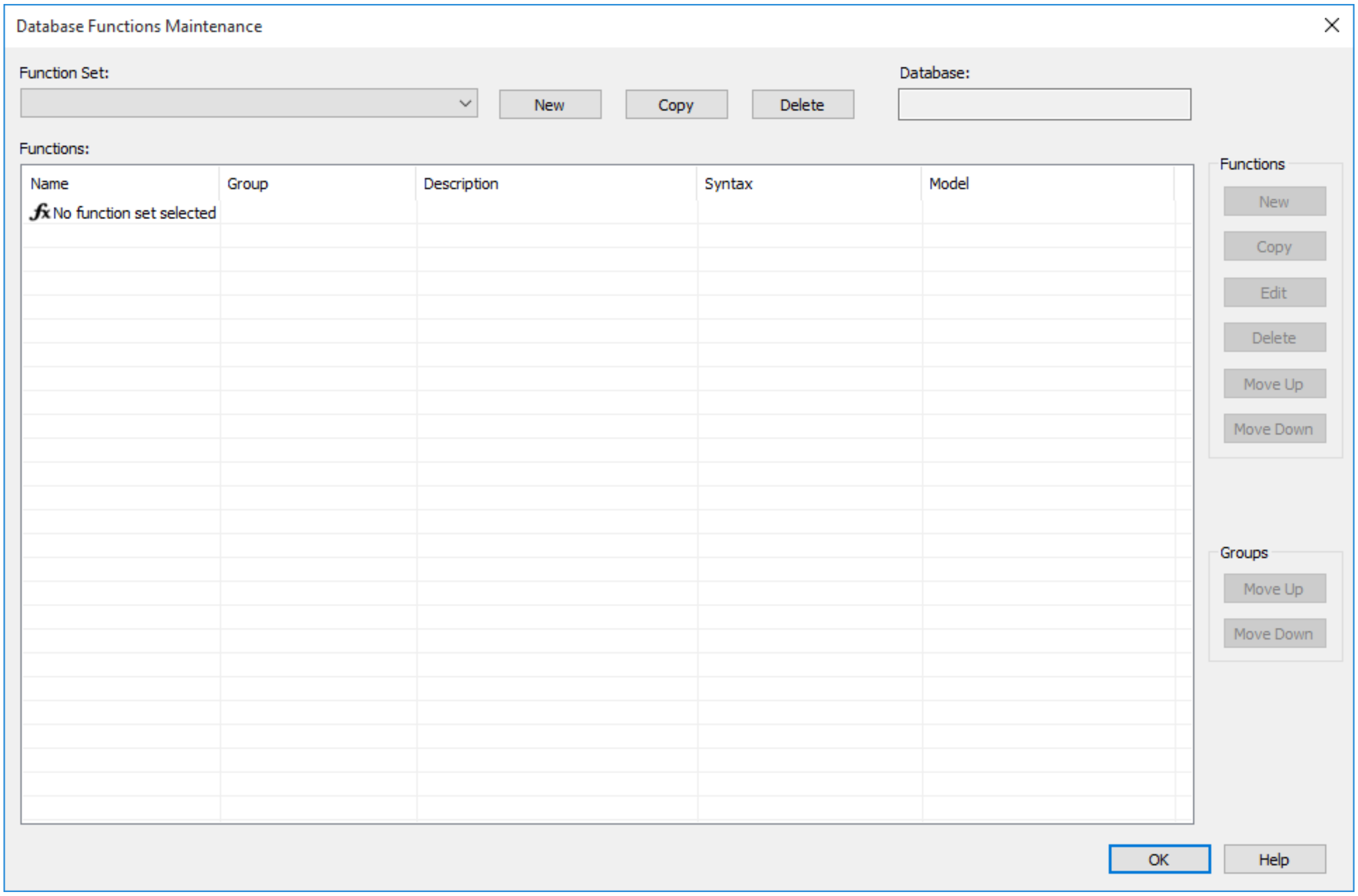
Default Transform Function Set
Function Set that is selected by default in the Transformation d

Maintaining Database Function Sets

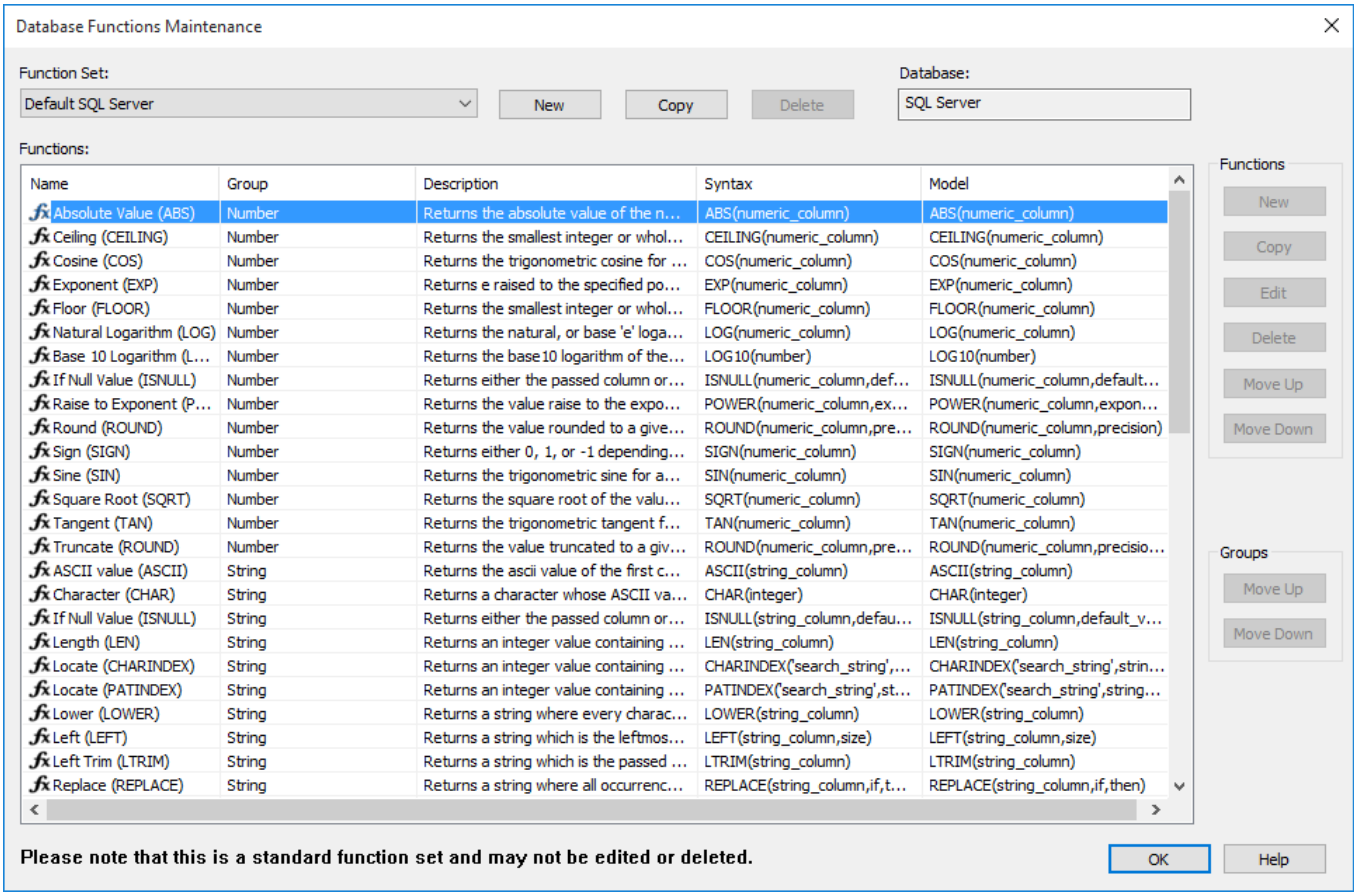
To maintain the database function sets, select **Tools > Database Functions > Maintain Database Functions**.



The following screen is displayed.



Select a database function set from the **Function Set** drop-down list.



Function Set: Default SQL Server

Database: SQL Server

Functions:

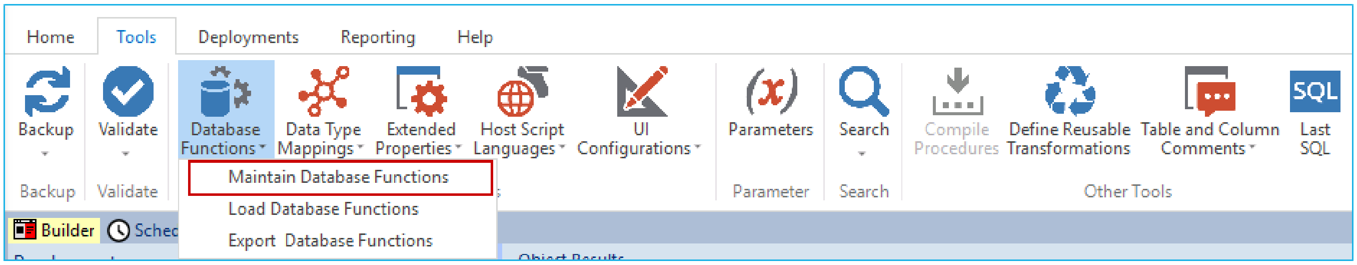
Name	Group	Description	Syntax	Model
Absolute Value (ABS)	Number	Returns the absolute value of the n…	ABS(numeric_column)	ABS(numeric_column)
Ceiling (CEILING)	Number	Returns the smallest integer or whol…	CEILING(numeric_column)	CEILING(numeric_column)
Cosine (COS)	Number	Returns the trigonometric cosine for …	COS(numeric_column)	COS(numeric_column)
Exponent (EXP)	Number	Returns e raised to the specified po…	EXP(numeric_column)	EXP(numeric_column)
Floor (FLOOR)	Number	Returns the smallest integer or whol…	FLOOR(numeric_column)	FLOOR(numeric_column)
Natural Logarithm (LOG)	Number	Returns the natural, or base 'e' loga…	LOG(numeric_column)	LOG(numeric_column)
Base 10 Logarithm (L…	Number	Returns the base10 logarithm of the…	LOG10(number)	LOG10(number)
If Null Value (ISNULL)	Number	Returns either the passed column or…	ISNULL(numeric_column,def…	ISNULL(numeric_column,default…
Raise to Exponent (P…	Number	Returns the value raise to the expo…	POWER(numeric_column,ex…	POWER(numeric_column,expon…
Round (ROUND)	Number	Returns the value rounded to a give…	ROUND(numeric_column,pre…	ROUND(numeric_column,precision)
Sign (SIGN)	Number	Returns either 0, 1, or -1 depending…	SIGN(numeric_column)	SIGN(numeric_column)
Sine (SIN)	Number	Returns the trigonometric sine for a…	SIN(numeric_column)	SIN(numeric_column)
Square Root (SQRT)	Number	Returns the square root of the valu…	SQRT(numeric_column)	SQRT(numeric_column)
Tangent (TAN)	Number	Returns the trigonometric tangent f…	TAN(numeric_column)	TAN(numeric_column)
Truncate (ROUND)	Number	Returns the value truncated to a giv…	ROUND(numeric_column,pre…	ROUND(numeric_column,precisio…
ASCII value (ASCII)	String	Returns the ascii value of the first c…	ASCII(string_column)	ASCII(string_column)
Character (CHAR)	String	Returns a character whose ASCII va…	CHAR(integer)	CHAR(integer)
If Null Value (ISNULL)	String	Returns either the passed column or…	ISNULL(string_column,defau…	ISNULL(string_column,default_v…
Length (LEN)	String	Returns an integer value containing …	LEN(string_column)	LEN(string_column)
Locate (CHARINDEX)	String	Returns an integer value containing …	CHARINDEX('search_string',…	CHARINDEX('search_string',strin…
Locate (PATINDEX)	String	Returns an integer value containing …	PATINDEX('search_string',st…	PATINDEX('search_string',string…
Lower (LOWER)	String	Returns a string where every charac…	LOWER(string_column)	LOWER(string_column)
Left (LEFT)	String	Returns a string which is the leftmos…	LEFT(string_column,size)	LEFT(string_column,size)
Left Trim (LTRIM)	String	Returns a string which is the passed …	LTRIM(string_column)	LTRIM(string_column)
Replace (REPLACE)	String	Returns a string where all occurrenc…	REPLACE(string_column,if,t…	REPLACE(string_column,if,then)

Please note that this is a standard function set and may not be edited or deleted.

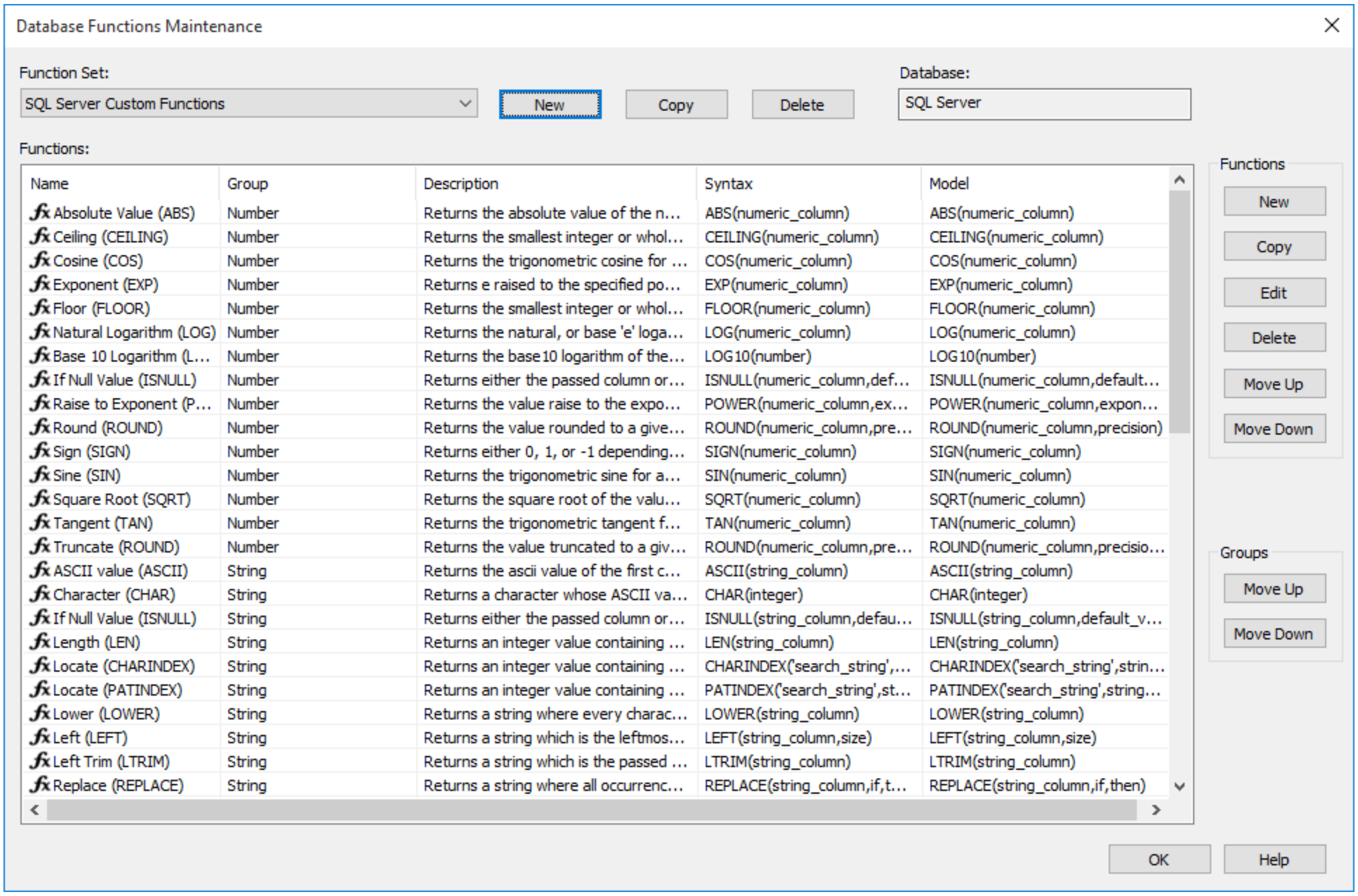
- To create a database function set, refer to Creating a New Database Function Set
- To copy a database function set, refer to Copying a Database Function Set
- To edit a database function set, refer to Editing a Database Function Set
- To delete a database function set, refer to Deleting a Database Function Set

Creating a New Database Function Set

To create a new database function set, select **Tools > Database Functions > Maintain Database Functions**.



Click the **New** button.

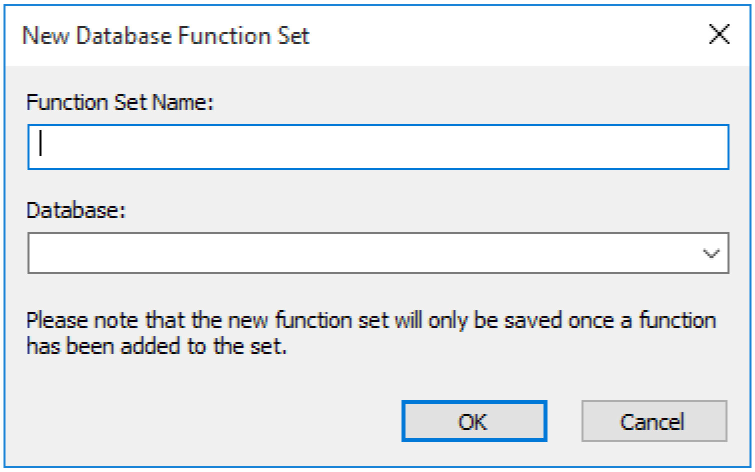


Function Set: SQL Server Custom Functions

Database: SQL Server

Name	Group	Description	Syntax	Model
Absolute Value (ABS)	Number	Returns the absolute value of the n...	ABS(numeric_column)	ABS(numeric_column)
Ceiling (CEILING)	Number	Returns the smallest integer or whol...	CEILING(numeric_column)	CEILING(numeric_column)
Cosine (COS)	Number	Returns the trigonometric cosine for ...	COS(numeric_column)	COS(numeric_column)
Exponent (EXP)	Number	Returns e raised to the specified po...	EXP(numeric_column)	EXP(numeric_column)
Floor (FLOOR)	Number	Returns the smallest integer or whol...	FLOOR(numeric_column)	FLOOR(numeric_column)
Natural Logarithm (LOG)	Number	Returns the natural, or base 'e' loga...	LOG(numeric_column)	LOG(numeric_column)
Base 10 Logarithm (L...	Number	Returns the base 10 logarithm of the...	LOG10(number)	LOG10(number)
If Null Value (ISNULL)	Number	Returns either the passed column or...	ISNULL(numeric_column,def...	ISNULL(numeric_column,default...
Raise to Exponent (P...	Number	Returns the value raise to the expo...	POWER(numeric_column,ex...	POWER(numeric_column,expon...
Round (ROUND)	Number	Returns the value rounded to a give...	ROUND(numeric_column,pre...	ROUND(numeric_column,precision)
Sign (SIGN)	Number	Returns either 0, 1, or -1 depending...	SIGN(numeric_column)	SIGN(numeric_column)
Sine (SIN)	Number	Returns the trigonometric sine for a...	SIN(numeric_column)	SIN(numeric_column)
Square Root (SQRT)	Number	Returns the square root of the valu...	SQRT(numeric_column)	SQRT(numeric_column)
Tangent (TAN)	Number	Returns the trigonometric tangent f...	TAN(numeric_column)	TAN(numeric_column)
Truncate (ROUND)	Number	Returns the value truncated to a giv...	ROUND(numeric_column,pre...	ROUND(numeric_column,precisio...
ASCII value (ASCII)	String	Returns the ascii value of the first c...	ASCII(string_column)	ASCII(string_column)
Character (CHAR)	String	Returns a character whose ASCII va...	CHAR(integer)	CHAR(integer)
If Null Value (ISNULL)	String	Returns either the passed column or...	ISNULL(string_column,defau...	ISNULL(string_column,default_v...
Length (LEN)	String	Returns an integer value containing ...	LEN(string_column)	LEN(string_column)
Locate (CHARINDEX)	String	Returns an integer value containing ...	CHARINDEX('search_string',...	CHARINDEX('search_string',strin...
Locate (PATINDEX)	String	Returns an integer value containing ...	PATINDEX('search_string',st...	PATINDEX('search_string',string...
Lower (LOWER)	String	Returns a string where every charac...	LOWER(string_column)	LOWER(string_column)
Left (LEFT)	String	Returns a string which is the leftmos...	LEFT(string_column,size)	LEFT(string_column,size)
Left Trim (LTRIM)	String	Returns a string which is the passed ...	LTRIM(string_column)	LTRIM(string_column)
Replace (REPLACE)	String	Returns a string where all occurrenc...	REPLACE(string_column,if,t...	REPLACE(string_column,if,then)

Enter a **Function Set Name** and select a **Database** from the drop-down list and then click **OK**.

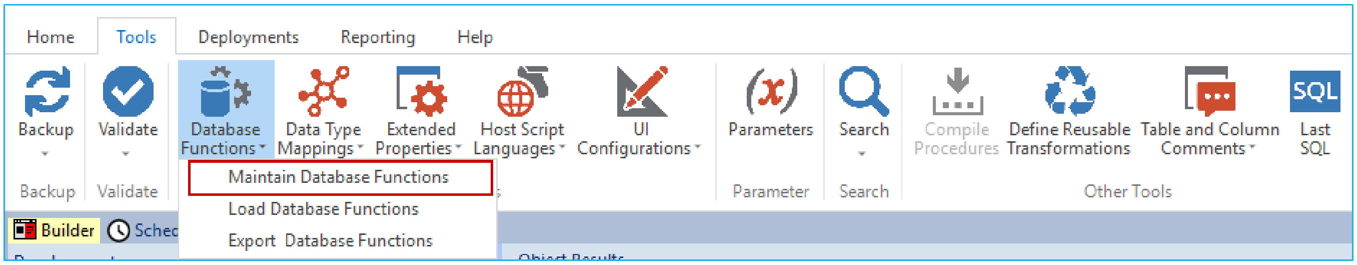


Please note that the new function set will only be saved once a function has been added to the set.

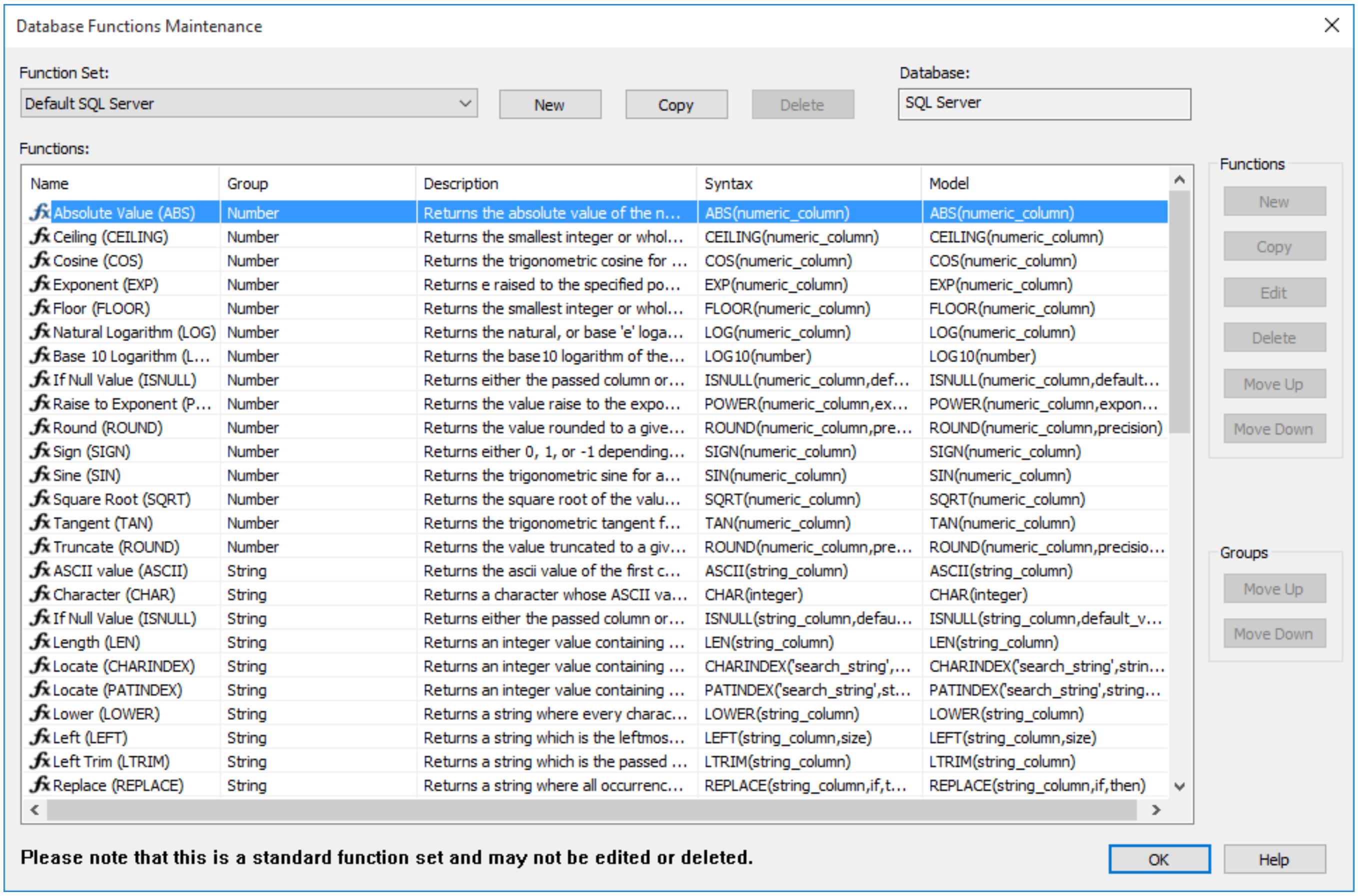
Field	Description
Function Set Name	Enter a unique function set name. **Note** The function set name may not contain the phrase "Standard Functions" as this is reserved for WhereScape supplied function sets.
Database	Select the database from the drop-down list or type in the name of the database if not already in the list. This field is mandatory.

Copying a Database Function Set

To copy an existing database function set, select **Tools > Database Functions > Maintain Database Functions**.



Select a **Function Set** from the drop-down list and then click the **Copy** button.



Function Set: Default SQL Server

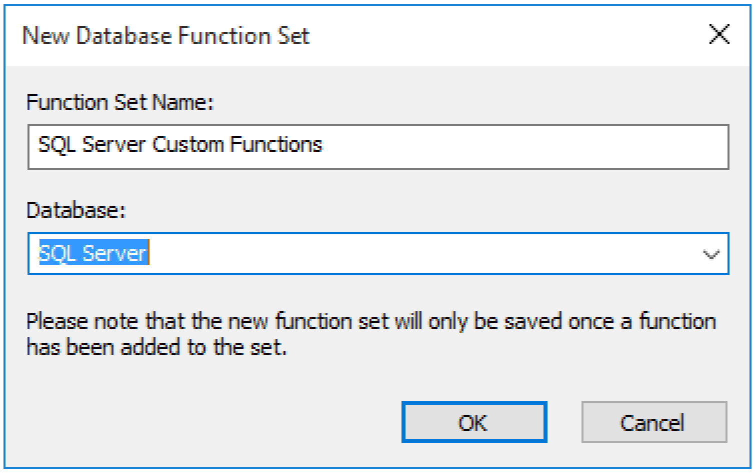
Database: SQL Server

Functions:

Name	Group	Description	Syntax	Model
Absolute Value (ABS)	Number	Returns the absolute value of the n…	ABS(numeric_column)	ABS(numeric_column)
Ceiling (CEILING)	Number	Returns the smallest integer or whol…	CEILING(numeric_column)	CEILING(numeric_column)
Cosine (COS)	Number	Returns the trigonometric cosine for …	COS(numeric_column)	COS(numeric_column)
Exponent (EXP)	Number	Returns e raised to the specified po…	EXP(numeric_column)	EXP(numeric_column)
Floor (FLOOR)	Number	Returns the smallest integer or whol…	FLOOR(numeric_column)	FLOOR(numeric_column)
Natural Logarithm (LOG)	Number	Returns the natural, or base 'e' loga…	LOG(numeric_column)	LOG(numeric_column)
Base 10 Logarithm (L…	Number	Returns the base10 logarithm of the…	LOG10(number)	LOG10(number)
If Null Value (ISNULL)	Number	Returns either the passed column or…	ISNULL(numeric_column,def…	ISNULL(numeric_column,default…
Raise to Exponent (P…	Number	Returns the value raise to the expo…	POWER(numeric_column,ex…	POWER(numeric_column,expon…
Round (ROUND)	Number	Returns the value rounded to a give…	ROUND(numeric_column,pre…	ROUND(numeric_column,precision)
Sign (SIGN)	Number	Returns either 0, 1, or -1 depending…	SIGN(numeric_column)	SIGN(numeric_column)
Sine (SIN)	Number	Returns the trigonometric sine for a…	SIN(numeric_column)	SIN(numeric_column)
Square Root (SQRT)	Number	Returns the square root of the valu…	SQRT(numeric_column)	SQRT(numeric_column)
Tangent (TAN)	Number	Returns the trigonometric tangent f…	TAN(numeric_column)	TAN(numeric_column)
Truncate (ROUND)	Number	Returns the value truncated to a giv…	ROUND(numeric_column,pre…	ROUND(numeric_column,precisio…
ASCII value (ASCII)	String	Returns the ascii value of the first c…	ASCII(string_column)	ASCII(string_column)
Character (CHAR)	String	Returns a character whose ASCII va…	CHAR(integer)	CHAR(integer)
If Null Value (ISNULL)	String	Returns either the passed column or…	ISNULL(string_column,defau…	ISNULL(string_column,default_v…
Length (LEN)	String	Returns an integer value containing …	LEN(string_column)	LEN(string_column)
Locate (CHARINDEX)	String	Returns an integer value containing …	CHARINDEX('search_string',…	CHARINDEX('search_string',strin…
Locate (PATINDEX)	String	Returns an integer value containing …	PATINDEX('search_string',st…	PATINDEX('search_string',string…
Lower (LOWER)	String	Returns a string where every charac…	LOWER(string_column)	LOWER(string_column)
Left (LEFT)	String	Returns a string which is the leftmos…	LEFT(string_column,size)	LEFT(string_column,size)
Left Trim (LTRIM)	String	Returns a string which is the passed …	LTRIM(string_column)	LTRIM(string_column)
Replace (REPLACE)	String	Returns a string where all occurrenc…	REPLACE(string_column,if,t…	REPLACE(string_column,if,then)

Please note that this is a standard function set and may not be edited or deleted.

Enter the new **Function Set Name** and select a **Database** from the drop-down list and then click **OK**.



New Database Function Set

Function Set Name: SQL Server Custom Functions

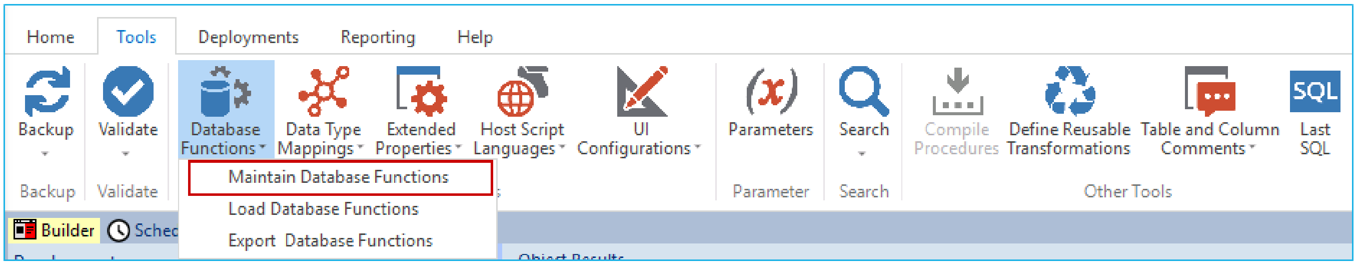
Database: SQL Server

Please note that the new function set will only be saved once a function has been added to the set.

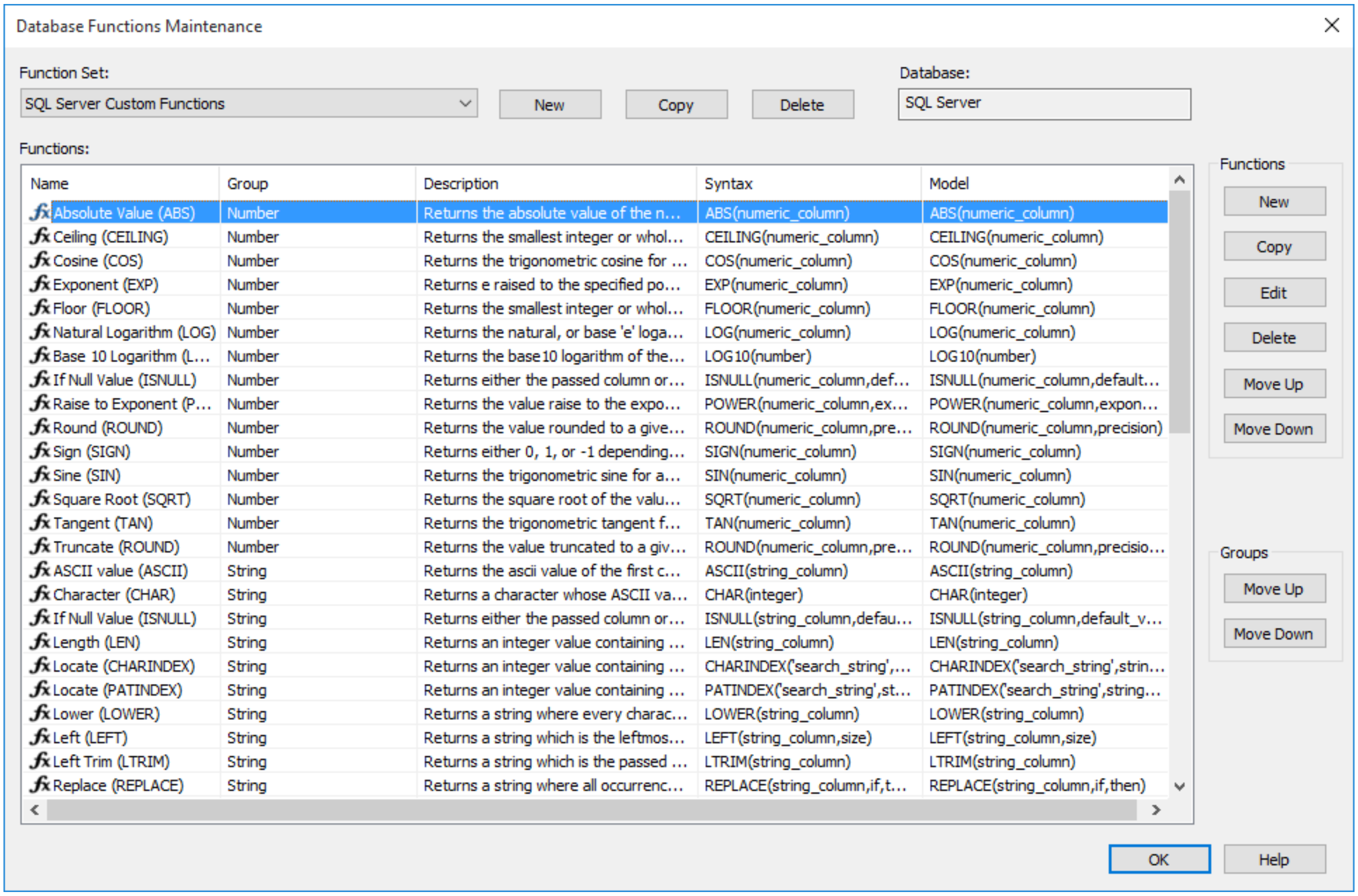
Field	Description
Function Set Name	Enter a unique function set name. **Note** The function set name may not contain the phrase "Standard Functions" as this is reserved for WhereScape supplied function sets.
Database	Select the database from the drop-down list or type in the name of the database if not already in the list. This field is mandatory.

Editing a Database Function Set

To edit a database function set, select **Tools > Database Functions > Maintain Database Functions**.



Select a database function set from the **Function Set** drop-down list.



Function Set: SQL Server Custom Functions

Database: SQL Server

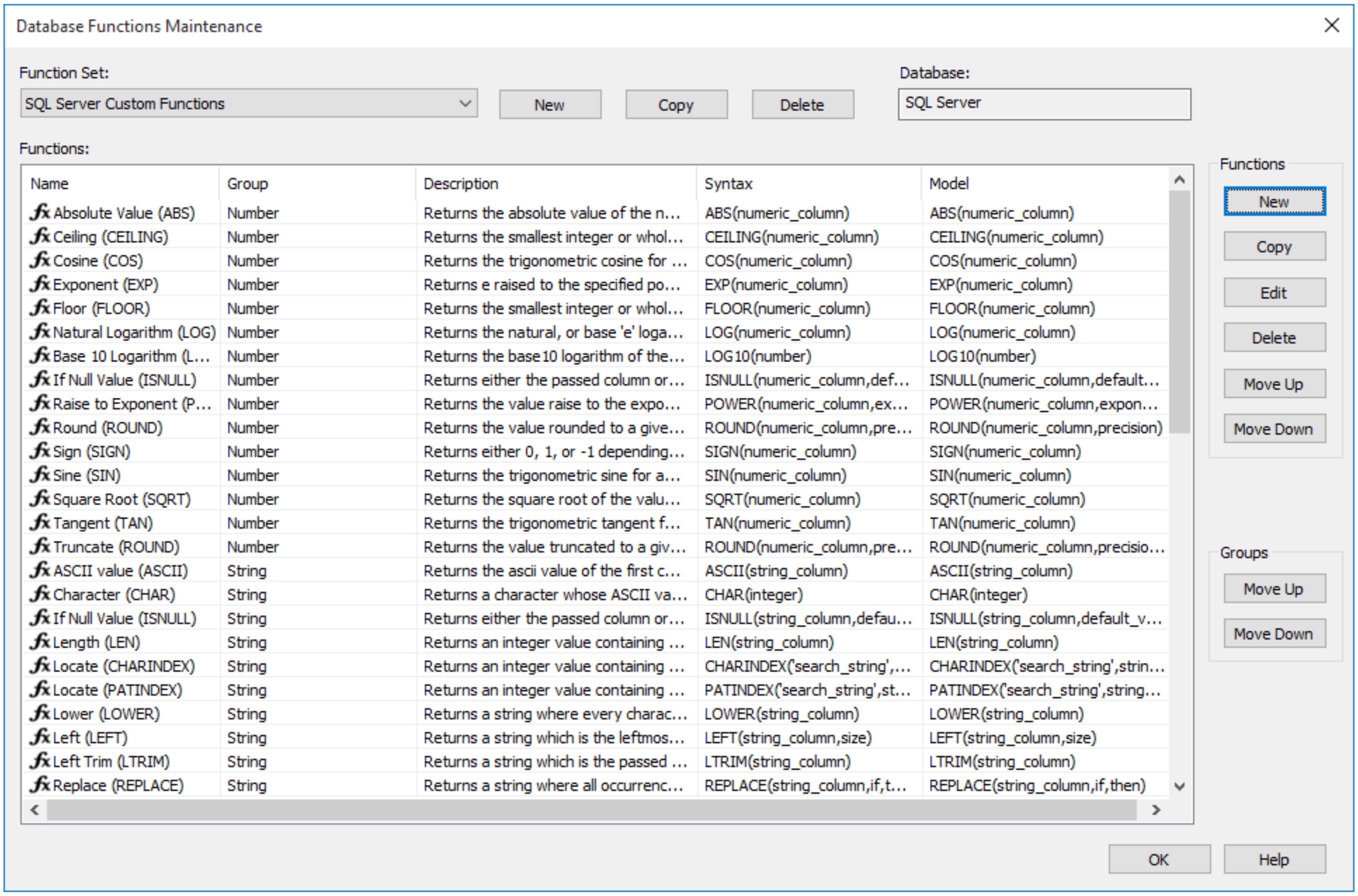
Functions:

Name	Group	Description	Syntax	Model
Absolute Value (ABS)	Number	Returns the absolute value of the n...	ABS(numeric_column)	ABS(numeric_column)
Ceiling (CEILING)	Number	Returns the smallest integer or whol...	CEILING(numeric_column)	CEILING(numeric_column)
Cosine (COS)	Number	Returns the trigonometric cosine for ...	COS(numeric_column)	COS(numeric_column)
Exponent (EXP)	Number	Returns e raised to the specified po...	EXP(numeric_column)	EXP(numeric_column)
Floor (FLOOR)	Number	Returns the smallest integer or whol...	FLOOR(numeric_column)	FLOOR(numeric_column)
Natural Logarithm (LOG)	Number	Returns the natural, or base 'e' loga...	LOG(numeric_column)	LOG(numeric_column)
Base 10 Logarithm (L...	Number	Returns the base10 logarithm of the...	LOG10(number)	LOG10(number)
If Null Value (ISNULL)	Number	Returns either the passed column or...	ISNULL(numeric_column,def...	ISNULL(numeric_column,default...
Raise to Exponent (P...	Number	Returns the value raise to the expo...	POWER(numeric_column,ex...	POWER(numeric_column,expon...
Round (ROUND)	Number	Returns the value rounded to a give...	ROUND(numeric_column,pre...	ROUND(numeric_column,precision)
Sign (SIGN)	Number	Returns either 0, 1, or -1 depending...	SIGN(numeric_column)	SIGN(numeric_column)
Sine (SIN)	Number	Returns the trigonometric sine for a...	SIN(numeric_column)	SIN(numeric_column)
Square Root (SQRT)	Number	Returns the square root of the valu...	SQRT(numeric_column)	SQRT(numeric_column)
Tangent (TAN)	Number	Returns the trigonometric tangent f...	TAN(numeric_column)	TAN(numeric_column)
Truncate (ROUND)	Number	Returns the value truncated to a giv...	ROUND(numeric_column,pre...	ROUND(numeric_column,precisio...
ASCII value (ASCII)	String	Returns the ascii value of the first c...	ASCII(string_column)	ASCII(string_column)
Character (CHAR)	String	Returns a character whose ASCII va...	CHAR(integer)	CHAR(integer)
If Null Value (ISNULL)	String	Returns either the passed column or...	ISNULL(string_column,defau...	ISNULL(string_column,default_v...
Length (LEN)	String	Returns an integer value containing ...	LEN(string_column)	LEN(string_column)
Locate (CHARINDEX)	String	Returns an integer value containing ...	CHARINDEX('search_string',...	CHARINDEX('search_string',strin...
Locate (PATINDEX)	String	Returns an integer value containing ...	PATINDEX('search_string',st...	PATINDEX('search_string',string...
Lower (LOWER)	String	Returns a string where every charac...	LOWER(string_column)	LOWER(string_column)
Left (LEFT)	String	Returns a string which is the leftmos...	LEFT(string_column,size)	LEFT(string_column,size)
Left Trim (LTRIM)	String	Returns a string which is the passed ...	LTRIM(string_column)	LTRIM(string_column)
Replace (REPLACE)	String	Returns a string where all occurrenc...	REPLACE(string_column,if,t...	REPLACE(string_column,if,then)

A group of buttons used to maintain the list of functions in a function set is available on the right side of the screen. These buttons are not available for standard function sets.

To add a new function to the database function set:

Click the **New** button.



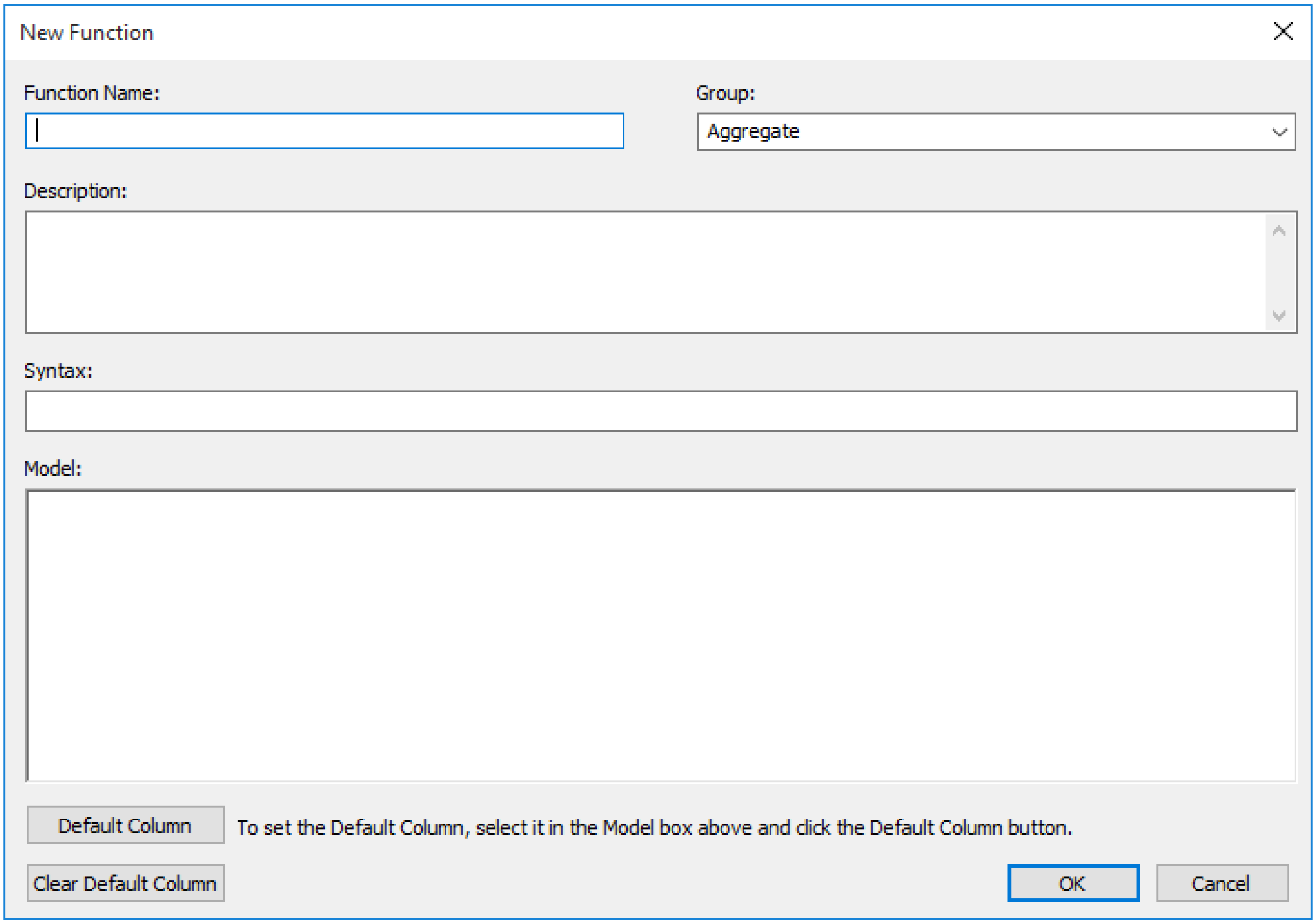
Function Set: SQL Server Custom Functions

Database: SQL Server

Functions:

Name	Group	Description	Syntax	Model
Absolute Value (ABS)	Number	Returns the absolute value of the n…	ABS(numeric_column)	ABS(numeric_column)
Ceiling (CEILING)	Number	Returns the smallest integer or whol…	CEILING(numeric_column)	CEILING(numeric_column)
Cosine (COS)	Number	Returns the trigonometric cosine for …	COS(numeric_column)	COS(numeric_column)
Exponent (EXP)	Number	Returns e raised to the specified po…	EXP(numeric_column)	EXP(numeric_column)
Floor (FLOOR)	Number	Returns the smallest integer or whol…	FLOOR(numeric_column)	FLOOR(numeric_column)
Natural Logarithm (LOG)	Number	Returns the natural, or base 'e' loga…	LOG(numeric_column)	LOG(numeric_column)
Base 10 Logarithm (L…	Number	Returns the base10 logarithm of the…	LOG10(number)	LOG10(number)
If Null Value (ISNULL)	Number	Returns either the passed column or…	ISNULL(numeric_column,def…	ISNULL(numeric_column,default…
Raise to Exponent (P…	Number	Returns the value raise to the expo…	POWER(numeric_column,ex…	POWER(numeric_column,expon…
Round (ROUND)	Number	Returns the value rounded to a give…	ROUND(numeric_column,pre…	ROUND(numeric_column,precision)
Sign (SIGN)	Number	Returns either 0, 1, or -1 depending…	SIGN(numeric_column)	SIGN(numeric_column)
Sine (SIN)	Number	Returns the trigonometric sine for a…	SIN(numeric_column)	SIN(numeric_column)
Square Root (SQRT)	Number	Returns the square root of the valu…	SQRT(numeric_column)	SQRT(numeric_column)
Tangent (TAN)	Number	Returns the trigonometric tangent f…	TAN(numeric_column)	TAN(numeric_column)
Truncate (ROUND)	Number	Returns the value truncated to a giv…	ROUND(numeric_column,pre…	ROUND(numeric_column,precisio…
ASCII value (ASCII)	String	Returns the ascii value of the first c…	ASCII(string_column)	ASCII(string_column)
Character (CHAR)	String	Returns a character whose ASCII va…	CHAR(integer)	CHAR(integer)
If Null Value (ISNULL)	String	Returns either the passed column or…	ISNULL(string_column,defau…	ISNULL(string_column,default_v…
Length (LEN)	String	Returns an integer value containing …	LEN(string_column)	LEN(string_column)
Locate (CHARINDEX)	String	Returns an integer value containing …	CHARINDEX('search_string',…	CHARINDEX('search_string',strin…
Locate (PATINDEX)	String	Returns an integer value containing …	PATINDEX('search_string',st…	PATINDEX('search_string',string…
Lower (LOWER)	String	Returns a string where every charac…	LOWER(string_column)	LOWER(string_column)
Left (LEFT)	String	Returns a string which is the leftmos…	LEFT(string_column,size)	LEFT(string_column,size)
Left Trim (LTRIM)	String	Returns a string which is the passed …	LTRIM(string_column)	LTRIM(string_column)
Replace (REPLACE)	String	Returns a string where all occurrenc…	REPLACE(string_column,if,t…	REPLACE(string_column,if,then)

Enter the details for the new function and then click **OK**.



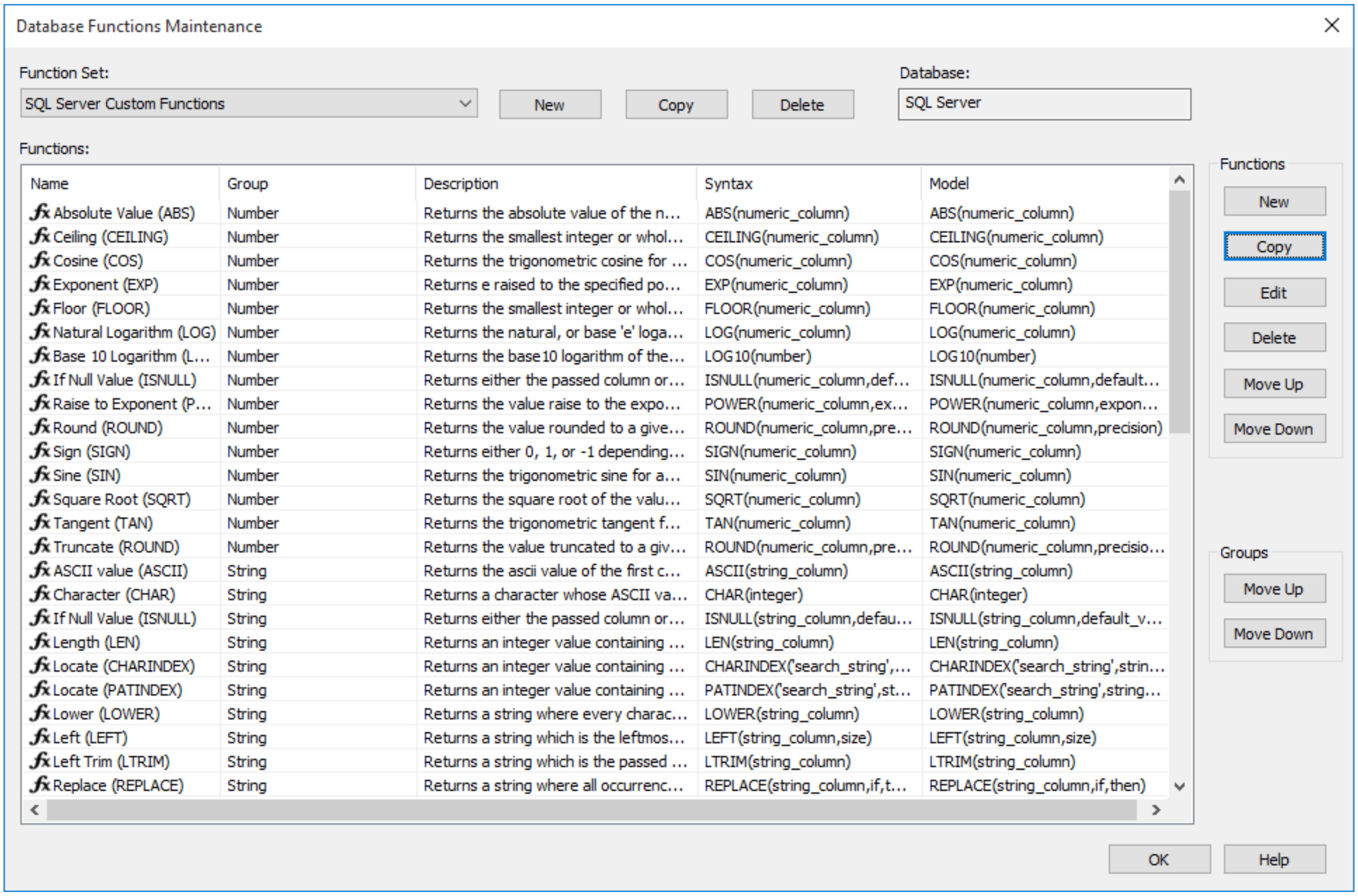
To set the Default Column, select it in the Model box above and click the Default Column button.

Field	Description
Function Name	This field is mandatory and must be unique within the group.
Group	Select a group from the drop-down list or add a new group name. This field is mandatory.
Description	Enter a description for the function.
Syntax	Enter the syntax for the function.
Model	This is the text that will be pasted into the transformation/ model fields when the function is selected. This field is mandatory.
Default Column	This is the text that will automatically be highlighted when the function is used in the transformation/model screens. To set the default column, highlight it in the model field and then click the **Default Column** button. The default column is shown in red in the **Model** field.

Field	Description
	Model: CHAR(*integer*) Default Column
Clear Default Column	Click this button to clear the default column.

To copy an existing function in the database function set:

Select the function and then click the **Copy** button.



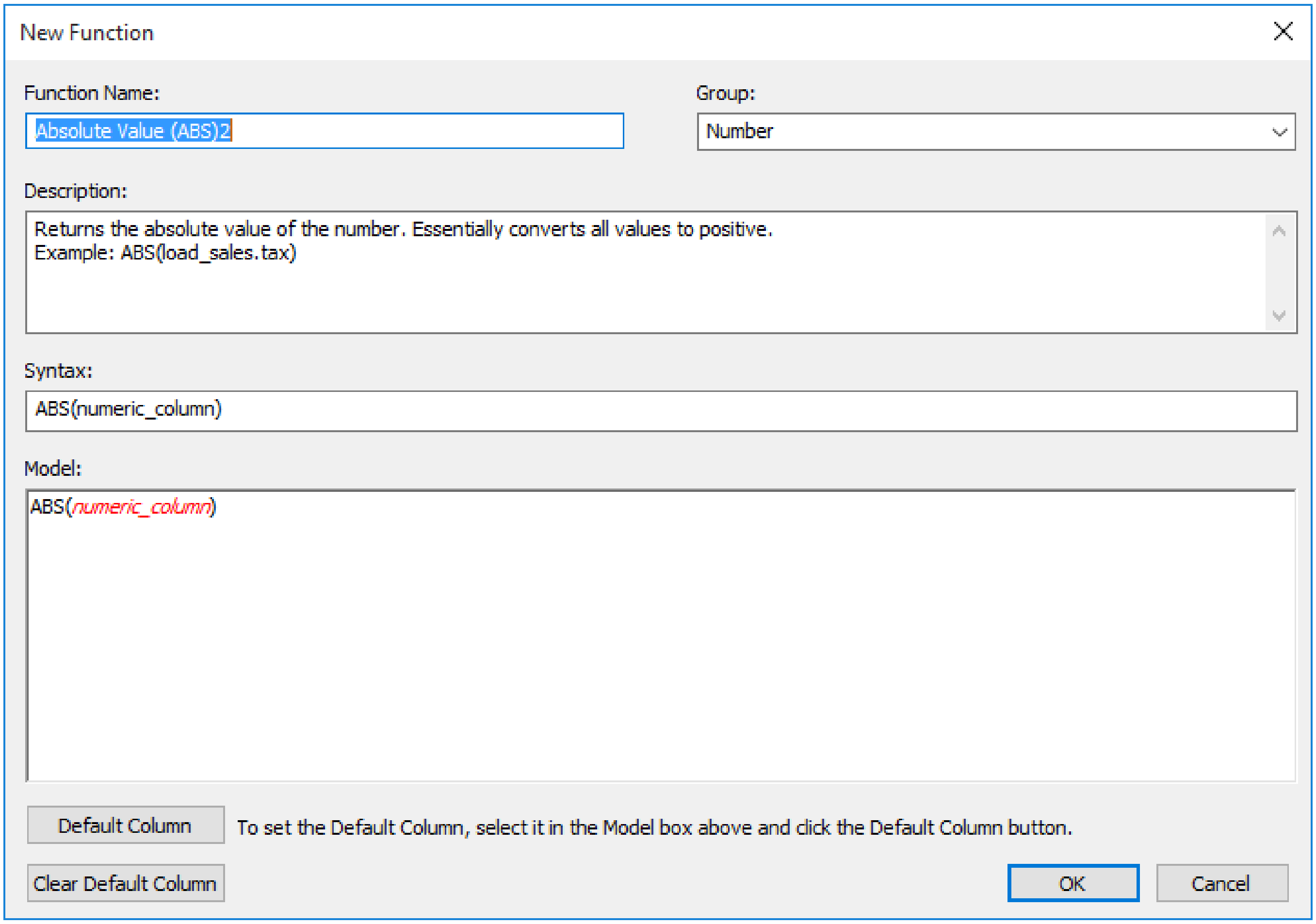
Function Set: SQL Server Custom Functions

Database: SQL Server

Functions:

Name	Group	Description	Syntax	Model
Absolute Value (ABS)	Number	Returns the absolute value of the n...	ABS(numeric_column)	ABS(numeric_column)
Ceiling (CEILING)	Number	Returns the smallest integer or whol...	CEILING(numeric_column)	CEILING(numeric_column)
Cosine (COS)	Number	Returns the trigonometric cosine for ...	COS(numeric_column)	COS(numeric_column)
Exponent (EXP)	Number	Returns e raised to the specified po...	EXP(numeric_column)	EXP(numeric_column)
Floor (FLOOR)	Number	Returns the smallest integer or whol...	FLOOR(numeric_column)	FLOOR(numeric_column)
Natural Logarithm (LOG)	Number	Returns the natural, or base 'e' loga...	LOG(numeric_column)	LOG(numeric_column)
Base 10 Logarithm (L...	Number	Returns the base10 logarithm of the...	LOG10(number)	LOG10(number)
If Null Value (ISNULL)	Number	Returns either the passed column or...	ISNULL(numeric_column,def...	ISNULL(numeric_column,default...
Raise to Exponent (P...	Number	Returns the value raise to the expo...	POWER(numeric_column,ex...	POWER(numeric_column,expon...
Round (ROUND)	Number	Returns the value rounded to a give...	ROUND(numeric_column,pre...	ROUND(numeric_column,precision)
Sign (SIGN)	Number	Returns either 0, 1, or -1 depending...	SIGN(numeric_column)	SIGN(numeric_column)
Sine (SIN)	Number	Returns the trigonometric sine for a...	SIN(numeric_column)	SIN(numeric_column)
Square Root (SQRT)	Number	Returns the square root of the valu...	SQRT(numeric_column)	SQRT(numeric_column)
Tangent (TAN)	Number	Returns the trigonometric tangent f...	TAN(numeric_column)	TAN(numeric_column)
Truncate (ROUND)	Number	Returns the value truncated to a giv...	ROUND(numeric_column,pre...	ROUND(numeric_column,precisio...
ASCII value (ASCII)	String	Returns the ascii value of the first c...	ASCII(string_column)	ASCII(string_column)
Character (CHAR)	String	Returns a character whose ASCII va...	CHAR(integer)	CHAR(integer)
If Null Value (ISNULL)	String	Returns either the passed column or...	ISNULL(string_column,defau...	ISNULL(string_column,default_v...
Length (LEN)	String	Returns an integer value containing ...	LEN(string_column)	LEN(string_column)
Locate (CHARINDEX)	String	Returns an integer value containing ...	CHARINDEX('search_string',...	CHARINDEX('search_string',strin...
Locate (PATINDEX)	String	Returns an integer value containing ...	PATINDEX('search_string',st...	PATINDEX('search_string',string...
Lower (LOWER)	String	Returns a string where every charac...	LOWER(string_column)	LOWER(string_column)
Left (LEFT)	String	Returns a string which is the leftmos...	LEFT(string_column,size)	LEFT(string_column,size)
Left Trim (LTRIM)	String	Returns a string which is the passed ...	LTRIM(string_column)	LTRIM(string_column)
Replace (REPLACE)	String	Returns a string where all occurrenc...	REPLACE(string_column,if,t...	REPLACE(string_column,if,then)

Enter a new **Function Name** and change any other details; then click **OK.**



New Function

Function Name: Absolute Value (ABS)2

Group: Number

Description:

Returns the absolute value of the number. Essentially converts all values to positive.
Example: ABS(load_sales.tax)

Syntax: ABS(numeric_column)

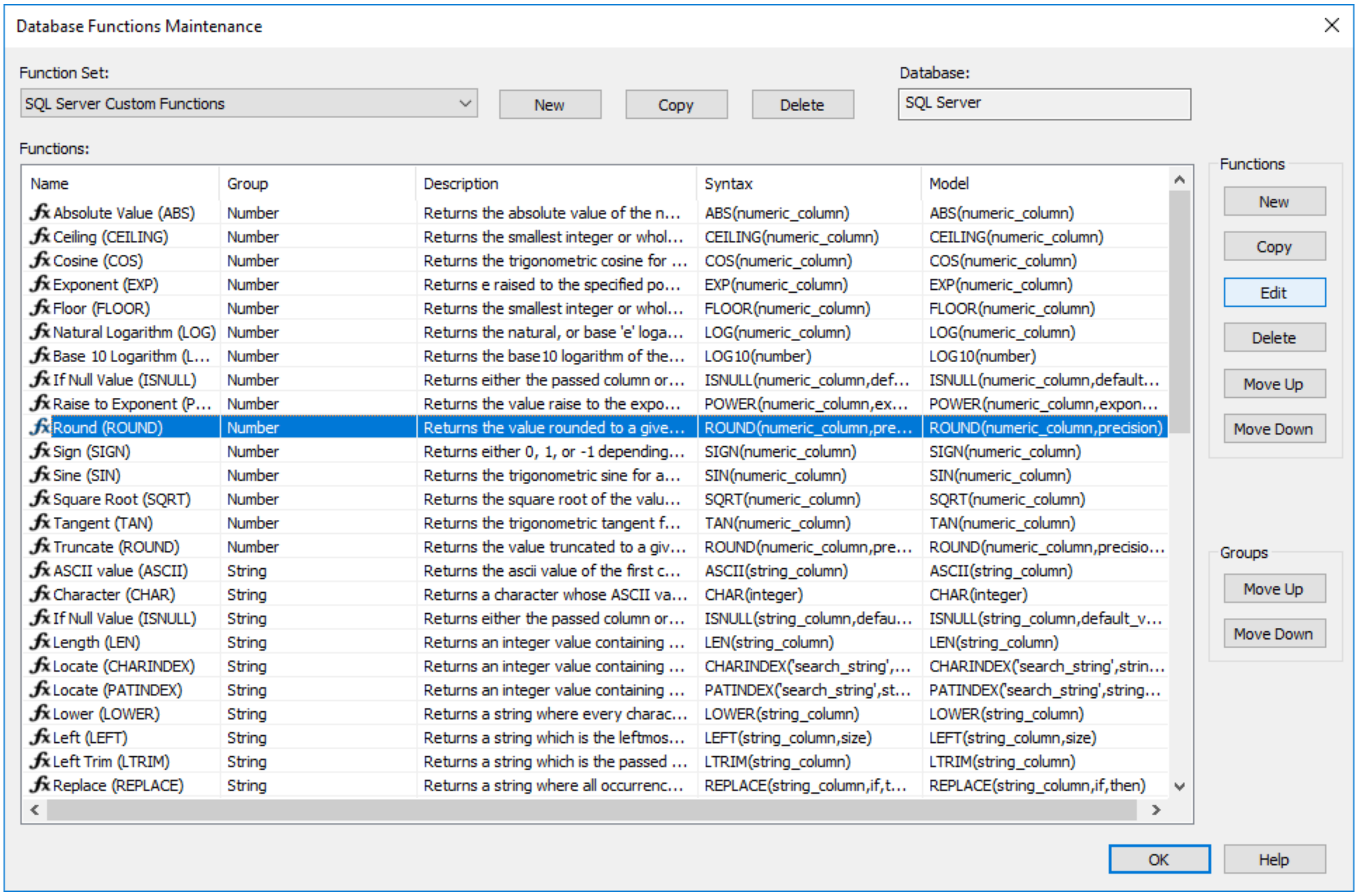
Model:

ABS(*numeric_column*)

To set the Default Column, select it in the Model box above and click the Default Column button.

To edit an existing function in the database function set:

Select the function and then click the **Edit** button on the right.



Database Functions Maintenance

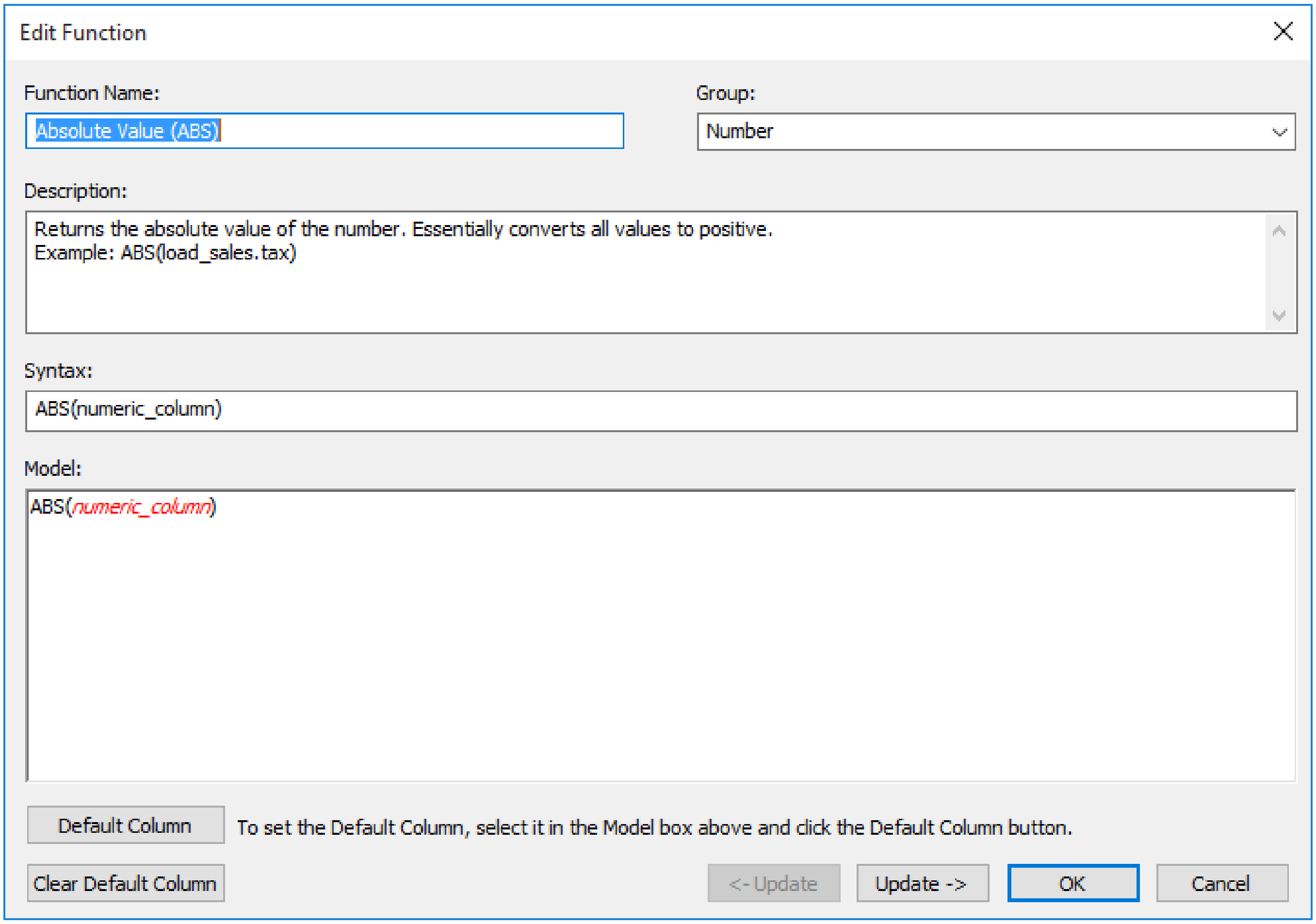
Function Set: SQL Server Custom Functions

Database: SQL Server

Functions:

Name	Group	Description	Syntax	Model
Absolute Value (ABS)	Number	Returns the absolute value of the n...	ABS(numeric_column)	ABS(numeric_column)
Ceiling (CEILING)	Number	Returns the smallest integer or whol...	CEILING(numeric_column)	CEILING(numeric_column)
Cosine (COS)	Number	Returns the trigonometric cosine for ...	COS(numeric_column)	COS(numeric_column)
Exponent (EXP)	Number	Returns e raised to the specified po...	EXP(numeric_column)	EXP(numeric_column)
Floor (FLOOR)	Number	Returns the smallest integer or whol...	FLOOR(numeric_column)	FLOOR(numeric_column)
Natural Logarithm (LOG)	Number	Returns the natural, or base 'e' loga...	LOG(numeric_column)	LOG(numeric_column)
Base 10 Logarithm (L...	Number	Returns the base10 logarithm of the...	LOG10(number)	LOG10(number)
If Null Value (ISNULL)	Number	Returns either the passed column or...	ISNULL(numeric_column,def...	ISNULL(numeric_column,default...
Raise to Exponent (P...	Number	Returns the value raise to the expo...	POWER(numeric_column,ex...	POWER(numeric_column,expon...
Round (ROUND)	Number	Returns the value rounded to a give...	ROUND(numeric_column,pre...	ROUND(numeric_column,precision)
Sign (SIGN)	Number	Returns either 0, 1, or -1 depending...	SIGN(numeric_column)	SIGN(numeric_column)
Sine (SIN)	Number	Returns the trigonometric sine for a...	SIN(numeric_column)	SIN(numeric_column)
Square Root (SQRT)	Number	Returns the square root of the valu...	SQRT(numeric_column)	SQRT(numeric_column)
Tangent (TAN)	Number	Returns the trigonometric tangent f...	TAN(numeric_column)	TAN(numeric_column)
Truncate (ROUND)	Number	Returns the value truncated to a giv...	ROUND(numeric_column,pre...	ROUND(numeric_column,precisio...
ASCII value (ASCII)	String	Returns the ascii value of the first c...	ASCII(string_column)	ASCII(string_column)
Character (CHAR)	String	Returns a character whose ASCII va...	CHAR(integer)	CHAR(integer)
If Null Value (ISNULL)	String	Returns either the passed column or...	ISNULL(string_column,defau...	ISNULL(string_column,default_v...
Length (LEN)	String	Returns an integer value containing ...	LEN(string_column)	LEN(string_column)
Locate (CHARINDEX)	String	Returns an integer value containing ...	CHARINDEX('search_string',...	CHARINDEX('search_string',strin...
Locate (PATINDEX)	String	Returns an integer value containing ...	PATINDEX('search_string',st...	PATINDEX('search_string',string...
Lower (LOWER)	String	Returns a string where every charac...	LOWER(string_column)	LOWER(string_column)
Left (LEFT)	String	Returns a string which is the leftmos...	LEFT(string_column,size)	LEFT(string_column,size)
Left Trim (LTRIM)	String	Returns a string which is the passed ...	LTRIM(string_column)	LTRIM(string_column)
Replace (REPLACE)	String	Returns a string where all occurrenc...	REPLACE(string_column,if,t...	REPLACE(string_column,if,then)

Change any fields as required and then click **OK**.


Edit Function

Function Name: Absolute Value (ABS)

Group: Number

Description:
Returns the absolute value of the number. Essentially converts all values to positive.
Example: ABS(load_sales.tax)

Syntax: ABS(numeric_column)

Model:
ABS(*numeric_column*)

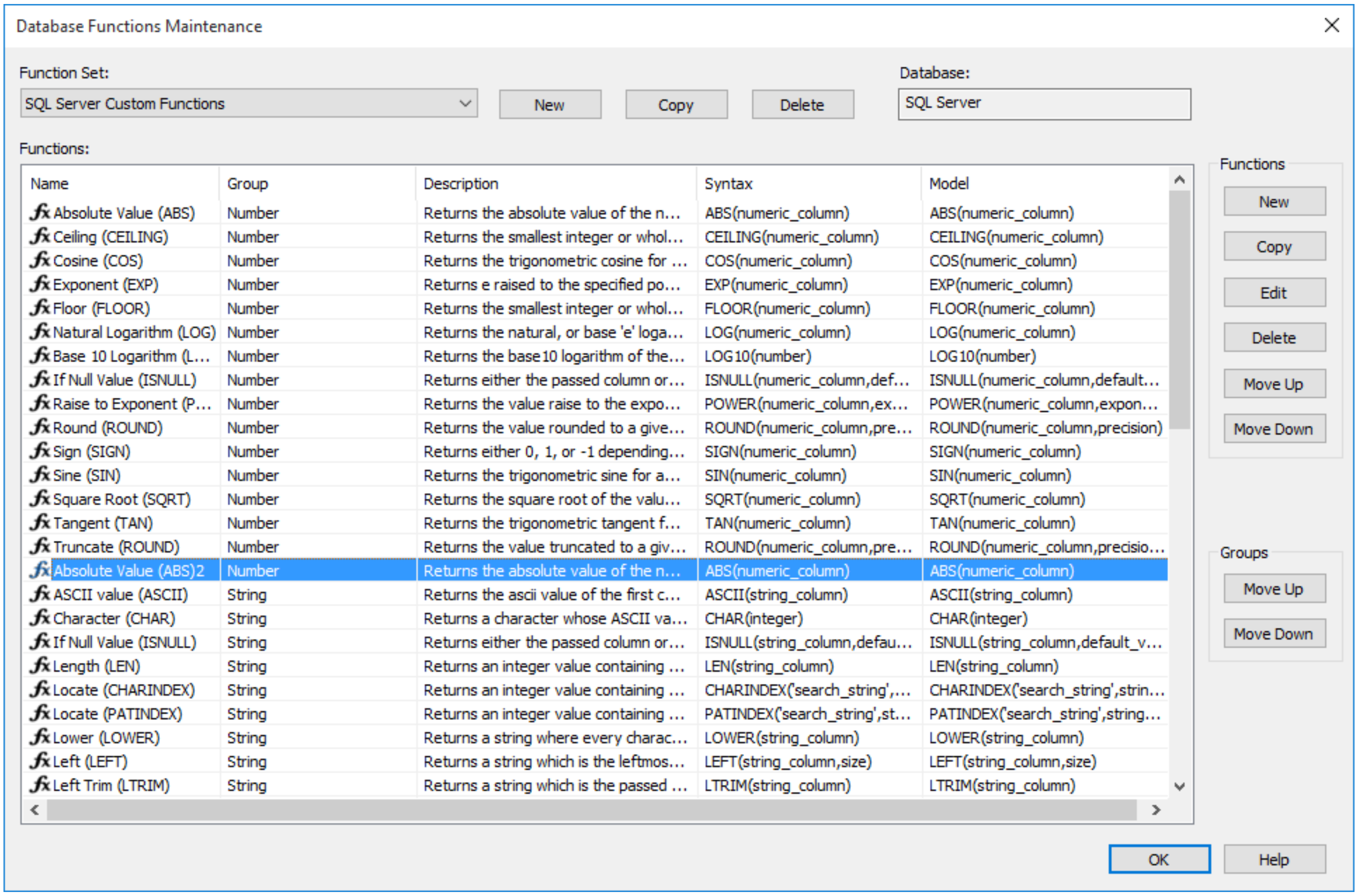
To set the Default Column, select it in the Model box above and click the Default Column button.

Note

A function can also be edited by double-clicking the function.

To delete an existing function in the database function set:

Select the function and then click the **Delete** button.



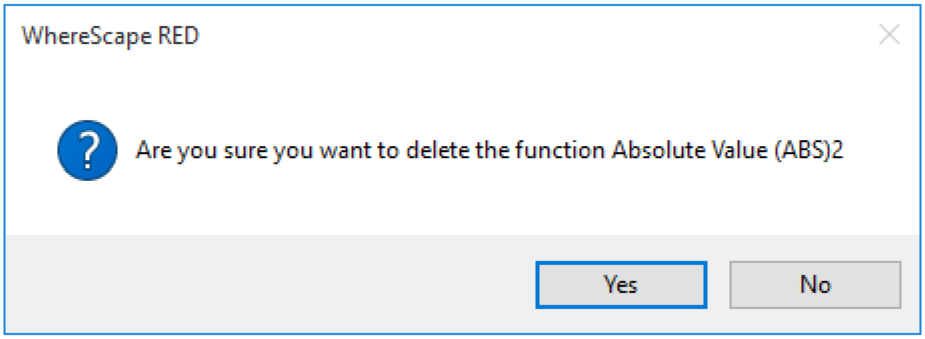
Function Set: SQL Server Custom Functions

Database: SQL Server

Functions:

Name	Group	Description	Syntax	Model
Absolute Value (ABS)	Number	Returns the absolute value of the n...	ABS(numeric_column)	ABS(numeric_column)
Ceiling (CEILING)	Number	Returns the smallest integer or whol...	CEILING(numeric_column)	CEILING(numeric_column)
Cosine (COS)	Number	Returns the trigonometric cosine for ...	COS(numeric_column)	COS(numeric_column)
Exponent (EXP)	Number	Returns e raised to the specified po...	EXP(numeric_column)	EXP(numeric_column)
Floor (FLOOR)	Number	Returns the smallest integer or whol...	FLOOR(numeric_column)	FLOOR(numeric_column)
Natural Logarithm (LOG)	Number	Returns the natural, or base 'e' loga...	LOG(numeric_column)	LOG(numeric_column)
Base 10 Logarithm (L...	Number	Returns the base10 logarithm of the...	LOG10(number)	LOG10(number)
If Null Value (ISNULL)	Number	Returns either the passed column or...	ISNULL(numeric_column,def...	ISNULL(numeric_column,default...
Raise to Exponent (P...	Number	Returns the value raise to the expo...	POWER(numeric_column,ex...	POWER(numeric_column,expon...
Round (ROUND)	Number	Returns the value rounded to a give...	ROUND(numeric_column,pre...	ROUND(numeric_column,precision)
Sign (SIGN)	Number	Returns either 0, 1, or -1 depending...	SIGN(numeric_column)	SIGN(numeric_column)
Sine (SIN)	Number	Returns the trigonometric sine for a...	SIN(numeric_column)	SIN(numeric_column)
Square Root (SQRT)	Number	Returns the square root of the valu...	SQRT(numeric_column)	SQRT(numeric_column)
Tangent (TAN)	Number	Returns the trigonometric tangent f...	TAN(numeric_column)	TAN(numeric_column)
Truncate (ROUND)	Number	Returns the value truncated to a giv...	ROUND(numeric_column,pre...	ROUND(numeric_column,precisio...
Absolute Value (ABS)2	Number	Returns the absolute value of the n...	ABS(numeric_column)	ABS(numeric_column)
ASCII value (ASCII)	String	Returns the ascii value of the first c...	ASCII(string_column)	ASCII(string_column)
Character (CHAR)	String	Returns a character whose ASCII va...	CHAR(integer)	CHAR(integer)
If Null Value (ISNULL)	String	Returns either the passed column or...	ISNULL(string_column,defau...	ISNULL(string_column,default_v...
Length (LEN)	String	Returns an integer value containing ...	LEN(string_column)	LEN(string_column)
Locate (CHARINDEX)	String	Returns an integer value containing ...	CHARINDEX('search_string',...	CHARINDEX('search_string',strin...
Locate (PATINDEX)	String	Returns an integer value containing ...	PATINDEX('search_string',st...	PATINDEX('search_string',string...
Lower (LOWER)	String	Returns a string where every charac...	LOWER(string_column)	LOWER(string_column)
Left (LEFT)	String	Returns a string which is the leftmos...	LEFT(string_column,size)	LEFT(string_column,size)
Left Trim (LTRIM)	String	Returns a string which is the passed ...	LTRIM(string_column)	LTRIM(string_column)

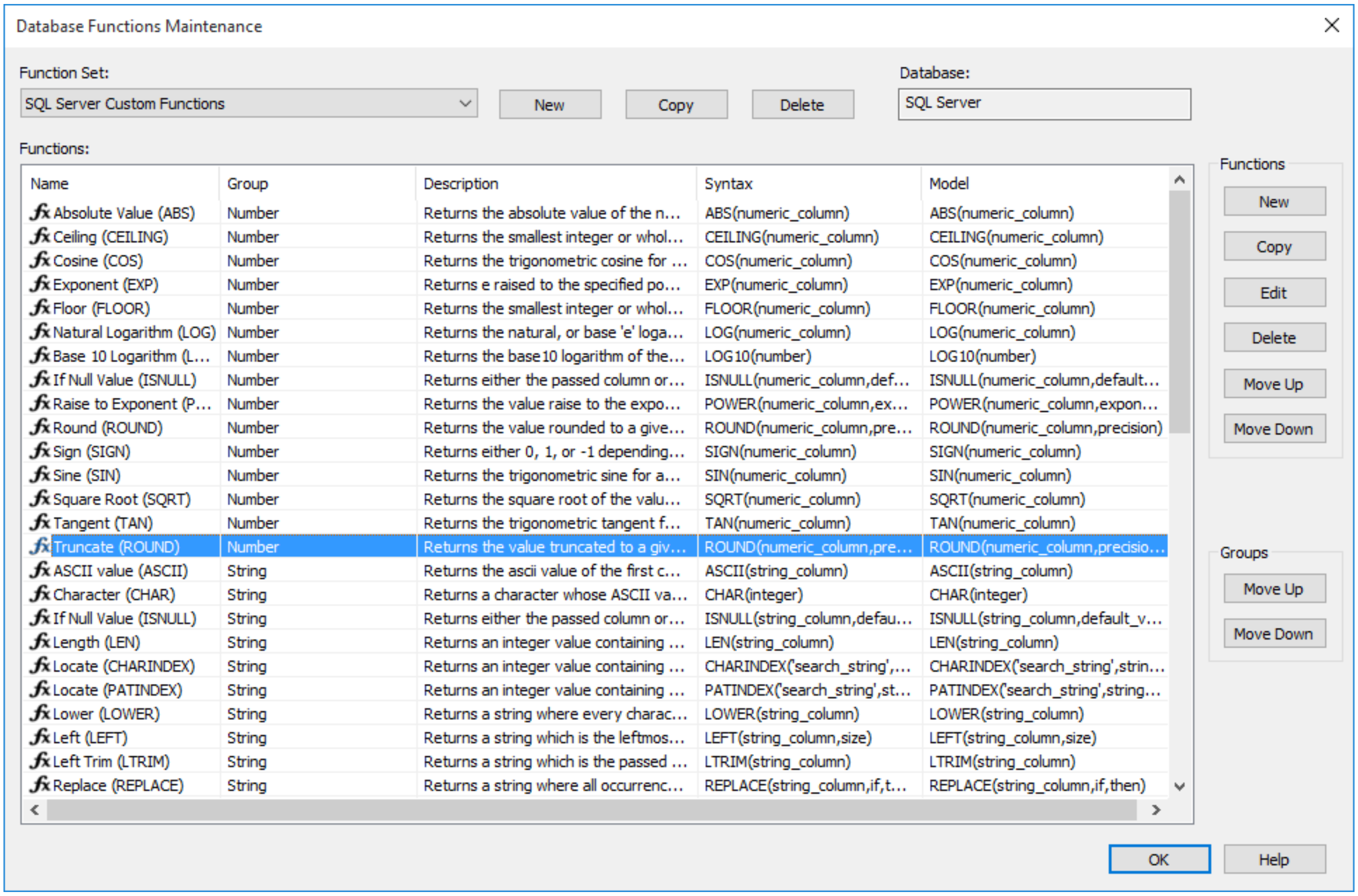
Click **Yes** to confirm the delete.



Are you sure you want to delete the function Absolute Value (ABS)2

To move a function in the database function set up or down in the list:

Select a function and then click the **Move Up** button on the right to move the function up in the function list, within its group; else click the **Move Down** button on the right to move the function down in the function list, within its group.



Function Set: SQL Server Custom Functions

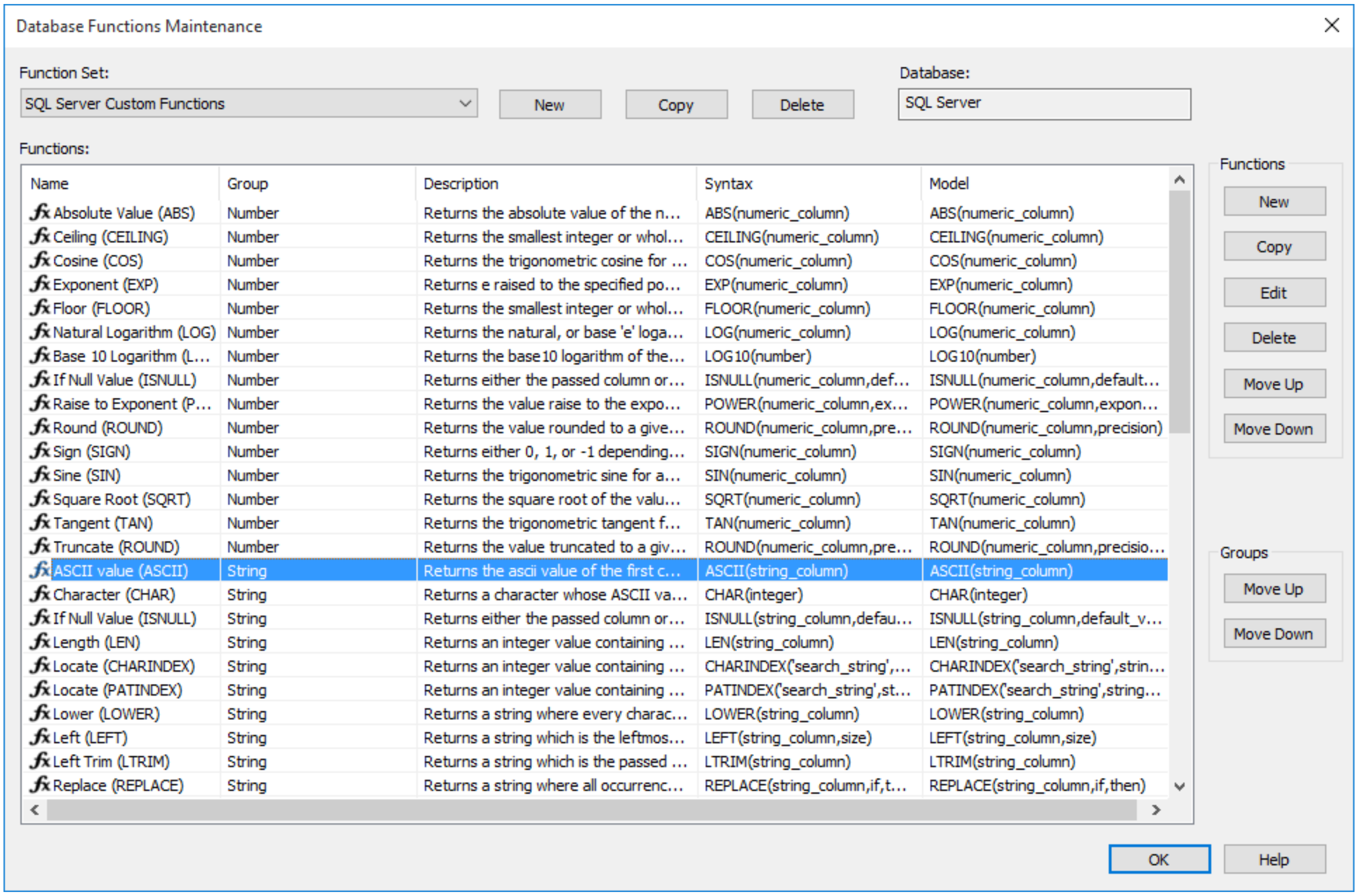
Database: SQL Server

Functions:

Name	Group	Description	Syntax	Model
Absolute Value (ABS)	Number	Returns the absolute value of the n...	ABS(numeric_column)	ABS(numeric_column)
Ceiling (CEILING)	Number	Returns the smallest integer or whol...	CEILING(numeric_column)	CEILING(numeric_column)
Cosine (COS)	Number	Returns the trigonometric cosine for ...	COS(numeric_column)	COS(numeric_column)
Exponent (EXP)	Number	Returns e raised to the specified po...	EXP(numeric_column)	EXP(numeric_column)
Floor (FLOOR)	Number	Returns the smallest integer or whol...	FLOOR(numeric_column)	FLOOR(numeric_column)
Natural Logarithm (LOG)	Number	Returns the natural, or base 'e' loga...	LOG(numeric_column)	LOG(numeric_column)
Base 10 Logarithm (L...	Number	Returns the base10 logarithm of the...	LOG10(number)	LOG10(number)
If Null Value (ISNULL)	Number	Returns either the passed column or...	ISNULL(numeric_column,def...	ISNULL(numeric_column,default...
Raise to Exponent (P...	Number	Returns the value raise to the expo...	POWER(numeric_column,ex...	POWER(numeric_column,expon...
Round (ROUND)	Number	Returns the value rounded to a give...	ROUND(numeric_column,pre...	ROUND(numeric_column,precision)
Sign (SIGN)	Number	Returns either 0, 1, or -1 depending...	SIGN(numeric_column)	SIGN(numeric_column)
Sine (SIN)	Number	Returns the trigonometric sine for a...	SIN(numeric_column)	SIN(numeric_column)
Square Root (SQRT)	Number	Returns the square root of the valu...	SQRT(numeric_column)	SQRT(numeric_column)
Tangent (TAN)	Number	Returns the trigonometric tangent f...	TAN(numeric_column)	TAN(numeric_column)
Truncate (ROUND)	Number	Returns the value truncated to a giv...	ROUND(numeric_column,pre...	ROUND(numeric_column,precisio...
ASCII value (ASCII)	String	Returns the ascii value of the first c...	ASCII(string_column)	ASCII(string_column)
Character (CHAR)	String	Returns a character whose ASCII va...	CHAR(integer)	CHAR(integer)
If Null Value (ISNULL)	String	Returns either the passed column or...	ISNULL(string_column,defau...	ISNULL(string_column,default_v...
Length (LEN)	String	Returns an integer value containing ...	LEN(string_column)	LEN(string_column)
Locate (CHARINDEX)	String	Returns an integer value containing ...	CHARINDEX('search_string',...	CHARINDEX('search_string',strin...
Locate (PATINDEX)	String	Returns an integer value containing ...	PATINDEX('search_string',st...	PATINDEX('search_string',string...
Lower (LOWER)	String	Returns a string where every charac...	LOWER(string_column)	LOWER(string_column)
Left (LEFT)	String	Returns a string which is the leftmos...	LEFT(string_column,size)	LEFT(string_column,size)
Left Trim (LTRIM)	String	Returns a string which is the passed ...	LTRIM(string_column)	LTRIM(string_column)
Replace (REPLACE)	String	Returns a string where all occurrenc...	REPLACE(string_column,if,t...	REPLACE(string_column,if,then)

To move a group of functions in the database function set up or down in the list:

Under the **Group** column, select any function in a particular group and then click the **Move Up** button under the **Groups** heading on the right to move the function group up in the function list. Similarly, use the **Move Down** button under the Groups heading to move a function group down in the function list.



Database Functions Maintenance

Function Set: SQL Server Custom Functions

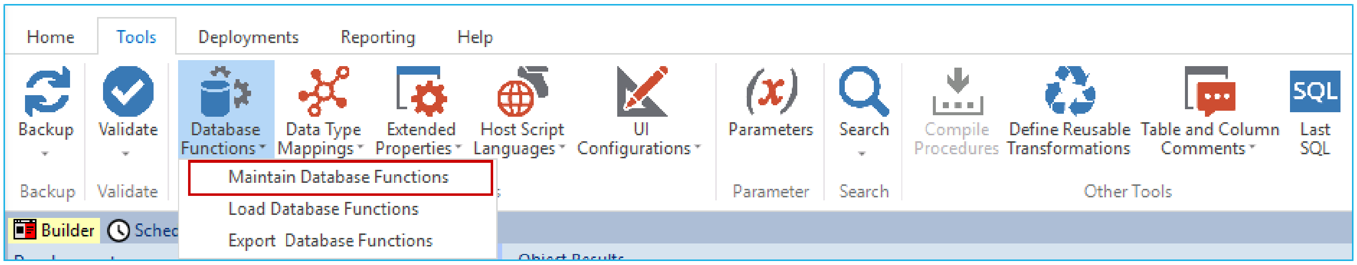
Database: SQL Server

Functions:

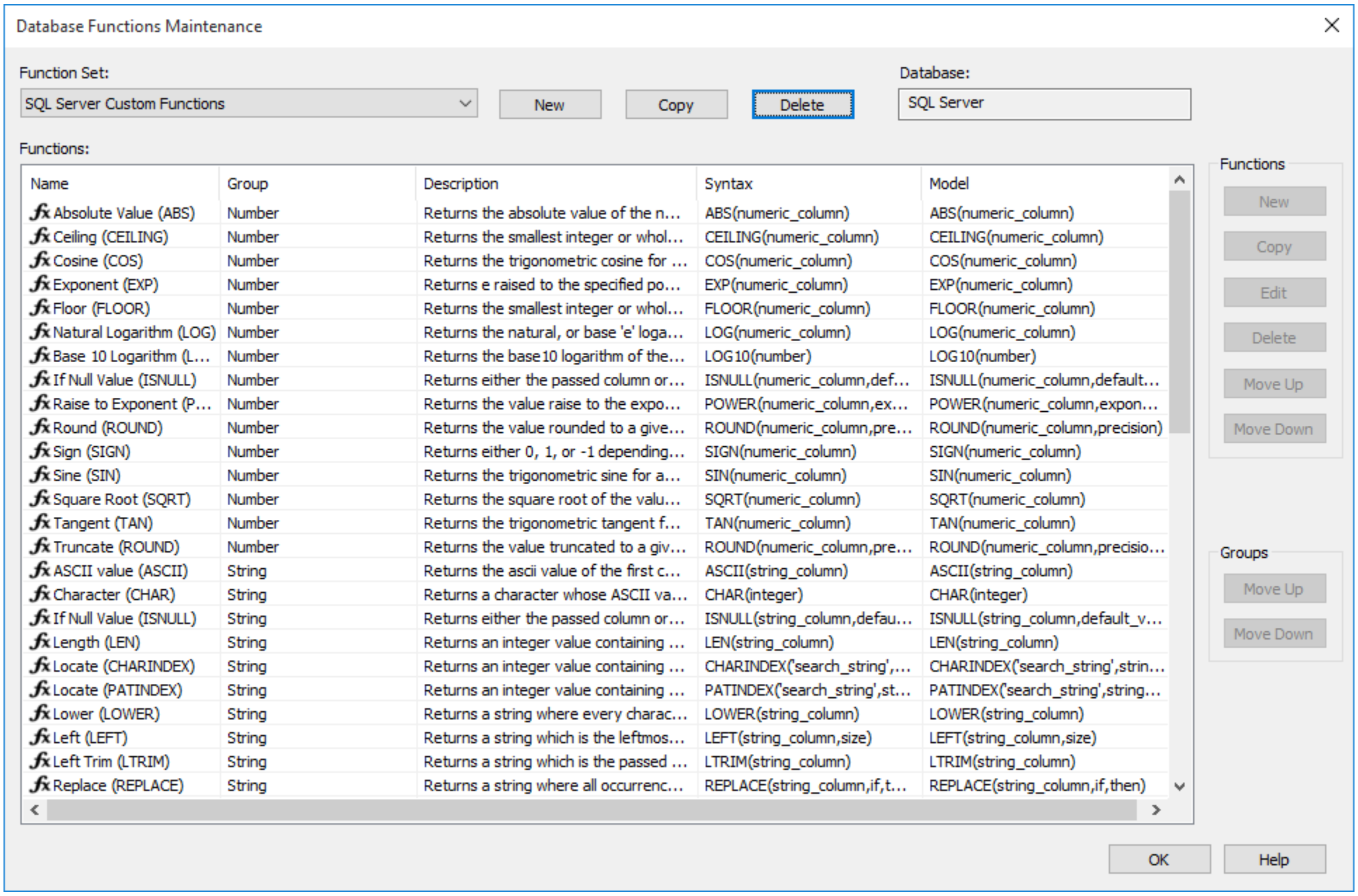
Name	Group	Description	Syntax	Model
Absolute Value (ABS)	Number	Returns the absolute value of the n…	ABS(numeric_column)	ABS(numeric_column)
Ceiling (CEILING)	Number	Returns the smallest integer or whol…	CEILING(numeric_column)	CEILING(numeric_column)
Cosine (COS)	Number	Returns the trigonometric cosine for …	COS(numeric_column)	COS(numeric_column)
Exponent (EXP)	Number	Returns e raised to the specified po…	EXP(numeric_column)	EXP(numeric_column)
Floor (FLOOR)	Number	Returns the smallest integer or whol…	FLOOR(numeric_column)	FLOOR(numeric_column)
Natural Logarithm (LOG)	Number	Returns the natural, or base 'e' loga…	LOG(numeric_column)	LOG(numeric_column)
Base 10 Logarithm (L…	Number	Returns the base10 logarithm of the…	LOG10(number)	LOG10(number)
If Null Value (ISNULL)	Number	Returns either the passed column or…	ISNULL(numeric_column,def…	ISNULL(numeric_column,default…
Raise to Exponent (P…	Number	Returns the value raise to the expo…	POWER(numeric_column,ex…	POWER(numeric_column,expon…
Round (ROUND)	Number	Returns the value rounded to a give…	ROUND(numeric_column,pre…	ROUND(numeric_column,precision)
Sign (SIGN)	Number	Returns either 0, 1, or -1 depending…	SIGN(numeric_column)	SIGN(numeric_column)
Sine (SIN)	Number	Returns the trigonometric sine for a…	SIN(numeric_column)	SIN(numeric_column)
Square Root (SQRT)	Number	Returns the square root of the valu…	SQRT(numeric_column)	SQRT(numeric_column)
Tangent (TAN)	Number	Returns the trigonometric tangent f…	TAN(numeric_column)	TAN(numeric_column)
Truncate (ROUND)	Number	Returns the value truncated to a giv…	ROUND(numeric_column,pre…	ROUND(numeric_column,precisio…
ASCII value (ASCII)	String	Returns the ascii value of the first c…	ASCII(string_column)	ASCII(string_column)
Character (CHAR)	String	Returns a character whose ASCII va…	CHAR(integer)	CHAR(integer)
If Null Value (ISNULL)	String	Returns either the passed column or…	ISNULL(string_column,defau…	ISNULL(string_column,default_v…
Length (LEN)	String	Returns an integer value containing …	LEN(string_column)	LEN(string_column)
Locate (CHARINDEX)	String	Returns an integer value containing …	CHARINDEX('search_string',…	CHARINDEX('search_string',strin…
Locate (PATINDEX)	String	Returns an integer value containing …	PATINDEX('search_string',st…	PATINDEX('search_string',string…
Lower (LOWER)	String	Returns a string where every charac…	LOWER(string_column)	LOWER(string_column)
Left (LEFT)	String	Returns a string which is the leftmos…	LEFT(string_column,size)	LEFT(string_column,size)
Left Trim (LTRIM)	String	Returns a string which is the passed …	LTRIM(string_column)	LTRIM(string_column)
Replace (REPLACE)	String	Returns a string where all occurrenc…	REPLACE(string_column,if,t…	REPLACE(string_column,if,then)

Deleting a Database Function Set

To delete a database function set, select **Tools > Database Functions > Maintain Database Functions**.



Select a **Function Set** from the drop-down list and then click the **Delete** button.



Function Set: SQL Server Custom Functions

Database: SQL Server

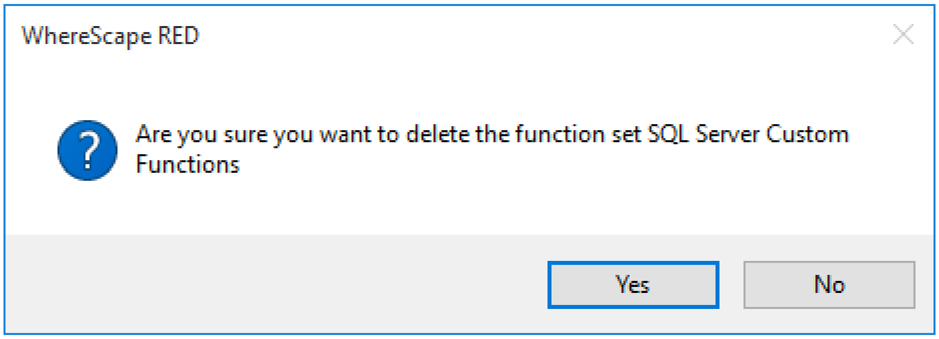
Functions:

Name	Group	Description	Syntax	Model
Absolute Value (ABS)	Number	Returns the absolute value of the n…	ABS(numeric_column)	ABS(numeric_column)
Ceiling (CEILING)	Number	Returns the smallest integer or whol…	CEILING(numeric_column)	CEILING(numeric_column)
Cosine (COS)	Number	Returns the trigonometric cosine for …	COS(numeric_column)	COS(numeric_column)
Exponent (EXP)	Number	Returns e raised to the specified po…	EXP(numeric_column)	EXP(numeric_column)
Floor (FLOOR)	Number	Returns the smallest integer or whol…	FLOOR(numeric_column)	FLOOR(numeric_column)
Natural Logarithm (LOG)	Number	Returns the natural, or base 'e' loga…	LOG(numeric_column)	LOG(numeric_column)
Base 10 Logarithm (L…	Number	Returns the base10 logarithm of the…	LOG10(number)	LOG10(number)
If Null Value (ISNULL)	Number	Returns either the passed column or…	ISNULL(numeric_column,def…	ISNULL(numeric_column,default…
Raise to Exponent (P…	Number	Returns the value raise to the expo…	POWER(numeric_column,ex…	POWER(numeric_column,expon…
Round (ROUND)	Number	Returns the value rounded to a give…	ROUND(numeric_column,pre…	ROUND(numeric_column,precision)
Sign (SIGN)	Number	Returns either 0, 1, or -1 depending…	SIGN(numeric_column)	SIGN(numeric_column)
Sine (SIN)	Number	Returns the trigonometric sine for a…	SIN(numeric_column)	SIN(numeric_column)
Square Root (SQRT)	Number	Returns the square root of the valu…	SQRT(numeric_column)	SQRT(numeric_column)
Tangent (TAN)	Number	Returns the trigonometric tangent f…	TAN(numeric_column)	TAN(numeric_column)
Truncate (ROUND)	Number	Returns the value truncated to a giv…	ROUND(numeric_column,pre…	ROUND(numeric_column,precisio…
ASCII value (ASCII)	String	Returns the ascii value of the first c…	ASCII(string_column)	ASCII(string_column)
Character (CHAR)	String	Returns a character whose ASCII va…	CHAR(integer)	CHAR(integer)
If Null Value (ISNULL)	String	Returns either the passed column or…	ISNULL(string_column,defau…	ISNULL(string_column,default_v…
Length (LEN)	String	Returns an integer value containing …	LEN(string_column)	LEN(string_column)
Locate (CHARINDEX)	String	Returns an integer value containing …	CHARINDEX('search_string',…	CHARINDEX('search_string',strin…
Locate (PATINDEX)	String	Returns an integer value containing …	PATINDEX('search_string',st…	PATINDEX('search_string',string…
Lower (LOWER)	String	Returns a string where every charac…	LOWER(string_column)	LOWER(string_column)
Left (LEFT)	String	Returns a string which is the leftmos…	LEFT(string_column,size)	LEFT(string_column,size)
Left Trim (LTRIM)	String	Returns a string which is the passed …	LTRIM(string_column)	LTRIM(string_column)
Replace (REPLACE)	String	Returns a string where all occurrenc…	REPLACE(string_column,if,t…	REPLACE(string_column,if,then)

Note

The **Delete** button is not available for standard function sets.

Click **Yes** to confirm.

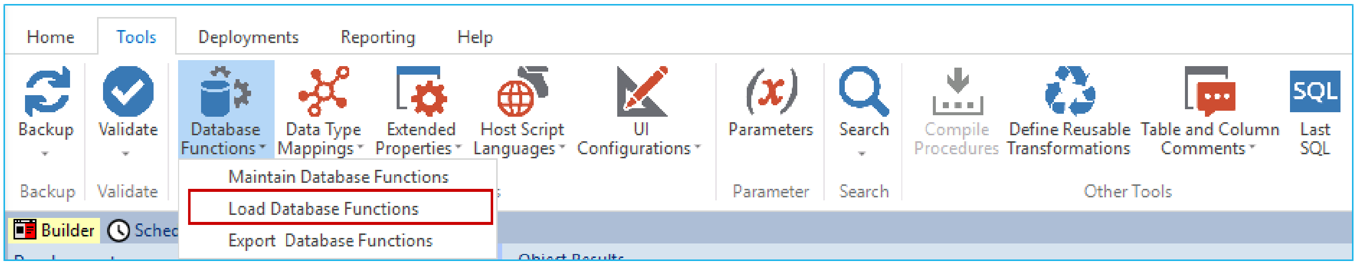


Are you sure you want to delete the function set SQL Server Custom Functions

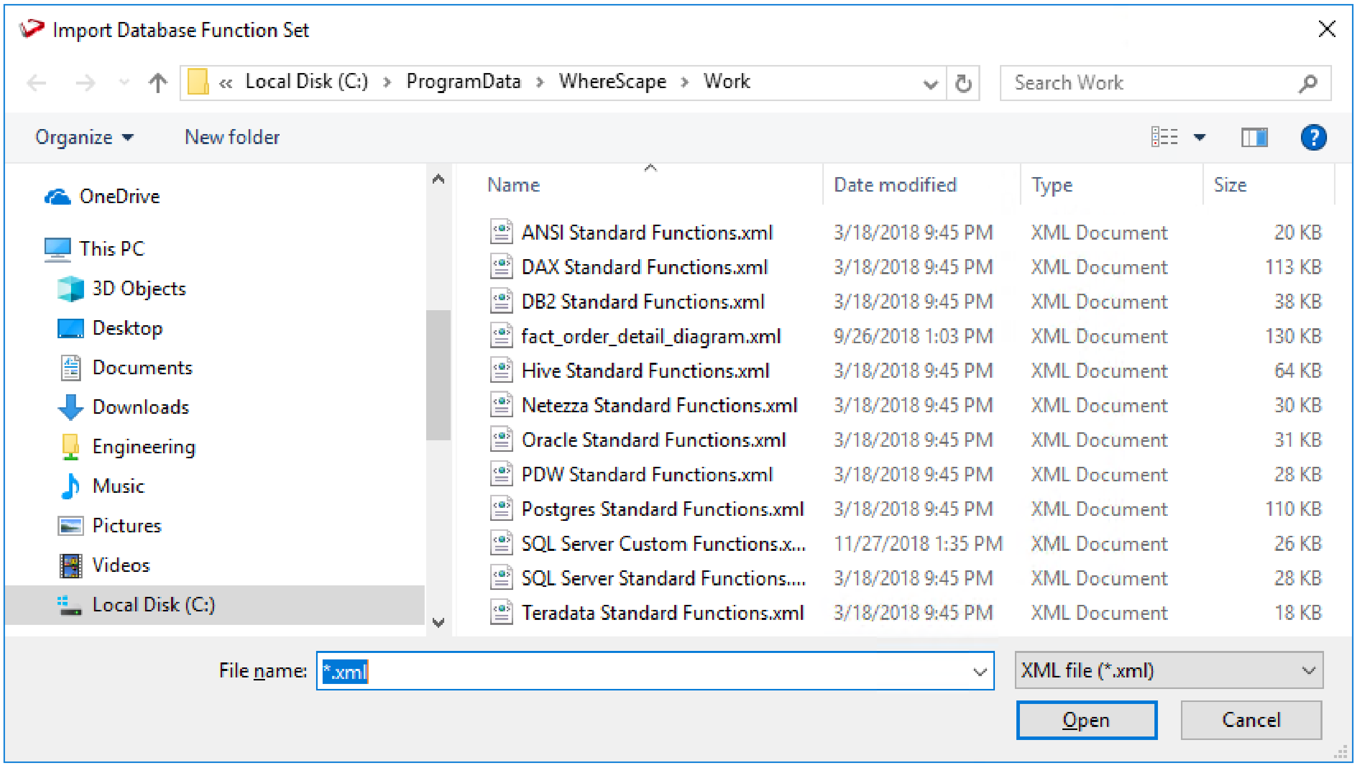
When all functions are deleted, the function set ceases to exist.

Loading Database Function Sets

To load a database function set, select **Tools > Database Functions > Load Database Functions**.



The following screen is displayed. Select the xml file to load the database functions. By default, RED expects the xml files to be in **ProgramData\WhereScape\Work** directory.



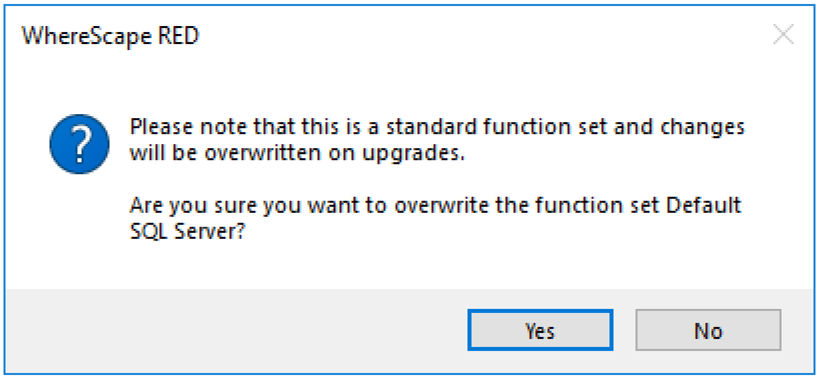
Name	Date modified	Type	Size
ANSI Standard Functions.xml	3/18/2018 9:45 PM	XML Document	20 KB
DAX Standard Functions.xml	3/18/2018 9:45 PM	XML Document	113 KB
DB2 Standard Functions.xml	3/18/2018 9:45 PM	XML Document	38 KB
fact_order_detail_diagram.xml	9/26/2018 1:03 PM	XML Document	130 KB
Hive Standard Functions.xml	3/18/2018 9:45 PM	XML Document	64 KB
Netezza Standard Functions.xml	3/18/2018 9:45 PM	XML Document	30 KB
Oracle Standard Functions.xml	3/18/2018 9:45 PM	XML Document	31 KB
PDW Standard Functions.xml	3/18/2018 9:45 PM	XML Document	28 KB
Postgres Standard Functions.xml	3/18/2018 9:45 PM	XML Document	110 KB
SQL Server Custom Functions.x...	11/27/2018 1:35 PM	XML Document	26 KB
SQL Server Standard Functions....	3/18/2018 9:45 PM	XML Document	28 KB
Teradata Standard Functions.xml	3/18/2018 9:45 PM	XML Document	18 KB

File name: *.xml

XML file (*.xml)

The xml file is validated using the schema definition file at <install directory>\Administrator\Function Sets\Database Function Set.xsd

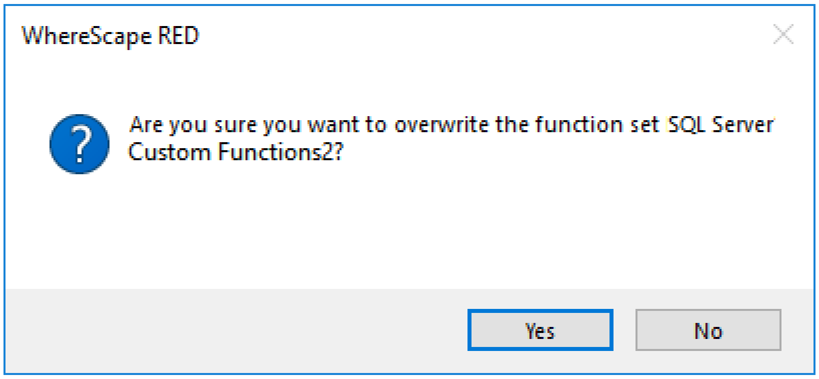
If a function set containing the phrase **Standard Function** is loaded, a warning is displayed:


WhereScape RED

Please note that this is a standard function set and changes will be overwritten on upgrades.

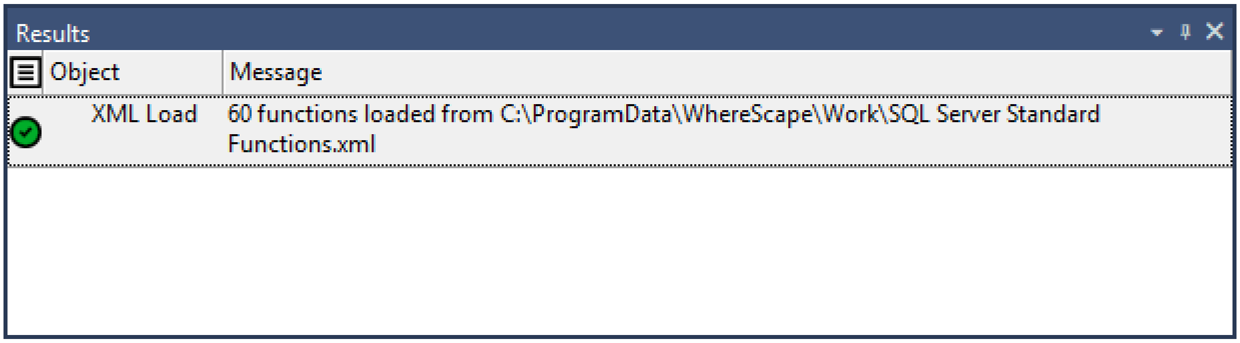
Are you sure you want to overwrite the function set Default SQL Server?

If an existing function set (not a standard set) is about to be overwritten, a warning is displayed:



Are you sure you want to overwrite the function set SQL Server Custom Functions2?

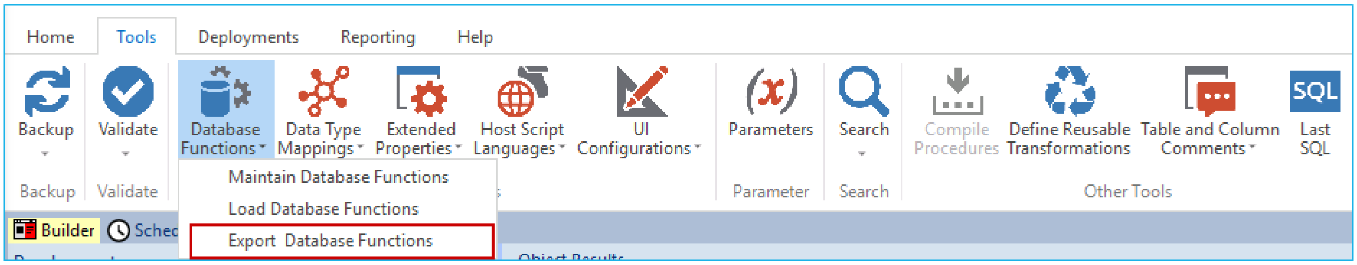
A message is displayed in the **Results** pane.



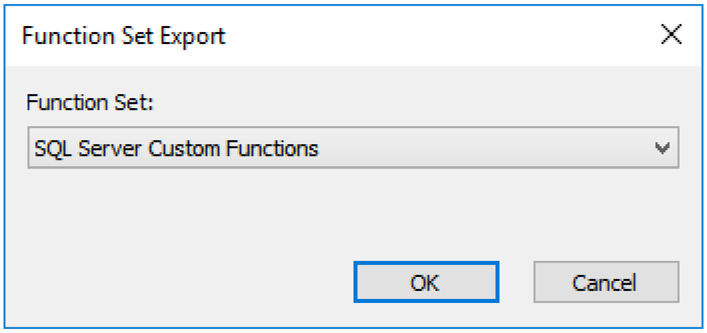
Object	Message
XML Load	60 functions loaded from C:\ProgramData\WhereScape\Work\SQL Server Standard Functions.xml

Exporting Database Function Sets

To export Database Function Sets, select **Tools > Database Functions > Export Database Functions**.

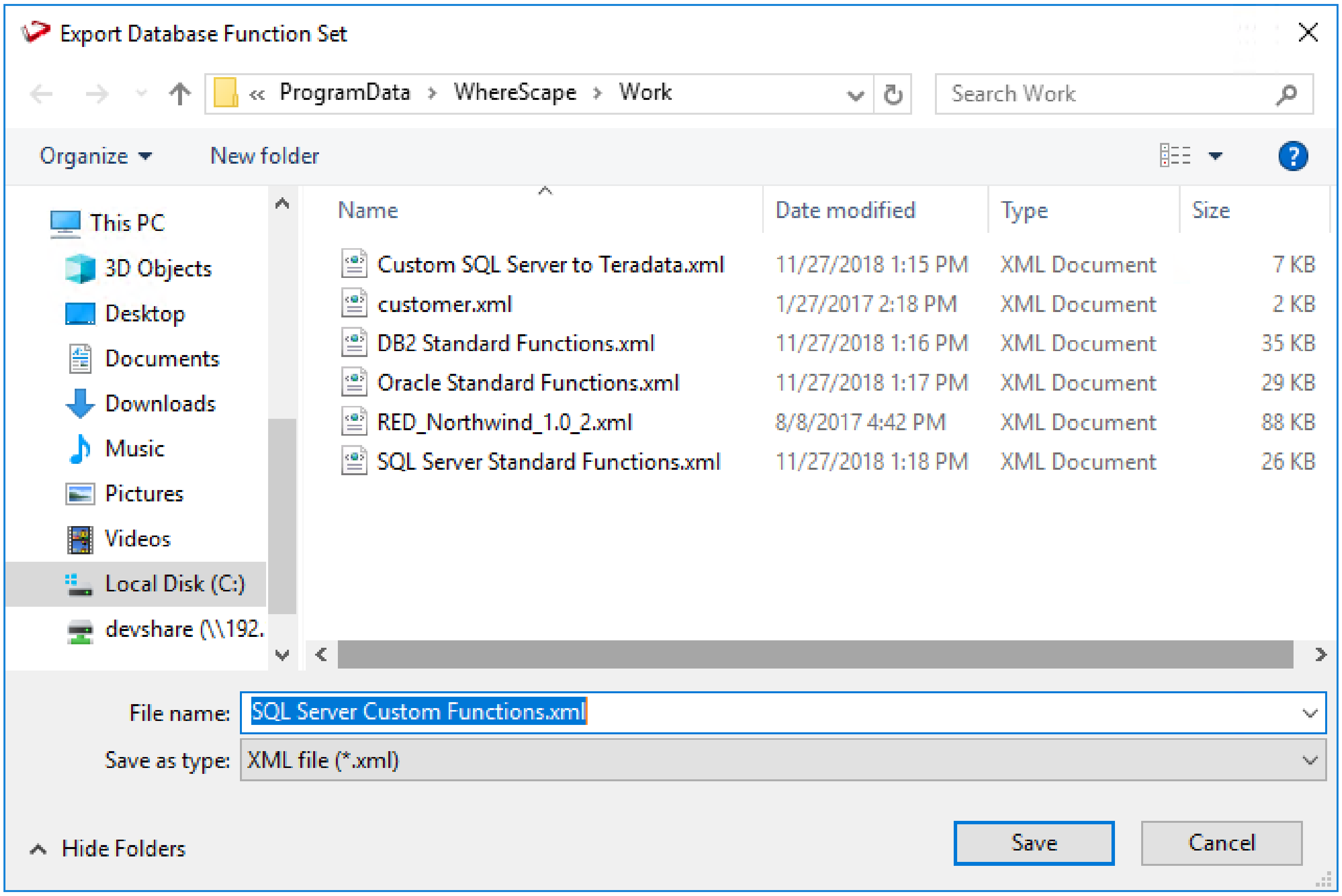


Select the **Function Set** to export from the drop-down list and then click **OK**.



Function Set: SQL Server Custom Functions

By default, RED exports the xml file to the **ProgramData\WhereScape\Work** directory but this can be changed. Change the File name if necessary and then click **Save**.

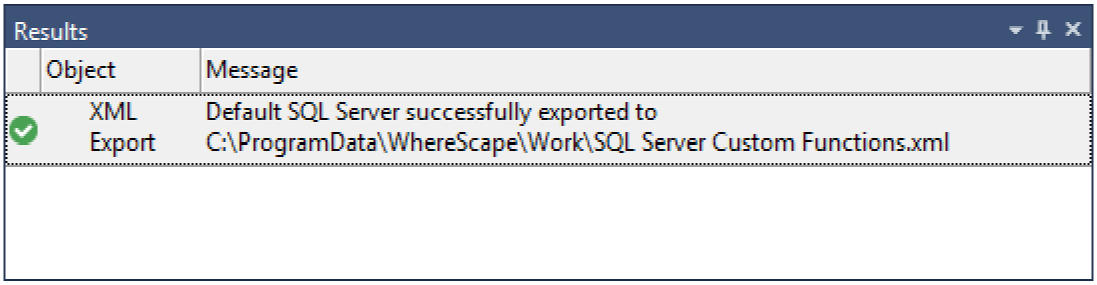


Name	Date modified	Type	Size
Custom SQL Server to Teradata.xml	11/27/2018 1:15 PM	XML Document	7 KB
customer.xml	1/27/2017 2:18 PM	XML Document	2 KB
DB2 Standard Functions.xml	11/27/2018 1:16 PM	XML Document	35 KB
Oracle Standard Functions.xml	11/27/2018 1:17 PM	XML Document	29 KB
RED_Northwind_1.0_2.xml	8/8/2017 4:42 PM	XML Document	88 KB
SQL Server Standard Functions.xml	11/27/2018 1:18 PM	XML Document	26 KB

File name: SQL Server Custom Functions.xml

Save as type: XML file (*.xml)

A message is displayed in the **Results** pane.



Object	Message
✓ XML Export	Default SQL Server successfully exported to C:\ProgramData\WhereScape\Work\SQL Server Custom Functions.xml

Extended Properties

WhereScape RED provides a facility for defining and managing extended database properties which can be used in a range of scenarios, one of the motives for their introduction was to aid interaction with the variety of new database technologies that RED customers want to exploit for their Business Intelligence infrastructure

To support these emerging database technologies, WhereScape provides enablement packs for a variety of target “custom” database types, which includes Snowflake, Amazon Redshift and Microsoft Azure SQL data warehouse.

One component of these packs are a set of extended properties to facilitate the connection to a specific database technology.

This feature also supports importing a set of extended property definitions into RED from an external source, and exporting extended properties defined in RED to a file. This enables you to migrate and reuse extended property definitions between different repositories.

Using the extended properties feature involves defining the extended properties to specify the variables and their scope; and then assigning values to the defined extended properties.

Extended Property Definitions

Extended property definitions primarily consist of a unique name and a scope (database type and object type), which specifies where the extended property will be available.

The extended property definitions are created and maintained in the **Extended Properties Maintenance** screen.

Extended Property Value Assignment

Once an extended property has been defined for a connection or table object, users can then assign values to each of the variable attributes available in the extended property.

The extended property values are specified and maintained in the **Extended Properties** settings in the Connection or Table object's **Properties** screen.

Extended Property Lookup

The extended property variables and their assigned values are referenced in PowerShell scripts. PowerShell script based processing is available for all objects in a custom database target. However, in a non-custom database target; PowerShell scripts are only available to process Load and Export objects.

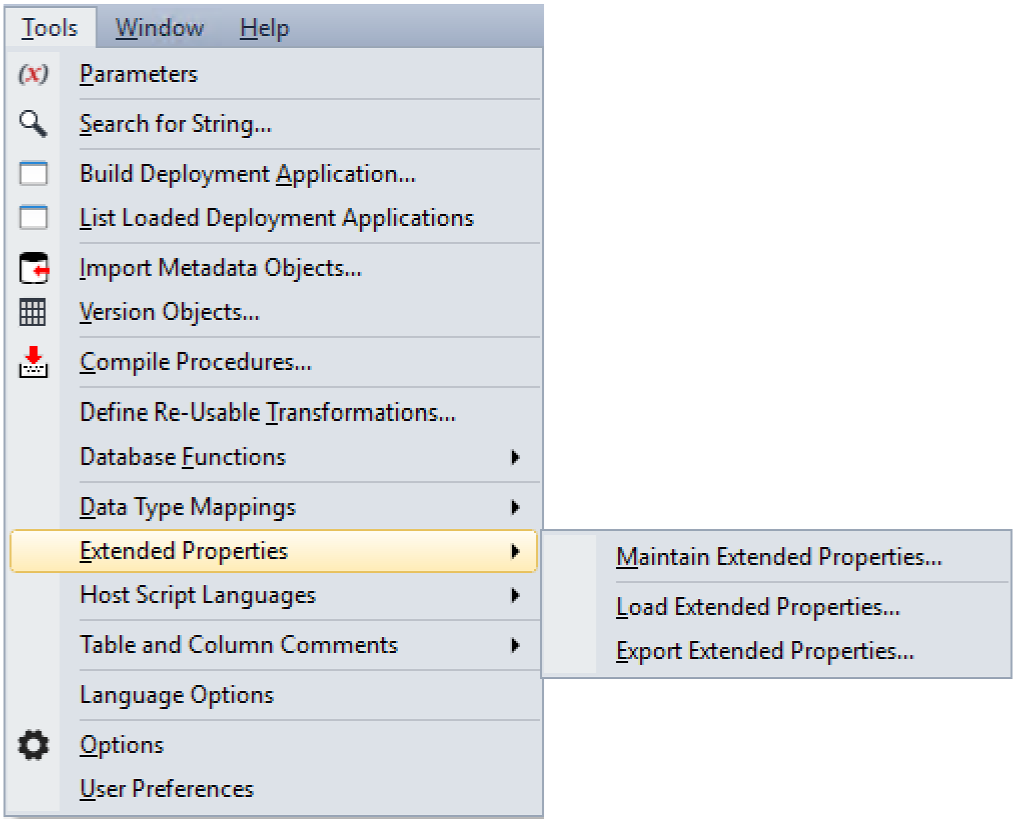
WhereScape provides a specialized enablement pack for custom database customers for their database platform, which includes a utility template with extended property lookup functions. For more information, please contact your WhereScape representative.

Defining Extended Properties

An extended property definition mainly consists of the following attributes:

- **Variable** - the name of the extended property in the format it is accessed from RED and included into scripts.
- **Scope** - the scope is a sub-set definition of an extended property variable which specifies the subset of database types and objects that will have access to the variable.

The user defined extended properties can be created and managed, using the **Extended Properties** facility which is launched from the **Tools** menu.


Tools
Window
Help
Parameters
Search for String...
Build Deployment Application...
List Loaded Deployment Applications
Import Metadata Objects...
Version Objects...
Compile Procedures...
Define Re-Usable Transformations...
Database Functions
Data Type Mappings
Extended Properties
Host Script Languages
Table and Column Comments
Language Options
Options
User Preferences
Maintain Extended Properties...
Load Extended Properties...
Export Extended Properties...

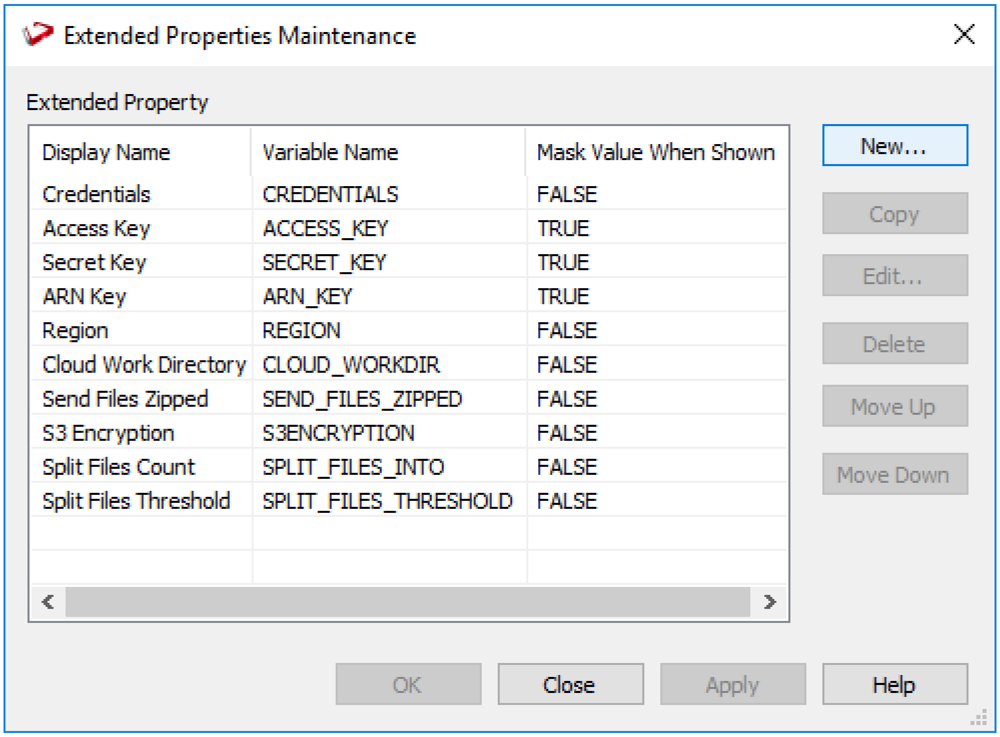
The following options are available:

- **Maintain Extended Properties** is used to add new or edit existing extended property definition settings.
- **Load Extended Properties** is used to load a set of extended property definitions into RED from an external source. This option is typically used with a WhereScape provided enablement pack.
- **Export Extended Properties** is used to export the available extended properties to an external properties file (.extprop).

Creating an Extended Property Definition

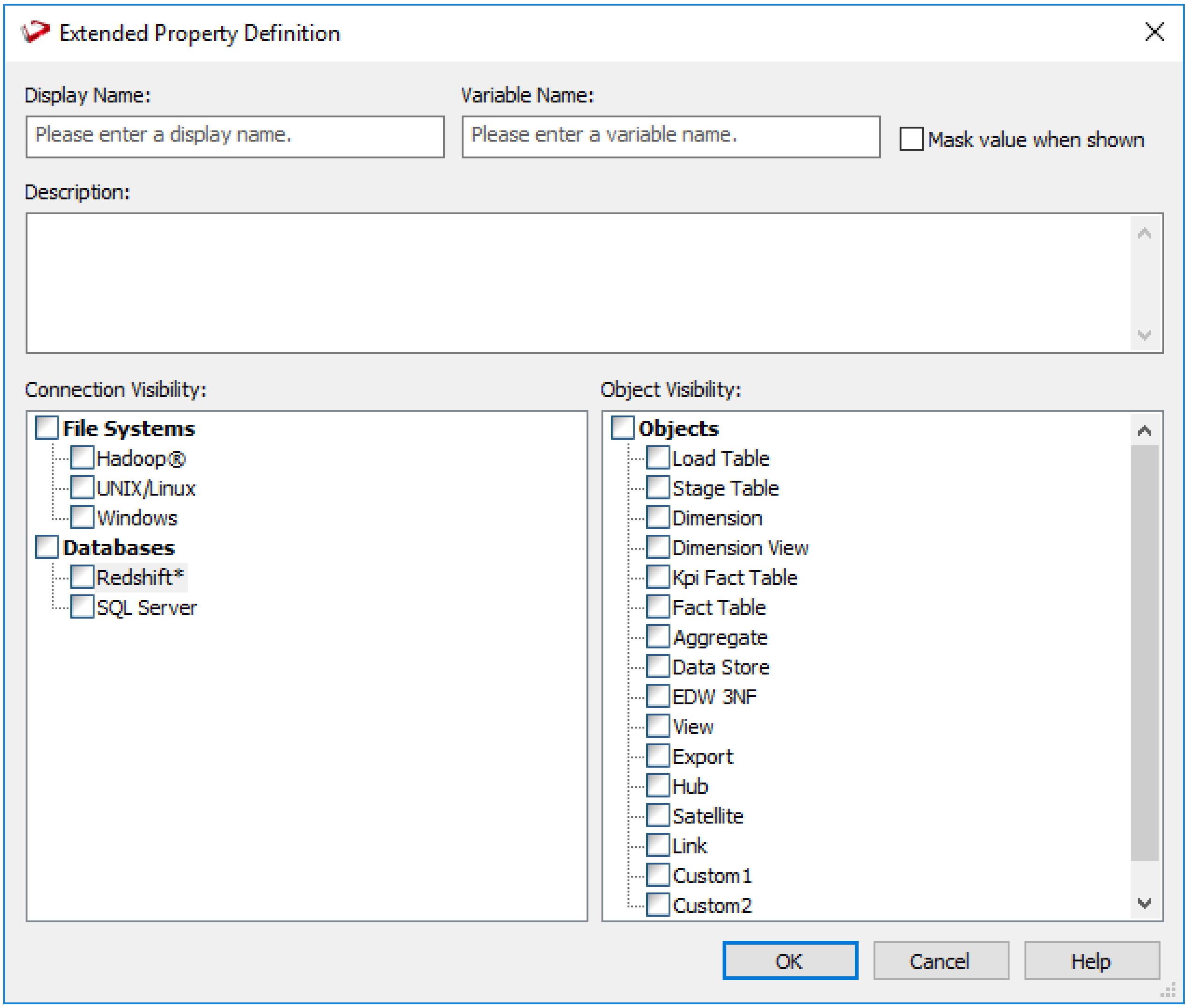
To create a new extended property definition, select **Tools > Extended Properties > Maintain Extended Properties**.

On the **Extended Properties Maintenance** screen, click **New**.



Display Name	Variable Name	Mask Value When Shown
Credentials	CREDENTIALS	FALSE
Access Key	ACCESS_KEY	TRUE
Secret Key	SECRET_KEY	TRUE
ARN Key	ARN_KEY	TRUE
Region	REGION	FALSE
Cloud Work Directory	CLOUD_WORKDIR	FALSE
Send Files Zipped	SEND_FILES_ZIPPED	FALSE
S3 Encryption	S3ENCRYPTION	FALSE
Split Files Count	SPLIT_FILES_INTO	FALSE
Split Files Threshold	SPLIT_FILES_THRESHOLD	FALSE

On the **Extended Property Definition** screen, enter the display and variable names and then configure the settings as described below.



Variable Settings

Menu Options	Description
Display Name	Enter a display name for the variable. This is the extended property name displayed in the Connection or Table properties.
Variable Name	Enter a unique variable name in the format it is accessed from the RED metadata service and included into scripts.
Mask Value when shown	Select this check box if you want to mask the value of the extended property, when displayed in the Connection or Table properties. The extended property value is also masked in the RED generated technical documentation. **Note** During an application creation process (via the **Tools > Build Deployment Application** menu), any extended property variables/assignments applied to a connection or table object are included in the generated application files. However, a connection or table object extended property with a masked value will not be included in the generated application files.
Description	Enter a description for the variable and its purpose/use

Scope Settings

Menu Options	Description
Connection Visibility	Enables you to control the scope of the extended property definition, by selecting a Connection **File System** and/or a **Database** type that can access the extended property variable and their assigned value.For example, if only the SQL Server check box is selected under the **Databases** option, then the variable will only appear in the Extended Properties screen of connections with an SQL Server database type and not in the Extended Properties screen of any other connections that is not an SQL Server database.
Object Visibility	Enables you to control the scope of the extended property definition, by selecting an **Object** type that can access the extended property variable and their assigned value. For example, if only the **Load Table** check box is selected under the **Objects** option, then the variable will only appear in the Extended Properties screen of Load tables and not in the Extended Properties screen of any other table objects.

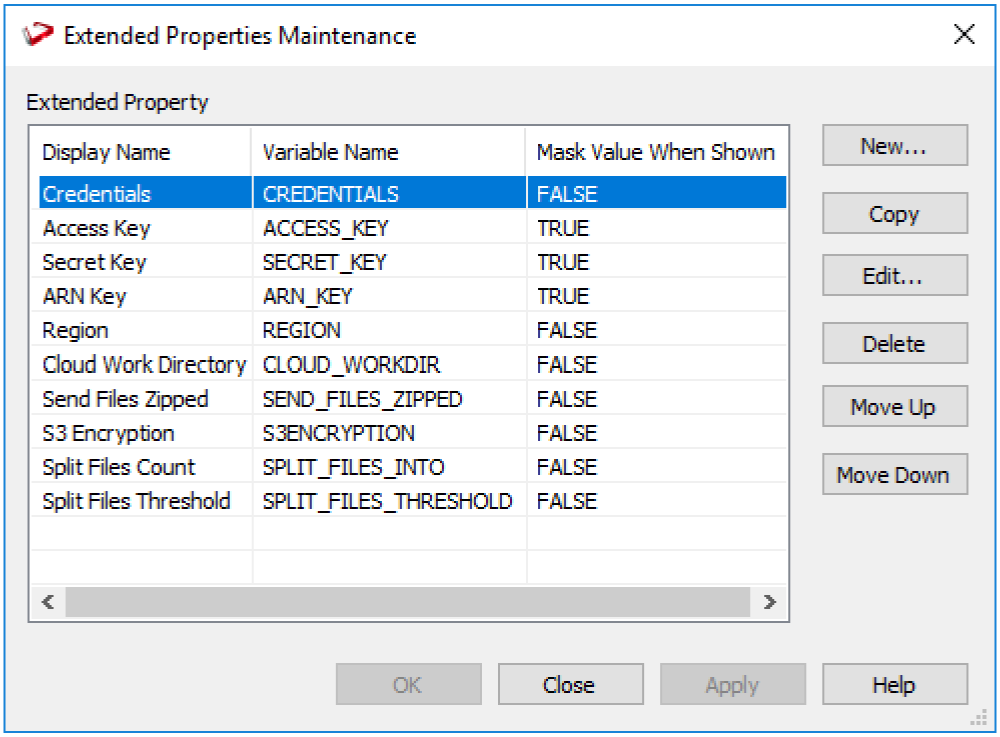
Notes:

- Extended property variables can be set for index objects. An extended property's scope is still defined by object type but the value for the variable can be set for the table and each index independently.
- Automatically created indexes are recreated each time the table's procedure is rebuilt. Therefore, the values set for extended properties on automatically created indexes are lost when they are recreated. Manually created indexes and their extended properties are persisted.
- As with extended properties defined for tables and connections, extended properties defined on indexes can be accessed from templates, including create index DDL, drop index DDL and update procedure/script.

Maintaining Extended Property Definitions

The **Extended Properties Maintenance** screen displays a list of existing extended property definitions and enables you to add a **New** extended property definition. Clicking an extended property definition from the list enables you to **Copy**, **Edit** or **Delete** the selected extended property.

In addition, you can change the position of a selected extended property definition by moving it **Up** or **Down** within the list to specify a preferred order or grouping arrangement, when the extended property definitions are displayed in the Connection or Table properties screens. The display order has no impact upon the operation.



Display Name	Variable Name	Mask Value When Shown
Credentials	CREDENTIALS	FALSE
Access Key	ACCESS_KEY	TRUE
Secret Key	SECRET_KEY	TRUE
ARN Key	ARN_KEY	TRUE
Region	REGION	FALSE
Cloud Work Directory	CLOUD_WORKDIR	FALSE
Send Files Zipped	SEND_FILES_ZIPPED	FALSE
S3 Encryption	S3ENCRYPTION	FALSE
Split Files Count	SPLIT_FILES_INTO	FALSE
Split Files Threshold	SPLIT_FILES_THRESHOLD	FALSE

The following describe the elements of the **Extended Properties Maintenance** screen.

Extended Property

This pane lists the extended property definitions that have been setup in RED. It details the display and variable name and indicates if the value of the extended property is masked, when displayed in the Connection or Table properties.

Function Buttons

These buttons enable you to create and manage the extended property definitions.

Button	Description
New	Launches the **Extended Property Definition** screen which enables you to create a new variable and specify the subset of database types and objects that will have access to the variable. Refer to **Creating an Extended Property Definition** for details.
Copy	Duplicates the selected extended property and adds a _**Copy1** suffix to the **Display Name** and **Property Name**. All fields are populated with the values of the original extended property. These can be edited as required.
Edit	Launches the **Extended Property Definition** screen which enables you to edit the selected extended property.
Delete	Deletes the selected extended property, if it is not currently in use. The user is warned if it is in use and asked to confirm or cancel the delete action.
Move Up/Move Down	Moves the selected extended property Up or Down the list. The display order of the extended property has no impact on its use.

Notes

- If an extended property is deleted, both the definition and all corresponding value assignments are deleted from the meta data repository. It is removed from the **Extended Properties** settings of the Connection or Table object Properties screen.
- If an extended property definition variable name is changed, the change is applied in the **Extended Properties** settings of the Connection or Table object Properties screen and its corresponding value

Notes

assignment is retained.

- If an extended property definition scope is changed, its variable and corresponding value assignments are not deleted from the meta data repository. It is hidden from the **Extended Properties** settings of the Connection or Table object Properties screen. For example, an extended property variable has the Load object type set as in-scope and a corresponding value assignment has also been set in the Extended Properties setting in the Table Properties screen.
  If the extended property definition is edited to remove the Load object type from its scope, then the value assignment is not removed but becomes inactive and RED displays a warning that the scope change impacts a value assignment.

Extended Properties Value Assignment

After defining the extended property, you can assign values to each of the variable available in the **Extended Properties** tab of in-scope objects.

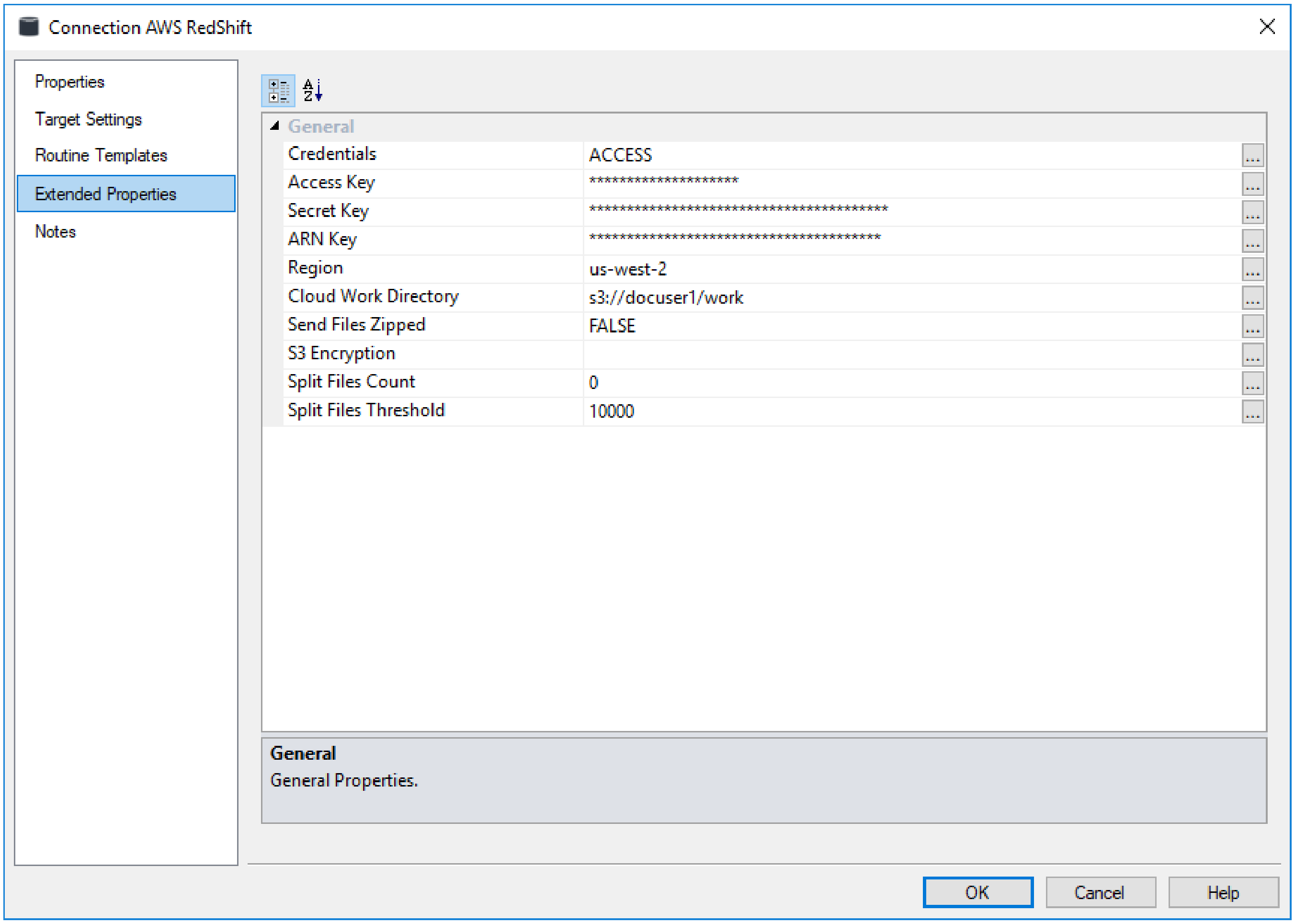
One extended property can have several connections and objects in its scope and therefore can have several values and assignments—one per connection and object.

An extended property assignment establishes the relationship between a variable and its value, which is specific for a particular connection or object.

Setting Up Extended Property Values for a Connection

The extended property values for a connection are assigned and maintained in the **Extended Properties** tab of the connection **Properties** screen.

Connection Extended Properties



General

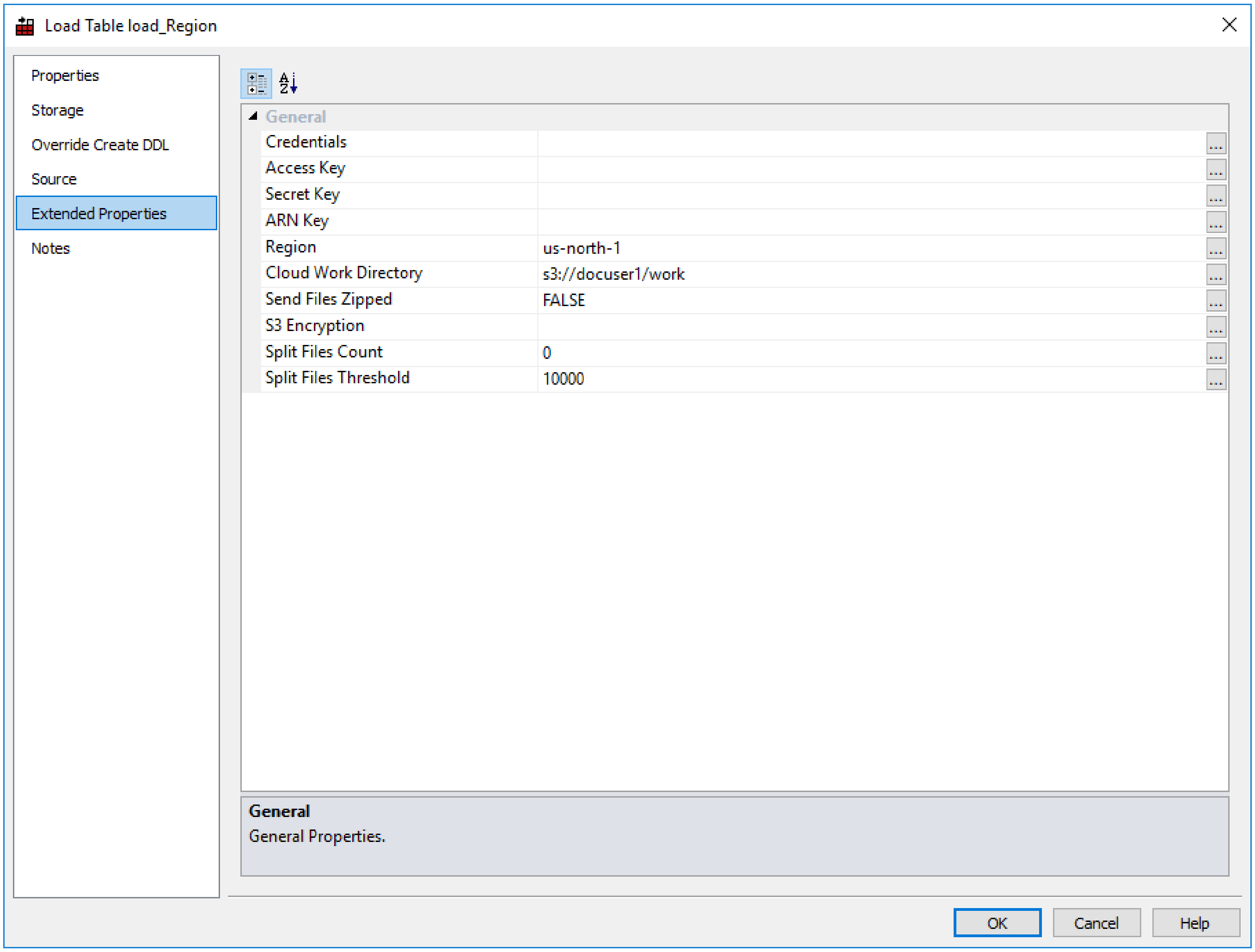
Credentials	ACCESS
Access Key	********************
Secret Key	**
ARN Key	***
Region	us-west-2
Cloud Work Directory	s3://docuser1/work
Send Files Zipped	FALSE
S3 Encryption	
Split Files Count	0
Split Files Threshold	10000

General
General Properties.

Setting Up Extended Property Values for an Object

The extended property values for an object are assigned and maintained in the **Extended Properties** tab of the table **Properties** screen.

WhereScape RED Object Extended Properties



General	
Credentials	
Access Key	
Secret Key	
ARN Key	
Region	us-north-1
Cloud Work Directory	s3://docuser1/work
Send Files Zipped	FALSE
S3 Encryption	
Split Files Count	0
Split Files Threshold	10000

General
General Properties.

Extended Properties Lookup

Extended properties can be accessed in the following four ways:

1. All template driven code generation

Template engine reference examples using pebble syntax:

table.extendedPropertyValuesByName[“<propertyName>”]

table.loadInfo.sourceConnection.extendedPropertyValuesByName[“<propertyName>”]

table.target.connection.extendedPropertyValuesByName[“<propertyName>”]

2. Automatic token replacement at run-time in Windows scripts

The automatic token replacement applies to Windows based scripts only. Each token is assessed at run-time immediately before executing a script in the same way parameters are replaced:

Token format: $WSL_EXT_<propertyName>$

In the token replacement process, the first non-empty Extended Property value found is replaced in this order of preference:

1. Table or Export Object
2. Associated Source Connection
3. Associated Target Connection

If the Extended Property was found without any value set then an empty string is returned.

If the Extended Property was not found in the associated objects the token is left unchanged.

3. Calling the WslMetadataServiceClient.dll at run-time from Windows scripts

This is an advanced feature and not recommended for general use as it puts a reliance on loading this DLL at run-time from Windows scripts.

Below, an example on the PowerShell code block inside a RED PowerShell Script:

```
Add-Type -Path "${env:WSL_BINDIR}WslMetadataServiceClient.dll"
 $repo = New-Object WslMetadataServiceClient.Repo("${env:WSL_META_DSN}",
'SQLServer',
"${env:WSL_META_USER}", "${env:WSL_META_PWD}", "${env:WSL_META_SCHEMA}",
"_${env:WSL_META_DSN_ARCH}")
$repo.objectsByName[“<objectName>”].extendedPropertyValuesByName["<propertyName>"]
```

4. WhereScape Database Enablement Packs

WhereScape provides Database Enablement Packs for many target platforms, each pack includes a utility template with extended property lookup functions as well as a PowerShell and Python functions to call at run-time. For more information please contact your WhereScaperepresentative.

Extended Properties Data Migration Between Repositories

Once extended properties are defined and set, they can be propagated to the other RED repositories as follows:

The definitions are exported from the source repository and imported to the target repository via the **Tools > Extended Properties** menu.

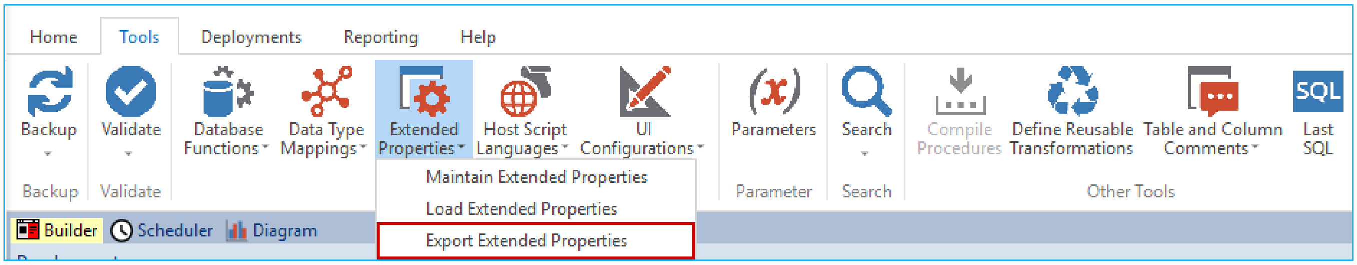
The values assigned to extended properties are included in a RED application that includes the parent connection or table object and are imported to the target repository during the application load process.

Notes:

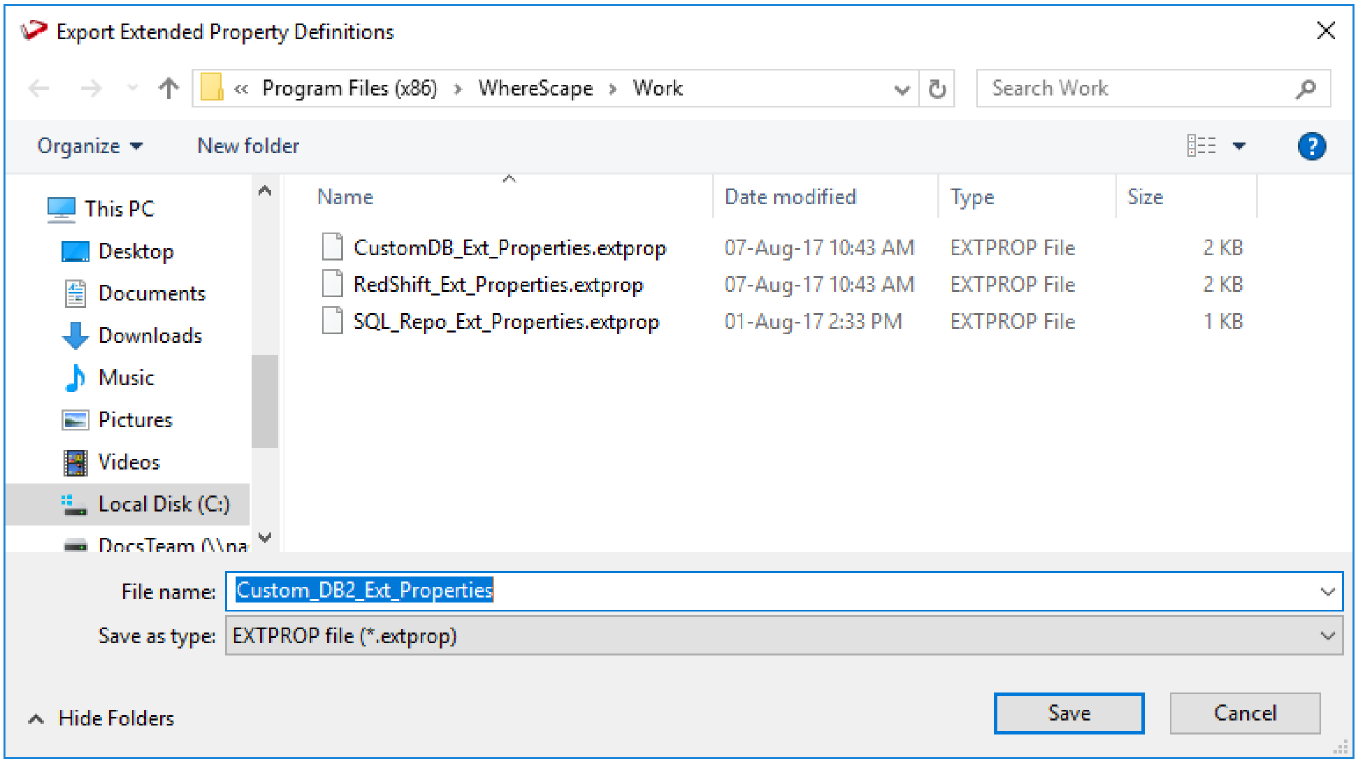
1. Any new or changed extended property definition must first be exported and then imported into the downstream repository, prior to loading an application that references them. Failing to load the definition first will result in the value assignment not being loaded for the object.
2. Extended properties are designed to be accessed from Windows PowerShell scripts, via the WhereScape RED metadata service. To aid this access, WhereScape provides a metadata service client library that customers are suggested to use, to access the metadata service.

Exporting Extended Properties

To export extended properties from RED, select **Tools > Extended Properties > Export Extended Properties**.



The following window is displayed. By default, RED exports the extended property (EXTPROP) file to **Program Files\WhereScape\Work** directory, but this can be changed. Type in the **File name** and then click **Save**.

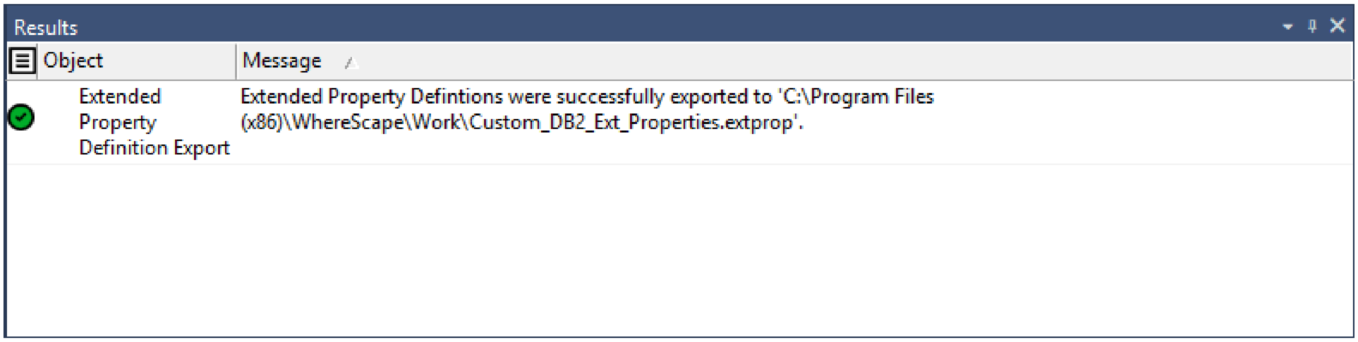


Name	Date modified	Type	Size
CustomDB_Ext_Properties.extprop	07-Aug-17 10:43 AM	EXTPROP File	2 KB
RedShift_Ext_Properties.extprop	07-Aug-17 10:43 AM	EXTPROP File	2 KB
SQL_Repo_Ext_Properties.extprop	01-Aug-17 2:33 PM	EXTPROP File	1 KB

File name: Custom_DB2_Ext_Properties

Save as type: EXTPROP file (*.extprop)

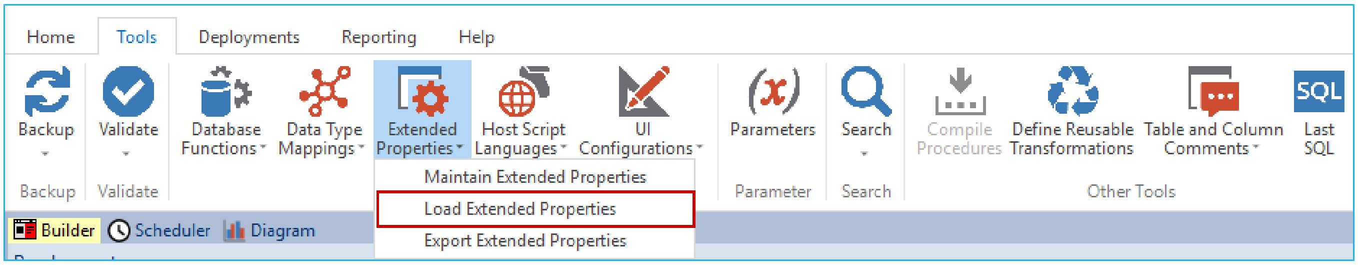
A confirmation message is displayed in the **Results** pane.



Object	Message
Extended Property Definition Export	Extended Property Defintions were successfully exported to 'C:\Program Files (x86)\WhereScape\Work\Custom_DB2_Ext_Properties.extprop'.

Loading Extended Properties

To load an extended property file, select **Tools > Extended Properties > Load Extended Properties**.

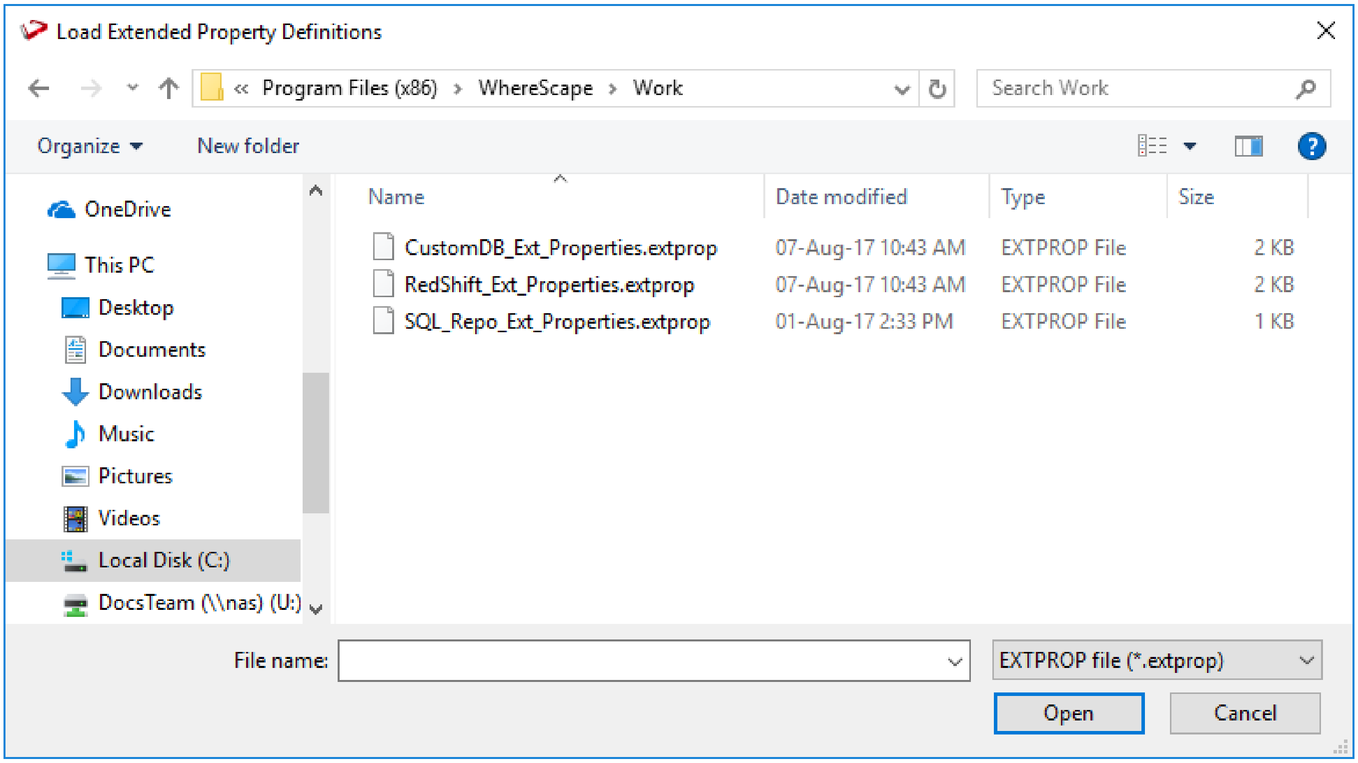


Note

The values assigned to extended properties are included in a RED application that includes the parent Connection or table Object and are imported to the target repository, during the application load process.

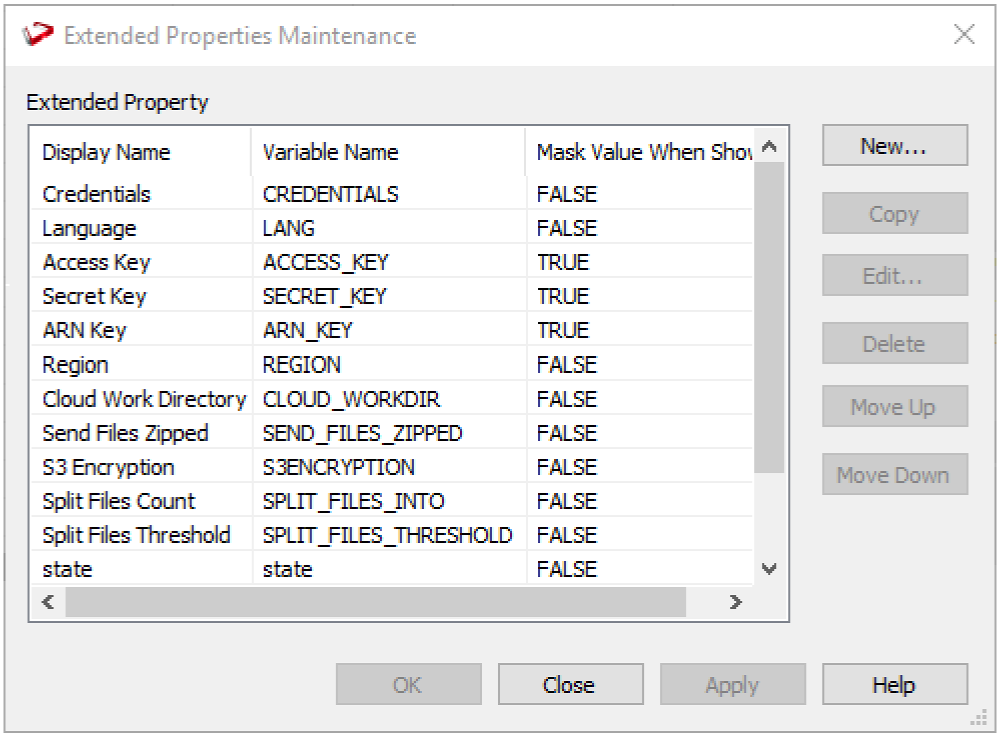
An application including an object with an extended property value set, is dependent on the extended property definition being present in the target repository.

The following window is displayed. Select the extended property (EXTPROP) file to load. By default, RED expects the extprop files to be in **Program Files\WhereScape\Work** directory, but this can be changed.



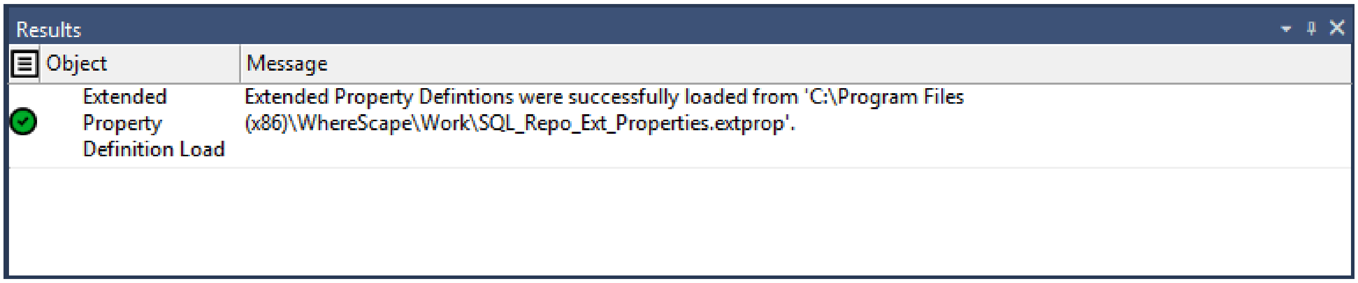
Name	Date modified	Type	Size
CustomDB_Ext_Properties.extprop	07-Aug-17 10:43 AM	EXTPROP File	2 KB
RedShift_Ext_Properties.extprop	07-Aug-17 10:43 AM	EXTPROP File	2 KB
SQL_Repo_Ext_Properties.extprop	01-Aug-17 2:33 PM	EXTPROP File	1 KB

The selected extprop file is loaded and the attributes are listed in the **Extended Properties Maintenance** window.



Display Name	Variable Name	Mask Value When Shov
Credentials	CREDENTIALS	FALSE
Language	LANG	FALSE
Access Key	ACCESS_KEY	TRUE
Secret Key	SECRET_KEY	TRUE
ARN Key	ARN_KEY	TRUE
Region	REGION	FALSE
Cloud Work Directory	CLOUD_WORKDIR	FALSE
Send Files Zipped	SEND_FILES_ZIPPED	FALSE
S3 Encryption	S3ENCRYPTION	FALSE
Split Files Count	SPLIT_FILES_INTO	FALSE
Split Files Threshold	SPLIT_FILES_THRESHOLD	FALSE
state	state	FALSE

A confirmation message is displayed in the **Results** pane.



Object	Message
✓ Extended Property Definition Load	Extended Property Defintions were successfully loaded from 'C:\Program Files (x86)\WhereScape\Work\SQL_Repo_Ext_Properties.extprop'.

Multi Source Processing

WhereScape RED enables you to process data from more than one source query into a single consolidated table. The multi-source processing capability is designed to provide users with flexibility in the definition of component source statements and the ability to independently process multi source tables that are associated with a target table.

Multi Source Functions and Features

The following describes the WhereScape RED functions and features that are used to support multi source processing of data warehouse objects, created in RED.

Note

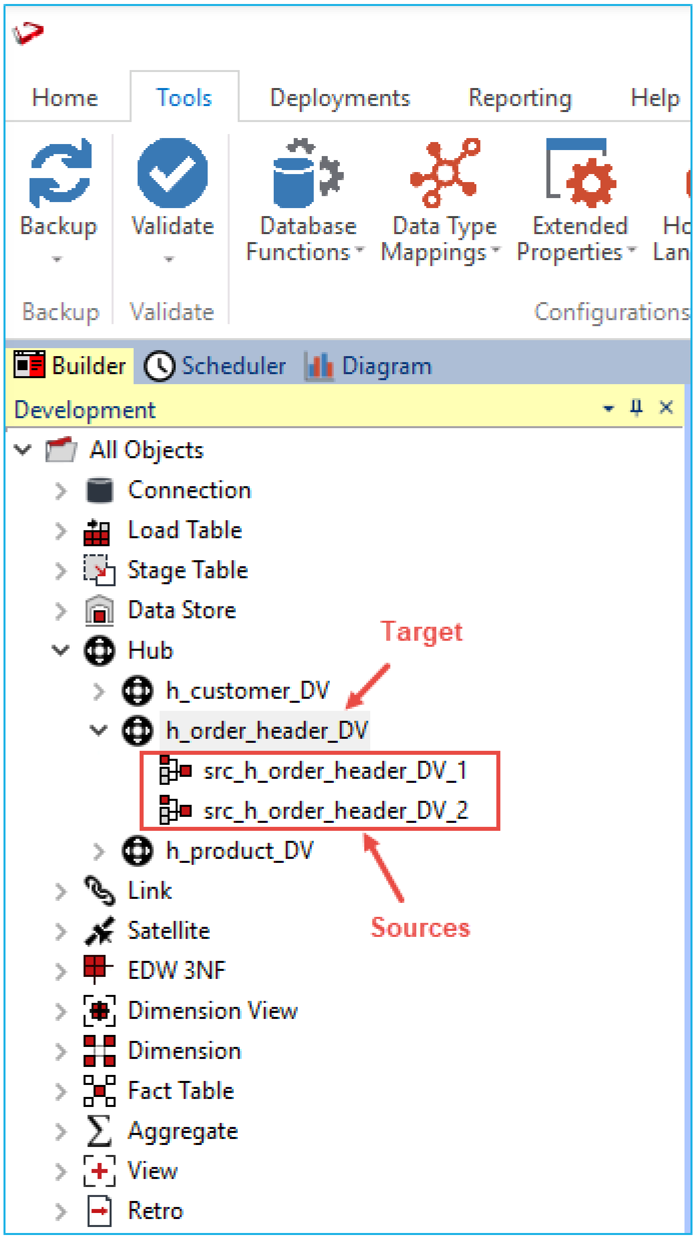
The source mapping feature can be used with all RED table objects, except for Tabular objects, partitioned tables and Load tables.

Source Mapping Object

Source Mapping objects are child objects that are used in WhereScape RED to map columns from one or more source tables to an existing target table in RED.

A Source Mapping object is built from the Data Warehouse connection and can be created from one or more source tables.

Refer to **Adding Source Mapping to Objects** for details.

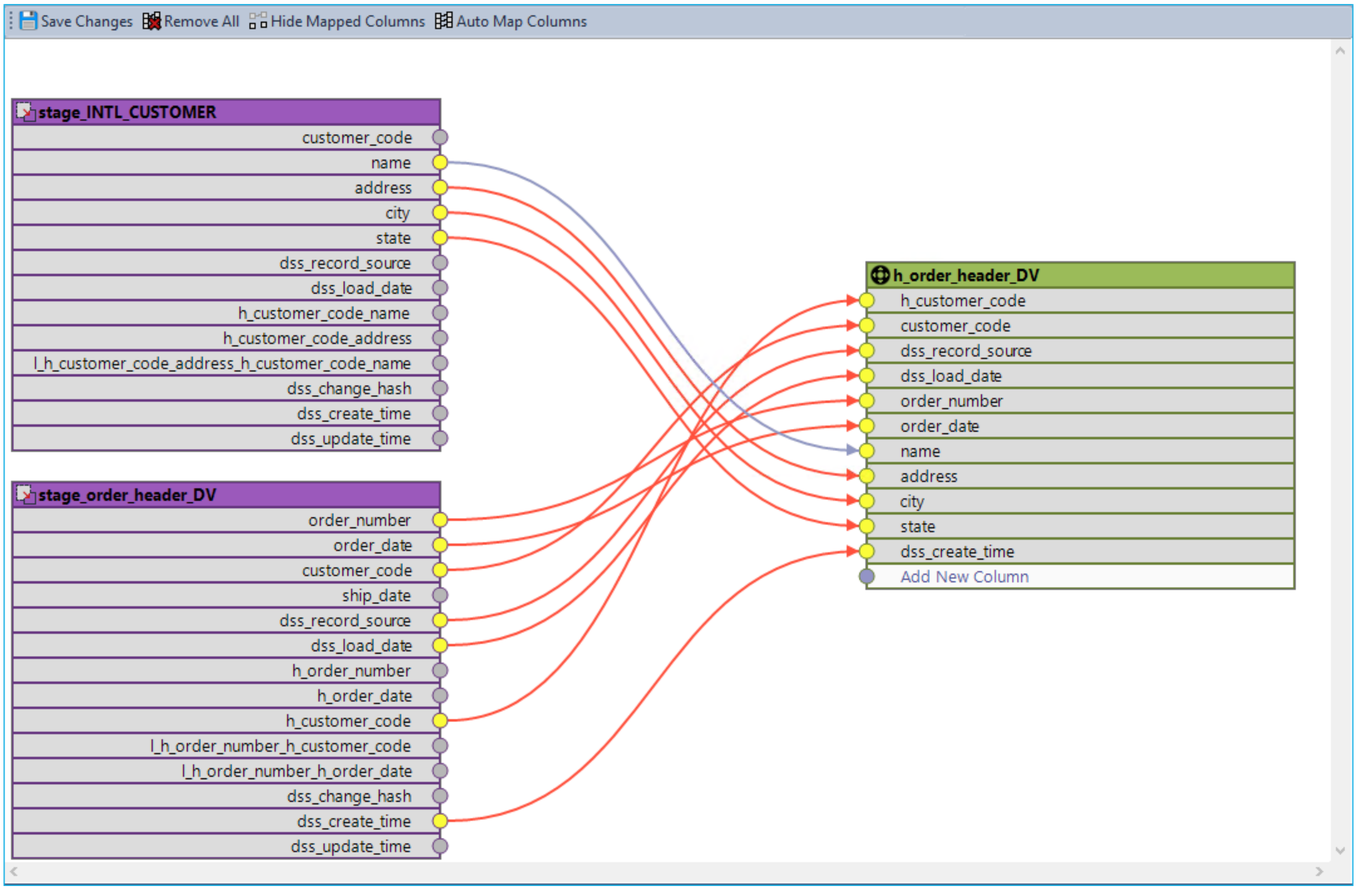


Source Mapping Tool

A Source Mapping tool is available in WhereScape RED which enables you to graphically map columns from one or more source tables to an existing target table in RED.

This tool provides a graphical representation of the mapped columns between the source tables and the target table. The columns from the source table that exists in the target table are automatically mapped. You can edit the mappings by clicking a source column connection point and dragging the line that appears to the target column's connection point.

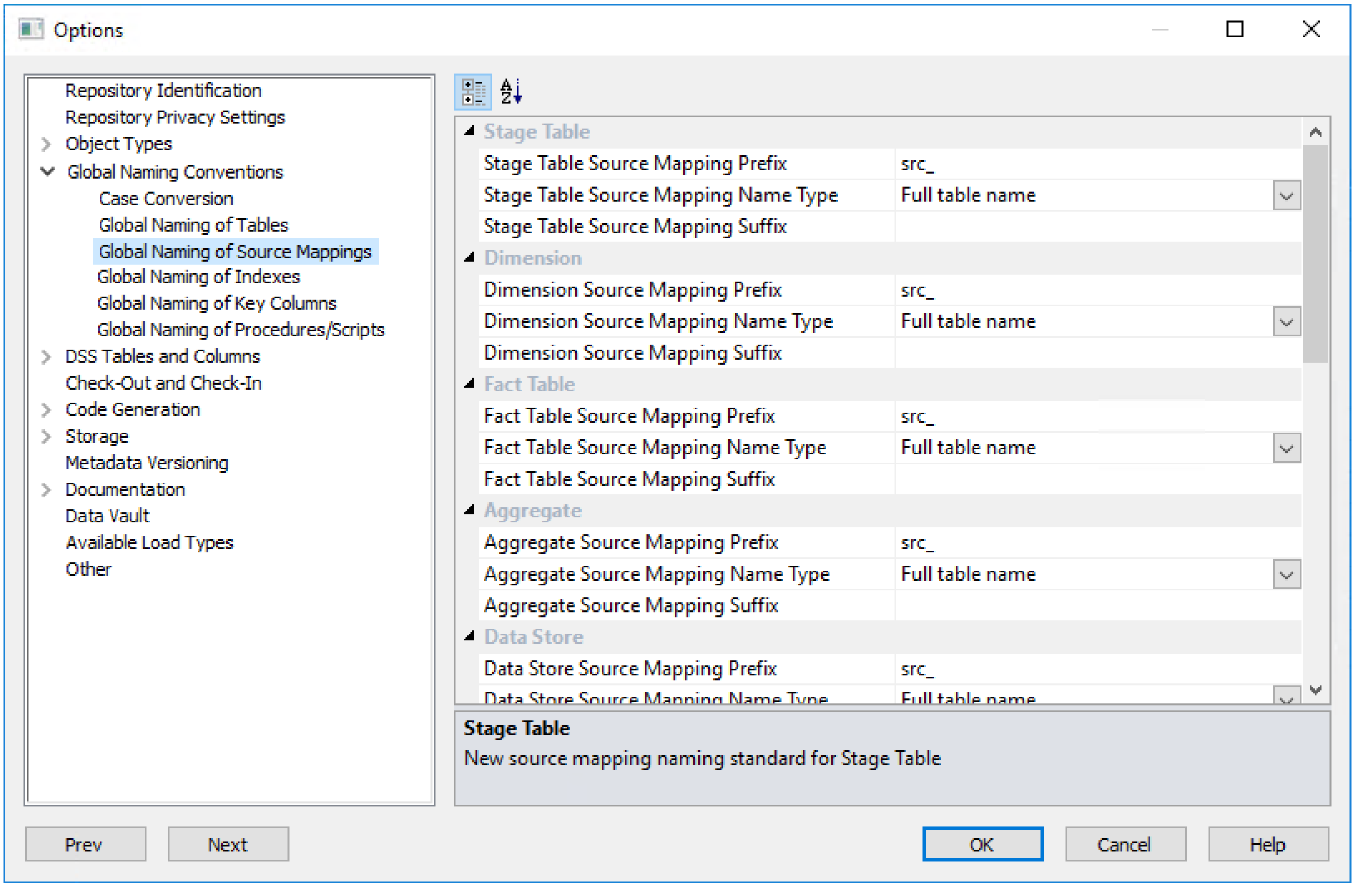
Refer to the section **Maintaining Source Column Mappings** for details.



Global Naming of Source Mappings

The **Global Naming Conventions** option includes the **Global Naming of Source Mappings** setting that enables you to define naming standards for the source mapping objects in RED.

Refer to **Global Naming Conventions > Global Naming of Source Mappings** for details.



Stage Table	
Stage Table Source Mapping Prefix	src_
Stage Table Source Mapping Name Type	Full table name
Stage Table Source Mapping Suffix	
Dimension	
Dimension Source Mapping Prefix	src_
Dimension Source Mapping Name Type	Full table name
Dimension Source Mapping Suffix	
Fact Table	
Fact Table Source Mapping Prefix	src_
Fact Table Source Mapping Name Type	Full table name
Fact Table Source Mapping Suffix	
Aggregate	
Aggregate Source Mapping Prefix	src_
Aggregate Source Mapping Name Type	Full table name
Aggregate Source Mapping Suffix	
Data Store	
Data Store Source Mapping Prefix	src_
Data Store Source Mapping Name Type	Full table name

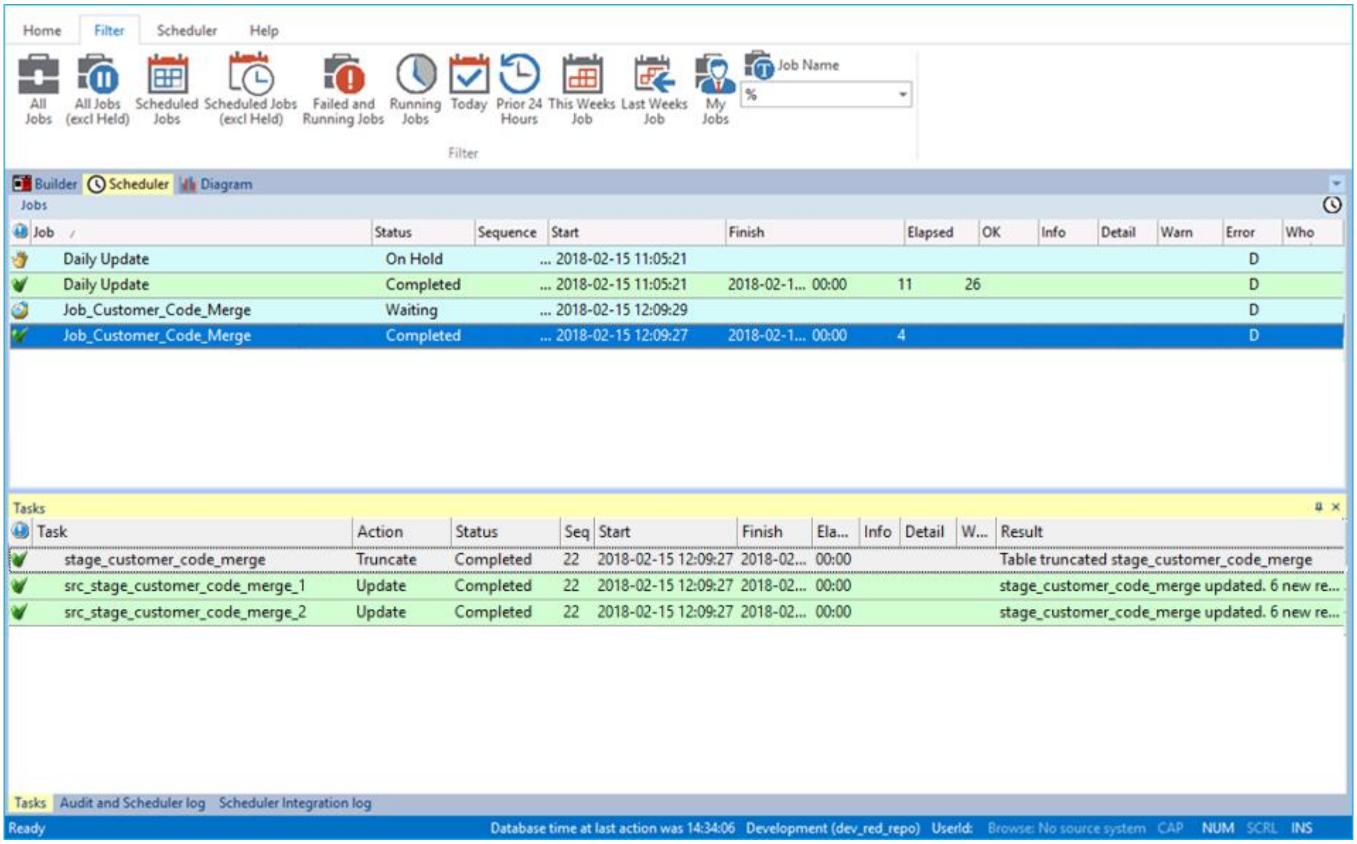
Stage Table
New source mapping naming standard for Stage Table

Independent Execution of Update Procedures

The update procedures associated with the target table and the source mapping objects can be executed individually, separate from each other. Similar to other table objects in RED, the update procedures can be executed on demand or via the **Scheduler**. This provides the flexibility of scheduling the update procedures for each individual source mapping object as required, e.g. based on the update frequency of the tables where the source columns are obtained.

The update procedures for source mapping objects can only be generated using a template.

Refer to **Generating Update Procedures for Source Mapping Objects** for details

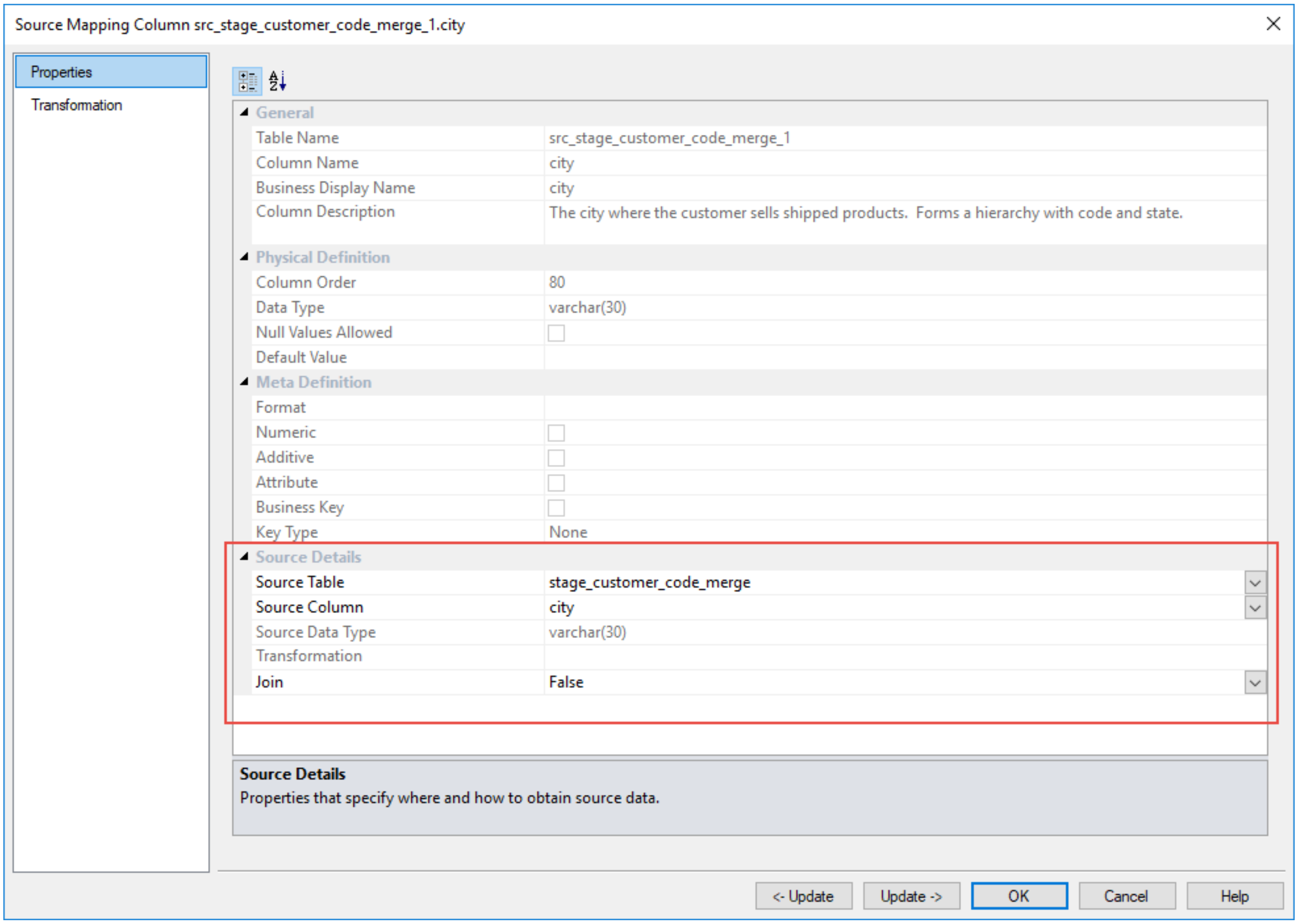


Job	Status	Sequence	Start	Finish	Elapsed	OK	Info	Detail	Warn	Error	Who
Daily Update	On Hold	...	2018-02-15 11:05:21							D	
Daily Update	Completed	...	2018-02-15 11:05:21	2018-02-1... 00:00	11	26				D	
Job_Customer_Code_Merge	Waiting	...	2018-02-15 12:09:29							D	
Job_Customer_Code_Merge	Completed	...	2018-02-15 12:09:27	2018-02-1... 00:00	4					D	

Task	Action	Status	Seq	Start	Finish	Ela...	Info	Detail	W...	Result
stage_customer_code_merge	Truncate	Completed	22	2018-02-15 12:09:27	2018-02...	00:00				Table truncated stage_customer_code_merge
src_stage_customer_code_merge_1	Update	Completed	22	2018-02-15 12:09:27	2018-02...	00:00				stage_customer_code_merge updated. 6 new re...
src_stage_customer_code_merge_2	Update	Completed	22	2018-02-15 12:09:27	2018-02...	00:00				stage_customer_code_merge updated. 6 new re...

Table Column Properties

The **Source Details** table column properties is used to specify where to obtain the source data and provides options for processing (transformation and join settings) the source data. You can also define and manage column source mappings from this screen.



Source Mapping Column src_stage_customer_code_merge_1.city

General	
Table Name	src_stage_customer_code_merge_1
Column Name	city
Business Display Name	city
Column Description	The city where the customer sells shipped products. Forms a hierarchy with code and state.
Physical Definition	
Column Order	80
Data Type	varchar(30)
Null Values Allowed	□
Default Value	
Meta Definition	
Format	
Numeric	□
Additive	□
Attribute	□
Business Key	□
Key Type	None
Source Details	
Source Table	stage_customer_code_merge
Source Column	city
Source Data Type	varchar(30)
Transformation	
Join	False

Source Details
Properties that specify where and how to obtain source data.

Refer to the relevant table Column Properties sections in this user guide for details.

Adding Source Mapping to Objects

When adding source columns to a table object in RED, the target table's current source columns are used to create the first source mapping object and then proceeds to create the second source mapping object which contains the additional source columns that need to be mapped to the target table. The columns from the source table that exists in the target table are automatically mapped.

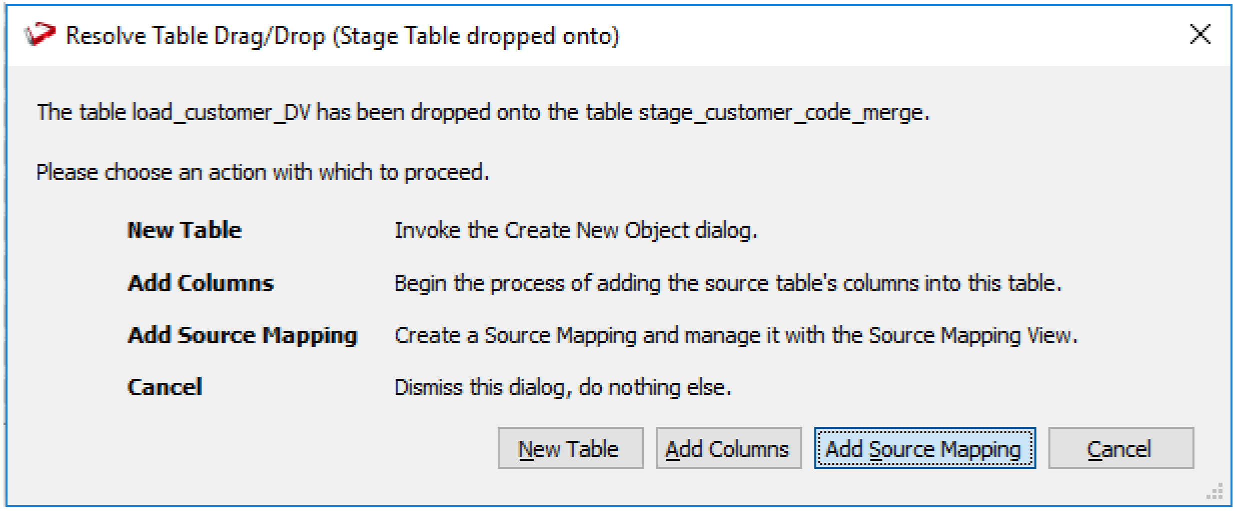
The process of adding Source Mappings is initiated when a source table is dragged and dropped to an existing table object in RED.

The user is prompted with the following options:

- New Table - Opens a new dialog that enables you to create the dropped table as a new object in the metadata repository.
- Add Columns - Inserts the columns of the dropped table into the target table.
- Add Source Mapping - Inserts a new source mapping into the target table.

Note

If you are creating the first source mapping, the target table is first converted to a source mappable object, and then a new source mapping is inserted. If it is not the first source mapping, a new source mapping is just inserted into the target table.



Resolve Table Drag/Drop (Stage Table dropped onto)

The table load_customer_DV has been dropped onto the table stage_customer_code_merge.

Please choose an action with which to proceed.

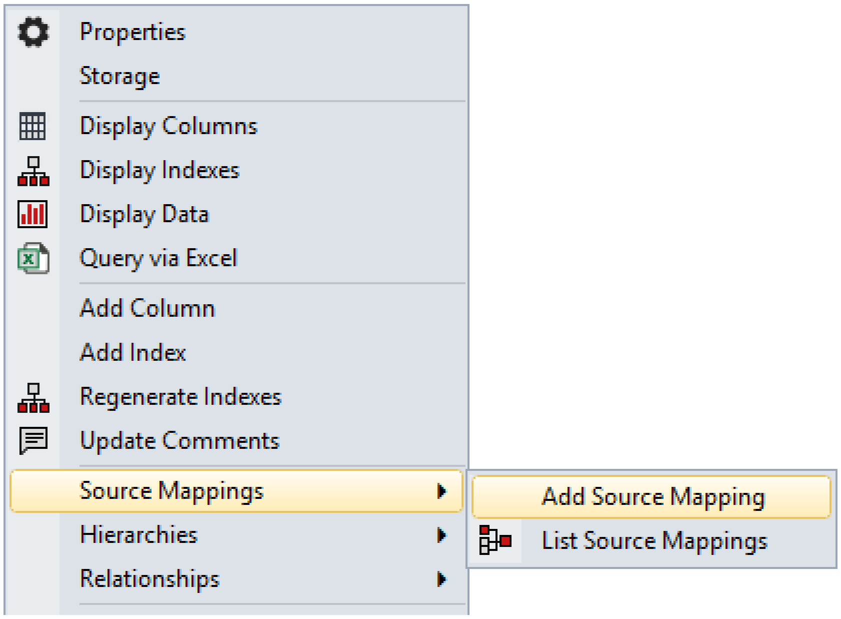
New Table Invoke the Create New Object dialog.

Add Columns Begin the process of adding the source table's columns into this table.

Add Source Mapping Create a Source Mapping and manage it with the Source Mapping View.

Cancel Dismiss this dialog, do nothing else.

The alternative method is to right click the target table from the **Objects** list pane and then select the **Add Source Mapping** option from the context menu:



Drag and Drop

The common approach to create source mappings is to select the source table that contains the columns that you want to add and then drag this table into the target table.

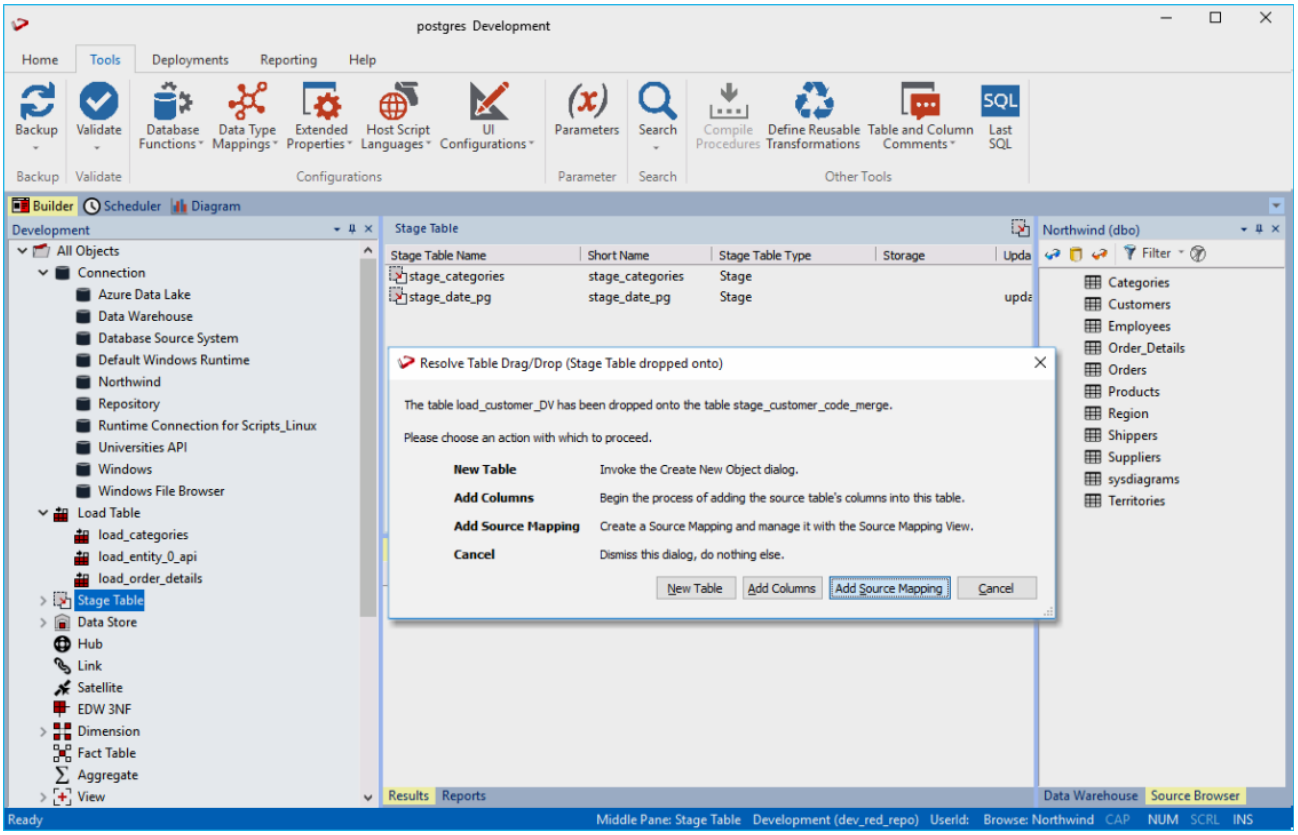
This process creates two source mapping objects, the first one contains the original source mapping of the target table and second one is for the additional columns from the source table selected in the **Browser** pane.

You can drag additional source tables to create additional source mappings objects until you have all the source data required for your target table.

1. Click the target table from the **Objects** list in the left pane.

   The middle pane displays the properties of the selected target table, the pane is identified as a target for new Source Mapping objects.
2. Browse to the Data Warehouse via the **Browse > Source Table** menu option.
3. Click the source table that contains the columns you want to include as sources from the **Browser** pane, and drag it to the middle pane.

4. Click the **Add Source Mapping** button in the **Resolve Table Drag/Drop** prompt.



Resolve Table Drag/Drop (Stage Table dropped onto)

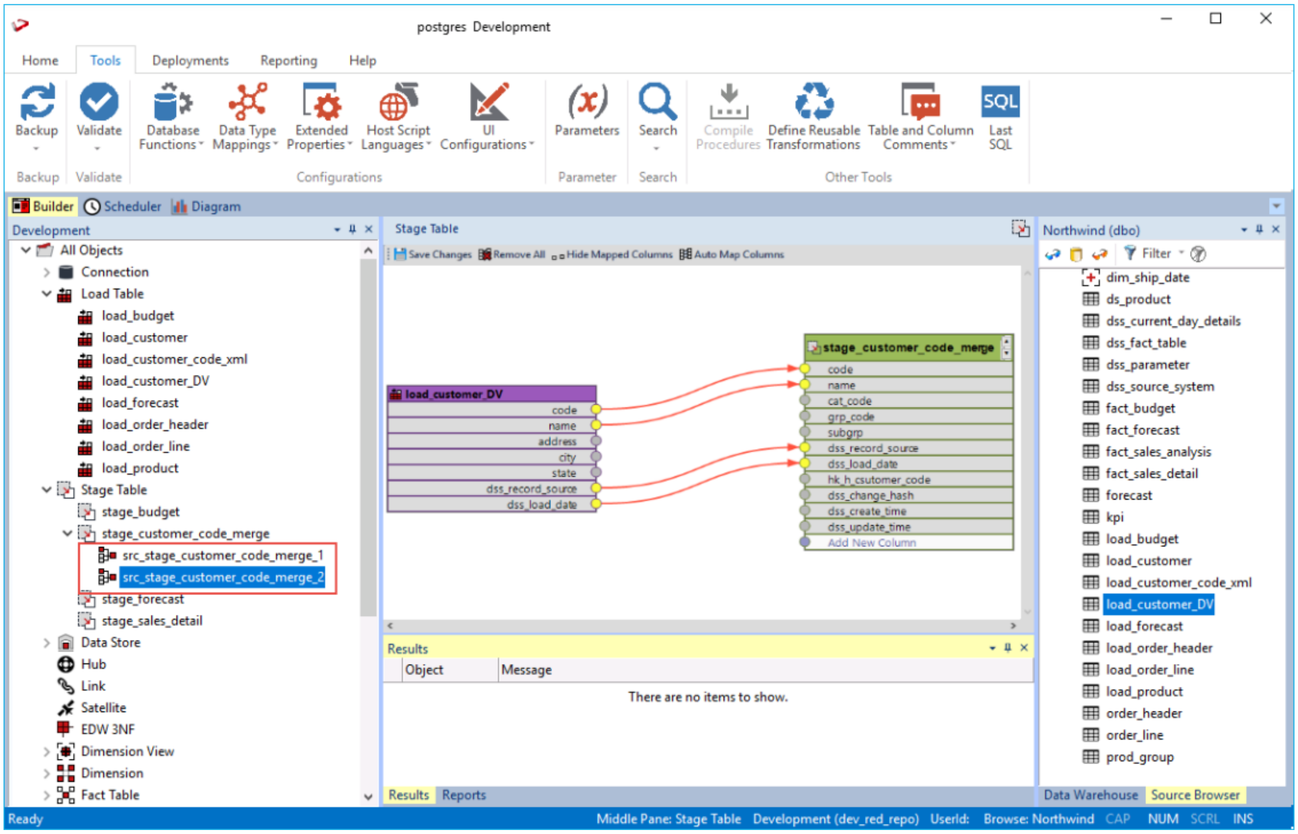
The table load_customer_DV has been dropped onto the table stage_customer_code_merge.

Please choose an action with which to proceed.

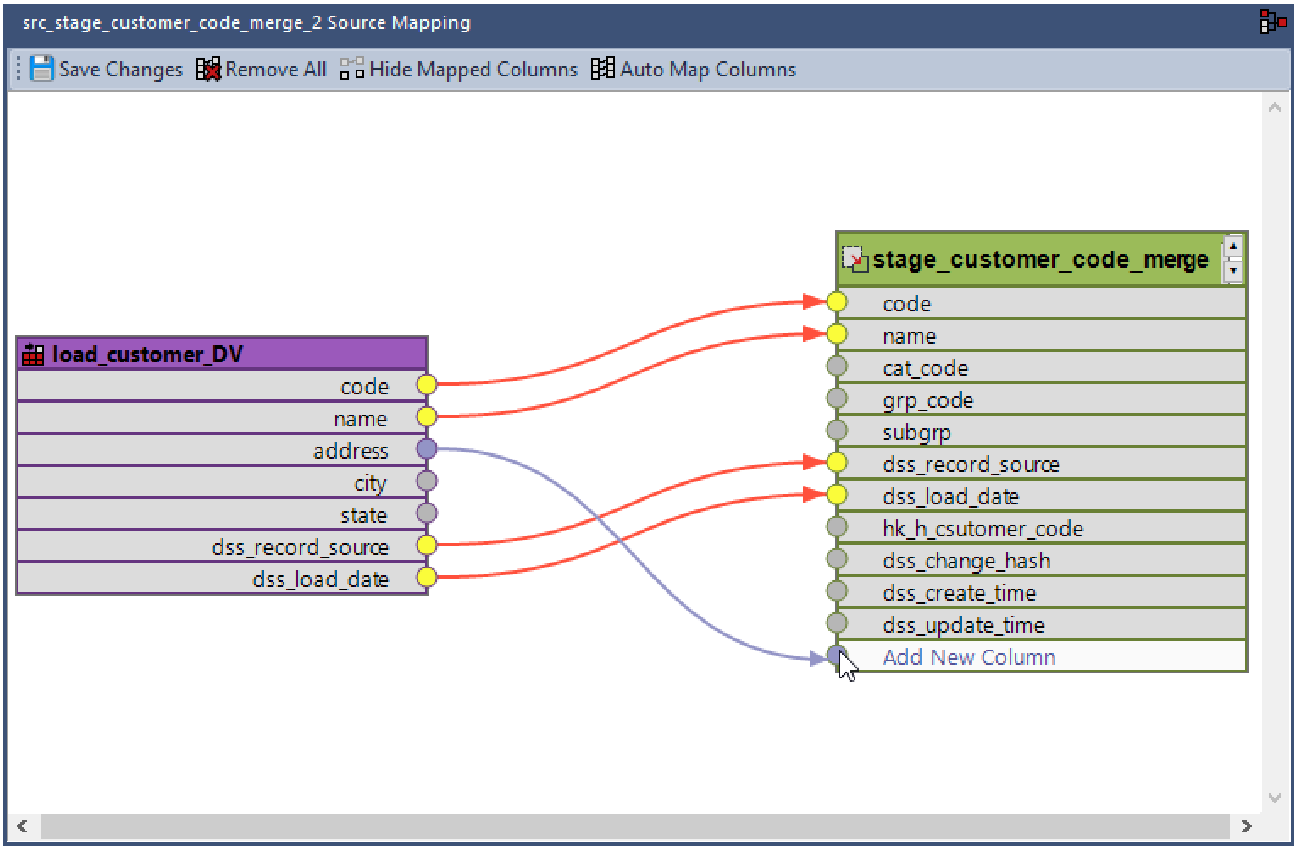
New Table	Invoke the Create New Object dialog.
Add Columns	Begin the process of adding the source table's columns into this table.
Add Source Mapping	Create a Source Mapping and manage it with the Source Mapping View.
Cancel	Dismiss this dialog, do nothing else.

5. The new Source Mapping objects are created and displayed under the target table in the **Objects** list pane.

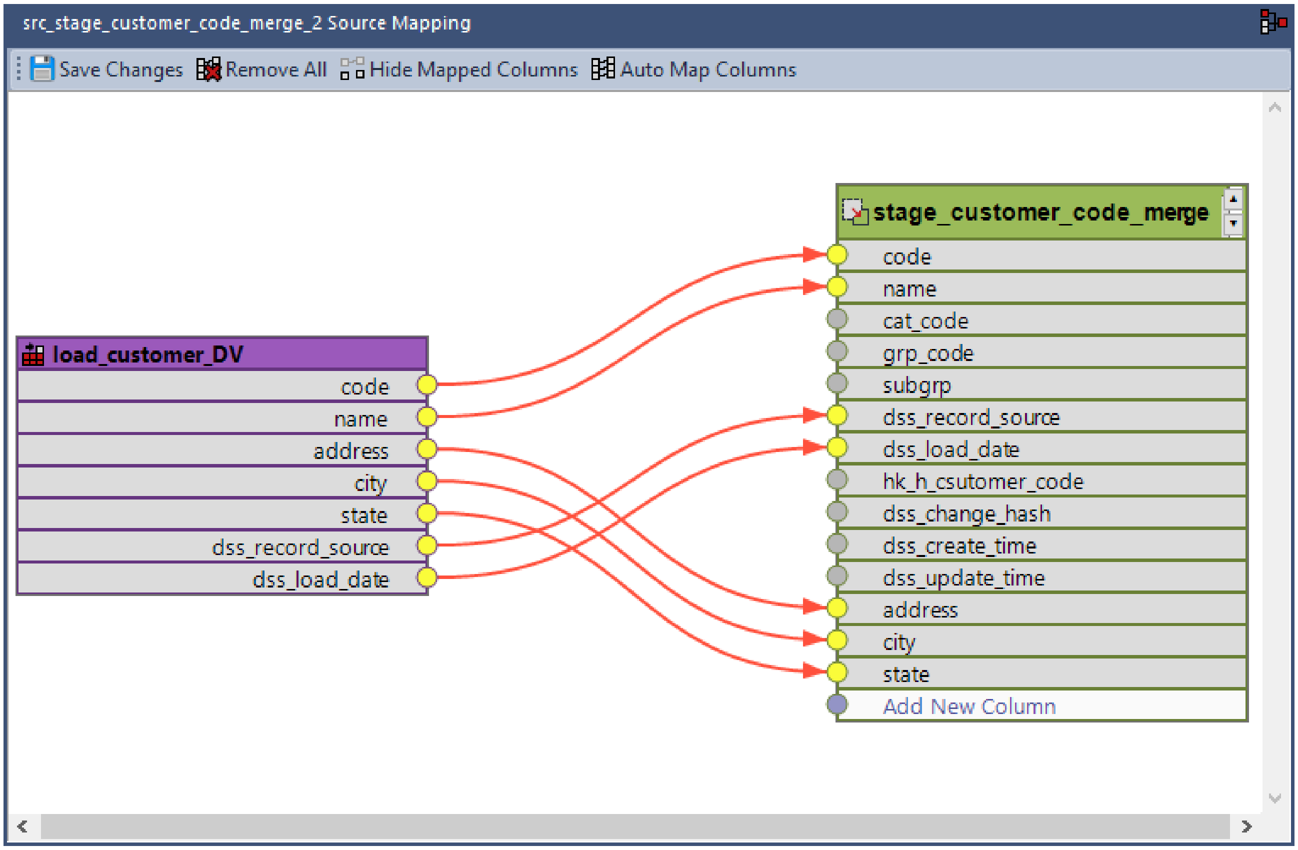
   The middle pane displays the Source Mapping tool which provides a graphical representation of the mapped columns between the second source table and the target table. The columns from the source table that exists in the target table are automatically mapped.



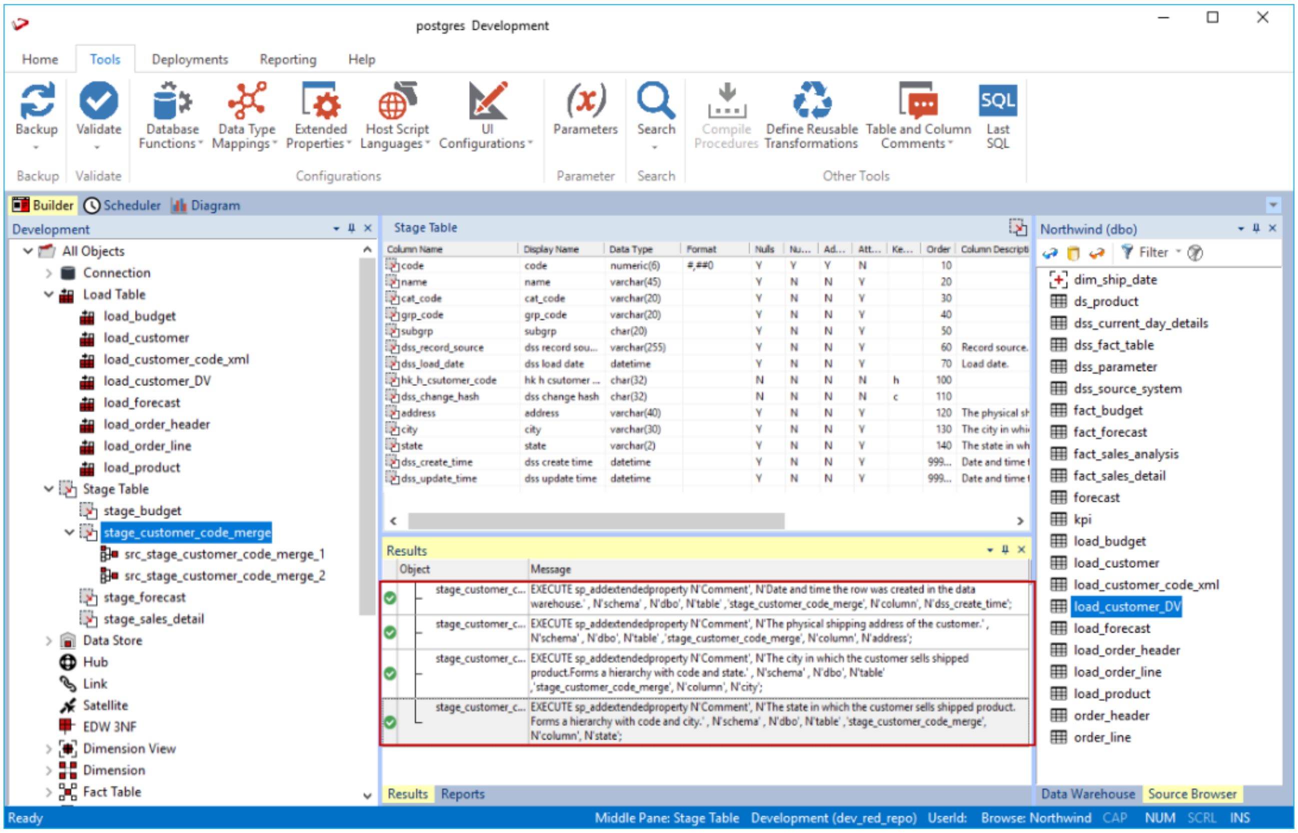
6. To map the additional source columns that does not exist in the target table, click the connection point of the source column you want to map and drag the line that appears to the connection point of the **Add New Column** row in the target table.



7. Repeat the same steps to map all the other source columns to the target table.



8. Click **Save Changes** to save the defined source mappings.
9. Right-click the target table from the **Objects** list in the left pane and select **Create(ReCreate)** from the context menu to recreate the table. The **Results** pane displays confirmation that the target table was successfully recreated.



Column Name	Display Name	Data Type	Format	Nulls	Nu...	Ad...	Att...	Ke...	Order	Column Descripti
code	code	numeric(6)	#,##0	Y	Y	Y	N		10	
name	name	varchar(45)		Y	N	N	Y		20	
cat_code	cat_code	varchar(20)		Y	N	N	Y		30	
grp_code	grp_code	varchar(20)		Y	N	N	Y		40	
subgrp	subgrp	char(20)		Y	N	N	Y		50	
dss_record_source	dss record sou...	varchar(255)		Y	N	N	Y		60	Record source.
dss_load_date	dss load date	datetime		Y	N	N	Y		70	Load date.
hk_h_csutomer_code	hk h csutomer ...	char(32)		N	N	N	N	h	100	
dss_change_hash	dss change hash	char(32)		N	N	N	N	c	110	
address	address	varchar(40)		Y	N	N	Y		120	The physical sh
city	city	varchar(30)		Y	N	N	Y		130	The city in whi
state	state	varchar(2)		Y	N	N	Y		140	The state in wh
dss_create_time	dss create time	datetime		Y	N	N	Y		999...	Date and time t
dss_update_time	dss update time	datetime		Y	N	N	Y		999...	Date and time t

Results

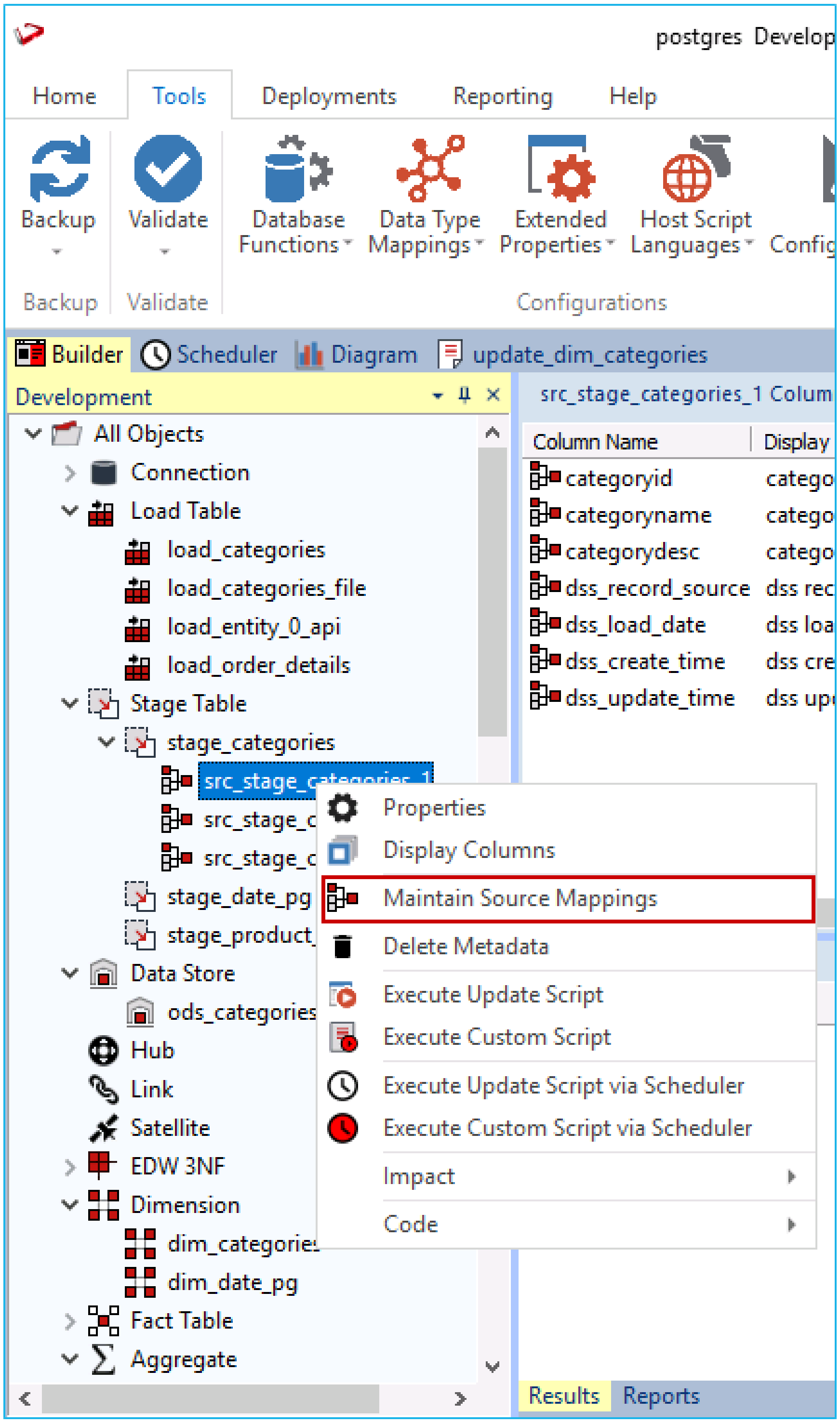
Object	Message
stage_customer_c...	EXECUTE sp_addextendedproperty N'Comment', N'Date and time the row was created in the data warehouse.' , N'schema' , N'dbo', N'table' ,'stage_customer_code_merge', N'column', N'dss_create_time';
stage_customer_c...	EXECUTE sp_addextendedproperty N'Comment', N'The physical shipping address of the customer.' , N'schema' , N'dbo', N'table' ,'stage_customer_code_merge', N'column', N'address';
stage_customer_c...	EXECUTE sp_addextendedproperty N'Comment', N'The city in which the customer sells shipped product.Forms a hierarchy with code and state.' , N'schema' , N'dbo', N'table' ,'stage_customer_code_merge', N'column', N'city';
stage_customer_c...	EXECUTE sp_addextendedproperty N'Comment', N'The state in which the customer sells shipped product. Forms a hierarchy with code and city.' , N'schema' , N'dbo', N'table' , 'stage_customer_code_merge', N'column', N'state';

Once the target table is defined and created, you need to generate the update procedure for each source mapping object to populate the target table. The update procedure is created using a template—refer to Generating Update Procedures for Source Mapping Objects for details.

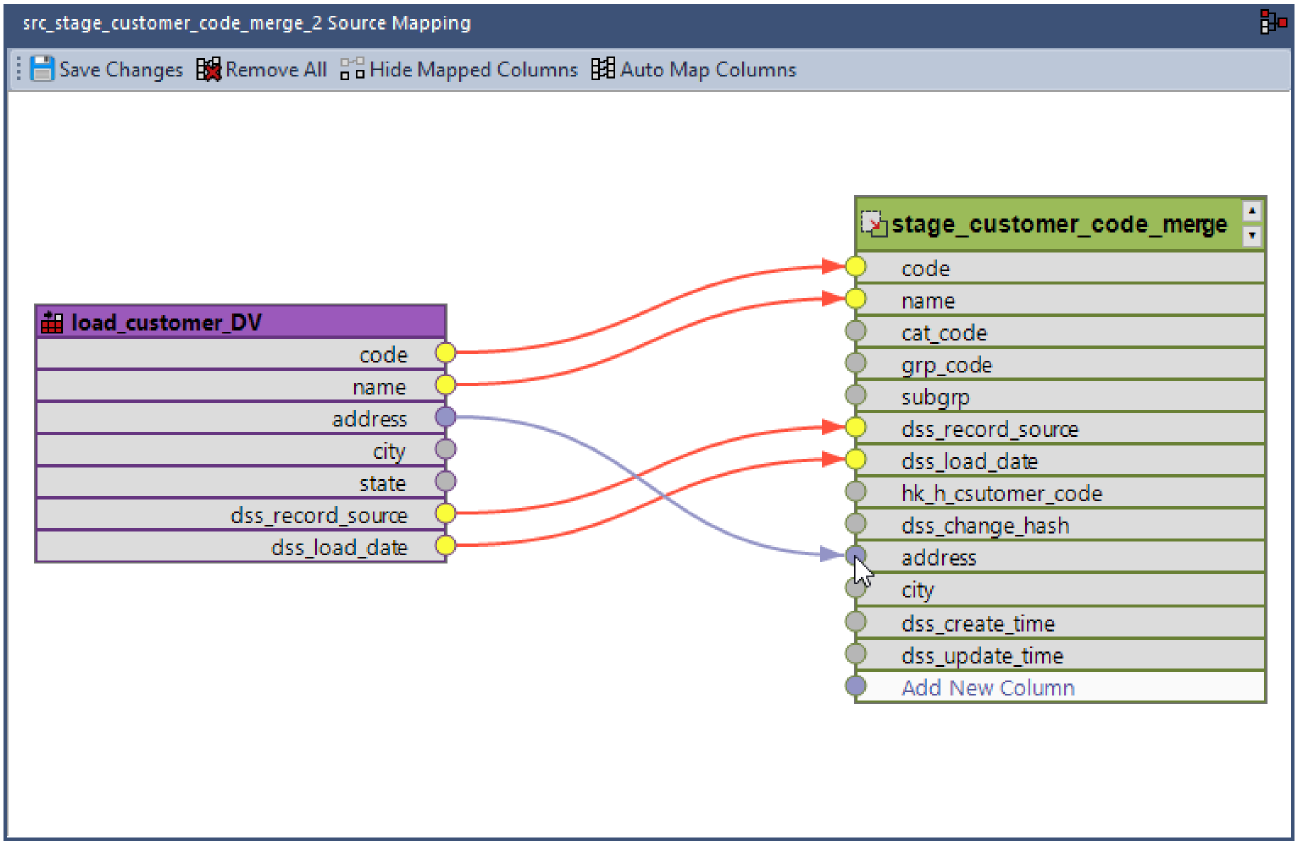
Maintaining Source Column Mappings

The following describe the process for managing source mappings between the source table(s) and the target table.

1. Right click the Source Mapping object from the left-pane and then select **Maintain Source Mappings** from the context menu.



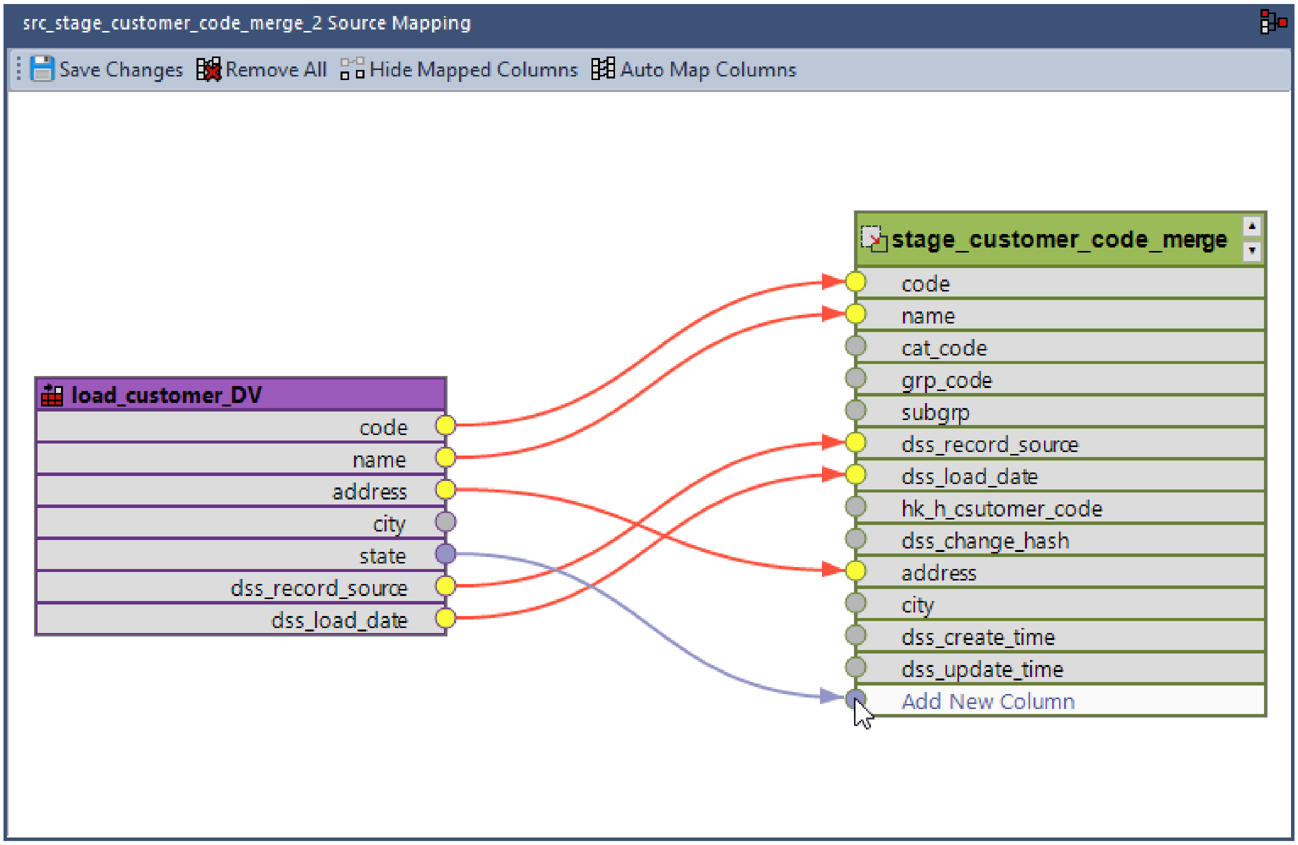
2. Click the connection point of the source column you want to map and then drag the line that appears to the connection point of the target column.



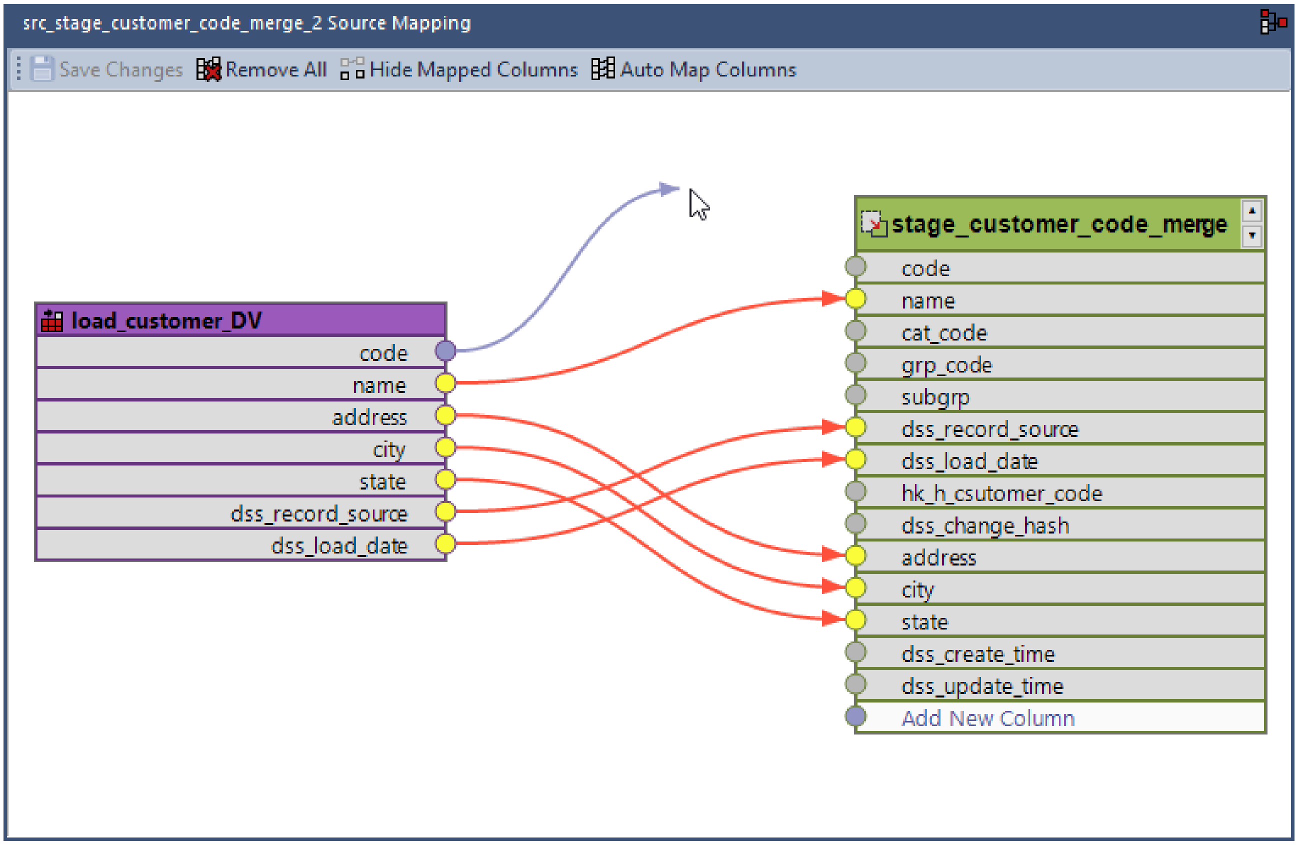
Tips:

- The target table can be clicked and dragged vertically (up/down) within the diagram view. This is useful when there are more columns in the target table that can fit in the visible area of the diagram view. You can also use the mouse wheel for scrolling up/down the diagram view.
- You can undock any of the panes (**Object**, **Browser** and **Results** pane) in the **Builder window**, to gain a bigger work area for the Source Mapping tool.

3. To map a source column that does not exist in the target table, click the connection point of the source column you want to map and then drag the line that appears to the connection point of the **Add New Column** row in the target table.



4. To remove an existing source mapping, click the connection point from the target table and then drag away the line that appears to disconnect.



Notes:

- Clicking **Remove All** removes all the current mappings between the source table and the target table.
- Clicking **Hide Mapped Columns** hides all the current mappings and only the unmapped columns are displayed. This is useful when there are more columns in the target table that can fit in the visible area of the diagram view and you only want to work on the columns that are not mapped.
- Clicking **Auto Map Columns** automatically maps all columns from the source table that exists in the target table. This is useful if you removed all or some of the mappings and would like to restore them.

Generating Update Procedures for Source Mapping Objects

After successfully defining and creating the Source Mapping objects, you can generate the update procedure for each object via a template to populate the target table.

Generating update and custom procedures for Source Mapping objects is completed using the same workflow as all objects in RED that support template based procedure/script generation.

Executing a procedure associated with the Source Mapping object can be performed both interactively from the object's right-click context menu or via the Scheduler.

Notes:

- Please ensure that you have created your own Source Mapping object templates, before performing the steps below.
- You can also have an update or custom procedure defined on the target table, if a single consolidated procedure to process the table is required.

1. Click the **Rebuild** button beside the **Update Procedure** drop-down in the **Properties** screen of the Source Mapping object, to launch the procedure generation Wizard.

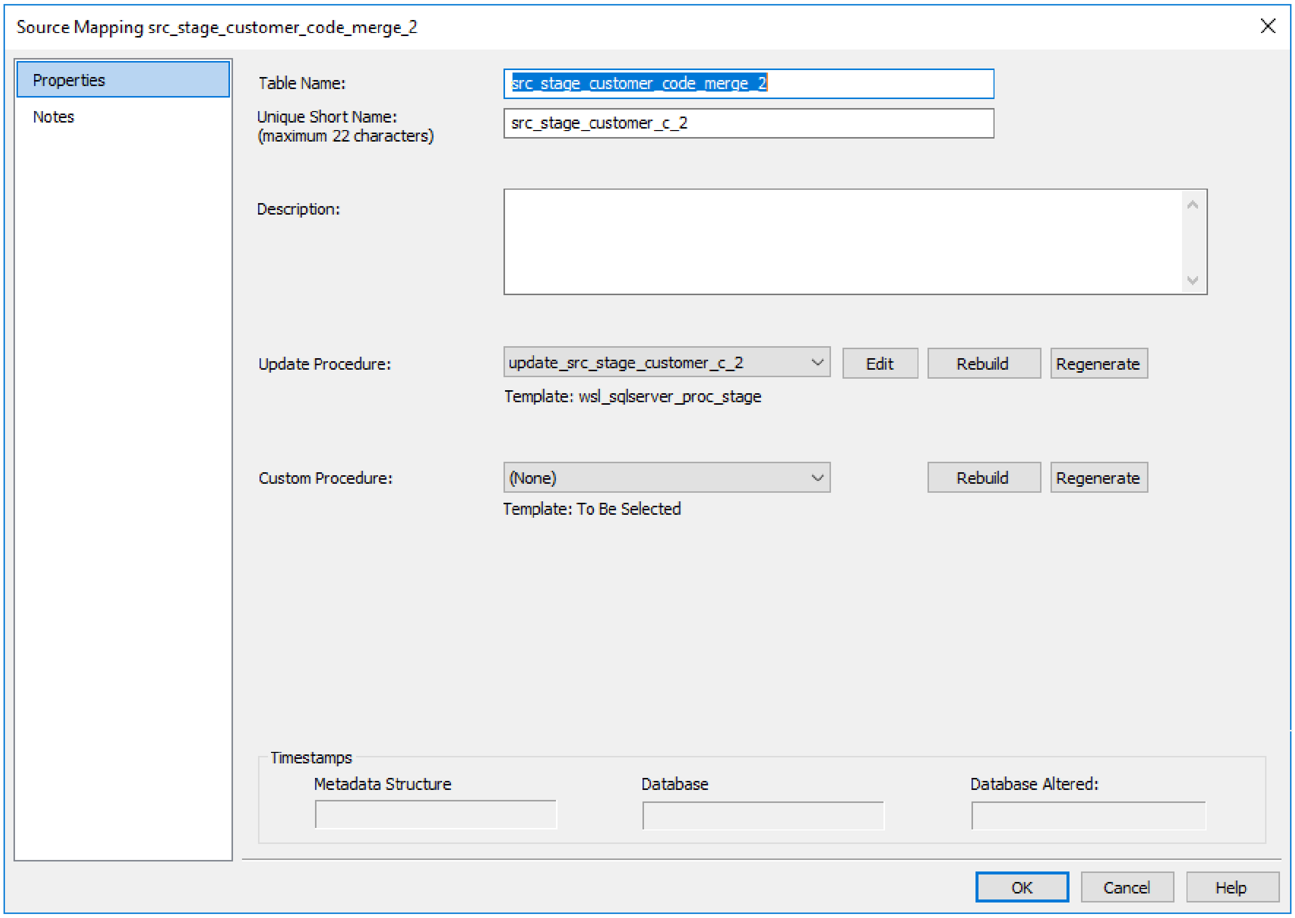

Source Mapping src_stage_customer_code_merge_2

Table Name: src_stage_customer_code_merge_2

Unique Short Name: (maximum 22 characters) src_stage_customer_c_2

Description:

Update Procedure: update_src_stage_customer_c_2

Template: wsl_sqlserver_proc_stage

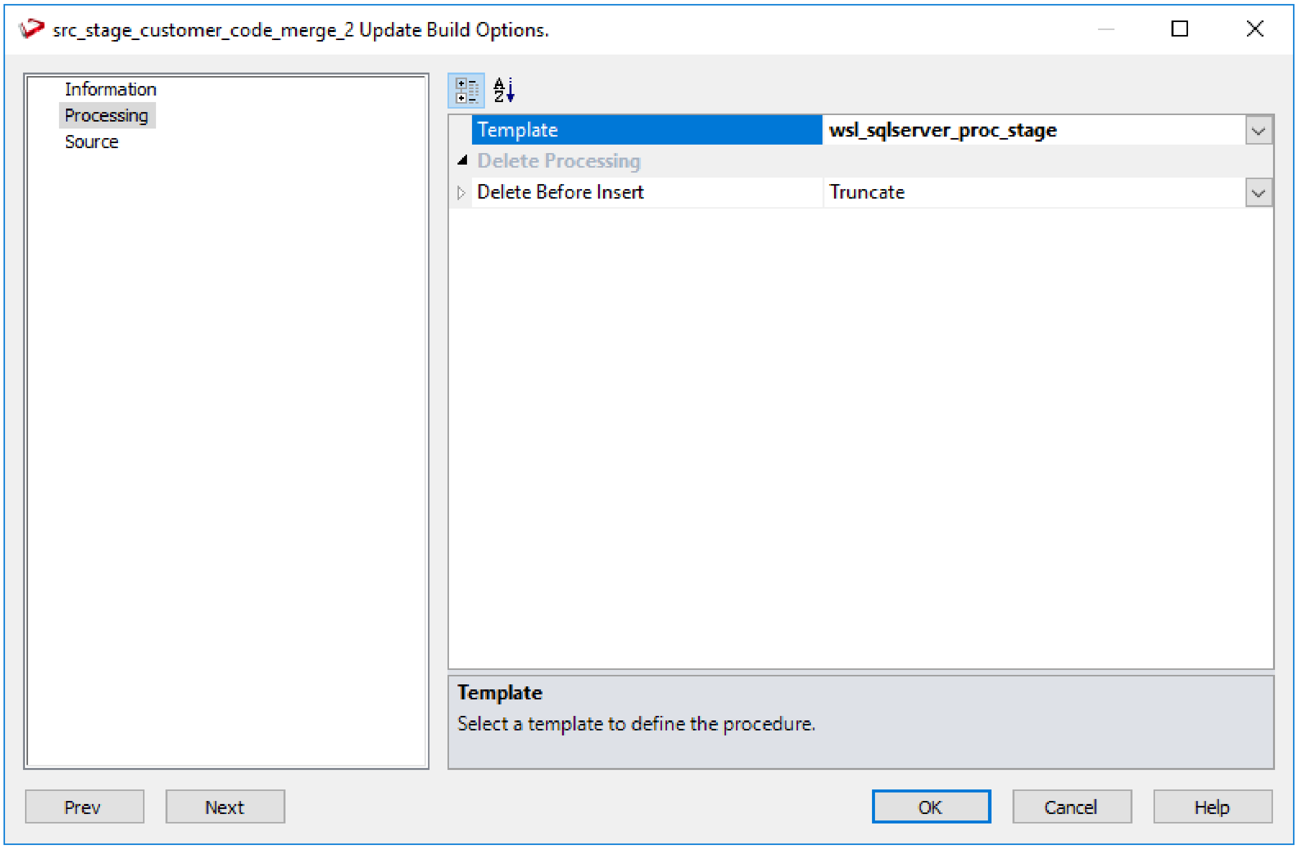
Custom Procedure: (None)

Template: To Be Selected

Note:

RED displays the name of the previously used update/custom procedure template below the **Update Procedure** and **Custom Procedure** drop-down fields by default.

2. On the **Processing** tab of the **Update Build Options** screen, select the template to use from the **Template** drop-down or use the previous update procedure template. Configure the other options available as required.



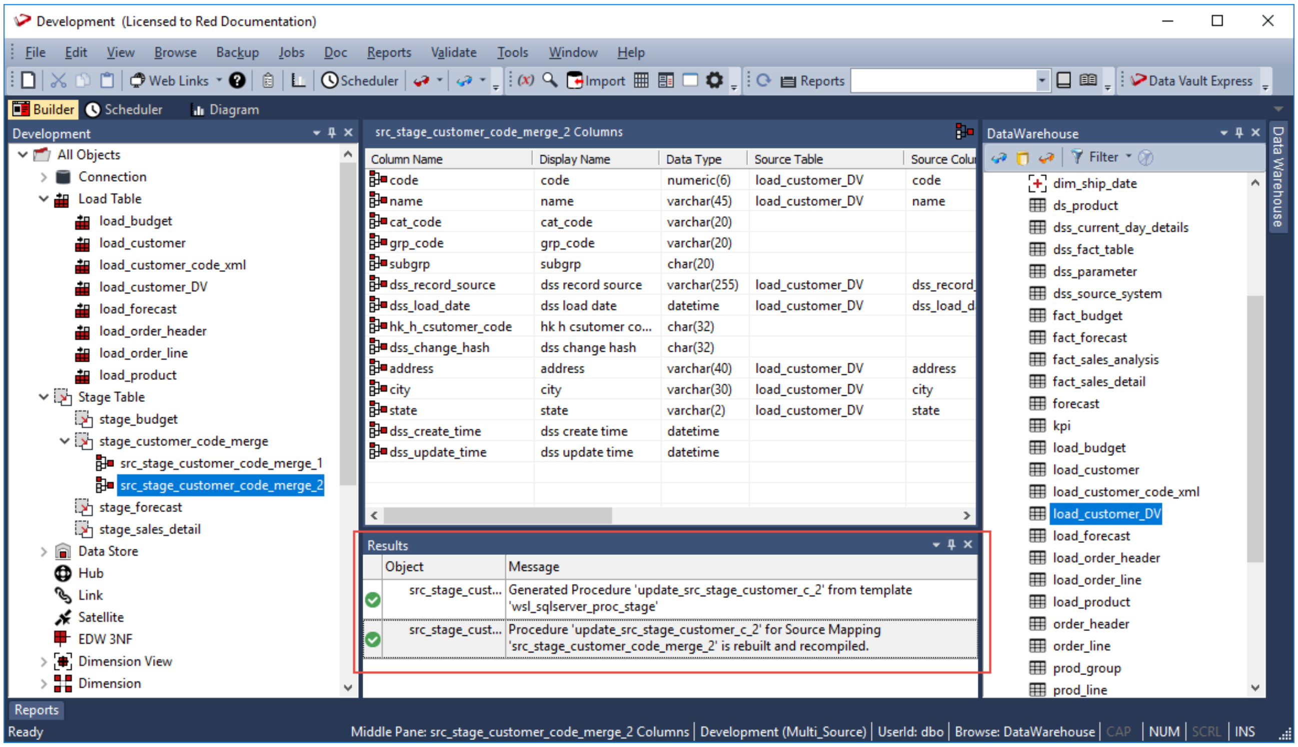
src_stage_customer_code_merge_2 Update Build Options.

Template	wsl_sqlserver_proc_stage
Delete Processing	
Delete Before Insert	Truncate

Template

Select a template to define the procedure.

3. Click **OK** to proceed with the procedure generation. The **Results** pane displays confirmation that the procedure was generated.



src_stage_customer_code_merge_2 Columns

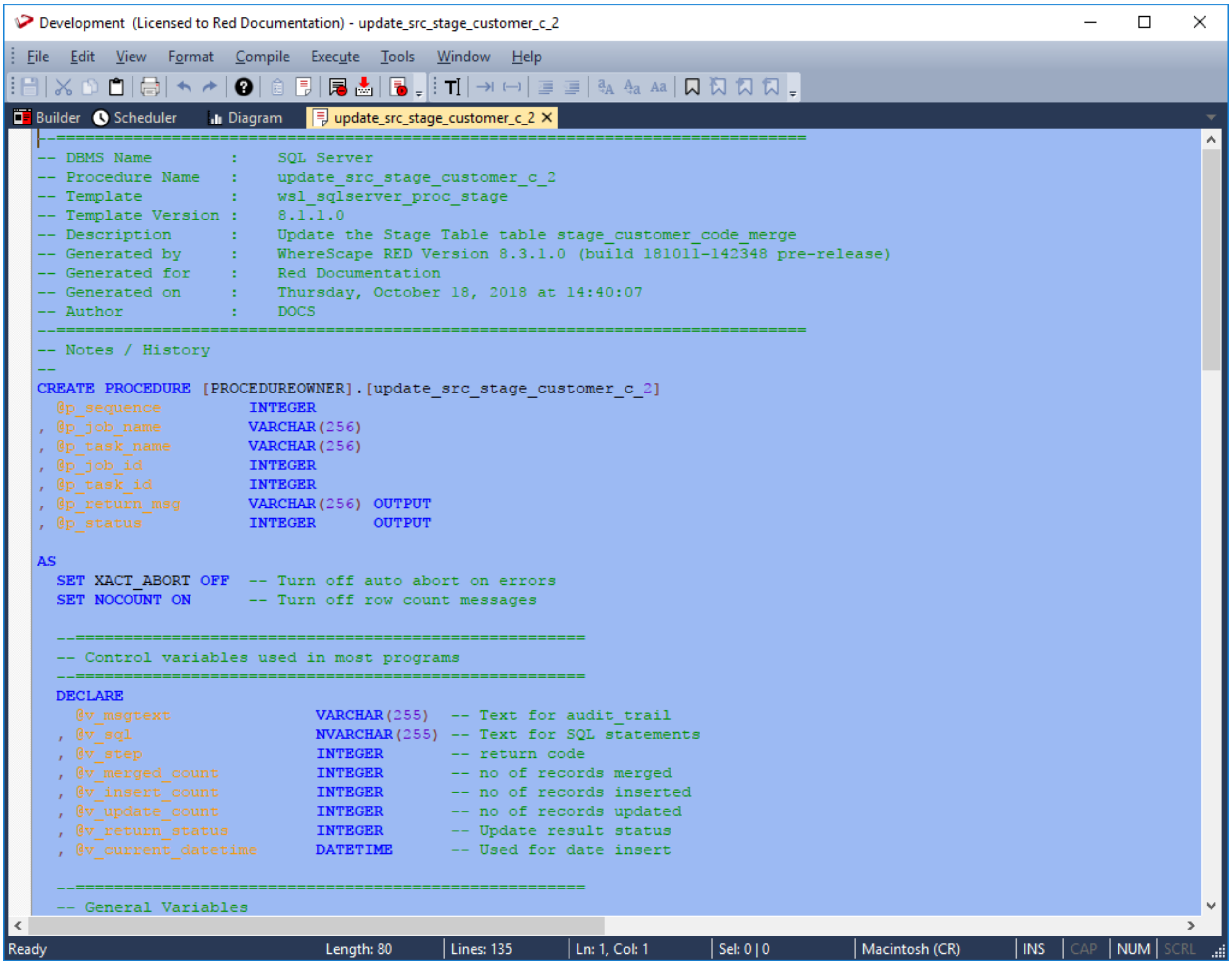
Column Name	Display Name	Data Type	Source Table	Source Colu
code	code	numeric(6)	load_customer_DV	code
name	name	varchar(45)	load_customer_DV	name
cat_code	cat_code	varchar(20)		
grp_code	grp_code	varchar(20)		
subgrp	subgrp	char(20)		
dss_record_source	dss record source	varchar(255)	load_customer_DV	dss_record_
dss_load_date	dss load date	datetime	load_customer_DV	dss_load_d
hk_h_csutomer_code	hk h csutomer co...	char(32)		
dss_change_hash	dss change hash	char(32)		
address	address	varchar(40)	load_customer_DV	address
city	city	varchar(30)	load_customer_DV	city
state	state	varchar(2)	load_customer_DV	state
dss_create_time	dss create time	datetime		
dss_update_time	dss update time	datetime		

Results

	Object	Message
✓	src_stage_cust...	Generated Procedure 'update_src_stage_customer_c_2' from template 'wsl_sqlserver_proc_stage'
✓	src_stage_cust...	Procedure 'update_src_stage_customer_c_2' for Source Mapping 'src_stage_customer_code_merge_2' is rebuilt and recompiled.

Ready | Middle Pane: src_stage_customer_code_merge_2 Columns | Development (Multi_Source) | UserId: dbo | Browse: DataWarehouse | CAP | NUM | SCRL | INS

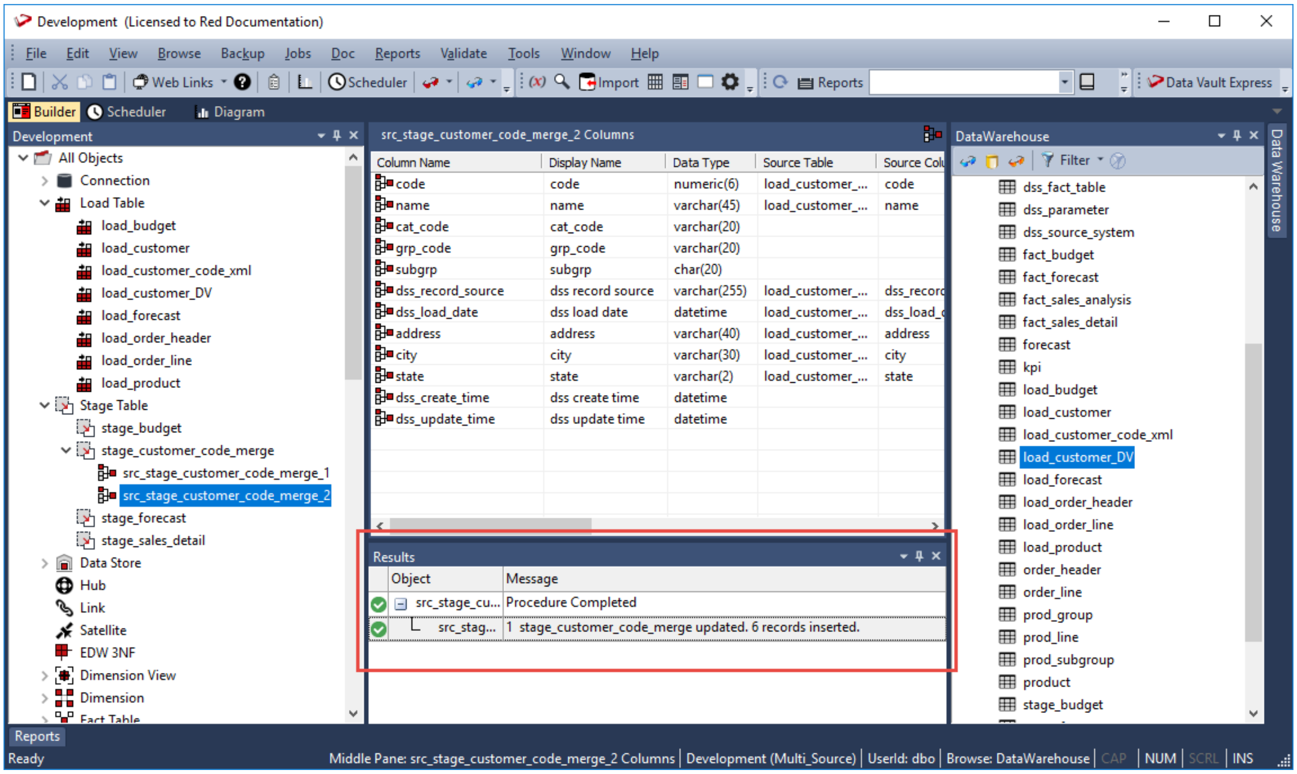
4. Right-click the source table from the left pane, and select **Code > View update** from the context menu to view the contents of the update procedure generated.



Development (Licensed to Red Documentation) - update_src_stage_customer_c_2
File Edit View Format Compile Execute Tools Window Help
Builder Scheduler Diagram update_src_stage_customer_c_2

```
--==============================================================================
-- DBMS Name        :    SQL Server
-- Procedure Name   :    update_src_stage_customer_c_2
-- Template         :    wsl_sqlserver_proc_stage
-- Template Version :    8.1.1.0
-- Description      :    Update the Stage Table table stage_customer_code_merge
-- Generated by     :    WhereScape RED Version 8.3.1.0 (build 181011-142348 pre-release)
-- Generated for    :    Red Documentation
-- Generated on     :    Thursday, October 18, 2018 at 14:40:07
-- Author           :    DOCS
--==============================================================================
-- Notes / History
--
CREATE PROCEDURE [PROCEDUREOWNER].[update_src_stage_customer_c_2]
  @p_sequence         INTEGER
, @p_job_name         VARCHAR(256)
, @p_task_name        VARCHAR(256)
, @p_job_id           INTEGER
, @p_task_id          INTEGER
, @p_return_msg       VARCHAR(256) OUTPUT
, @p_status           INTEGER      OUTPUT

AS
  SET XACT_ABORT OFF  -- Turn off auto abort on errors
  SET NOCOUNT ON      -- Turn off row count messages

  --=====================================================
  -- Control variables used in most programs
  --=====================================================
  DECLARE
    @v_msgtext               VARCHAR(255)  -- Text for audit_trail
  , @v_sql                   NVARCHAR(255) -- Text for SQL statements
  , @v_step                  INTEGER       -- return code
  , @v_merged_count          INTEGER       -- no of records merged
  , @v_insert_count          INTEGER       -- no of records inserted
  , @v_update_count          INTEGER       -- no of records updated
  , @v_return_status         INTEGER       -- Update result status
  , @v_current_datetime      DATETIME      -- Used for date insert

  --=====================================================
  -- General Variables
```

5. Right-click the source table from the left pane and select **Execute Update Procedure** from the context menu to run the procedure. The **Results** pane displays the number of records created.



src_stage_customer_code_merge_2 Columns

Column Name	Display Name	Data Type	Source Table	Source Colu
code	code	numeric(6)	load_customer_...	code
name	name	varchar(45)	load_customer_...	name
cat_code	cat_code	varchar(20)		
grp_code	grp_code	varchar(20)		
subgrp	subgrp	char(20)		
dss_record_source	dss record source	varchar(255)	load_customer_...	dss_recor
dss_load_date	dss load date	datetime	load_customer_...	dss_load_
address	address	varchar(40)	load_customer_...	address
city	city	varchar(30)	load_customer_...	city
state	state	varchar(2)	load_customer_...	state
dss_create_time	dss create time	datetime		
dss_update_time	dss update time	datetime		

Results

Object	Message
src_stage_cu...	Procedure Completed
src_stag...	1 stage_customer_code_merge updated. 6 records inserted.

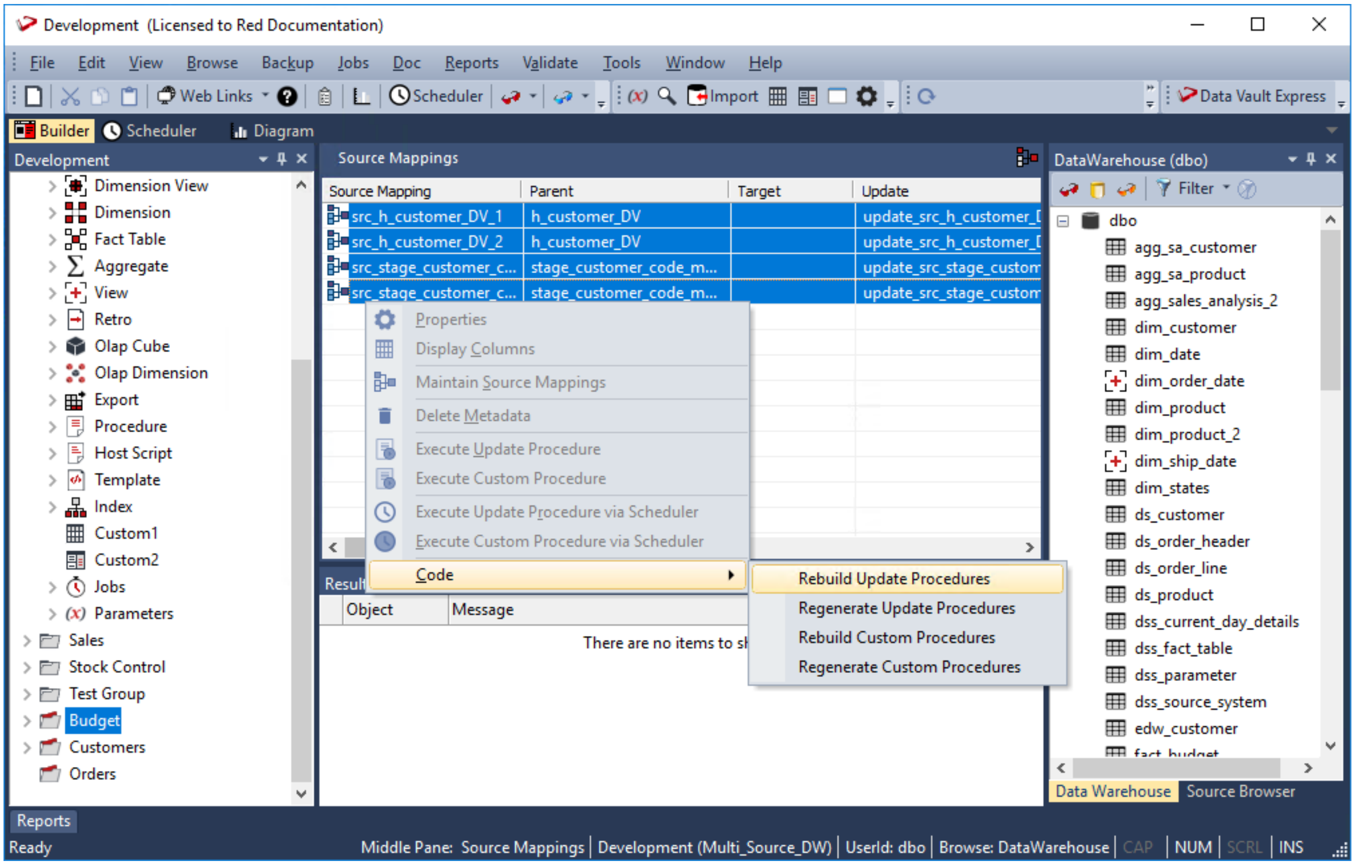
Ready | Middle Pane: src_stage_customer_code_merge_2 Columns | Development (Multi_Source) | UserId: dbo | Browse: DataWarehouse | CAP | NUM | SCRL | INS

Repeat the same steps described above to create update procedures for the other source mapping objects, defined for the target table.

Bulk Update Procedure Regeneration

Once the update procedure code has been generated for the source mapping objects, RED enables you to perform bulk code regeneration in source mapping objects from multiple tables.

You can list all the source mapping objects in a Project, Project Group or Object Group and then perform bulk update via the right click context menu:



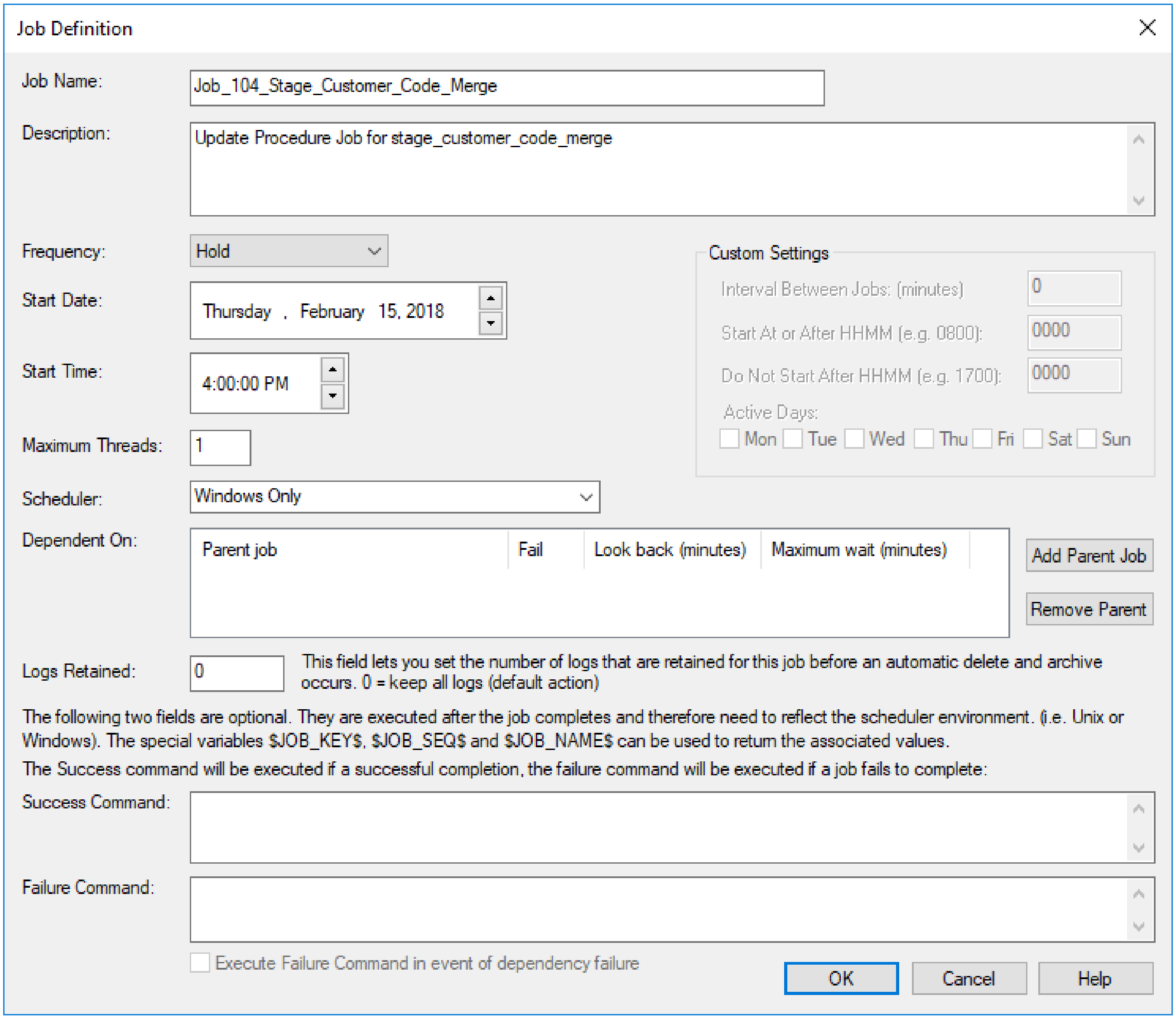
Source Mapping	Parent	Target	Update
src_h_customer_DV_1	h_customer_DV		update_src_h_customer_[
src_h_customer_DV_2	h_customer_DV		update_src_h_customer_[
src_stage_customer_c...	stage_customer_code_m...		update_src_stage_custom
src_stage_customer_c...	stage_customer_code_m...		update_src_stage_custom

Refer to **Listing Source Mapping Objects** for details

Executing Update Procedures via Scheduler

The update procedures associated with the target table and the source mapping objects can be executed individually, separate from each other. Similar to other table objects in RED, the update procedures can be executed via the Scheduler.

1. Click the **Scheduler** tab to open the **Scheduler** window.
2. Click the **New Job** button from the **Scheduler** window toolbar to create a new job
3. Complete the fields in the **Job Definition** window. Refer to **Creating a Job** for the descriptions of the fields.


Job Definition

Job Name: Job_104_Stage_Customer_Code_Merge

Description: Update Procedure Job for stage_customer_code_merge

Frequency: Hold

Start Date: Thursday , February 15, 2018

Start Time: 4:00:00 PM

Maximum Threads: 1

Scheduler: Windows Only

Custom Settings

Interval Between Jobs: (minutes) 0

Start At or After HHMM (e.g. 0800): 0000

Do Not Start After HHMM (e.g. 1700): 0000

Active Days:
□ Mon □ Tue □ Wed □ Thu □ Fri □ Sat □ Sun

Dependent On:

Parent job	Fail	Look back (minutes)	Maximum wait (minutes)

Logs Retained: 0 This field lets you set the number of logs that are retained for this job before an automatic delete and archive occurs. 0 = keep all logs (default action)

The following two fields are optional. They are executed after the job completes and therefore need to reflect the scheduler environment. (i.e. Unix or Windows). The special variables JOB_KEY, JOB_SEQ and JOB_NAME can be used to return the associated values.

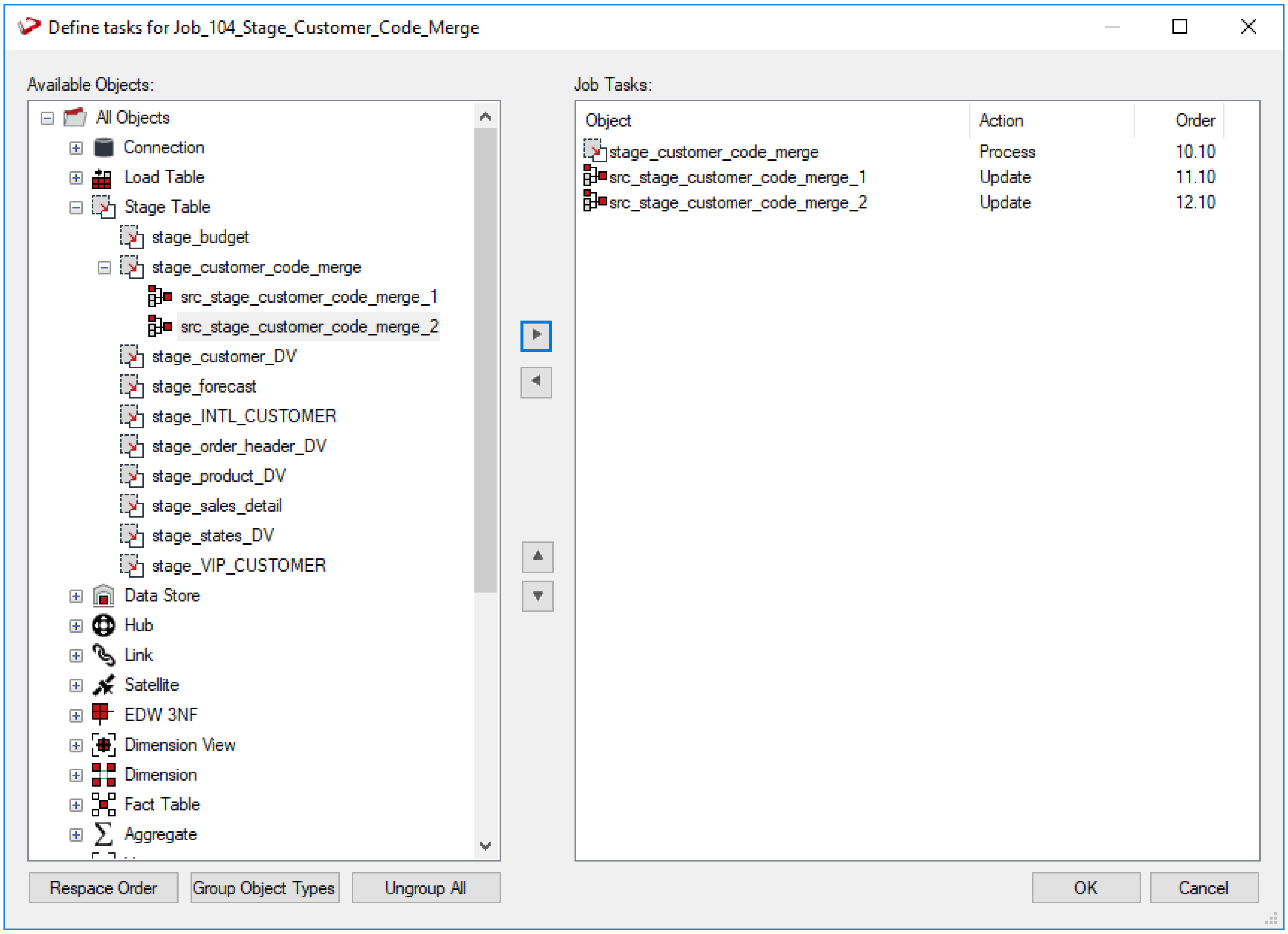
The Success command will be executed if a successful completion, the failure command will be executed if a job fails to complete:

Success Command:

Failure Command:

□ Execute Failure Command in event of dependency failure

4. On the **Define tasks** window, select the target object and the its associated source mapping objects from the **Available Objects** pane and add them to **Job Tasks** pane to define the tasks to be run for the job.

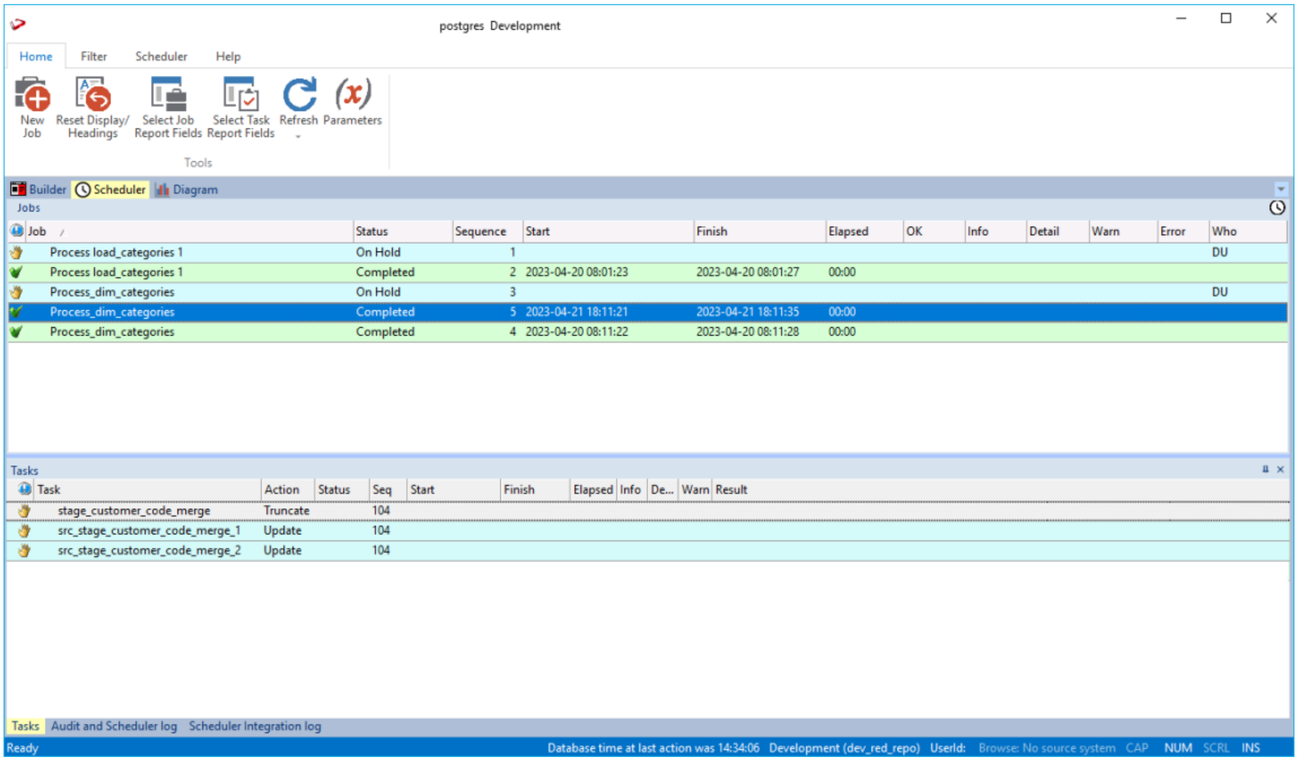


Object	Action	Order
stage_customer_code_merge	Process	10.10
src_stage_customer_code_merge_1	Update	11.10
src_stage_customer_code_merge_2	Update	12.10

Note:

The update procedures associated with the target table and the source mapping objects can be executed individually, separate from each other.

5. Click **OK** to save and exit to the **Scheduler** window.



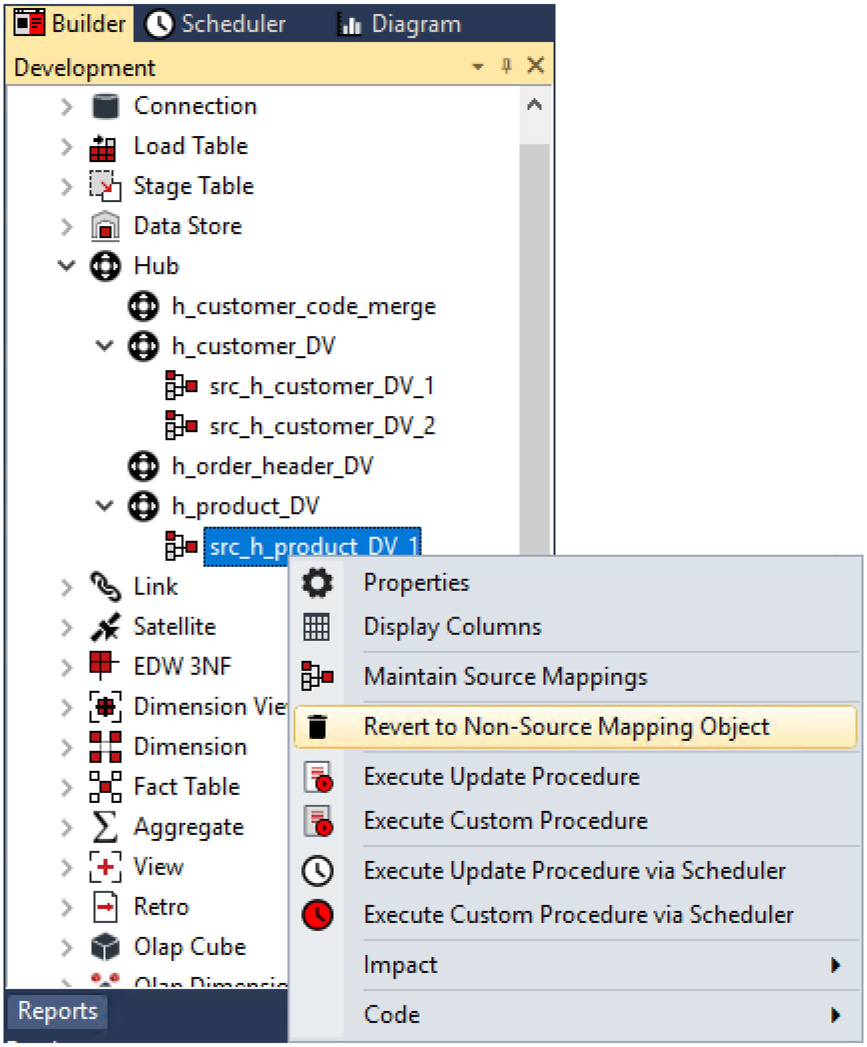
Job	Status	Sequence	Start	Finish	Elapsed	OK	Info	Detail	Warn	Error	Who
Process load_categories 1	On Hold	1									DU
Process load_categories 1	Completed	2	2023-04-20 08:01:23	2023-04-20 08:01:27	00:00						
Process_dim_categories	On Hold	3									DU
Process_dim_categories	Completed	5	2023-04-21 18:11:21	2023-04-21 18:11:35	00:00						
Process_dim_categories	Completed	4	2023-04-20 08:11:22	2023-04-20 08:11:28	00:00						

Task	Action	Status	Seq	Start	Finish	Elapsed	Info	De...	Warn	Result
stage_customer_code_merge	Truncate		104							
src_stage_customer_code_merge_1	Update		104							
src_stage_customer_code_merge_2	Update		104							

Ready   Database time at last action was 14:34:06   Development (dev_red_repo)   UserId: Browse: No source system   CAP   NUM   SCRL   INS

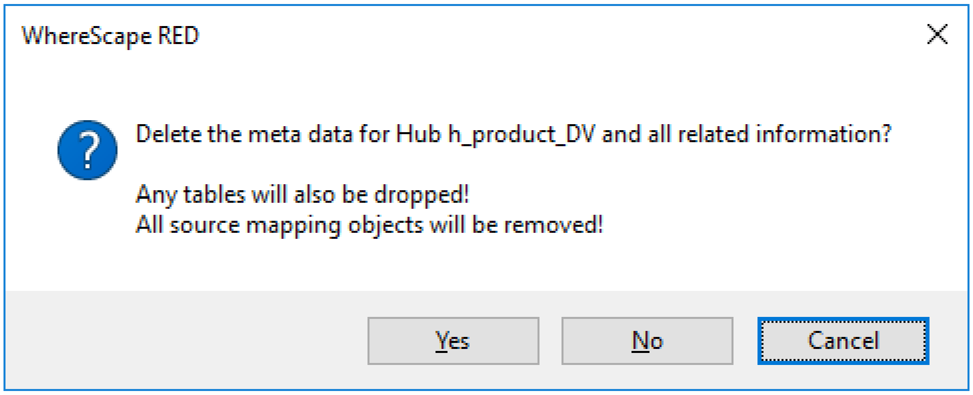
Reverting to Non Source Mapping Object

You can revert the target table to a non-source mapping object if you delete its associated source mapping objects. If only a single source mapping object is left, right clicking the remaining source mapping object provides the option **Revert to Non-Source Mapping Object**.



Note

Deleting the target table, deletes all its source mapping objects, including its associated meta data.


WhereScape RED

Delete the meta data for Hub h_product_DV and all related information?

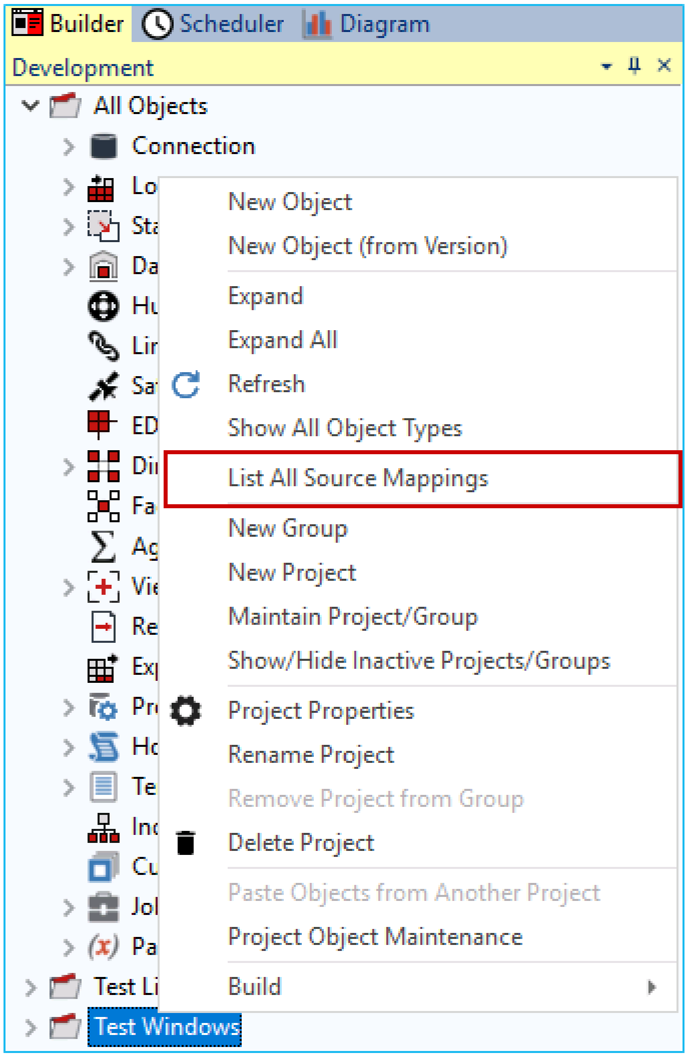
Any tables will also be dropped!
All source mapping objects will be removed!

Listing Source Mapping Objects

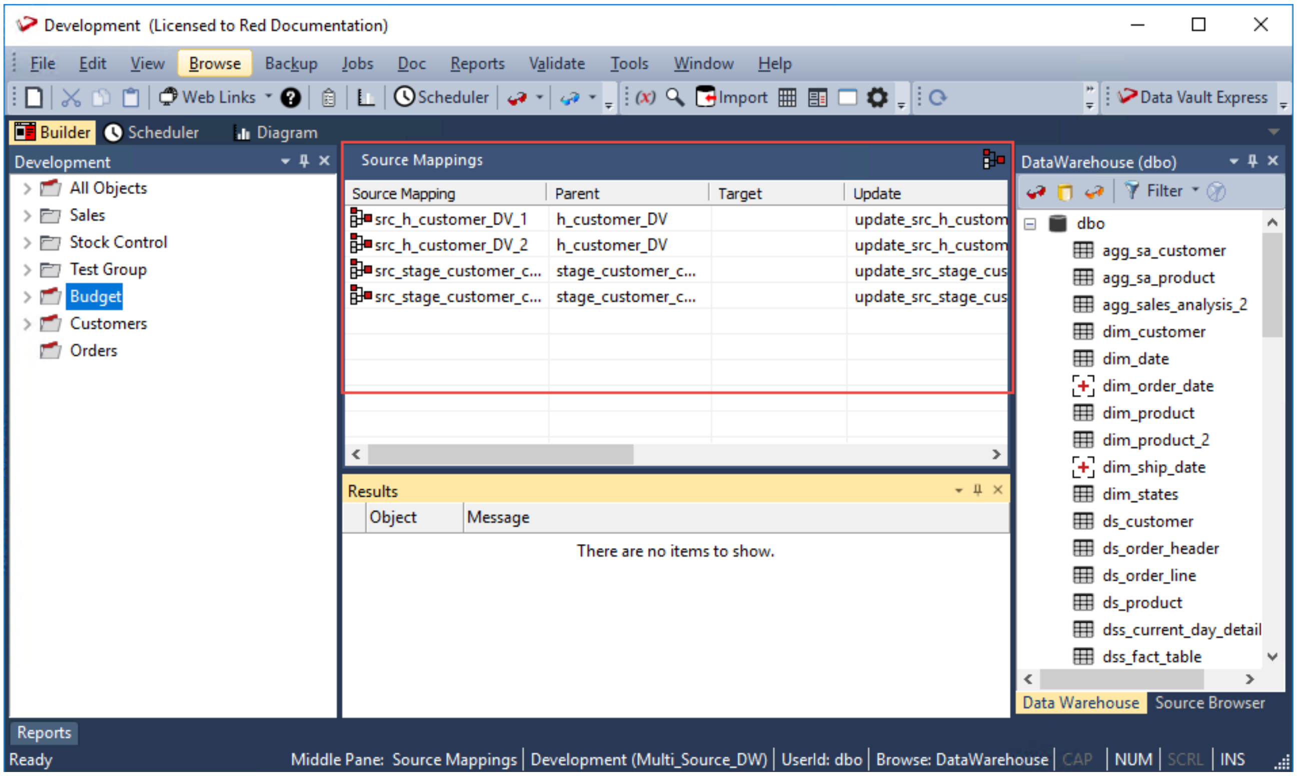
You can view a list of source mapping objects that are defined in a **Project, Project Group or Object Group** in RED. This enables you to quickly view and manage source mapping objects that exist within your data warehouse.

List Source Mapping Objects in a Project or Project Group:

1. Right-click the **Project** or **Project Group** from the left pane and select **List All Source Mappings** from the context menu:



2. The middle pane displays a list of all the source mapping objects in the selected Project or Project Group:



Source Mappings

Source Mapping	Parent	Target	Update
src_h_customer_DV_1	h_customer_DV		update_src_h_custom
src_h_customer_DV_2	h_customer_DV		update_src_h_custom
src_stage_customer_c...	stage_customer_c...		update_src_stage_cus
src_stage_customer_c...	stage_customer_c...		update_src_stage_cus

Results

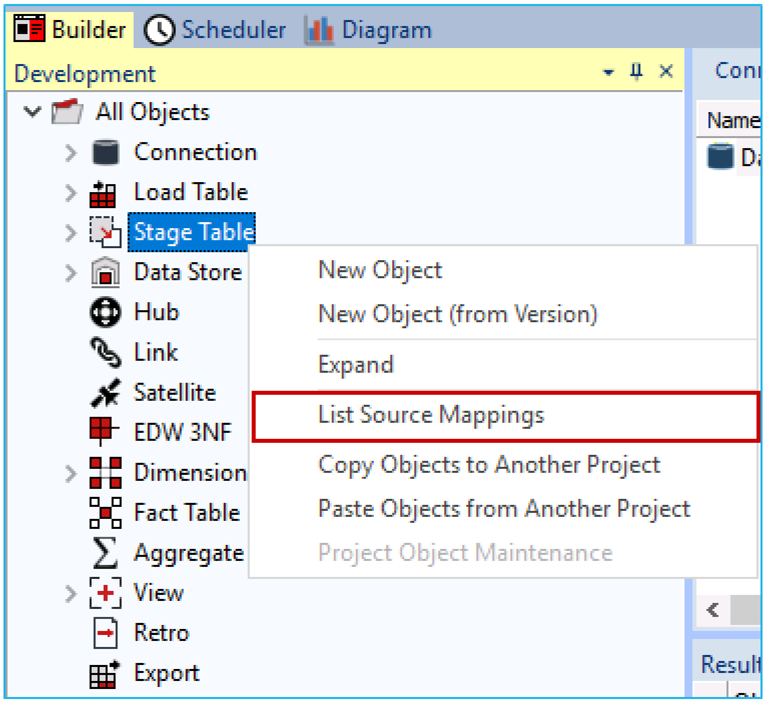
Object	Message

There are no items to show.

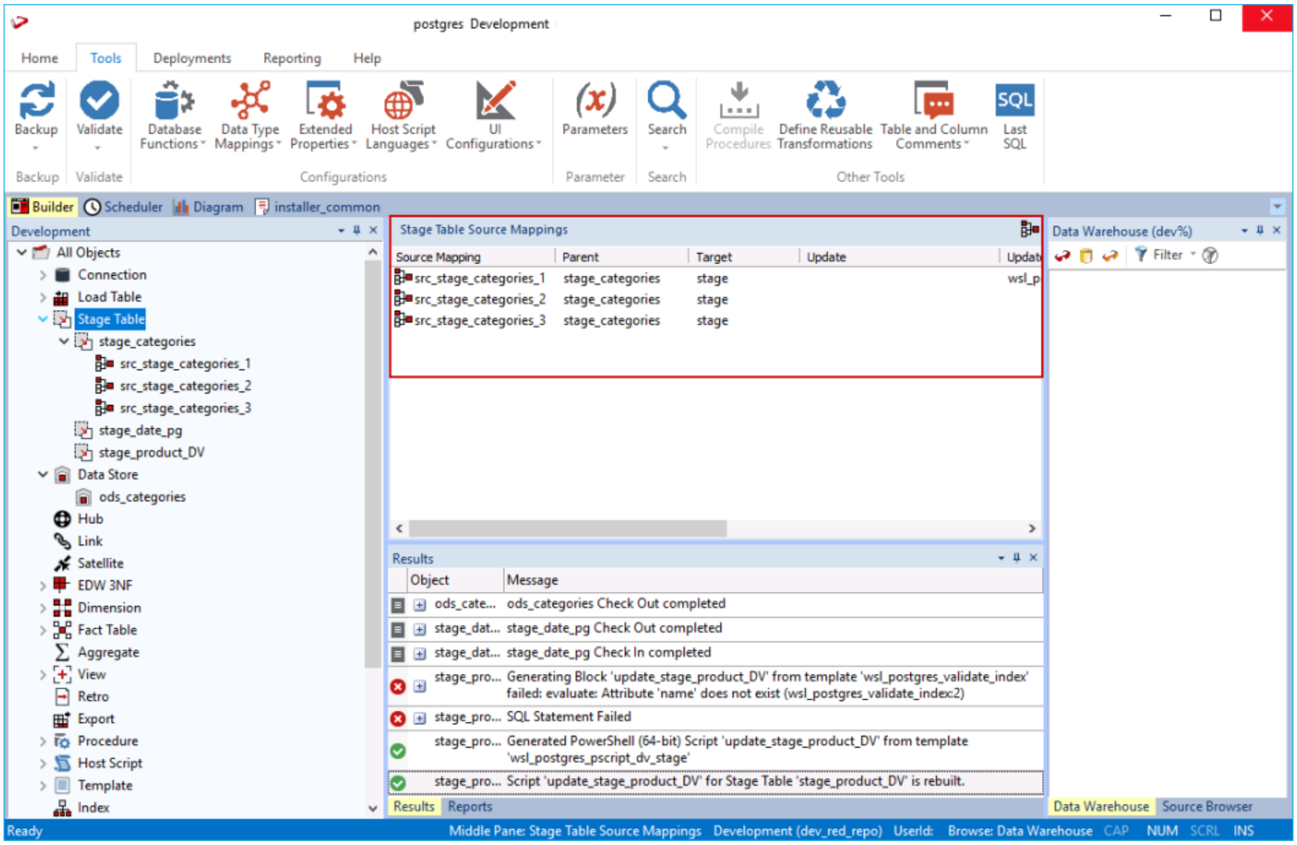
Ready   Middle Pane: Source Mappings | Development (Multi_Source_DW) | UserId: dbo | Browse: DataWarehouse | CAP | NUM | SCRL | INS

List Source Mapping Objects in an Object Group:

1. Right-click the **Object Group** from the left pane and select **List Source Mappings** from the context menu:



2. The middle pane displays a list of all the source mapping objects in the selected Object Group:



Source Mapping	Parent	Target	Update	Updat
src_stage_categories_1	stage_categories	stage		wsl_p
src_stage_categories_2	stage_categories	stage		
src_stage_categories_3	stage_categories	stage		

Object	Message
ods_cate...	ods_categories Check Out completed
stage_dat...	stage_date_pg Check Out completed
stage_dat...	stage_date_pg Check In completed
stage_pro...	Generating Block 'update_stage_product_DV' from template 'wsl_postgres_validate_index' failed: evaluate: Attribute 'name' does not exist (wsl_postgres_validate_index:2)
stage_pro...	SQL Statement Failed
stage_pro...	Generated PowerShell (64-bit) Script 'update_stage_product_DV' from template 'wsl_postgres_pscript_dv_stage'
stage_pro...	Script 'update_stage_product_DV' for Stage Table 'stage_product_DV' is rebuilt.

Ready   Middle Pane: Stage Table Source Mappings   Development (dev_red_repo)   UserId: Browse: Data Warehouse   CAP   NUM   SCRL   INS

Note:

Similar to other RED objects listed in the middle pane, you can select individual or multiple source mapping objects to perform operations available from the right-click context menu (e.g. rebuild update procedures for single or multiple source mapping objects).

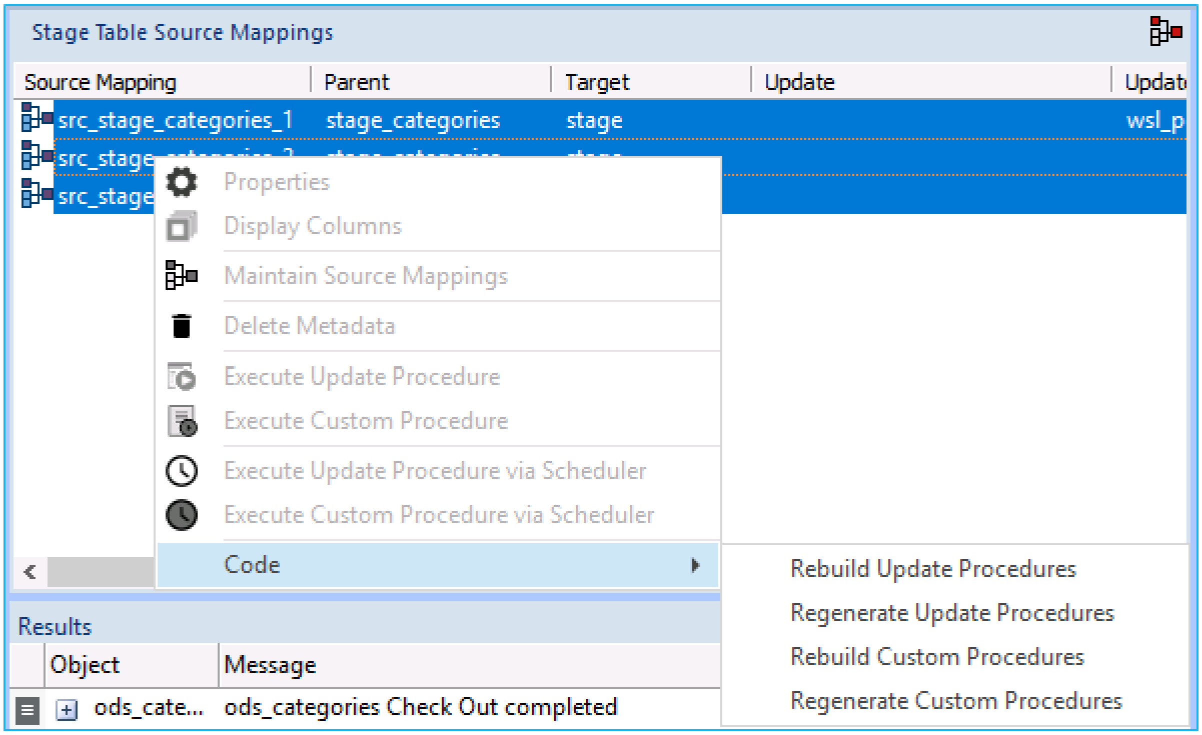


Table and Column Comments

WhereScape RED enables you to manage table and column comments outside your data warehouse environment and then load the updated comments back into the metadata repository; and subsequently copy them to the target database.

The comments can be exported from the metadata repository and imported back, via the **Tools > Table and Column Comments** menu option.

The **export** function enables you to output existing table or column comments into a Microsoft Excel format (.csv) file. You can open the file to view, modify or delete existing comments, as well as create new comments.

The **import** function enables you to load back the updated file into RED and perform a bulk update of the associated metadata.

The **Update Comments** context menu command enables you to copy comments from the metadata repository to the target database, e.g. updates the description and business display name fields (EUL table objects only) in the table and column **Properties** screen.

The following Object types are supported:

- Normal
- Hub
- Satellite
- Link
- Custom1
- Custom2
- Datastore
- Load
- Dimension
- Dimension View
- Stage
- Fact
- Fact KPI
- Aggregate
- Export
- Retro
- Retro Copy
- View
- Join

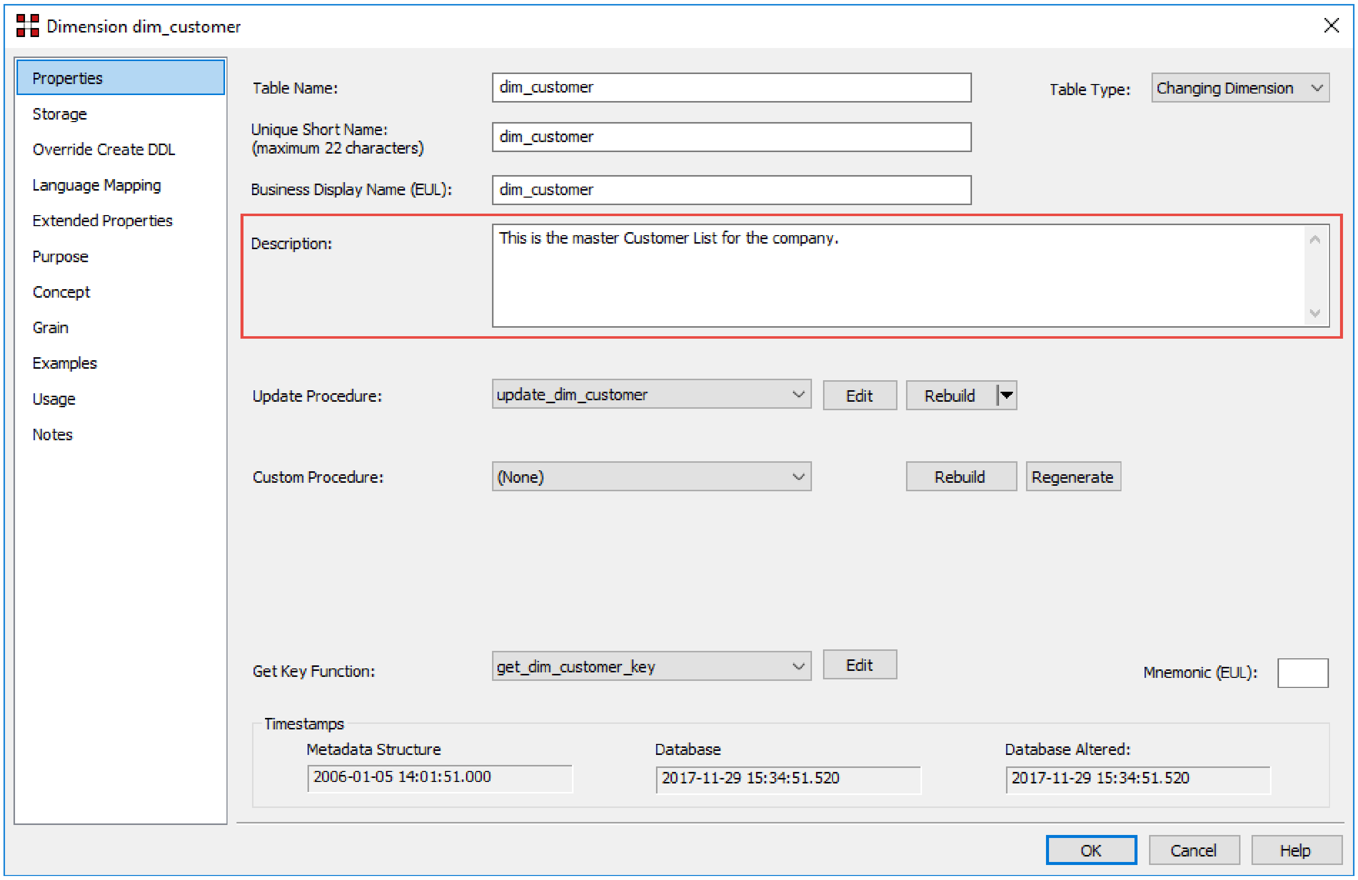
Notes:

- The table and column comments export/import feature only supports UTF-8 character encoded files.
- The exported data includes the Business Display Name (EUL table objects only) which can also be updated if required.

Defining Table Comments

Table Properties

The table comments are defined in the **Description** field of the table's Properties. The comments in this field, along with the **Table Name** and **Business Display Name (EUL)** fields are used when exporting/importing table comments.



Dimension dim_customer

- Properties
- Storage
- Override Create DDL
- Language Mapping
- Extended Properties
- Purpose
- Concept
- Grain
- Examples
- Usage
- Notes

Table Name: dim_customer

Table Type: Changing Dimension

Unique Short Name: (maximum 22 characters) dim_customer

Business Display Name (EUL): dim_customer

Description: This is the master Customer List for the company.

Update Procedure: update_dim_customer — Edit — Rebuild

Custom Procedure: (None) — Rebuild — Regenerate

Get Key Function: get_dim_customer_key — Edit

Mnemonic (EUL):

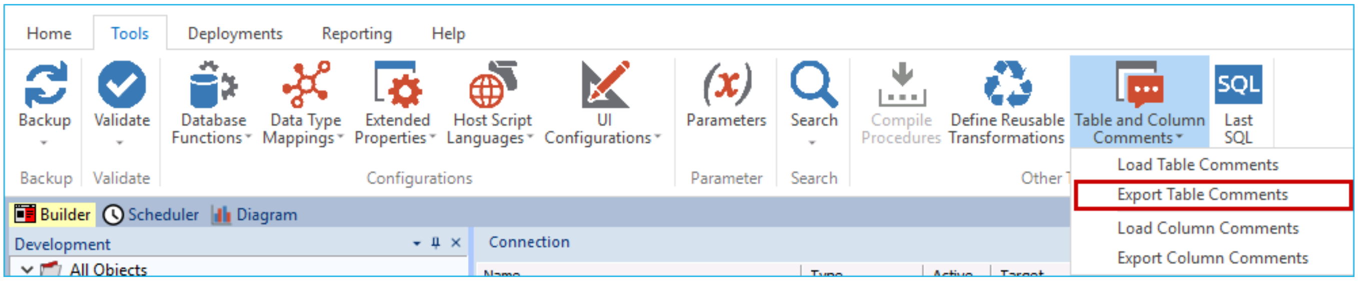
Timestamps

Metadata Structure	Database	Database Altered:
2006-01-05 14:01:51.000	2017-11-29 15:34:51.520	2017-11-29 15:34:51.520

OK — Cancel — Help

Exporting Table Comments

To export table comments from RED, select **Tools > Table and Column Comments > Export Table Comments**.



The **Table Comments Export** pop-up window is displayed:

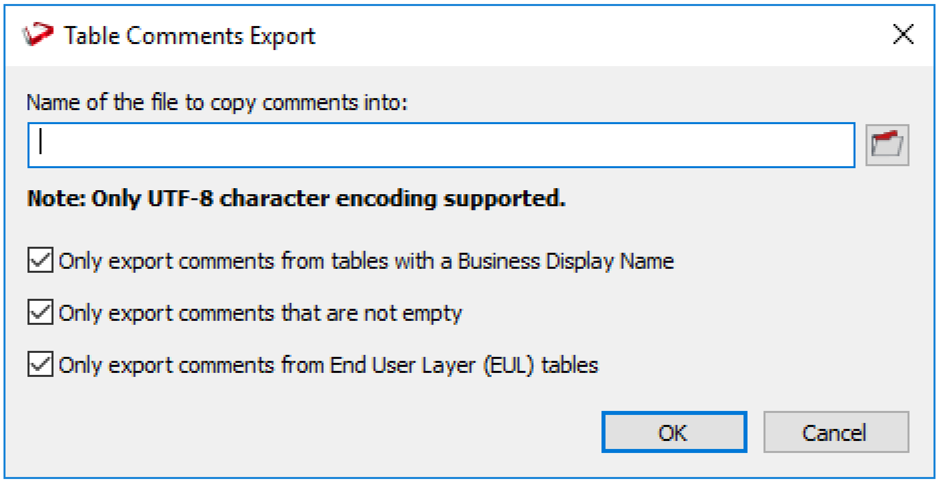


Table Comments Export

Name of the file to copy comments into:

Note: Only UTF-8 character encoding supported.

☑ Only export comments from tables with a Business Display Name

☑ Only export comments that are not empty

☑ Only export comments from End User Layer (EUL) tables

1. Type in the directory and file name of the export file or click the folder icon to navigate to the required directory.
2. Select or de-select the required export options.
   - **Only export comments from tables with a Business Display Name** - This option enables you to only process tables that have a **Business Display Name (EUL)** defined in the table's **Properties** screen. Clear the check box if you want to process all tables in your data warehouse.
   - **Only export comments that are not empty** - This option enables you to only process tables that have comments defined in the **Description** field of the table's **Properties** screen. Clear the check box if you want to process all tables in your data warehouse.
   - **Only export comments from End User Layer (EUL) tables** - This option enables you to only process tables that are visible to the end user. Clear the check box if you want to process all tables in your data warehouse. The visibility of table objects to end users in RED are set in **Tools > Options > Object Types > Object Type End User Setting**, refer to Object End User Setting for details.

   All the above options are selected by default.
3. Click **OK** to proceed with the export. The **Table Comments Export** pop-up window displays the progress of your export. You can click **Cancel** to abort the process.

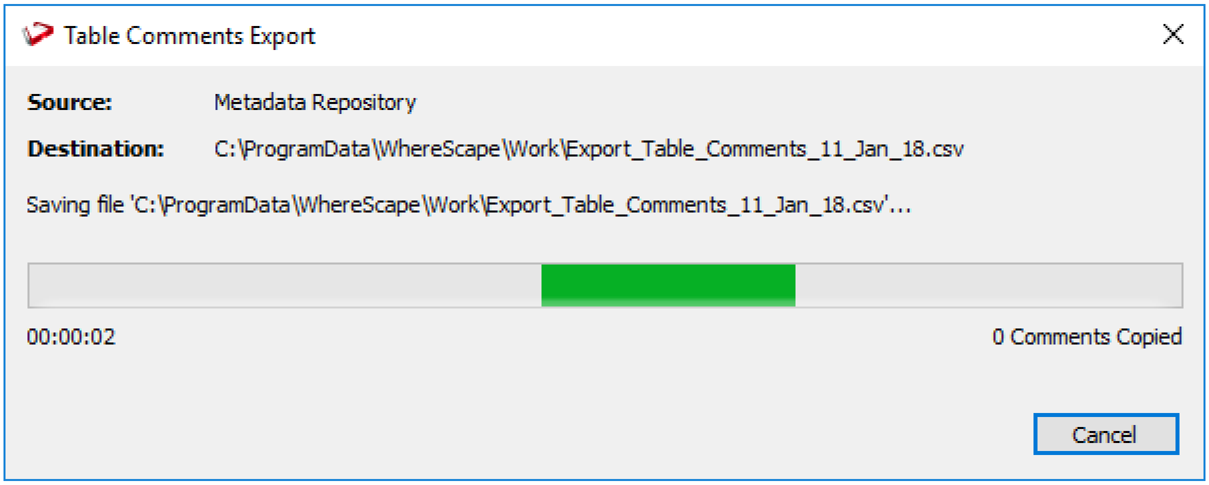


Table Comments Export

Source: Metadata Repository

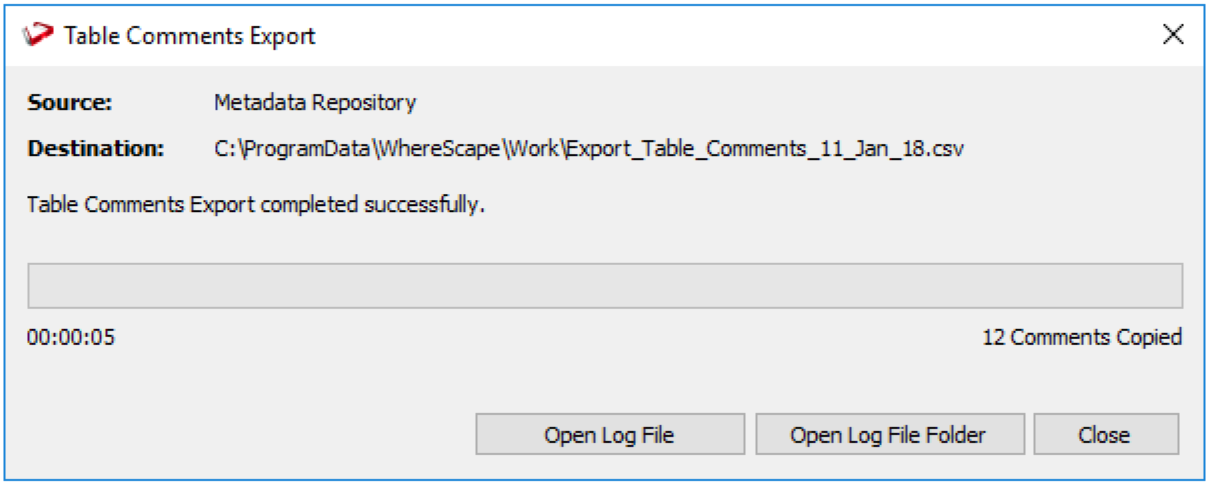
Destination: C:\ProgramData\WhereScape\Work\Export_Table_Comments_11_Jan_18.csv

Saving file 'C:\ProgramData\WhereScape\Work\Export_Table_Comments_11_Jan_18.csv'...

00:00:02

0 Comments Copied

4. Once the export process is completed, the pop-window displays the processing time and the number of comments copied to the export file.



Source: Metadata Repository

Destination: C:\ProgramData\WhereScape\Work\Export_Table_Comments_11_Jan_18.csv

Table Comments Export completed successfully.

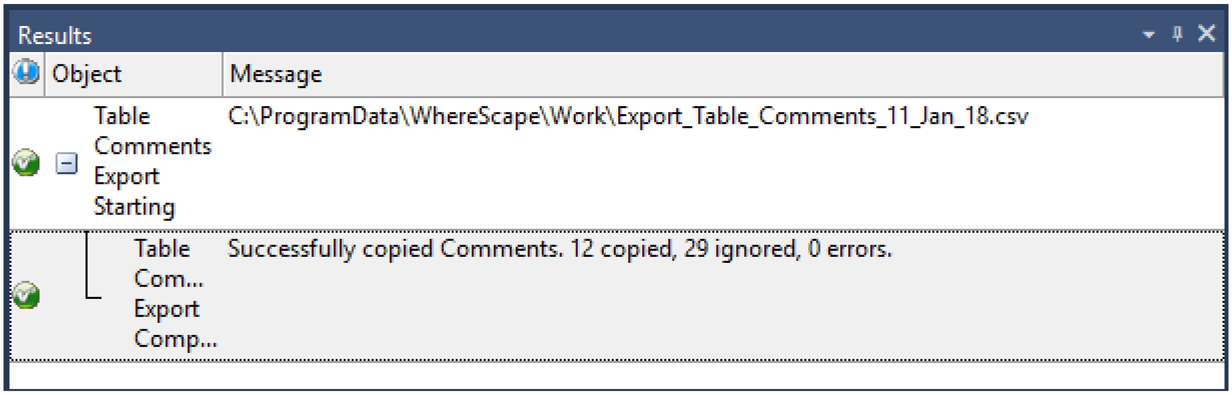
00:00:05

12 Comments Copied

Note:

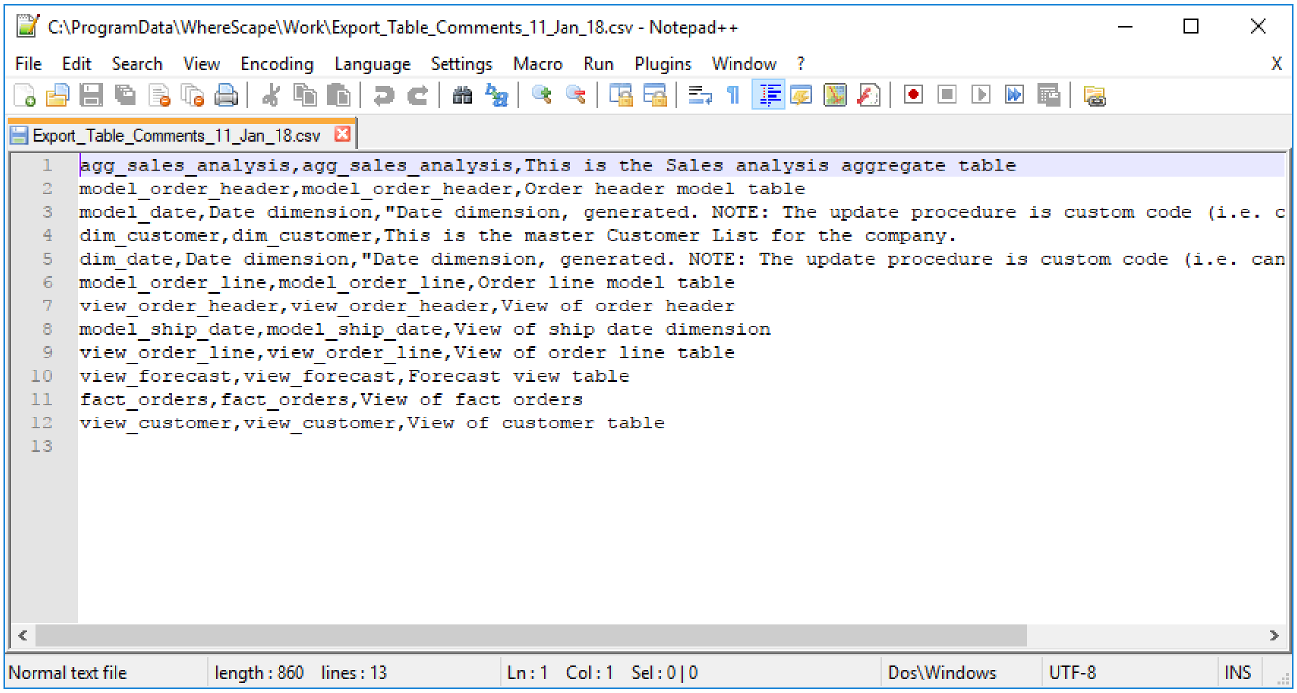
After completing the export process, you can click **Open Log File** or **Open Log File Folder** to view the log file generated. Refer to **Viewing the Import/Export Logs** for more details.

5. Click **Close** to exit the pop-up window. The **Results** pane displays a summary about the comments exported and confirms successful completion of the export.



Object	Message
Table Comments Export Starting	C:\ProgramData\WhereScape\Work\Export_Table_Comments_11_Jan_18.csv
Table Com... Export Comp...	Successfully copied Comments. 12 copied, 29 ignored, 0 errors.

6. Use Microsoft Excel 2013 or Notepad to view/edit the exported comments.


C:\ProgramData\WhereScape\Work\Export_Table_Comments_11_Jan_18.csv - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Export_Table_Comments_11_Jan_18.csv
Normal text file
length : 860 lines : 13
Ln : 1 Col : 1 Sel : 0 | 0
Dos\Windows
UTF-8
INS

```
agg_sales_analysis,agg_sales_analysis,This is the Sales analysis aggregate table
model_order_header,model_order_header,Order header model table
model_date,Date dimension,"Date dimension, generated. NOTE: The update procedure is custom code (i.e. c
dim_customer,dim_customer,This is the master Customer List for the company.
dim_date,Date dimension,"Date dimension, generated. NOTE: The update procedure is custom code (i.e. can
model_order_line,model_order_line,Order line model table
view_order_header,view_order_header,View of order header
model_ship_date,model_ship_date,View of ship date dimension
view_order_line,view_order_line,View of order line table
view_forecast,view_forecast,Forecast view table
fact_orders,fact_orders,View of fact orders
view_customer,view_customer,View of customer table
```

Each row of data includes the Table Name, Business Display Name (EUL table objects only) and Table Comments (Description).

Notes:

The table and column comments import/export only supports UTF-8 character encoded files.

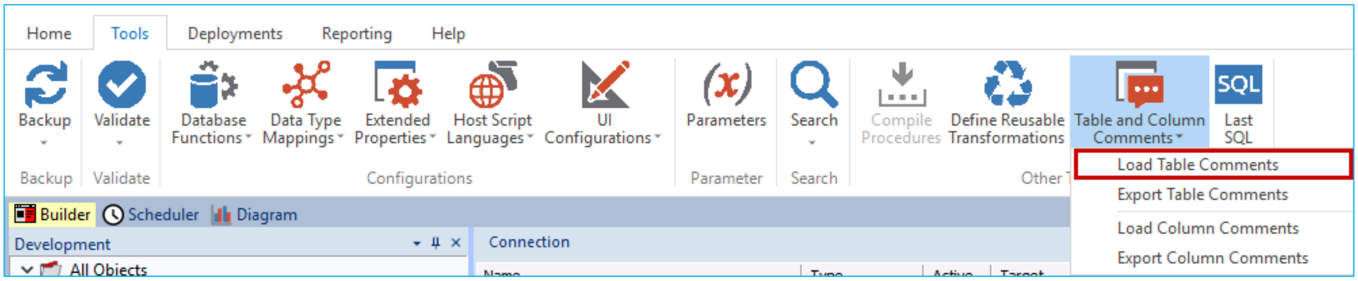
You also need to be aware of the following points, if you are going to use another tool other than Microsoft Excel 2013 to edit the comments:

- The comment portion can be surrounded in double quotes.
- Double quotes can be escaped within the comment by prefixing the quote with another quote.
- Comma characters can be used within the comment, but only if the comment is inside double quotes.
- Comments can span multiple lines, but only if the comment is inside double quotes.

Refer to **Importing Comments from an External Source** for details.

Loading Table Comments

To load/import table comments to RED from an external (.csv) file, select **Tools > Table and Column Comments > Load Table Comments**.



The **Table Comments Import** pop-up window is displayed:

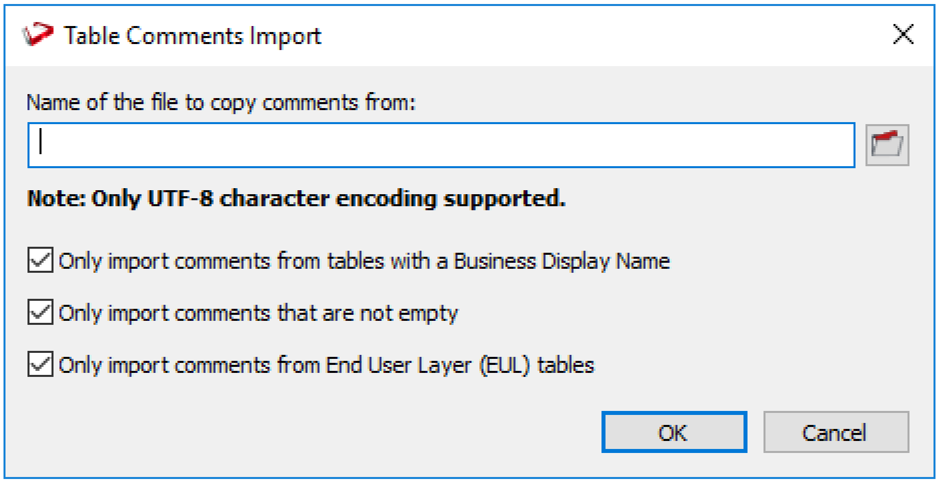


Table Comments Import

Name of the file to copy comments from:

Note: Only UTF-8 character encoding supported.

☑ Only import comments from tables with a Business Display Name

☑ Only import comments that are not empty

☑ Only import comments from End User Layer (EUL) tables

1. Type in the directory and file name of the source file or click the folder icon to navigate to the required directory.
2. Select or de-select the check box of the required import options.
   - **Only import comments from tables with a Business Display Name** - This option enables you to only process tables in the import file that have a **Business Display Name** defined. Clear the check box if you want to process all tables from your import file.
   - **Only import comments that are not empty** - This option enables you to only process tables that have comments defined in the import file. Clear the check box if you want to process all tables from your import file.
   - **Only import comments from End User Layer (EUL) tables** - This option enables you to only process tables in the import file that are visible to the end user. Clear the check box if you want to process all tables from your import file. The visibility of table objects to end users in RED are set in **Tools > Options > Object Types > Object Type End User Setting**, refer to **Object End User Setting** for details.

   All the above options are selected by default.
3. Click **OK** to proceed with the import. The **Table Comments Import** pop-up window displays the progress of your import. You can click **Cancel** to abort the process.

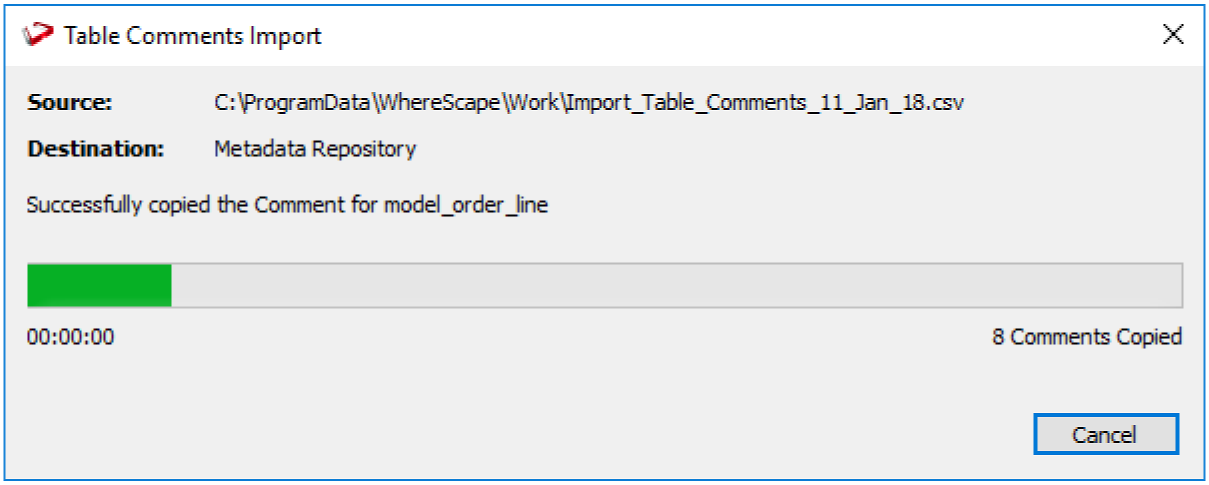


Table Comments Import

Source: C:\ProgramData\WhereScape\Work\Import_Table_Comments_11_Jan_18.csv

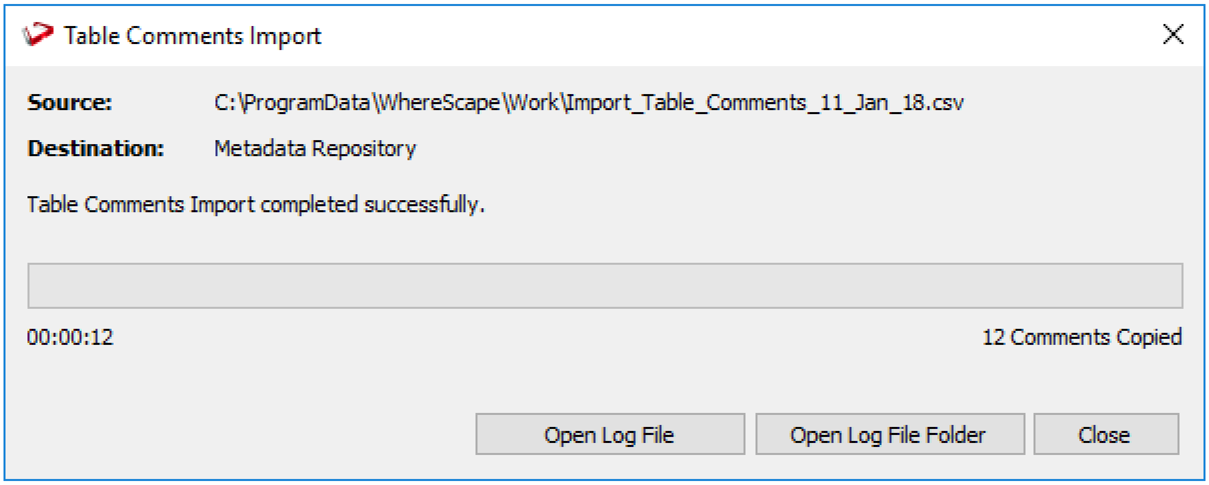
Destination: Metadata Repository

Successfully copied the Comment for model_order_line

00:00:00

8 Comments Copied

4. Once the import process is completed, the pop-window displays the processing time and the number of comments copied to the metadata repository.



Source: C:\ProgramData\WhereScape\Work\Import_Table_Comments_11_Jan_18.csv

Destination: Metadata Repository

Table Comments Import completed successfully.

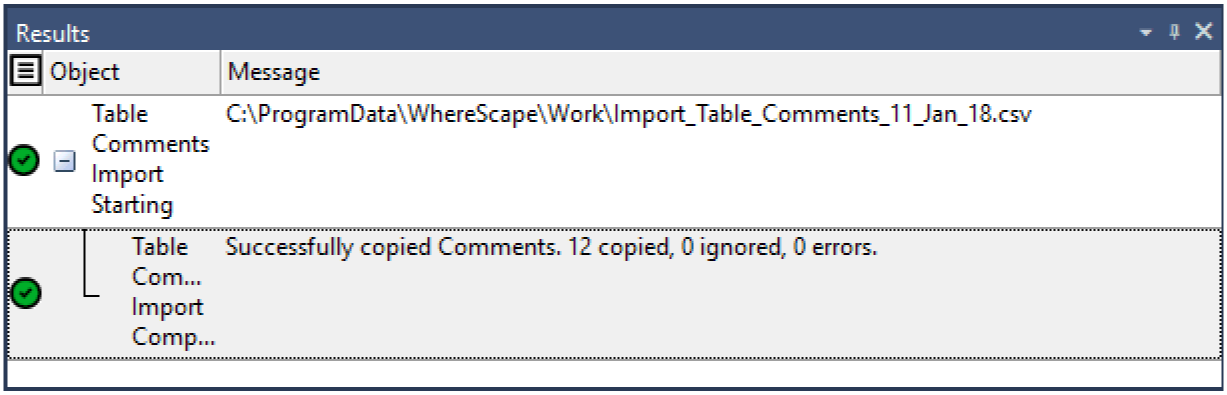
00:00:12

12 Comments Copied

Note:

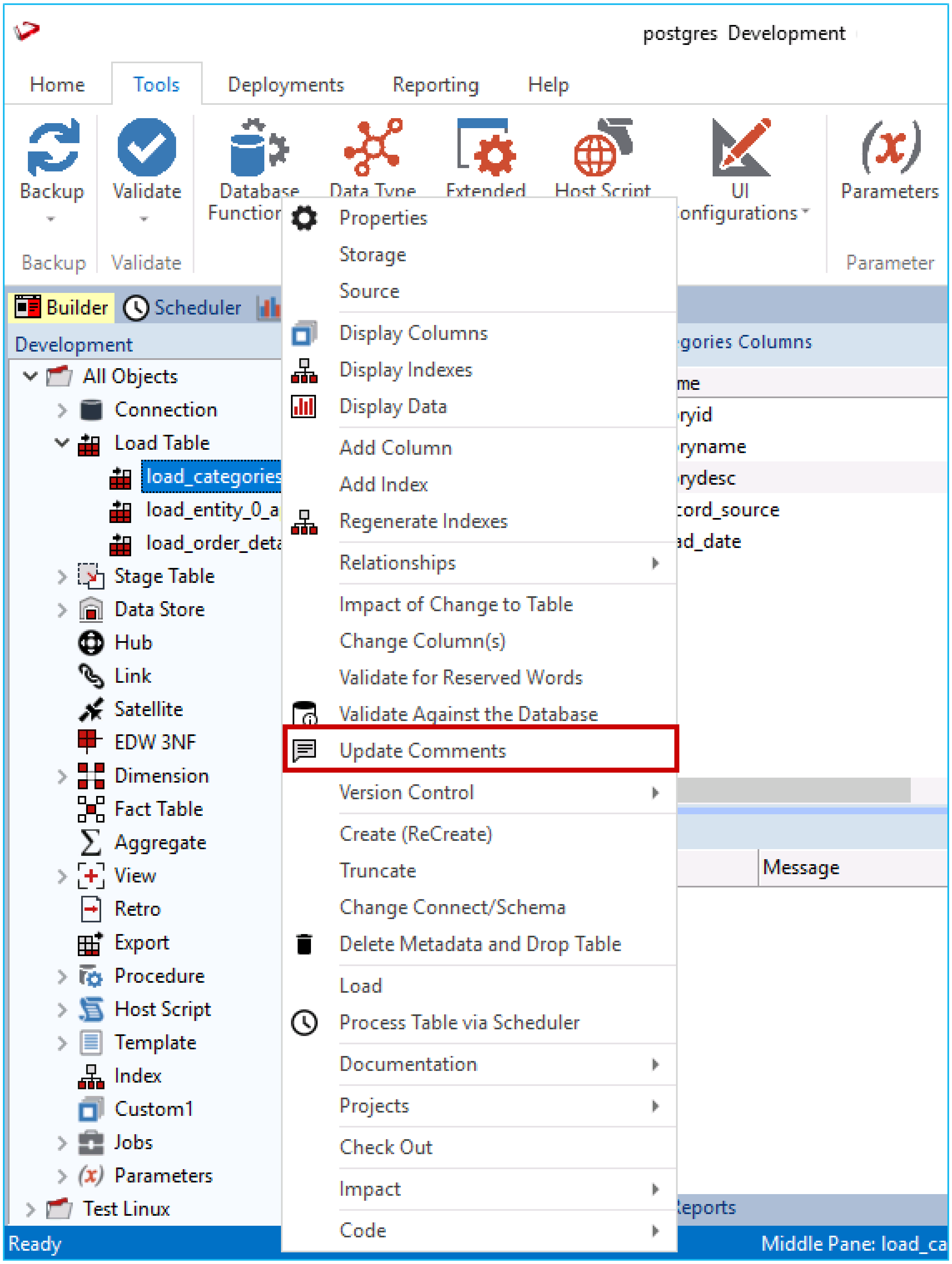
After completing the import process, you can click **Open Log File** or **Open Log File Folder** to view the log file generated. Refer to **Viewing the Import/Export** Logs for details.

5. Click **Close** to exit the pop-up window. The **Results** pane displays a summary about the comments imported and confirms successful completion of the import.

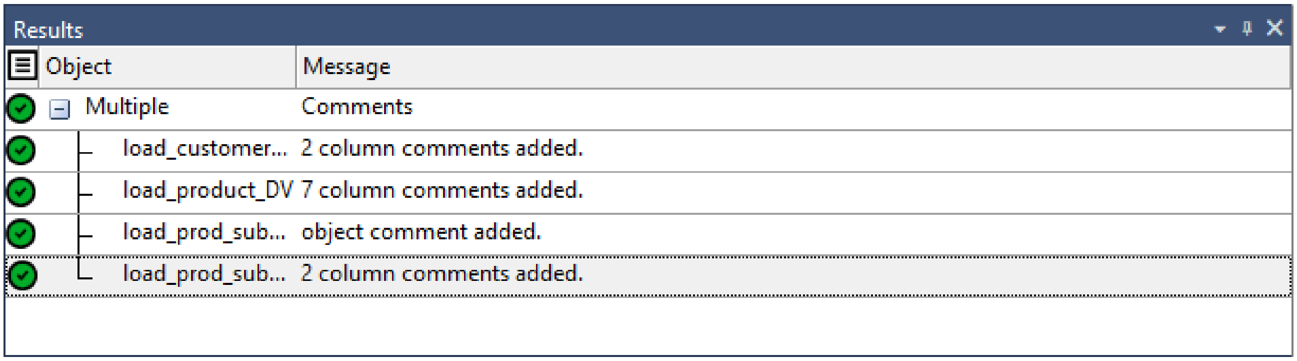


Object	Message
Table Comments Import Starting	C:\ProgramData\WhereScape\Work\Import_Table_Comments_11_Jan_18.csv
Table Com... Import Comp...	Successfully copied Comments. 12 copied, 0 ignored, 0 errors.

6. Use the **Update Comments** command from the context menu to copy the comments from the metadata repository to the corresponding tables in the target database.



The **Results** pane displays the table(s) and column(s) that have added comments.



Results

	Object	Message
✓	⊟ Multiple	Comments
✓	├ load_customer...	2 column comments added.
✓	├ load_product_DV	7 column comments added.
✓	├ load_prod_sub...	object comment added.
✓	└ load_prod_sub...	2 column comments added.

Notes:

The table and column comments import/export only supports UTF-8 character encoded files. You also need to be aware of the following points, if you are going to use another tool other than Microsoft Excel 2013 to edit the comments:

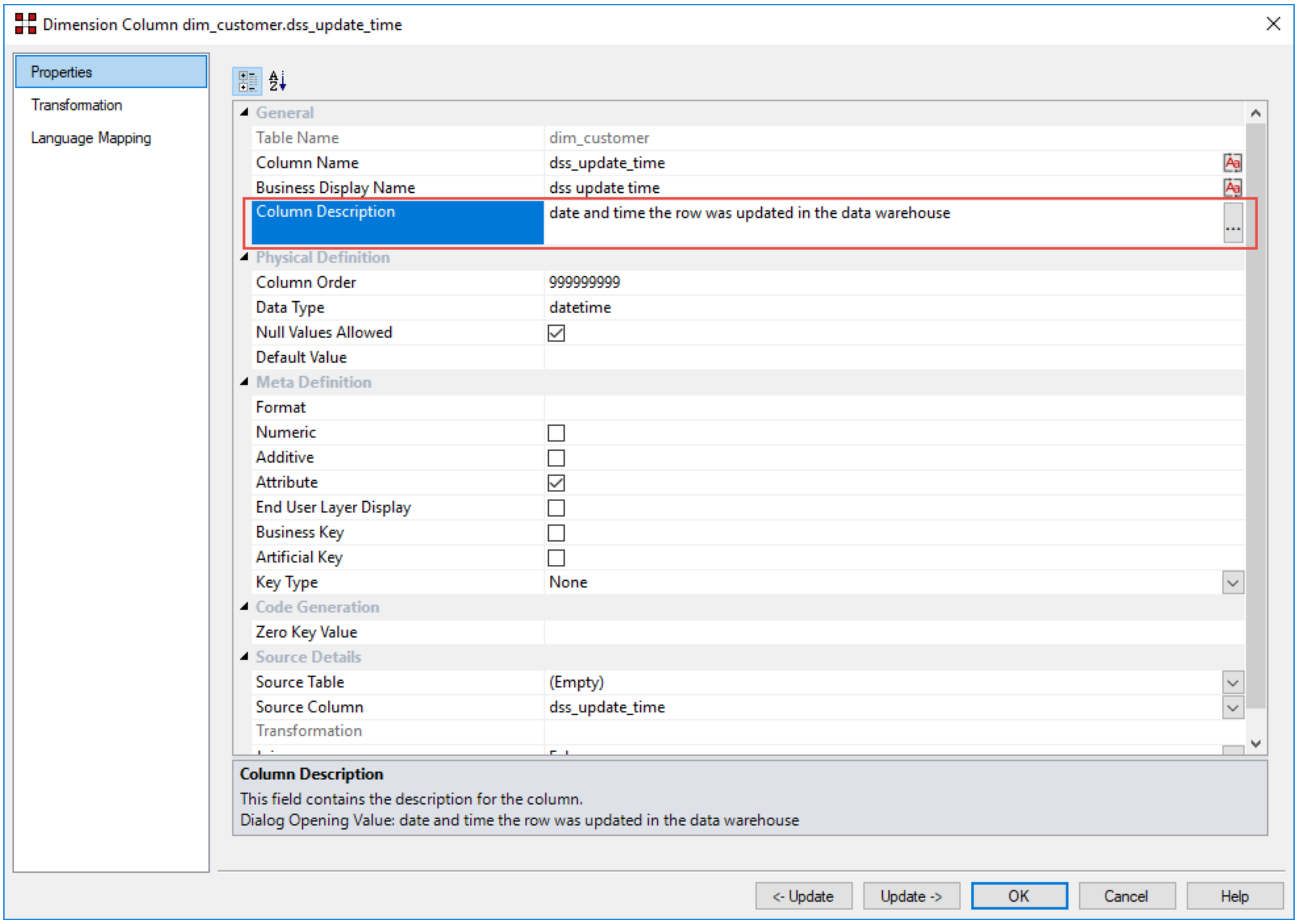
- The comment portion can be surrounded in double quotes.
- Double quotes can be escaped within the comment by prefixing the quote with another quote.
- Comma characters can be used within the comment, but only if the comment is inside double quotes.
- Comments can span multiple lines, but only if the comment is inside double quotes.

Refer to **Importing Comments from an External Source** for details.

Defining Column Comments

Column Properties

The column comments are defined in the **Column Description** field of the column's **Properties**. The comments in this field, along with the **Table Name**, **Column Name**, and **Column Title/Business Display Name** fields are used when exporting/importing column comments.



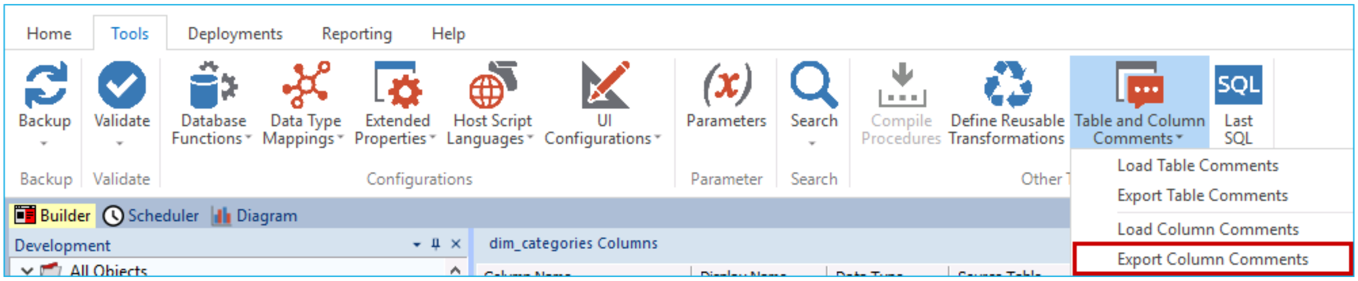
Dimension Column dim_customer.dss_update_time

General	
Table Name	dim_customer
Column Name	dss_update_time
Business Display Name	dss update time
Column Description	date and time the row was updated in the data warehouse
Physical Definition	
Column Order	999999999
Data Type	datetime
Null Values Allowed	☑
Default Value	
Meta Definition	
Format	
Numeric	□
Additive	□
Attribute	☑
End User Layer Display	□
Business Key	□
Artificial Key	□
Key Type	None
Code Generation	
Zero Key Value	
Source Details	
Source Table	(Empty)
Source Column	dss_update_time
Transformation	

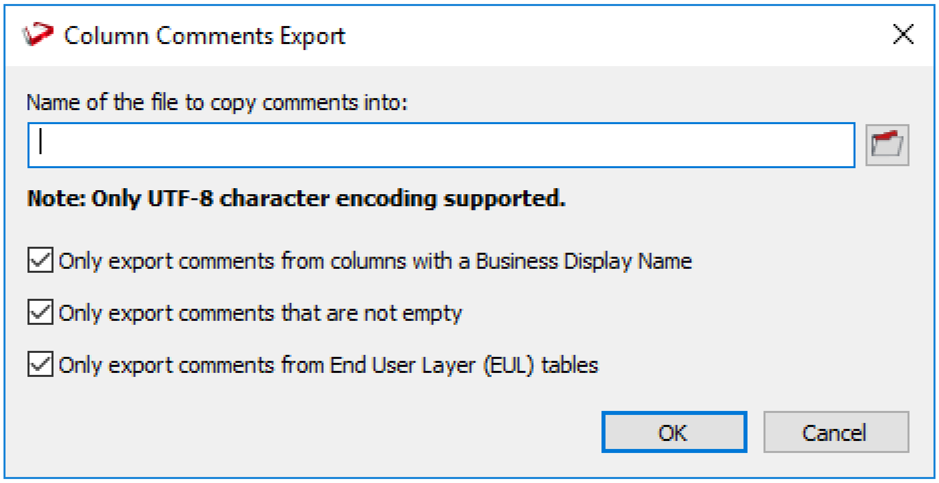
Column Description
This field contains the description for the column.
Dialog Opening Value: date and time the row was updated in the data warehouse

Exporting Column Comments

To export table comments from RED, select **Tools > Table and Column Comments > Export Column Comments**.



The **Column Comments Export** pop-up window is displayed:



Column Comments Export

Name of the file to copy comments into:

Note: Only UTF-8 character encoding supported.

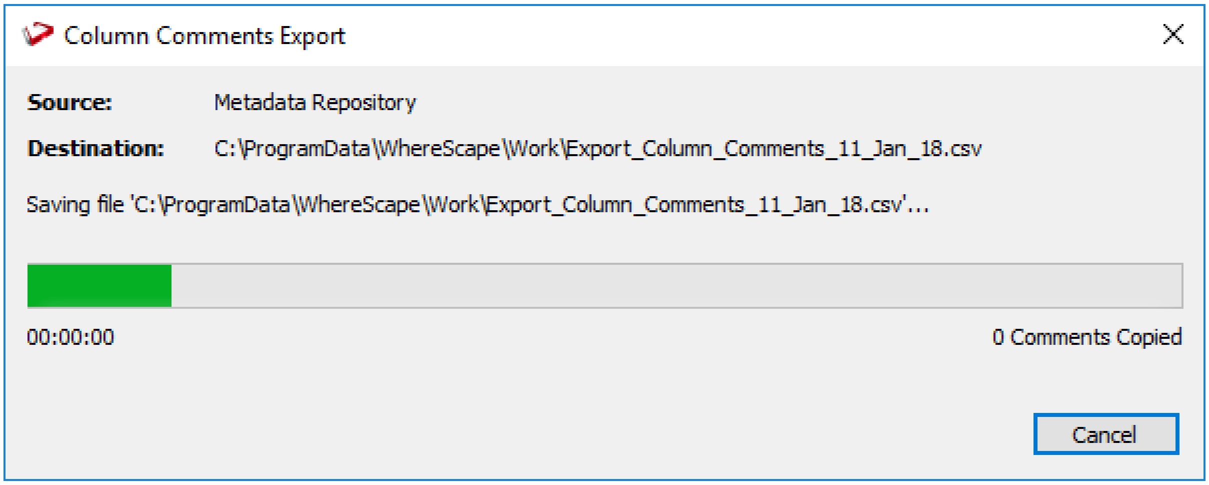
☑ Only export comments from columns with a Business Display Name

☑ Only export comments that are not empty

☑ Only export comments from End User Layer (EUL) tables

1. Type in the directory and file name of the export file or click the folder icon to navigate to the required directory.
2. Select or de-select the required export options.
   - **Only export comments from columns with a Business Display Name** - This option enables you to only process columns that have a **Business Display Name (Column Title)** defined in the column's **Properties** screen. Clear the check box if you want to process all columns in your data warehouse.
   - **Only export comments that are not empty** - This option enables you to only process columns that have comments defined in Column Description field of the column's **Properties** screen. Clear the check box if you want to process all columns in your data warehouse.
   - **Only export comments from End User Layer (EUL) tables** - This option enables you to only process columns from tables that are visible to the end user. Clear the check box if you want to process all columns in your data warehouse.
     The visibility of table objects to end users in RED are set in **Tools > Options > Object Types > Object Type End User Setting**, refer to Object End User Setting for details.

   All the above options are selected by default.
3. Click **OK** to proceed with the export. The **Column Comments Export** pop-up window displays the progress of your export. You can click **Cancel** to abort the process.



Column Comments Export

Source: Metadata Repository

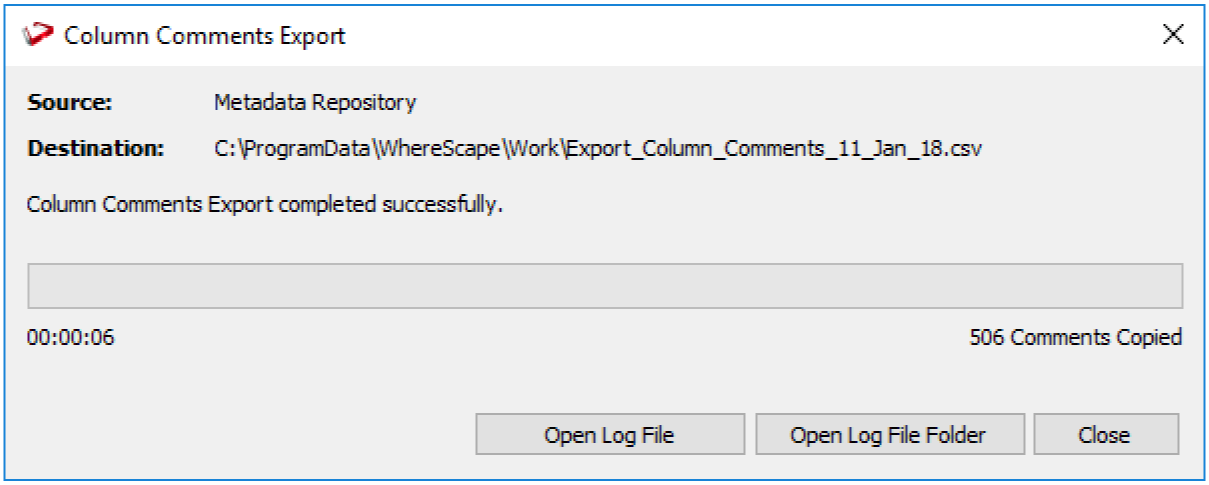
Destination: C:\ProgramData\WhereScape\Work\Export_Column_Comments_11_Jan_18.csv

Saving file 'C:\ProgramData\WhereScape\Work\Export_Column_Comments_11_Jan_18.csv'...

00:00:00

0 Comments Copied

4. Once the export process is completed, the pop-window displays the processing time and the number of comments copied to the export file.



Source: Metadata Repository

Destination: C:\ProgramData\WhereScape\Work\Export_Column_Comments_11_Jan_18.csv

Column Comments Export completed successfully.

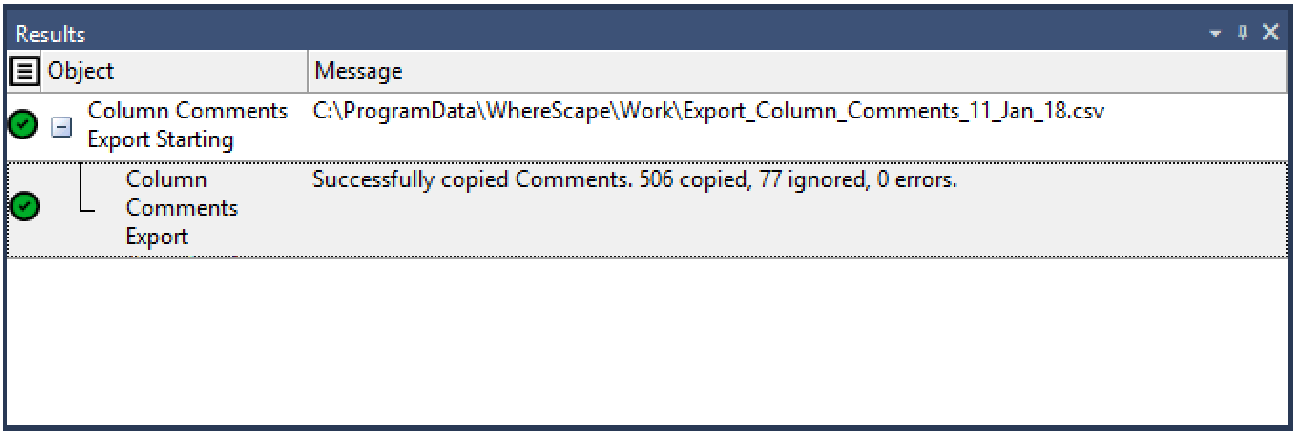
00:00:06

506 Comments Copied

Note

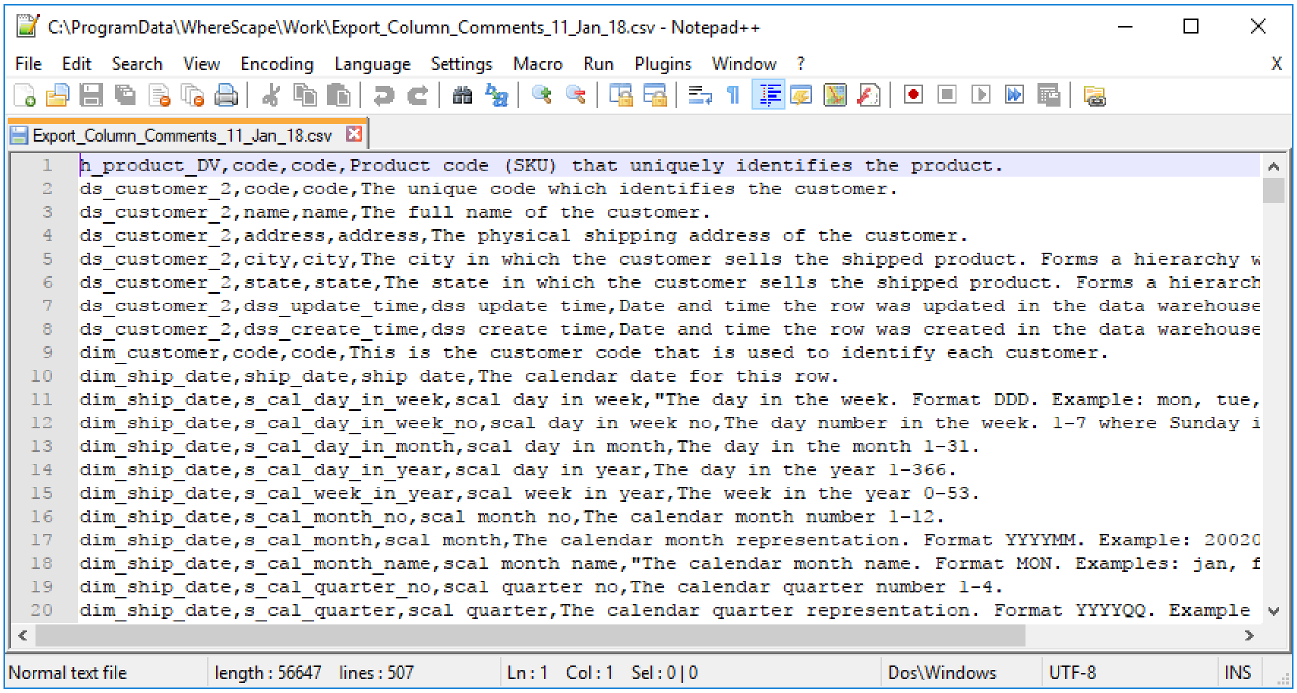
After completing the export process, you can click **Open Log File** or **Open Log File Folder** to view the log file generated. Refer to **Viewing the Import/Export Logs** for more details.

5. Click **Close** to exit the pop-up window. The **Results** pane displays a summary about the comments exported and confirms successful completion of the export.



Object	Message
Column Comments Export Starting	C:\ProgramData\WhereScape\Work\Export_Column_Comments_11_Jan_18.csv
Column Comments Export	Successfully copied Comments. 506 copied, 77 ignored, 0 errors.

6. Use Microsoft Excel 2013 or Notepad to view/edit the exported comments.


C:\ProgramData\WhereScape\Work\Export_Column_Comments_11_Jan_18.csv - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Export_Column_Comments_11_Jan_18.csv
Normal text file
length : 56647 lines : 507
Ln : 1 Col : 1 Sel : 0 | 0
Dos\Windows
UTF-8
INS

```
h_product_DV,code,code,Product code (SKU) that uniquely identifies the product.
ds_customer_2,code,code,The unique code which identifies the customer.
ds_customer_2,name,name,The full name of the customer.
ds_customer_2,address,address,The physical shipping address of the customer.
ds_customer_2,city,city,The city in which the customer sells the shipped product. Forms a hierarchy w
ds_customer_2,state,state,The state in which the customer sells the shipped product. Forms a hierarch
ds_customer_2,dss_update_time,dss update time,Date and time the row was updated in the data warehouse
ds_customer_2,dss_create_time,dss create time,Date and time the row was created in the data warehouse
dim_customer,code,code,This is the customer code that is used to identify each customer.
dim_ship_date,ship_date,ship date,The calendar date for this row.
dim_ship_date,s_cal_day_in_week,scal day in week,"The day in the week. Format DDD. Example: mon, tue,
dim_ship_date,s_cal_day_in_week_no,scal day in week no,The day number in the week. 1-7 where Sunday i
dim_ship_date,s_cal_day_in_month,scal day in month,The day in the month 1-31.
dim_ship_date,s_cal_day_in_year,scal day in year,The day in the year 1-366.
dim_ship_date,s_cal_week_in_year,scal week in year,The week in the year 0-53.
dim_ship_date,s_cal_month_no,scal month no,The calendar month number 1-12.
dim_ship_date,s_cal_month,scal month,The calendar month representation. Format YYYYMM. Example: 20020
dim_ship_date,s_cal_month_name,scal month name,"The calendar month name. Format MON. Examples: jan, f
dim_ship_date,s_cal_quarter_no,scal quarter no,The calendar quarter number 1-4.
dim_ship_date,s_cal_quarter,scal quarter,The calendar quarter representation. Format YYYYQQ. Example
```

Each row of data includes the Table Name, Column Name, Column Title/Business Display Name and Column Comments (Description).

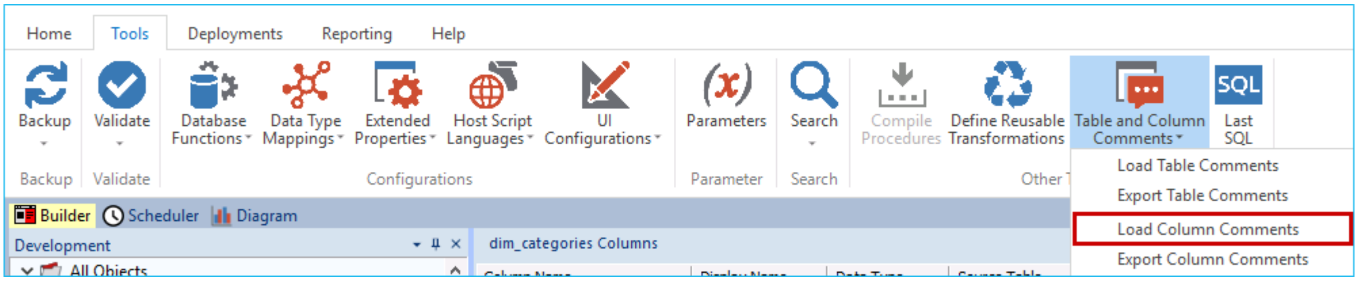
Notes

The table and column comments import/export only supports UTF-8 character encoded files.
You also need to be aware of the following points, if you are going to use another tool other than Microsoft Excel 2013 to edit the comments:

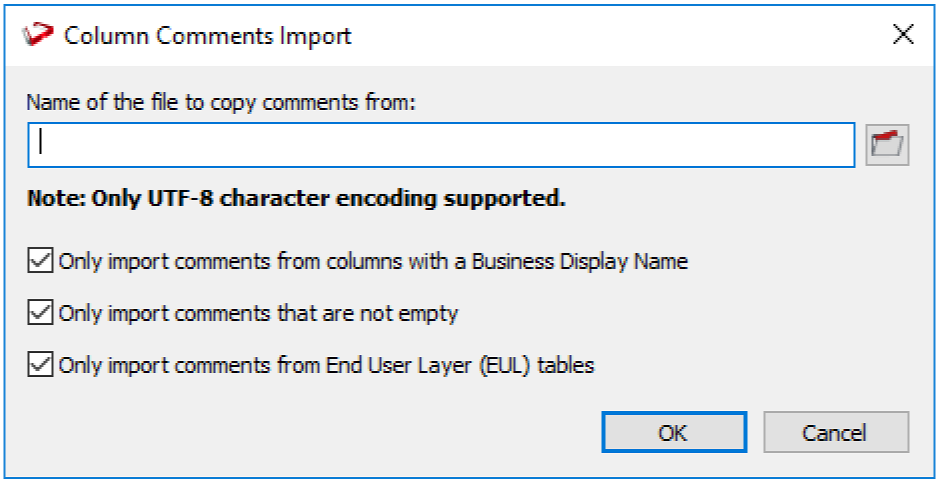
- The comment portion can be surrounded in double quotes.
- Double quotes can be escaped within the comment by prefixing the quote with another quote.
- Comma characters can be used within the comment, but only if the comment is inside double quotes.
- Comments can span multiple lines, but only if the comment is inside double quotes.
- Refer to **Importing Comments from an External Source** for details.

Loading Column Comments

To load/import column comments to RED from an external (.csv) file, select **Tools > Table and Column Comments > Load Column Comments**.



The **Column Comments Import** pop-up window is displayed:



Name of the file to copy comments from:

Note: Only UTF-8 character encoding supported.

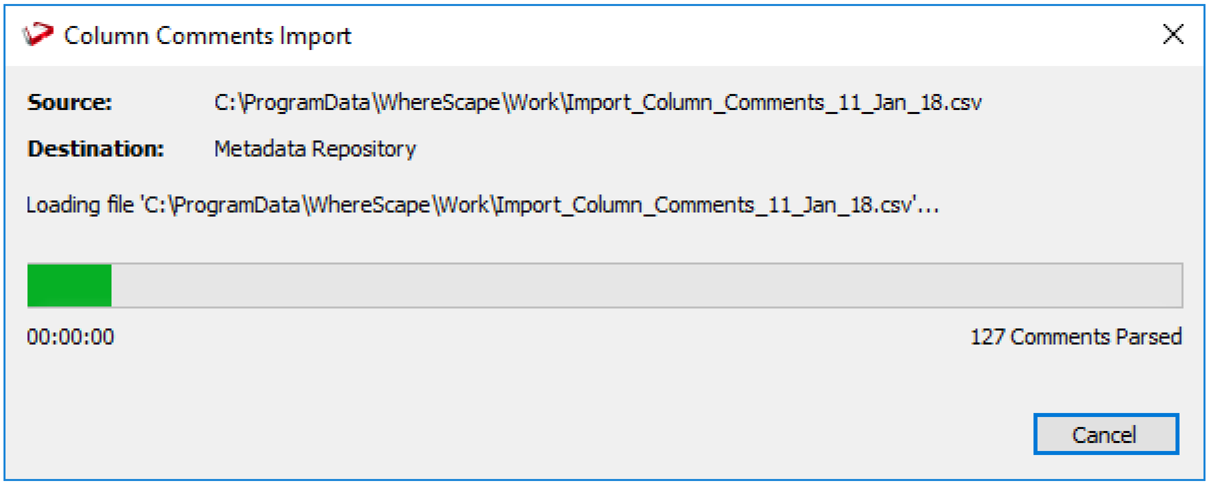
☑ Only import comments from columns with a Business Display Name

☑ Only import comments that are not empty

☑ Only import comments from End User Layer (EUL) tables

1. Type in the directory and file name of the source file or click the folder icon to navigate to the required directory.
2. Select or de-select the check box of the required import options.
   - **Only import comments from tables with a Business Display Name -** This option enables you to only process columns in the import file that have a Business Display Name/Column Title defined. Clear the check box if you want to process all tables from your import file.
   - **Only import comments that are not empty -** This option enables you to only process columns that have comments defined in the import file. Clear the check box if you want to process all tables from your import file.
   - **Only import comments from End User Layer (EUL) tables -** This option enables you to only process columns from tables in the import file that are visible to the end user. Clear the check box if you want to process all tables from your import file. The visibility of table objects to end users in RED are set in **Tools > Options > Object Types > Object Type End User Setting**, refer to Object End User Setting for details.

   All the above options are selected by default.
3. Click **OK** to proceed with the import. The **Column Comments Import** pop-up window displays the progress of your import. You can click **Cancel** to abort the process.



Source: C:\ProgramData\WhereScape\Work\Import_Column_Comments_11_Jan_18.csv

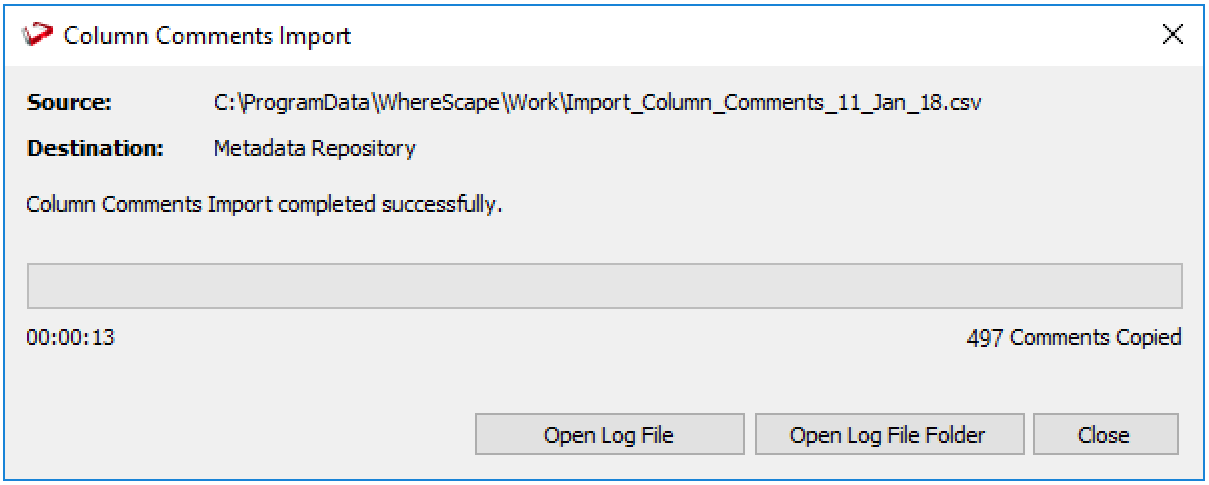
Destination: Metadata Repository

Loading file 'C:\ProgramData\WhereScape\Work\Import_Column_Comments_11_Jan_18.csv'...

00:00:00

127 Comments Parsed

4. Once the import process is completed, the pop-window displays the processing time and the number of comments copied to the metadata repository.



Source: C:\ProgramData\WhereScape\Work\Import_Column_Comments_11_Jan_18.csv

Destination: Metadata Repository

Column Comments Import completed successfully.

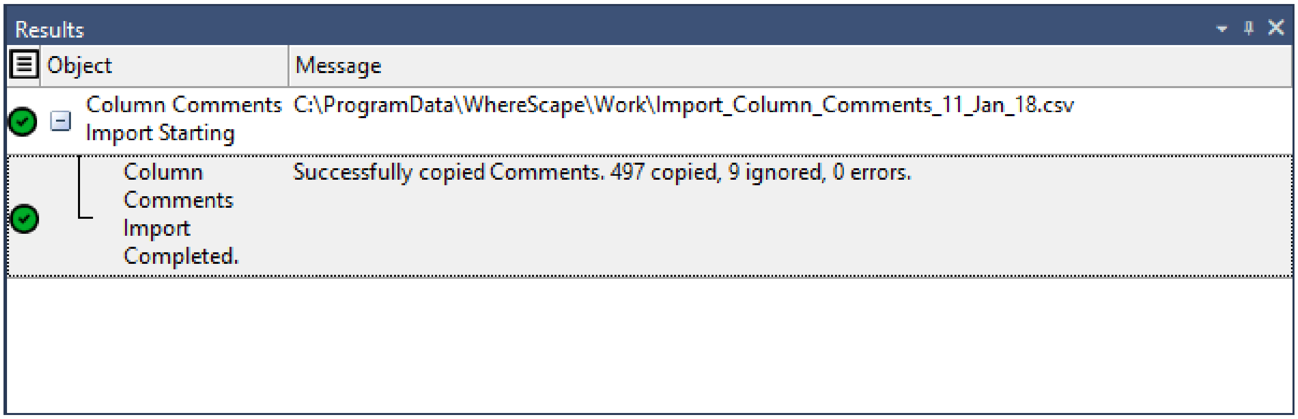
00:00:13

497 Comments Copied

Note

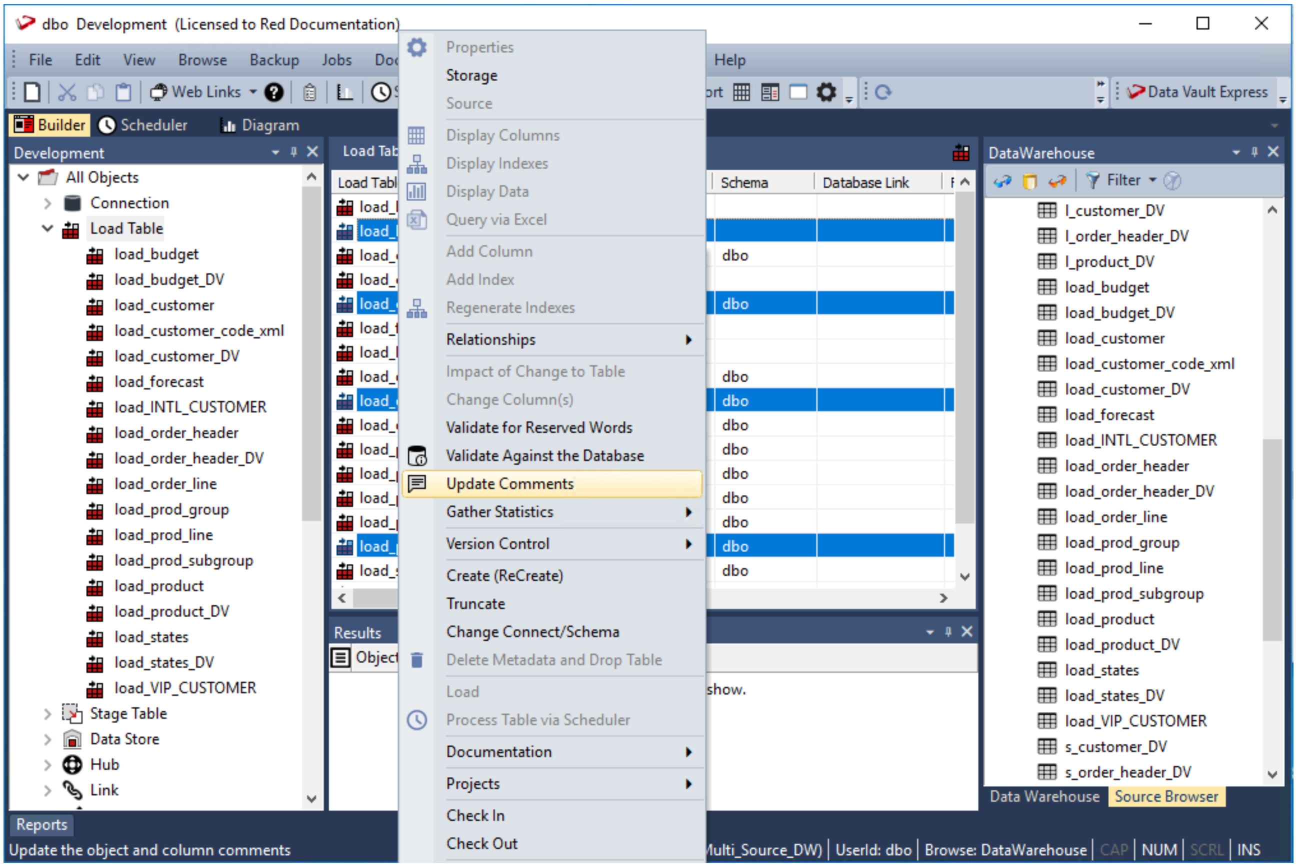
After completing the import process, you can click **Open Log File** or **Open Log File Folder** to view the log file generated. Refer to **Viewing the Import/Export Logs** for details.

5. Click **Close** to exit the pop-up window. The **Results** pane displays a summary about the comments imported and confirms successful completion of the import.

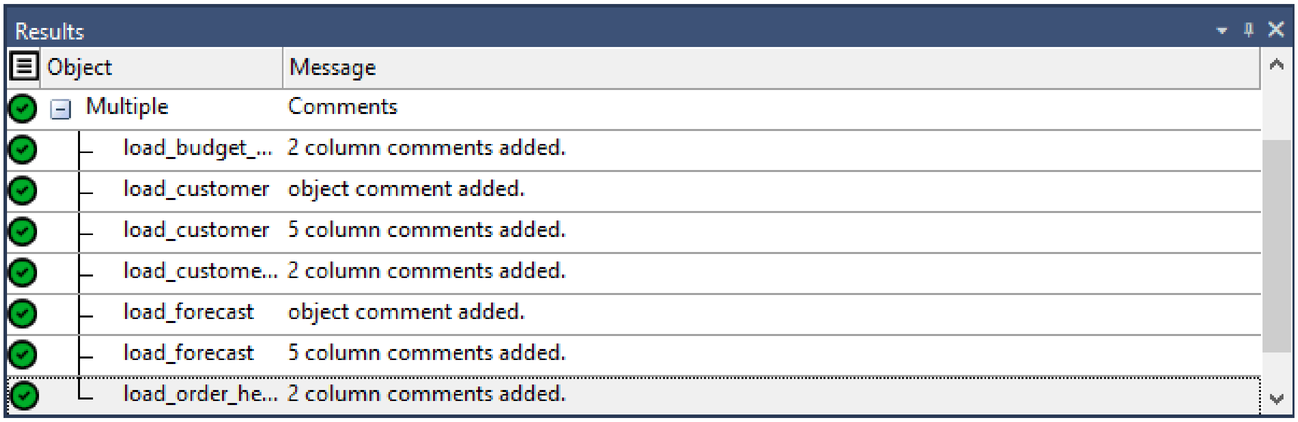


Object	Message
Column Comments Import Starting	C:\ProgramData\WhereScape\Work\Import_Column_Comments_11_Jan_18.csv
Column Comments Import Completed.	Successfully copied Comments. 497 copied, 9 ignored, 0 errors.

6. Use the **Update Comments** command from the context menu, to copy the comments from the metadata repository to the corresponding columns in the target database.



The **Results** pane displays the table(s) and column(s) that have added comments.



Object	Message
Multiple	Comments
load_budget_...	2 column comments added.
load_customer	object comment added.
load_customer	5 column comments added.
load_custome...	2 column comments added.
load_forecast	object comment added.
load_forecast	5 column comments added.
load_order_he...	2 column comments added.

Notes

The table and column comments import/export only supports UTF-8 character encoded files. You also need to be aware of the following points, if you are going to use another tool other than Microsoft Excel 2013 to edit the comments:

- The comment portion can be surrounded in double quotes.
- Double quotes can be escaped within the comment by prefixing the quote with another quote.
- Comma characters can be used within the comment, but only if the comment is inside double quotes.
- Comments can span multiple lines, but only if the comment is inside double quotes.
- Refer to **Importing Comments from an External Source** for details.

Importing Comments from an External Source

The import function supports Microsoft Excel 2013 comma separated value (.csv) file format for importing comments to RED.

Table Comments

Each row of data includes the Table Name, Business Display Name (EUL table objects only) and Table Comments (Description).

```
view_order_header,view_order_header,View of order header.
view_order_line,view_order_line,View of order line table.
view_forecast,view_forecast,View of Forecast table.
```

Column Comments

Each row of data includes the Table Name, Column Name, Column Title/Business Display Name and Column Comments (Description).

```
view_order_header,order_number,order_number,A code which identifies the order.
view_order_line,order_line_no,order_line_no,A number which identifies a line on the order.
view_forecast,forecast_quantity,forecast_quantity,Quantity of product forecasted (i.e. number
of product units forecasted to be sold).
```

Notes

- The comment portion can be surrounded in double quotes. For example:

  ```
  view_order_header,view_order_header,"View of order header."
  view_order_line,view_order_line,"View of order line table."
  view_forecast,view_forecast,"View of Forecast table."
  ```

- Double quotes can be escaped within the comment by prefixing the quote with another quote. For example:

  ```
  view_order_header,view_order_header,"View ""of"" order header."
  view_order_line,view_order_line,View of order line table.
  view_forecast,view_forecast,View of Forecast table.
  ```

- Comma characters can be used within the comment, but only if the comment is inside double quotes. For example:

  ```
  view_order_header,view_order_header,"View of order header,comment."
  view_order_line,view_order_line,"View of order line table,comment."
  view_forecast,view_forecast,"View of Forecast table,comment."
  ```

- Comments can span multiple lines, but only if the comment is inside double quotes. For example:

  ```
  view_order_header,view_order_header,"View of order header comment."
  view_order_line,view_order_line,"View of order line
  table comment."
  view_forecast,view_forecast,"View of Forecast
  table comment."
  ```

Viewing the Import/Export Logs

RED creates a log to aid in troubleshooting, in case issues are encountered during the import/export process.

After completing the import/export process, you can click **Open Log File** or **Open Log File Folder** from the **Table Comments Import/Export** pop-up window to view the log file generated.

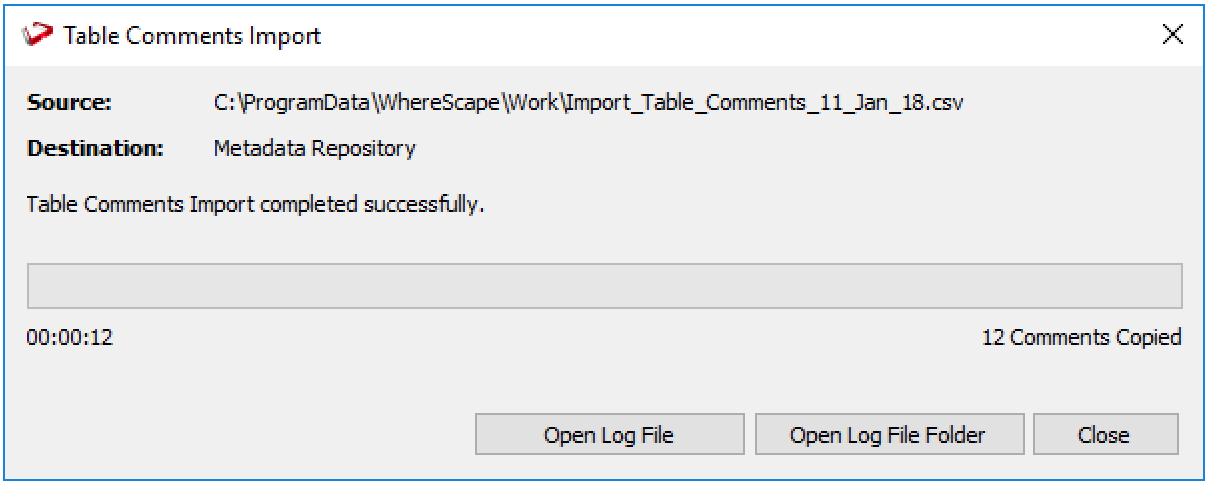


Table Comments Import

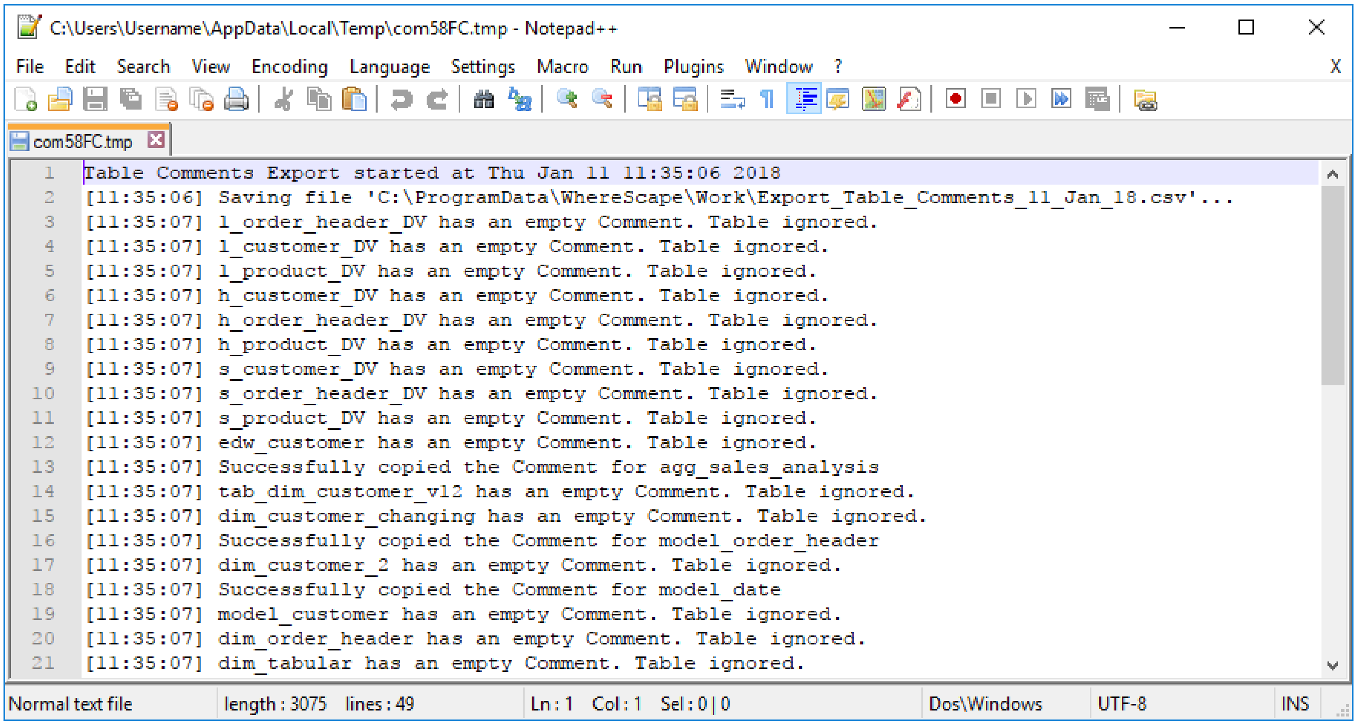
Source: C:\ProgramData\WhereScape\Work\Import_Table_Comments_11_Jan_18.csv

Destination: Metadata Repository

Table Comments Import completed successfully.

00:00:12 12 Comments Copied

The import/export logs are stored in your Windows Temp directory (`c:\Users\Username\AppData\Local\Temp`).



C:\Users\Username\AppData\Local\Temp\com58FC.tmp - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
com58FC.tmp

```
Table Comments Export started at Thu Jan 11 11:35:06 2018
[11:35:06] Saving file 'C:\ProgramData\WhereScape\Work\Export_Table_Comments_11_Jan_18.csv'...
[11:35:07] l_order_header_DV has an empty Comment. Table ignored.
[11:35:07] l_customer_DV has an empty Comment. Table ignored.
[11:35:07] l_product_DV has an empty Comment. Table ignored.
[11:35:07] h_customer_DV has an empty Comment. Table ignored.
[11:35:07] h_order_header_DV has an empty Comment. Table ignored.
[11:35:07] h_product_DV has an empty Comment. Table ignored.
[11:35:07] s_customer_DV has an empty Comment. Table ignored.
[11:35:07] s_order_header_DV has an empty Comment. Table ignored.
[11:35:07] s_product_DV has an empty Comment. Table ignored.
[11:35:07] edw_customer has an empty Comment. Table ignored.
[11:35:07] Successfully copied the Comment for agg_sales_analysis
[11:35:07] tab_dim_customer_v12 has an empty Comment. Table ignored.
[11:35:07] dim_customer_changing has an empty Comment. Table ignored.
[11:35:07] Successfully copied the Comment for model_order_header
[11:35:07] dim_customer_2 has an empty Comment. Table ignored.
[11:35:07] Successfully copied the Comment for model_date
[11:35:07] model_customer has an empty Comment. Table ignored.
[11:35:07] dim_order_header has an empty Comment. Table ignored.
[11:35:07] dim_tabular has an empty Comment. Table ignored.
```

Normal text file | length : 3075 lines : 49 | Ln : 1 Col : 1 Sel : 0 | 0 | Dos\Windows | UTF-8 | INS

UI Configurations

UI Configurations allow the user to define the fields available on certain configurable UI's. RED stores any data captured in these fields and treats it like any other object metadata, making it available to all automation aspects of the tool.

Configurable UI's

- ### Connection Properties

  Allows configuration of fields on the Connection Properties page, set and used in creating an **Extensible Source Connection**.

- ### Load Table Properties

  Allows configuration of fields on the Source page of Load Table Properties, used as the UI Configuration when sourcing from an **Extensible Source Connection** where this configuration is set.

- ### Column Properties

  Allows configuration of fields on the Column Properties page, set on a **Target Connection** and applied for any table residing on that Connection's Targets.

Configurations Maintenance

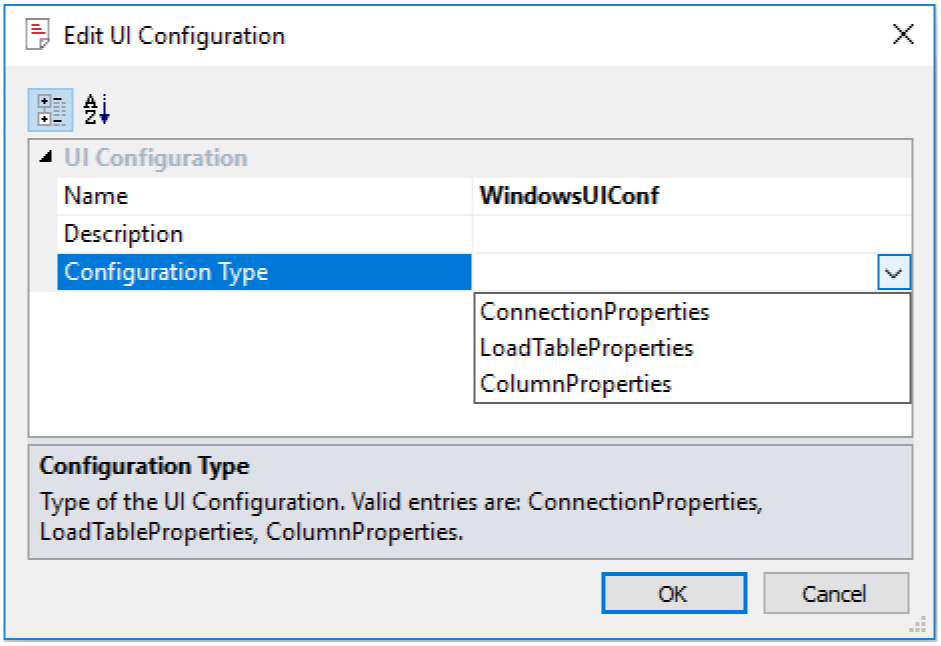
The UI Configurations actions are located under the Tools menu. The UI Configurations menu action items are:

- Maintain UI Configurations
- Load UI Configuration: Select this option to load a UI configuration from file.
- Export UI Configuration: Select this option to export a UI configuration to file.

To mantain UI Configurations select **Tools > UI Configurations> UI Configurations Maintenance.**

- Click **New...** to create a new UI Configuration.

  - In the Edit UI Configuration window, enter a **Name** for the UI Configuration.
  - Add a **Description** of UI configuration.
  - Select the **Configuration Type** among **ConnectionProperties**, **LoadTableProperties**, and **ColumnProperties**.
  - Click **OK** to proceed to the **Field Configuration JSON** where a sample field configuration is created to get you started.



Edit UI Configuration

UI Configuration

Name	WindowsUIConf
Description	
Configuration Type	

Configuration Type

Type of the UI Configuration. Valid entries are: ConnectionProperties, LoadTableProperties, ColumnProperties.

- Click **Copy...** to copy an existing UI Configuration. Edit the name and it creates a new copied configuration under a new name.
- **Edit..,** opens the configuration in the editor to view or edit the configured fields.
- Click **Rename** to update the Configuration name.
- Click **Delete** to remove any listed UI Configuration, deletion is prevented for configurations in use.

Referencing a configured field’s metadata

There are three ways to reference the metadata set in configured fields:

Token replacement in scripts

In Scripts of Objects which have an association to an Extensible Source Connection you can reference any configured field from the connection in the script using the following token format which will be replaced at run-time:

```
$WSL_SRCCFG_<cofigured_field_name>$
```

Environment variables in scripts

In Scripts of Objects which have an association to an Extensible Source Connection, when a configured field has the attribute “setEnvVarForScripts” = TRUE then the environment variable RED exposes at run-time takes the following form:

```
WSL_SRCCFG_<cofigured_field_name>
```

Templated code generation

The following pebble syntax can be used in templates to reference the metadata stored in configured fields, replace <field_name> with the field name in your configuration. The full pebble examples can be generated through the Template Editor using the ‘Tools’ menu ‘Create API Example Template’. Configured fields are available as a key:value array.

- For Extensible Source Connections:

  ```
  table.loadInfo.sourceConnection.uiConfigFields.<field_name>
  ```

- For Load Tables:

`table.loadInfo.uiConfigLoadProperties.<field_name>`

- For Columns:

  `column.uiConfigColumnProperties.<field_name>`

Field Configuration JSON

The UI Configurations allow you to add user-defined UI component including List boxes, Text boxes, and Numeric fields to certain properties pages in the UI. The table below lists the required common and specific attributes of each UI component. Example configurations follow the table:

Options	Applicable Configs	Description
description	ALL	Add a description for the field which will be displayed on mouse-hover and in the bottom description pane of the connection.
displayName	ALL	Add the name that displays in the configuration field.
documentation	ALL	If the connection is part of documentation and the value is set to "true", therefore it adds this attribute to the generated documentation.
encryption	CONNECTIONS	Optional encryption settings when persisting the field value to the database: (add bullet points for these) 'none' : the value will not be encrypted. 'wherescape' : the value will be encrypted by RED and decrypted as required. 'external' : equivalent to 'none' but allows you to flag that values have been encrypted externally. This option is required, if you do not add a value in this field, you get an error message.
group	ALL	Add a group name for the configuration attributes.
masked	CONNECTIONS, COLUMNS	Set the value to "true" to mask the value.
order	ALL	Set the order value for the created groups.
page	ALL	The UI configuration is being applied under the Properties page.
required	ALL	If this value is set to "true", the field must be filled in; otherwise, a message stops you to close the dialog, after clicking OK.
setEnvVarForScripts	CONNECTIONS	If the value is set to "true", it allows you to access your environment variable with the token name, which is found under the description field. For example:
textBoxSettings	ALL	"default": Any created field can have a default. When you save a value as the default, that value is saved to the metadata for that particular connection. "length": Number of characters that you can type in the field. "multiLineEdit": The "False" value is set by default. This field is applicable for future use.
listViewSettings	ALL	"defaultListItem": When the value is '0' the field is blank. "editable": Set to "true" when the field will be editable. "list": Enter the list of items you want to display in the configuration field. "listMessage": This message displays when you hover the mouse on the field.
numericSettings	ALL	"default": The default value field.
type	ALL	Describes the type of the configuration field. The type goes along with the textBoxSettings, listViewSettings, and numericSettings with the specifics described above.

Text Box Example

Note

If you want to use the example below for Column or Load Table UI Configurations you have to remove

Note

"setEnvVarForScripts" and **"encryption"**

```
 "uiConfigFields": [
       {
             "description": "For token replacement in scripts use:\r\n
$WSL_SRCCFG_simpleTextBox$",
             "displayName": "Simple text box",
             "documentation": false,
             "encryption": "none",

             "group": "Example Text Box Fields",
             "masked": false,
             "name": "simpleTextBox",
             "order": 100,
             "page": "Properties",
             "required": false,
             "setEnvVarForScripts": false,
             "textBoxSettings": {
                    "default": "This is my default file c:\temp\text.txt",
                    "length": 256,
                    "multiLineEdit": false
       },
                    "type": "textBox"
       },
       {
             "description": "For access at runtime use environment variable:
WSL_SRCCFG_encryptedTextBox",
             "displayName": "Encrypted and masked text box",
             "documentation": false,
             "encryption": "wherescape",
             "group": "Example Text Box Fields",
             "masked": true,
             "name": "encryptedTextBox",
             "order": 200,
             "page": "Properties",
             "required": true,
             "setEnvVarForScripts": true,
             "textBoxSettings": {
                    "default": "This is my default",
                    "length": 256,
                    "multiLineEdit": false
             },
                    "type": "textBox"

       },
```

List View Example

```
                {
                    "description": "For token replacement in scripts use: $WSL_SRCCFG_simpleListView$",
                    "displayName": "Simple list view",
                    "documentation": false,
                    "encryption": "none",
                    "group": "Example List View Fields",
                    "listViewSettings": {
                        "defaultListItem": 3,
                        "editable": false,
                        "list": [
                            "My list item one",
                            "My list item two",
                            "My list item three",
                            "My list item four",
                            "My list item five"
                        ],
                        "listMessage": "Select an item from the list"
                    },
                    "masked": false,
                    "name": "simpleListView",
                    "order": 300,
                    "page": "Properties",
                    "required": false,
                    "setEnvVarForScripts": false,
                    "type": "listView"
                }, {
                    "description": "For token replacement in scripts use: $WSL_SRCCFG_editableListView$",
                    "displayName": "Editable list view",
                    "documentation": false,
                    "encryption": "none",
                    "group": "Example List View Fields",
                    "listViewSettings": {
                        "defaultListItem": 0,
                        "editable": true,
                        "length": 256,
                        "list": [
                            "My list item one",
                            "My list item two",
                            "My list item three",
                            "My list item four",
                            "My list item five"
                        ],
                        "listMessage": "Select an item from the list or type your own value"
                    },
                    "masked": false,
                    "name": "editableListView",
                    "order": 300,
                    "page": "Properties",
                    "required": true,
                    "setEnvVarForScripts": false,
                    "type": "listView"
                }, {
                    "description": "For token replacement in scripts use: $WSL_SRCCFG_booleanListView$",
                    "displayName": "Boolean list view",
                    "documentation": false,
                    "encryption": "none",
                    "group": "Example List View Fields",
                    "listViewSettings": {
                        "defaultListItem": 1,
                        "editable": false,
                        "list": [
                            "TRUE",
                            "FALSE"
                        ],
                        "listMessage": "Set something True or False"
                    },
                    "masked": false,
                    "name": "booleanListView",
                    "order": 400,
                    "page": "Properties",
                    "required": false,
                    "setEnvVarForScripts": false,
                    "type": "listView"
                },
```

Numeric Fields Example

```
{
        "description": "For token replacement in scripts use: $WSL_SRCCFG_numeric$",
        "displayName": "Simple numeric field",
        "documentation": false,
        "encryption": "none",
        "group": "Example Numeric Fields",
        "masked": false,
        "name": "numeric",
        "numericSettings": {
                "defaultValue": 999
        },
        "order": 500,
        "page": "Properties",
        "required": false,
        "setEnvVarForScripts": false,
        "type": "numeric"
}
```

Host Script Languages

WhereScape RED provides a facility for defining and managing host script languages to enable users to implement host scripts in their preferred scripting language, e.g. Python, Perl, PowerShell, etc.

This provide RED users with the flexibility of using different types of script languages for the various script-based processing that they perform within RED, such as the script based update of the data warehouse content, created in custom database targets. Refer to **Script Templates for Custom Database Table Objects** for details.

This feature also supports importing a set of host script language definitions into RED from an external source, and exporting host script languages defined in RED to a file. This enables you to migrate and reuse host script language definitions between different repositories.

Using the host script languages feature involves defining the language properties which includes the command to run and execute a script in the defined language.

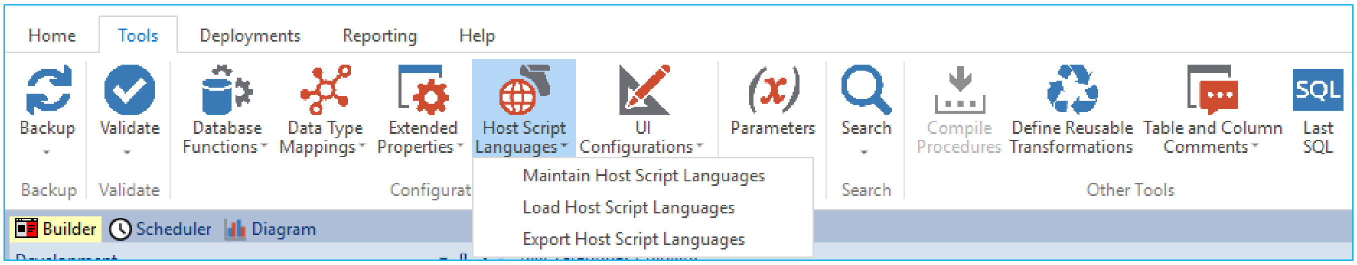
The host script language definitions are created and maintained in the **Maintain Host Script Languages** screen.

Defining Host Script Languages

Host Script Language definitions consist of the following :

Property	Description
Name	The name for the host script language, as displayed in the drop-down fields for selecting the script type (e.g. **Type** field in the **Host Script** and **Template** Properties screens—see **Applying Host Script Languages**). This name must be consistent and match across repositories, to preserve the associations with the host script language when deploying applications.
Description	The description of the host script language which is for display purposes only.
File Extension	The file extension used when the scripts are written to files, for example **py** for Python script files. If this is left blank, the generated files will have no extension.
Command	The command to execute a script in the defined language. The token $SCRIPT_NAME$ is replaced with the full path and file name of the script file to execute. The substituted command is used as the arguments for the PowerShell call operator (&). For example, "C:\Program Files\Python37\python.exe" "$SCRIPT_NAME$". The defined path must be valid for all RED clients and schedulers that are required to execute scripts for the defined host script language.

These user defined host script languages can be created and managed, using the **Host Script Languages** facility which is launched from the **Tools** menu.



The following options are available:

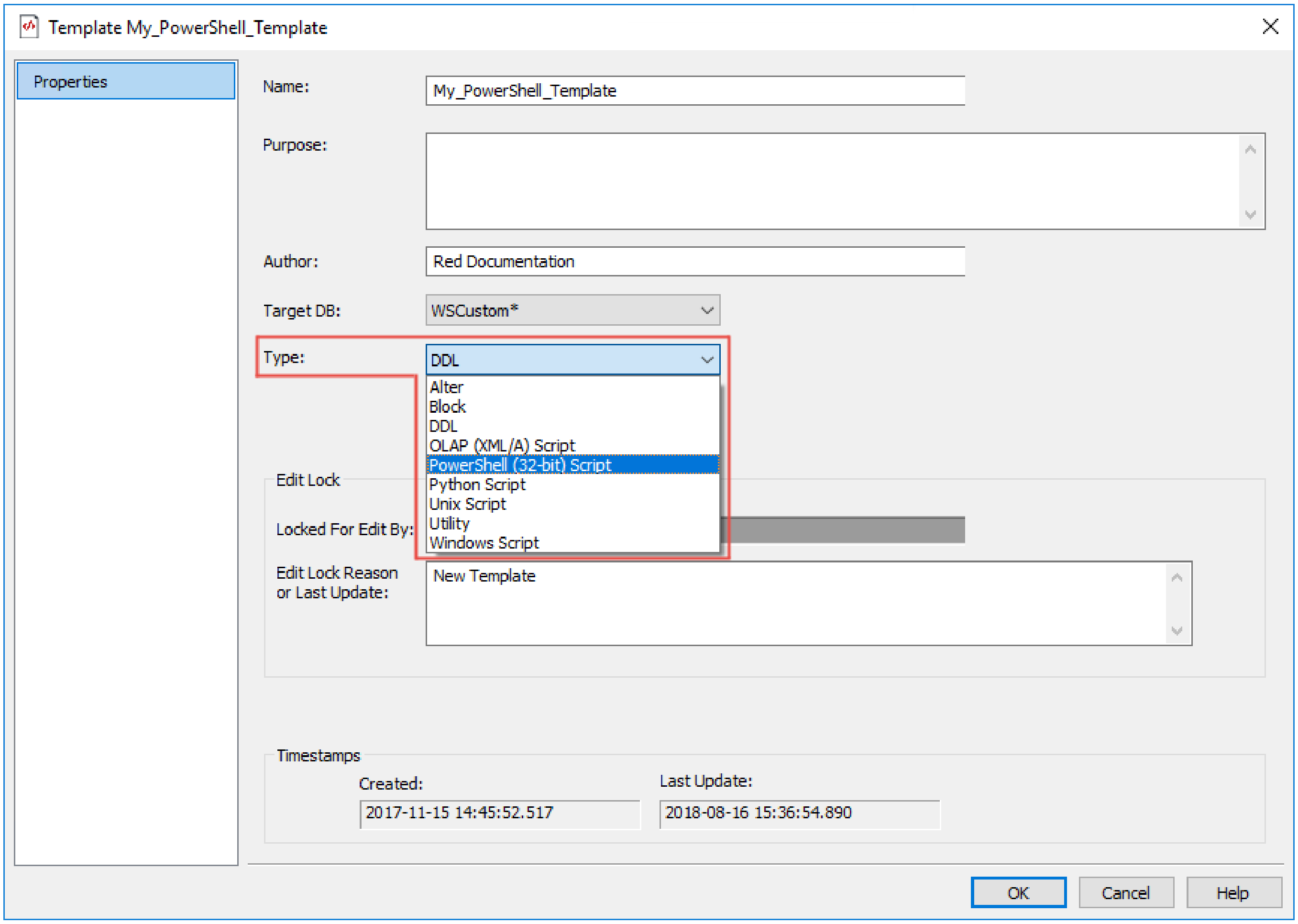
- **Maintain Host Script Languages** is used to add new or edit existing host script language definition settings.
- **Load Host Script Languages** to load a set of host script language definitions into RED from an external source. This option is typically used with a WhereScape provided enablement pack.
- **Export Host Script Languages** is used to export the available host script languages in RED to a host script language file (.hsclang).

Applying Host Script Languages

The defined Host Script Language types can be applied as a Template type or as the type of a Host Script object in RED. This is done via the **Properties** screen of the Template or Host Script object.

Template Properties

The **Type** field specifies what the template can be used for in RED. Refer to **Template Properties** for details.



Template My_PowerShell_Template

Properties

Name: My_PowerShell_Template

Purpose:

Author: Red Documentation

Target DB: WSCustom*

Type: DDL

- Alter
- Block
- DDL
- OLAP (XML/A) Script
- PowerShell (32-bit) Script
- Python Script
- Unix Script
- Utility
- Windows Script

Edit Lock

Locked For Edit By:

Edit Lock Reason or Last Update: New Template

Timestamps

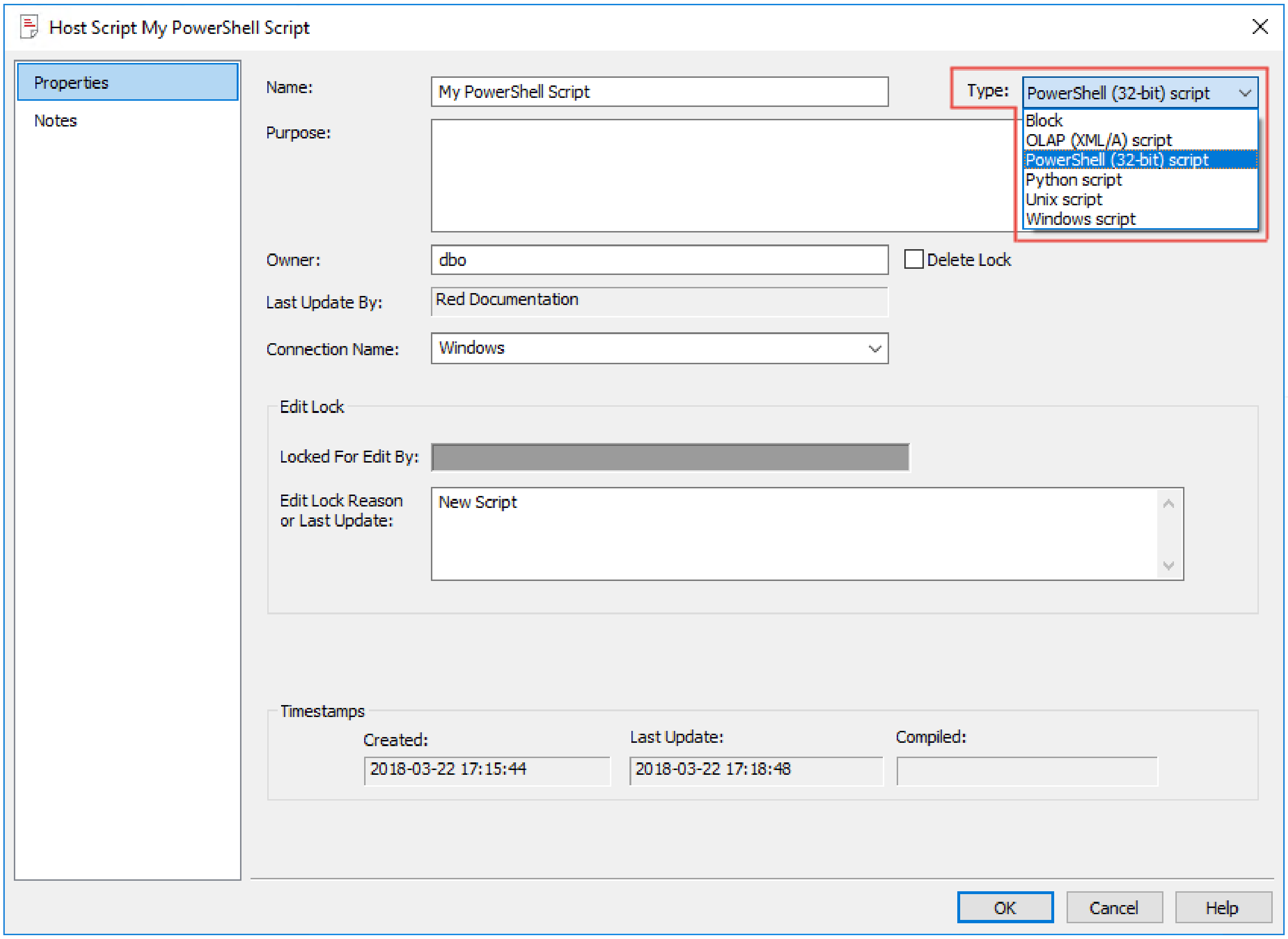
Created: 2017-11-15 14:45:52.517

Last Update: 2018-08-16 15:36:54.890

Host Script Properties

The **Type** field specifies what type of Script object is being created in RED.

Refer to **Script Generation** for details.



Host Script My PowerShell Script

Name: My PowerShell Script

Type: PowerShell (32-bit) script

Purpose:

Owner: dbo

☐ Delete Lock

Last Update By: Red Documentation

Connection Name: Windows

Edit Lock

Locked For Edit By:

Edit Lock Reason or Last Update: New Script

Timestamps

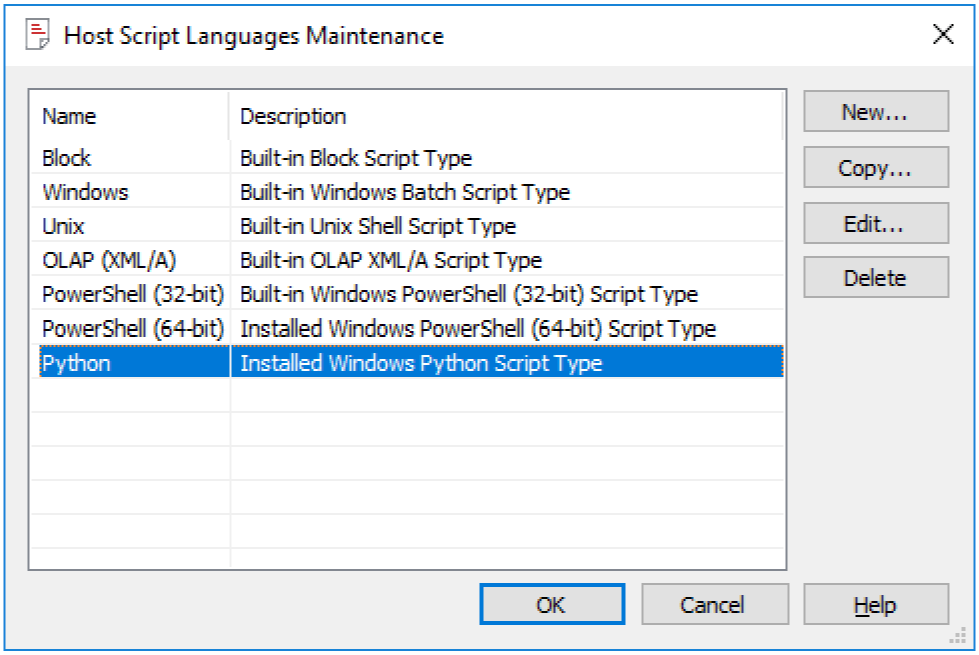
Created: 2018-03-22 17:15:44

Last Update: 2018-03-22 17:18:48

Compiled:

Maintaining Host Script Languages

The **Maintain Host Script Languages** screen displays a list of host script language types configured in RED and enables you to add a **New** host script language type. Clicking a host script language type from the list enables you to **Copy**, **Edit** or **Delete** the selected language type.



Name	Description
Block	Built-in Block Script Type
Windows	Built-in Windows Batch Script Type
Unix	Built-in Unix Shell Script Type
OLAP (XML/A)	Built-in OLAP XML/A Script Type
PowerShell (32-bit)	Built-in Windows PowerShell (32-bit) Script Type
PowerShell (64-bit)	Installed Windows PowerShell (64-bit) Script Type
Python	Installed Windows Python Script Type

The following describe the elements of the **Maintain Host Script Languages** screen.

Host Script Languages

This pane lists the built-in and user defined host script language types that have been setup in RED. It details the name and description of each host script language available in RED.

Function Buttons

These buttons enable you to create and manage the user defined host script language type definitions.

Button	Description
New	Launches the **Edit Host Script Language** screen which enables you to create and specify the properties of the new host script language type. Refer to **Adding a Host Script Language Type** for details.
Copy	Duplicates the selected host script language type. All fields are populated with the values of the original host script language type which can be edited as required.
Edit	Launches the **Edit Host Script Language** screen which enables you to edit the selected host script language type.
Delete	Deletes the selected host script language type, if it is not currently in use. The user is notified if it is in use and cannot be deleted.

Notes:

- Built-in host script languages in RED can only be viewed, they cannot be edited or deleted. Only **user defined** host script language types can be edited or deleted.
- Host script languages that are currently in use cannot be deleted, RED displays a message if you attempt to delete a language type that is in use.

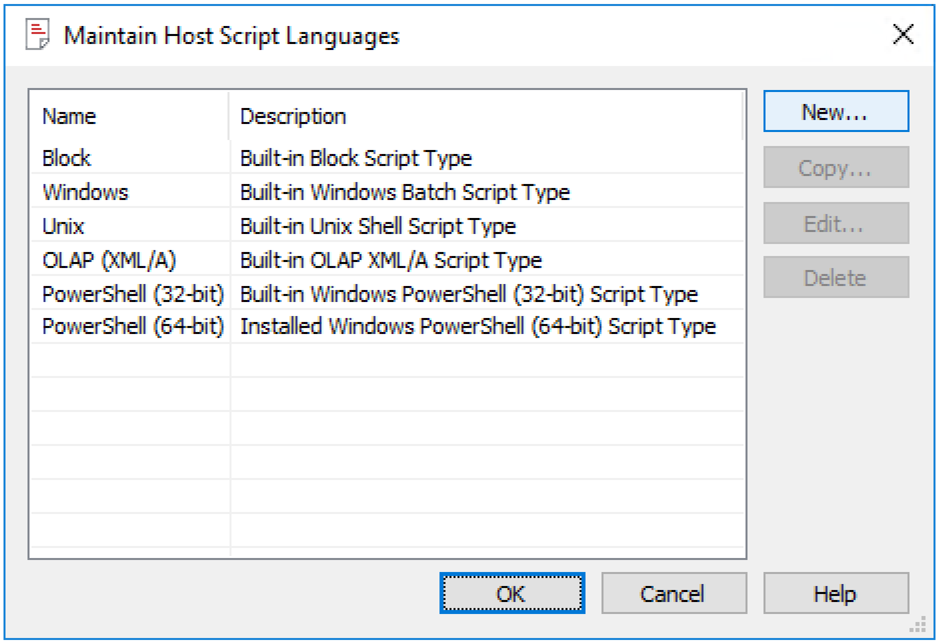
Adding a Host Script Language Type

To add a new host script language type, select **Tools > Host Script Languages > Maintain Host Script Languages**.

Notes:

For Unix host scripts for Load objects, RED leverages the Unix host script standard of defining the interpreter directive, after the Shebang (#!) character sequence in the first line of the Unix host script. This line provides the path to the command interpreter the operating system must use to execute the script.
For example:
#!/usr/bin/perl -T
This enables RED users to also leverage a number of Unix host script languages for loading data into their Load objects.

1. On the **Maintain Host Script Languages** screen, click **New**.

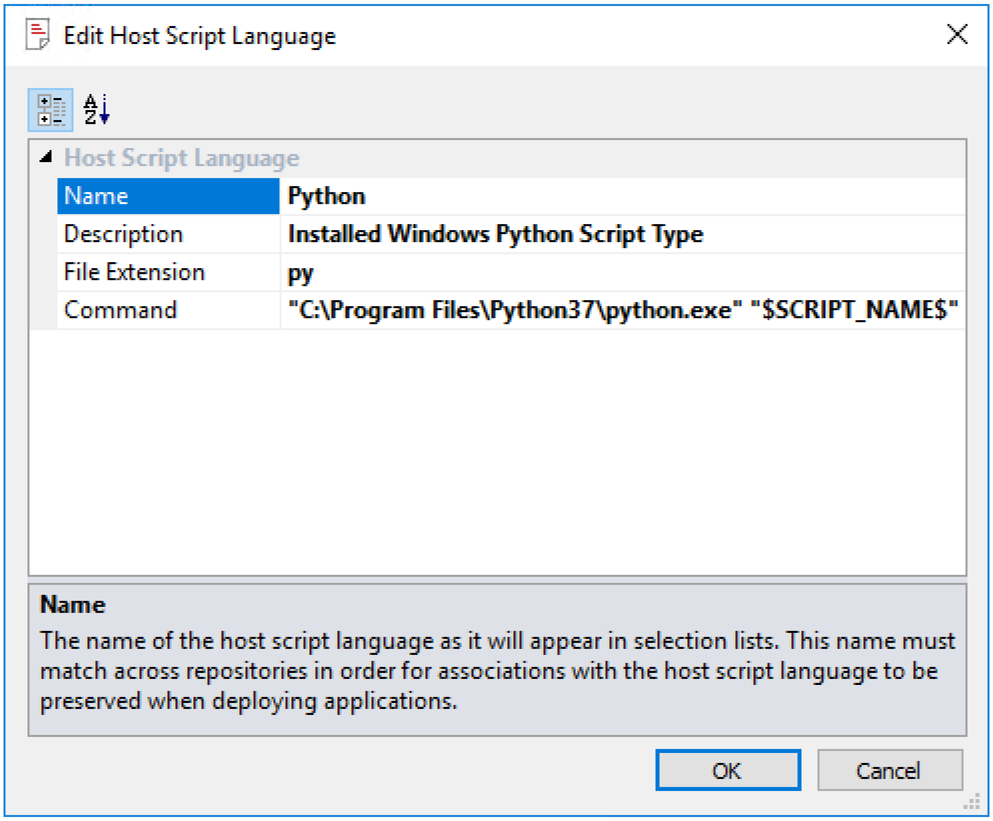


Name	Description
Block	Built-in Block Script Type
Windows	Built-in Windows Batch Script Type
Unix	Built-in Unix Shell Script Type
OLAP (XML/A)	Built-in OLAP XML/A Script Type
PowerShell (32-bit)	Built-in Windows PowerShell (32-bit) Script Type
PowerShell (64-bit)	Installed Windows PowerShell (64-bit) Script Type

2. On the **Edit Host Script Language** screen, enter the properties of the host script language being added.

Tip:

Clicking a field name displays a description of the property at the bottom of the screen,



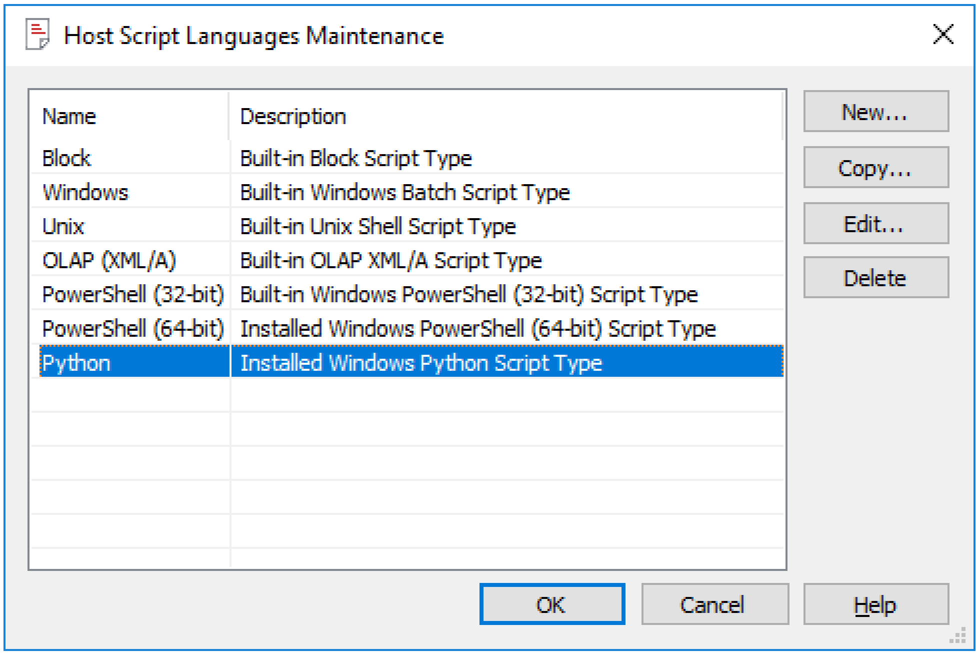
Host Script Language

Name	Python
Description	Installed Windows Python Script Type
File Extension	py
Command	"C:\Program Files\Python37\python.exe" "$SCRIPT_NAME$"

Name

The name of the host script language as it will appear in selection lists. This name must match across repositories in order for associations with the host script language to be preserved when deploying applications.

3. Click **OK** to exit the **Edit Host Script Language** screen. The newly added host script language type is listed in the **Maintain Host Script Languages** screen.



Name	Description
Block	Built-in Block Script Type
Windows	Built-in Windows Batch Script Type
Unix	Built-in Unix Shell Script Type
OLAP (XML/A)	Built-in OLAP XML/A Script Type
PowerShell (32-bit)	Built-in Windows PowerShell (32-bit) Script Type
PowerShell (64-bit)	Installed Windows PowerShell (64-bit) Script Type
Python	Installed Windows Python Script Type

4. Click **OK** to save the new host script language type.

Note:

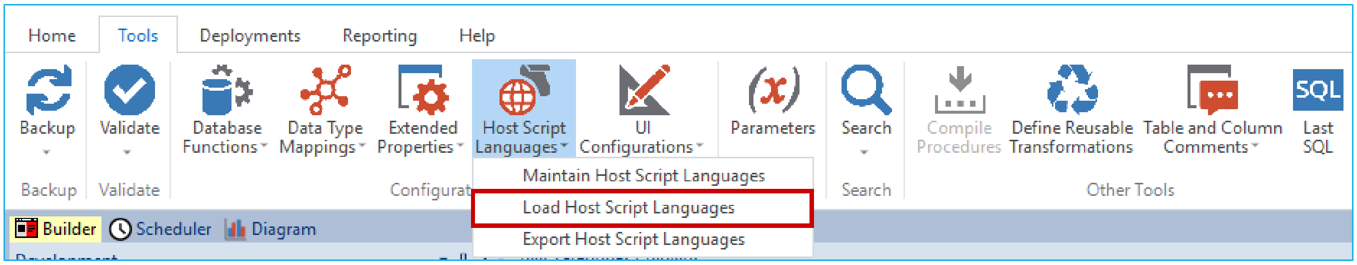
User defined host script language definitions can be copied between RED meta data repositories, using **Tools > Import Metadata Objects** feature, which enables you to select an object (host script or template) to import that has the required host script type.

Host Script Languages Data Migration Between Repositories

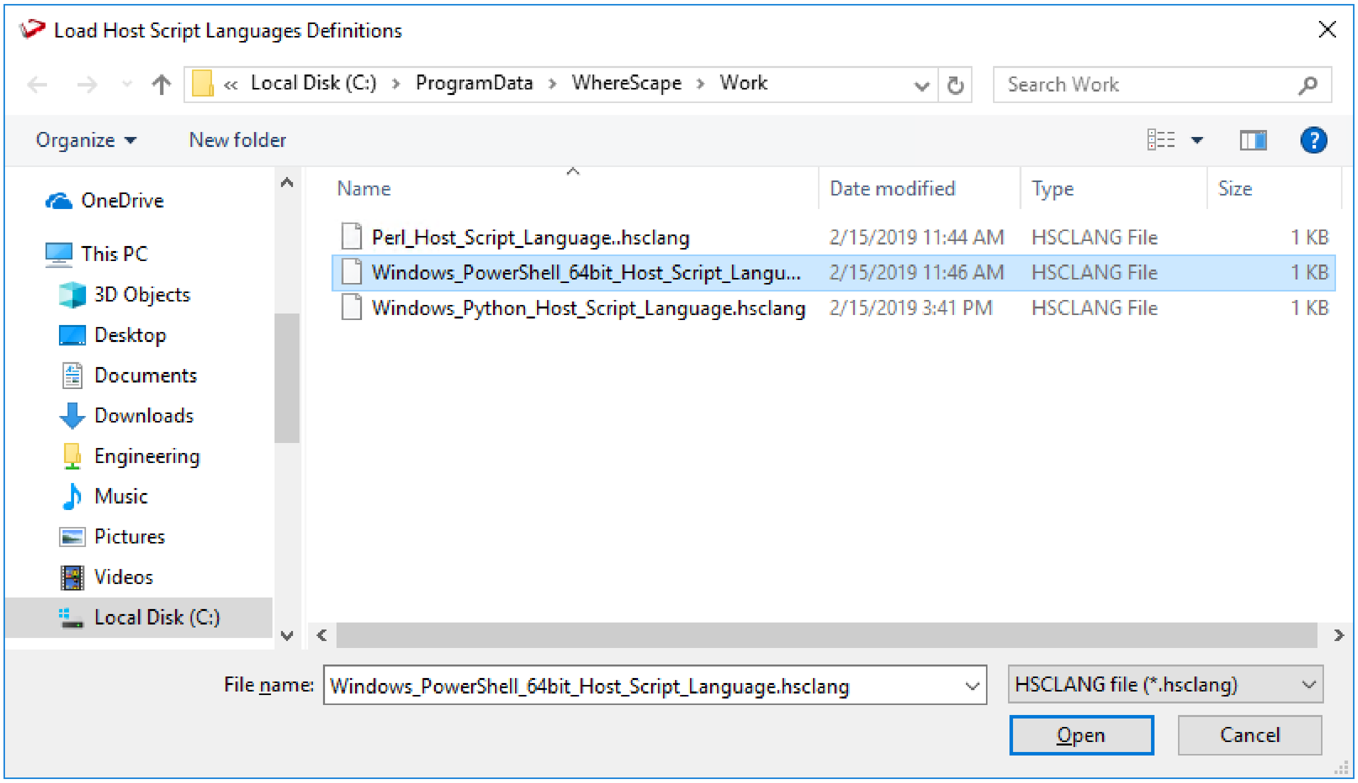
Host script languages defined and set can be propagated to the other RED repositories. The definitions are exported from the source repository and imported to the target repository via the **Tools > Host Script Languages** menu.

Loading Host Languages

To load host script language file, select **Tools > Host Script Languages > Load Host Script Languages**.



The following window is displayed. Select the host script language (HSCLANG) file to load . By default, RED expects the hsclang files to be in **Program Files\WhereScape\Work** directory, but this can be changed.

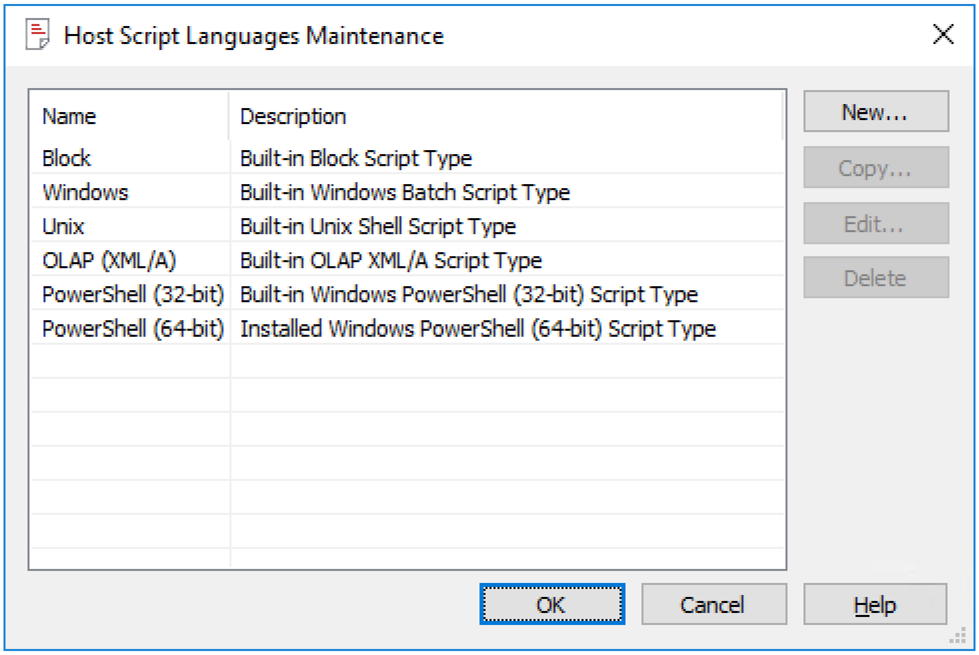


Name	Date modified	Type	Size
Perl_Host_Script_Language..hsclang	2/15/2019 11:44 AM	HSCLANG File	1 KB
Windows_PowerShell_64bit_Host_Script_Langu...	2/15/2019 11:46 AM	HSCLANG File	1 KB
Windows_Python_Host_Script_Language.hsclang	2/15/2019 3:41 PM	HSCLANG File	1 KB

File name: Windows_PowerShell_64bit_Host_Script_Language.hsclang

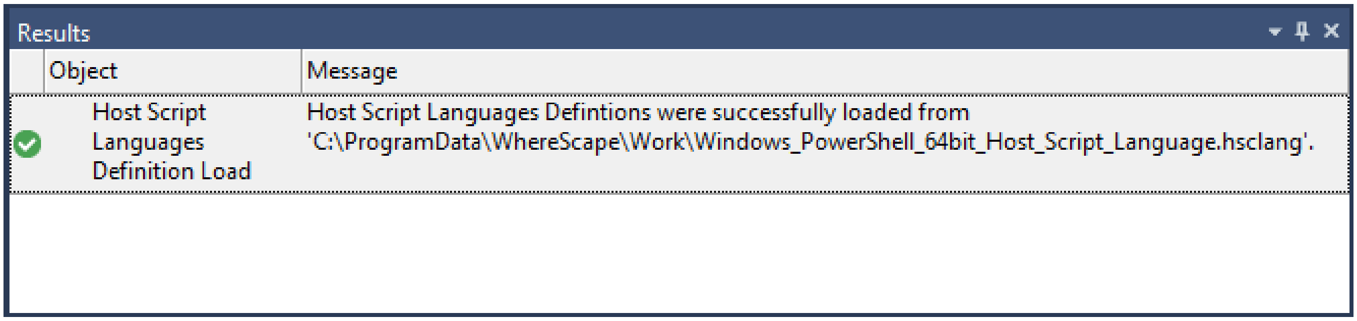
HSCLANG file (*.hsclang)

The selected hsclang file is loaded and listed in the **Maintain Host Script Languages** screen.



Name	Description
Block	Built-in Block Script Type
Windows	Built-in Windows Batch Script Type
Unix	Built-in Unix Shell Script Type
OLAP (XML/A)	Built-in OLAP XML/A Script Type
PowerShell (32-bit)	Built-in Windows PowerShell (32-bit) Script Type
PowerShell (64-bit)	Installed Windows PowerShell (64-bit) Script Type

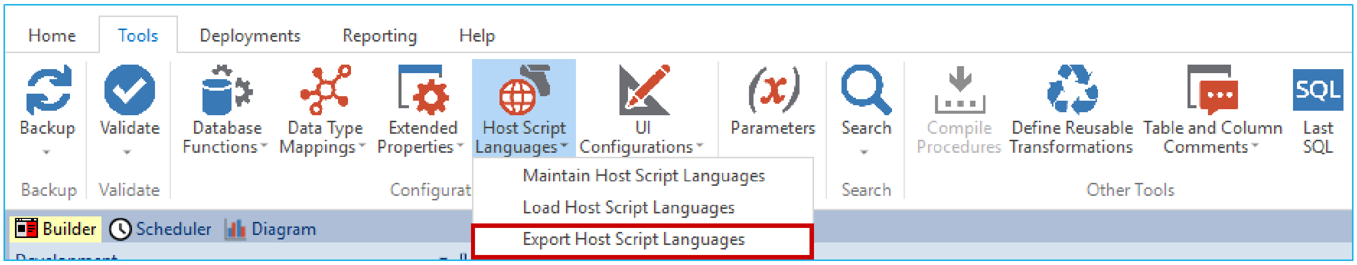
A confirmation message is displayed in the **Results** pane.



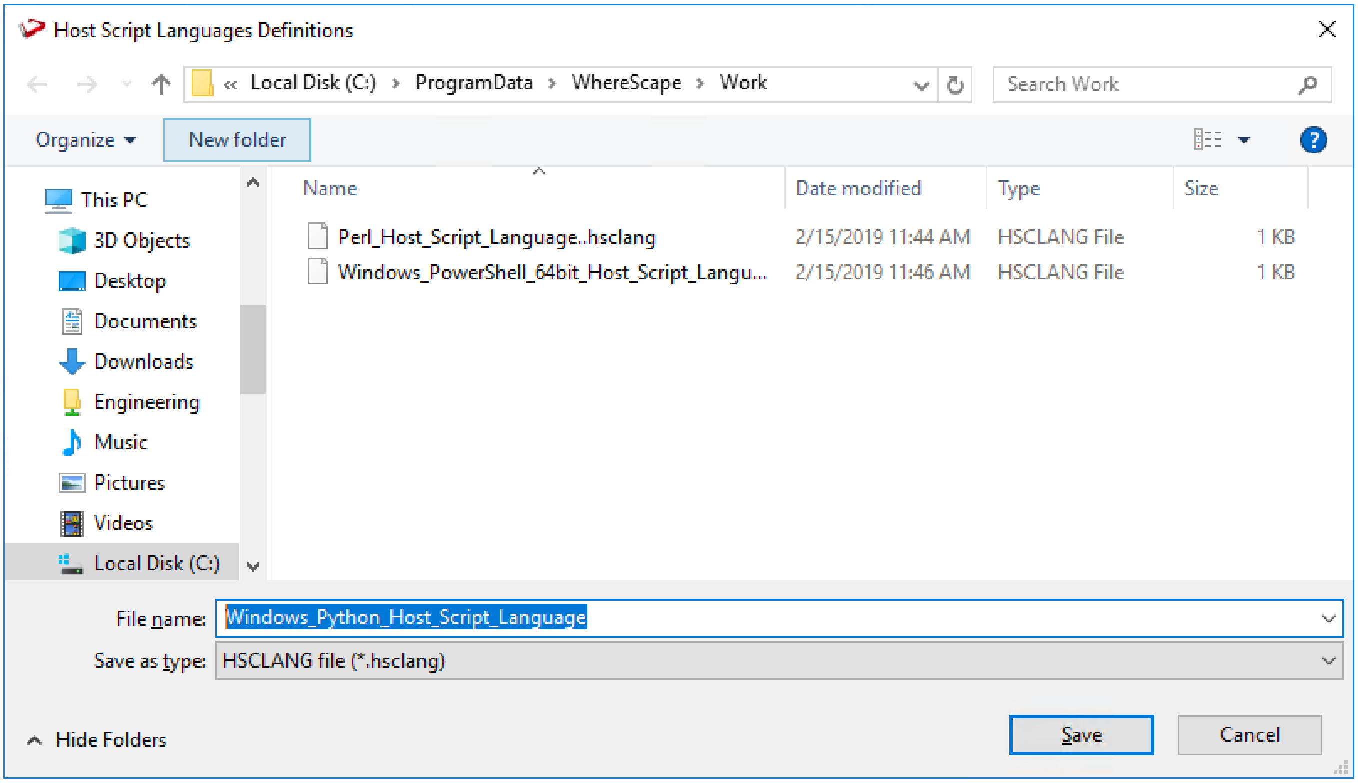
Object	Message
✓ Host Script Languages Definition Load	Host Script Languages Defintions were successfully loaded from 'C:\ProgramData\WhereScape\Work\Windows_PowerShell_64bit_Host_Script_Language.hsclang'.

Exporting Host Script Languages

To export host script languages from RED, select **Tools > Host Script Languages > Export Host Script Languages**.



The following window is displayed. By default, RED exports the host script language (HSCLANG) file to **Program Files\WhereScape\Work** directory, but this can be changed. Type in the **File name** and then click **Save**.

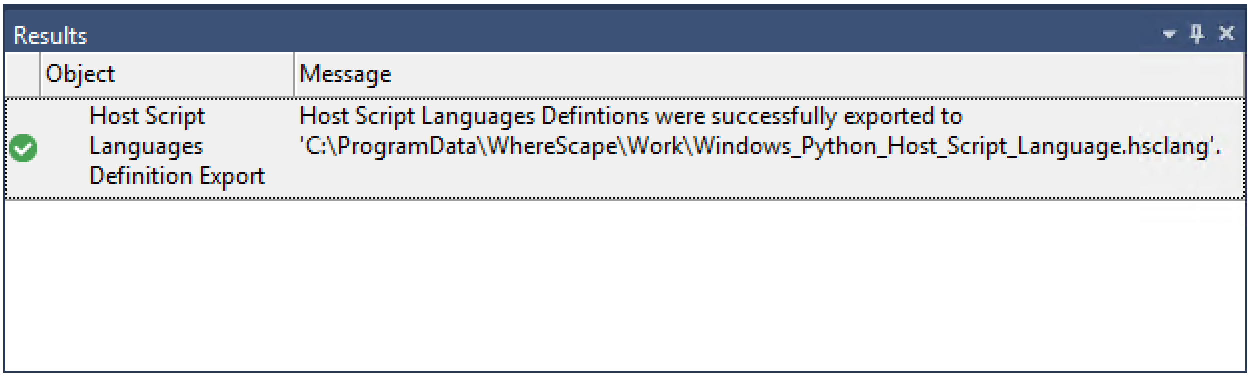


Name	Date modified	Type	Size
Perl_Host_Script_Language..hsclang	2/15/2019 11:44 AM	HSCLANG File	1 KB
Windows_PowerShell_64bit_Host_Script_Langu...	2/15/2019 11:46 AM	HSCLANG File	1 KB

File name: Windows_Python_Host_Script_Language

Save as type: HSCLANG file (*.hsclang)

A confirmation message is displayed in the **Results** pane.



Object	Message
Host Script Languages Definition Export	Host Script Languages Defintions were successfully exported to 'C:\ProgramData\WhereScape\Work\Windows_Python_Host_Script_Language.hsclang'.

Dedicated Command Line Interface (REDCLI)

WhereScape RED provides a dedicated command line interface called **REDCLI** which enables you to perform command functions. The **REDCLI** tool is available in the RED Windows installation directory. This tool can be called from 3rd party applications or Windows scripting languages to perform common functions.

REDCLI provides three usage modes:

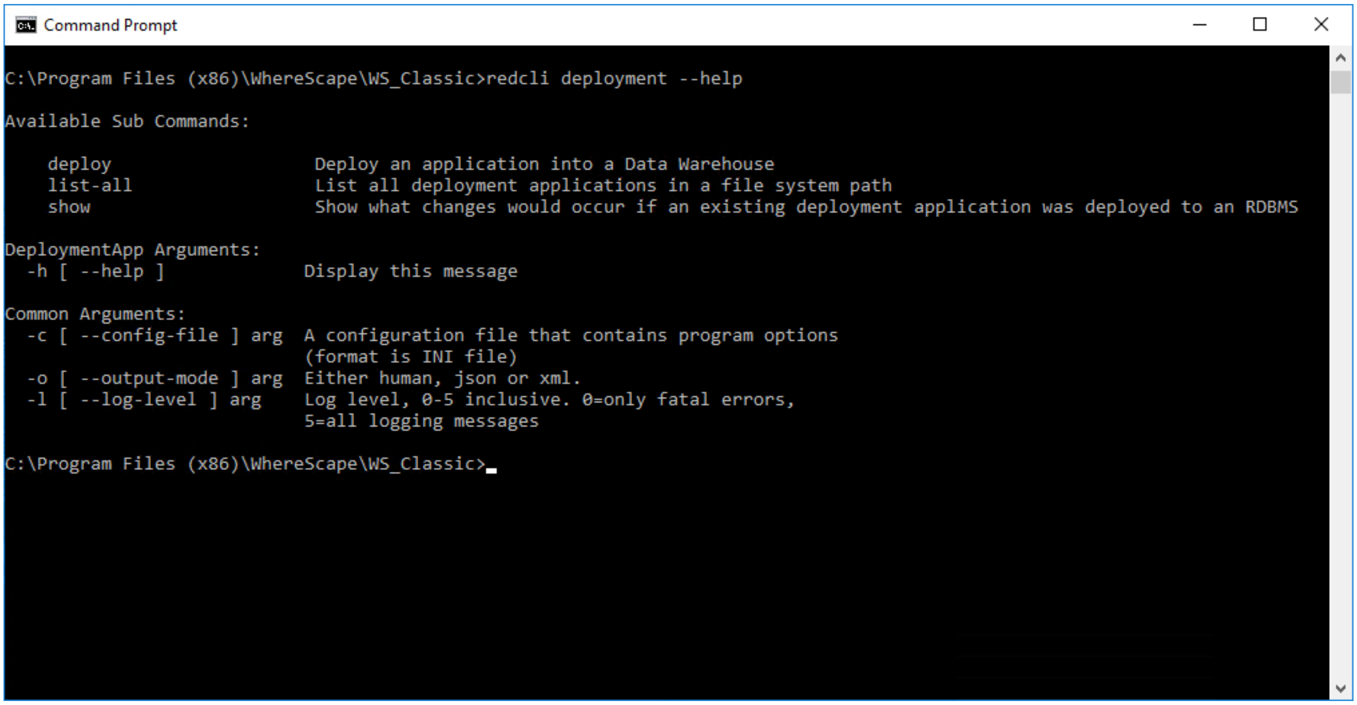
Mode	Description
Standard	In this mode, one action is run and you authenticate to the RED repository for each action. The use case for this mode is if you only have one action to perform, e.g. deploying an application.
Interactive	In this mode, you connect to the target RED repository once and have a persistent connection to run multiple **REDCLI** actions. If you have several commands to run, using this mode removes the overhead of several database connections and reduces the command line options required for each action by not requiring the meta connection and logging options that were established on the initial interactive command line. This mode might be useful in cases where you have multiple actions to perform but have not prepared a batch file.
Batch	In this mode, you have a file containing a series of **REDCLI** actions that are executed sequentially. One use case is loading an enablement pack of templates, data type mapping sets, database function sets. This is particularly useful if you want to repeat a set of actions in multiple RED repositories.

A command line help for each command within the **REDCLI** is available and the option level actions are as follows:

Command	Description
batch	Execute a batch of commands, and generate an empty batch file.
connection	Manage all aspects of connections.
dfs	Manage all aspects of database function sets.
dtm	Manage all aspects of data type mapping sets.
deployment	List, deploy and show the contents of deployment applications.
ext-prop-definition	Manage all aspects of extended property definitions.
ext-prop-value	Manage all aspects of extended property values.
script-lang-definition	Manage all aspects of host script language definitions.
interactive	Collect commands interactively from standard input and execute them.
license	Install, uninstall or show a license.
object	List information and generate update routines for objects in a metadata repository.

Command	Description
	The **REDCLI** command 'object generate-routine' works in conjunction with the **Generate load scripts and update routine for objects** setting in the **Load Application > Metadata Options** window of the RED Setup Administrator. Refer to the section **Application Load Process** of the RED Install Guide for details. The default template for the table objects must be set in the target connection, before generating update routines via **REDCLI**. Refer to Connection Properties and **Connection Routine Templates** for details. **Notes:** Template-based update routine code generation for table objects: • Existing RED table objects uses the Template associated with the object, e.g. the current procedure or script that was used to generate the update routine. Refer to **Rebuilding Update Procedures > Generating an Update Procedure via a Template** for details. • New RED table objects use the default update routine Template, set in the target connection. Refer to **Connection Routine Templates** for details.
odbc	List and validate ODBC DSNs in the system.
options	List, find, import and export Tools Options.
parameter	Manage all aspects of parameters.
powershell	Install WhereScape specific PowerShell modules.
procedure	Import, export and list all procedures.
repository	Create, upgrade and validate a metadata repository.
scheduler	Manage all aspects of scheduler configurations.
script	Import, export and list all scripts.
target	Manage all aspects of targets.
template	Import, export and list all templates.
ui-config	Import, export, list-all, show, and delete UI Configurations

Each command has its own help, for example `'RedCli deployment --help'` lists all available commands relevant to deployment applications.



```
C:\Program Files (x86)\WhereScape\WS_Classic>redcli deployment --help

Available Sub Commands:

    deploy                    Deploy an application into a Data Warehouse
    list-all                  List all deployment applications in a file system path
    show                      Show what changes would occur if an existing deployment application was deployed to an RDBMS

DeploymentApp Arguments:
  -h [ --help ]              Display this message

Common Arguments:
  -c [ --config-file ] arg   A configuration file that contains program options
                             (format is INI file)
  -o [ --output-mode ] arg   Either human, json or xml.
  -l [ --log-level ] arg     Log level, 0-5 inclusive. 0=only fatal errors,
                             5=all logging messages

C:\Program Files (x86)\WhereScape\WS_Classic>
```

Example to deploy a RED application:

```
REDCLI deployment deploy --app-number myproject --app-version v01 --app-directory
C:\Workspace\RedCli\Applications\ --options-location defaults --dest-connection-name
DataWarehouse --dest-target-name Schema --dest-scheduler-name qa_scheduler --
continue-ver-mismatch --meta-database orcl12_ODW --meta-dsn ORCL-orcl12_ODW --meta-
password Wsl12345 --meta-user-name orcl12_ODW --output-mode json --log-level 5
```

RED Command Line Interface

WhereScape RED provides command line interface applications that you can use to perform different tasks or execute functions and operations outside the RED GUI.

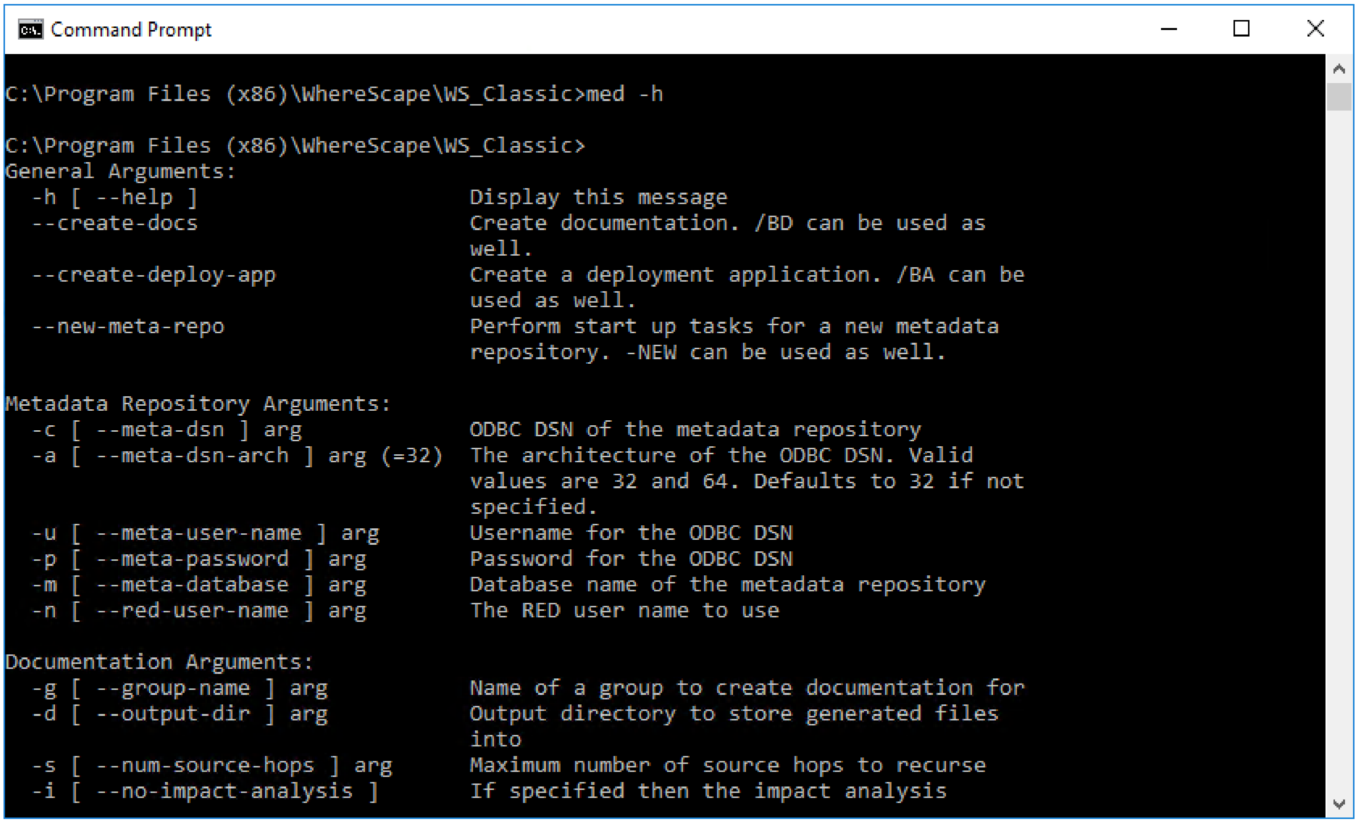
- RED Client Command Line
- RED Setup Administrator Command Line

RED Client Command Line (med.exe)

The RED Client command line interface enables you to execute commands to:

- Generate documentation in batch
- Create and deploy applications in batch
- Perform start up tasks for a new metadata repository

The following is a sample Windows Command Prompt that displays the help for RED Client (**med.exe**) arguments.



```
C:\Program Files (x86)\WhereScape\WS_Classic>med -h

C:\Program Files (x86)\WhereScape\WS_Classic>
General Arguments:
  -h [ --help ]                  Display this message
  --create-docs                  Create documentation. /BD can be used as
                                 well.
  --create-deploy-app            Create a deployment application. /BA can be
                                 used as well.
  --new-meta-repo                Perform start up tasks for a new metadata
                                 repository. -NEW can be used as well.

Metadata Repository Arguments:
  -c [ --meta-dsn ] arg          ODBC DSN of the metadata repository
  -a [ --meta-dsn-arch ] arg (=32)  The architecture of the ODBC DSN. Valid
                                 values are 32 and 64. Defaults to 32 if not
                                 specified.
  -u [ --meta-user-name ] arg    Username for the ODBC DSN
  -p [ --meta-password ] arg     Password for the ODBC DSN
  -m [ --meta-database ] arg     Database name of the metadata repository
  -n [ --red-user-name ] arg     The RED user name to use

Documentation Arguments:
  -g [ --group-name ] arg        Name of a group to create documentation for
  -d [ --output-dir ] arg        Output directory to store generated files
                                 into
  -s [ --num-source-hops ] arg   Maximum number of source hops to recurse
  -i [ --no-impact-analysis ]    If specified then the impact analysis
```

RED Client Command Line Arguments

The following describe the command line arguments that can be used in RED Client (**med.exe**) arguments.

Description	Argument long name	Argument short name
Display the RED arguments. Note: No other arguments required.	--help	-h or /h
Batch documentation create.	--create-docs	-BD or /BD

Description	Argument long name	Argument short name
Note: Must have other arguments, refer to **Batch Documentation Creation**.		
Batch deployment application create. Note: Must have other arguments, refer to **Batch Deployment Application Creation**.	--create-deploy-app	-BA or /BA
First start with a new repository. Note: Must have other arguments, refer to **New Metadata Repository Start up Task Creation**.	--new-meta-repo	-NEW or /NEW
Group name Note: All objects included, if group name is not specified.	--group-name <name>	-G or /G
Multiple use: Output directory or Database schema.	--output-dir <path name> or <db_schema>	-D or /D
Number of processes/ hops in the source diagrams.	--num-source-hops <number>	-S or /S
Disable impact analysis	--no-impact-analysis	-I- or /I-
ODBC DSN	--meta-dsn <name>	-O or /O
ODBC user name	--meta-user-name <name>	-L or /L
ODBC DSN architecture	--meta-dsn-arch <32|64>	-A or /A
ODBC username	--meta-user-name <name>	-U or /U
ODBC password	--meta-password <password>	-P or /P
Connection name	--meta-dsn <name>	-C or /C
Session user name	--red-user-name <name>	-N or /N
Application ID number	--app-number <name>	-I or /I
Application version number	--app-version <number>	-V or /V
Application ID file name	--app-file-name <path name>	-AF or /AF
Remove connections	--remove-connections	-RC or /RC
Remove parameters	--remove-parameters	-RP or /RP
Remove scheduler jobs	--remove-jobs	-RJ or /RJ
Group name	--group-name <name>	-AG or /AG
Project name	--project-name <name>	-AP or /AP
All objects	--deploy-all	-ALL or /ALL

The succeeding sections outline how the arguments are used.

Batch Documentation Creation

The following describe the relevant command line arguments for creating batch documentation (--create-docs or -BD):

Description	Argument long name	Argument short name
ODBC DSN	--meta-dsn <name>	-C or /C
ODBC DSN architecture	--meta-dsn-arch <32|64>	-A or /A
ODBC user name	--meta-user-name <name>	-U or /U
ODBC password	--meta-password <password>	-P or /P
RED user name	--red-user-name <name>	-N or /N
Group name Note: All objects included if group name is not specified.	--group-name <name>	-G or /G
Output directory	--output-dir <path name>	-D or /D
Number of processes/ hops in the source diagrams	--num-source-hops <number>	-S or /S
Disable impact analysis	--no-impact-analysis	-I- or /I-

Note:

Refer to **Batch Documentation Creation** for details.

Batch Deployment Application Creation

The following describe the relevant command line arguments for creating batch deployment application (--create-deploy-app or -BA):

Description	Argument long name	Argument short name
Output directory	--output-dir <path name>	-D or /D
ODBC DSN	--meta-dsn <name>	-C or /C
ODBC DSN architecture	--meta-dsn-arch <32|64>	-A or /A
ODBC user name	--meta-user-name <name>	-U or /U
ODBC password	--meta-password <password>	-P or /P
RED user name	--red-user-name <name>	-N or /N
Application ID number	--app-number <name>	-I or /I
Application version number	--app-version <number>	-V or /V
Application ID file name (e.g. app_id_xxx.wst)	--app-file-name <path name>	-AF or /AF
Remove connections	--remove-connections	-RC or /RC
Remove parameters	--remove-parameters	-RP or /RP
Remove scheduler jobs	--remove-jobs	-RJ or /RJ
Group name	--group-name <name>	-AG or /AG
Project name	--project-name <name>	-AP or /AP
All objects	--deploy-all	-ALL or /ALL

Note:

Refer to **Creating and Loading Applications from the Command Line** for details.

New Metadata Repository Start up Task Creation

The following describe the relevant command line arguments for creating start up tasks for a new metadata repository (--new-meta-repo or -NEW):

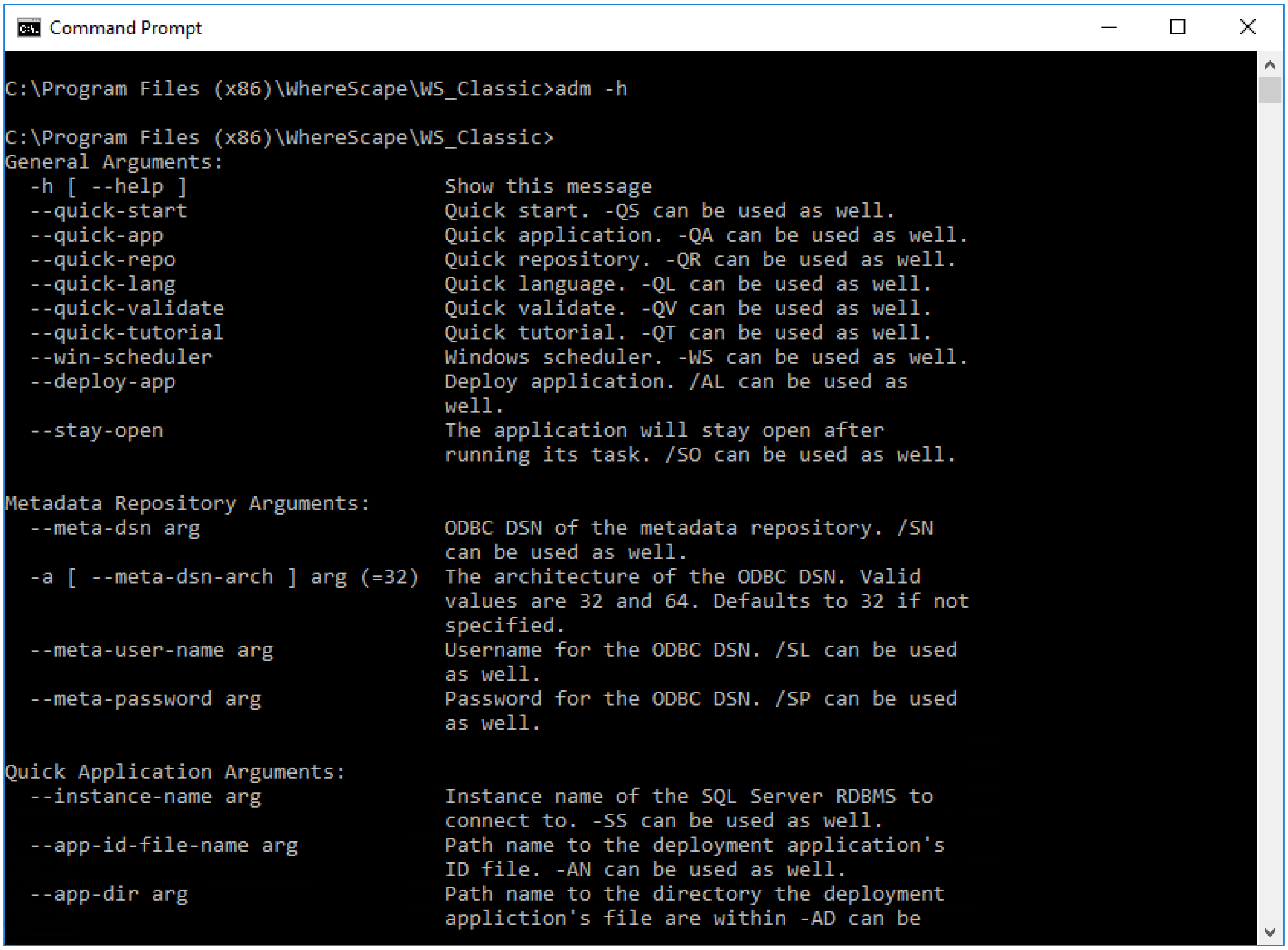
Description	Argument long name	Argument short name
ODBC DSN	--meta-dsn <name>	-O or /O
ODBC DSN architecture	--meta-dsn-arch <32|64>	-A or /A
ODBC user name	--meta-user-name <name>	-L or /L
Metadata database	--meta-database <name>	-D or /D

RED Setup Administrator Command Line (adm.exe)

The RED Setup Administrator command line interface enables you to execute commands to :

- Quickly deploy a WhereScape RED application to create and populate a data warehouse
- Quickly create a RED repository with the required metadata tables
- Quickly load a file that contains language data into RED
- Quickly validate a RED repository and its metadata tables
- Quickly connect to a tutorial database to use in RED
- Install and configure a Windows scheduler for RED
- Deploy a RED application

The following is a sample Windows Command Prompt that displays the help for RED Setup Administrator (**adm.exe**) arguments.



```
C:\Program Files (x86)\WhereScape\WS_Classic>adm -h

C:\Program Files (x86)\WhereScape\WS_Classic>
General Arguments:
  -h [ --help ]                      Show this message
  --quick-start                      Quick start. -QS can be used as well.
  --quick-app                        Quick application. -QA can be used as well.
  --quick-repo                       Quick repository. -QR can be used as well.
  --quick-lang                       Quick language. -QL can be used as well.
  --quick-validate                   Quick validate. -QV can be used as well.
  --quick-tutorial                   Quick tutorial. -QT can be used as well.
  --win-scheduler                    Windows scheduler. -WS can be used as well.
  --deploy-app                       Deploy application. /AL can be used as
                                     well.
  --stay-open                        The application will stay open after
                                     running its task. /SO can be used as well.

Metadata Repository Arguments:
  --meta-dsn arg                     ODBC DSN of the metadata repository. /SN
                                     can be used as well.
  -a [ --meta-dsn-arch ] arg (=32)   The architecture of the ODBC DSN. Valid
                                     values are 32 and 64. Defaults to 32 if not
                                     specified.
  --meta-user-name arg               Username for the ODBC DSN. /SL can be used
                                     as well.
  --meta-password arg                Password for the ODBC DSN. /SP can be used
                                     as well.

Quick Application Arguments:
  --instance-name arg                Instance name of the SQL Server RDBMS to
                                     connect to. -SS can be used as well.
  --app-id-file-name arg             Path name to the deployment application's
                                     ID file. -AN can be used as well.
  --app-dir arg                      Path name to the directory the deployment
                                     appliction's file are within -AD can be
```

RED Setup Administrator Command Line Arguments

The following describe the command line arguments that can be used in RED Setup Administrator (**adm.exe**).

Description	Argument long name	Argument short name
Display the RED arguments. Note: No other arguments required.	--help	-h or /h
Quick start Note: No other arguments required.	--quick-start	-QS or /QS
Quick application Note: Must have other arguments, refer to **Quick Application**.	--quick-app	-QA or /QA
Quick repository Note: Must have other arguments, refer to **Quick Repository**.	--quick-repo	-QR or /QR
Quick language	--quick-lang	-QL or /QL

Description	Argument long name	Argument short name
Note: Must have other arguments, refer to **Quick Language**.		
Quick validate Note: Must have other arguments, refer to **Quick Validate**.	--quick-validate	-QV or /QV
Quick tutorial Note: Must have other arguments, refer to **Quick Tutorial**.	--quick-tutorial	-QT or /QT
Windows scheduler Note: Must have other arguments, refer to **Windows Scheduler**.	--win-scheduler	-WS or /WS
Deploy application Note: Must have other arguments, refer to **Deploy Application**.	--deploy-app	-AL or /AL
Stay open Notes: • Use to make Setup Administrator remain open, after the application has been created. • Relevant for -AL LOAD option. • Relevant for -QA, -QR, -WS, -QL and -QV application switch options.	--stay-open	-SO or /SO

Note:

When --quick-start (-QS or /QS) is specified, then no other parameters are required.

The succeeding sections outline how the arguments are used.

Quick Application

The following describe the relevant command line arguments for quick application (--quick-app or -QA):

Description	Argument long name	Argument short name
Instance name	--instance-name <name>	-SS <name> or /SS <name>
New repository name	--meta-dsn <name>	-SN <name> or /SN <name>
Database login name	--meta-user-name <name>	-SL <name> or /SL <name>

Description	Argument long name	Argument short name
Database password	--meta-password <password>	-SP <password> or /SP <password>
Application ID file name	--app-id-file-name <name>	-AN <file name> or /AN <name>
Application file's directory	--app-dir <path name>	-AD <directory name> or /AD <name>
Application load instance	--app-instance <name>	-LI <name> or /LI <name>
Application trusted flag	--app-trusted	-LT or /LT
Application ODBC DSN	--app-meta-dsn <name>	-LD <name> or /LD <name>
ODBC DSN architecture	--meta-dsn-arch <32|64>	-a <32|64> or /a <32|64>
Application load database	--app-db-name <name>	-LN <name> or /LN <name>
Application job name	--app-job-name <name>	-JN <name> or /JN <name>
Datawarehouse connection name	--dw-con-name <name>	-DC <name> or /DC <name>
MSAS cube connection name	--msas-con-name <name>	-CC <name> or /CC <name>
MSAS cube database name	--msas-db-name <name>	-CD <name> or /CD <name>
Scheduler login name	--sched-login-name <name>	-ML <name> or /ML <name>
Scheduler password	--sched-password <password>	-MP <password> or /MP <password>

Note:

Refer to the **SQL Server Quick Application** section of the WhereScape RED Setup Administrator User Guide for details.

Quick Repository

The following describe the relevant command line arguments for quick repository (--quick-repo or -QR):

Description	Argument long name	Argument short name
ODBC DSN	--meta-dsn <name>	-SN <name> or /SN <name>
ODBC DSN architecture	--meta-dsn-arch <32|64>	-a <32|64> or /a <32|64>
ODBC login name	--meta-user-name <name>	-SL <name> or /SL <name>
ODBC password	--meta-password <password>	-SP <password> or /SP <password>
TDPID	--tdpid <id>	-DI <name> or /DI <name>
Dimension tablespace	--dim-space <name>	-DS <name> or /DS <name>
Index tablespace	--index-space <name>	-IS <name> or /IS <name>
Metadata tablespace	--meta-space <name>	-MS <name> or /MS <name>

Note:

Refer to the **SQL Server Quick Application** section of the WhereScape RED Setup Administrator User Guide for details.

Quick Language

The following describe the relevant command line arguments for quick language (--quick-lang or -QL):

Description	Argument long name	Argument short name
ODBC DSN	--meta-dsn <name>	-SN <name> or /SN <name>
ODBC DSN architecture	--meta-dsn-arch <32|64>	-a <32|64> or /a <32|64>
ODBC login name	--meta-user-name <name>	-SL <name> or /SL <name>
ODBC password	--meta-password <password>	-SP <password> or /SP <password>
Language file	--lang-file-name <name>	-LF <path name> or /LF <path name>
Add language if absent	--add-lang	-Add or /ADD

Note:

Refer to the **Languages** section of the WhereScape RED Setup Administrator User Guide for details.

Quick Validate

The following describe the relevant command line arguments for quick validate (--quick-validate or -QV):

Description	Argument long name	Argument short name
ODBC DSN	--meta-dsn <name>	-SN <name> or /SN <name>
ODBC DSN architecture	--meta-dsn-arch <32|64>	-a <32|64> or /a <32|64>
ODBC login name	--meta-user-name <name>	-SL <name> or /SL <name>
ODBC password	--meta-password <password>	-SP <password> or /SP <password>
Dimension tablespace	--dim-space <name>	-DS <name> or /DS <name>
Index tablespace	--index-space <name>	-IS <name> or /IS <name>
Metadata tablespace	--meta-space <name>	-MS <name> or /MS <name>

Note:

Refer to the **Validating a Metadata Repository using the Command Line** section of the WhereScape RED Setup Administrator User Guide for details.

Quick Tutorial

The following describe the relevant command line arguments for quick tutorial (--quick-tutorial or -QT):

Description	Argument long name	Argument short name
ODBC DSN	--meta-dsn <name>	-SN <name> or /SN <name>
ODBC DSN architecture	--meta-dsn-arch <32|64>	-a <32|64> or /a <32|64>

Description	Argument long name	Argument short name
ODBC login name	--meta-user-name <name>	-SL <name> or /SL <name>
ODBC password	--meta-password <password>	-SP <password> or /SP <password>

Note:

Refer to the **Tutorial Data** section of the WhereScape RED Install Guide for details.

Windows Scheduler

The following describe the relevant command line arguments for Windows scheduler (--win-scheduler or -WS):

Description	Argument long name	Argument short name
Scheduler name	--sched-name <name>	-ID <name> or /ID <name>
ODBC DSN	--meta-dsn <name>	-SN <name> or /SN <name>
ODBC DSN architecture	--meta-dsn-arch <32|64>	-a <32|64> or /a <32|64>
ODBC login name	--meta-user-name <name>	-SL <name> or /SL <name>
ODBC password	--meta-password <password>	-SP <password> or /SP <password>
Metadata schema/database	--meta-database <name>	-MS <name> or /MS <name>
Service login name	--service-user-name <name>	-LN <name> or /LN <name>
Internal scheduler name	--int-sched-name <name>	-IN <name> or /IN <name>
Log level	--log-level <number>	-LL <number> or /LL <name>
Polling interval	--poll-interval <number>	-PI <number> or /PI <name>
Work directory	--work-dir <path name>	-WD <path name> or /WD <path name>
Log file name	--log-file-name <name>	-WL <path name> or /WL <path name>
TNS name	--tns-name <name>	-TN <name> or /TN <name>
Service password	--service-password <password>	-LP <password> or /LP <password>

Note:

Refer to the **Installing a Windows Scheduler** section of the WhereScape RED Install Guide for details.

Deploy Application

The following describe the relevant command line arguments for deploy application (--deploy-app or -AL):

Description	Argument long name	Argument short name
Application ID file name	--app-id-file-name <path name>	-AF <file name> or /AF <name>
Application parameter file name	--app-param-file-name <path name>	-PF <file name> or /PF <name>

Description	Argument long name	Argument short name
Log file name	--log-file-name <path name>	-LF <path name> or /LF <path name>

Note:

Refer to the **Creating and Loading Applications from the Command Line** for details.

Copyright

Due to continued product development this information may change without notice. WhereScape, Inc. does not warrant that this document is error-free.

Trademark

WhereScape, Inc.
10801 N Mopac Expressway
Building 1, Suite 100
Austin, Texas 78759

www.wherescape.com