



Data Exchange 3

The easy way to turn your data into valuable *information*.

VERSION 3.0

May-2019

TABLE OF CONTENTS

- SECTION 1. Overview5**
- 1.1. MAIN GOALS5
- 1.2. How it Works.....5
 - Data Exchange 3 Engine6
- 1.3. Main Definitions7
- 1.4. Platforms.....8
 - 1.4.1 Database Systems.....8
 - 1.4.2 Operating Systems8
- 1.5. Minimum System Requirements.....8
- 1.6. Notes for Users of Previous Versions of Data Exchange II, 2.0, and 2.5.....9
- SECTION 2. Data Exchange 3 Installation.....10**
- 2.1. Data Exchange Application Features 10
 - 2.1.1 Installation Features to Install 10
 - A. Management..... 10
 - B. Triggers..... 10
 - C. Service 10
 - D. Remote OPC Service 10
 - E. MatrikonOPC Simulation Server..... 10
 - F. OPC Shared..... 10
 - 2.1.2 Installation Steps 11
- 2.2. Firewall Settings 14
- 2.3. Data Exchange 3 Registration 15
 - 2.3.1 Automatic Authorization..... 16
 - 2.3.2 Manual Authorization 17
- SECTION 3. Data Exchange 3 Manager22**
- 3.1. Overview 22
- 3.2. Engine..... 27
- 3.3. Engine Configuration Settings 30
 - A. Process Priority 30
 - B. Master Timer Interval..... 30
 - C. Maximum Threads 30
 - D. Max Queued Instanced per Operation 31
 - E. Trace Level 31
 - F. Local Configurations Folder..... 31
 - G. Allow Remote Connections..... 31
 - H. Authorized Roles 32
 - I. Port 31
 - J. Web Server 31
 - K. Link 32
- 3.4. Remote OPC Service Configuration Settings..... 33
- 3.5. Engine Control (Engine Menu Item) 34
 - A. Connect..... 34
 - B. New Configuration 36
 - C. Add Existing Configuration..... 36
 - D. Restart All Configurations 37
 - E. Start All Configurations 37
 - F. Stop All Configurations 37
 - G. Recent Connections 37
 - H. Settings **Error! Bookmark not defined.**
 - I. Exit..... **Error! Bookmark not defined.**
- 3.6. Engine Configuration Control (Tree Menu - Right Click) 37
 - A. Edit..... 38
 - B. Rename 38
 - C. Validate 38

- D. Upload from Engine 40
- E. Restart/Start/Stop 40
- F. Remove..... 41
- 3.7. Operations Tab 43
- 3.8. Events Tab..... 44
 - A. Filtering Events 45
 - B. Clear All Events 45
 - C. Max Entries 45
- 3.9. Configurations..... **Error! Bookmark not defined.**
 - 3.9.1 Configuration Control..... **Error! Bookmark not defined.**
 - A. Edit..... **Error! Bookmark not defined.**
 - B. Rename **Error! Bookmark not defined.**
 - C. Compare to Local **Error! Bookmark not defined.**
 - D. Upload from Engine **Error! Bookmark not defined.**
 - E. Restart/Start/Stop **Error! Bookmark not defined.**
 - F. Remove..... **Error! Bookmark not defined.**
 - 3.9.2 Offline Folder Configuration Control **Error! Bookmark not defined.**
 - A. Connect to Engine **Error! Bookmark not defined.**
 - B. New Configuration..... **Error! Bookmark not defined.**
 - C. Add Existing Configuration..... **Error! Bookmark not defined.**
 - D. Open Folder..... **Error! Bookmark not defined.**
 - E. Delete Folder **Error! Bookmark not defined.**
 - F. Refresh **Error! Bookmark not defined.**
 - 3.9.3 Offline File Configuration Control **Error! Bookmark not defined.**
 - A. Edit..... **Error! Bookmark not defined.**
 - B. Rename **Error! Bookmark not defined.**
 - C. Validate **Error! Bookmark not defined.**
 - D. Compare to Engine..... **Error! Bookmark not defined.**
 - E. Download to Engine..... **Error! Bookmark not defined.**
 - F. Delete..... **Error! Bookmark not defined.**
- 3.10. Operation Control 48
 - 3.10.1 Reset Counters..... 48
 - 3.10.2 View Events 48
 - 3.10.3 Verbose Mode 49
 - 3.10.4 Active 49
 - 3.10.5 Start Operation 49
- SECTION 4. Data Exchange Editor 50**
 - 4.1. Engine Connection 55
 - 4.2. File Menu 56
 - 4.3. Import Previous Versions of Data Exchange Configurations 59
 - 4.4. Tree Menu 60
 - 4.5. New Configuration Wizard 61
 - 4.5.1 Connect to Engine 62
 - 4.5.2 Add Database 62
 - 4.5.3 Add OPC Server 64
 - 4.6. Connections - Databases 66
 - 4.6.1 Build the Database Connection 66
 - A. Provider..... 67
 - B. Connection..... 68
 - C. Read Schema 70
 - D. Connection Timeout..... 70
 - E. Show Links..... 70
 - F. Test..... 70
 - G. Database Notification Triggers 70
 - H. Database CLR Integration 70
 - 4.6.2 Database Configuration Menu **Error! Bookmark not defined.**
 - A. Refresh **Error! Bookmark not defined.**

- B. Delete..... **Error! Bookmark not defined.**
- C. Revert **Error! Bookmark not defined.**
- D. Find..... **Error! Bookmark not defined.**
- E. Replace..... **Error! Bookmark not defined.**
- 4.7. Connections OPC Servers..... 71
 - 4.7.1 Add an OPC Server 71
 - 4.7.2 Optimize OPC Groups 73
 - 4.7.3 Show Links..... 73
 - 4.7.4 OPC Server Configuration Menu **Error! Bookmark not defined.**
 - A. Delete..... **Error! Bookmark not defined.**
 - B. Revert **Error! Bookmark not defined.**
 - C. Find..... **Error! Bookmark not defined.**
 - D. Replace..... **Error! Bookmark not defined.**
- 4.8. Connections - Web Services 74
- 4.9. Connections - Heads UP 74
 - 4.9.1 Heads Up Connection..... 74
 - 4.9.2 Heads Up Operation 74
- 4.10. Operations 76
 - 4.10.1 Folders..... 77
 - 4.10.2 Operation Right-click **Error! Bookmark not defined.**
 - A. Add Transfer 79
 - B. Enabled..... 79
 - C. Active on Start 79
- 4.11. Variables..... 79
 - 4.11.1 Default (Automatic) Variables 80
 - 4.11.2 Default (Automatic) Legacy Variables From Previous Versions of Data Exchange 82
 - 4.11.3 Creating Local /Global/Folder Variables 82
 - 4.11.4 Local Running Variables 84
- 4.12. General Operation Settings 87
 - A. Operation Execution Timeout 87
 - B. Parallel Execution 87
 - C. Database Buffering 88
- 4.13. Trigger Types..... 89
 - 4.13.1 Schedule 89
 - 4.13.2 OPC Data Change..... 90
 - 4.13.3 Database Polling..... 91
 - 4.13.4 Database Notification..... 92
 - 4.13.5 External..... 94
- 4.14. General Transfer Types Descriptions 99
 - 4.14.1 OPC Transfers 101
 - 4.14.2 Simple Query Transfers 101
 - 4.14.3 Custom Query Transfers 102
 - 4.14.4 Control Transfers 102
 - 4.14.5 Miscellaneous Transfers 103
- 4.15. General Transfer Settings..... 104
 - 4.15.1 Name 104
 - 4.15.2 On Error 104
 - 4.15.3 OPC Connection 104
 - 4.15.4 Read from Cache..... 104
 - 4.15.5 Read from Device 104
 - 4.15.6 Use NULL for Bad Quality 104
 - 4.15.7 Browse 105
 - 4.15.8 Validate Tag..... 105
- 4.16. Transfers..... 106
 - 4.16.1 Name 106
 - 4.16.2 On Error 106
 - 4.16.3 OPC Connection 106
 - A. Read from Cache..... 106

B.	Read from Device	106
C.	Use NULL for Bad Quality	107
4.16.4	Format.....	107
4.16.5	Preview	107
4.17.	OPC Transfers.....	109
4.17.1	Read Variables from OPC	109
4.17.2	Write OPC Receipt	111
4.17.3	Write Variables to OPC.....	112
4.18.	Simple Query Transfers.....	113
4.18.1	Read from OPC to Database.....	113
A.	SQL Action.....	113
B.	SQL Query.....	114
4.18.2	Read Variables from DB	114
A.	Error on no data	115
B.	Filter Columns.....	115
4.18.3	Write to OPC from Database	117
4.18.4	Write Variables to DB	118
4.19.	Custom Query Transfers	120
4.19.1	Execute Custom SQL (Advanced Variable DB)	120
4.19.2	Custom Read from OPC to Database	122
4.19.3	Write to OPC from Database	125
4.20.	Control Transfers	129
4.20.1	Pause.....	129
4.20.2	Script.....	129
A.	Using.....	51
B.	References.....	134
C.	Browse.....	134
4.20.3	Start Operation	135
4.20.4	Test OPC Condition.....	135
4.21.	Miscellaneous Transfers.....	137
4.21.1	Call Web Service	Error! Bookmark not defined.
4.22.	Backups	139
4.23.	Exporting Tags & Operations	140
4.23.1	Tag Data	140
4.23.2	Operation Data	141
SECTION 5.	Performance Monitoring.....	142

SECTION 1. OVERVIEW

1.1. MAIN GOALS

Data Exchange 3 (abbreviated DXC3) provides simple and intuitive tools to design complex communication between databases and industrial devices. With no programming effort or extensive knowledge of databases, Data Exchange gathers process data for future analysis and warehousing. Data Exchange 3 moves data between an industrial device, such as a PLC, and a database. This occurs in a variety of ways based on the process needs and complexity.

1.2. HOW IT WORKS

Data Exchange 3 communicates with industrial devices using OPC. OPC is an industry standard communication interface to industrial devices. OPC servers are identified by a unique OPC server name. For example:

Software	OPC Server Name
RSLinx (local computer)	RSLinx OPC Server
RSLinx (Remote computer)	RSLinx Remote OPC Server
Simatic NET	OPC.SimaticNET
WinAC	OPCServer.WinAC
WinCC	OPCServer.WinCC

Consult the documentation of the specific OPC server for more information.

Data Exchange 3 communicates with databases using OLE DB. OLE DB is an industry standard communication interface to databases. OLE DB providers for most popular databases and comes with Microsoft Windows.

A diagram of a system that connects several different databases with a number of PLCs is shown. OPC servers and databases can be located on the same machine as Data Exchange 3 or they can reside on other machines on the network.

The user creates Configurations using the Data Exchange 3 Editor. The Configurations define the database connections, OPC connections, and the data transfers. The Data Exchange 3 Engine can run multiple Configurations at once, and each Configuration can be independently controlled.

Data Exchange 3 consists of the following components:

- Data Exchange 3 Engine (Service)
- Data Exchange 3 Remote OPC Service (Service)
- Data Exchange 3 Editor (Application)
- Data Exchange 3 Manager (Application)
- Data Exchange 3 Web Manager (Optionally enabled Web Interface)
- MatrikonOPC Server for Simulation (Optional)

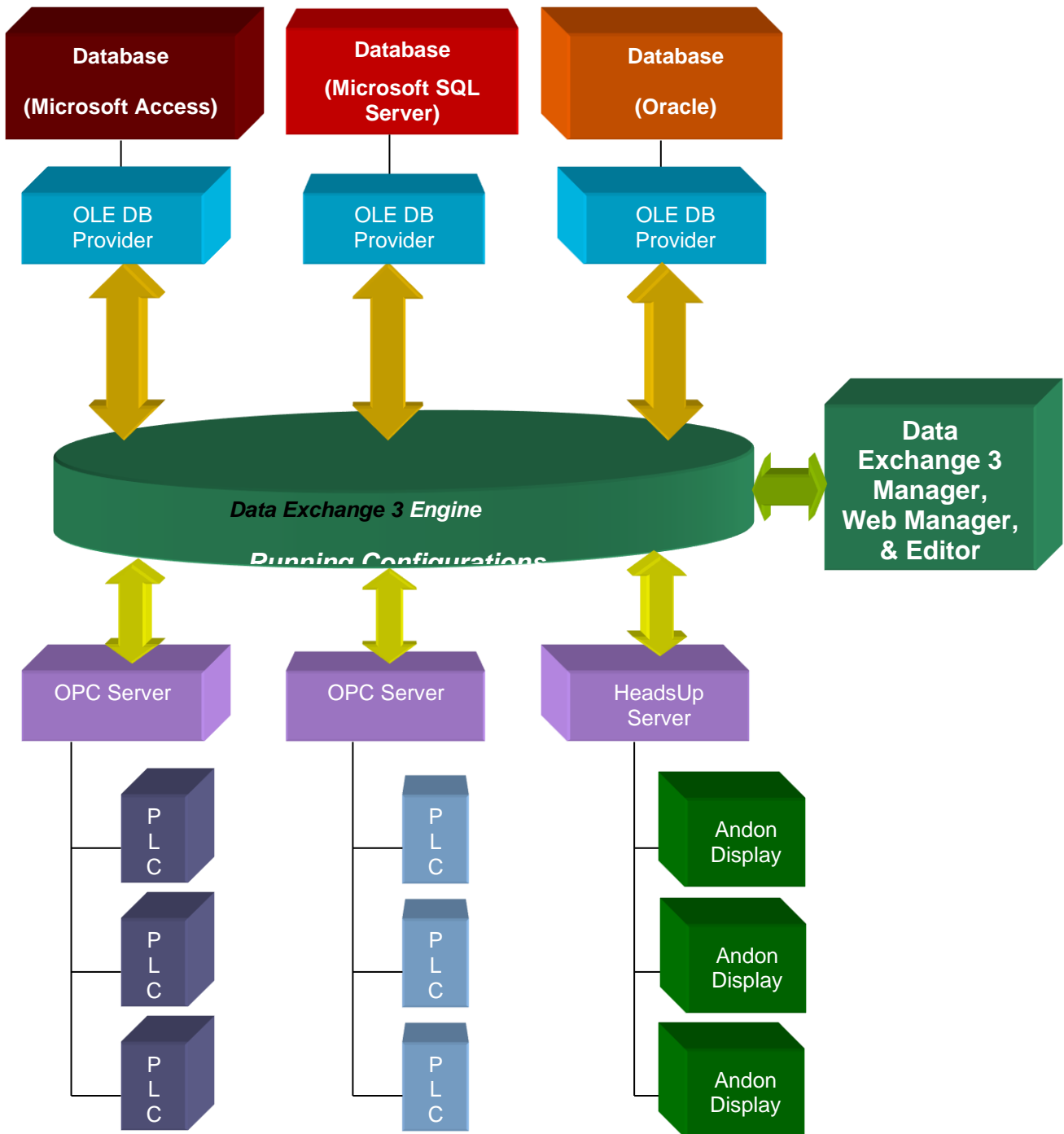
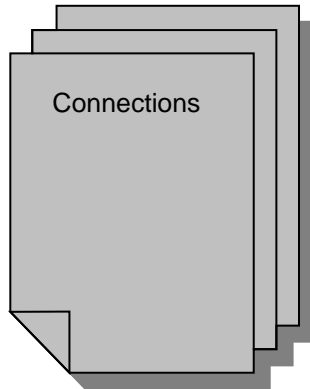


Figure Example of communication diagram.

The Data Exchange 3 Engine runs as a service on Windows platforms. The Data Exchange 3 Manager allows the user to configure and load a configuration. This tells the engine how, when, and where to move the data.

1.3. MAIN DEFINITIONS

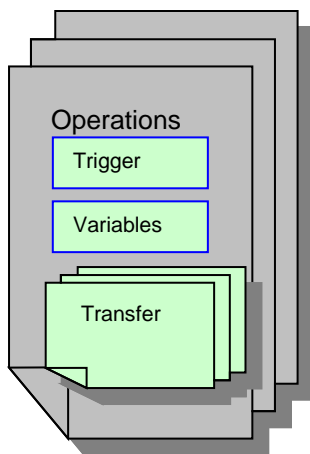
Configuration – a file that holds all the information needed to run Data Exchange 3 Engine. The user builds this file using Data Exchange 3 Editor.



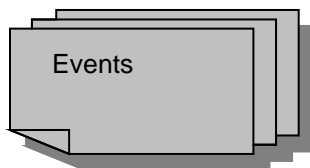
Databases – a collection of database connections. Each database connection contains all the information needed to connect to the database.

OPC Servers – a collection of OPC connections. Each OPC connection contains all the information needed to connect to the OPC server (usually a PLC or other industrial device).

Web Services – a collection of web service connections. Each web service connection contains all the information needed to connect to the web server. Web services include SOAP, JSON, Socket, and HeadsUp services.



Operations – a collection of operations. Each operation consists of an optional Trigger, and a set of Transfers. The Trigger tells Data Exchange 3 Engine WHEN to start a particular operation. Each Transfer tells the Data Exchange 3 Engine WHAT and WHERE to move. Information may be passed between transfers through the use of variables.



Events – a collection of event messages that are saved in the Windows Event Log, and viewable within the Data Exchange 3 Manager. The main purpose of Events is for diagnostics.

1.4. PLATFORMS

1.4.1 DATABASE SYSTEMS

Data Exchange 3 can connect to any database available using the OLE DB providers available on your Microsoft Windows computer. These include drivers for ODBC, Oracle and SQL Server. Database Notification Triggers and Database CLR Integration require SQL Server.

1.4.2 OPERATING SYSTEMS

The Data Exchange engine runs as a service under Windows Server 2012, Windows Server 2016, Windows 7, Windows 8 and Windows 10. It is a 32-bit application that will run on 64-bit operating systems.

1.5. MINIMUM SYSTEM REQUIREMENTS

Windows Operating System

.NET Framework 4.8

IE11, Edge or Chrome for Web Manager

Database

Database Notification Triggers SQL Server 2008 R2 or greater

Database CLR Integration SQL Server 2012 or greater

MatrikonOPC

Microsoft .NET Framework 1.1 with Service Pack 1

1.6. NOTES FOR USERS OF PREVIOUS VERSIONS OF DATA EXCHANGE II, 2.0, AND 2.5

- ✓ The Data Exchange 3 Manager and Editor can now be installed on different computers than the Data Exchange 3 Engine. Data Exchange 3 Remote OPC Service greatly simplifies OPC communications between computers.
- ✓ A new Data Exchange 3 Web Manager can perform most of the functions of the Data Exchange 3 Manager.
- ✓ Data Exchange II and 2.5 configurations can be imported into Data Exchange 3 by simply opening the configuration.
 - Disabled Operations should be enabled before importing and then disabled after importing.
- ✓ Navigation of the Operations and Transfers is now done through the Tree on the left of the editor. Individual saving of Operation and Transfer settings is no longer needed. The Configuration must still be saved after changes.
- ✓ Copy and Paste and Drag and Drop of Operations, Transfers, and Schedules is supported within or between Configurations.
- ✓ Backup Configurations are stored in the same file as the Configuration and can be accessed from the File->Manage Backups menu item.
- ✓ Configurations can be compared to Back up versions or the current engine configuration to see what has changed.
- ✓ When using OPC tags in Database Transfers, single quotes around the OPC tags are no longer needed because Data Exchange 3 will add the single quotes as needed.
- ✓ The Data Exchange 3 Engine will use NULL for values when appropriate.
- ✓ Data Exchange now includes the use of scoped variables.
- ✓ New Trigger types, Transfer types, and other features.
- ✓ Configurations can be validated before downloading to the engine.
- ✓ Enhanced Search, Search and Replace, and new Revert feature.
- ✓ Because Data Exchange 3 works with the MatrikonOPC Explorer, a simple click of “View Tags” will now allow you view the real-time OPC data values of a Transfer, Operation, or all Operations.
- ✓ New Web Service features.

SECTION 2. INSTALLATION

2.1. DATA EXCHANGE APPLICATION FEATURES

There are 5 different features that can be installed. These can all be installed on the same or on different computers. The Service (i.e. Data Exchange 3 Engine) is licensed per installation by the total number of OPC tags and variables used.

2.1.1 INSTALLATION FEATURES TO INSTALL

A. *Management*

This installs Data Exchange 3 Manager and Data Exchange 3 Editor.

B. *Triggers*

This installs only the drivers required by External Triggers and Database CLR Integration.

C. *Service*

This installs the Data Exchange 3 Engine that is run as a service. It is recommended to install the **MatrikonOPC Simulation Server** with this feature. If the **Service** is installed, the **OPC Shared** should also be installed.

D. *Remote OPC Service*

This installs the Data Exchange 3 Remote OPC Service that is run as a service. It is recommended to install the **MatrikonOPC Simulation Server** with this feature. . If the **Remote OPC Service** is installed, the **OPC Shared** should also be installed.

E. *MatrikonOPC Simulation Server*

This installs the following:

MatrikonOPC Server for Simulation

MatrikonOPC Analyzer

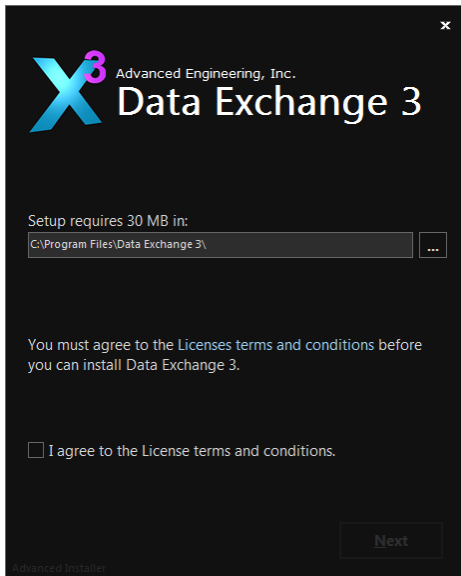
MatrikonOPC Explorer

F. *OPC Shared*

This installs shared items that are required by the **Service** and the **Remote OPC Service**. If the **Service** or the **Remote OPC Service** is installed, the **OPC Shared** should also be installed.

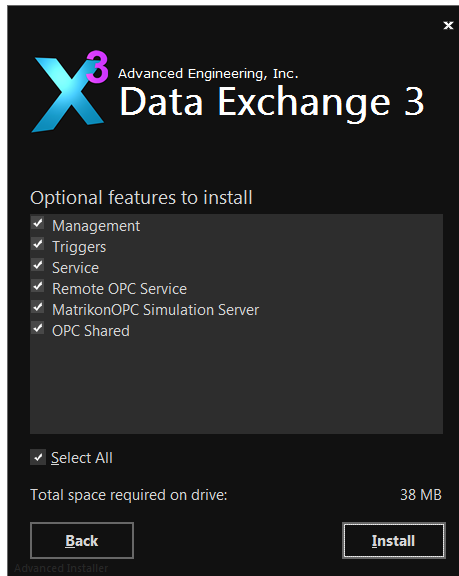
2.1.2 INSTALLATION STEPS

Run **Data Exchange 3 Setup v{version number}.exe**



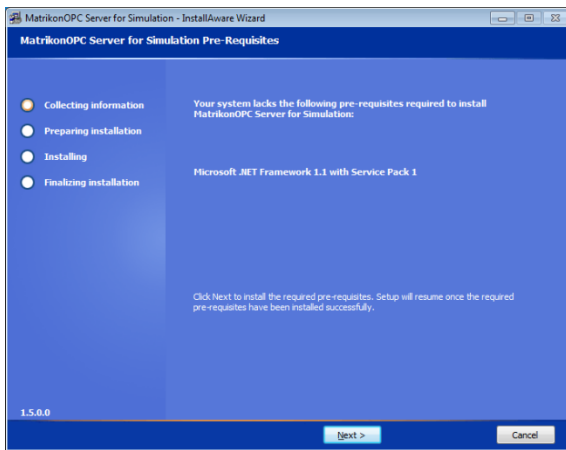
Agree with License terms

Click **Next** button



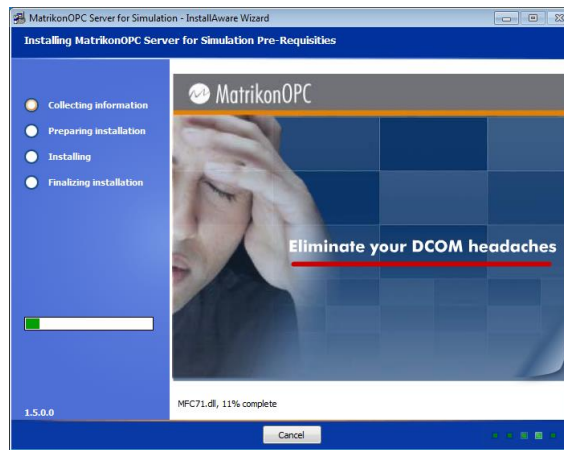
Select features to install

Click **Install** button

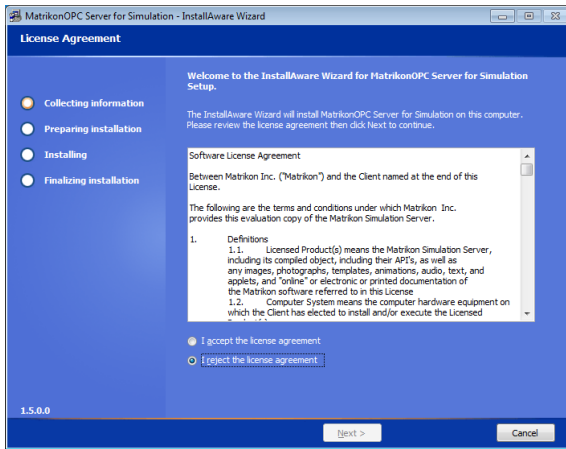


MatrikonOPC starts its installation

Click **Next** button

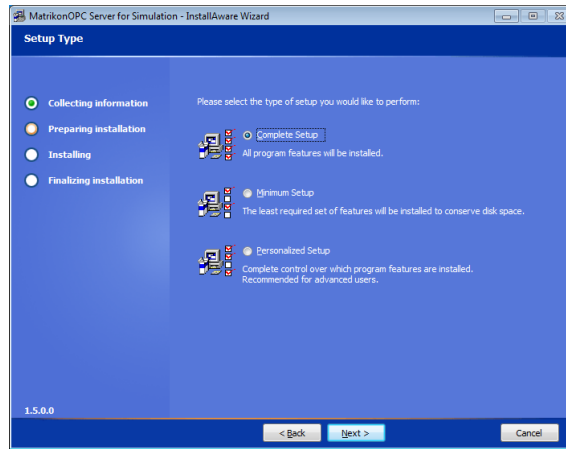


MatrikonOPC installation being processed



Accept the license agreement

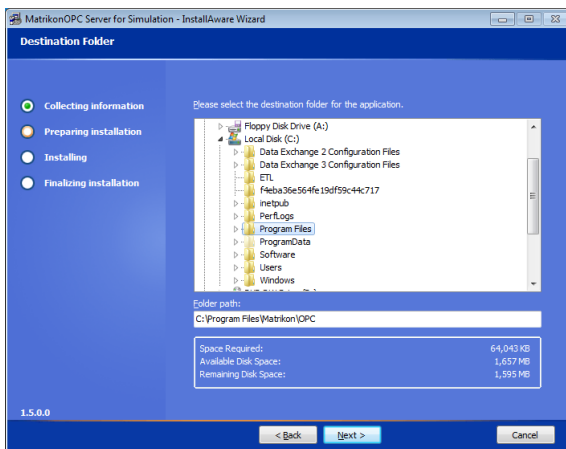
Click **Next >** button



Select the type of Setup.

Choose Complete Setup by default

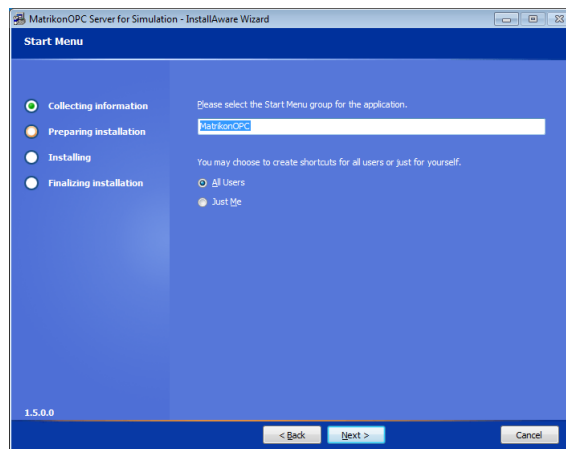
Click **Next >** button



Select destination folder

Recommend using default setting as shown

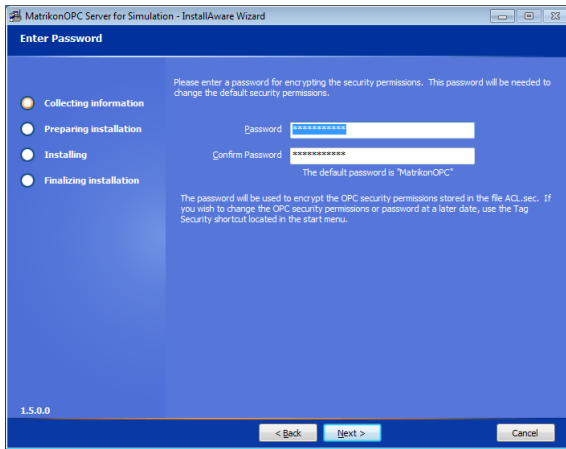
Click **Next >** button



Select Start Menu settings

Recommend using default settings as shown

Click **Next >** button

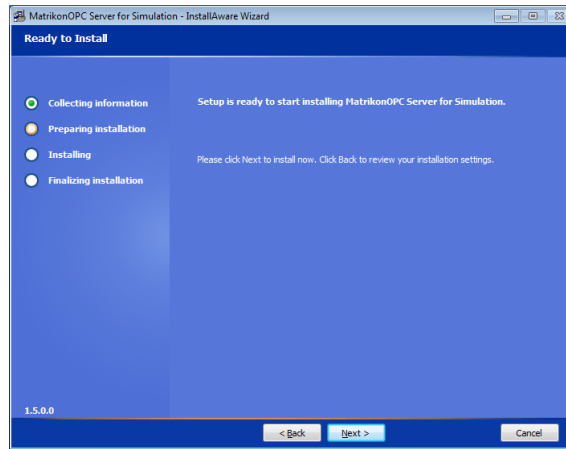


Select Password

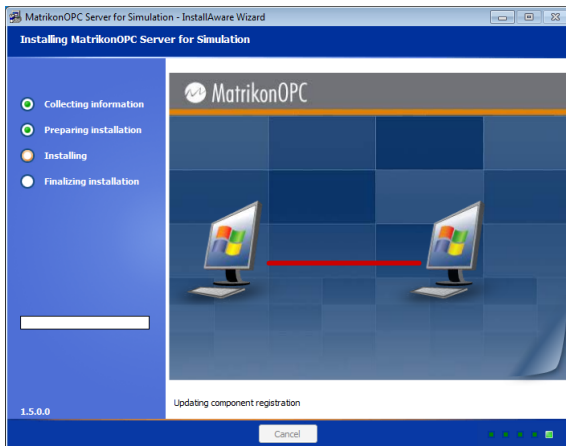
Recommend using default settings as shown

Note: The default password is **MatrikonOPC**

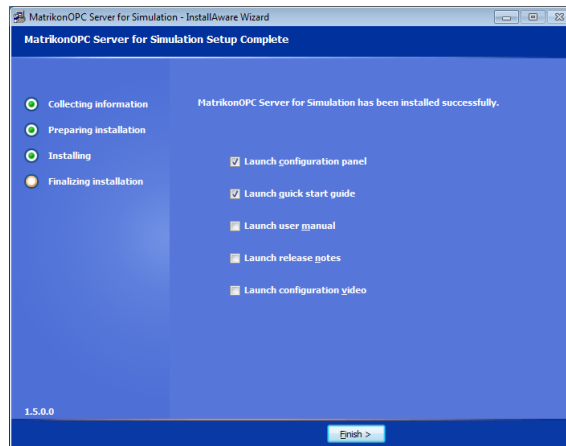
Click **Next >** button



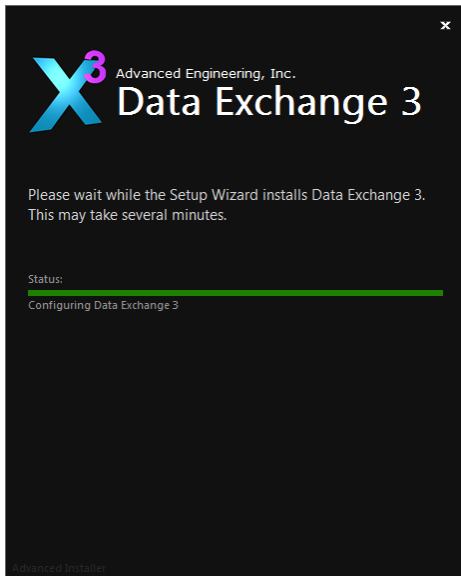
Click **Next >** button



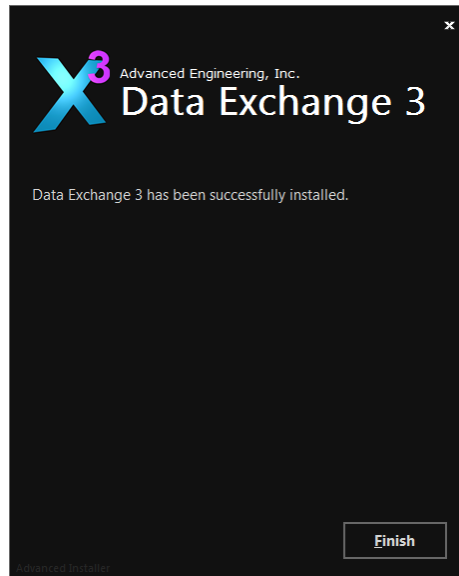
Processing Installation



Click **Finish >** button



Processing Installation



Click **Finish** button

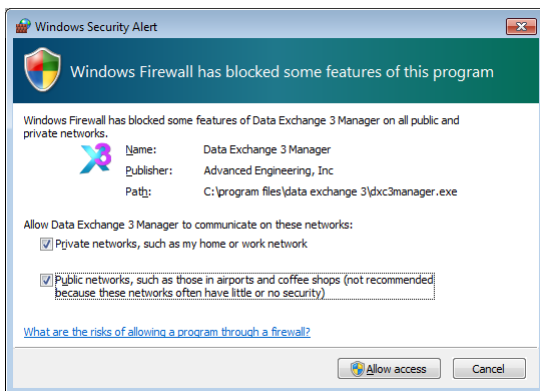
Installation is now complete. Firewall settings may require configuration

The installation log file is located at:

C:\package.log

2.2. FIREWALL SETTINGS

Starting Data Exchange 3 Manager or Manager may automatically cause Windows to request permissions to modify the firewall.



Manual Firewall Settings

<u>Process</u>	<u>Description</u>
DXC3.exe	Data Exchange 3 Engine
DXC3OPCRemote.exe	Data Exchange 3 Remote OPC
DXC3Editor.exe	Data Exchange 3 Editor
DXC3Manager.exe	Data Exchange 3 Manager

Default Ports used by Data Exchange 3

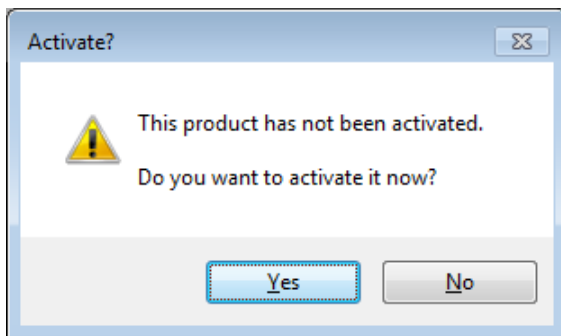
- 5130 Data Exchange 3 Engine - HTTP Web Manager and Web Service (used by Triggers)
- 5131 Remote OPC Service - Net TCP Web Service (used by the Engine)
- 5132 Remote OPC Service - HTTP Web Service
- 5139 Data Exchange 3 Engine - Management Interface (used by the Manager)

Database and OPC Servers may require firewall settings particular to the specific application.

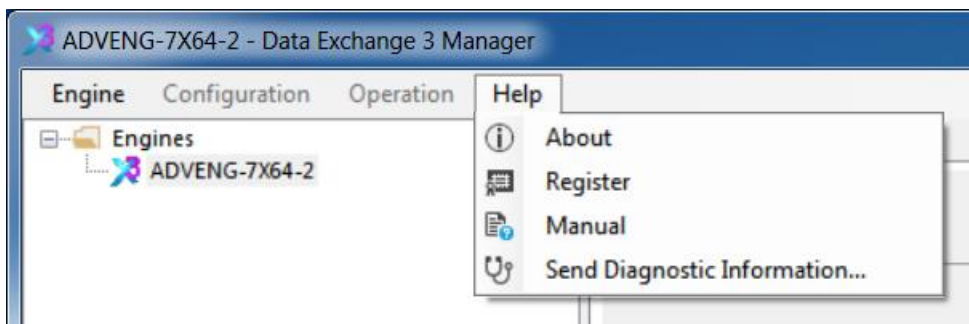
2.3. DATA EXCHANGE 3 REGISTRATION

Each running instance of the Data Exchange 3 Engine must be registered to enable full functionality. Configurations will be stopped automatically every 12 hours if the Engine is in Demo mode.

Initially Data Exchange 3 Manager will prompt for Activation

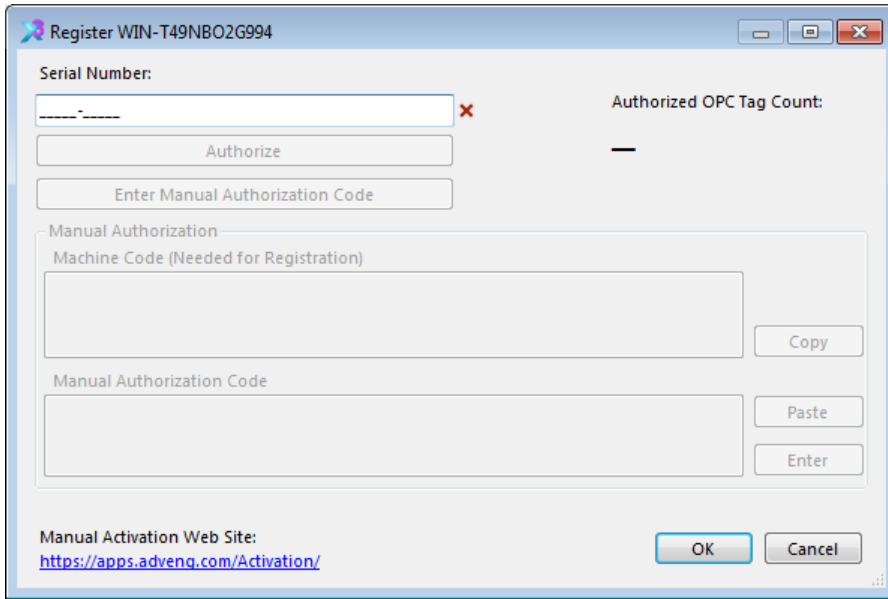


Activation can also be started by selecting **Register** in the **Help** menu.

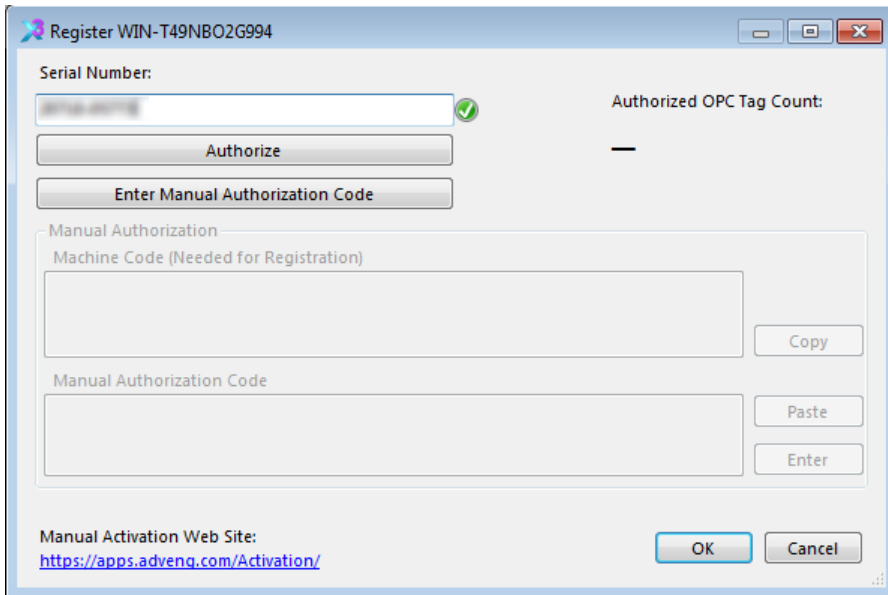


2.3.1 AUTOMATIC AUTHORIZATION

Automatic authorization will attempt to connect to the Advanced Engineering web site and authorize the serial number entered.



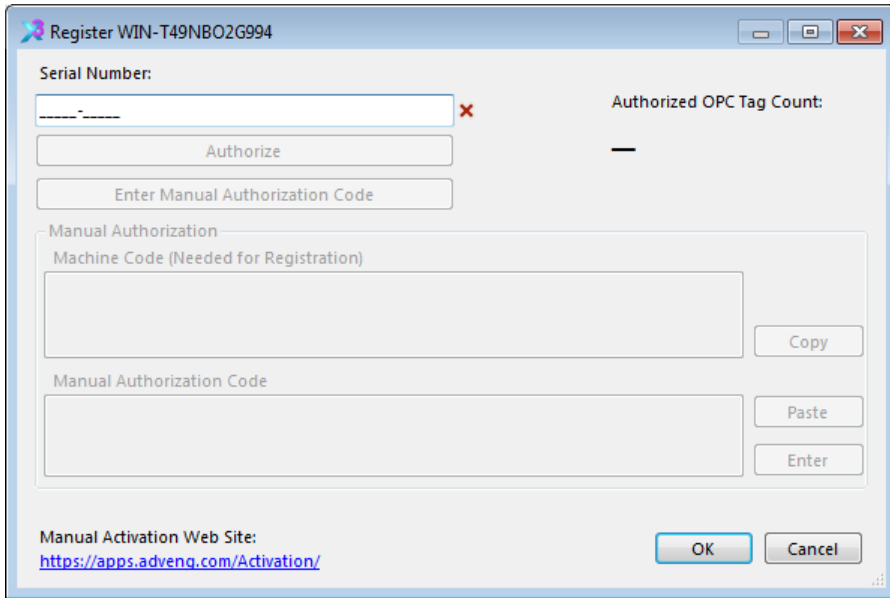
Enter Serial Number



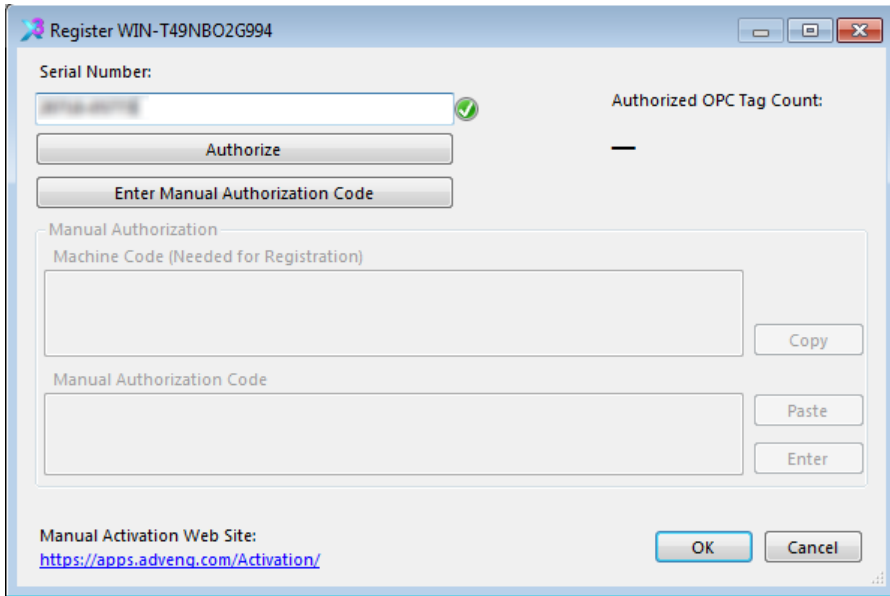
Click Authorize

If unable to Authorize, perform the **Manual Authorization**

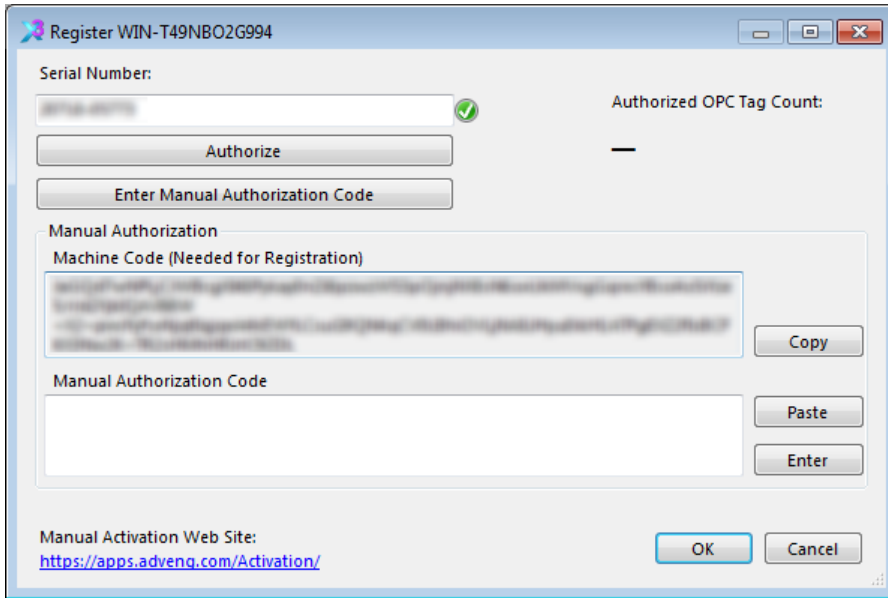
2.3.2 MANUAL AUTHORIZATION



Enter Serial Number

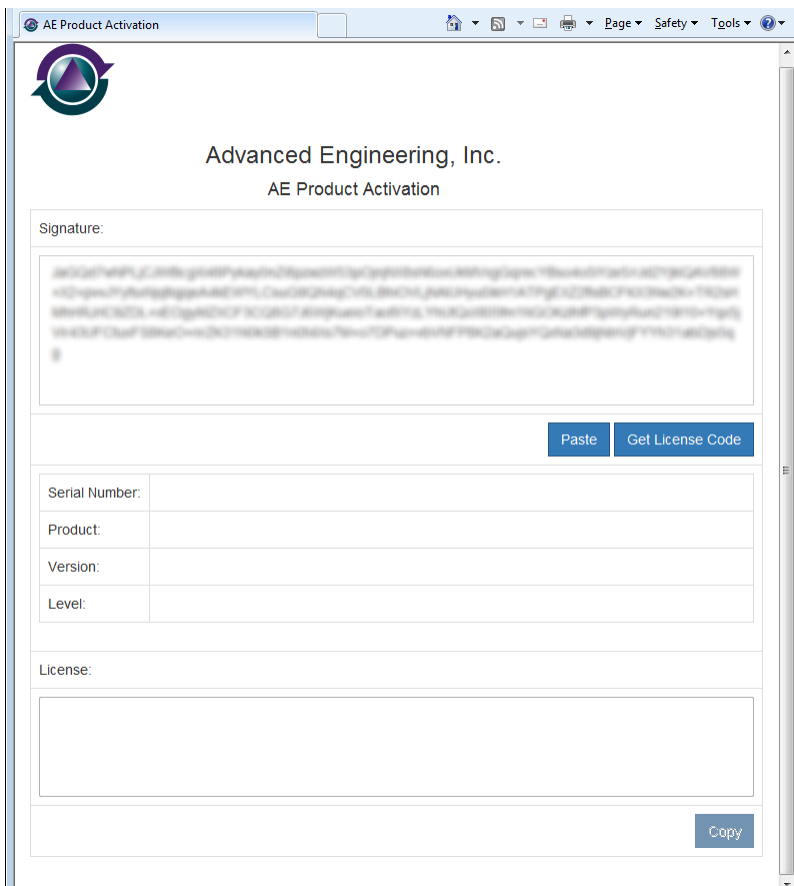


Click **Enter Manual Authorization Code** button



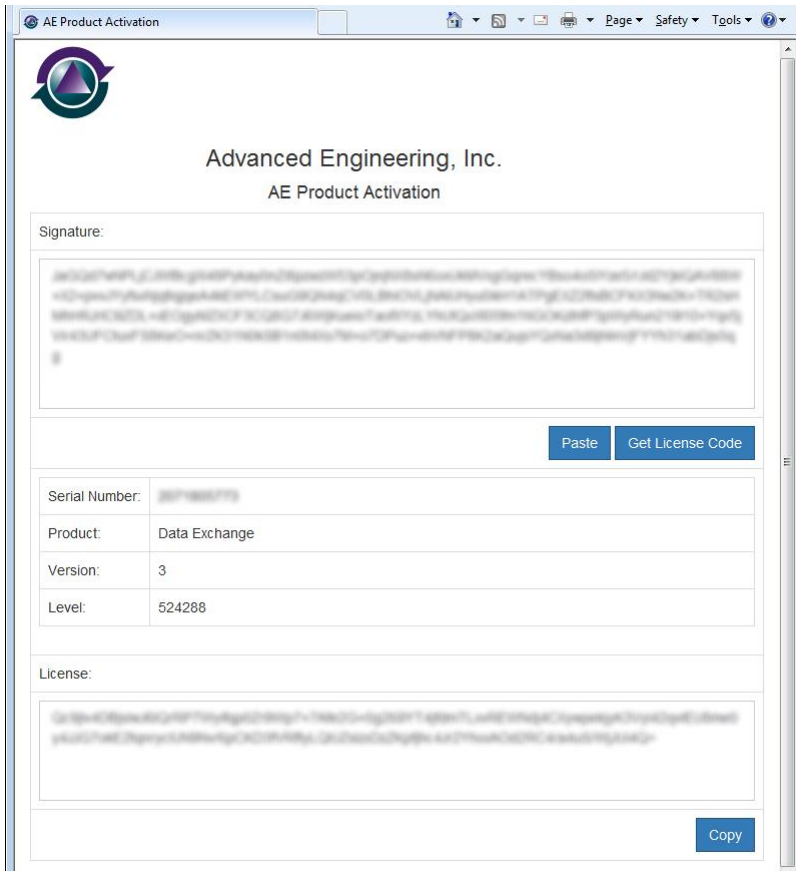
Click **Copy** button

Click <https://apps.adveng.com/Activation> link, or directly copy Machine Code

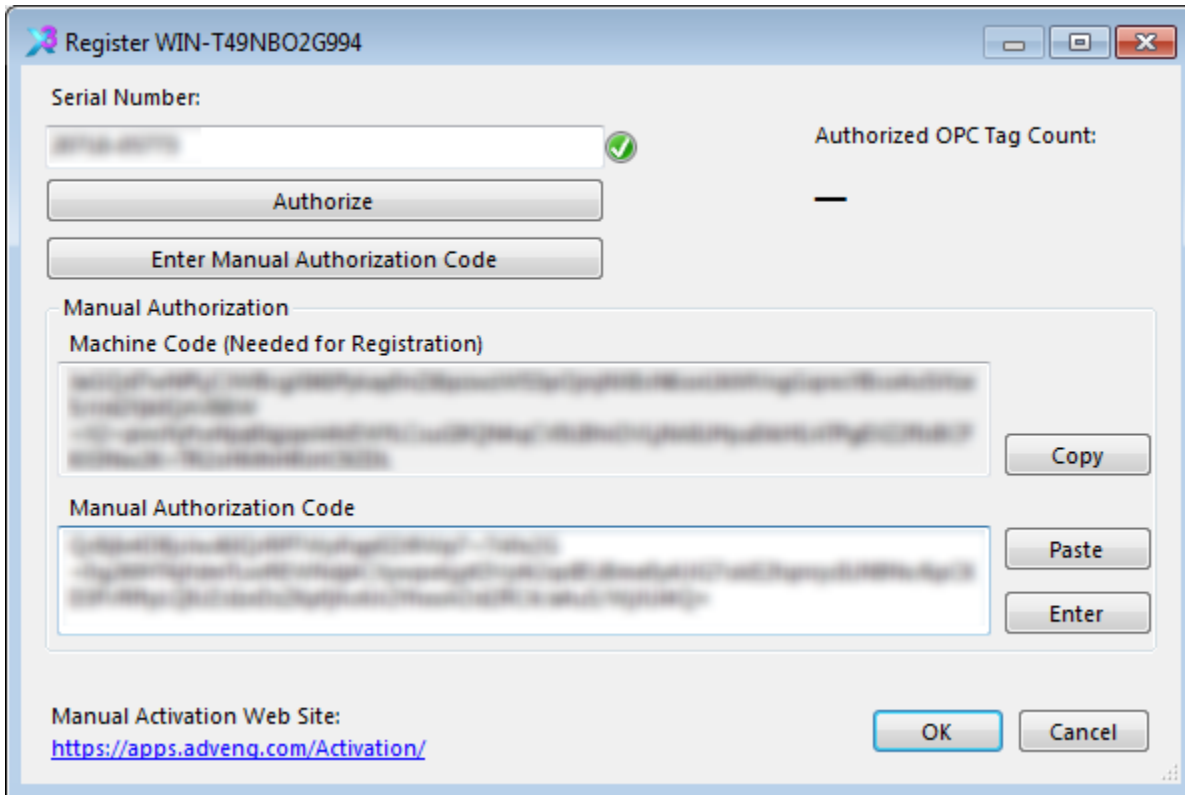


Paste **Machine Code** from Data Exchange 3 Registration into box under Signature

Click **Get License Code** button

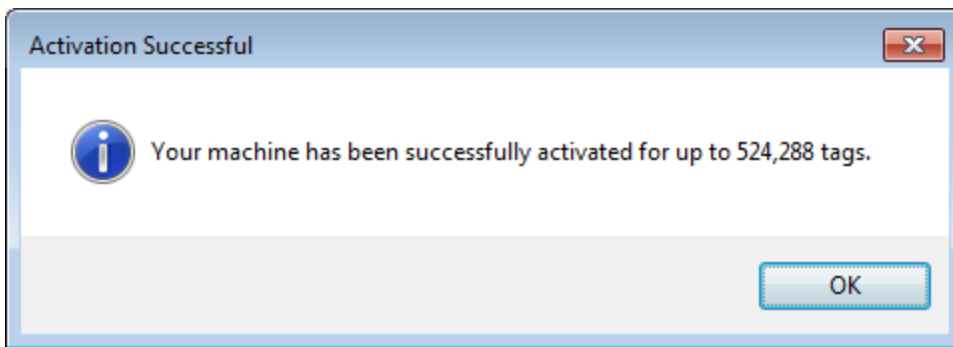


Copy text under License



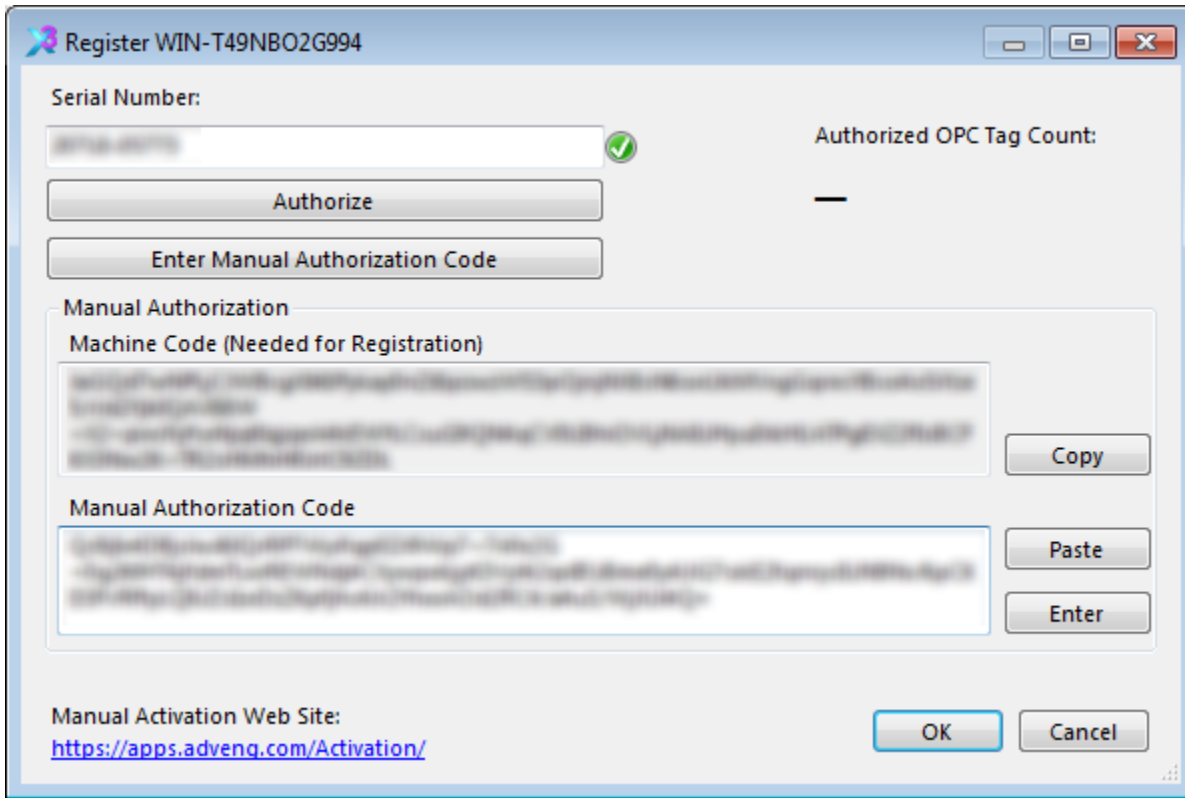
Paste License from AE Product Registration Website into **Manual Authorization Code**

Click **Enter** button

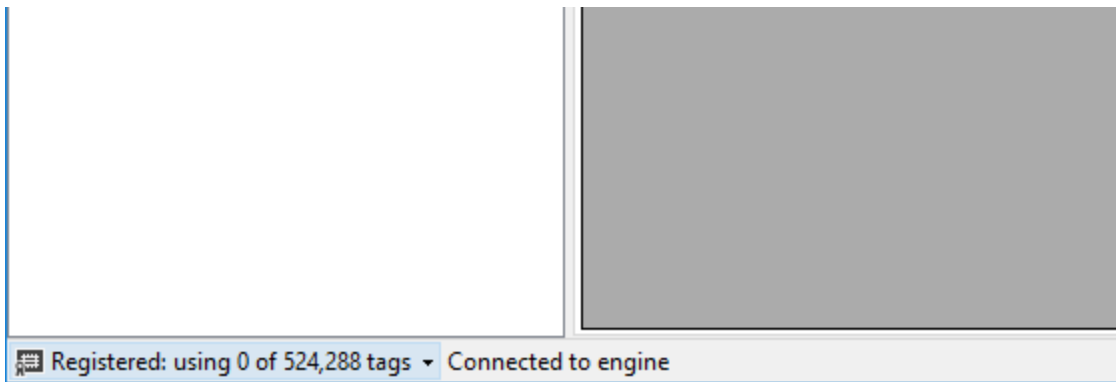


Tags are either unique OPC items or unique Variables used in a configuration. The number of allowed tags is determined by your license.

Click **OK** button



Click **OK** button

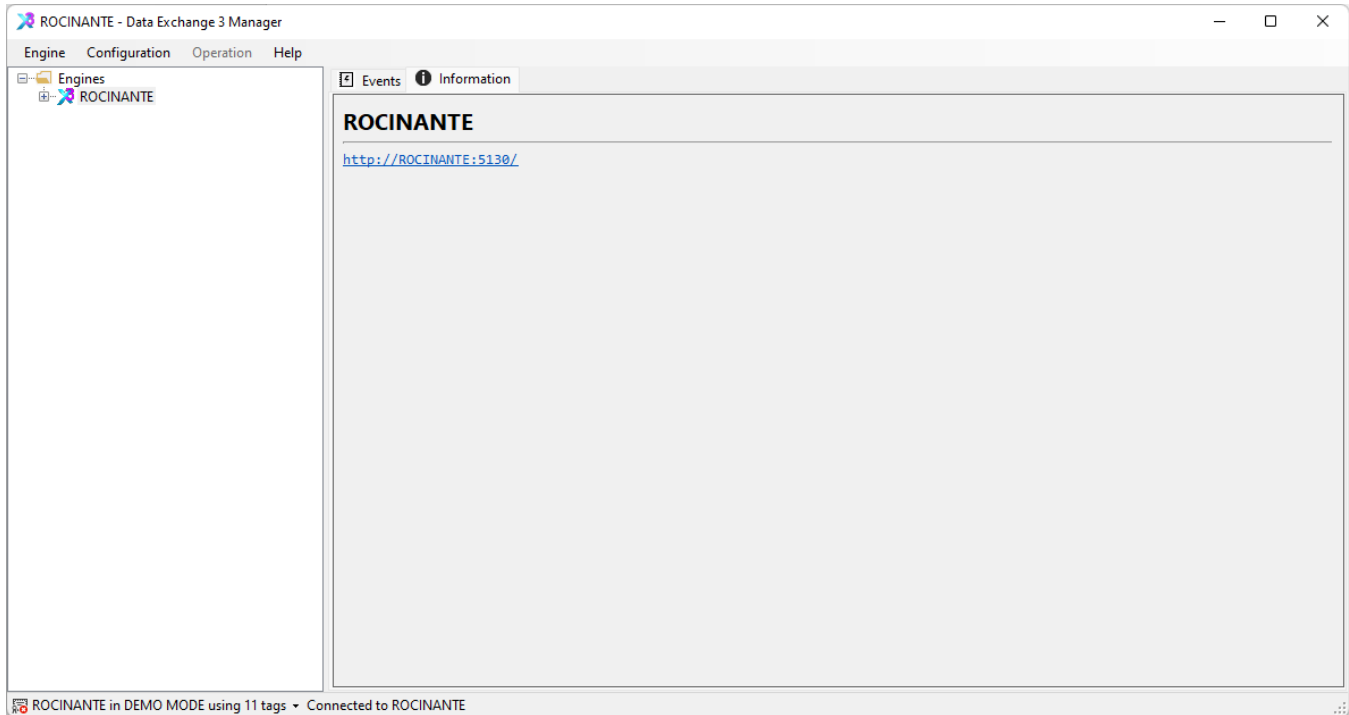


Bottom left of Data Exchange 3 Manager form displays the current registration status.

SECTION 3. MANAGER

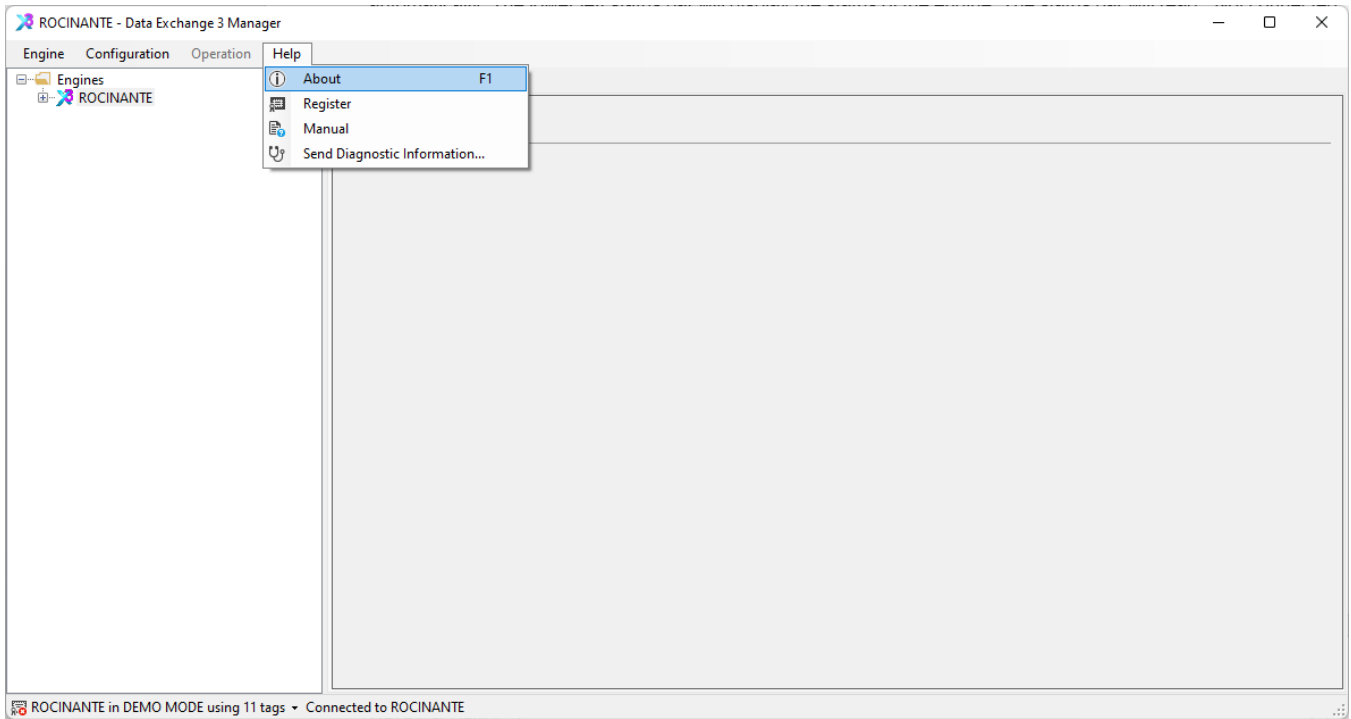
3.1. OVERVIEW

Data Exchange Manager controls the configurations that are running inside the local or remote Data Exchange engine and create new configurations.



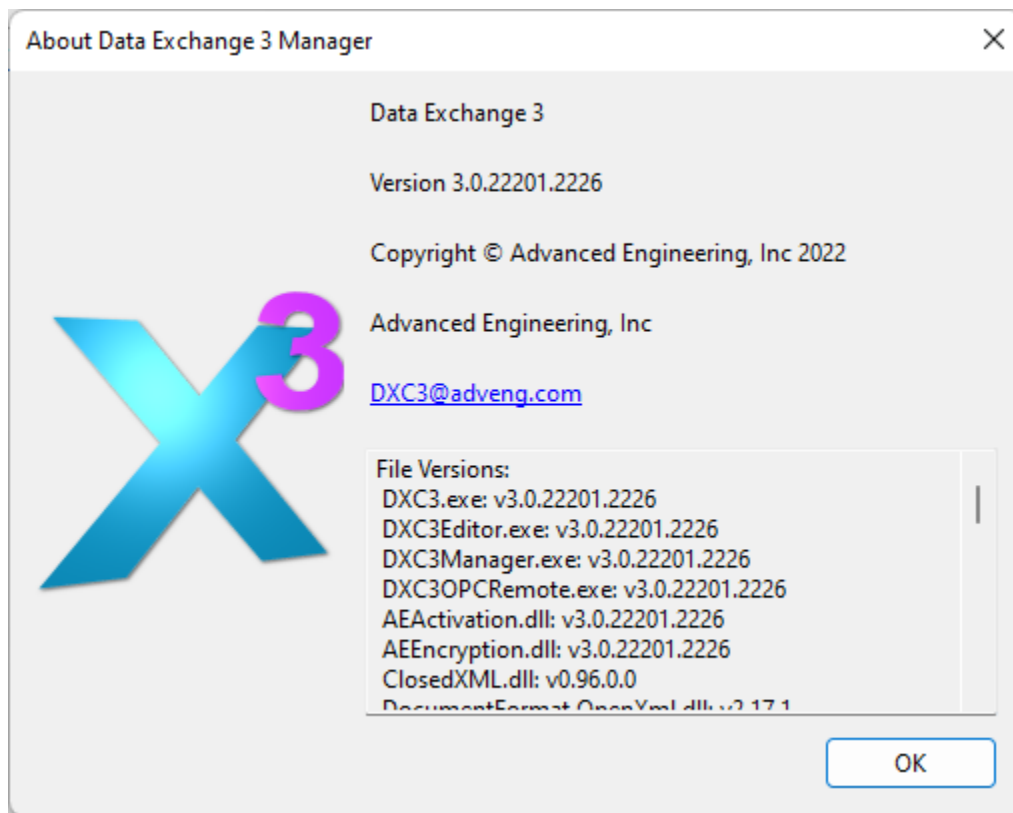
When the Data Exchange Manager starts, if the engine is running it will connect to the engine on the local computer automatically. The lower left status bar will display the status of the engine. The status bar will read, "Not connected to the Data Exchange 3 Service" if the local Data Exchange engine is stopped.

3.2. HELP



3.2.1 ABOUT

Displays the current versions of the different libraries that make up DXC3. Please contact us at DXC3@adveng.com for any DXC3 questions or feature recommendations you may have.

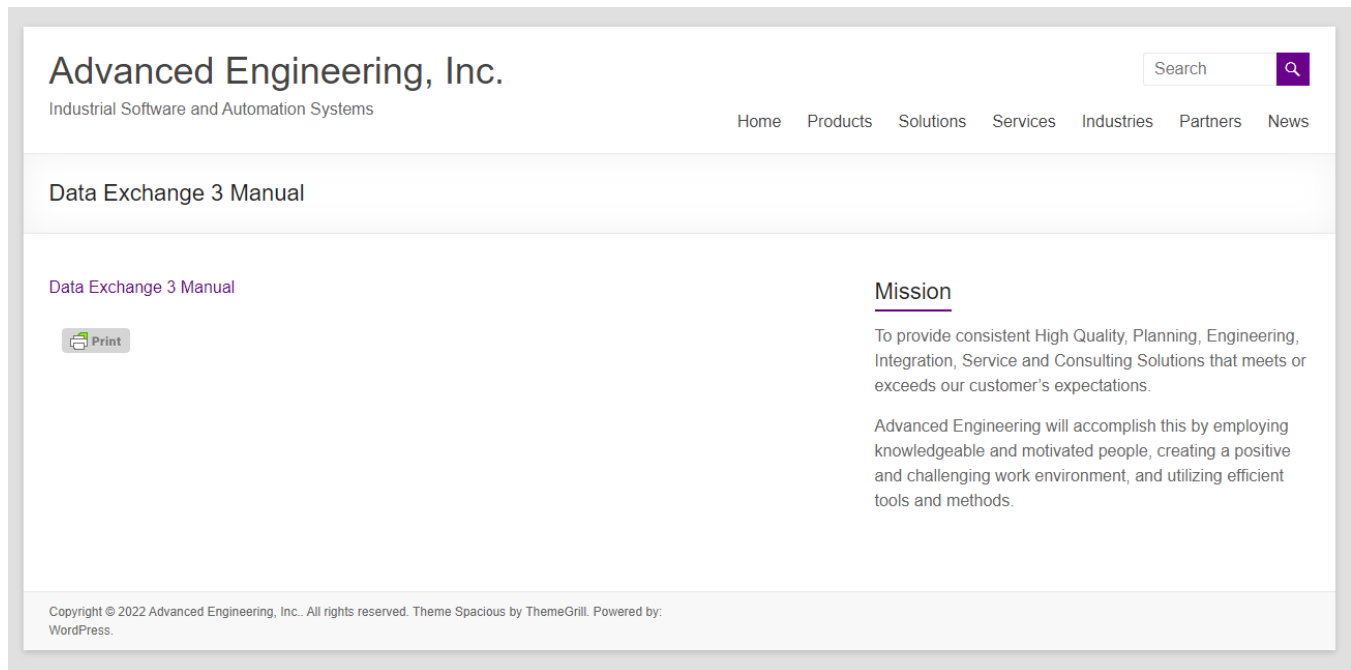


3.2.2 REGISTER

See the registration section for details.

3.2.3 MANUAL

Select Manual to download the latest version of the DXC3 documentation.

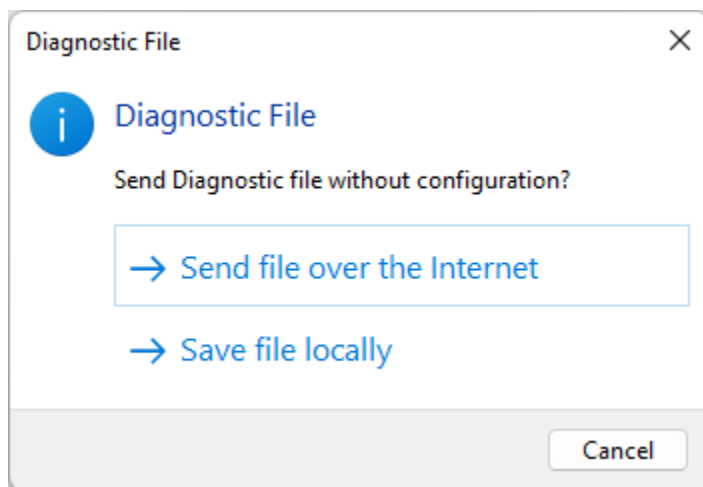


3.2.4 SEND DIAGNOSTIC INFORMATION

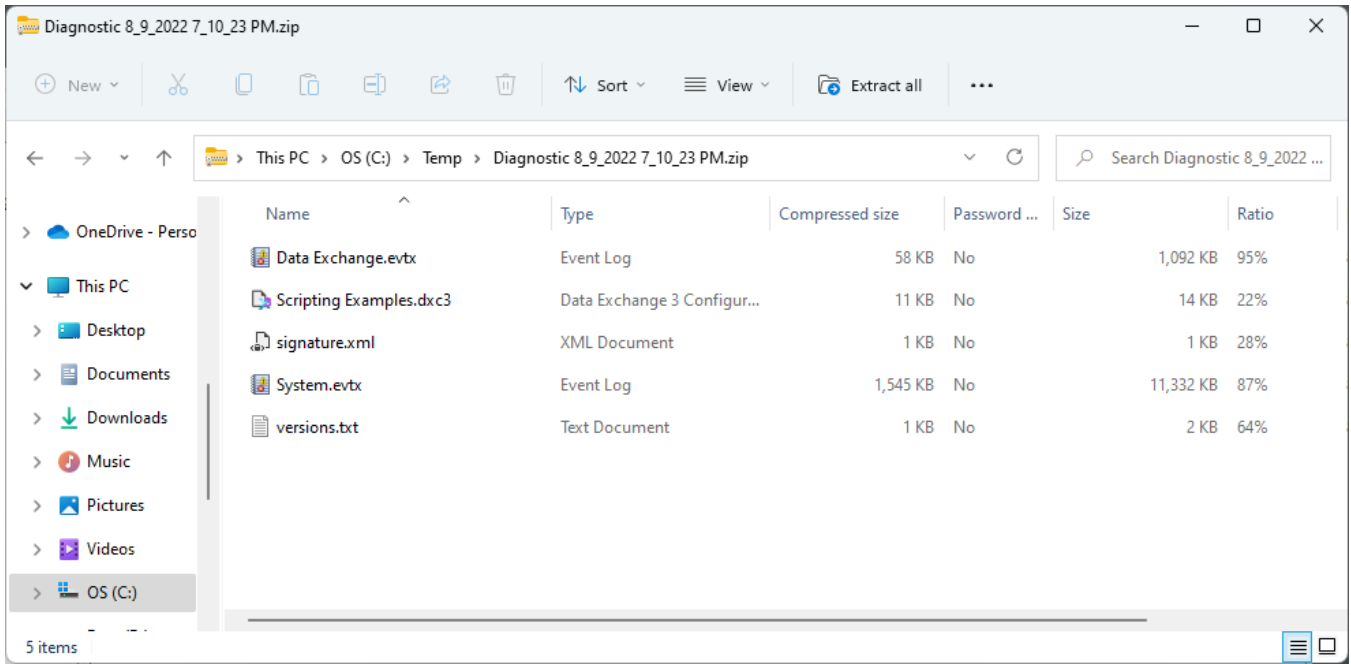
This feature allows you to email the set of files AE can use to help diagnose issues with your Data Exchange system.

Send file over the internet will automatically package the files together into a zip and upload them to the Advanced Engineering file server.

Save file locally will zip the files and allow you to store them on your computer in case the DXC3 computer does not have internet access.

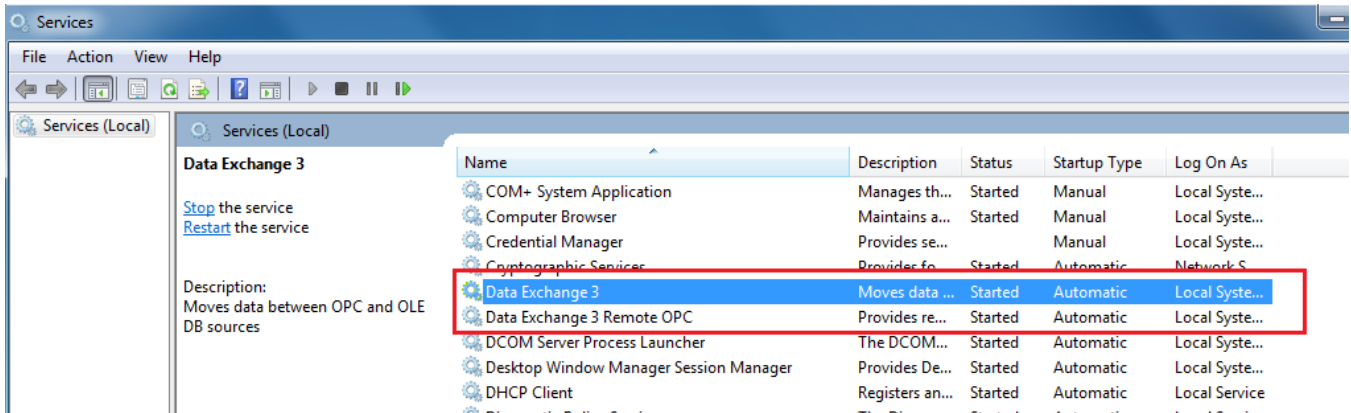


The zip will contain the selected configuration file (if any), the System event log, the Data Exchange event log, the signature of the computer (including computer name, domain, OS versions, etc.), and a list of all the versions of the Data Exchange files:

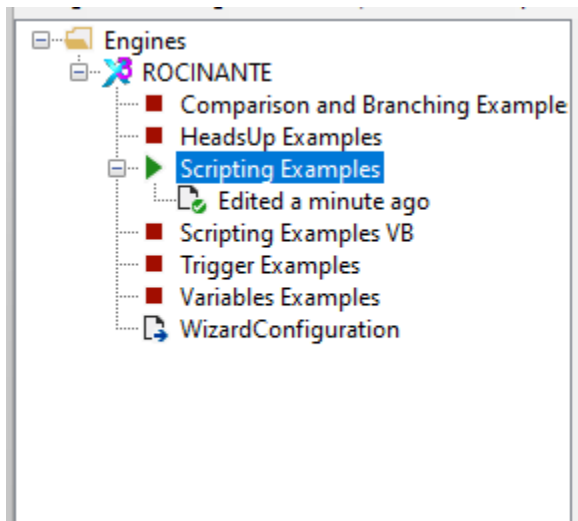
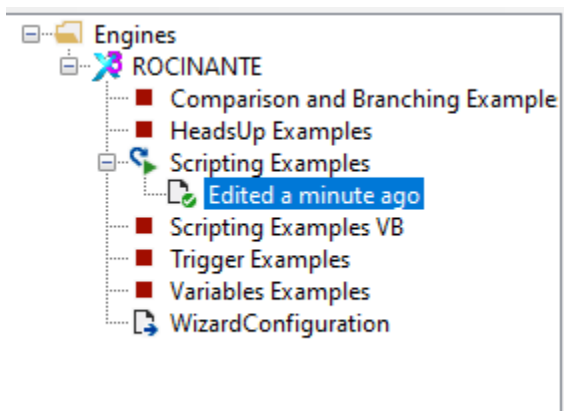
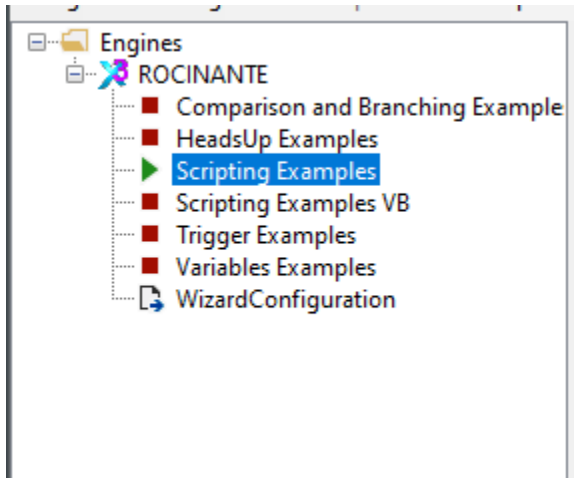


3.3. ENGINE


In most environments, Data Exchange is configured to start automatically when the server starts. Access Control Panel -> Administrative Tools -> Services to manually start and stop the engine.



The Data Exchange engines you have connected to will be shown under the “Engines” folder on the left side of the Manager. Under each engine will be a list of configurations that are either downloaded to the engine or have been created locally in your “Data Exchange Configurations” folder. If the version of the file that is running on the engine is different from the local version in your “Data Exchange Configurations” folder, the edited version of the file will show under the configuration node. Double clicking a configuration will download the file from the engine if it isn’t already in your local folder and will open it in the editor.



Open configuration ✕

 **Different Versions**

The engine configuration has changed since your last upload or download. You can open your older copy or upload the current engine configuration and overwrite your local copy

→ **Open Local**

Opens your local copy of the configuration

→ **Upload from Engine**

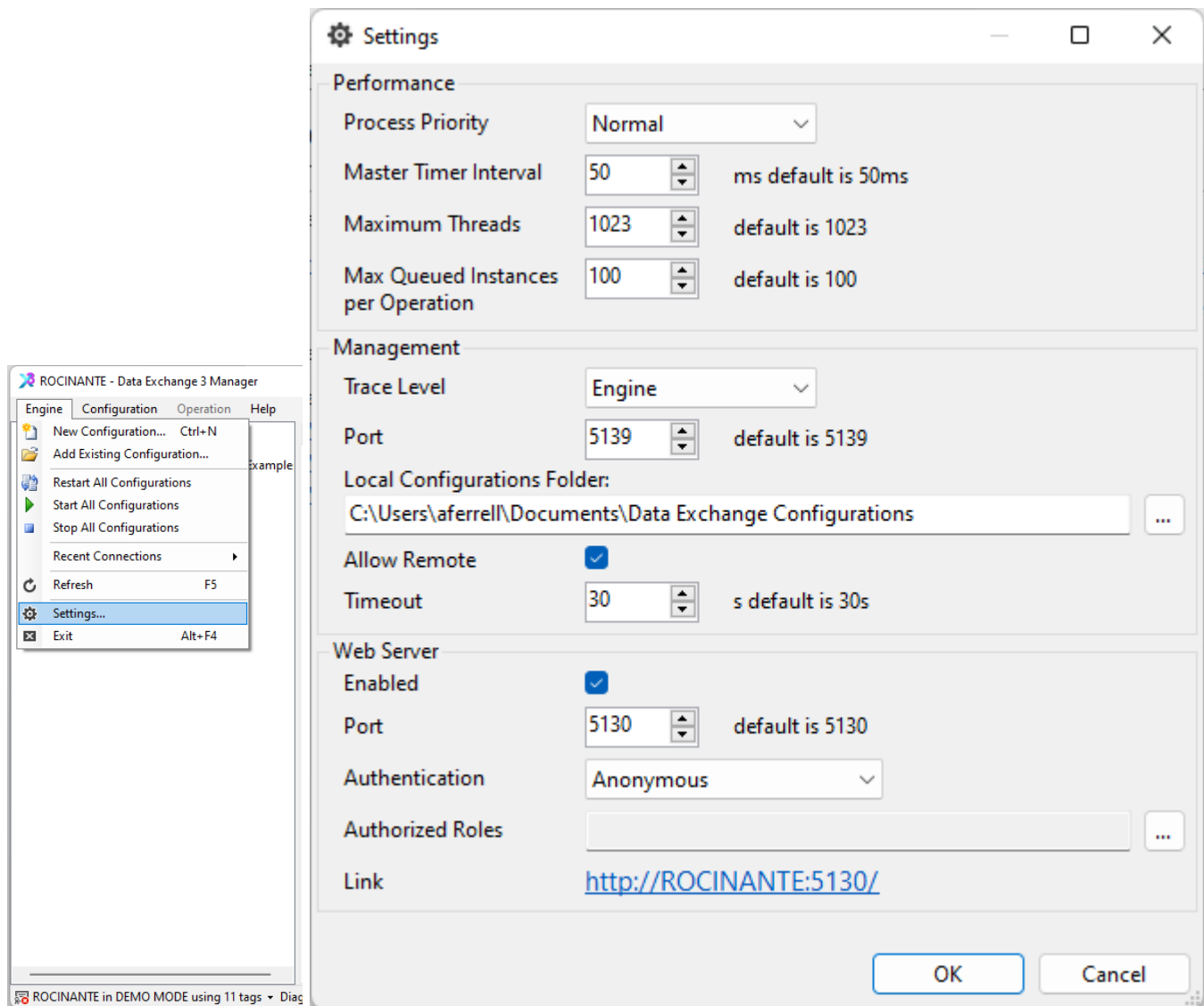
Uploads the current engine copy of the configuration and overwrites the older local copy

→ **Compare to Engine**

Compare the local configuration to the engine configuration

∨ Show Configuration Details Cancel

3.4. ENGINE CONFIGURATION SETTINGS



Configuration settings for the Engine are set in the Manager using the Engine Settings. The configuration settings are also available by executing the Data Exchange Engine 3 executable DXC3.exe.

A. *Process Priority*

Sets the priority given to allocating processor time to the Data Exchange 3 Engine. Changing this setting will affect the performance of Data Exchange 3 Engine and indirectly affect other applications.

B. *Master Timer Interval*

Sets the timer that is used for scheduling within the Data Exchange 3 Engine. If less resolution is required, a higher value can be set. All Scheduled triggers must be evenly divisible by this setting.

C. *Maximum Threads*

Sets the maximum threads that can be used to process operations at one time. All configurations share this pool.

D. Max Queued Instanced per Operation

Sets the maximum number of queued instances per Operation. Operations exceeding the maximum queued number will be dropped.

E. Trace Level

Allows setting for additional detailed logging of Events. Changes to this setting takes affect when this dialog window is closed by clicking OK

None Default, no extra Event Logging

Engine Include additional detailed logging for the Engine.

Operations Includes Engine Trace Level, plus sets the Detail Logging for all Operations.

This will cause a large number of events to be logged and is intended for temporary usage.

F. Local Configurations Folder

Contains the directory to which configurations are stored on this computer.

NOTE: The default is to store configurations at the User level and not at the computer level. Different users can be editing their own copies of a configuration running on the engine.

G. Allow Remote Connections

Allows remote computers to access the DXC3 Engine running on this computer. Turning on remote connections requires a restart of DXC3. This can be disabled to increase security but all would have to be done from the local computer.

H. Timeout

Sets the timeout from the Manger to the engine. This applies to all synchronous management operations (e.g. Starting an operation, Renaming a configuration, etc.)

I. Enabled

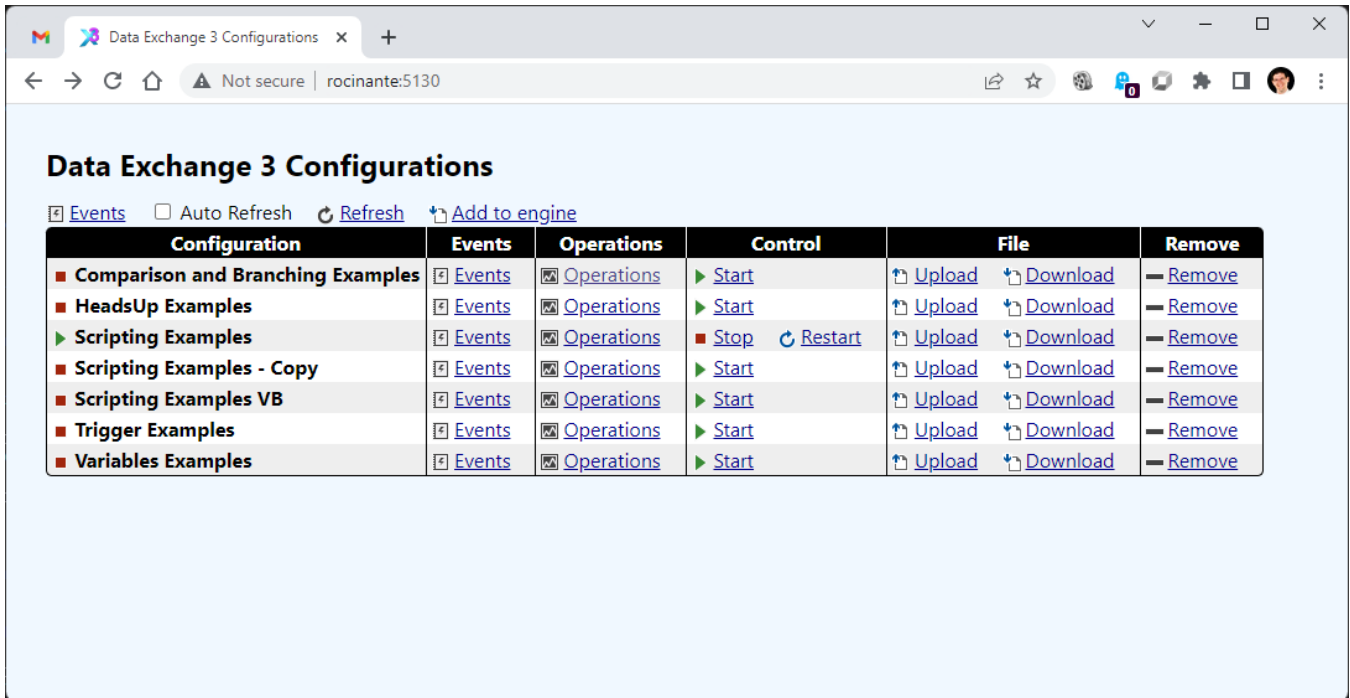
Enables the DXC3 service's built-in webserver to allow management via browser or REST API and remote triggering of operations (using the DXC3COM object or SQL CLR procedures).

J. Port

Sets the port used by the DXC3 Manager and DXC3 Editor to communicate with the DXC3 Engine. If DXC3 is running in parallel with a previous version of Data Exchange, the Port may need to be changed from default port of 5139.

K. Web Server

The Web Server allows a RESTful Web Services interface with the DXC3 Engine. There is a web version Manager that is available on this port if the Web Service is enabled. This can be disabled to increase security but all external triggers would have to be done from the local computer.

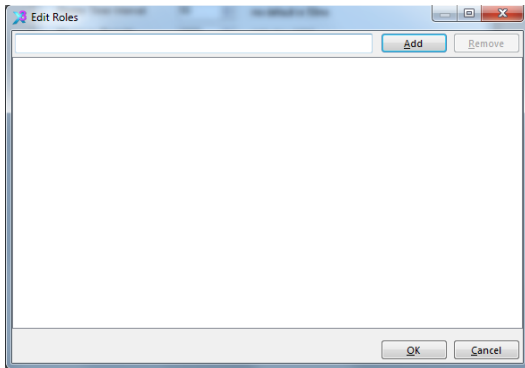


L. Authentication

Determines the kind of Authentication the management web server uses.

M. Authorized Roles

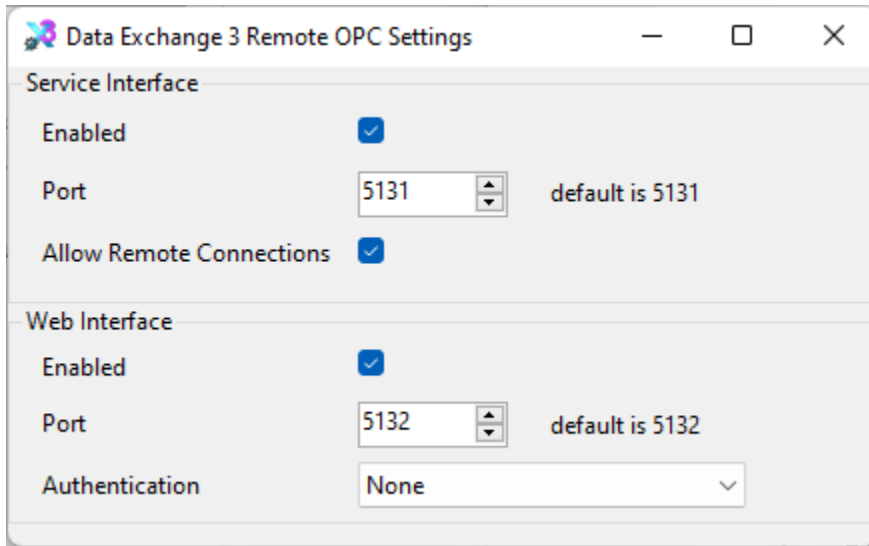
User names or groups that are allowed to access the management web server. Only works if the Authentication selection is set to something other than Anonymous or None.



N. Link

Link brings you to the Web Manager.

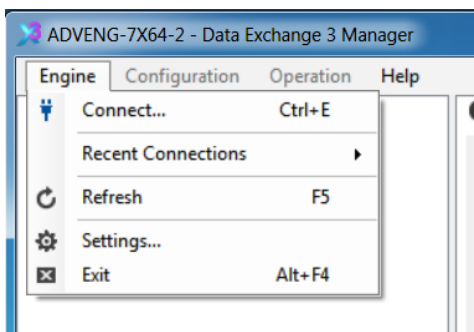
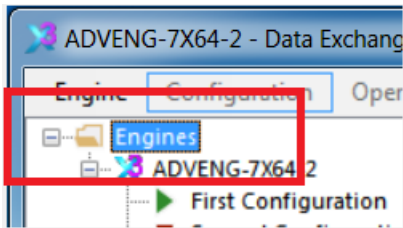
3.5. REMOTE OPC SERVICE CONFIGURATION SETTINGS



Configuration settings for the Data Exchange 3 Remote OPC Service are available by executing the Data Exchange 3 Remote OPC Service executable DXC3OPCRemote.exe.

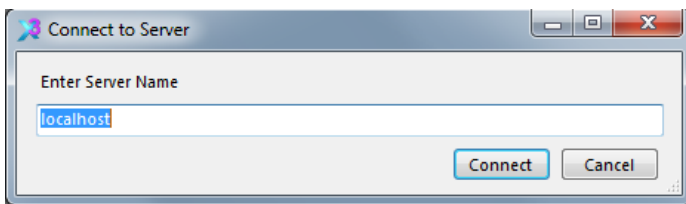
3.6. ENGINE CONTROL (ENGINE MENU ITEM)

When the DXC3 Manager starts it will connect to the local engine automatically. To connect to a remote engine, select the Engine Menu option.

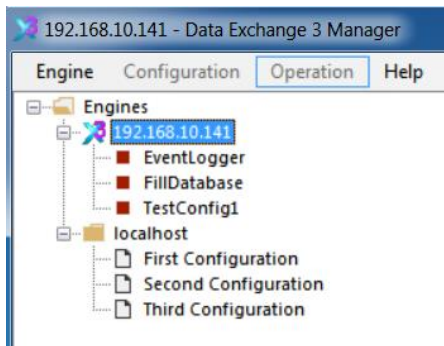


A. Connect...

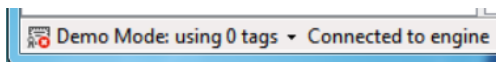
Connect allows you to connect to the local Data Exchange engine or a remote one. To connect to a remote DXC engine, type in the name of the server (or IP Address) and press Enter. Be sure Windows Firewall allows the Data Exchange Manager to connect to the engine (default port 5139). To connect to the engine on the same computer running the manager, enter "localhost" for the server name. The port for the Engine can be specified by using this format: "localhost:5139". Select "Connect" to connect to the engine.



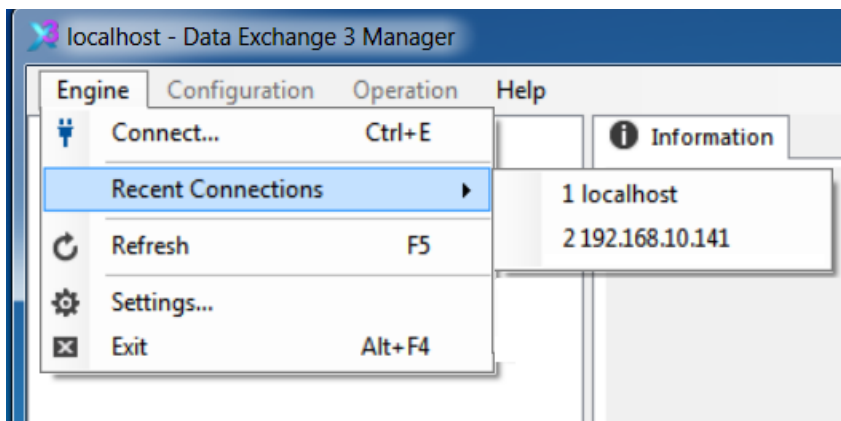
In this first example, the manager is connected to remote computer 192.168.10.141. The DXC3 symbol is next to the engine name indicating that it is connected.



Once the Manager connects to an engine, the status message in the lower left of the Manager main window will read, "Connected to engine".



B. Recent Connections



C. Refresh

Refreshes the folders displayed in the Engine.

D. Settings

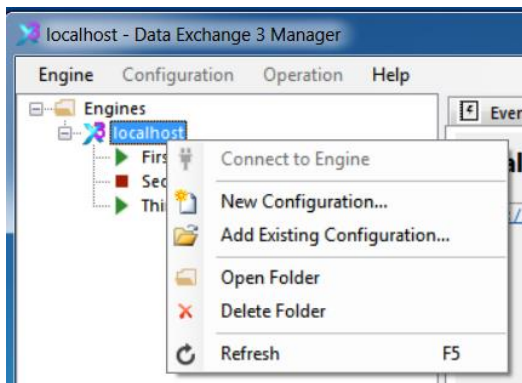
Please see the Engine Configuration section in this document.

E. Exit

Exits the DXC Manager.

3.7. SERVER CONTROL

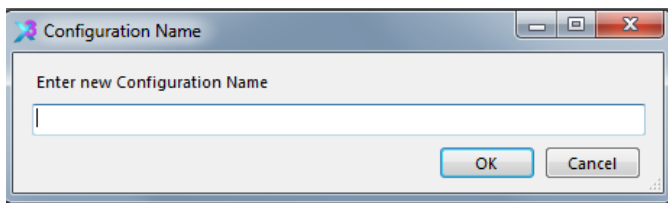
Right click on a specific DXC Server to configuration to modify the server.



A. **Connect to Engine**

Connects to a new engine.

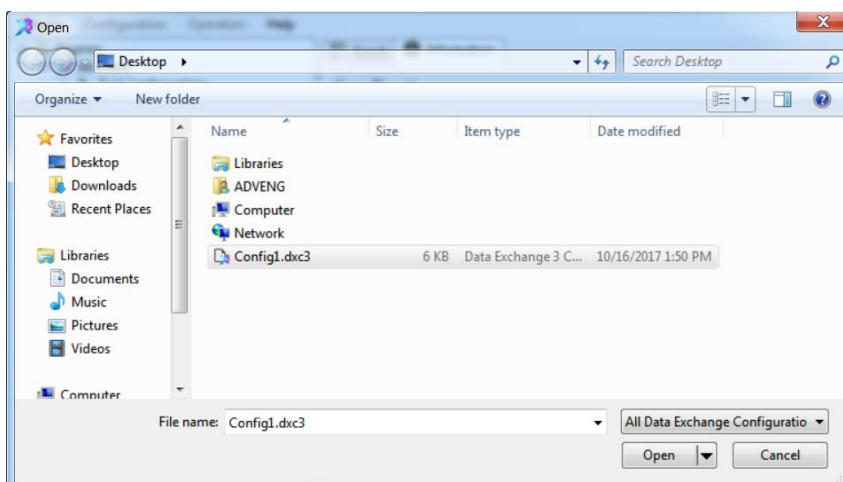
B. **New Configuration...**

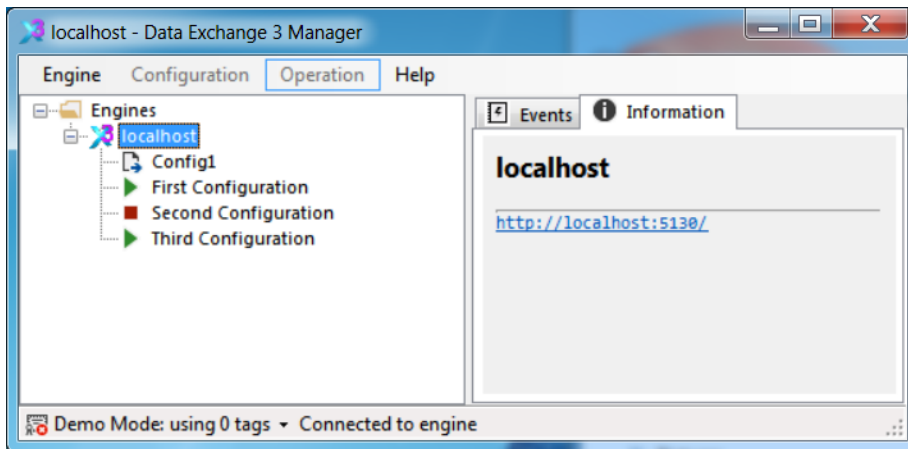


Creates a new configuration file in the Local Configurations Folder and adds the configuration to the Engine. The Data Exchange 3 Editor is automatically opened with the new configuration for editing. If changes are made the configuration will need to be downloaded to the Engine.

C. **Add Existing Configuration...**

Add Existing Configuration adds it to the engine however; it will not start the configuration until you select it to start. Adding a configuration to the engine does not add the file under Configurations. If you double-click on the configuration, it will be added to the Configuration directory and editable.





D. Restart All Configurations

Restart All restarts all configurations that are currently started. It will not start a stopped configuration.

E. Start All Configurations

Starts all DXC configurations uploaded to the selected Engine.

F. Stop All Configurations

Stops all running configurations.

G. Recent Connections

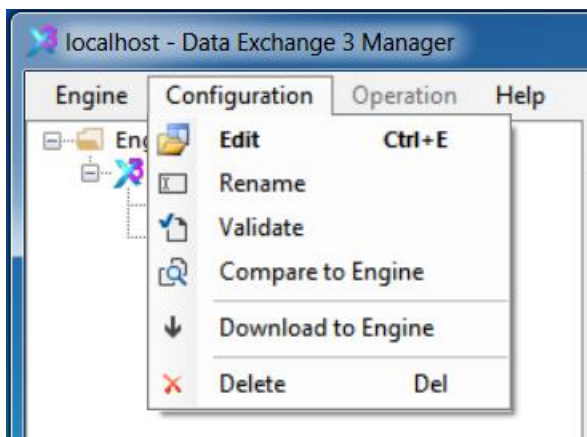
List of recently connected Data Exchange Engines that can be clicked to connect to the displayed connection.

H. Refresh

Refreshes the folders.

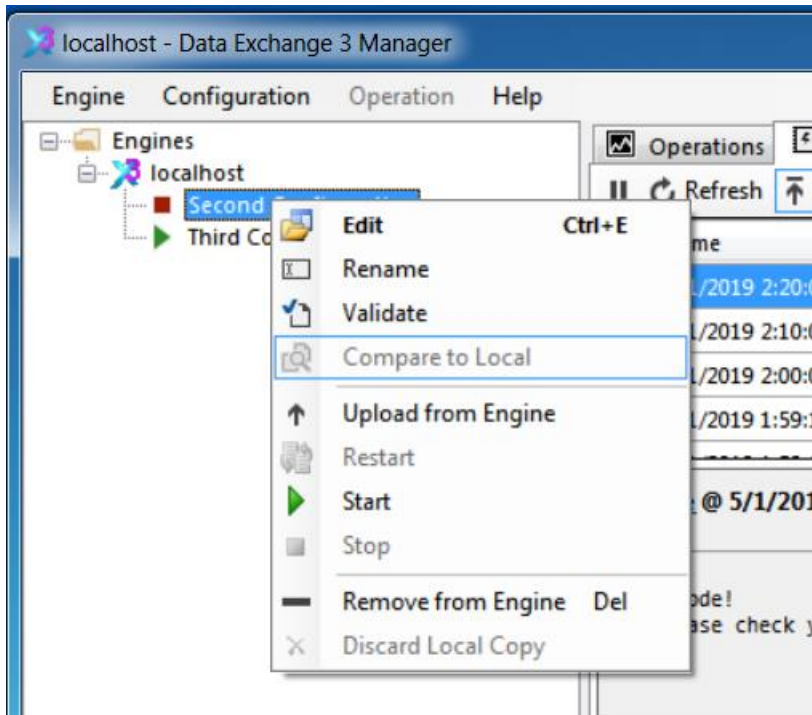
3.8. DXC FILE CONTROL (TREE MENU - RIGHT CLICK)

Select a configuration and select the Configuration Menu item. OR...



If the directories are deleted, any configurations within the deleted directories will be lost.

Selecting a configuration file and right clicking gives you additional options when editing files.



A. **Edit**

Selecting Edit will display the details of the configuration in the Data Exchange Editor. Data Exchange will not allow you to open the Editor more than once for the same configuration file.

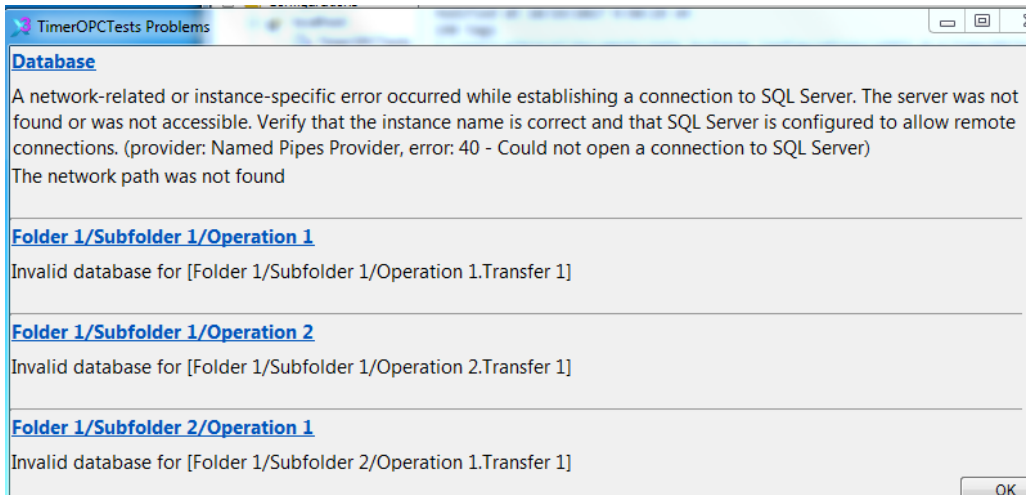
B. **Rename**

Rename the selected configuration. If a configuration with that name already exists, a warning will be shown. Configuration names can be any valid Unicode string except the following characters:

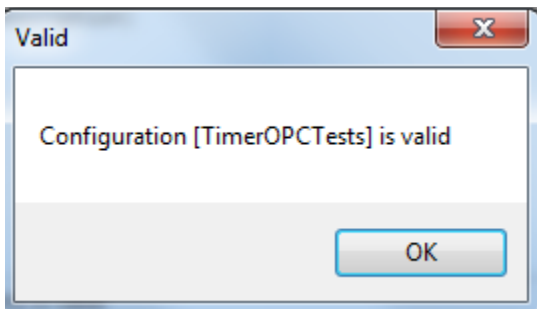
< > : / \ | ? * . "

C. **Validate**

Validate will confirm that all connections to databases and OPC servers are working properly, that all scripts compile and all OPC items are valid. In this example, DXC does not have a connection to the database and has logged errors for each operation that uses the database:

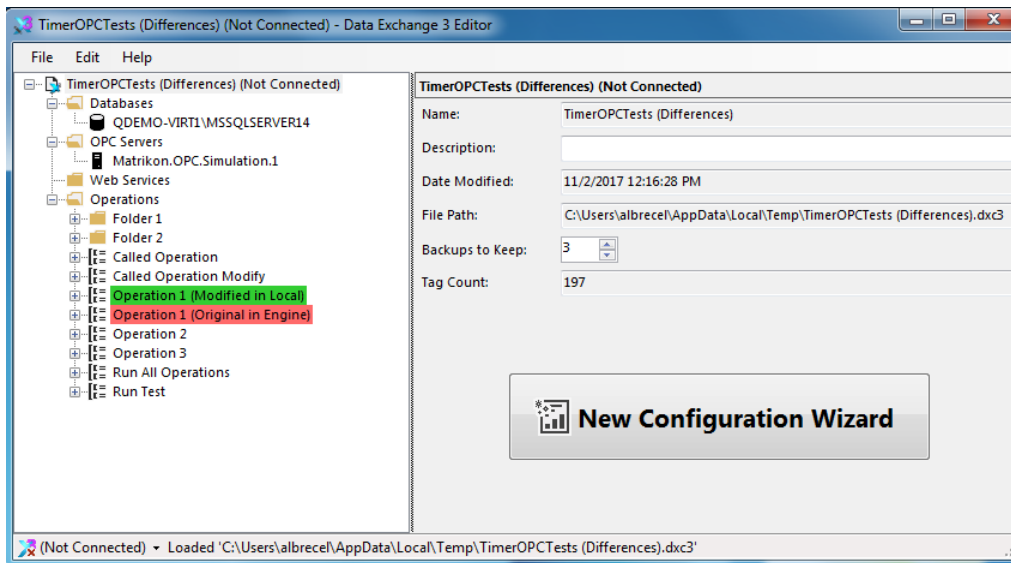


Fixing the connection to the database produces a valid configuration message:



D. Compare to Local

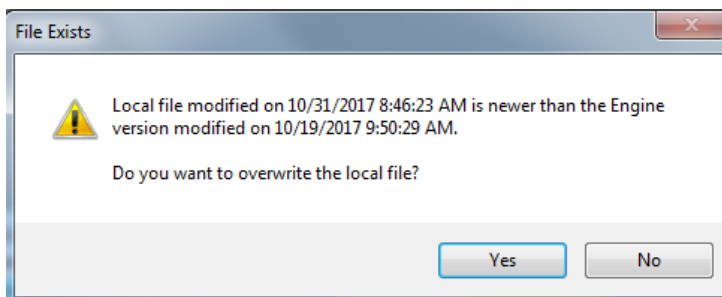
Compares the configuration running in the Data Exchange 3 Engine with the configuration stored in the Local Configurations Folder. The differences are highlighted in the Data Exchange 3 Editor along with comments added to the object names. Operations are highlighted if differences are found within the items of the Operations. Transfers are also highlighted if differences are found. Items such as schedules are not highlighted and are only indicated by the highlighting of the Operation.



E. Upload from Engine

If the configuration file does not exist on this computer or it is out of date with what is running, Upload the file from the engine.

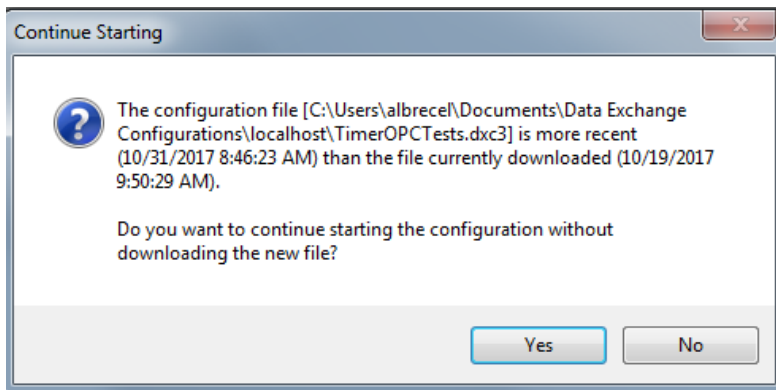
If the local file is newer than what is running in the engine you will be prompted with a message:



F. Restart/Start/Stop

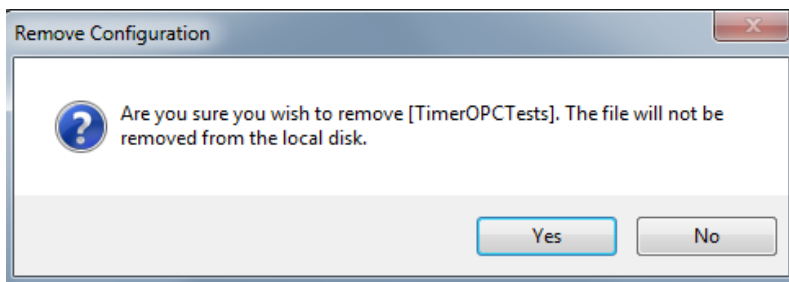
To change the runtime status of the configuration file, select Restart, Start or Stop. DXC Manager will not prompt if you are sure.

If you start or restart a configuration and if the local file is newer than the file uploaded to the engine, Data Exchange will prompt to upload the latest version.



G. Remove

To remove a configuration from the engine, select Remove and click Yes to the confirmation prompt. The configuration stops if it is running and is removed from the engine. This will not affect the other configurations that are running.

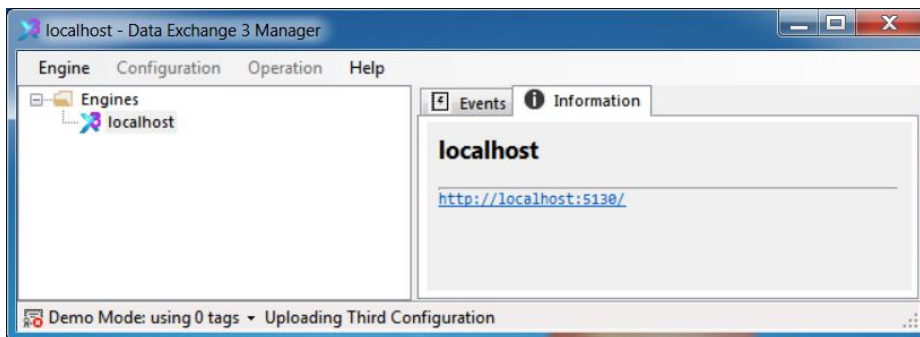




Note. If you remove the file from the engine and it is NOT stored locally then you will have lost the configuration.

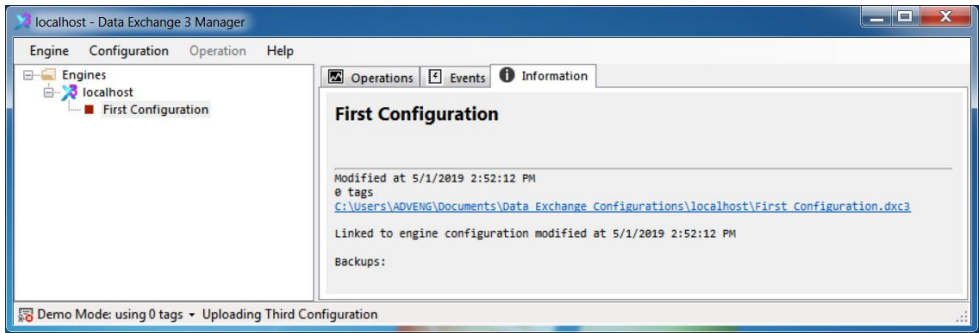
H. File Icons


The Icons on the file will help you understand the versions that are running as well as stored for the current user.

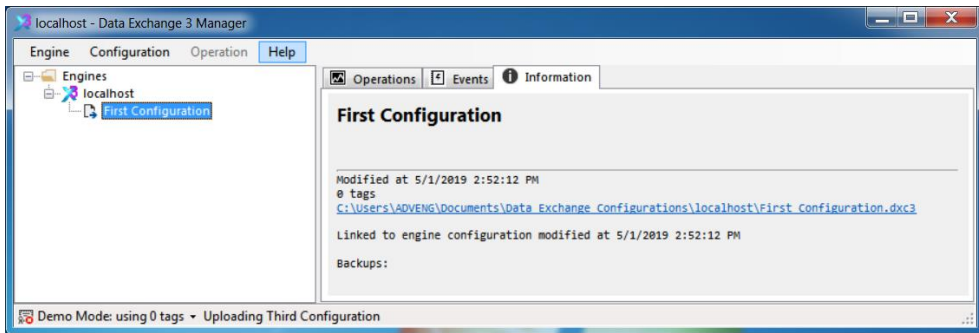
Starting with no configurations stored for the local user and no configurations in the Engine:




Adding a new configuration adds the file to the local offline configuration directory as well as the engine. The information page shows the offline version location of the file and the file date information. The   symbol means that the file is not running but the file is located in the engine.



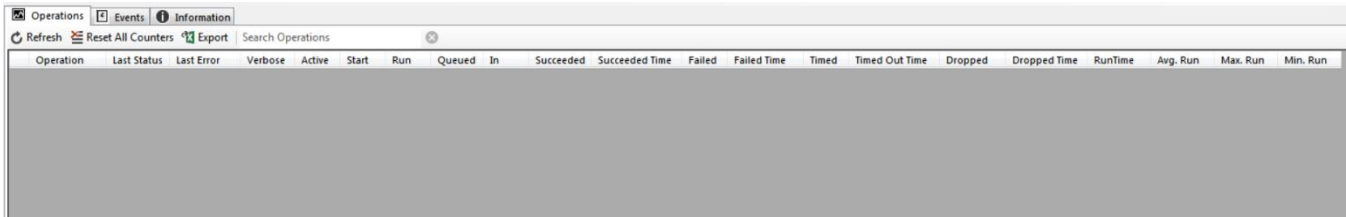
Removing the file from the engine will cause a prompt for confirmation of the removal of the file. If the user selects 'Yes', the file will still be stored in the offline configuration directory but will no longer be in the engine. The  symbol shows the file is offline only.



The play symbol  on a file means that the configuration is running in the engine. Hovering over the filename will give you additional status on the file. The file may be "Up to Date" or have "No Local File" so be careful when adding/deleting files. The Information tab gives more information on the status of the files.

3.9. OPERATIONS TAB

The Operations tab displays the operations that are currently running in the engine. If there is nothing running or you are not connected, the window will be blank. For additional details, please refer to the Data Exchange Runtime Features section of this document.



Operation: Operation name

Last Status: Last status is the last code returned for the operation. Valid codes are:

- 0 – Unknown
- 1 – Success
- 2 – Failure
- 3 – Timeout
- 4 – Inactive
- 5 – Dropped

Last Error: The text of the last error.

Verbose: Enables verbose event logging, which will generate additional event messages.

Active: Allows dynamically enabling and disabling the operations.

Start: Allows manual triggering of the operation to start.

Run: The number of times the operation has been triggered.

Queued: The number of operations that have been queued and are waiting to run.

In Process: The number of operations that are currently running.

Succeeded: The number of times the operation ran successfully.

Succeeded Time: The last time the operation ran successfully.

Failed: The number of times the operation failed. An operation will fail if there is an issue with the configuration or the operation, i.e. a variable is unavailable.

Failed Time: The last time the operation encountered a failure.

Timed Out: The number of times the operation timed out while executing. This occurs any time the operation takes longer than the specified timeout value, for example, if a database connection time occurs.

Timed Out Time: The last time the operation encountered a timeout.

Dropped: The number of times the operation was triggered but wasn't queued. For example, operations that are not set for parallel execution will drop operations if the previous execution of the operation is still executing and operations that are set for parallel executing will drop operations if the number of queued operations exceeds the maximum set in the settings dialog.

Dropped Time: The last time the operation was dropped.

Run Time: The amount of execution time of the last cycle of the operation.

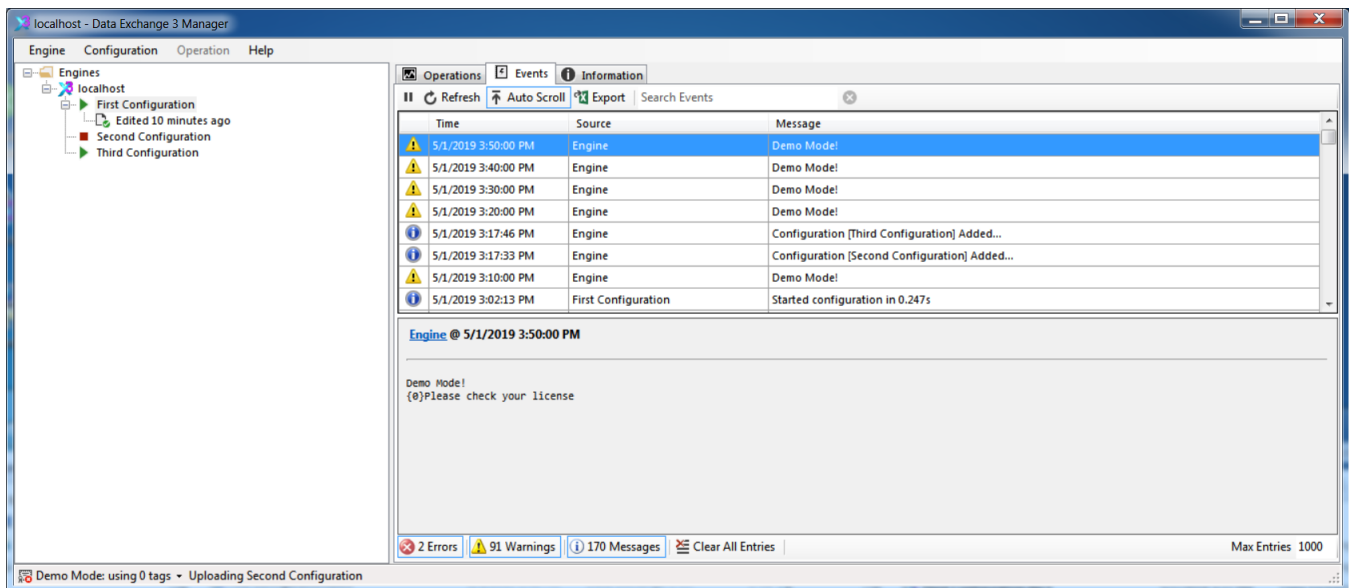
Avg. Run: The average duration of the execution of the operation.



Max. Run: The longest duration of the execution of the operation.


Min. Run: The shortest duration of the execution of the operation.

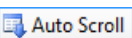
3.10. EVENTS TAB


The Events tab displays all system events occurring on the computer to which you are connected.

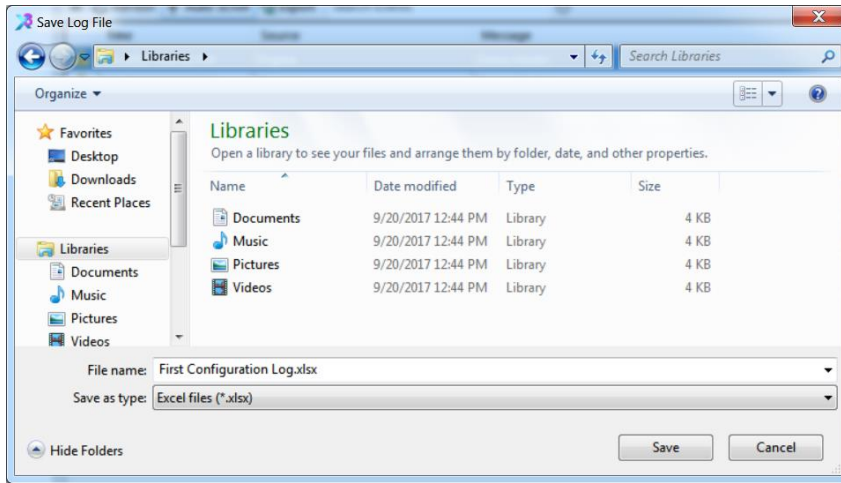


The Pause button  stops the auto loading of the events. When loading is paused the button changes to a Play button. Press Play  to continue auto loading.

Select  Refresh to refresh the list of events. If you select Refresh, auto loading will resume.

 Auto Scroll turns on or off Auto scrolling. If Auto scrolling is on, the box has a border around it. If it is off it does not. Auto Scroll will scroll the list to the top when new events are added. If Auto Scroll is turned off, the selected event will stay on the screen.

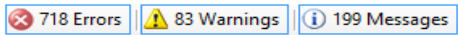
Use  Export to export the Event Log to an Excel file. Give the file a name and press Save to store the file to the computer.



To search for a specific event, enter a word or phrase in the search text box to search in the event log. To clear the search results, press the button.

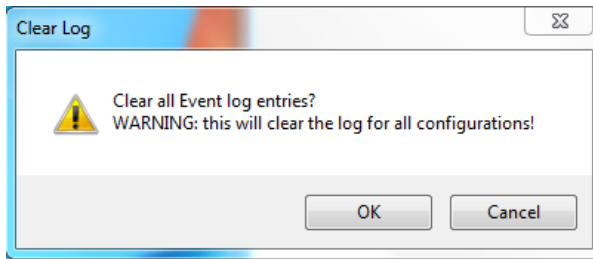
A. Filtering Events

To turn on or off viewing of the three types of events toggle the event type buttons at the bottom of the window.



B. Clear All Events

The button will clear all messages from the event log. Clearing the event log will remove all events from all configurations, not just the selected configuration.

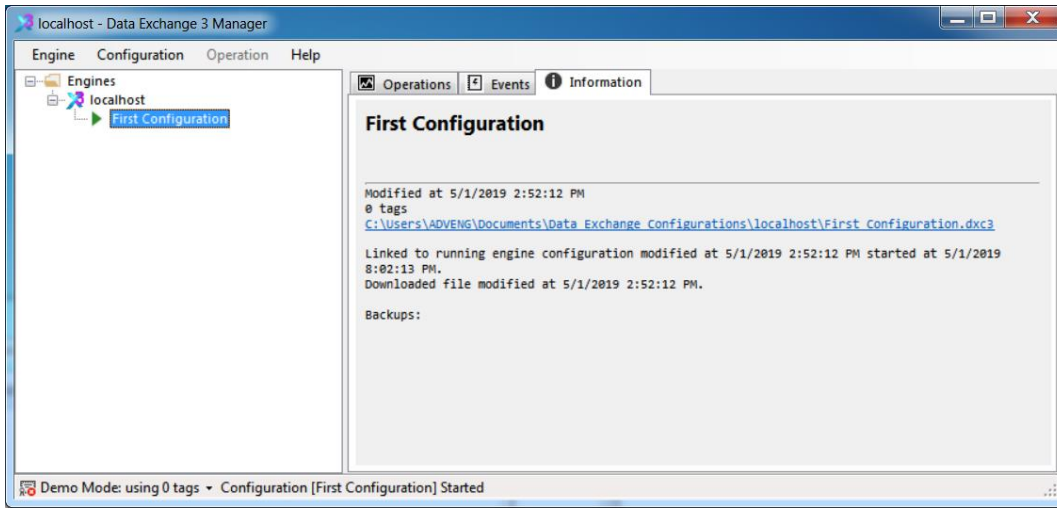


C. Max Entries

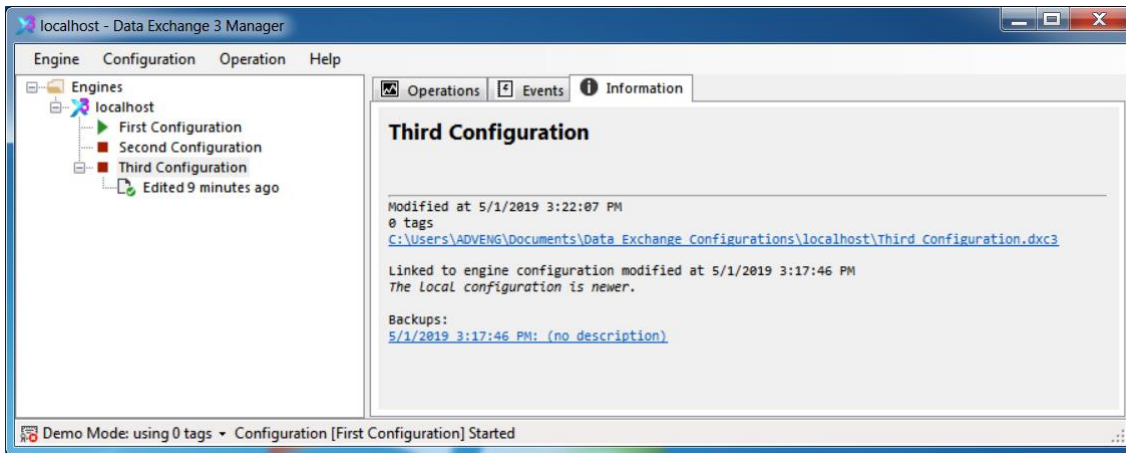
The button controls the number of messages loaded from the Event Log.

3.11. INFORMATION TAB

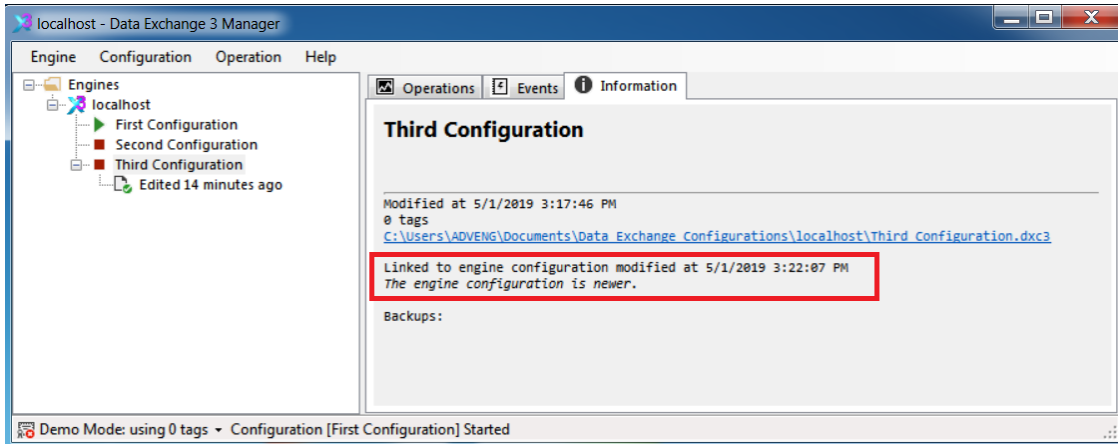
The Information tab displays the status information for each configuration file that exists for a DXC Server.



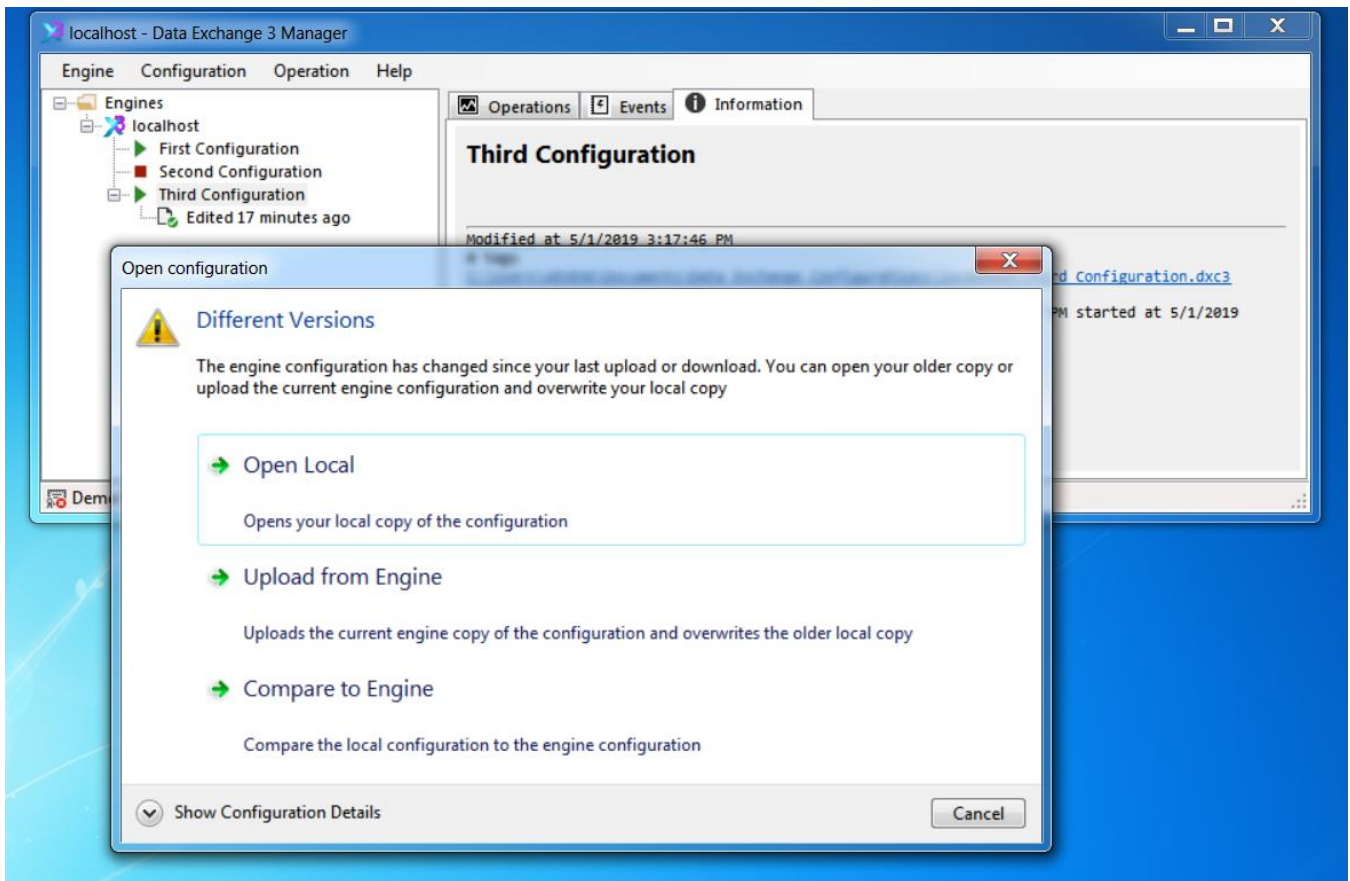
In this screenshot, the “Third Configuration” file was edited and is newer than what is running in the engine:



And here the DXC file in the engine is newer than the offline file.

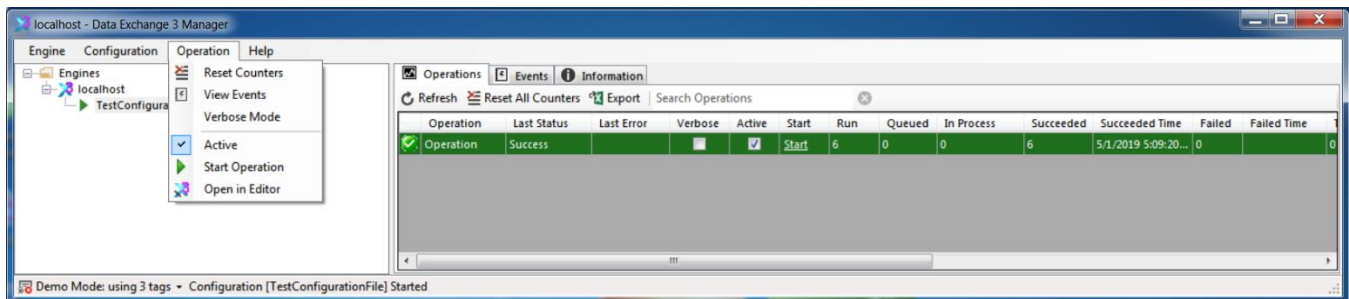


Double clicking on the filename will edit the running version of the file. If there is a discrepancy between the file in the engine and the offline file, DXC will prompt requesting a choice of which file to open.



3.12. OPERATION CONTROL

Operation control is available when you are clicked on a configuration loaded into the engine.

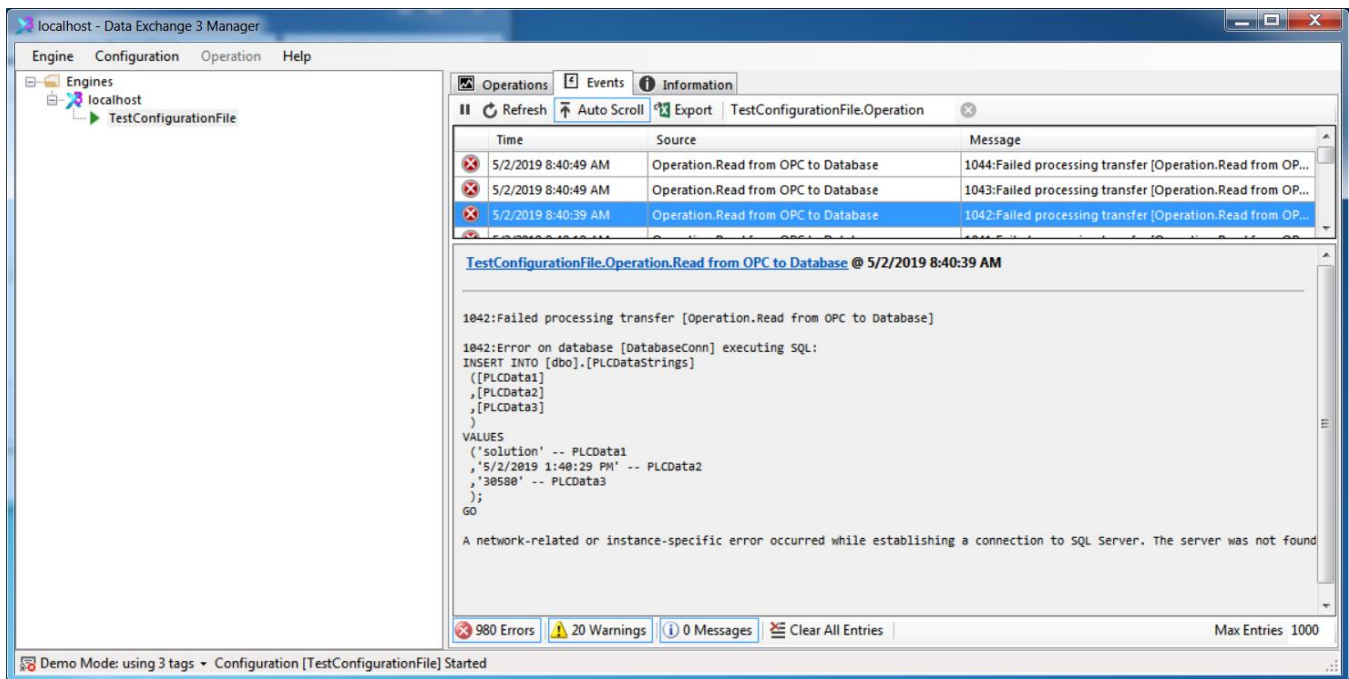


3.12.1 RESET COUNTERS

Resets the runtime counter information on the Operations page for the Configuration file.

3.12.2 VIEW EVENTS

Displays the Events tab for the highlighted operation in the current Configuration file.



In this example, I selected the Operation “Operation” inside the config1.dxc configuration file. This operation generated errors due to a lost connection with the database.

Only the event messages for Operation are displayed. To view all events, clear the search box by clicking on the X next to the search box.

3.12.3 VERBOSE MODE

Click Verbose Mode to turn on additional debug printing. Use caution when using this feature as there is a lot of detail printed. It is possible that the operation will slow.

3.12.4 ACTIVE

Displays the current Active status of an operation. Unclick the checkbox to deactivate the operation and prevent it from running.

3.12.5 START OPERATION



Triggers the selected operation to run.

See the Operation Tab section for further details regarding other fields displayed on the Operations page.

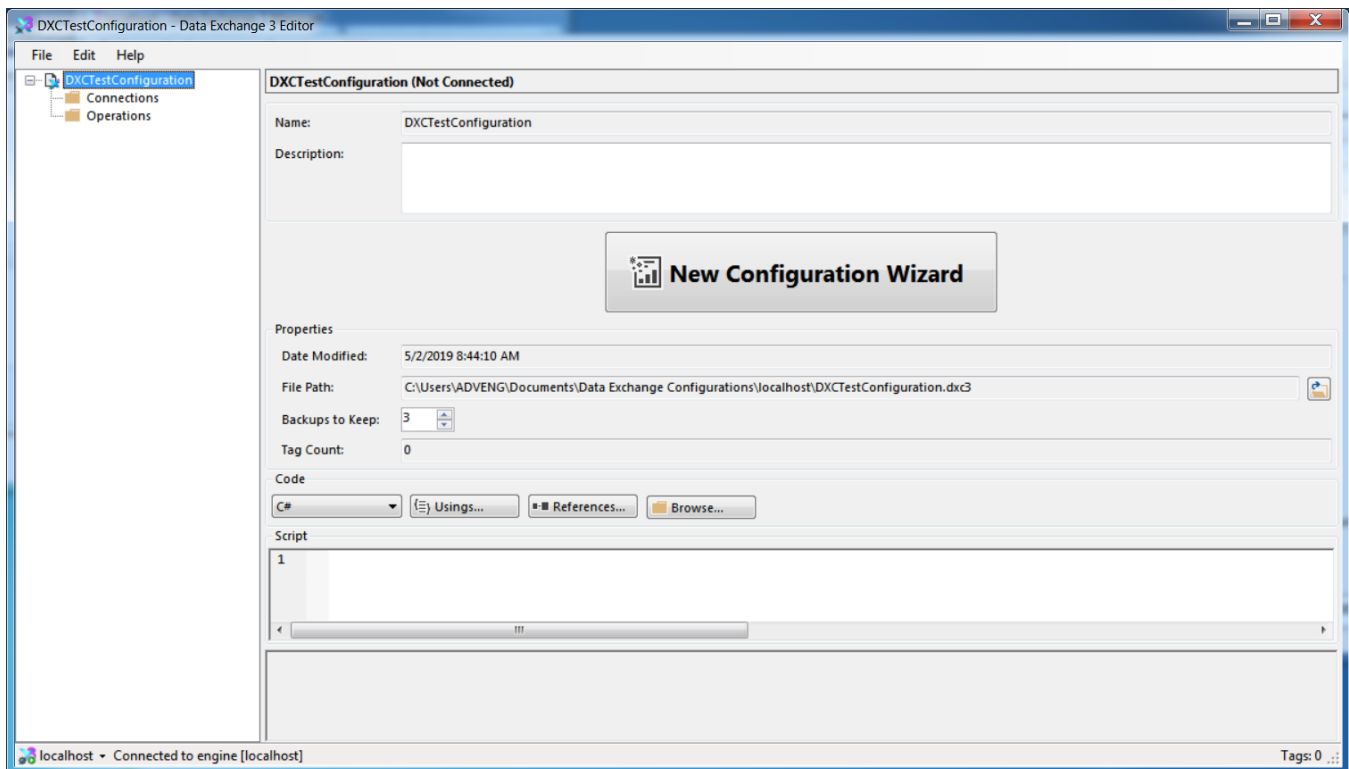
SECTION 4. EDITOR

4.1. OPENING THE EDITOR

You can invoke the Data Exchange Editor from a shortcut or from the Data Exchange Manager.

A DXC configuration file consists of three different components that are required to store OPC data to a database. These are database connections, OPC Servers and Operations to transfer the data between the database and OPC Server.

Double-click the shortcut to invoke the editor. A blank configuration file is displayed.



A. File Date Modified/File Path

Specific information for this configuration file.

B. Backups to Keep

The .dxc file contains the current configuration file as well as maintains the backups so that moving files from one computer to another requires you to copy only one file. You can choose the number of the configuration backups to keep. By default, the number of backups to maintain is 3.

C. Tag Count

Tag count of the configuration as it applies to the license. This includes both OPC tags and Variables.

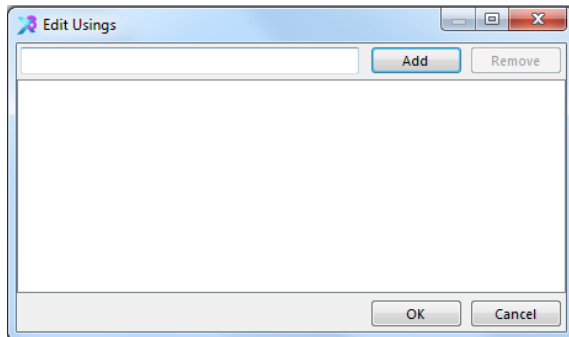
D. Code

DXC supports scripting in either C# or VB syntax. What code selection style is selected here is what dictates what is done in the remainder of the configuration file.

E. Using

Allows you to add namespaces that are included in the script by default. The following namespaces are included by default.

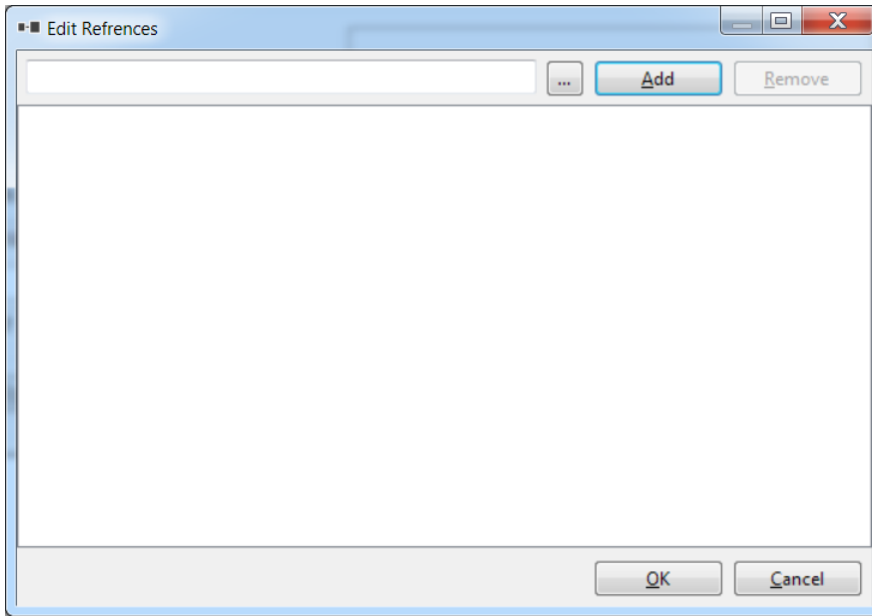
```
DXC3
Scripting
System
System.Collections
System.Diagnostics
System.Collections.Generic
System.Text.RegularExpressions
System.Linq
System.Text
System.Xml.Linq
```



F. References

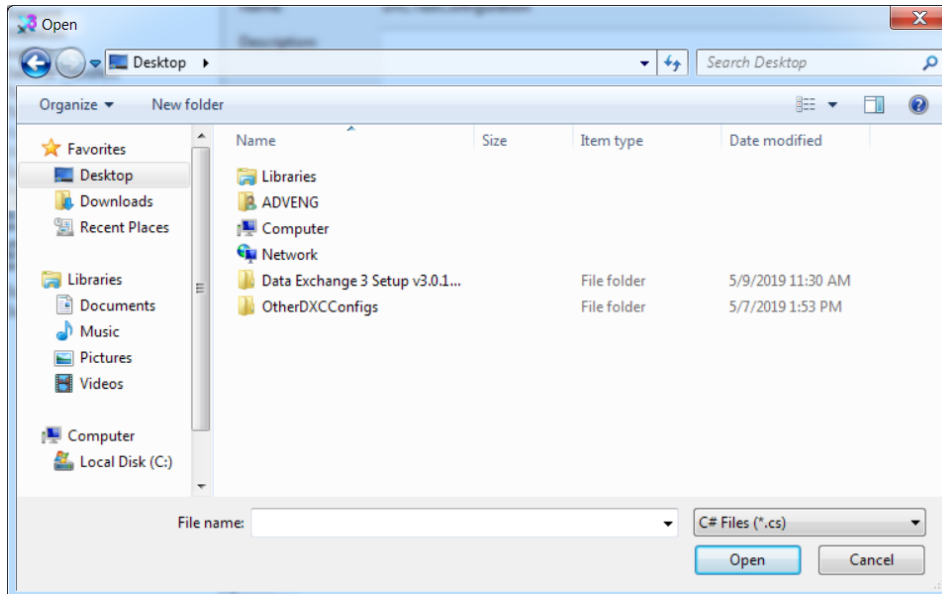
Allows you to reference other assemblies (DLL's) to be used in scripting. The following assemblies are included by default

```
DXC3Lib.dll
DXC3Interface.dll
DXC3Management.dll
System.dll
System.Core.dll
System.Xml.dll
System.Xml.Linq.dll
Microsoft.CSharp.dll
```



G. Browse

Allows you to import code from a file.



H. Edit in Visual Studio

Invokes Visual Studio and allows you to edit and test scripts in a more interactive environment. If you don't have Visual Studio installed on the machine that the editor is running on, you can download a free version from Microsoft at <http://visualstudio.microsoft.com/vs>

I. Script

Contains global methods that can be used throughout the DXC Operations in this configuration file.

4.2. EXAMPLES & SAMPLES

This documentation uses the Matrikon OPC Server as well as the Simulation Server available in Kepware. Matrikon can be downloaded during DXC installation. Kepware is available on the Kepware website.

Several SQL Tables are used throughout.

```
USE [DXCExampleDatabase]
GO
```

```
/****** Object: Table [dbo].[IntegerTable]  Script Date: 5/8/2019 1:18:52 PM *****/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[IntegerTable](
    [Integer1] [smallint] NULL,
    [Integer2] [smallint] NULL,
    [Integer3] [smallint] NULL,
    [Integer4] [smallint] NULL
) ON [PRIMARY]
```

```
GO
```

```
USE [DXCExampleDatabase]
GO
```

```
/****** Object: Table [dbo].[PLCDataStrings]  Script Date: 5/8/2019 1:19:16 PM *****/
SET ANSI_NULLS ON
GO
```

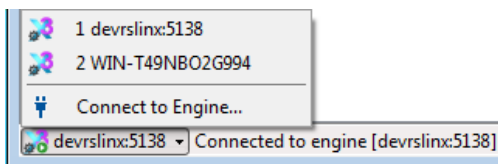
```
SET QUOTED_IDENTIFIER ON
GO
```

```
SET ANSI_PADDING ON
GO
```

```
CREATE TABLE [dbo].[PLCDataStrings](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [PLCData0] [varchar](150) NULL,
    [PLCData1] [varchar](150) NULL,
    [PLCData2] [varchar](150) NULL,
    [PLCData3] [varchar](150) NULL,
    [PLCData4] [varchar](150) NULL,
    [PLCData5] [varchar](150) NULL,
    [PLCData6] [varchar](150) NULL,
    [PLCData7] [varchar](150) NULL,
    [PLCData8] [varchar](150) NULL,
    [PLCData9] [varchar](150) NULL,
    [PLCData10] [varchar](150) NULL,
    [PLCData11] [varchar](150) NULL,
    [PLCData12] [varchar](150) NULL,
    [PLCData13] [varchar](150) NULL,
    [PLCData14] [varchar](150) NULL,
    [PLCData15] [varchar](150) NULL,
    [PLCData16] [varchar](150) NULL,
```

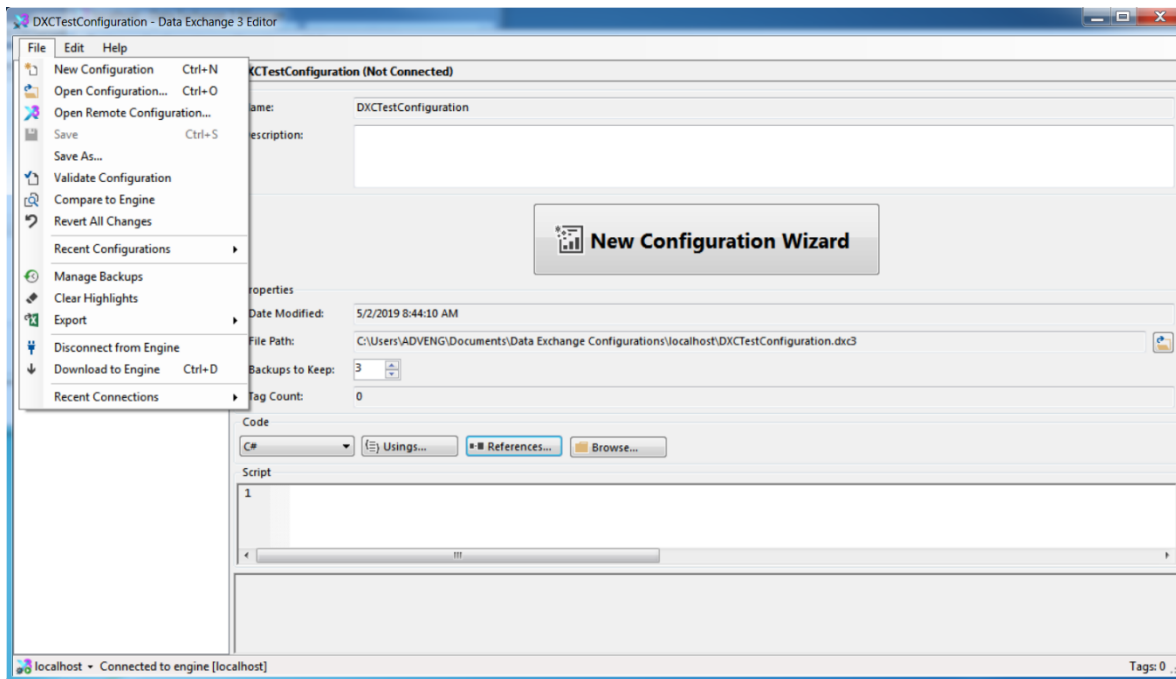
```
[PLCData17] [varchar](150) NULL,  
[PLCData18] [varchar](150) NULL,  
[PLCData19] [varchar](150) NULL  
) ON [PRIMARY]  
  
GO  
  
SET ANSI_PADDING OFF  
GO  
  
USE [DXCExampleDatabase]  
GO  
  
/***** Object: Table [dbo].[PLCTriggerTable]  Script Date: 5/8/2019 1:19:34 PM *****/  
SET ANSI_NULLS ON  
GO  
  
SET QUOTED_IDENTIFIER ON  
GO  
  
CREATE TABLE [dbo].[PLCTriggerTable](  
    [TriggerValue] [int] NULL  
) ON [PRIMARY]  
  
GO
```


4.3. ENGINE CONNECTION



The bottom left of the Editor form displays the current Engine connection status. Several of the functions of the Editor requires an active connection with the Engine. Click the ▼ down arrow to select a previous Engine connection, or **Connect to Engine...** to connect to a different Engine.

4.4. FILE MENU



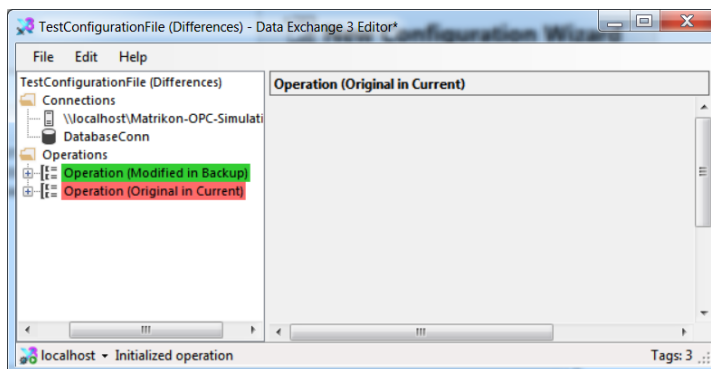
Note: The Editor must be connected to the engine for many of the File Menu items to be enabled. Use File Menu item  Connect to Engine... to Connect to the Data Exchange 3 Engine.

Validate

Validates the configuration and reports if problems are found. Disabled items and their subitems are ignored and will not be validated. Use before downloading to the engine to catch configuration errors before running the configuration.

Compare to Engine

Compares the configuration running in the Data Exchange 3 Engine with this configuration. The differences are highlighted in the Data Exchange 3 Editor along with comments added to the object names. Operations are highlighted if differences are found within the items of the Operations. Transfers are also highlighted if differences are found. Items such as schedules are not highlighted, and are only indicated by the highlighting of the Operation.



Revert All Changes

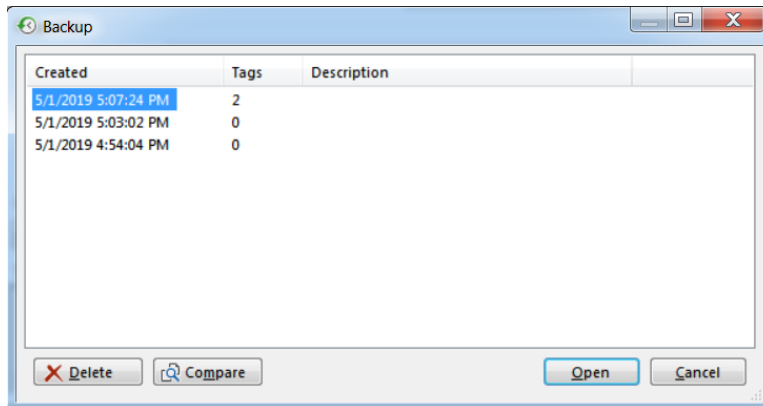
Removes all changes to the configuration file since the last save.

Recent Configurations

Lists previously opened configurations.

Manage Backups

To restore a backup, select "Manage Backups" from the Configuration drop down menu.



To restore a backup, select the specific configuration file to restore. Data Exchange will save the current version of the file as the newest backup file.

The Description column contains the description information typed into the Editor. The description helps document the versions of the configuration file to simplify restoring backups.

Click Compare to view differences in the backup and the current version.

Clear Highlights

To clear the highlighted items displayed in a search, select "Clear Highlights" to un-highlight the selected items.

Export Tags/Operations

The Editor allows you to export the tags and the operations to an Excel file. Tags are sorted based on OPC Server. For each tag, the Trigger and Transfers are identified. The Operations spreadsheet lists each Operation along with each trigger, transfer name and transfer type.

Sample of the Operation spreadsheet:

Name	Type	Description
/		
KepwareDataSave	Schedule	
Read from OPC to Database	Simple OPC Read	
MatrikonDataSave	Schedule	
Read from OPC to Database	Simple OPC Read	

Sample of the Tags spreadsheet:

Server	Tag	Operation and Transfer	Description
\\localhost\Kepware.KEPServerEX.V6			
	Simulation Examples.Functions.Ramp1	Operation.Read from OPC to Database	
	Simulation Examples.Functions.Ramp2	Operation.Read from OPC to Database	
	Simulation Examples.Functions.Ramp3	Operation.Read from OPC to Database	
\\localhost\Matrikon.OPC.Simulation.1			
	Random.Money	Operation1.Read from OPC to Database	
	Random.String	Operation1.Read from OPC to Database	
	Random.Time	Operation1.Read from OPC to Database	

Recent Configurations

Displays recent Configurations. Click on a Configuration to open it.

Connect to Engine

Connects the Data Exchange 3 Editor to the Data Exchange 3 Engine. Some features will be limited when the Data Exchange 3 Editor is disconnected from the Data Exchange 3 Engine. When connected to the Engine, this menu item becomes **Disconnect from Engine**.

Disconnect from Engine

Disconnects the Data Exchange 3 Editor from the Data Exchange 3 Engine. Some features will be limited when the Data Exchange 3 Editor is disconnected from the Data Exchange 3 Engine. When disconnected from the Engine, this menu item becomes **Connect to Engine**.

Download to Engine

Downloads this configuration file to the engine. If the configuration is already running, you will be required to confirm restarting of the configuration.

Recent Connections

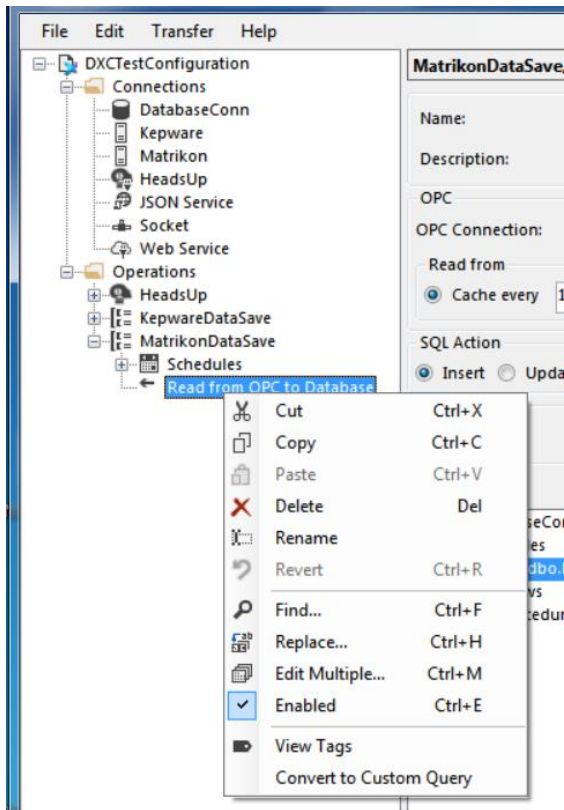
Displays recent Data Exchange 3 Engine connections. Click on a displayed connection to connect to the selected Data Exchange 3 Engine.

4.5. IMPORT PREVIOUS VERSIONS OF DATA EXCHANGE CONFIGURATIONS

Data Exchange II, 2.0, and 2.5 can be imported into Data Exchange 3 by simply opening the configuration. Data Exchange 1.x versions cannot be imported.

To prevent loss of configuration data, Disabled Operations should be enabled before importing, and then disable after imported.

4.6. TREE VIEW



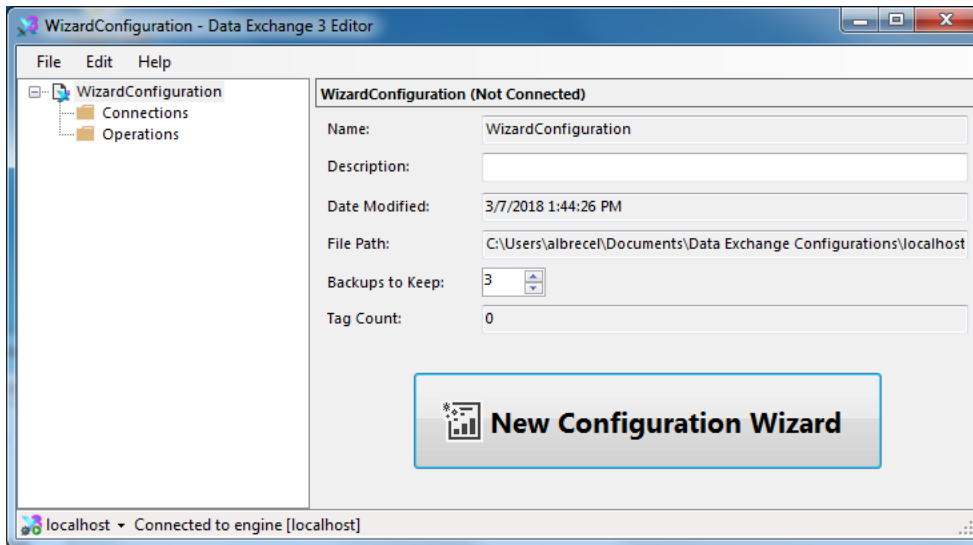
The tree menu on the left side of the Editor allows browsing through the setting of configuration items. Select the item on the tree menu, and the properties of the item is displayed on the left.

Items in the tree menu can be copied and pasted, which also allows copying existing items. Items in the tree menu can also be rearranged by dragging and dropping the items.

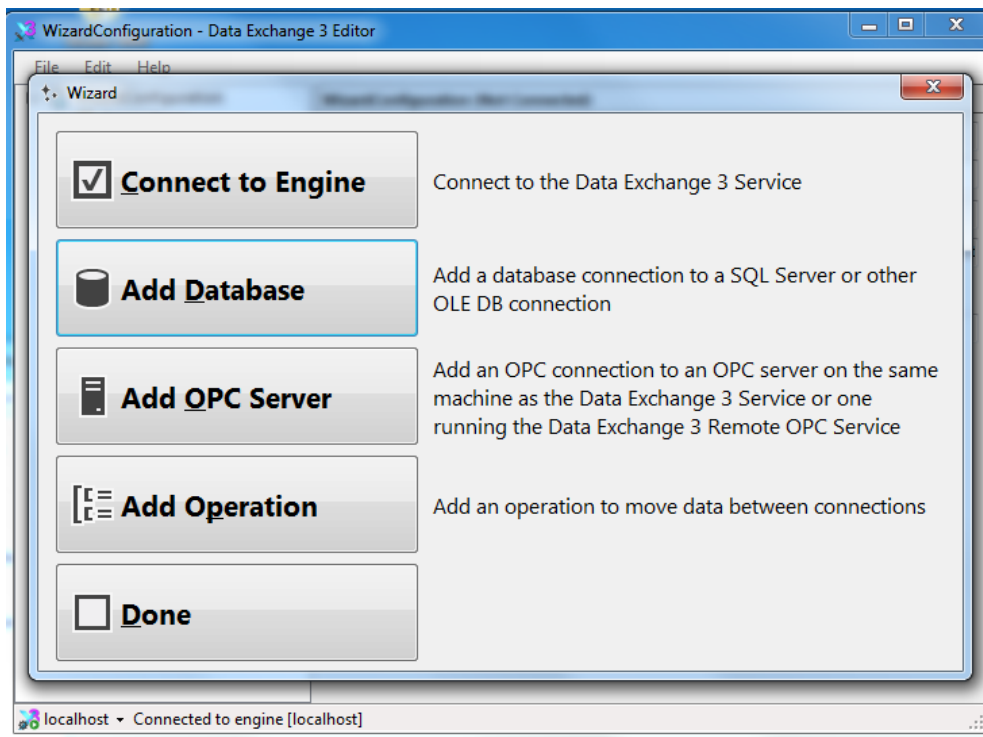
Right clicking items in the tree menu will display an item type sensitive menu. Most, but not all of the items in the right click menu are available in the top menu.

4.7. NEW CONFIGURATION WIZARD

Data Exchange has a “New Configuration” wizard for quickly creating new configuration files. In this example I created a configuration file called WizardConfiguration.

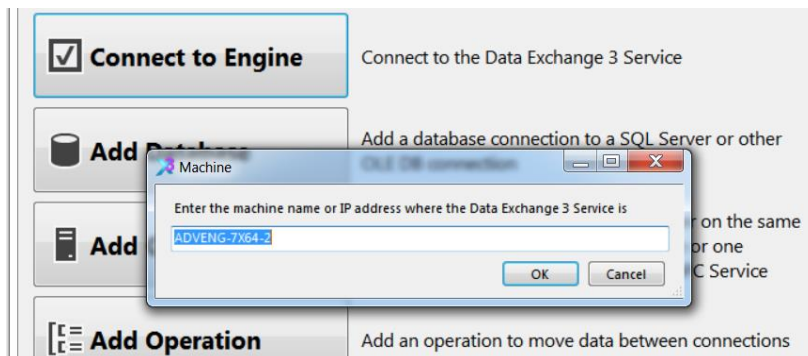


Click on the New Configuration Wizard button to access the Wizard menu.



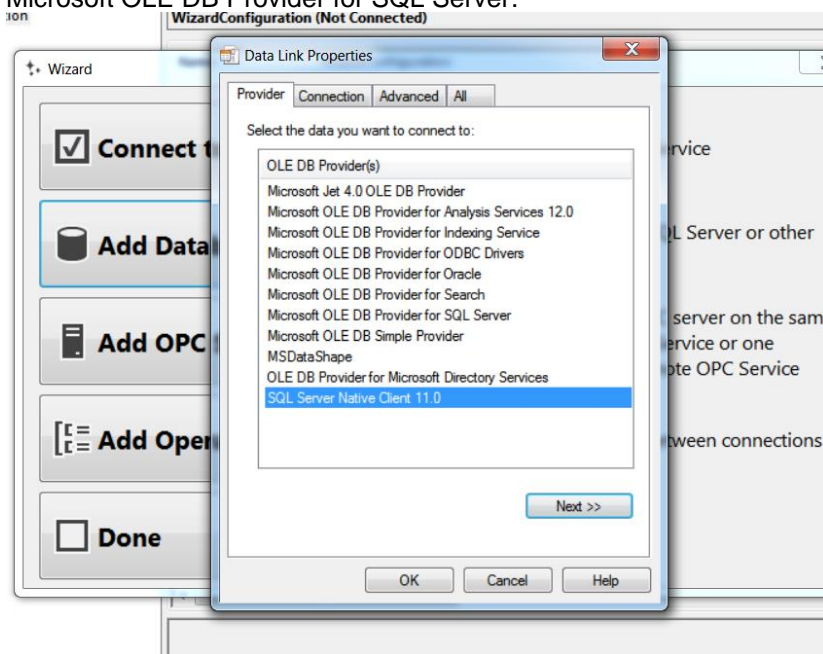
4.7.1 CONNECT TO ENGINE

Select Connect to Engine to connect to the Data Exchange engine that will be used to run the configuration file. By connecting to an engine, you can access the OPC tags during the development of the configuration file.

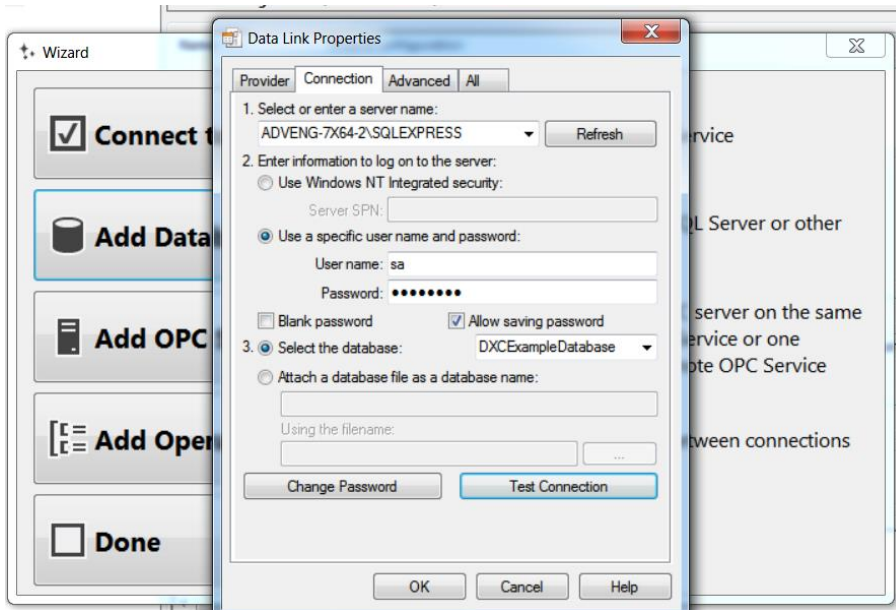


4.7.2 ADD DATABASE

Create a connection to a database. To communicate with SQL Server, use the SQL Server Native Client 11.0 or Microsoft OLE DB Provider for SQL Server.



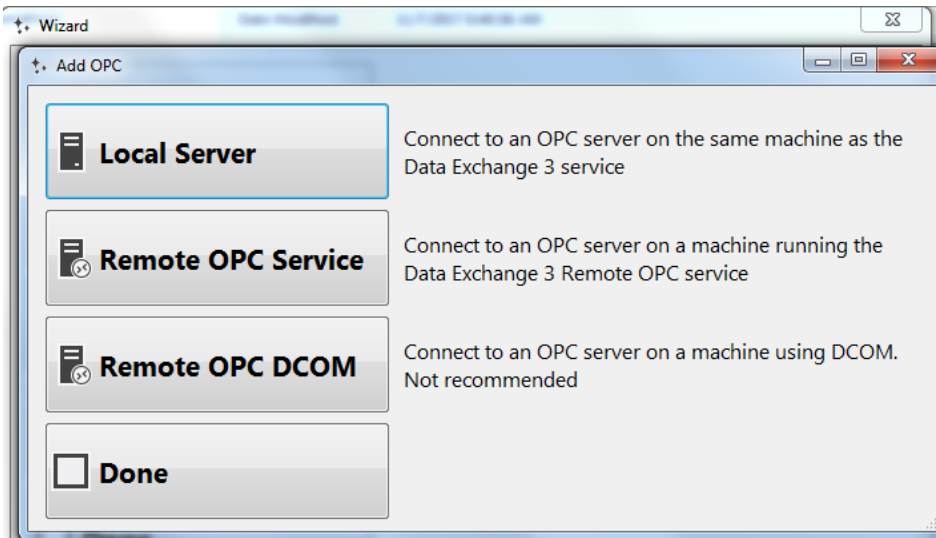
Enter relevant database connection information.



Select "Test Connection" to confirm a valid connection has been created to the database.

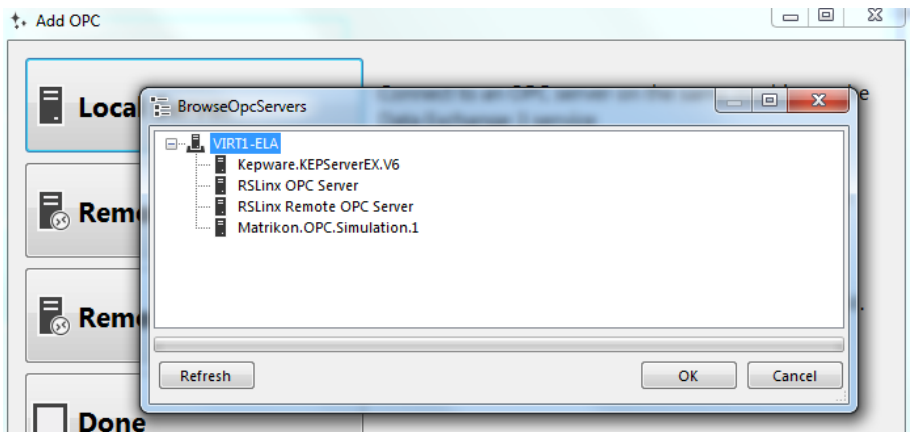
4.7.3 ADD OPC SERVER

Create a connection to a local or remote OPC Server. Connections to remote OPC are through the Data Exchange remote OPC service or DCOM. DCOM is not recommended but in some cases, may be necessary.



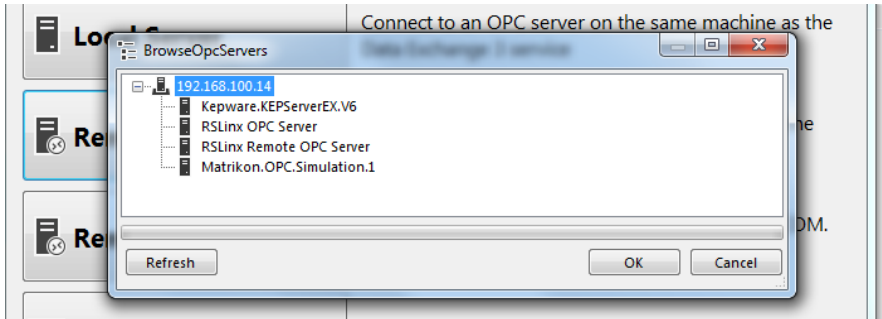
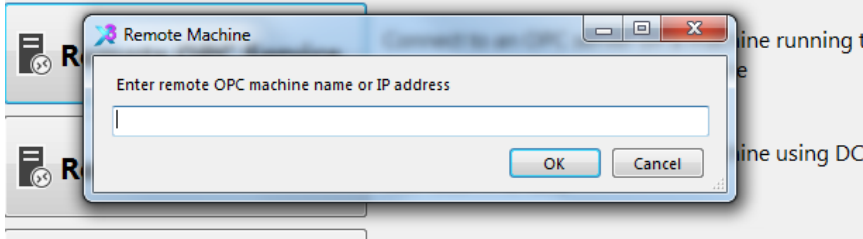
Local OPC Server

The local option will connect to the localhost and display all registered OPC servers for the computer.



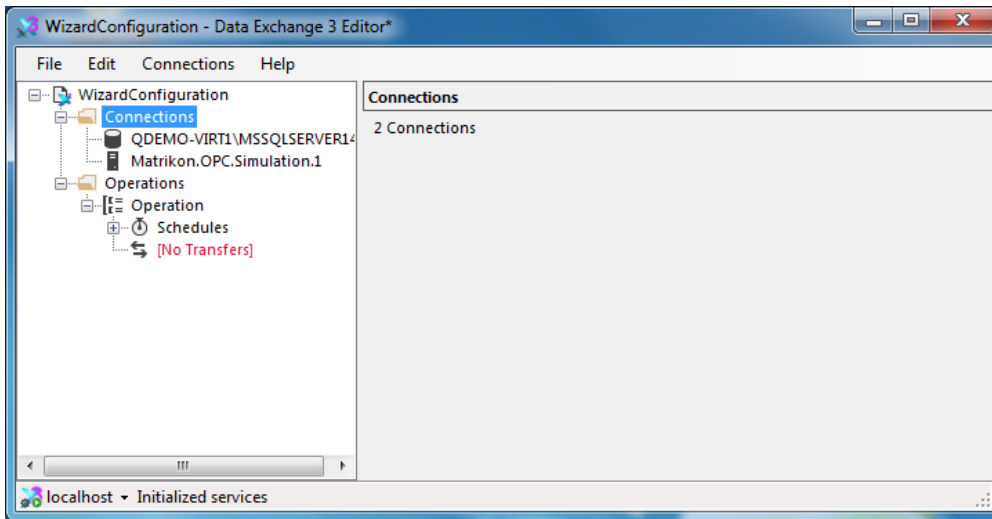
Remote OPC Server/Remote OPC DCOM

Enter the computer name or IP address to view the OPC servers on a remote computer.



Add Operation

All that's left to create are operations for your Data Exchange configuration. Click Operation to create an empty operation then Done.

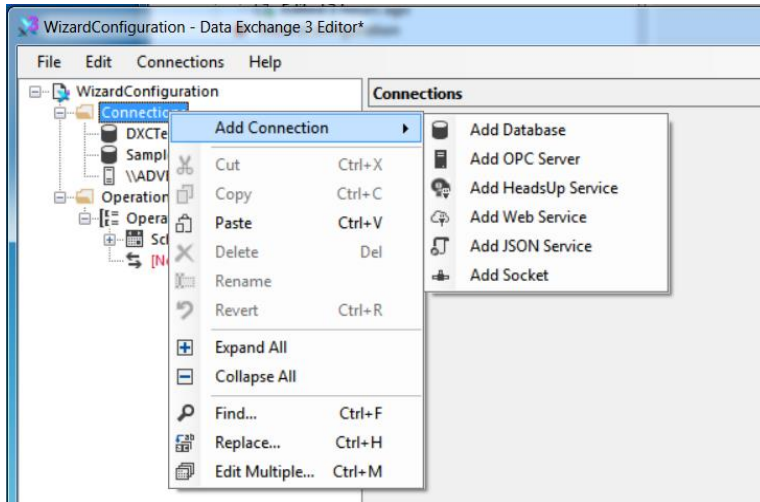


4.8. CONNECTIONS

4.8.1 DATABASE

To create a database connection right-mouse click on “Connections” or select menu Connections -> Add Database.

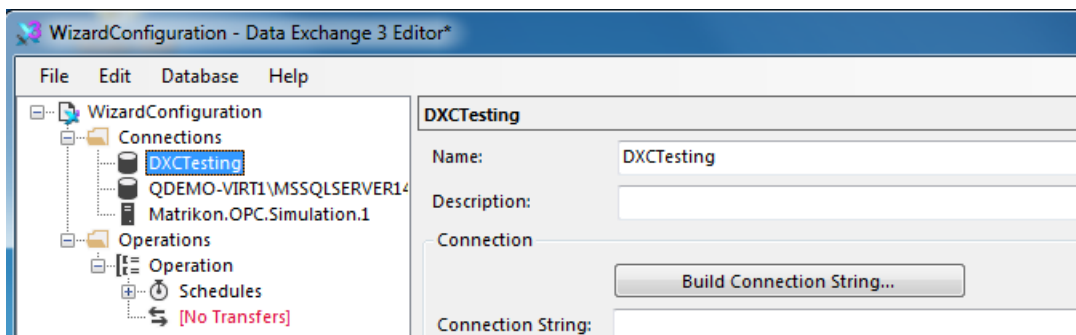
The user may create as many database connections as needed. Each database connection has an alias which is used to reference the database.



Type a name for the database connection and press enter to begin configuring the connection. To communicate with SQL Server, use the SQL Server Native Client 11.0 or Microsoft OLE DB Provider for SQL Server. See the Wizard section of the manual to see the Native Client used. Here we will use Microsoft OLE DB Provider.

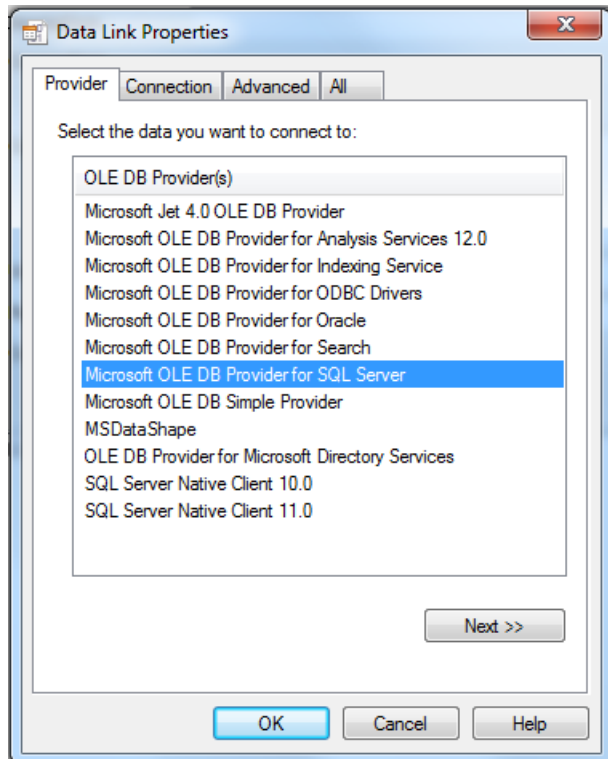
A. Build the Database Connection

Press “Build Connection String” to build the connection. Select “Microsoft OLE DB Provider for SQL Server” to create a connection to a SQL Server database and press Next to continue.



- Provider

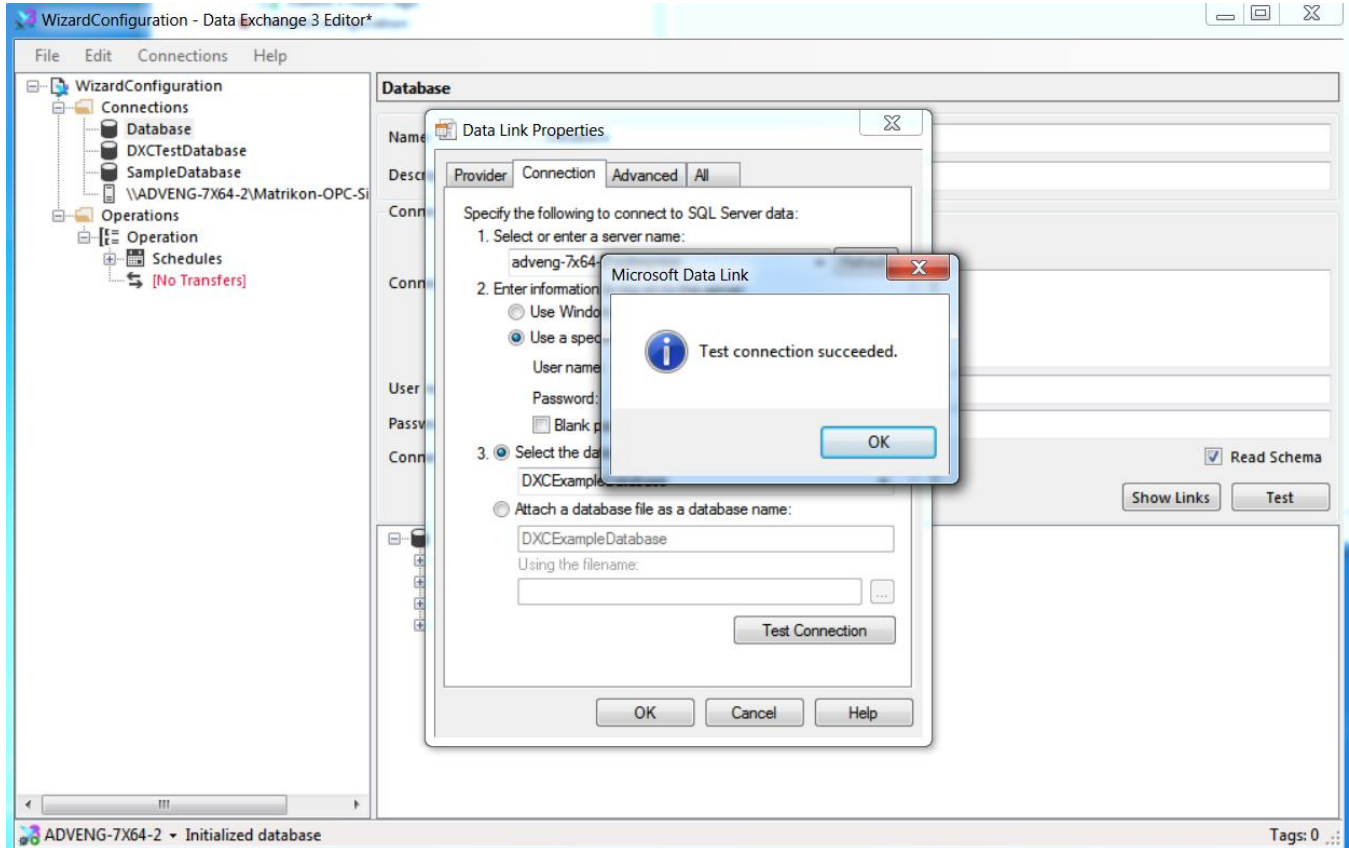
A data provider is used for connecting to a database, executing commands, and retrieving results. Once a provider is selected, a connection is defined and stored within the Configuration file. Passwords stored in the Configuration file are encrypted.



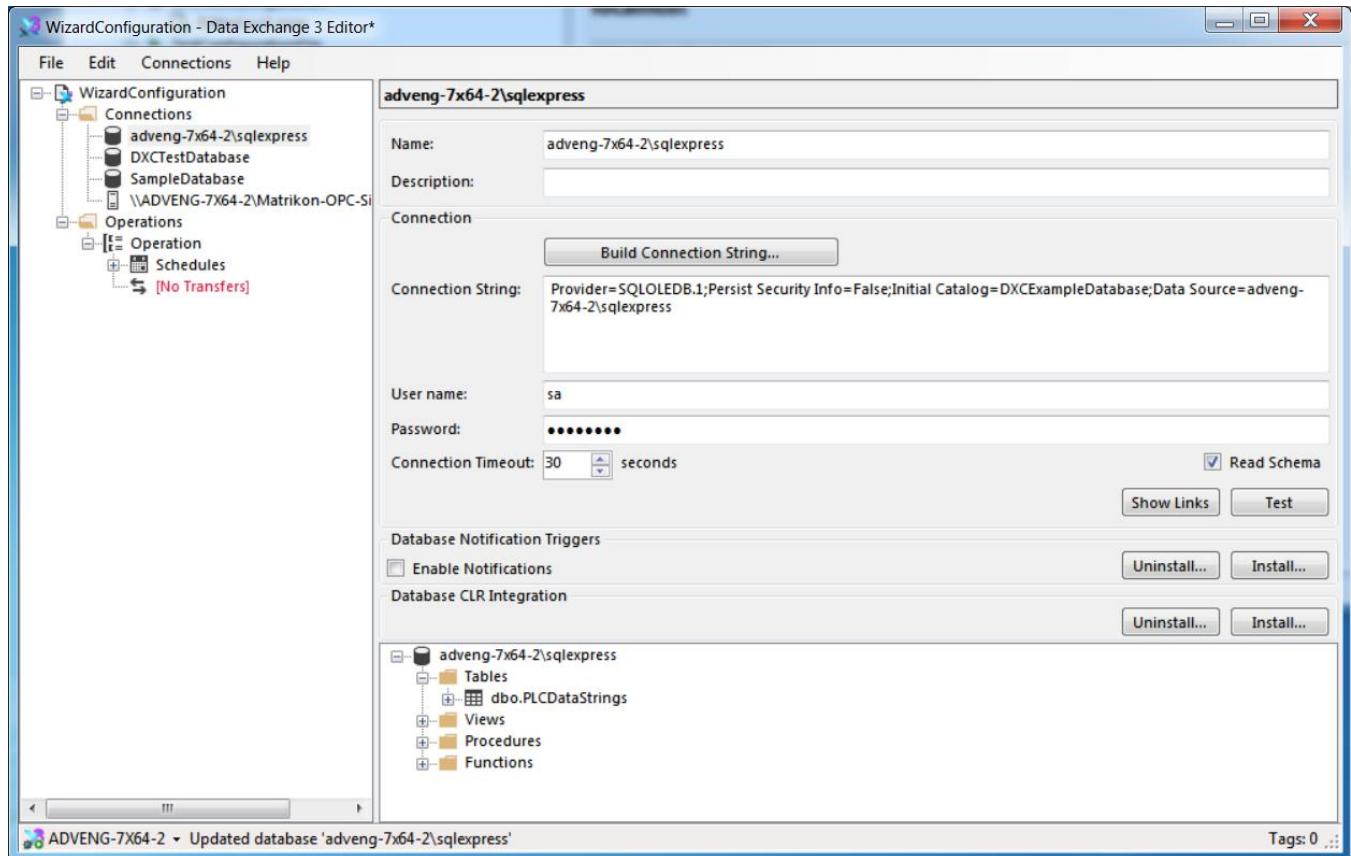
- Connection

Enter the database name, user name and password. DXC does not carry over the username and password from the database configuration window UNLESS “Allow saving password” is clicked. If “Allow saving” is not clicked, entering the username and password in this screen will not affect what is defined in the previous screen and will not be stored as part of the connection string.

Use the Advanced and All tabs to configure any additional settings. Test the connection and press OK. All data will be stored into the configuration for the database. Pressing OK returns focus to the DXC Editor.



If “Allow saving password” was not clicked, continue to define the connection by entering the Username and Password for the connection. Even though it appears the password has been passed in from the connection page, it is not. This is to maintain a tighter password security scheme.



- Read Schema

By checking the Read Schema box DXC will allow you to view the database schema in the Database Monitoring portion of the window.

Disable Schema Reading prevents the Data Exchange Manager from extracting the database schema and saving it into the configuration file. This is a useful feature when Data Exchange connects to a database that has many tables with multiple columns per table. In this case, the schema information can seriously increase the size of the configuration file with irrelevant information. The disadvantage of this selection is the absence of information needed to show columns and data types for simple transfers. If you plan to create only advanced transfers in your configuration, do not select this check box.

To refresh the schema, re-click the Read Schema button.

- Connection Timeout

Data Exchange uses the **Connection Timeout** property to terminate the connection to the database if the database is not available.

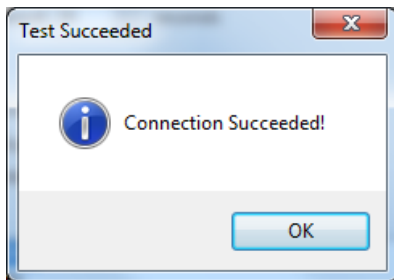
If the Connection Timeout is larger the timeout set for the Operation, the operation timeout is the timeout used for the connection.

- Show Links

Selecting Show Links will highlight every transfer that is using this database connection. To clear the highlights, select "Clear Highlights" from the Configuration menu.

- Test

Tests the connection to the database with the username and password entered on the Database configuration page.



- Database Notification Triggers

Click Enabled Notifications Turns on the ability to use Database Notifications as a trigger option for operations. **NEED MORE DETAIL HERE**

Select **Install** to generate the SQL scripts that must be executed within the SQL Server to allow Database Notifications to be used as an operation trigger. Select **Uninstall** to generate a script to turn off Notifications.

- Database CLR Integration

NEED MORE DETAIL HERE . Select **Install** to generate the SQL scripts that must be executed within the SQL Server to allow CLR Integration. Select **Uninstall** to generate a script to turn off CLR Integration. Requires SQL Server 2012 or greater.

4.8.2 OPC SERVER

The user may create as many OPC connections as needed.

When browsing for OPC Servers, Data Exchange 3 is using OPC Servers browsing capabilities. Some old OPC Servers do not have that feature and will not show up in the list.

When “Use Registry Scan” check box is selected Data Exchange 3 will scan the computer(s) Registry instead of OPC Servers. This is slower than to scan servers, but can give you more complete picture, because some servers (especially the old ones) do not have browsing capabilities.

“Minimize number of OPC groups” check box is dealing with OPC group organization. When this box is NOT SELECTED Data Exchange 3 creates as many OPC group as there are Transfers and OPC Triggers in your configuration.

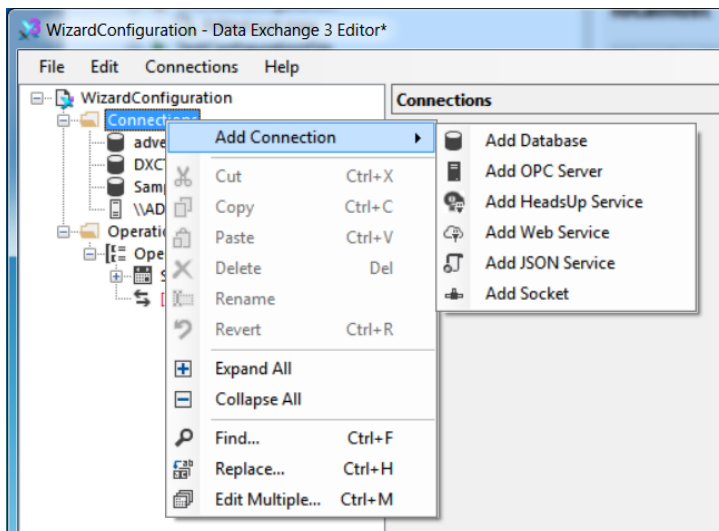
When this box is SELECTED Data Exchange 3 tries to minimize the number of OPC Groups presented to the OPC Servers by grouping items based on the update time interval and access method. Select this box if you have OPC Performance problems when dealing with large configurations.

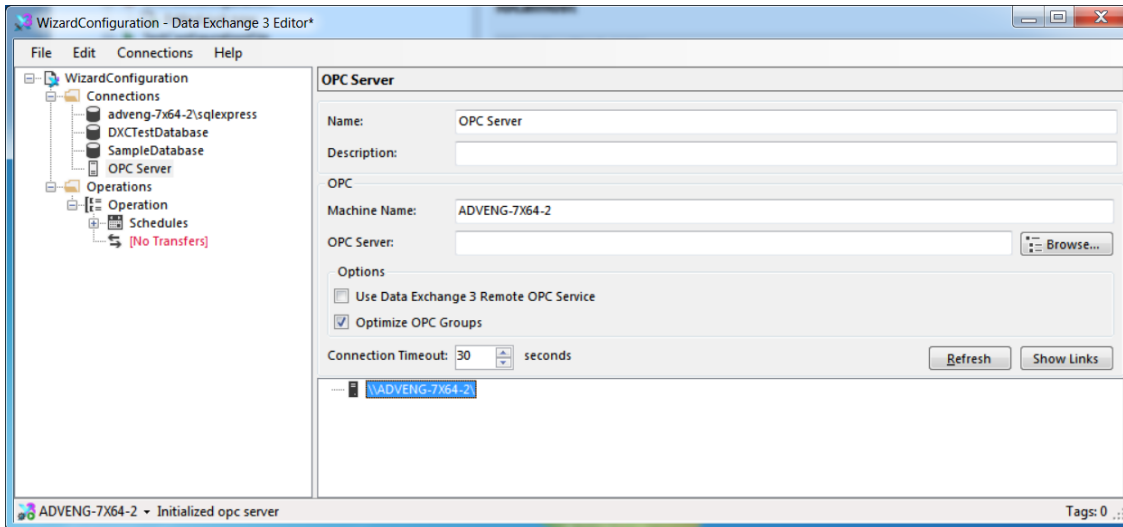
IMPORTANT! When accessing remote OPC servers, make sure that the Data Exchange 3 service is set to log on with an account that has permissions to access the remote computer. For more information about accessing remote OPC servers see the section “Remote OPC Connections” later in this manual.


Data Exchange 3 can connect to remote OPC servers using Microsoft’s DCOM technology. DCOM uses Windows security to validate requests from remote computers. This security is configured with the program **Distributed COM Configuration Properties (DCOMCNFG)**. To launch this program type dcomcnfg in the **Run** dialog accessible from the **Start** menu.

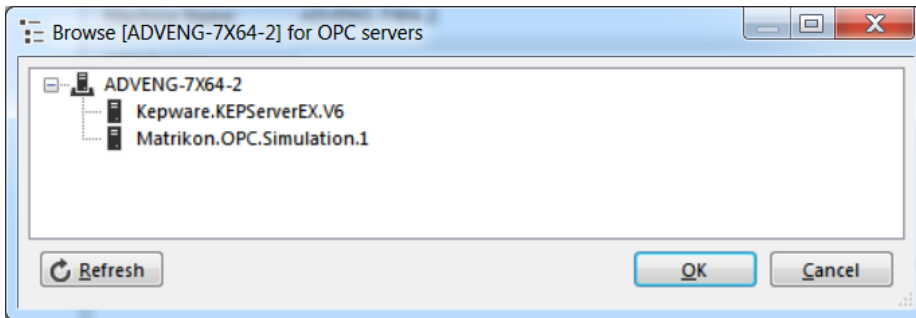
A. Add an OPC Server

An OPC Server is a software application that responds to requests for information and provides data to one or more OPC Clients based on a standard protocol to facilitate interoperability between systems.

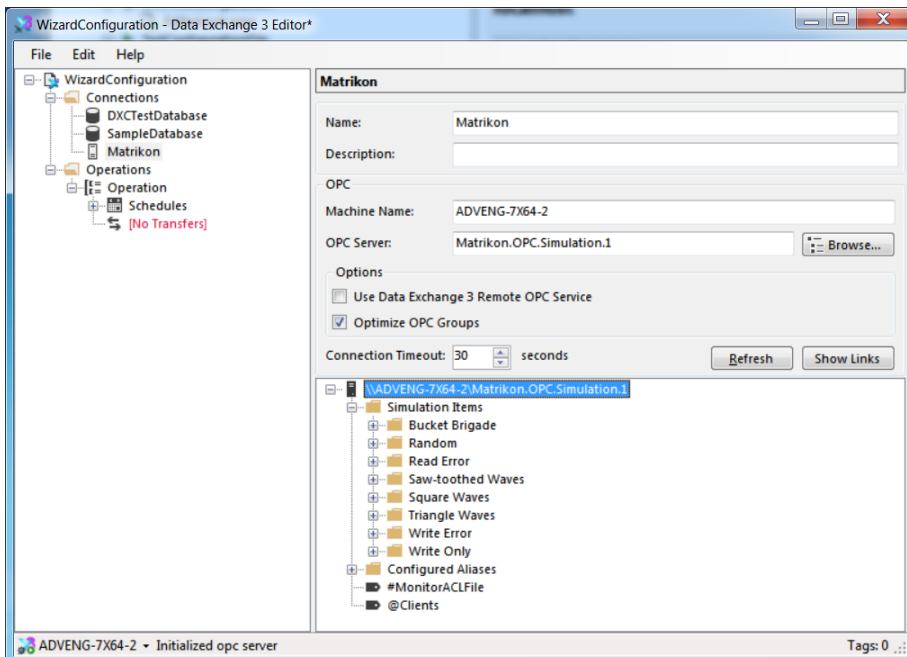




Click the  button and select an OPC Server.



For this example, we have connected to the Matrikon OPC Server that comes as part of the install for Data Exchange.



B. Use Data Exchange 3 Remote OPC Service**NEEDS MORE INFO****C. Optimize OPC Groups**

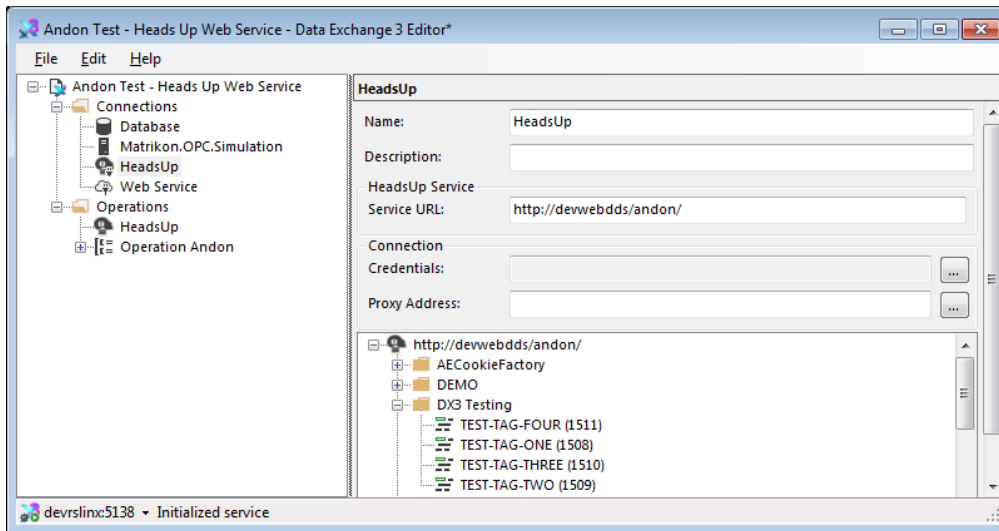
Some OPC Servers such as RSLinx do not like many OPC Groups to be created. Instead it prefers many tags, few groups. Other OPC Servers prefer many groups, few tags. Optimize OPC Groups tells Data Exchange to create as few groups as possible.

D. Connection Timeout**NEEDS MORE INFO****E. Show Links**

Selecting Show Links will highlight every transfer that is using this OPC Server. To clear the highlights, select “Clear Highlights” from the Configuration menu.

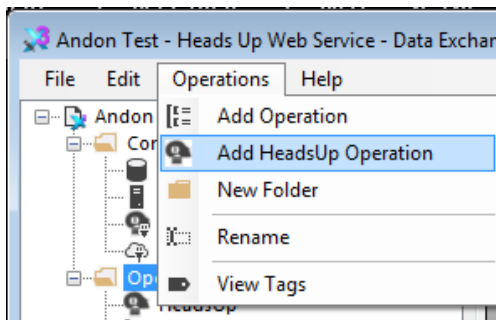
4.8.3 HEADSUP SERVICE

A. HeadsUp Service

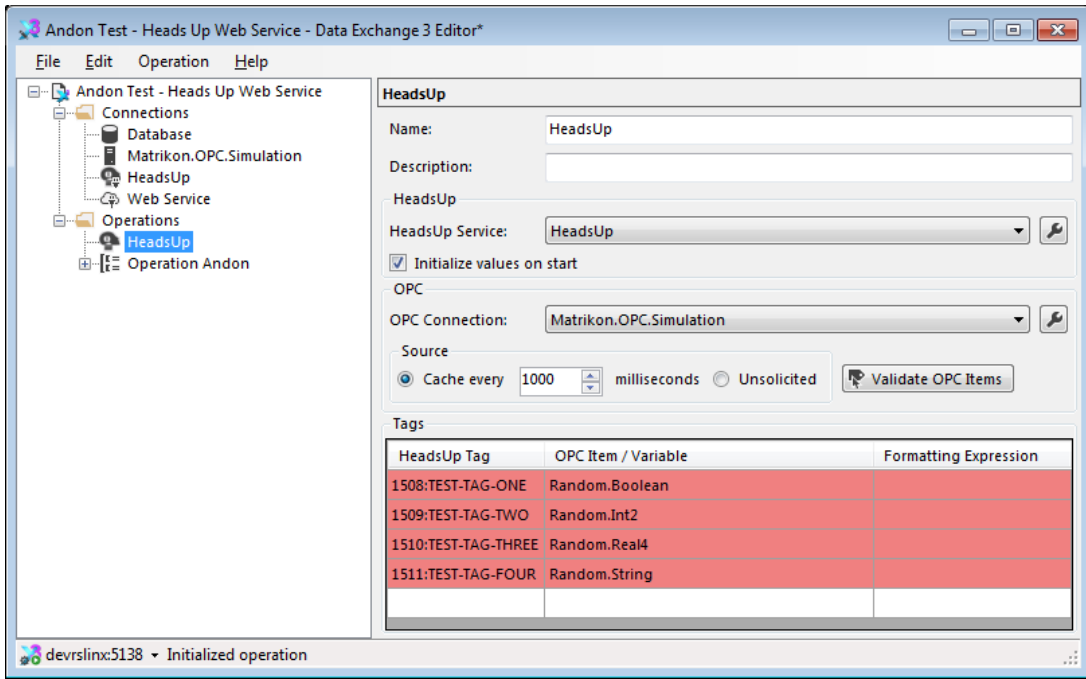


Data Exchange 3 includes a connection to the Advanced Engineering **Heads Up** system. To configure the **Heads Up** Connection, enter the **Heads Up** Service URL. Add the Connection Credentials and Proxy Address if required. If Data Exchange 3 is able to connect to **Heads Up**, the bottom right window will browse the available tags provided by **Heads Up**.

B. Heads Up Operation



To add a **Heads Up** operation, select **Add HeadsUp Operation** from the menu.

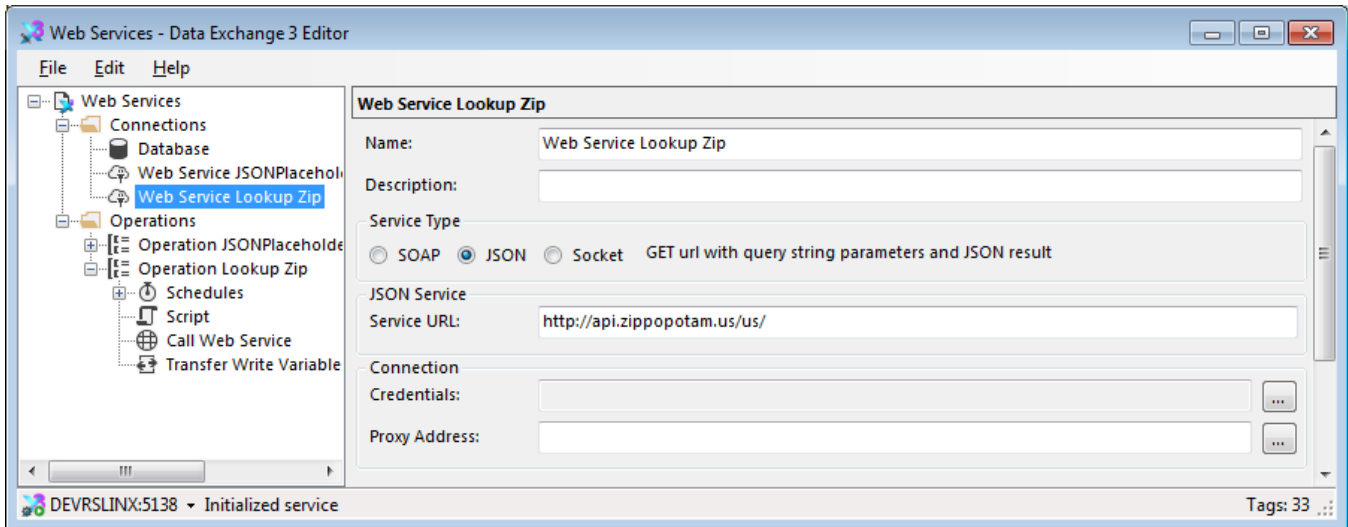


Add the OPC Items or Variables to be transferred to the **Heads Up** tags.

4.8.4 WEB SERVICE

? ?

An Example of a JSON web service. SOAP and Socket connections can also be used.



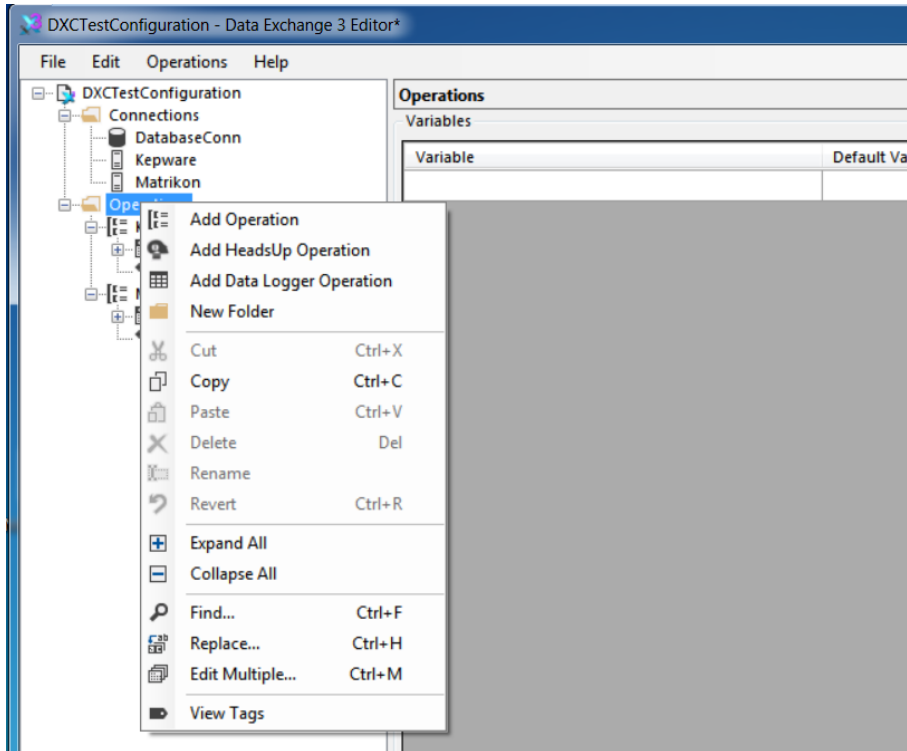
4.8.5 JSON SERVICE

4.8.6 SOCKET

4.9. OPERATIONS

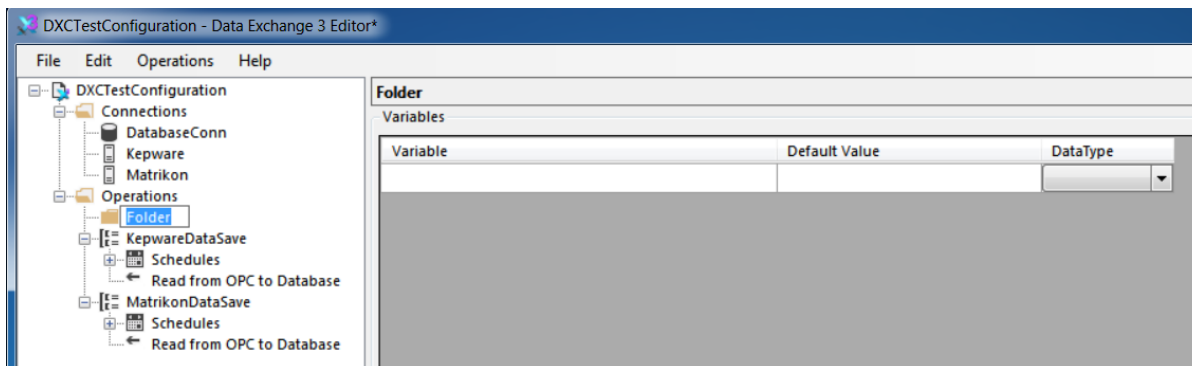
Operations are blocks of work in the configuration file. Each Operation has a trigger and transfers. The trigger initiates the work that is defined by the transfers in the Operation.

Some buttons are disabled if the Engine is not running such as selecting and validating tags. This is because the engine makes the connection to the OPC Servers not the Editor.

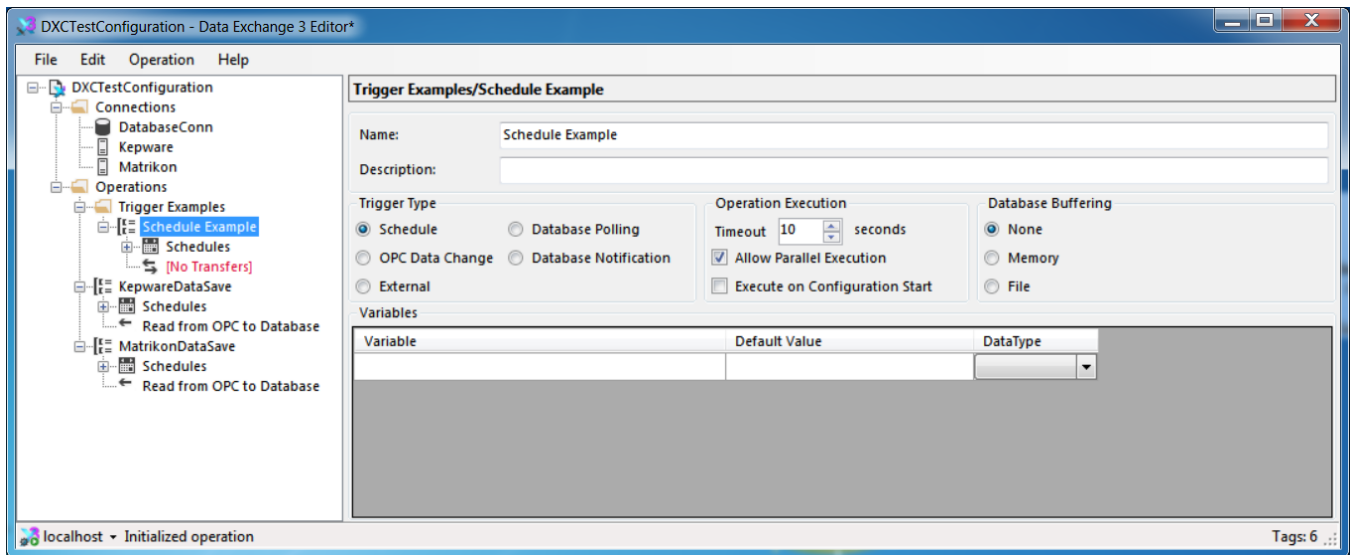


4.9.1 FOLDERS

Data Exchange allows users to group operations into folders to organize. Select New Folder and give the folder a name. Folders may be renamed. If you change a folder name and an operation in that folder is invoked by another Operation using the "Start Operation" transfer type, you will need to point to the new name for the Operation.



Add an operation to a folder or to the main Operations folder by right clicking and selecting Add Operation.



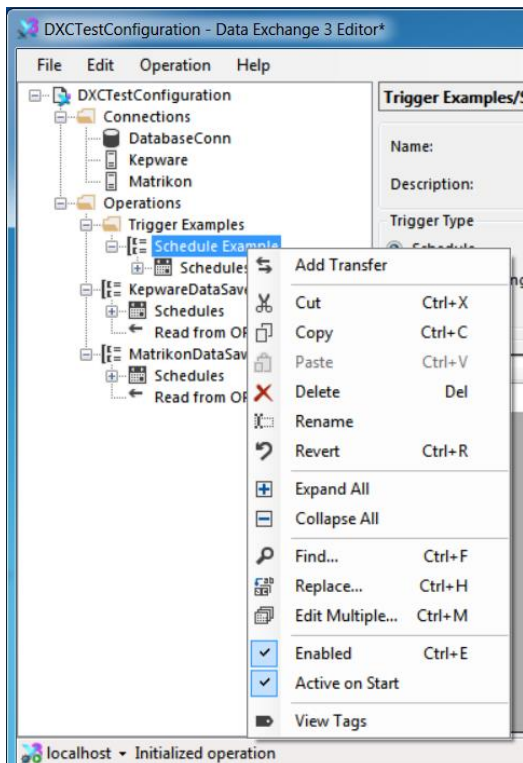
4.9.2 NAMES

Items in Data Exchange can be arbitrary length and have a few limitations. Names cannot contain the following characters and character combinations:

. / ~ :: :=

The maximum length of an event log entry including the path to the item, the name of the item, and the message is 32700.

4.9.3 OPERATION CONTEXT MENU



A. Add Transfer

Creates a transfer for this operation.

B. Edit Multiple

Allows you to edit common properties of multiple items at the same time

C. Enabled

Allows you to enable/disable an operation from running. This Enabled state can only be changed by changing the setting and restarting the configuration.

D. Active on Start

Active on Start allows you to enable an operation but will prevent that operation from being executed until the "Active" checkbox is checked during runtime.

E. View Tags

Creates a Matrikon OPC Client with the tags in the current Operation.

4.10. VARIABLES

Data Exchange supports two types of variables persistent and nonpersistent variables.

4.10.1 PERSISTENT VARIABLES

Persistent variables are loaded at configuration start and are persistent across operation executions. A change in the value by one operation execution will be reflected in the other operation executions and other operations. An optional default value can be set as the initial value when the configuration starts.

4.10.2 NONPERSISTENT VARIABLES

Nonpersistent variables are loaded for each operation instance. There is a new copy of each nonpersistent variable every time the operation is executed and any changes in one instance are not reflected in the other instances. Automatic variables are all nonpersistent.

4.10.3 NAMESPACES

All variables are declared in a namespace. Generally, in order to access a variable in an operation you will not need to use namespaces unless you are trying to access a variable in another operation or folder. If you reference a variable in an operation or script, Data Exchange will search in the operation's namespace for a variable with that name. If it does not find one, it will move up folder by folder searching each for a variable with that name until it reaches the configuration level variables. If the configuration does not have a variable (e.g. `_TriggerTime`), a null will be returned and a message will be displayed in the Event Viewer: `Couldn't find variable [_TriggerTime]`

Any operation can access any persistent variable in the configuration by using the variable's namespace. In order to reference a variable called `TotalCount` in an operation called `Main` that is in a folder called `First`, you would put `config/First/Main::TotalCount` for the variable name. The part before the double colon is the namespace and everything after that is the variable name.

When execution is passed to a different operation, currently available variables are passed also.

4.10.4 DEFAULT (AUTOMATIC) VARIABLES

A. Configuration Specific Variables

\$_Configuration

The name of the configuration (string).

\$_ConfigurationStartTime

The last start time of the configuration file (DateTime).

\$_ConfigurationModified

The last time the configuration file was modified (DateTime).

\$_MachineName

The name of the computer that is running DXC3 (string).

\$_EngineVersion

Version of the DXC Engine currently running this configuration file (Version).

\$CurrentTime

The current time on the computer running DXC3 (DateTime).

B. Operation Specific Variables

\$_Operation

The name of the current operation (string).

\$_OperationInstance

The instance number of the execution of this operation (long).

\$_OperationStartTime

The start time for the operation (DateTime).

C. Transfer Specific Variables

\$_Transfer

The name of the current transfer (string)

D. Trigger Specific Variables

Every operation has a default set of variables available for use in transfers, however every Trigger type does not produce a value in every variable. For example, the variable `_TriggerTime` is only available for operations running on a Schedule.

All Triggers

\$_TriggerName

The name of the trigger that initiated the current instance of the Operation. For schedule triggers it will be Schedule: plus the name of the schedule. For operations triggered by an Operation Start transfer it will be Operation: plus the name of the operation. For operations triggered from a script it will be the full name of the script transfer. For operations triggered from the web interface or externally, it will be Web. For OPC Data Change, Database Notification, and Database Polling it will be the type of trigger (string).

Schedule

\$_TriggerTime

The time the operation triggered to start. This variable is available for operations that run on a recurring timer (DateTime).

\$_FromMidnightMS

The number of milliseconds elapsed since midnight (int).

\$_IntervalMS

The number of milliseconds elapsed since the last time the engine started (long).

\$_FirstInterval

If this is the first time the Schedule has triggered (bool)

OPC Data Change

\$_Trigger

The value of the OPC item that triggered the operation

External

\$Param1, \$Param2, \$Param3, \$Param4, \$Param5, \$Param6, \$Param7, \$Param8, \$Param9, \$Param10

Parameters used for External Operation Triggers.

As an example, Variable1 is defined within an Operation, a Folder and as a Global which is accessible by all operations. The table below displays the runtime results for operations using this variable.

Failure

\$LastError

The last Exception that occurred in the Operation. This is only set on a transfer Failure and can be accessed in the OnError transfer or any transfer executed after that (Exception).

\$LastErrorMessage

The Message of the last Exception that occurred in the Operation (String).

\$LastErrorType

The Type of the last Exception that occurred in the Operation (String).

E. Configuration Variables

Variable1 = "This is the Global Value"

Folder1

Variable1 = "This is the Folder1 Value"

Operation1

Variable1 = "This is Operation1 Value"

Operation2

Folder2

OperationA

OperationB

OperationII

Operation	Value of Variable1
Operation1	This is Operation1 Value
Operation2	This is the Folder1 Value
OperationA	This is the Global Value
OperationB	This is the Global Value
OperationII	This is the Global Value

F. Legacy Variables From Previous Versions of Data Exchange

Compatibility with previous versions (II, 2.0, 2.5) of Data Exchange for the following legacy variables is supported by allowing them in Data Exchange 3. If previous versions of Data Exchange configurations are imported, they may include these legacy variables.

DXC.CurrentTime
 DXC.Parm1
 DXC.Parm2
 DXC.Parm3
 DXC.Parm4
 DXC.Parm5
 DXC.Parm6
 DXC.Parm7
 DXC.Parm8
 DXC.Parm9
 DXC.Parm10

4.10.5 NAMING VARIABLES

To avoid conflict with the internal Data Exchange variables and to avoid confusion as to which variables are local to an operation or are global, etc., follow a naming convention different from the internal variables.

EG:

Global Variables: g_VariableName or gVariableName

Folder Variables: f_VariableName or fVariableName

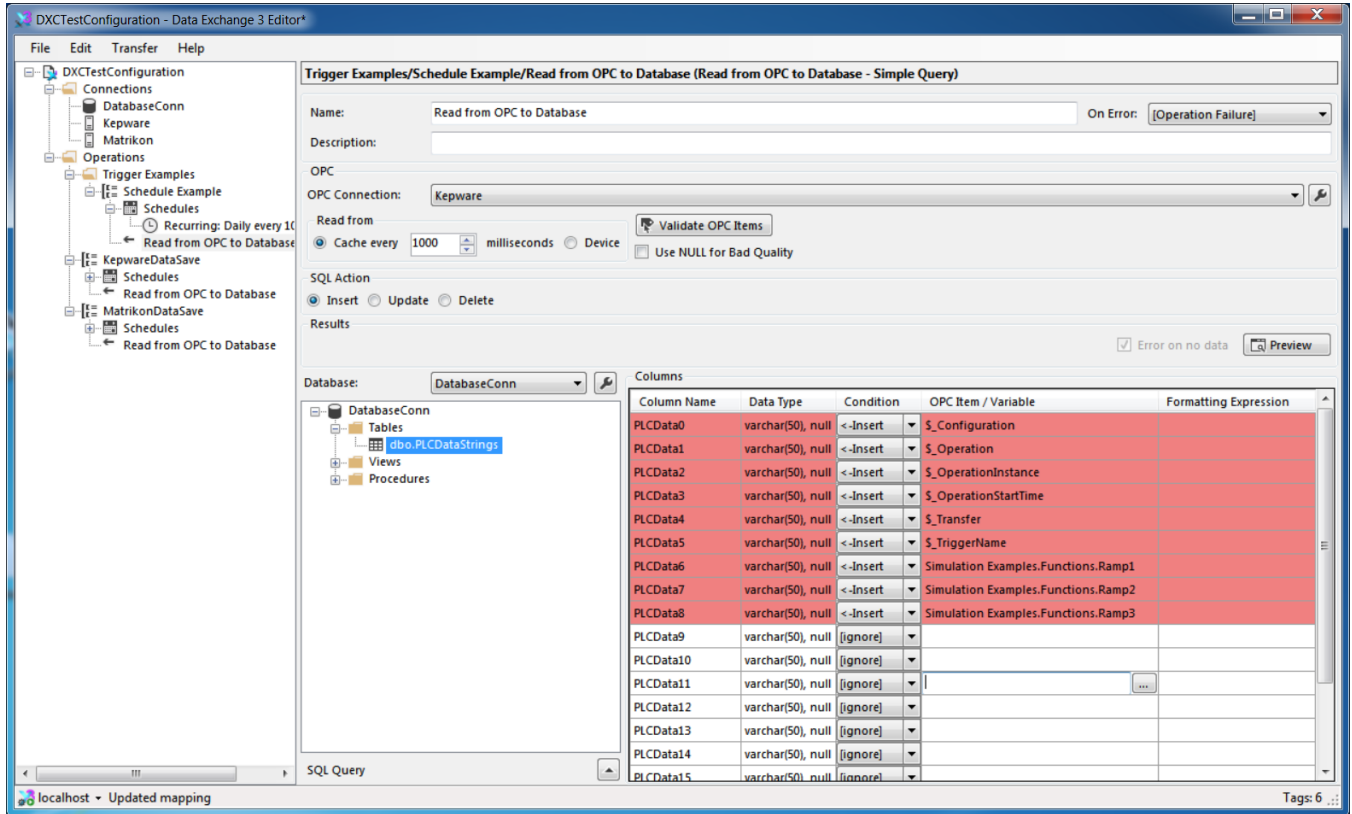
Operation Variables: o_VariableName or oVariableName

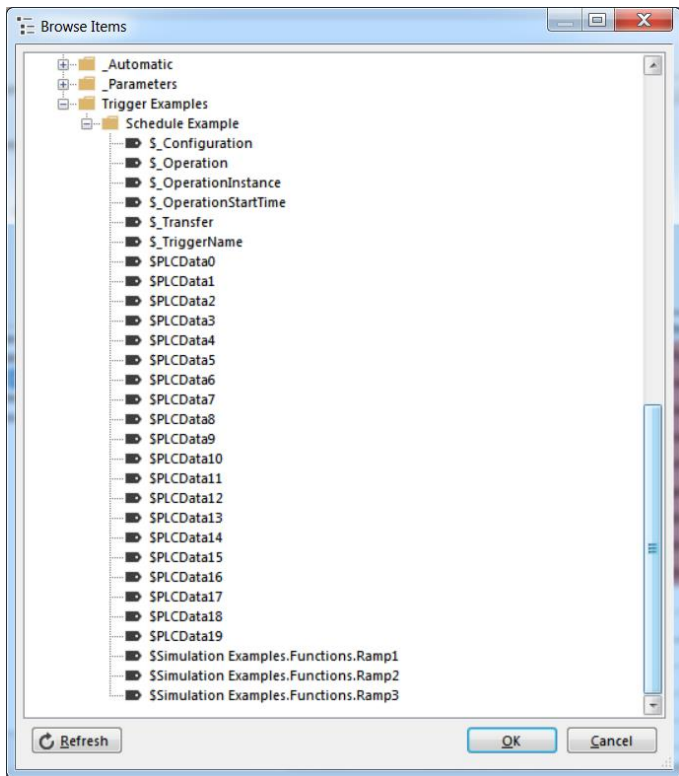
If some variables will act like constants, keep those variables upper case to distinguish them from modifiable variables. For example, a global constant would look like g_CONSTANTNAME or gCONSTANTNAME.

4.10.6 LOCAL RUNNING VARIABLES

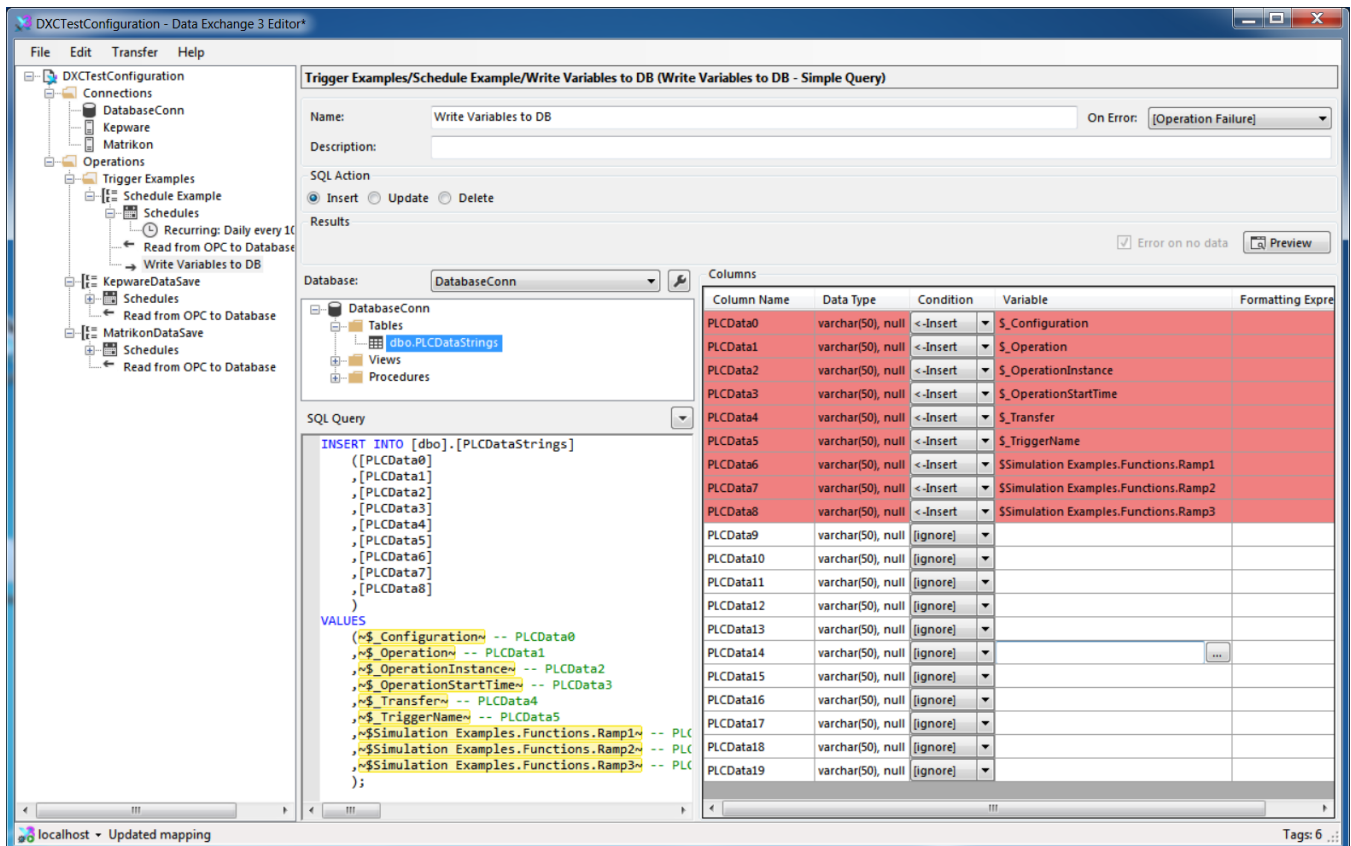
Certain Transfer types will create variables that can be used in the Operation. Simple OPC Read creates runtime variables for every OPC Tag that is read during the transfer.

In this example, OPC Tags are read during the Simple OPC Read transfer below. Variables representing the OPC tags are created and can be used in subsequent Transfers.





Next, create a “Write Variables to DB” transfer. This transfer has access to the internal variables created to contain the values of the OPC tags from the previous transfer.

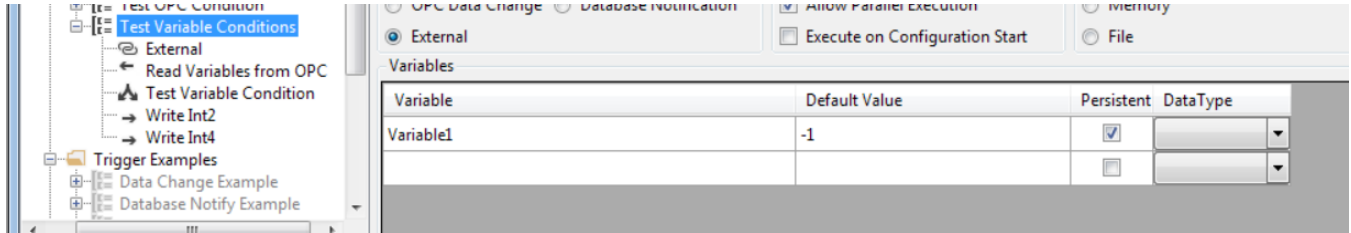


4.10.7 CREATING VARIABLES

Variables may be created at the Operation level, the Folder level and Configuration level.

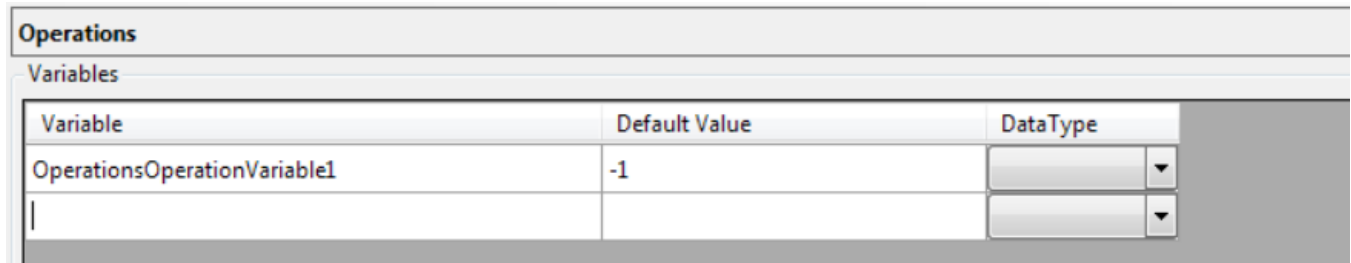
A. Operation Variables

Operation variables are “local” variables; that is variables that get re-initialized every time an Operation is invoked. The Persistent checkbox allows you to keep the value of the variable from the previous execution of the Operation. Unselect this checkmark to re-initialize the variable every time the Operation runs.

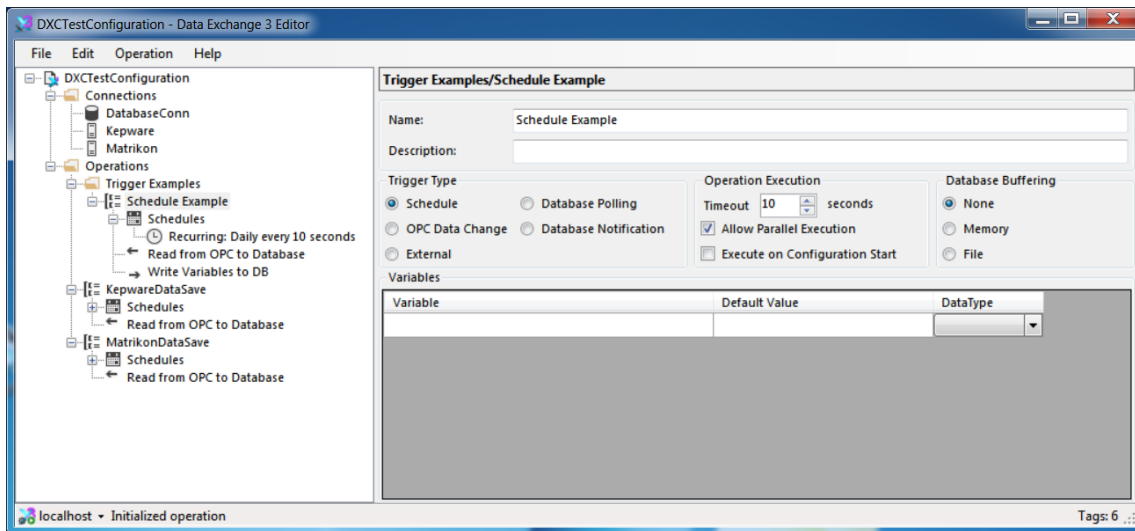


B. Folder Variables/Configuration Variables

These variables are “global” variables and that they maintain the last value.



4.11. GENERAL OPERATION SETTINGS



A. Trigger Types

There are a variety of ways to trigger a DXC operation. The Trigger section outlines each trigger and associated parameters.

B. Operation Execution Timeout

The maximum amount of time it should take an operation to complete. If an operation hangs Data Exchange will terminate the operation and the status of the operation will be set to failed.

If the Connection Timeout is larger than the timeout set for the Operation, the amount of potential time left to execute is used as the timeout for the connection.

C. Parallel Execution

Parallel execute allows multiple operations to execute at the same time. Data Exchange will not wait for the previous operation to finish before it begins execution of the next. During asynchronous mode, operations can finish out of order. **Only enable Parallel Execution if multiple asynchronous executions will not cause issues in your configuration.**

If parallel execution is disabled, a second operation will not start and therefore will be dropped. The drop status is recorded on the Operations tab for the configuration:

Operation	Last Status	Run	Succeeded	Failed	Timed Out	Dropped	Run Time	Run Time (avg)	Run Time (max)	Run Time (min)	Verbose	Active
ControlPause	Success	1	1	0	0	1	10.017	10.017	10.017	10.017	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ControlStartOperation	None	0	0	0	0	0	0.000	0.000	0.000	0.000	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ControlTestConditions	None	0	0	0	0	0	0.000	0.000	0.000	0.000	<input type="checkbox"/>	<input checked="" type="checkbox"/>

The number of operations that can be performed at the same time is based on system resources. If the engine runs out of resources, error messages will be displayed announcing operations cannot execute.

D. Execute on Configuration Start

Triggers the Operation to run every time the configuration is started.

E. Database Buffering

Data Exchange can buffer data destined for a database if it cannot connect to the database.

None

Database buffering is disabled.

Memory

Database buffering is stored internally within Data Exchange. If Data Exchange is restarted the buffered data is deleted and not sent to the database.

File

Database buffering is stored in a file in the Data Exchange runtime directory. If Data Exchange is restarted the buffered data remains. Once a connection is established with the database, the data is transferred to the database.

4.12. TRIGGER TYPES

There are several types of triggers for DXC Operations. These are:

- **Schedule:** Triggers an operation on an interval when the configuration file starts or at a specific time/date. Multiple times to trigger an operation may be defined if using Schedule.
- **OPC Data Change:** Triggers an operation when an OPC tag changes value or when it meets a specific condition.
- **Database Polling:** Triggers an operation based on input from the database.
- **Database Notification:** Triggers are generated by Data Exchange 3 using a Subscription to SQL Server Notification Services. This allows triggers to be created by Transact-SQL statements. Requires SQL Server 2008 or greater.
- **External:** Triggers are generated by other programs that communicate directly with the Data Exchange 3 Engine. The program will specify the required Data Exchange 3 Configuration, Operation, and Parameters.

Operations can only have one trigger. If the same Operation must be triggered on multiple conditions, the Start Operation Transfer type can be used to invoke the Operation from other Operations. More on this in the Transfers portion of the manual.

4.12.1 SCHEDULE

Several types of time-based Schedules are available in Data Exchange. The Schedule Trigger can have multiple Schedules. Multiple types of Schedules can be combined in the same Schedule Trigger.

One Time

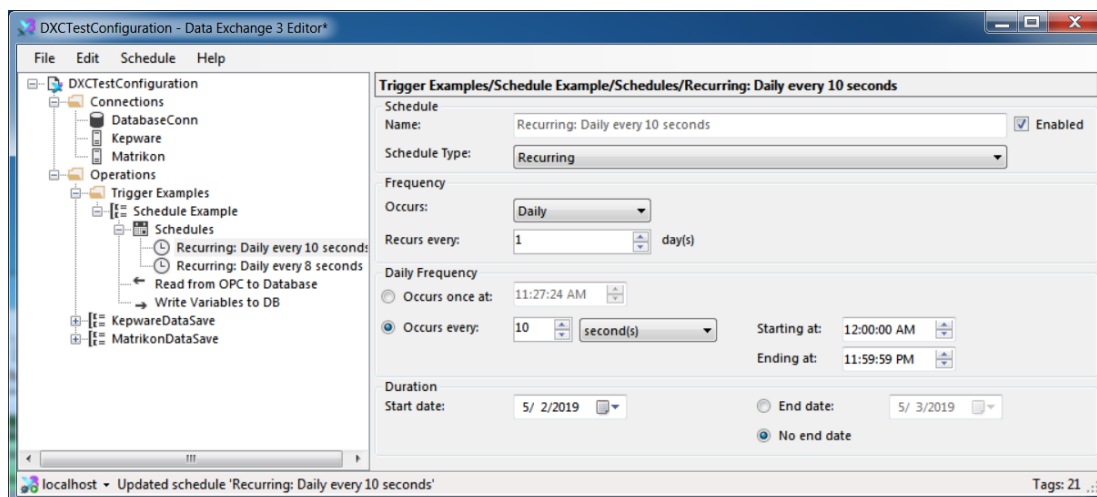
Triggers can be scheduled for specific dates and times. For example, if an event must happen at 10pm on the last day of the month, create a trigger for January 31 at 10pm, February 28 at 10pm etc.

Configuration Start

Trigger occurs only when a configuration starts in the engine.

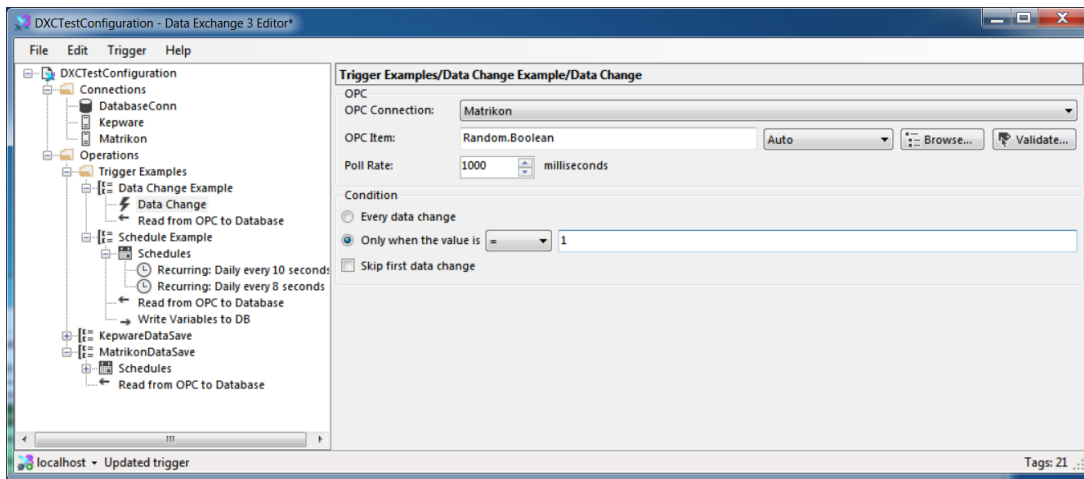
Recurring

A recurring event is any event that occurs more than once, according to a pre-defined rule such as weekly, daily, or hourly.



4.12.2 OPC DATA CHANGE

Trigger an operation based on the value range or specific value of an OPC Tag or the change event of that OPC Tag.



Browse

Browses the tags in the OPC Server and allows you to select one to use in this trigger. An error will occur if Data Exchange cannot connect to the OPC Server.

Note: Some OPC Servers allow you to select a folder name while others do not. Data Exchange does not restrict what is selected so take caution in not selecting something invalid.

Validate Tag

Verifies that the OPC Tag is good and Data Exchange can read the value.

Note: Some OPC Servers will always return that all tags are valid regardless of whether they are valid or not.

Poll Rate

The duration between checks for the trigger condition for an Operation. The minimum poll rate is dependent on the OPC Server.

Every data change

Trigger the operation any time the value of the OPC tag changes.

Only when the value is

Trigger the operation when the value of an OPC tag equals a specific value or if one of the following conditions is true:

=, <>, >, >=, <, <=

Skip first data change

When DXC first starts the value for a tag goes from "Unknown" to having a value. This will trigger an operation to run. Click this button to disable this feature.

OPC Trigger Tag

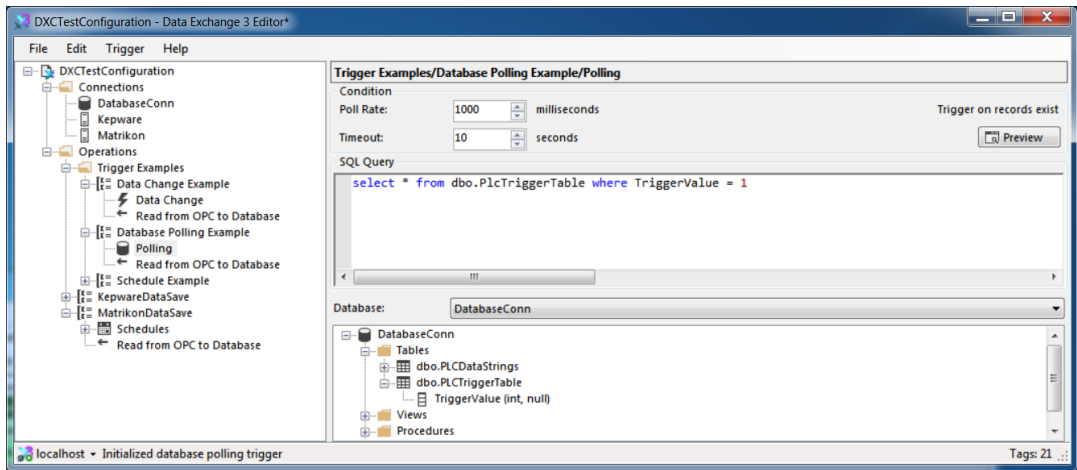
The variable `$_Trigger` can be used in the subsequent transfers of the trigger instead of re-reading the OPC Tag. The variable `$_Trigger` will contain the value of the OPC Tag at the time of the trigger.

Special Note

If you change the Operation's Trigger Type to a different type of trigger, the OPC Tag will be cleared.

4.12.3 DATABASE POLLING

Triggers an operation if a query statement returns at least one row. In this example, as long as the TriggerValue is 1, the operation will continue to execute every Poll Rate milliseconds.



Poll Rate:

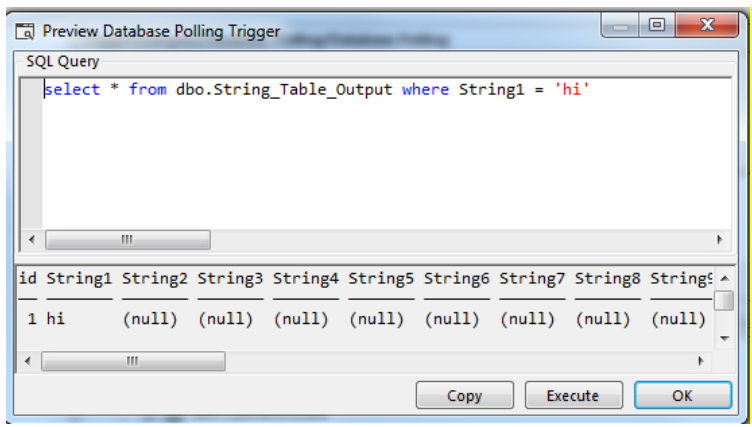
The frequency at which the database is queried. The minimum is 50ms.

Timeout:

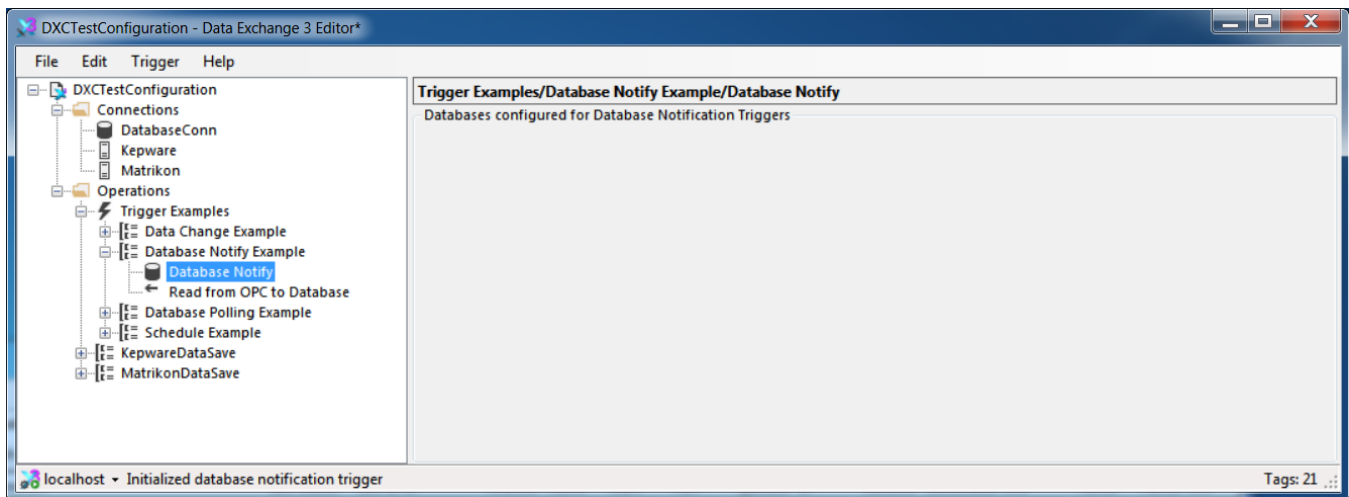
The maximum amount of time for the query to execute.

Preview:

Allows the user to view and execute the polling trigger query.



4.12.4 DATABASE NOTIFICATION

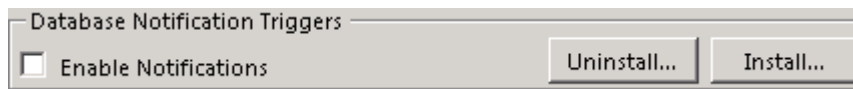


Description:

Database Notification Triggers allow the database to trigger a Data Exchange Operation with minimal delay. Any or all of the default parameters (Param1 – Param10) along with any other Data Exchange variables can be passed to the Data Exchange Operation.

Requirements:

Database Notification Triggers must be installed and enabled. This is done in the database connection configuration of the Data Exchange 3 Editor.



Triggers are generated by executing the stored procedure **dx3.usp_TriggerOperation**, and passing the appropriate parameters to the procedure. The example below shows how to execute **dx3.usp_TriggerOperation**, and the example can be used if you want to pass individual parameters without using XML.

Example stored procedure to create a Database Notification Trigger using individual parameters:

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[usp_TriggerOperationIndividualParams](
    @machine_name          [nvarchar](255),
    @configuration_name    [nvarchar](255),
    @operation_name        [nvarchar](max),
    @Param1                NVARCHAR(MAX) = NULL,
    @Param2                NVARCHAR(MAX) = NULL,
    @Param3                NVARCHAR(MAX) = NULL,
    @Param4                NVARCHAR(MAX) = NULL,
    @Param5                NVARCHAR(MAX) = NULL,
    @Param6                NVARCHAR(MAX) = NULL,
    @Param7                NVARCHAR(MAX) = NULL,
    @Param8                NVARCHAR(MAX) = NULL,
    @Param9                NVARCHAR(MAX) = NULL,
    @Param10               NVARCHAR(MAX) = NULL
)
AS
BEGIN
    DECLARE @Variables XML;

    SET @Variables =
    (
        SELECT  @Param1 AS Param1,
                @Param2 AS Param2,
                @Param3 AS Param3,
                @Param4 AS Param4,
                @Param5 AS Param5,
                @Param6 AS Param6,
                @Param7 AS Param7,
                @Param8 AS Param8,
                @Param9 AS Param9,
                @Param10 AS Param10
        FOR XML PATH
    );

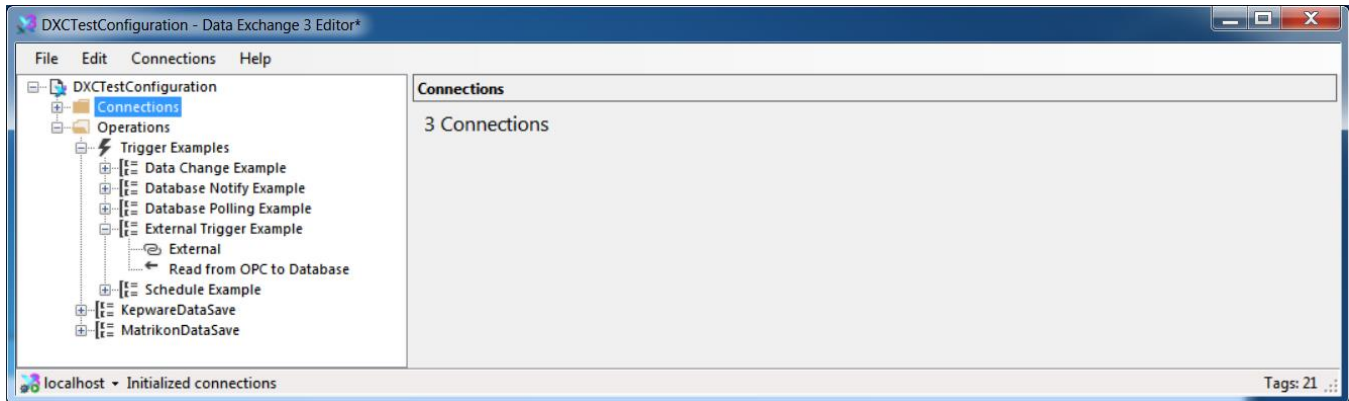
    EXECUTE [dxc3].[usp_TriggerOperation]
        @machine_name
        ,@configuration_name
        ,@operation_name
        ,@Variables;
END

```

4.12.5 EXTERNAL

A. From external program

Include DXC3COM as either a COM reference or a .NET library in your project.



VB.Net Example:

```
'Add Reference to DXC3COM.dll
'Target framework: .NET Framework 4.7.1
'Compile Prefer 32 Bit
Private Sub ExternalTrigger()
    Dim sURL As String = "192.168.10.52:5130"
    Dim Params As Object = New Object() {
        "Param 1 Value",
        "Param 2 Value",
        "Param 3 Value",
        "Param 4 Value",
        "Param 5 Value",
        "Param 6 Value",
        "Param 7 Value",
        "Param 8 Value",
        "Param 9 Value",
        "Param 10 Value"
    }
    Public Shared Function StartOperation(machineName As String, configName As String, operationName As String, values() As Object,
    timeout As System.TimeSpan) As Boolean
        DXC3COM.ExternalTrigger.StartOperation(sURL, "Trigger Testing", "Operation External Trigger", Params, New TimeSpan(0, 0, 10))
    End Function
```

C#.Net Example:

```
//Add Reference to DXC3COM.dll
//Target framework: .NET Framework 4.7.1
//Compile Prefer 32 Bit
private void ExternalTrigger()
{
    string sURL = "192.168.10.52:5130";
    object[] Params = new object[] {
        "Param 1 Value",
        "Param 2 Value",
        "Param 3 Value",
        "Param 4 Value",
        "Param 5 Value",
        "Param 6 Value",
        "Param 7 Value",
        "Param 8 Value",
        "Param 9 Value",
        "Param 10 Value"
    };
};
```

```
//Public Shared Function StartOperation(machineName as String, configName As String, operationName As String, values() as Object,
timeout As System.TimeSpan) As Boolean
    DXC3COM.ExternalTrigger.StartOperation(sURL, "Trigger Testing", "Operation External Trigger", Params, new TimeSpan(0, 0, 10));
}
```

SQL Server Example

```
EXEC sp_configure 'show advanced options', 1
GO
RECONFIGURE
GO
EXEC sp_configure 'Ole Automation Procedures', 1
GO
RECONFIGURE
GO
EXEC @hr = sp_OACreate 'DXC3COM.External', @object OUT;
```

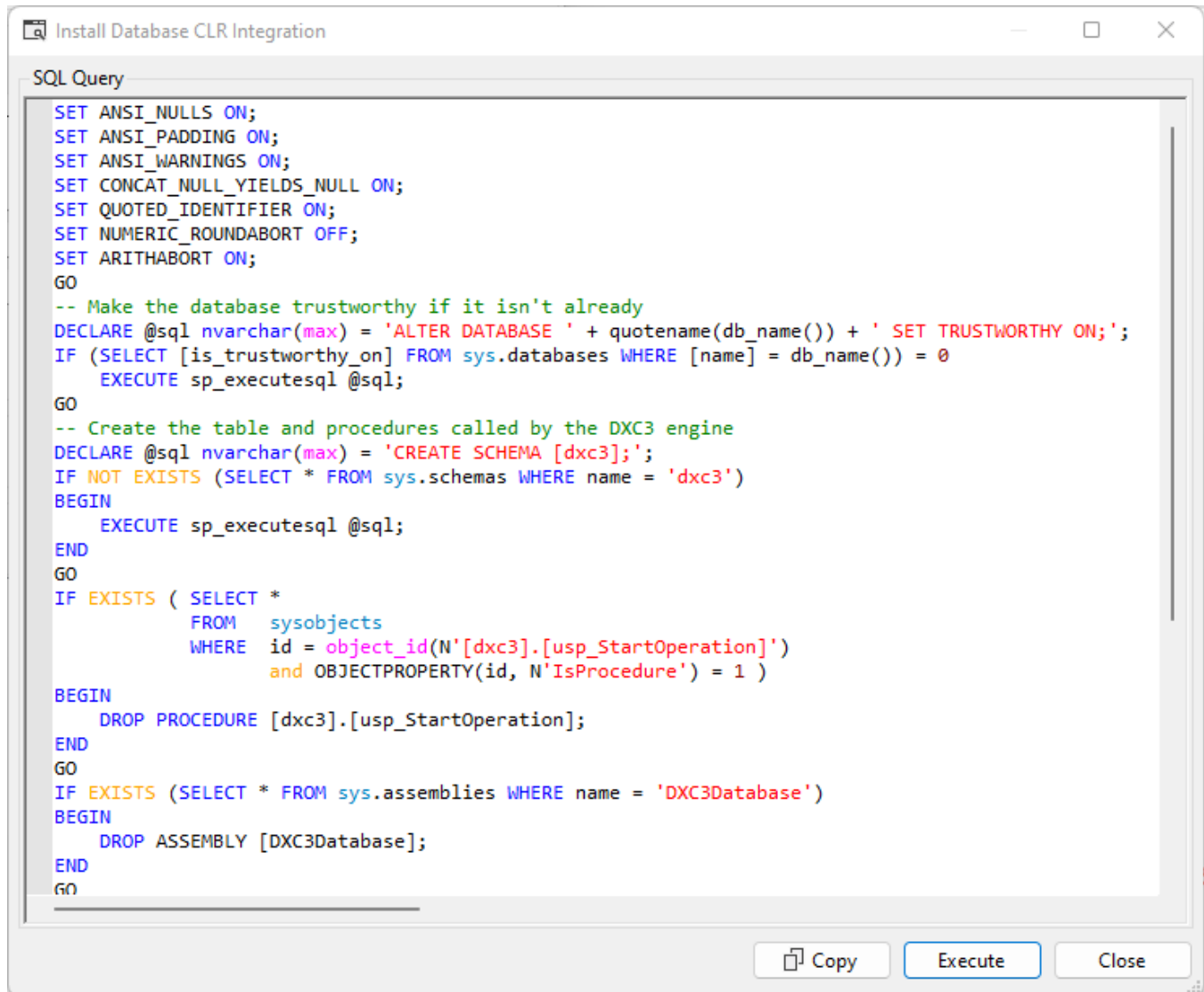

B. From SQL Server

Install the CLR Integration from the Database Connection

The screenshot shows a configuration window for a database connection named 'localhost'. The window is divided into several sections:

- Name:** localhost
- Description:** (empty field)
- Connection:**
 - Build Connection String... button
 - Connection String: Provider=SQLNCLI11.1;Persist Security Info=False;Initial Catalog=DXC3;Data Source=localhost;Initial File Name='';Server SPN=''
 - User name: sa
 - Password: (masked with dots)
 - Connection Timeout: 30 seconds
 - Read Schema:
 - Refresh, Show Links, and Test buttons
- Database Notification Triggers:**
 - Enable Notifications:
 - Uninstall... and Install... buttons
- Database CLR Integration:**
 - Uninstall... and Install... buttons
- Object Explorer:** A tree view showing the 'localhost' server with folders for Tables, Views, Procedures, and Functions.

Clicking “Install...” will open a dialog with an installation script. If the user that is configured for the Database Connection has administrator privileges on the SQL Server, you can click “Execute” and install directly from the Database Connection. If not, copy the script and execute using an administrator account.



```

SQL Query
SET ANSI_NULLS ON;
SET ANSI_PADDING ON;
SET ANSI_WARNINGS ON;
SET CONCAT_NULL_YIELDS_NULL ON;
SET QUOTED_IDENTIFIER ON;
SET NUMERIC_ROUNDABORT OFF;
SET ARITHABORT ON;
GO
-- Make the database trustworthy if it isn't already
DECLARE @sql nvarchar(max) = 'ALTER DATABASE ' + quotename(db_name()) + ' SET TRUSTWORTHY ON;';
IF (SELECT [is_trustworthy_on] FROM sys.databases WHERE [name] = db_name()) = 0
    EXECUTE sp_executesql @sql;
GO
-- Create the table and procedures called by the DXC3 engine
DECLARE @sql nvarchar(max) = 'CREATE SCHEMA [dxc3];';
IF NOT EXISTS (SELECT * FROM sys.schemas WHERE name = 'dxc3')
BEGIN
    EXECUTE sp_executesql @sql;
END
GO
IF EXISTS ( SELECT *
            FROM sysobjects
            WHERE id = object_id(N'[dxc3].[usp_StartOperation]')
            and OBJECTPROPERTY(id, N'IsProcedure') = 1 )
BEGIN
    DROP PROCEDURE [dxc3].[usp_StartOperation];
END
GO
IF EXISTS (SELECT * FROM sys.assemblies WHERE name = 'DXC3Database')
BEGIN
    DROP ASSEMBLY [DXC3Database];
END
GO
Copy Execute Close

```

This will install the CLR Assembly and create a dxc3.usp_StartOperation stored procedure in the selected database.

The stored procedure can then be called and passed the name of the server that DXC3 service is running on, the name of the configuration running on the service, the name of the operation to start and an XML element with the

parameters to pass in. The variables list is most easily created using the FOR XML PATH command and selected the values to pass with the column names being the name of the variable. E.g.

```

DECLARE @machine_name nvarchar(255)
DECLARE @configuration_name nvarchar(255)
DECLARE @operation_name nvarchar(max)
DECLARE @variables xml

SET @machine_name = 'localhost'
SET @configuration_name = 'Test'
SET @operation_name = 'Operation'
SET @variables =
    (SELECT
        'Some Value' AS FirstParameterName
        ,3 AS SecondParameterName
        FOR XML PATH)

EXECUTE [dxc3].[usp_StartOperation]
    @machine_name
    ,@configuration_name
    ,@operation_name
    ,@variables

```

C. Web Service

Operations can be triggered using a web service call to the DXC3 engine. The engine by default starts a web service on port 5130 and has a trigger method that can be called using a simple web browser or any method of executing a simple web request.

```

var url = string.Format("http://{0}:{1}/trigger", serverName, dxc3Port);
var web = WebRequest.Create(url);
var body =
    string.Format("config={0}\r\noperation={1}\r\n{2}",
        configName,
        operationName,
        string.Join("\r\n",
            values.Select(x =>
                string.Format("{0}={1}", x.Key, SerializeObject(x.Value)))));

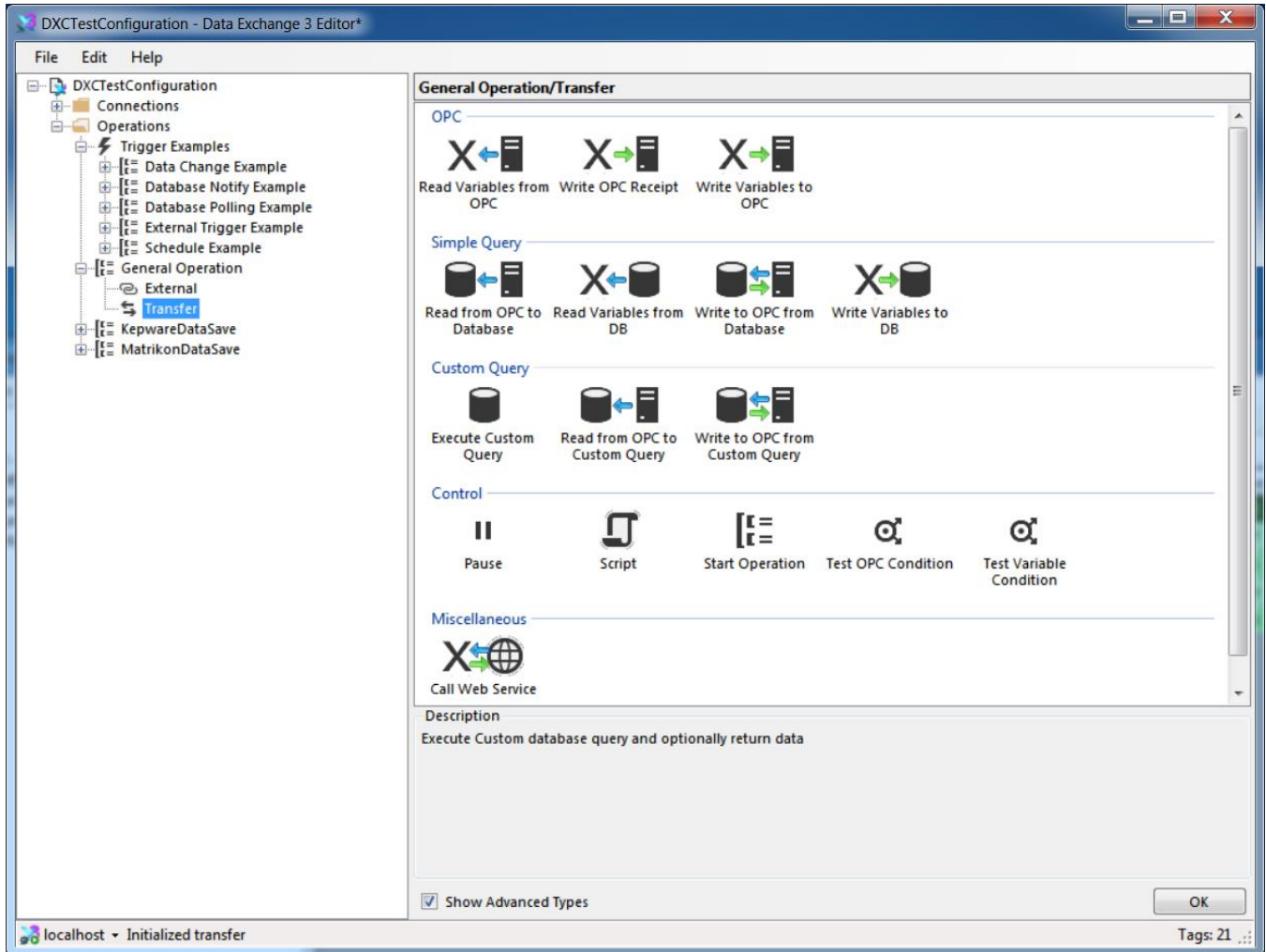
var content = Encoding.UTF8.GetBytes(body);
web.ContentType = "text/json";
web.Method = "POST";
var requestStream = await web.GetRequestStreamAsync();
requestStream.Write(content, 0, content.Length);
requestStream.Flush();
var response = await web.GetResponseAsync();

```

4.13. GENERAL TRANSFER TYPES DESCRIPTIONS

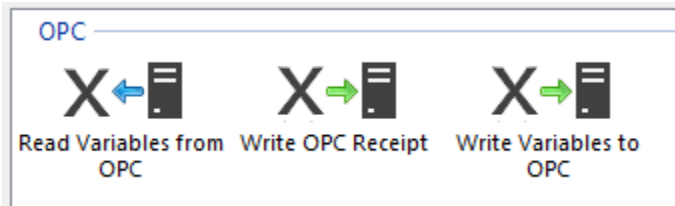
Transfers are the action portion of the Operation.

Data Exchange 3 offers several ways of creating transfers. Users not knowledgeable in database architecture and SQL can build configurations using simple transfer types. Advanced users knowledgeable in database architecture and SQL can build configurations using advanced transfer types.



4.13.1 OPC TRANSFERS

These transfers are read and write to the OPC Server without using a database connection.



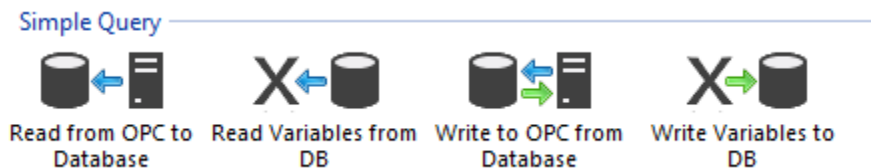
OPC: Read Variables from OPC – allows the user to move data from the OPC Server to variables that are created within an operation. These variables can be used in subsequent transfers in the operation.

OPC: Write OPC Receipt – allows the user to write a constant value to the OPC server. This transfer is used for feedback. It usually is placed at the end of the operation and it tells the industrial device that all transfers in this operation were successfully executed.

OPC: Write Variables to OPC – allows the user to move data stored in operation variables to the OPC Server.

4.13.2 SIMPLE QUERY TRANSFERS

Simple query transfers are for users unfamiliar with database queries. The Editor screen for these operations creates simple queries for reading and writing data to a database.



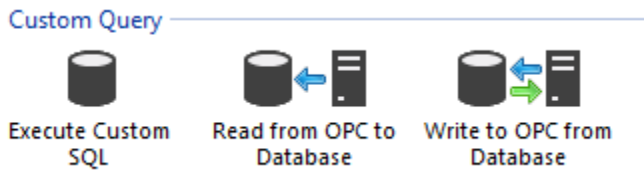
Simple Query: Read from OPC to Database – allows the user to move data from the OPC server to the database. This transfer is used to collect the information from the factory floor and is helpful for product traceability, data acquisition, process history, etc.

Simple Query: Read Variables from DB – allows the user to move data from the database to variables that are created within an operation. This transfer is used to store data that will be used in a transfer executed later in the operation.

Simple Query: Write to OPC from Database – allows the user to move data from the database to the OPC server. This transfer is used to download data to an industrial device. Example uses of this transfer are recipe downloads, changeover information, and product information downloads.

Simple Query: Write Variables to DB – allows the user to move data stored in operation variables to the database.

4.13.3 CUSTOM QUERY TRANSFERS

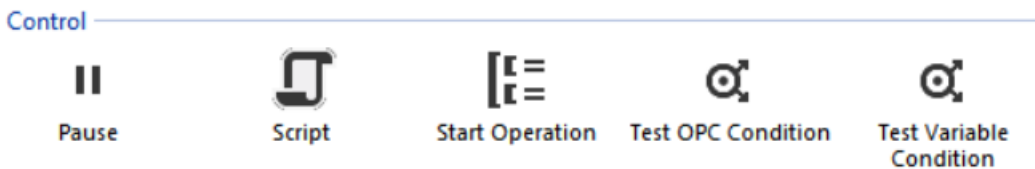


Custom Query: Execute Custom SQL – allows an advanced user to write a custom SQL statement. The SQL statement can be a complicated query or a stored procedure that does not use OPC server data. Variables that contain OPC data can be used. Data Exchange 3 assists in writing this statement by supplying a list of database objects (tables, views, stored procedures). This transfer type is used when OPC server data access is not required.

Custom Query: Read from OPC to Database – allows an advanced user to write a custom SQL statement. The SQL statement can be a complicated query or a stored procedure that uses OPC server data for parameters. Data Exchange 3 assists in writing this statement by supplying a list of database objects (tables, views, stored procedures). This transfer type is used in the same situations as the Simple Read from OPC to Database, but it is much more flexible.

Custom Query: Write to OPC from Database – allows an advanced user to write a custom SQL statement. The SQL statement can be a complicated query or a stored procedure that uses OPC server data for parameters. The results of the SQL statement can be mapped to items in the OPC server. Data Exchange 3 assists in writing this statement by supplying a list of the database objects (tables, views, stored procedures). This transfer type is used in the same situations as the Simple Write to OPC from Database transfer, but it is much more flexible.

4.13.4 CONTROL TRANSFERS



Control: Pause - This transfer allows you to pause the operation execution. It is useful when the operation has some interaction between the device (PLC) and Data Exchange 3 computer. Good example of this transfer usage would be a sequence of three transfers in one operation.

1. Setting a value in the PLC by using Write OPC Receipt transfer.
2. Operation Pause transfer to allow PLC to prepare the data.
3. Reading the data from the PLC.

Control: Script – This transfer allows custom C#, JavaScript, and Visual Basic scripts to be written.

Control: Start Operation – This transfer allows you to call an operation in the middle of another operation.

Control: Test OPC Condition – The transfer allows you to check the status of an OPC item to determine whether to proceed with the rest of the operation. If the condition is true, the operation proceeds to the next transfer, if it is false, it is redirected to the selected transfer. This transfer is emulating “If else” logic. This can be useful, for example, to only log data from a machine when it is running or when different data needs to be collected depending on the bit status.

Control: Test Variable Condition - The transfer allows you to check the status of an DXC variable to determine whether to proceed with the rest of the operation. If the condition is true, the operation proceeds to the next transfer, if it is false, it is redirected to the selected transfer. This transfer is emulating “If else” logic. This can be useful, for example, to only log data from a machine when it is running or when different data needs to be collected depending on the bit status.

4.13.5 MISCELLANEOUS TRANSFERS

Miscellaneous



Miscellaneous: Call Web Service – allows calling a SOAP, JSON, or Socket web service depending upon which of these options were selected in the Web Service Connection.

4.14. GENERAL TRANSFER SETTINGS

Many of the Transfers are similar and therefore have the same configuration settings.

4.14.1 NAME

Name of the transfer.

4.14.2 ON ERROR

If an error occurs while processing a transfer, the entire operation can be terminated or continue.

Operation Failure: The operation fails and is terminated after the failure of this transfer.

Operation Success: The operation fails and is terminated however, it returns success. The remainder of the operation will not execute. An error will be generated in the event log for the transfer that caused the error.

Next Transfer: Executes the next transfer in the list.

[Specific Transfers]: Operation can continue with a specific transfer.

4.14.3 OPC CONNECTION

Name of the OPC Server to use in this operation. Only one OPC Server can be used in a transfer.

4.14.4 READ FROM CACHE

Data Exchange can read the cache of the OPC Server. The OPC Server will reply with the value it already has in its memory, rather than read it from the source device. Reading Cache is faster than reading from the device, and requires more OPC communication traffic.

4.14.5 READ FROM DEVICE

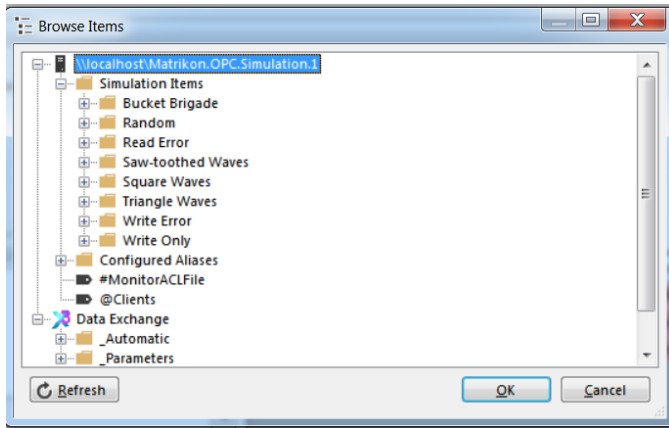
If checked, reading from the device guarantees a fresh value for the OPC tag. Device reads are slower than Cache reads, but is more accurate data, and less OPC communication traffic.

4.14.6 USE NULL FOR BAD QUALITY

If a bad quality code is read from the OPC Server return NULL instead of potentially old or bad data.

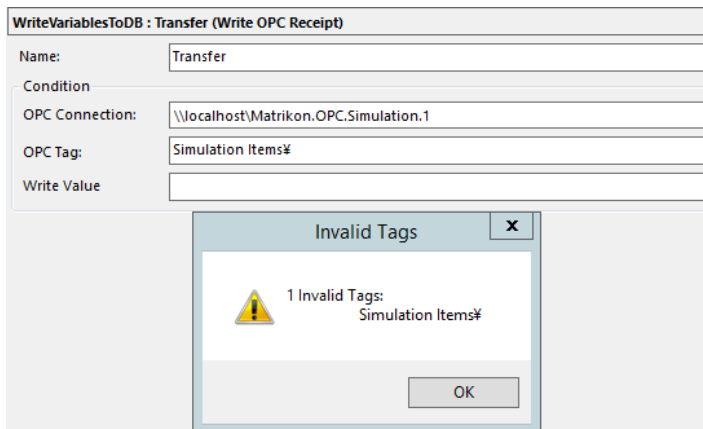
4.14.7 BROWSE

Browses the tags in the OPC Server and allows you to select one to use in this operation. An error will occur if Data Exchange cannot connect to the OPC Server.



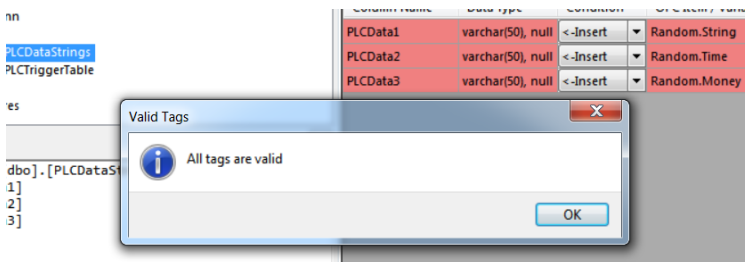
Some OPC Servers allow you to select a folder name while others do not. Data Exchange does not restrict what is selected so take caution in not selecting something invalid.

In this example, I selected “Simulation Items” folder in the Matrikon OPC Server. Selecting “Validate Tag” will return an “Invalid Tag” message.



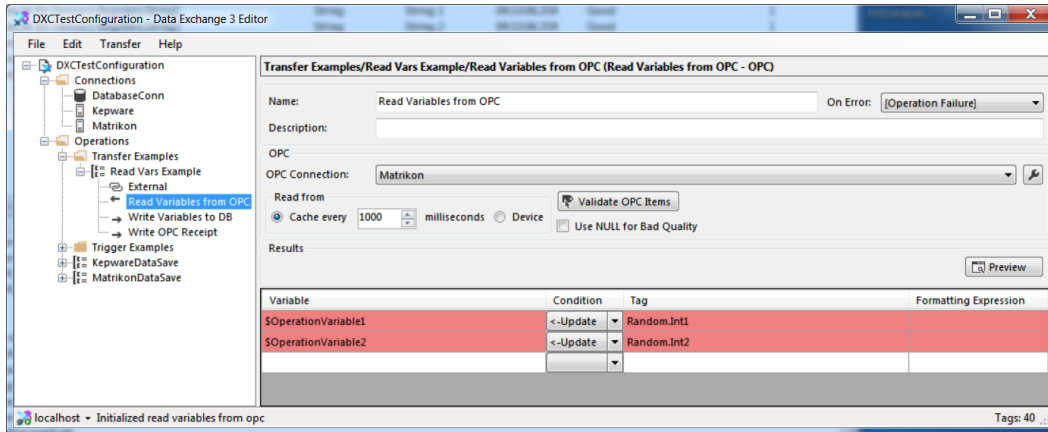
4.14.8 VALIDATE TAG

Verifies that the OPC Tag is good and Data Exchange can read the value.



4.15. TRANSFERS

Transfers have several common settings in which to control their execution. Some transfers may not have all of these options.



4.15.1 NAME

Name of the transfer.

4.15.2 ON ERROR

If an error occurs while processing a transfer, the entire operation can be terminated or continue.

Operation Failure: The operation fails and is terminated after the failure of this transfer.

Operation Success: The operation fails and is terminated however, it returns success. The remainder of the operation will not execute. An error will be generated in the event log for the transfer that caused the error.

Next Transfer: Executes the next transfer in the list.

[Specific Transfers]: Operation can continue with a specific transfer.

4.15.3 OPC CONNECTION

Name of the OPC Server to use in this operation. Only one OPC Server can be used in a transfer.

A. Read from Cache

Data Exchange can read the cache of the OPC Server. The OPC Server will reply with the value it already has in its memory, rather than read it from the source device. Reading Cache is faster than reading from the device, and requires more OPC communication traffic.

B. Read from Device

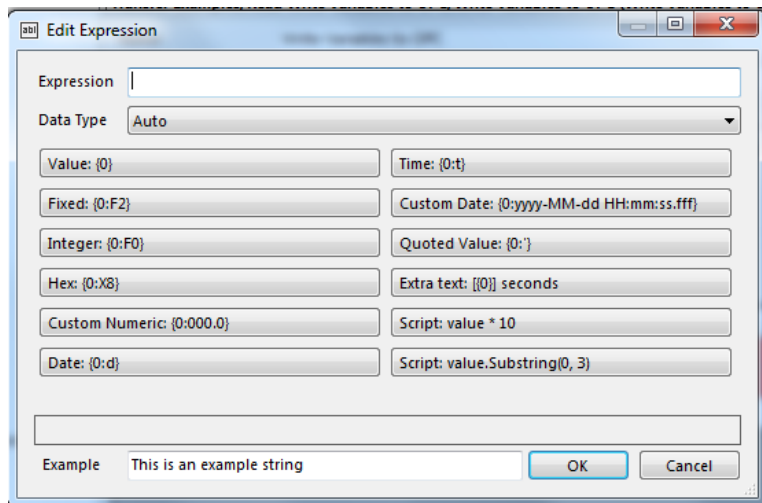
If checked, reading from the device guarantees a fresh value for the OPC tag. Device reads are slower than Cache reads, but is more accurate data, and less OPC communication traffic.

C. Use NULL for Bad Quality

If a bad quality code is read from the OPC Server return NULL instead of potentially old or bad data.

4.15.4 FORMAT

Allows you to format the data sent to the OPC Server. Manually edit the Expression or select from several predefined formats by clicking the 12 format buttons. The enabled predefined formats are determined by the Data Type selected.



4.15.5 PREVIEW

To preview the variables in a write or to test a transfer, select Preview. Select Execute to run the transfer. The Editor must be connected to the Engine to enable the Preview function.

Preview Write Variables to OPC

Parameters	
Item	Value
_Configuration	WizardConfiguration
_ConfigurationStartTime	11/10/2017 3:48:01 PM
_ConfigurationModified	11/10/2017 3:48:01 PM
_MachineName	VIRT1-ELA
_EngineVersion	3.0.1109.1107
_Operation	Transfer Examples/Read Write Variables to OPC
_OperationInstance	1
_OperationStartTime	11/10/2017 3:48:01 PM
_Transfer	Transfer Examples/Read Write Variables to OPC.Write Variables to OPC

Returned Data	
Item	Value
_Configuration	WizardConfiguration
_ConfigurationStartTime	11/10/2017 3:48:01 PM
_ConfigurationModified	11/10/2017 3:48:01 PM
_MachineName	VIRT1-ELA
_EngineVersion	3.0.1109.1107
_Operation	Transfer Examples/Read Write Variables to OPC
_OperationInstance	1
_OperationStartTime	11/10/2017 3:48:01 PM
_Transfer	Transfer Examples/Read Write Variables to OPC.Write Variables to OPC

Next Transfer:
(Next Transfer)

Execute Close

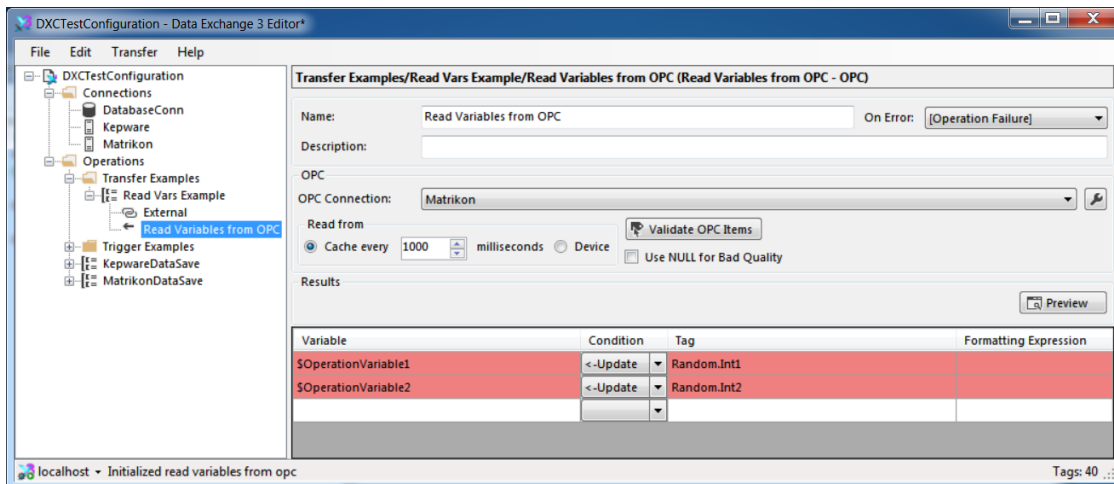
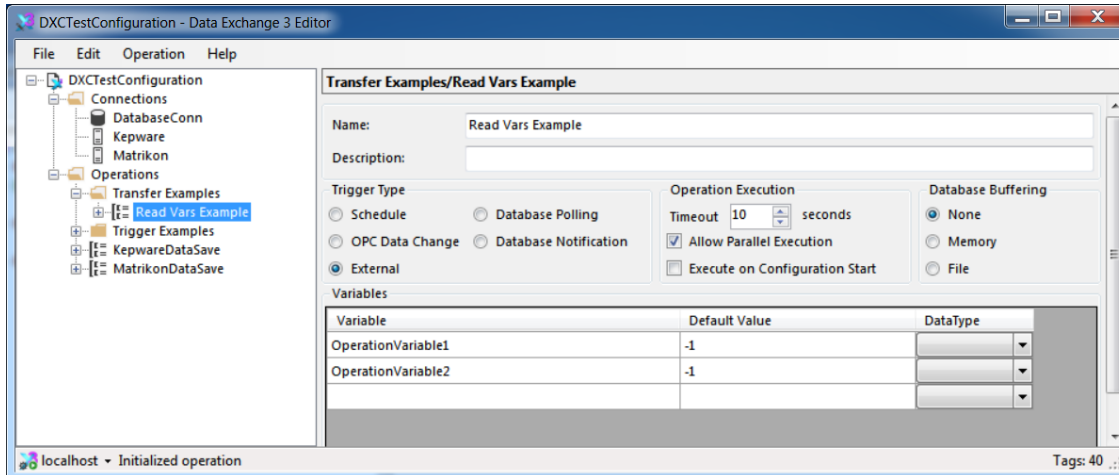
4.16. OPC TRANSFERS

The OPC Transfers do not require a connection to a database. They read and write data between an OPC Server and variables defined within the Data Exchange configuration.

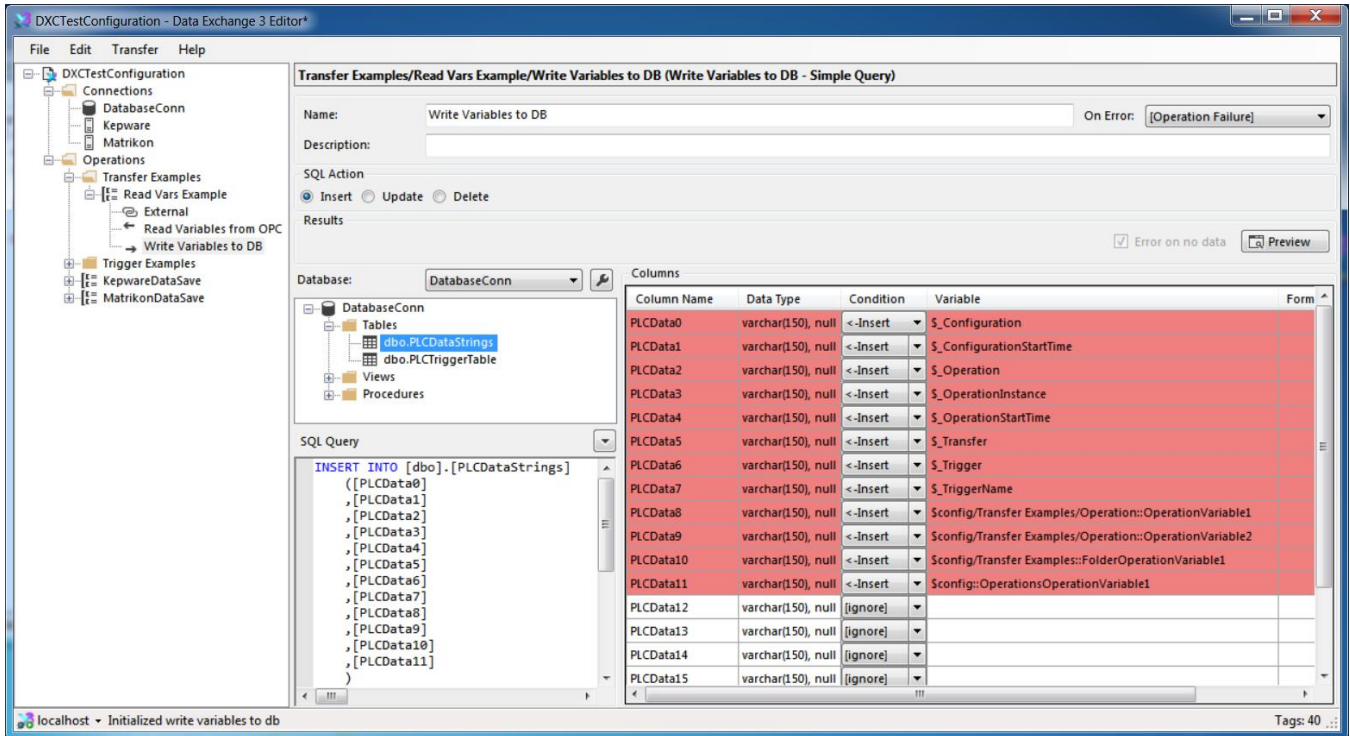
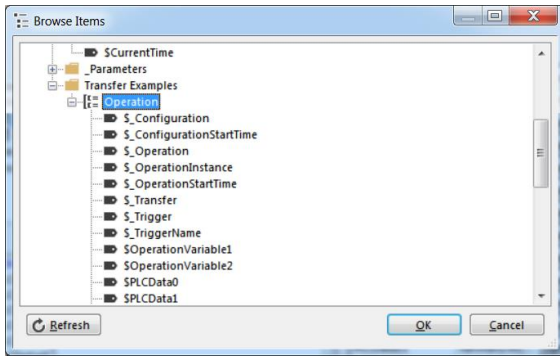
4.16.1 READ VARIABLES FROM OPC

Reads OPC tags and stores them into Data Exchange variables.

Define variables either at the Operation Level, Folder Level or Operations Level.



Once the OPC values are stored in tags they can be used in other places including in other Operations. In this example, the variables are used in the next transfer (See the Simple Query Transfers section for more details on Write Variables to DB):



Next a variable was added to the Folder level and the Operations level and they were used in this Transfer. Note the path to the variables:

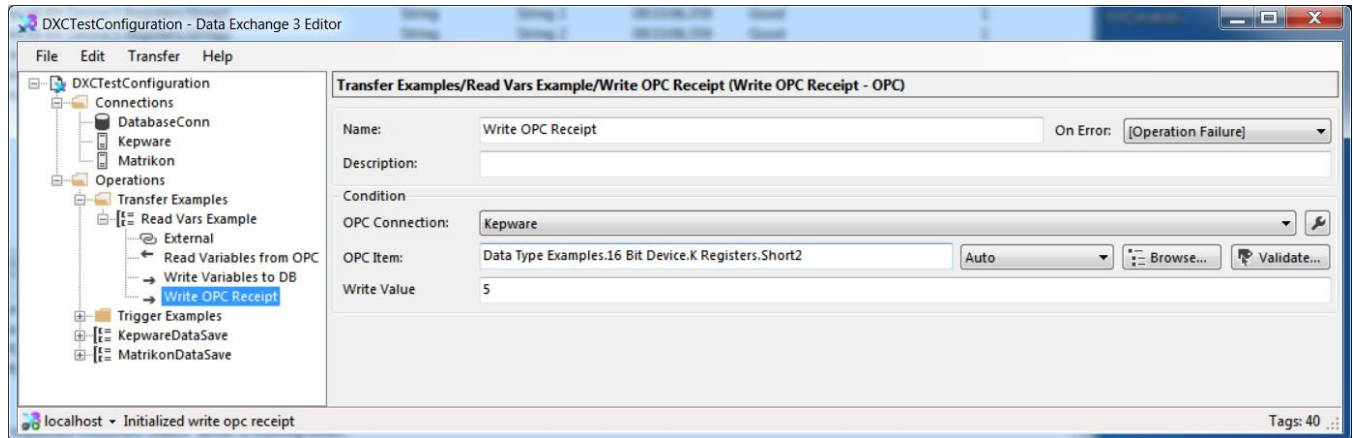
PLCData8	varchar(150), null	<-Insert	\$config/Transfer Examples/Operation::OperationVariable1	
PLCData9	varchar(150), null	<-Insert	\$config/Transfer Examples/Operation::OperationVariable2	
PLCData10	varchar(150), null	<-Insert	\$config/Transfer Examples::FolderOperationVariable1	
PLCData11	varchar(150), null	<-Insert	\$config::OperationsOperationVariable1	
PLCData12	varchar(150), null	[ignore]		

(See the General Transfer Settings section for details on the common transfer parameters.)

4.16.2 WRITE OPC RECEIPT

Writes a fixed value to an OPC Tag. If the value specified is inconsistent with the data type for the OPC Tag, an error condition will occur.

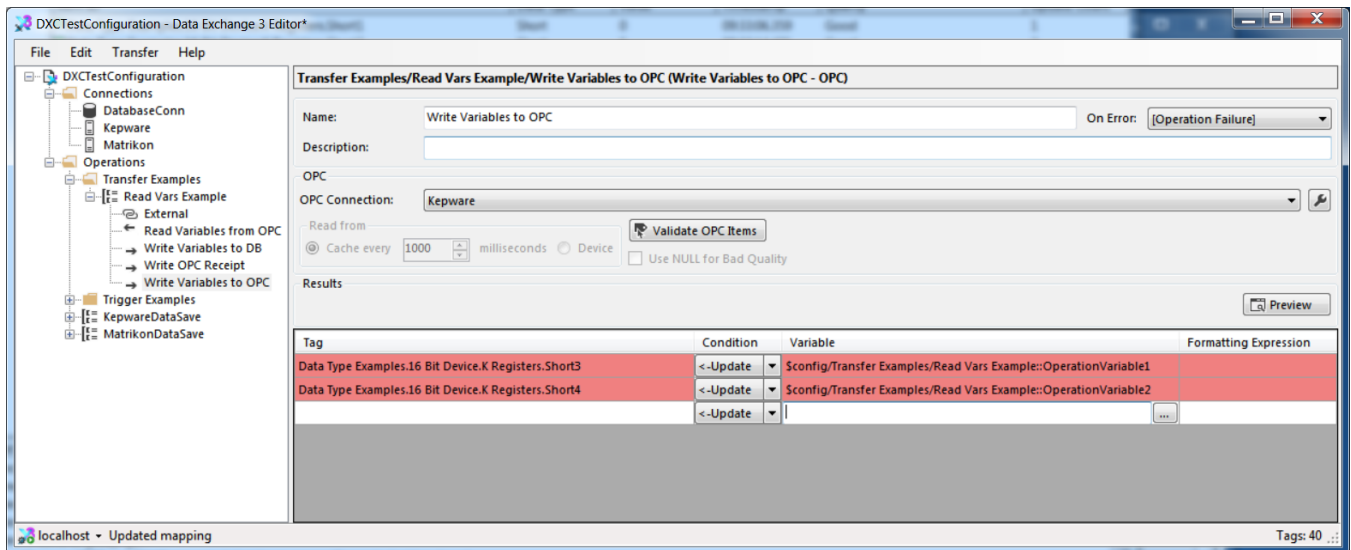
Continuing from the previous example, this Write OPC Receipt transfer writes a response to an OPC Server different from the one used in the preview transfers.



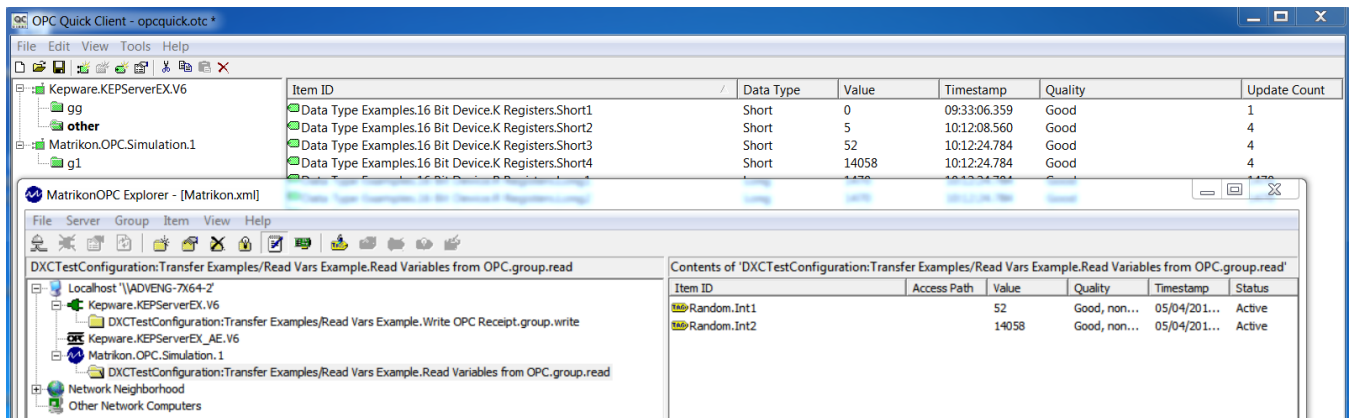
(See the General Transfer Settings section for details on the common transfer parameters.)

4.16.3 WRITE VARIABLES TO OPC

Writes the value of a variable to an OPC tag. If the value of the variable is inconsistent with the data type for the OPC Tag, an error condition will occur.



Continuing on in the example, the two values read from the Matrikon server are stored into variables in the Operation and then written to the Kepware Simulation server. Bringing up the tags in OPC clients we can see the values in each OPC Server match after executing the Operation.



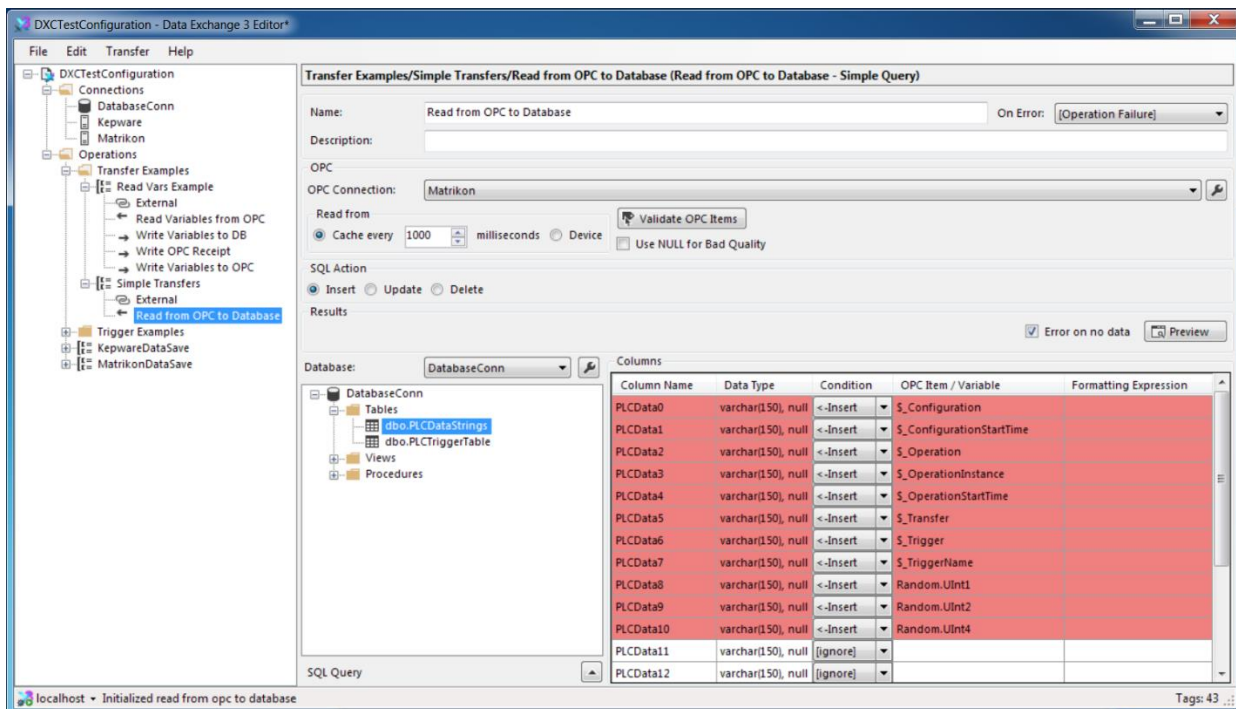
(See the General Transfer Settings section for details on the common transfer parameters.)

4.17. SIMPLE QUERY TRANSFERS

The Simple Query Transfers are transfers that do not require the user to program the SQL Server code specifically. Queries by selecting the table for the query and the columns.

4.17.1 READ FROM OPC TO DATABASE

Uses OPC Tag and Variable values to modify a database table. Select the table in which to use in the transfer. You may select other tables without losing the information you have entered in the previous table. If tables contain the same column names, the tag information entered in a previous table will be saved to any table containing the same column name.



See the General Transfer Settings section for details on the common transfer parameters.

A. SQL Action

Insert

Inserts the values of OPC Tags and Variables into the database.

Update

Updates the records in a database with the values of the specified OPC Tags and Variables. Where conditions are not available in Simple OPC Reads. If this transfer should only affect rows that meet specific criteria, use the Advanced OPC Read Transfer type. To convert this transfer to an advanced transfer right-mouse click on the transfer name and select "Convert to Advanced"

Delete

Deletes the records in a database. Where conditions are not available in Simple OPC Reads. If this transfer should only affect rows that meet specific criteria, use the Advanced OPC Read Transfer type. To convert this transfer to an advanced transfer right-mouse click on the transfer name and select “Convert to Advanced”

B. SQL Query

If you cannot see the actual SQL code, select the up arrow to see the query.



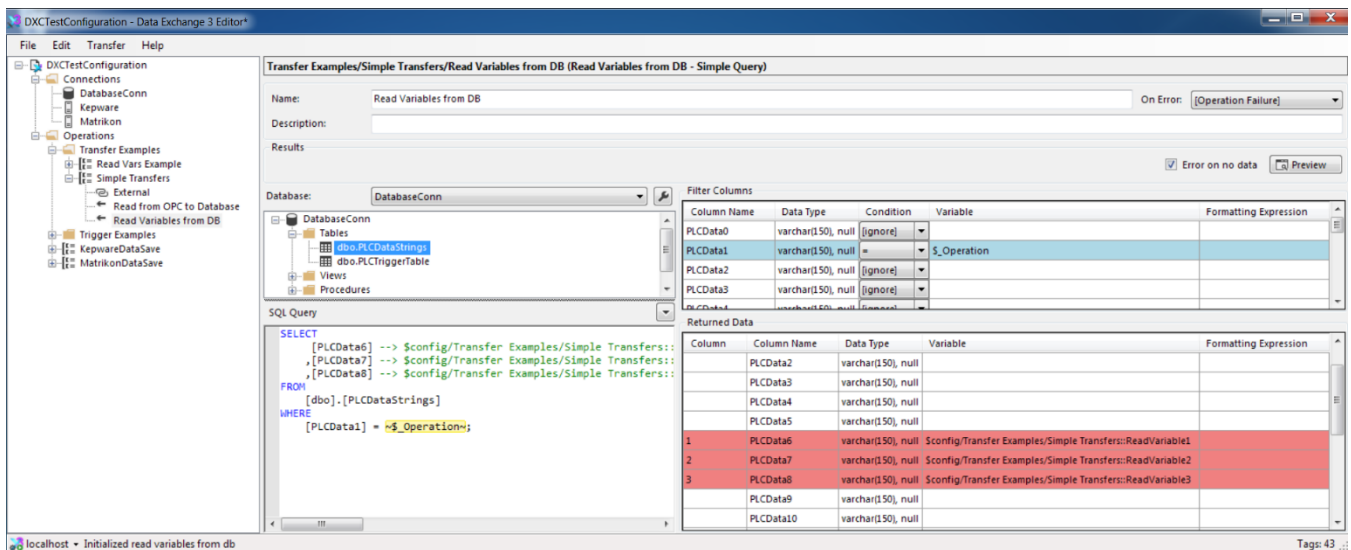
4.17.2 READ VARIABLES FROM DB

Reads data from the database and stores it in Data Exchange Variables. If the data returned to the variables must meet specific criteria, specific data is returned by using the Filter Columns.

If a query returns more than one row, only the first row is stored into the variables. Any other rows returned are not used.

Several variables were added to this Operation:

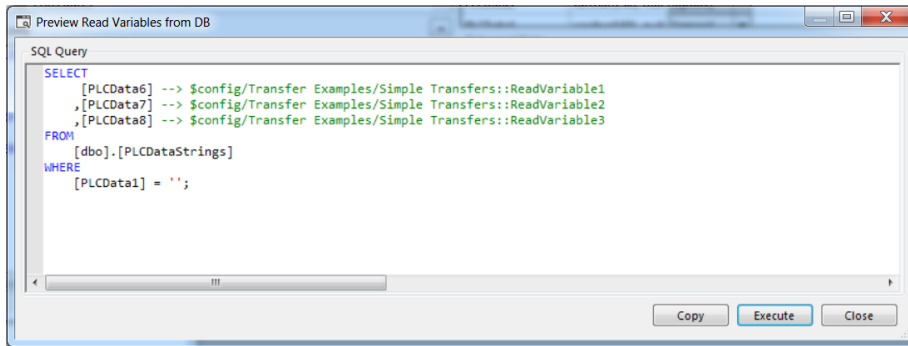
Variable	Default Value	DataType
ReadVariable1	-1	
ReadVariable2	-1	
ReadVariable3	-1	



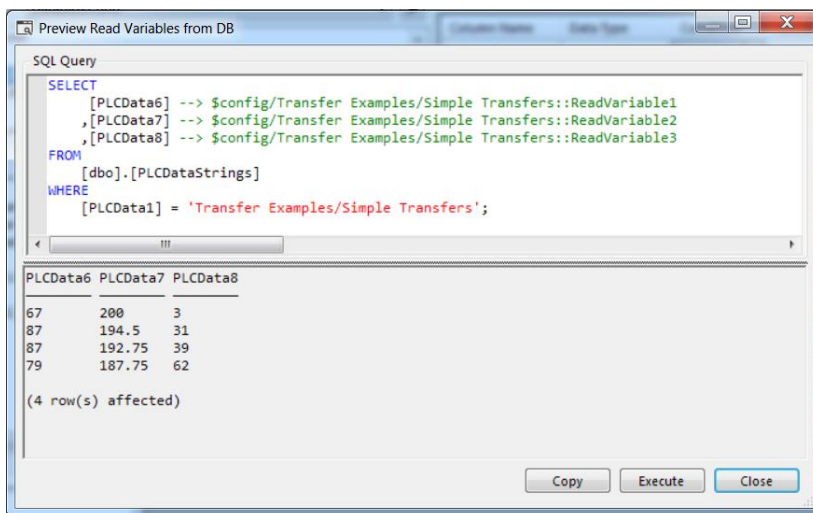
See the General Transfer Settings section for details on the common transfer parameters.

In this example, 3 variables are created. ReadVariable1, ReadVariable2, ReadVariable3. The Operation Name variable is the variable that will be used to filter the data returned to DXC. Table “PLCDataStrings” contains data that will be returned to the operation if the WHERE condition is true. Select Preview to display the variables used in this operation:

Make sure there is data in the table and confirm there is at least one row in the table where PLCData1 equals the Operation Name. Type in the value for the where clause:




Selecting Execute runs the query and returns the row of data that includes “Transfer Examples/Simple Transfers” in the PLCData1 column:



If two rows match then both rows will be returned. However, Data Exchange will only use the first returned row.

A. Error on no data

There are multiple ways to continue processing of the rest of the operation. Uncheck “Error on no data” if this condition is not an error and if processing of the operation should continue. This will cause the operation to complete successfully. If you would like to capture the “no data” event, check “Error on no data” but change the On Error condition to “Next Transfer” so that the operation will continue.

	10/3/2016 7:25 PM	ReadSQLtoVariablesToSQL.Transfer	1:Failed processing transfer [ReadSQLtoVariablesToSQL.Transfer]
---	-------------------	----------------------------------	---

B. Filter Columns

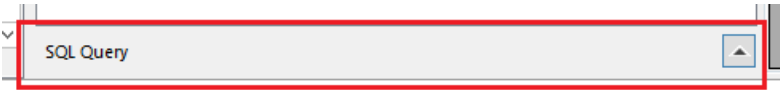
Use Filter Columns to return a specific data set from the database. Only Data Exchange variables are used to filter returned data. If it is necessary to have the value come from an OPC Server or another database, store the value into a variable before calling this transfer.

Filter Columns				
Column Name	Data Type	Condition	Variable	Formatting Expression
PLCData0	varchar(150), null	[ignore]		
PLCData1	varchar(150), null	=	\$_Operation	
PLCData2	varchar(150), null	[ignore]		
PLCData3	varchar(150), null	[ignore]		
PLCData4	varchar(150), null	[ignore]		

Returned Data

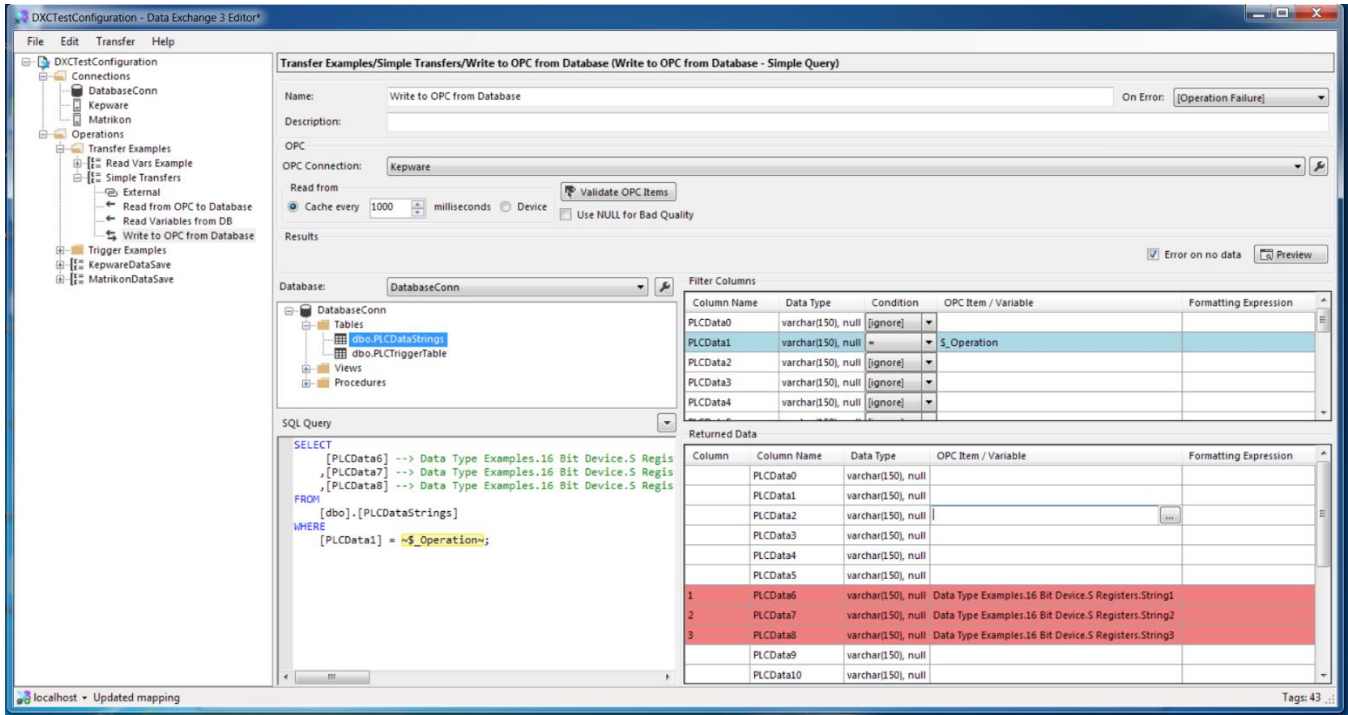
SQL Query

If you cannot see the actual SQL code, select the up arrow to see the query.



4.17.3 WRITE TO OPC FROM DATABASE

Write data from a database to an OPC Server.



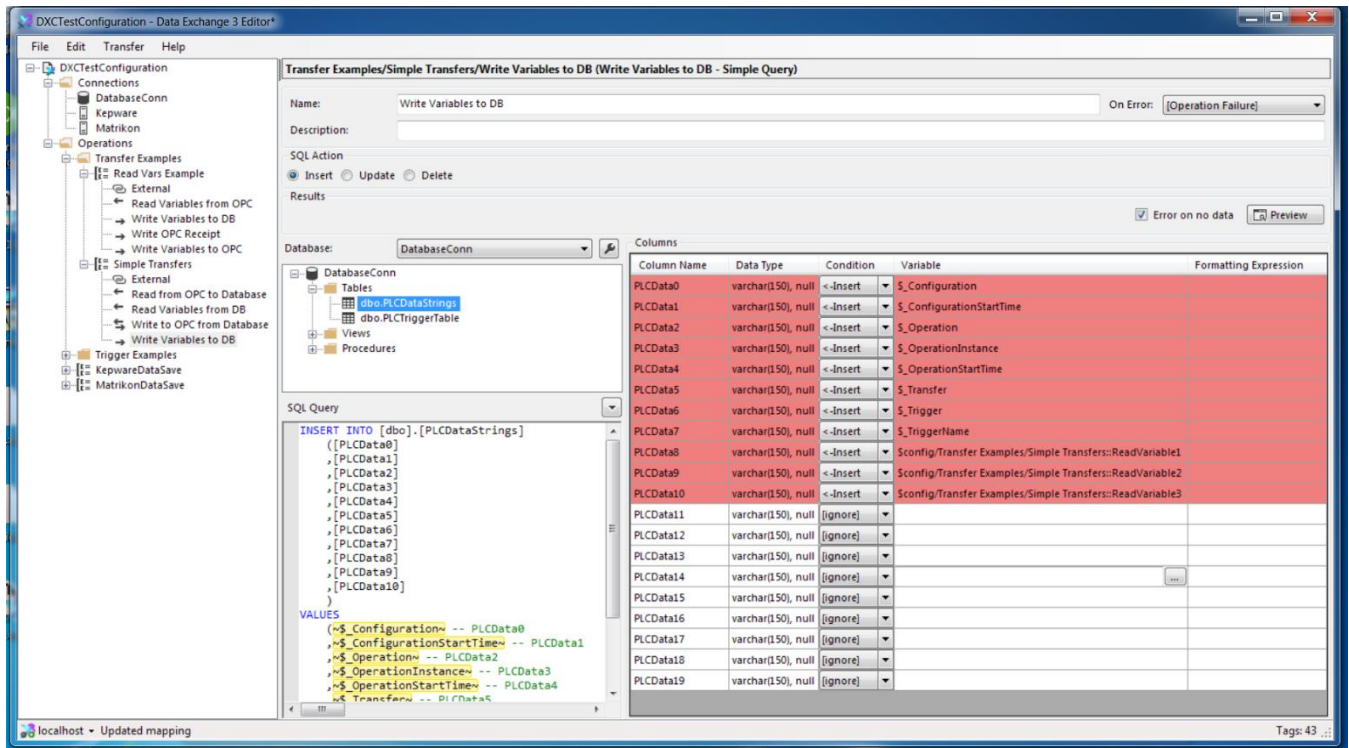
SQL Query

If you cannot see the actual SQL code, select the up arrow to see the query.



4.17.4 WRITE VARIABLES TO DB

Write Data Exchange Variables to a database.



SQL Action

Insert

Inserts the values of Data Exchange Variables into the database.

Update

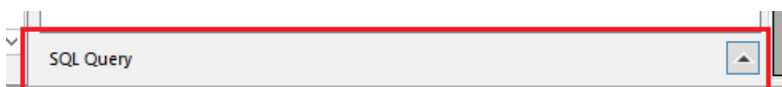
Updates the records in a database with the values of the specified Variables. Where conditions are not available in Variable DB Writes. If this transfer should only affect rows that meet specific criteria, use the Advanced OPC Read Transfer type. To convert this transfer to an advanced transfer right-mouse click on the transfer name and select "Convert to Advanced."

Delete

Deletes the records in a database. Where conditions are not available in Variable DB Writes. If this transfer should only affect rows that meet specific criteria, use the Advanced OPC Read Transfer type. To convert this transfer to an advanced transfer right-mouse click on the transfer name and select "Convert to Advanced."

SQL Query

If you cannot see the actual SQL code, select the up arrow to see the query.



Continuing the example from the previous section, the operation has 3 defined variables:

Variable	Default Value	DataType
ReadVariable1	-1	<input type="text"/>
ReadVariable2	-1	<input type="text"/>
ReadVariable3	-1	<input type="text"/>
		<input type="text"/>

The variables are filled with OPC data in the previous transfer:

	PLCData5	varchar(150), null	
1	PLCData6	varchar(150), null	\$config/Transfer Examples/Simple Transfers::ReadVariable1
2	PLCData7	varchar(150), null	\$config/Transfer Examples/Simple Transfers::ReadVariable2
3	PLCData8	varchar(150), null	\$config/Transfer Examples/Simple Transfers::ReadVariable3
	PLCData9	varchar(150), null	

4.18. CUSTOM QUERY TRANSFERS

4.18.1 EXECUTE CUSTOM QUERY

Create custom queries using Data Exchange Variables. Only variables can be passed in or returned from a Custom query.

For this Transfer we will use a stored procedure to return data to 3 tags in the PLC. The stored procedure will select data based on the value of an integer passed to it. Create a table and a stored procedure to be used in this example. Add data to the database so the query will return a resultset.

```
USE [DXExampleDatabase]
GO

/***** Object: Table [dbo].[IntegerTable]  Script Date: 5/7/2019 2:32:17 PM *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[IntegerTable](
    [Integer1] [smallint] NULL,
    [Integer2] [smallint] NULL,
    [Integer3] [smallint] NULL,
    [Integer4] [smallint] NULL
) ON [PRIMARY]

GO

CREATE PROCEDURE ReturnIntegerData
    @ValueToFind as SMALLINT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT Integer2, Integer3, Integer4
    FROM IntegerTable
    WHERE Integer1 = @ValueToFind
END
GO
```

Four variables are created for the Operation. These are ValueToFindVariable, ReturnValue1, ReturnValue2, ReturnValue3.

Create a new procedure called Execute Stored Procedure Query and add an “Execute Custom Query” transfer.

```
EXEC ReturnIntegerData
    @ValueToFind =
```

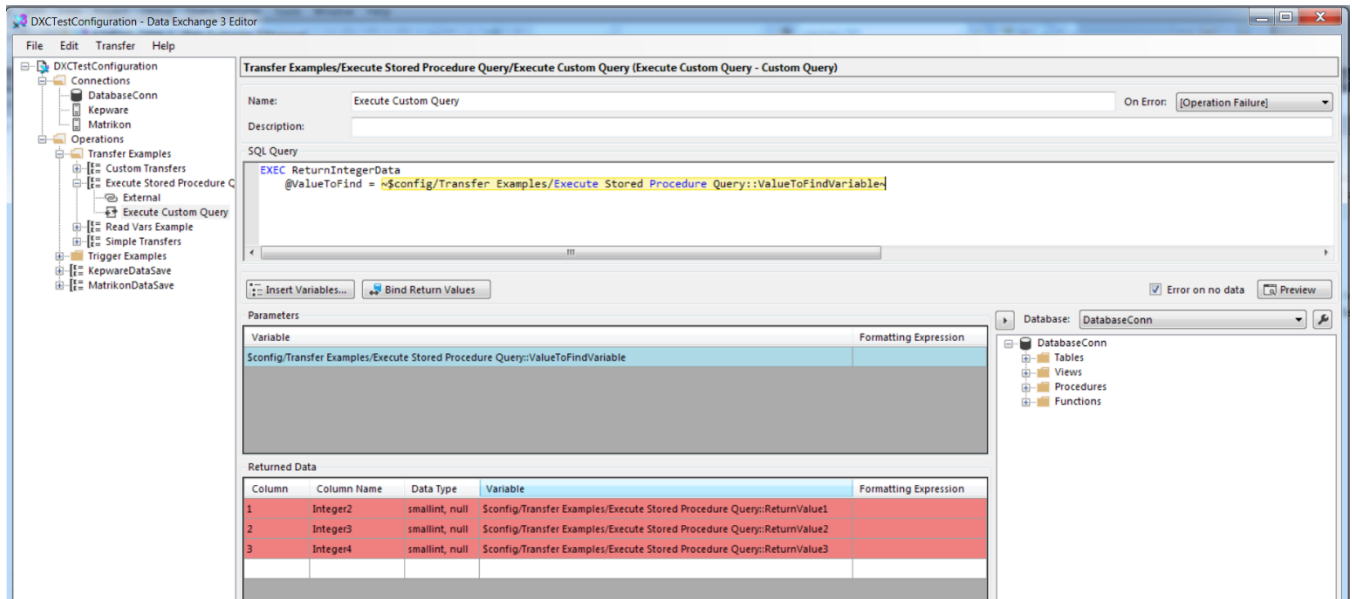
Enter query and add the ValueToFind variable to the call to the stored procedure.

```
EXEC ReturnIntegerData
    @ValueToFind = ~$config/Transfer Examples/Execute Stored Procedure Query::ValueToFindVariable~
```

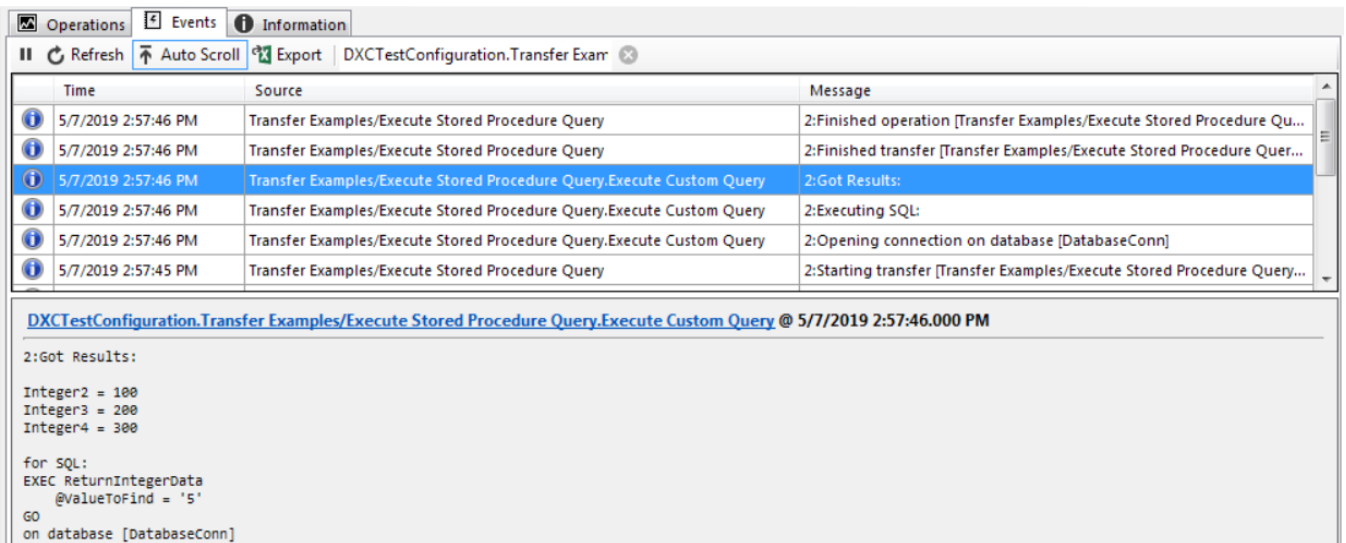
Parameters	
Variable	Formatting Expression
\$config/Transfer Examples/Execute Stored Procedure Query::ValueToFindVariable	

DXC will determine the return data information which will appear in the return data section. Insert the variables that will contain the results to the query.

Column	Column Name	Data Type	Variable	Formatting Expression
1	Integer2	smallint, null	\$config/Transfer Examples/Execute Stored Procedure Query::ReturnValue1	
2	Integer3	smallint, null	\$config/Transfer Examples/Execute Stored Procedure Query::ReturnValue2	
3	Integer4	smallint, null	\$config/Transfer Examples/Execute Stored Procedure Query::ReturnValue3	



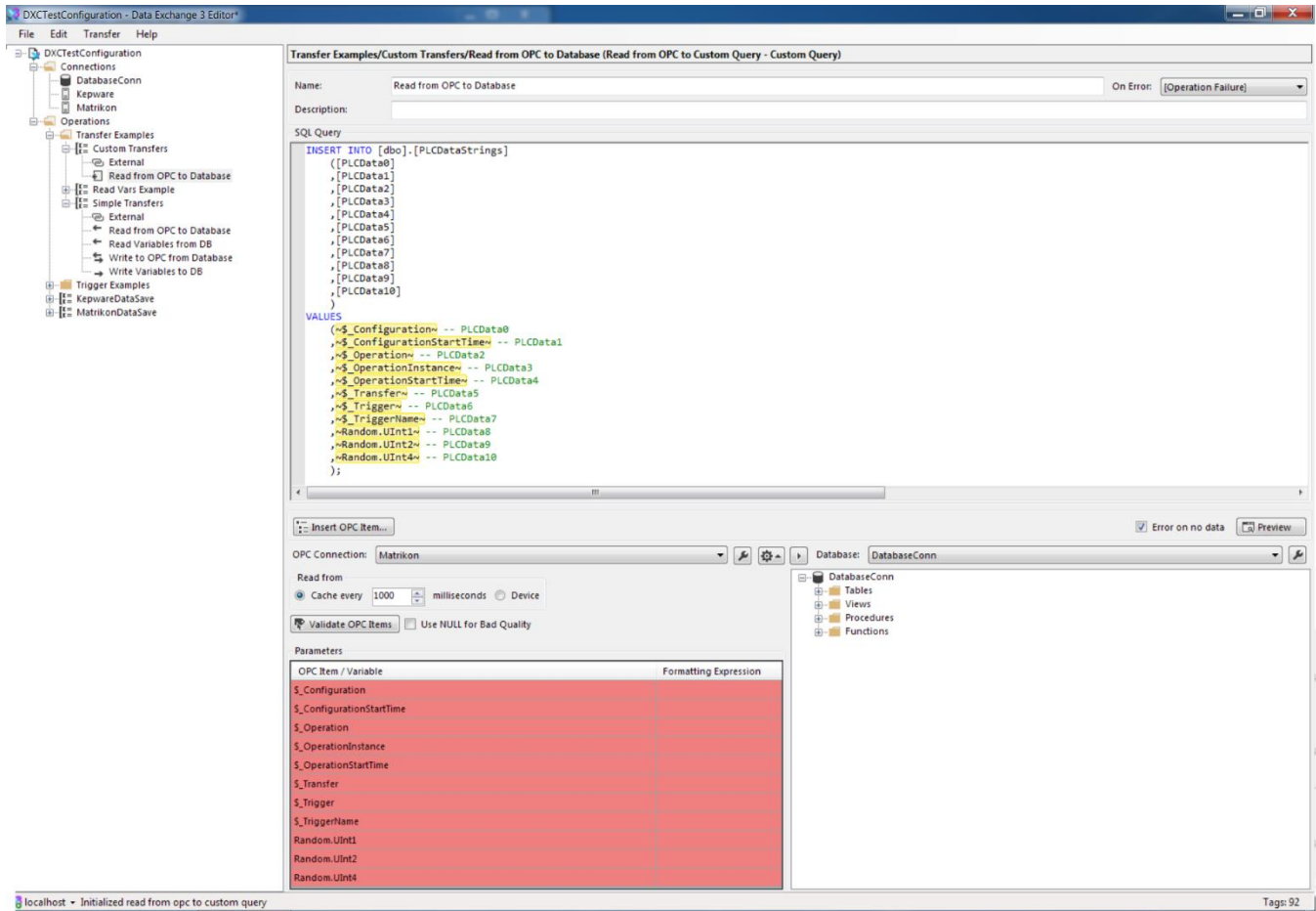
Turning on Verbose messages and running the query will produce the following result. If these variables need to be used somewhere else then at this point they will contain valid data.



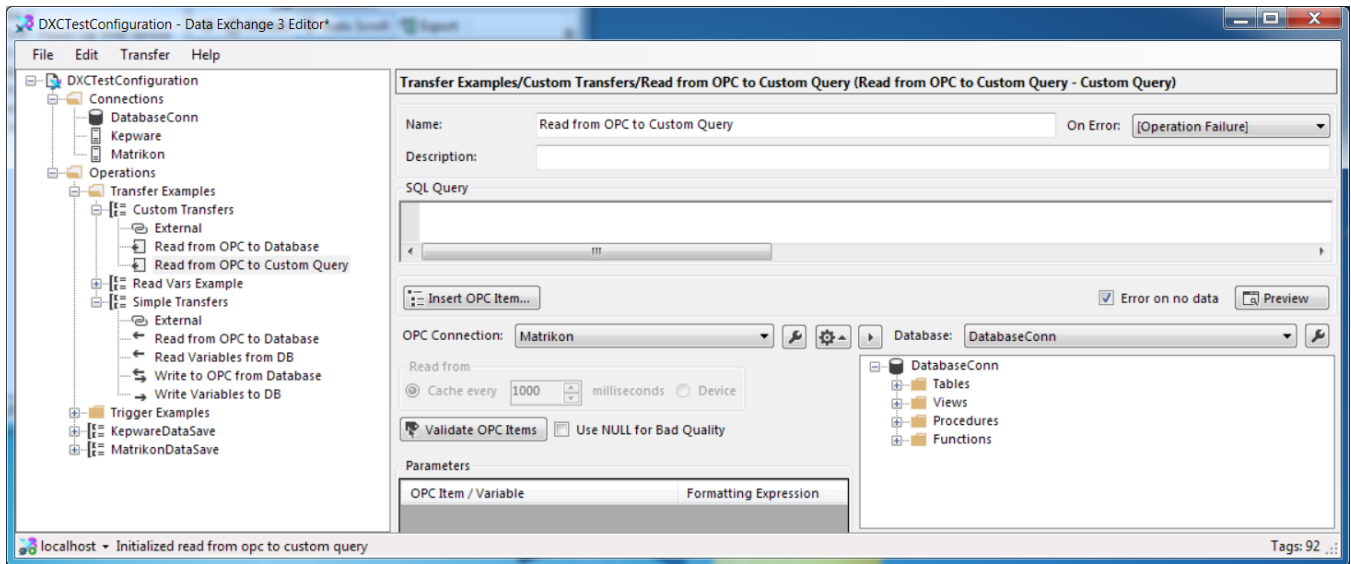
4.18.2 CUSTOM READ FROM OPC TO DATABASE

Read data from an OPC Server and store in a database table.

The custom query builder screen has several different components used in building a custom query. Any standard query can be turned into a custom by right-mouse clicking on the Operation and selecting 'Convert to Custom Query'. DXC breaks out the query from the parameters allowing additional code to be added to the query.



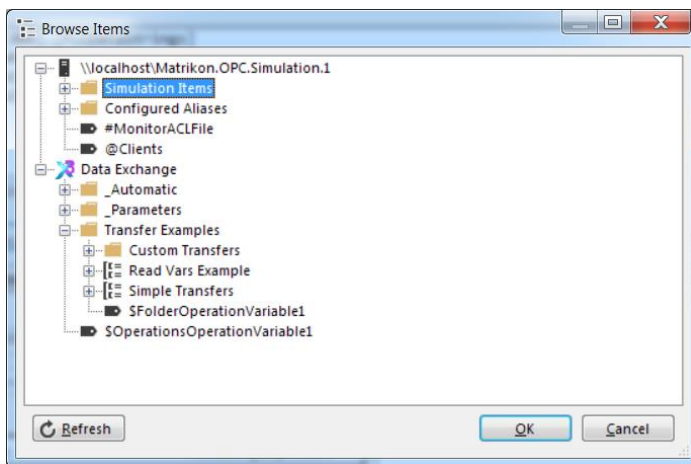
To create a custom Read query Select “Read from OPC to Custom Query” transfer type.



Begin typing the query into the SQL Query component in the window.

```
INSERT INTO [dbo].[PLCDataStrings]
([PLCData0]
,[PLCData1]
,[PLCData2])
VALUES ();
```

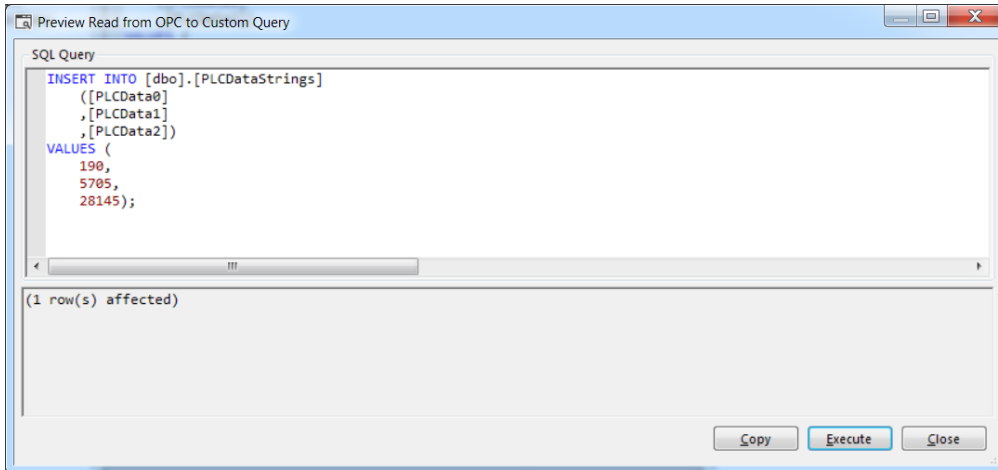
In this example, 3 columns in the database will be used. For each, a variable or OPC tag must be defined. Select the “Insert OPC Item” button to display the OPC tags for the selected OPC Connection and the DXC Variables. Select a tag and click ok to include that tag in the query. Do this for all columns in the query.



The query will look like the following:

```
INSERT INTO [dbo].[PLCDataStrings]
  ([PLCData0]
  ,[PLCData1]
  ,[PLCData2])
VALUES (
  ~Random.UInt1~,
  ~Random.UInt2~,
  ~Random.UInt4~);
```

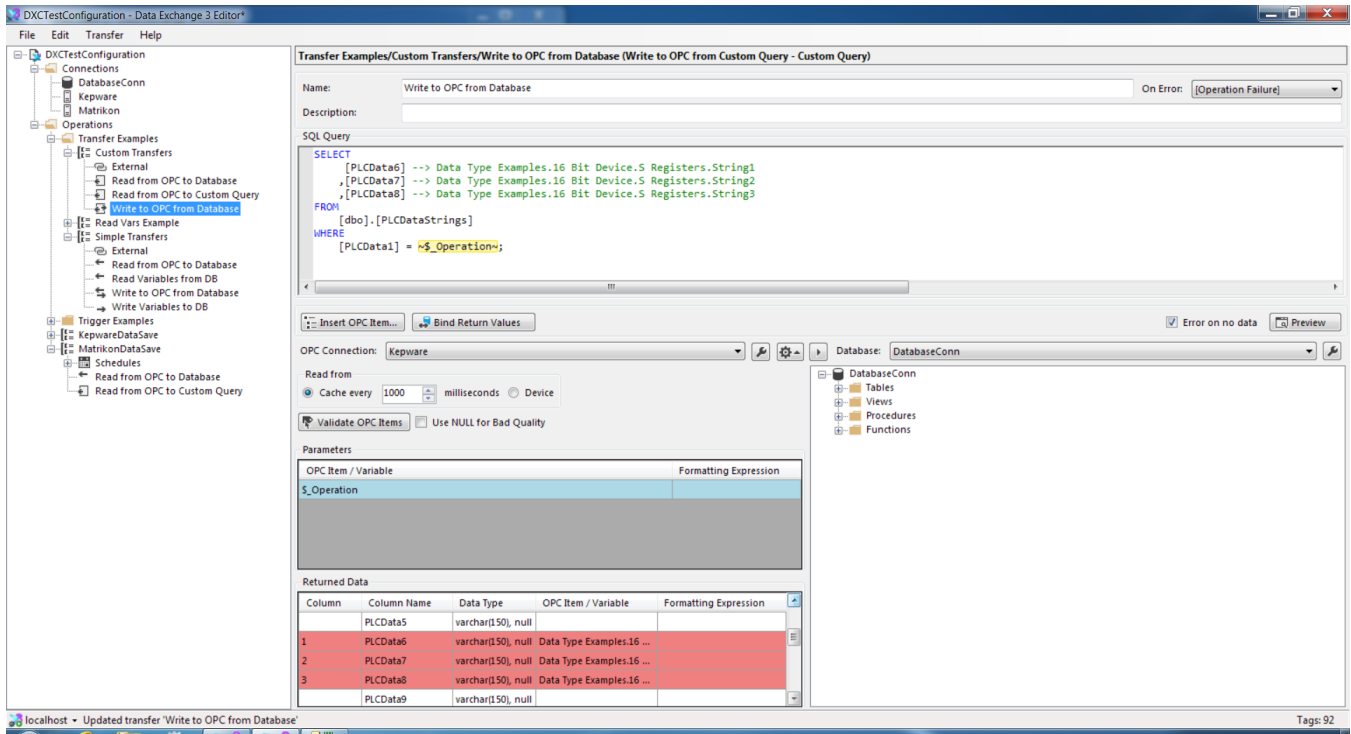
Select "Preview" to test the query.



4.18.3 WRITE TO OPC FROM DATABASE

Write OPC Server data to a database.

The custom query builder screen has several different components used in building a custom query. Any standard query can be turned into a custom by right-mouse clicking on the Operation and selecting 'Convert to Custom Query'. DXC breaks out the query from the parameters allowing additional code to be added to the query.

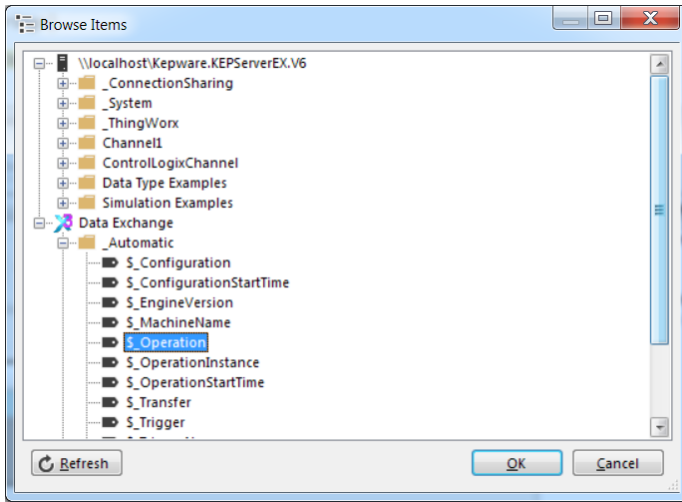


To create a custom Read query Select "Read from OPC to Custom Query" transfer type.

Begin typing the query into the SQL Query component in the window.

```
SELECT
  [PLCData6]
  , [PLCData7]
  , [PLCData8]
FROM
  [dbo].[PLCDataStrings]
WHERE
  [PLCData1] =
```

In this example, 3 columns in the database will be used. For each, a variable or OPC tag must be defined. Select the "Insert OPC Item" button to display the OPC tags for the selected OPC Connection and the DXC Variables. Select a tag and click ok to include that tag in the query. Do this for all columns in the query.



The variable is inserted into the query.

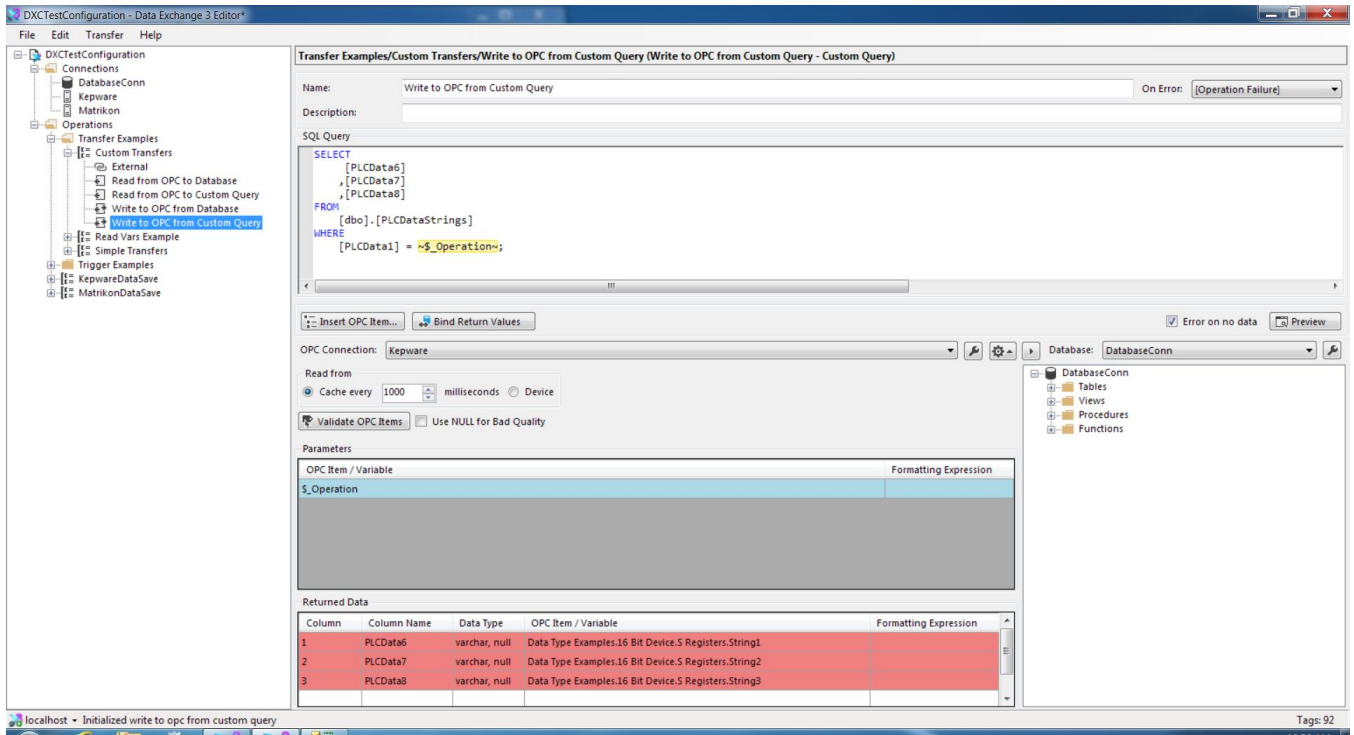
```

SELECT
    [PLCData6]
    , [PLCData7]
    , [PLCData8]
FROM
    [dbo].[PLCDataStrings]
WHERE
    [PLCData1] = ~$Operation~;
    
```

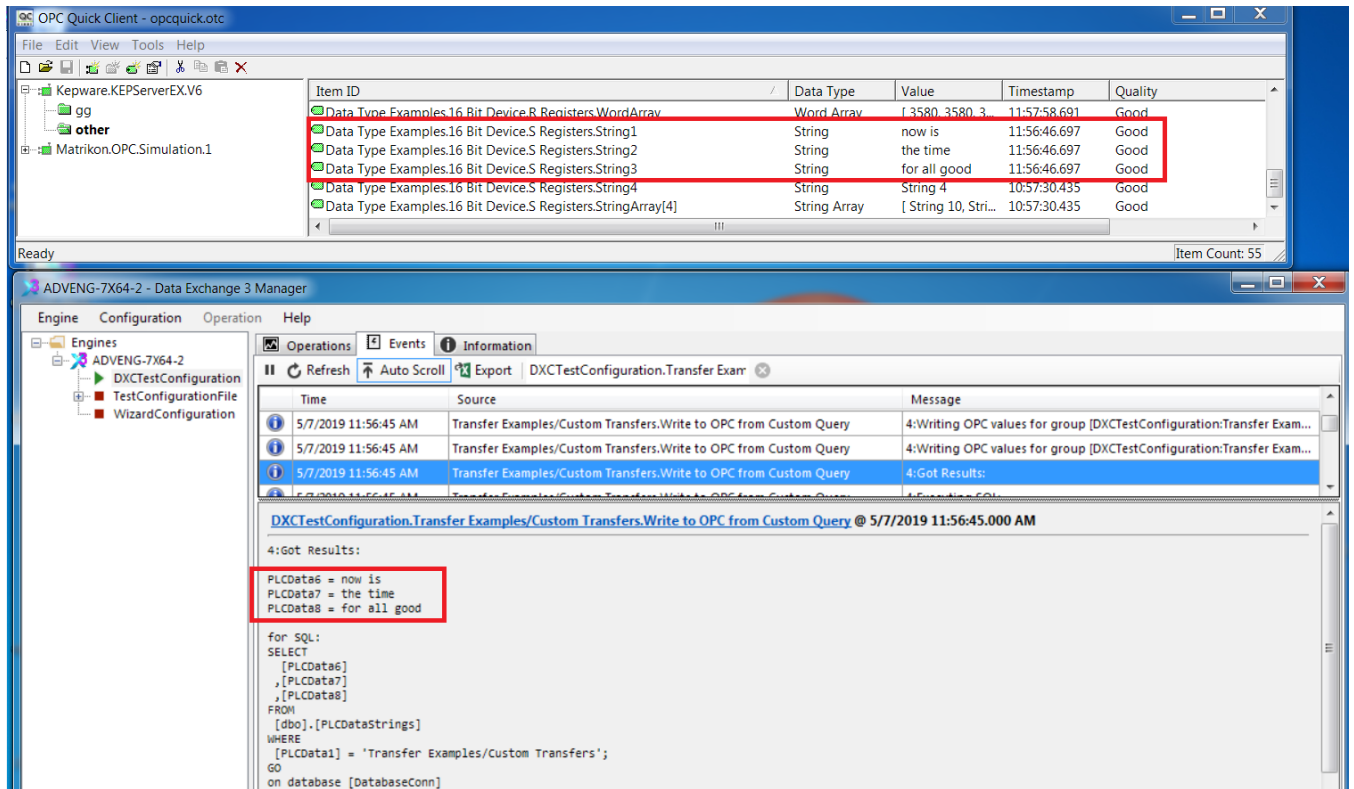
Fill in the OPC tags/variables that will be returned from the query.

Parameters	
OPC Item / Variable	Formatting Expression
\$_Operation	

Returned Data				
Column	Column Name	Data Type	OPC Item / Variable	Formatting Expression
	PLCData6	varchar, null	...	
	PLCData7	varchar, null		
	PLCData8	varchar, null		



Executing the Operation results in modifications to the 3 strings in the Kepware simulation server.



4.19. CONTROL TRANSFERS

4.19.1 PAUSE

Pauses the Operation for the specified amount of time. The minimum amount of time the operation can be paused is 1 millisecond.

Pause/Pause (Pause - Control)

Name:

Description:

Delay for

milliseconds

4.19.2 SCRIPT

Script operations allow you to create custom code into your operation. Code may be written in C#. DXC has some of its own functions which will be discussed below.

Script Examples/Script (1) (Script - Control)

Name: [Operation Failure]

Description:

Code

C#

Script

```
1
```

A. Variables

Variables that are defined within the Operations in the Configuration file can be accessed and manipulated from within the scripting functions. Variables are always available from the **variables** collection in a transfer. You can also reference them directly using the \$ notation

To set a value of a variable:

```
variables["$Variable1"].Value = 7
```

OR

```
$Variable1 = 7
```

To get the value of a variable:

```
(int)variables["$Variable1"]
```

OR

```
$Variable1
```

For these examples, there are 2 variables defined in the Operation. Variable1 and Variable2.

Variable	Default Value	DataType
Variable1	0	Auto
Variable2	0	Auto

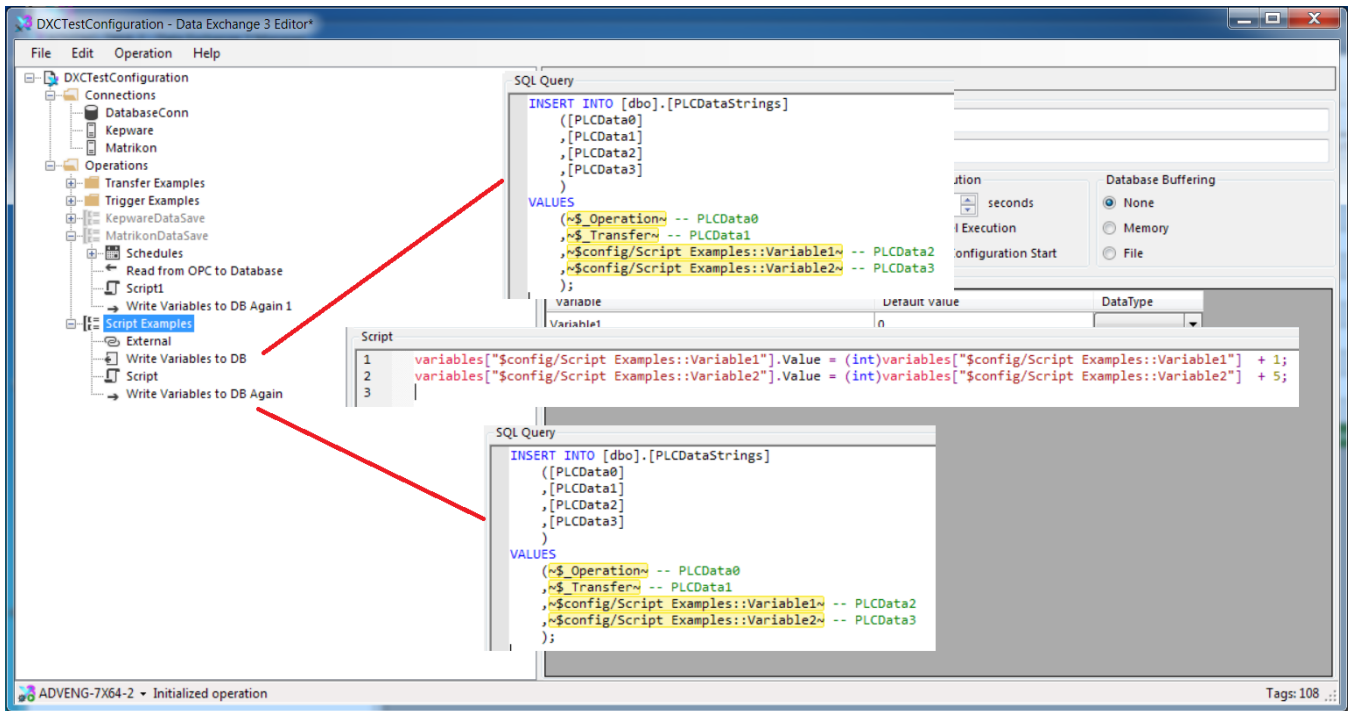
C# Script to increment a variable

Name:	Script
Description:	
Code	
C#	<input type="button" value="Browse..."/>
Script	
1	<code>variables["\$config/Script Examples::Variable1"].Value = (int)variables["\$config/Script Examples::Variable1"] + 1;</code>
2	<code> </code>

OR

```
$Variable1++;
```

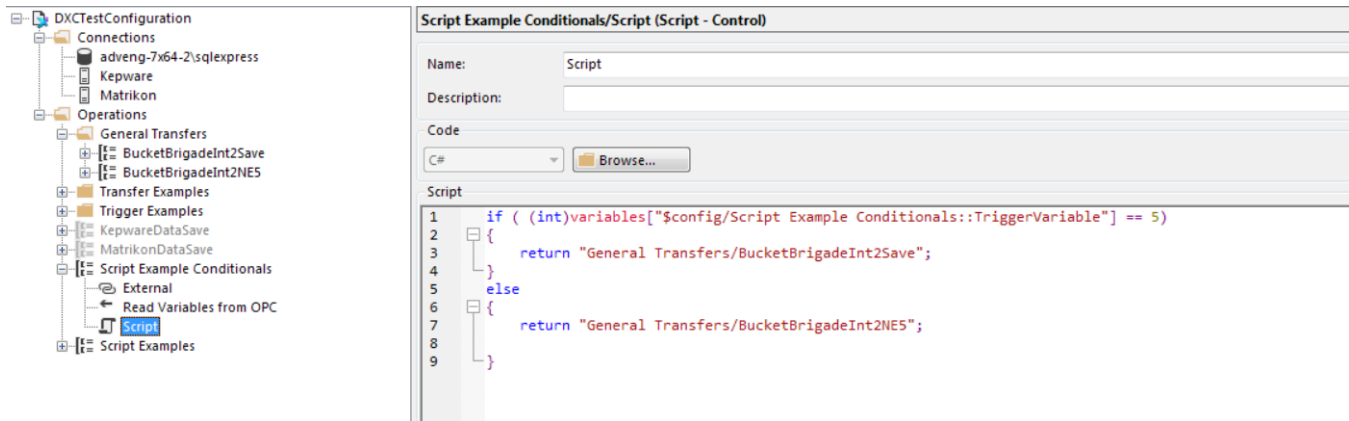
To see the modifications to the variable values, write the values to the database PLCDataStrings table, change the value and write the value again.



Running this Operation twice produces the following results:

PLCData0	PLCData1	PLCData2	PLCData3	P
Script Examples	Script Examples.Write Variables to DB	0	0	N
Script Examples	Script Examples.Write Variables to DB Again	1	5	N
Script Examples	Script Examples.Write Variables to DB	1	5	N
Script Examples	Script Examples.Write Variables to DB Again	2	10	N

Script to compare variables and branch to an Operation. In this example,



```

if ($TriggerVariable == 5)
{
    return "General Transfers/BucketBrigadeInt2Save";
}
else
{
    return "General Transfers/BucketBrigadeInt2NE5";
}

```

NextTransfer

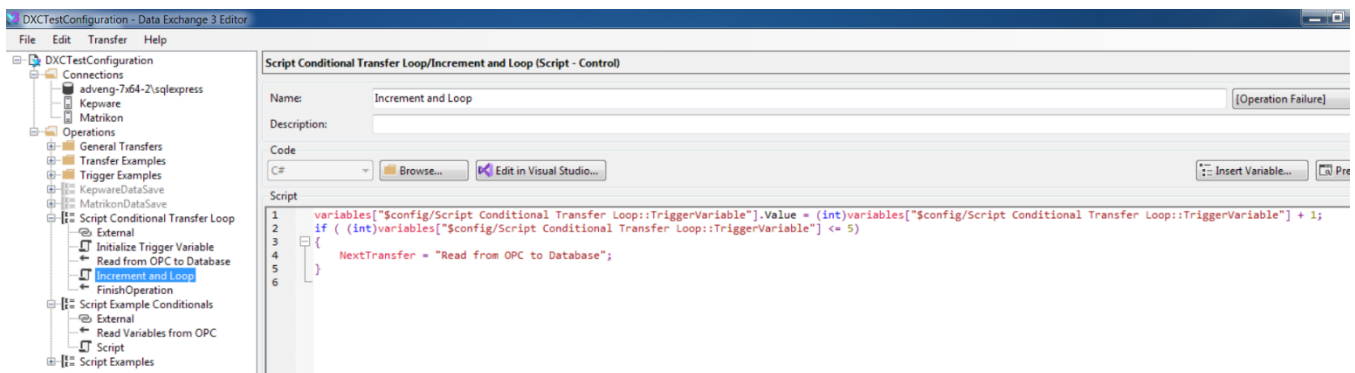
The NextTransfer function allows you to skip over transfers or loop back to a previous transfer within an Operation. In this example a variable is incremented and checked to see if it is greater than 5. If it is not, it will loop again. Each loop will write the value to a database.

The variable is first initialized and written to the database. It is then incremented and compared to 5. If it is higher than 5, the next Transfer “FinishOperation” is called and the operation is complete. Otherwise it loops back to “Read from OPC to Database”.

```

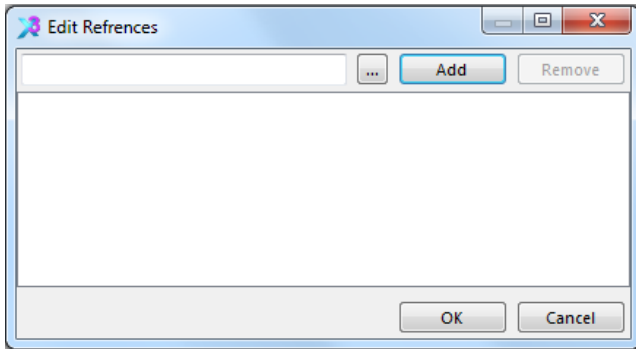
$TriggerVariable++;
if ( (int)variables["$config/Script Conditional Transfer Loop::TriggerVariable"] <= 5)
{
    NextTransfer = "Read from OPC to Database";
}

```



B. References

NEED INFO HERE

**C. Browse**

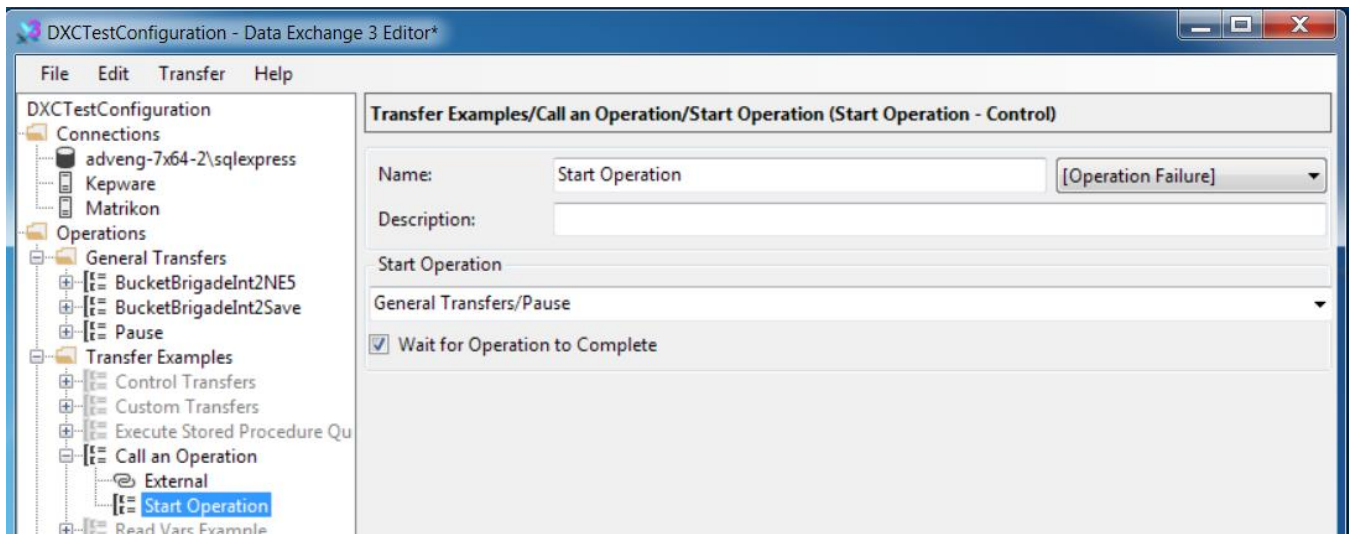
Browse script files to import into the Transfer

D. Edit in Visual Studio

Invokes Visual Studio and allows you to edit and test scripts in a more interactive environment.

4.19.3 START OPERATION

Call an operation from within another operation. If the invoked operation uses variables, those variables must be defined in the calling operation.

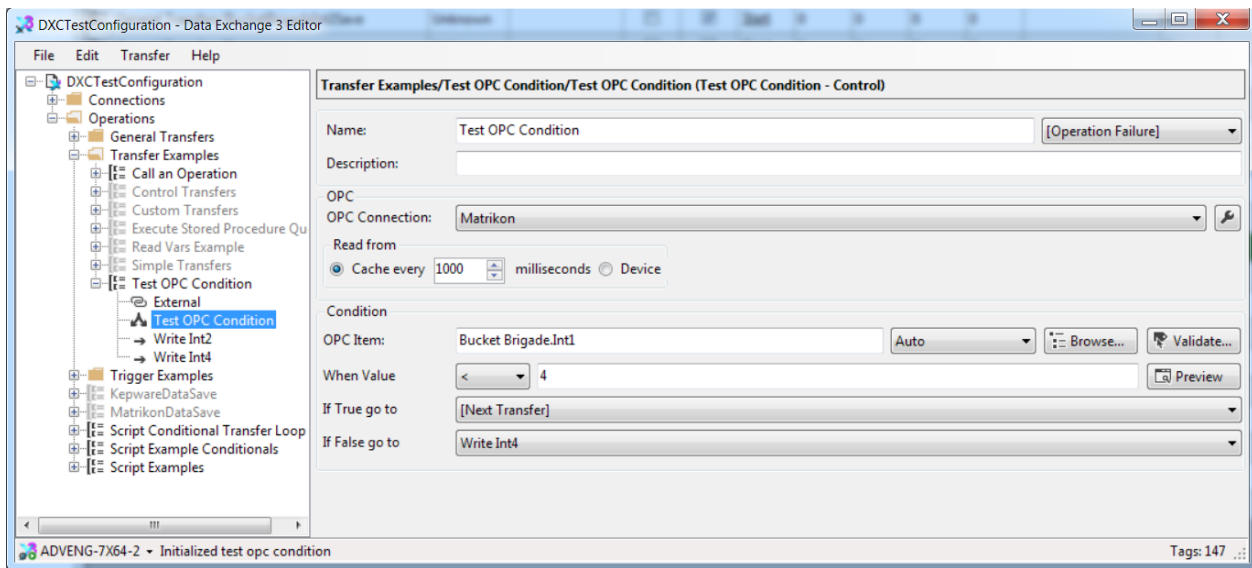


Wait for Operation to Complete

Click the Wait to have the operation wait for the called operation to complete before continuing. If the sequence of actions inside the operation do not matter, leave the button unchecked.

4.19.4 TEST OPC CONDITION

Checks the value of an OPC tag and calls a transfer according to its Boolean value. Once the specified transfer is executed, the remainder of the transfers in the operation are executed.

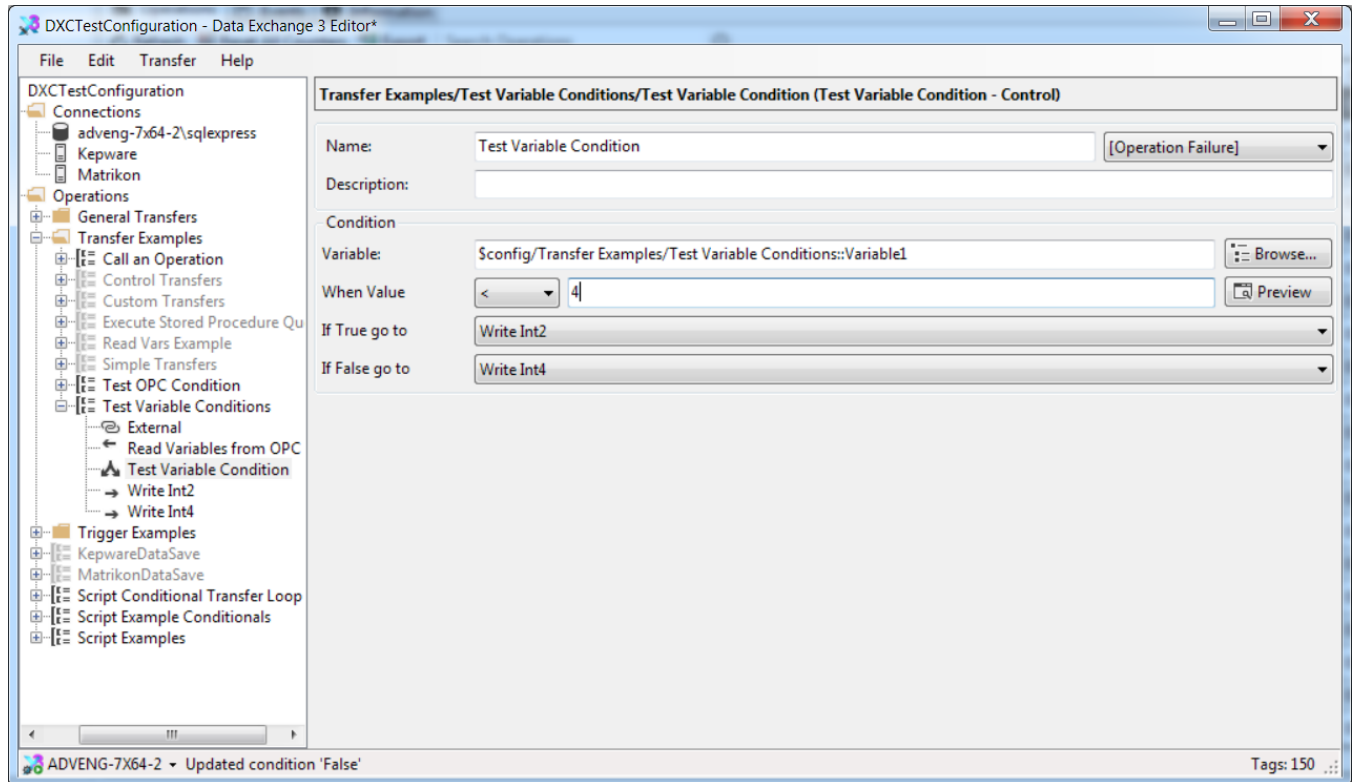


In this example, operation **Test OPC Condition** begins with transfer **Test OPC Condition** that checks the value of OPC Tag Bucket Brigade.Int1. If the value of the tag less than 4, the **Write Int2** transfer is executed, followed by the remaining transfer in the operation. If the value of the tag is 4 or more, **Write Int2** transfer is skipped, and the remaining transfer in the operation is executed.

4.19.5 TEST VARIABLE CONDITION

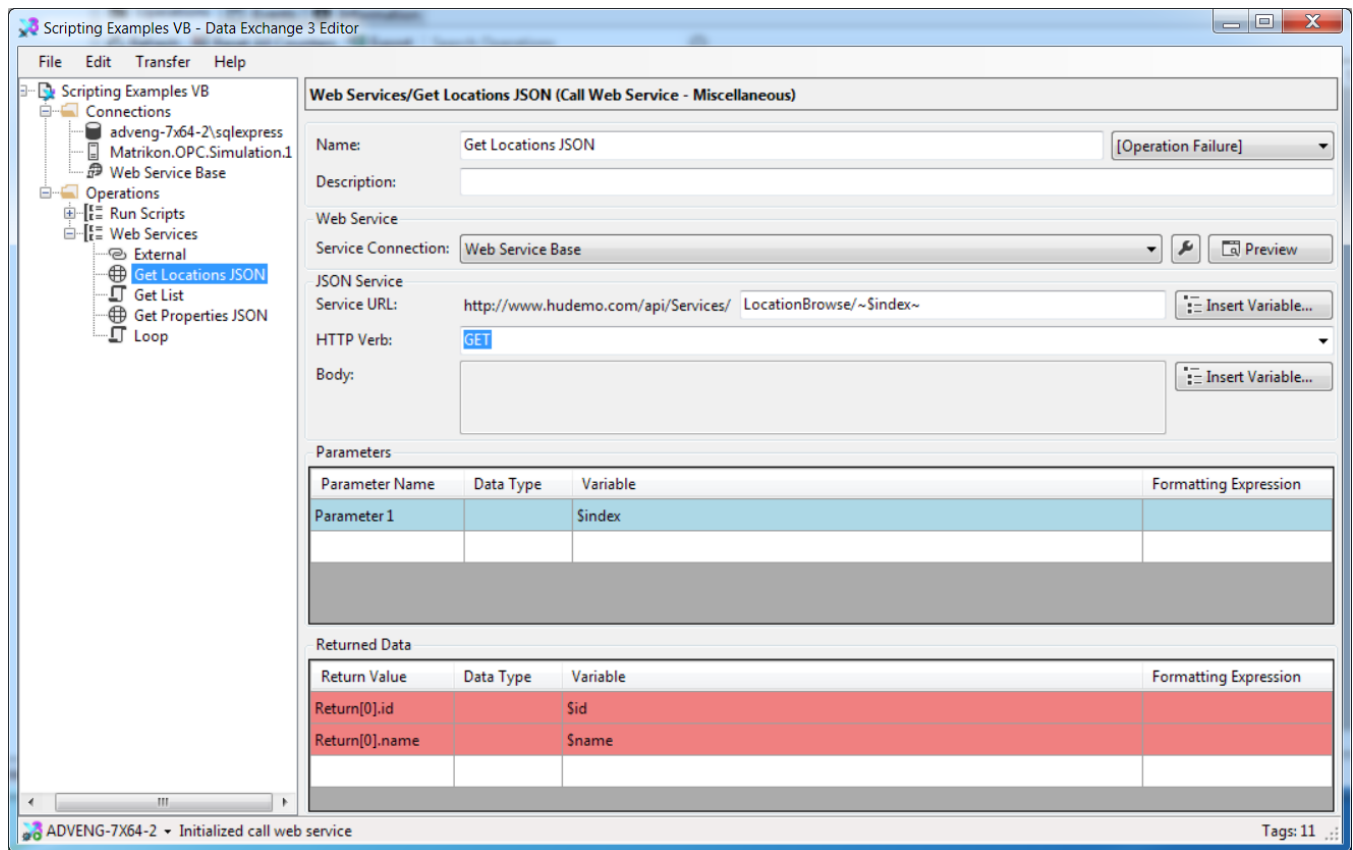
Tests the value of a variable and allows program execution to just to another transfer.

In this example, an OPC tag is read into a variable local to the Operation. If the value of the tag less than 4, the **Write Int2** transfer is executed, followed by the remaining transfer in the operation. If the value of the tag is 4 or more, **Write Int2** transfer is skipped, and the remaining transfer in the operation is executed.



4.20. MISCELLANEOUS TRANSFERS

4.20.1 CALL SOAP WEB SERVICE



GetList Transfer

```
Dim array As Object() = $ServiceResult
$ids =
  (From x In array
   Where Not x("folder")
   Select x("id")).ToArray()
```

```
$index = 1
```

Loop

```
$index = $index + 1
If $index < $ids.Length Then
  NextTransfer = "Get Properties JSON"
End If
```

Get Properties JSON

The screenshot shows the 'Scripting Examples VB - Data Exchange 3 Editor' window. On the left is a tree view with folders for 'Connections', 'Operations', and 'Web Services'. Under 'Web Services', 'Get Properties JSON' is selected. The main area is titled 'Web Services/Get Properties JSON (Call Web Service - Miscellaneous)'. It contains the following configuration fields:

- Name: Get Properties JSON
- Description: (empty)
- Web Service: (empty)
- Service Connection: Web Service Base
- JSON Service: (empty)
- Service URL: http://www.hudemo.com/api/Services/ TagInfo/~Sids~
- HTTP Verb: GET
- Body: (empty)

Below these fields are two tables:

Parameters

Parameter Name	Data Type	Variable	Formatting Expression
Parameter 1		Sids	#:value(\$index)

Returned Data

Return Value	Data Type	Variable	Formatting Expression
Return.tag_name		\$tag_name	

At the bottom of the window, a status bar shows 'ADVENG-7X64-2 - Initialized call web service' and 'Tags: 11'.

4.20.2 CALL JSON WEB SERVICE

4.20.3 SEND EMAIL

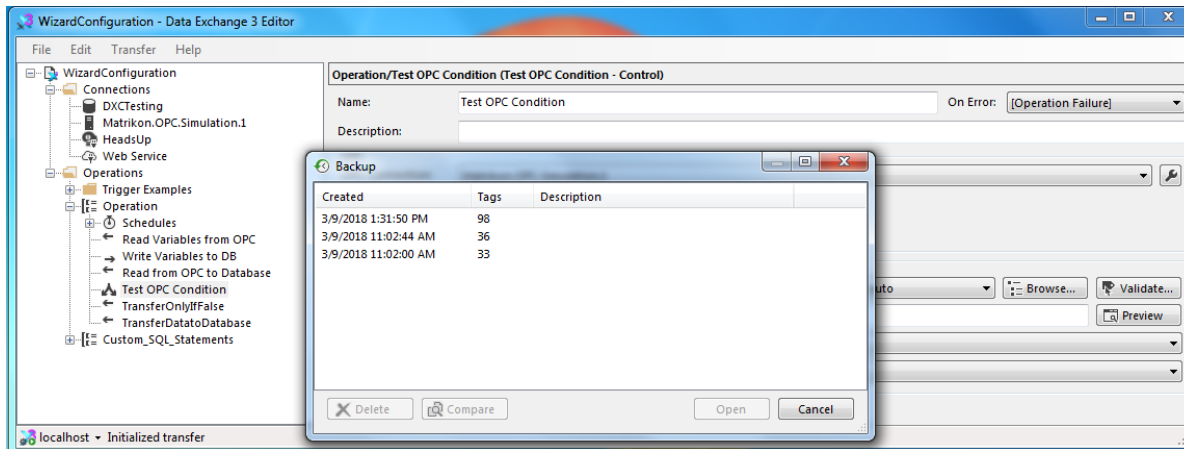
4.20.4 SOCKET

4.21. BACKUPS

During initial setup of a Data Exchange file, the configuration page will appear. This page displays the current name of the file, date, description and the number of backups to keep. All backups as well as the currently running file are stored in the .dxc3 file created and modified using the editor.

WizardConfiguration	
Name:	WizardConfiguration
Description:	
Date Modified:	11/15/2017 10:33:31 AM
File Path:	C:\Users\albrece\Documents\Data Exchange Configurations\VRT1-ELA\WizardConfiguration.d
Backups to Keep:	15
Tag Count:	69

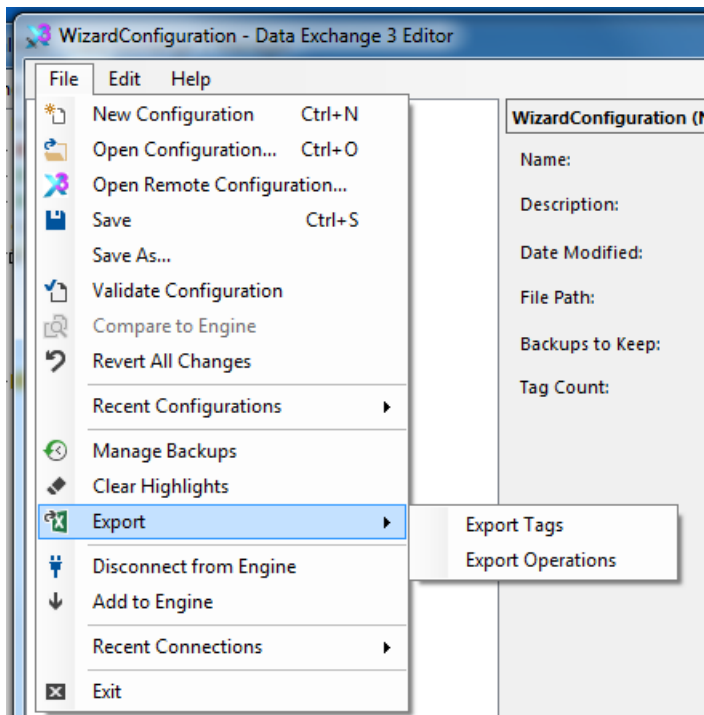
To view the backups select “Manage Backups” from the Configuration menu. Adding a description for each new version of the configuration file helps document your system.



To revert to an older version of the configuration file, save the current version of the file, select the version in which to revert and select Open. Save this version under the same file name or Save AS another file.

4.22. EXPORTING TAGS & OPERATIONS

The Editor allows you to export all the tags and operations that are used in the Configuration file.



4.22.1 TAG DATA

\\localhost\Matrikon.OPC.Simulation.1	
Bucket Brigade.ArrayOfString[0]	CustomSQLOperations/Write to OPC.Write to OPC from Database
Bucket Brigade.ArrayOfString[0]	Write to OPC from Database.Write to OPC from Database
Bucket Brigade.ArrayOfString[1]	CustomSQLOperations/Write to OPC.Write to OPC from Database
Bucket Brigade.ArrayOfString[1]	Write to OPC from Database.Write to OPC from Database
Bucket Brigade.ArrayOfString[2]	CustomSQLOperations/Write to OPC.Write to OPC from Database
Bucket Brigade.ArrayOfString[2]	Write to OPC from Database.Write to OPC from Database
Bucket Brigade.Int1	CustomSQLOperations/Custom SQL.Read Variables from OPC
Bucket Brigade.Int1	Transfer Examples/Read Write Variables to OPC.Read Variables from OPC
Bucket Brigade.Int1	Variables/VariablesExample.Read Variables from OPC
Bucket Brigade.Int2	CustomSQLOperations/Custom SQL.Read Variables from OPC
Bucket Brigade.Int2	Transfer Examples/Read Write Variables to OPC.Read Variables from OPC
Bucket Brigade.Int2	Variables/VariablesExample.Read Variables from OPC
Bucket Brigade.UInt1	Transfer Examples/Read Write Variables to OPC.Write Variables to OPC
Bucket Brigade.UInt2	Transfer Examples/Read Write Variables to OPC.Write Variables to OPC
Random.Boolean	ControlOperations/TestCondition.Test OPC Condition
Random.Int1	CustomSQLOperations/Read OPC to Database.Read from OPC to Database
Random.Int1	OPC to DB.Read from OPC to Database
Random.Int1	Transfer Examples/OPC Receipt.Write OPC Receipt
Random.Int2	CustomSQLOperations/Read OPC to Database.Read from OPC to Database
Random.Int2	OPC to DB.Read from OPC to Database
Random.Int4	CustomSQLOperations/Read OPC to Database.Read from OPC to Database
Random.Int4	OPC to DB.Read from OPC to Database

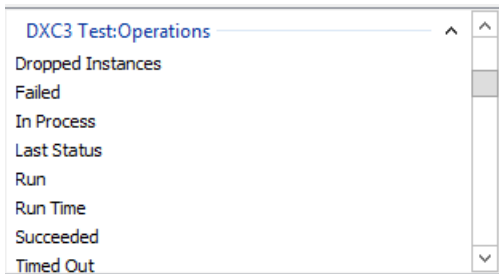
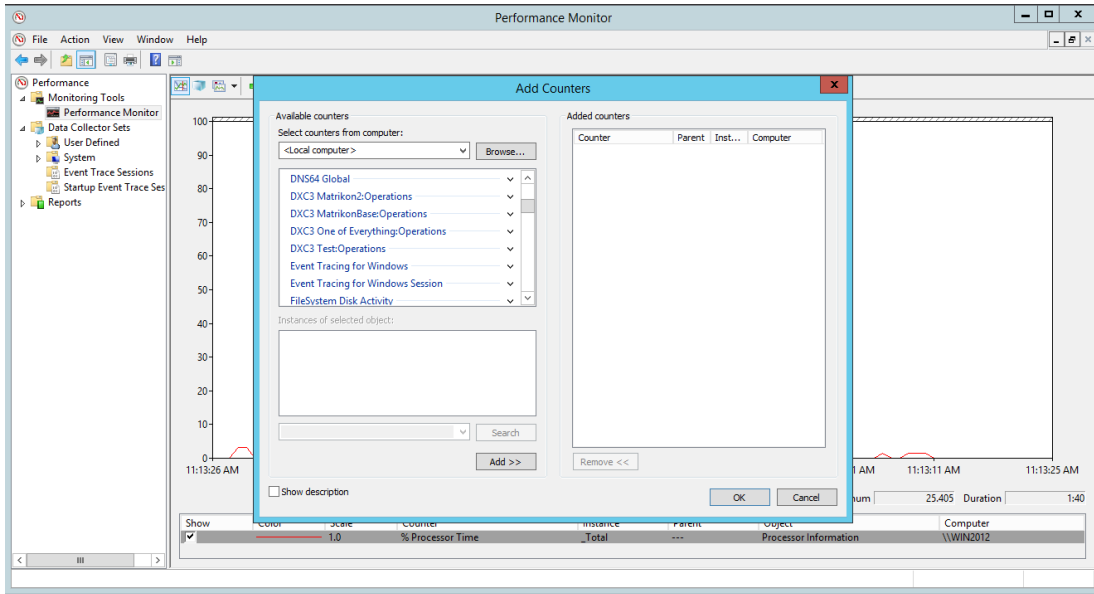
4.22.2 OPERATION DATA

ControlOperations	
Pause	Schedule
PauseOneSecond	Operation Pause
ScriptOperation	Schedule
IncrementGlobal	Run Script
Write Variables to DB	Variable DB Write
StartOperation	Schedule
Start Operation	Start Operation
TestCondition	Schedule
Test OPC Condition	Test OPC Condition
TransferOnlyIfFALSE	Simple OPC Read
TransferDataToDatabase	Simple OPC Read
Write OPC Receipt	Write OPC Receipt
CustomSQLOperations	
Custom SQL	Schedule
Read Variables from OPC	Variable OPC Read
Execute Custom SQL	Advanced Variables DB
Read OPC to Database	Schedule
Read from OPC to Database	Advanced OPC Read
Write to OPC	Schedule
Write to OPC from Database	Advanced OPC Write

SECTION 5. PERFORMANCE MONITORING

Local Computer

DXC3



SECTION 6. TROUBLESHOOTING