Layer2 Cloud Connector User Documentation

22nd of February 2017 - Version 7.7.0.0

Layer 2 GmbH | Eiffestraße 664b | D-20537 Hamburg
Tel.: +49 (40) 28 41 12 – 30 | Fax: +49 (40) 28 41 12 – 16
E-Mail: sales@layer2solutions.com | Web: www.layer2solutions.com/
General Managers: Wolfgang Cords, Matthias Hupe
Hamburg District Court: HRB 81259

Contents

Overview

The Layer2 Cloud Connector provides an easy and powerful way to synchronize or replicate content from many different data sources. Originally designed for Microsoft SharePoint and Office 365 integration, the Layer2 Cloud Connector has now become an all-purpose synchronization tool, allowing files and records to be synchronized between a virtually unlimited number of different data sources, such as Microsoft SharePoint, Office 365, Exchange, Dynamics, SQL servers, local files, and more. By integrating with Microsoft's ActiveX Data Objects (ADO.NET) platform, the Layer2 Cloud Connector is able to connect to a vast number of third-party data sources.

Getting Started

System Requirements

Minimum Requirements

The system requirements for a machine to run the Layer2 Cloud Connector are highly dependent on the amount of data that needs to be synchronized. The Layer2 Cloud Connector can run on any machine which has a supported Windows operating system installed. However, if large files need to be synchronized (for example, those greater than 1 GB), the host-machine should at least have 2 GB free for the Layer2 Cloud Connector to use. More memory is also recommended for large data syncs with hundreds of thousands of records.

Note: 32-bit installs can only use up-to 4GB of memory. If your scenario involves large files (2 GB+) or a large number of records (100K+), it is recommended that you install the 64-bit version, if possible, so that you do not run into memory issues.

Supported Operating Systems

The Layer2 Cloud Connector is supported on the following operating systems:

- Microsoft Windows 7 (Service Pack 1)
- Microsoft Windows 8
- Microsoft Windows 8.1
- Microsoft Windows 10
- Microsoft Windows Server 2008 (Service Pack 2)
- Microsoft Windows Server 2008 R2 (Service Pack 1)
- Microsoft Windows Server 2012 (Standard and R2)

Virtual Machines

Running the Cloud Connector on Virtual Machines (VM) locally or online, such as in the Microsoft Azure Cloud, is fully supported as long as the VM is running one of the above operating systems. It is important that the VM is able to reach your data source and destination, directly or via VPN.

Installing on SharePoint or Database Servers

While possible, it is not a best-practice to install the Cloud Connector on SharePoint, Database, or other Application servers. It is advised that you install on a separate Windows server or client in the network, where it could reach all the systems it needs to connect with.

Dependencies

The Layer2 Cloud Connector depends on the following components:

- Microsoft .NET Framework 4.5
- Microsoft Management Console 3.0

If the proper .NET Framework is not present on the host machine, the Cloud Connector will give an error during installation that it is missing. Please see the MSDN documentation for how to install the .NET Framework.

Installation

Choosing an Installer Package

The Layer2 Cloud Connector is provided with four different installation packages, for 32-bit (x86) and 64-bit (x64) systems, based on the Common Language Runtime (CLR) supported by the Windows Management Console:

- x64 (for .NET 4.5)
- x64 .NET 3.5
- x86 (for .NET 4.5)
- x86 .NET 3.5

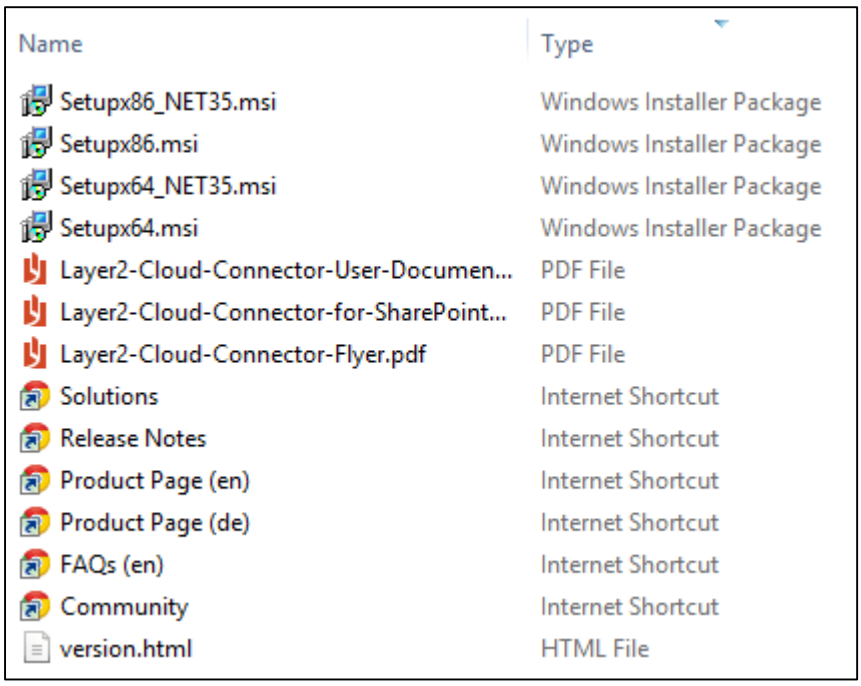


Figure 1 - Contents of installer package showing the different types

The .NET version of the installer to be used depends on the host-machine's Windows version. The installer for .NET 3.5 targets the CLR 2.0 whereas the .NET 4.5 edition targets CLR 4. For the recommended package, see the Server Operating Systems and Client Operating Systems lists below.

The 32-bit package can be installed and run on both 64-bit and 32-bit architectures, whereas the 64-bit package is only applicable for 64-bit systems. The Layer2 Cloud Connector is able to access data through any ADO.NET data provider, but some third-party providers might only support a specific processor architecture, especially older ones. Please check the third-party data provider documentation for more information.

If the 32-bit version of the connector is installed, only the 32-bit providers will be available; when the 64-bit package is installed, only 64-bit providers will be available. All Layer2 ADO.NET data providers which are delivered with the Layer2 Cloud Connector will support both architectures, as will most third-party providers.

Layer2 Cloud Connector Compatibility Matrix	.NET 3.5 (CLR 2.0)	.NET 4.5 (CLR 4)
Windows Server 2012	✓	✓
Windows Server 2012 R2	✓	✓
Windows 8	✓	✓
Windows 8.1	✓	✓
Windows 10	✓	✓
Windows Server 2008	✓	✗
Windows Server 2008 R2	✓	✗
Windows 7	✓	✗
Windows Server 2003 *	✓	✗
Windows Vista *	✓	✗
Windows XP *	✓	✗

** While there are no technical restrictions for using the Layer2 Cloud Connector on these platforms, please note that they are no longer supported.*

Server Operating Systems

Windows Server 2012 / Windows Server 2012 R2

Any installer package can be used.
Recommended: x64 for .NET 4.5

Windows Server 2008 / Windows Server 2008 R2

Either installer package x64 for .NET 3.5 or x86 for .NET 3.5 can be used.
Recommended: x64 for .NET 3.5

Windows Server 2003 32-bit

The installer package x86 for .NET 3.5 **must** be used.
Note: While there are no technical restrictions for using the Layer2 Cloud Connector on this platform, please be aware that it is no longer supported.

Windows Server 2003 64-bit

Either installer package x64 for .NET 3.5 or x86 for .NET 3.5 can be used.

Recommended: x64 for .NET 3.5

Note: While there are no technical restrictions for using the Layer2 Cloud Connector on this platform, please be aware that it is no longer supported.

Client Operating Systems

Windows 10 64-bit

Any installer package can be used.

Recommended: x64 .NET 4.5

Windows 10 32-bit

Either installer package x86 for .NET 3.5 or x86 for .NET 4.5 can be used.

Recommended: x86 for .NET 4.5

Windows 8 / Windows 8.1 64-bit

Any installer package can be used.

Recommended: x64 .NET 4.5

Windows 8 / Windows 8.1 32-bit

Either installer package x86 for .NET 3.5 or x86 for .NET 4.5 can be used.

Recommended: x86 for .NET 4.5

Windows 7 64-bit

Either installer package x64 for .NET 3.5 or x86 for .NET 3.5 can be used.

Recommended: x64 for .NET 3.5

Windows 7 32-bit

The installer package x86 for .NET 3.5 **must** be used.

Windows Vista / Windows XP 64-bit

Either installer package x64 for .NET 3.5 or x86 for .NET 3.5 can be used.

Recommended: x64 for .NET 3.5

Note: While there are no technical restrictions for using the Layer2 Cloud Connector on this platform, please be aware that it is no longer supported.

Windows Vista / Windows XP 32-bit

The installer package x86 for .NET 3.5 **must** be used.

Note: While there are no technical restrictions for using the Layer2 Cloud Connector on this platform, please be aware that it is no longer supported.

Setup

1. Extract all files from the provided .ZIP file into a folder on the host-machine.
2. Using the information provided above about the four installer packages, select the correct release version for the operating system you are installing onto.
3. Run the **Setup*.msi** file inside the appropriate release folder.
4. Read the license agreement carefully and accept it by clicking the box. If you have any questions concerning licensing, please do not hesitate to contact sales@layer2solutions.com. Otherwise, click **Next**.

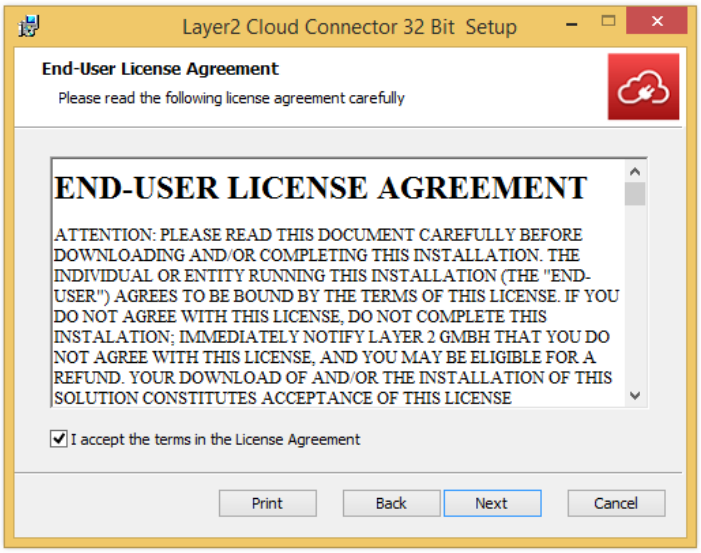


Figure 2 - Installer license agreement

5. Select appropriate install type. If unsure, select **Typical**.

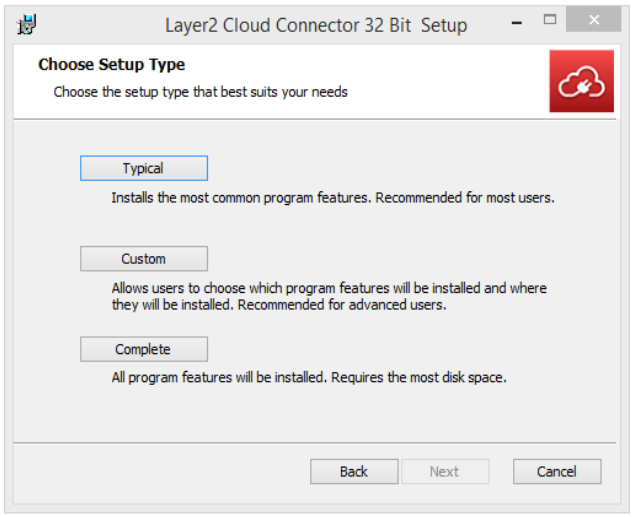


Figure 3 - Installer setup type selection

6. Click **Install** to start.

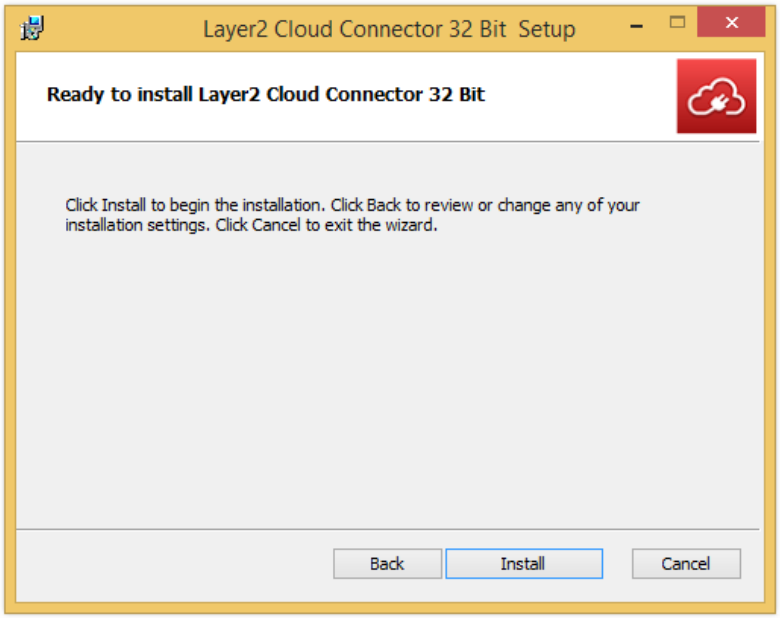


Figure 4 - Installation complete

7. Once the installer is done, you will have the option to launch the Connection Manager right away by checking the box (recommended). Click **Finish** to complete the installation process.
8. Install the license key file. See the Installing a License section for more details.

Congratulations! You have successfully installed the Layer2 Cloud Connector, and are now ready to get started with using the Connector to connect and synchronize data between different data sources.

Configuring and Executing Connections

Below are the basic steps necessary to set up a connection between two data sources (referred to as data entities henceforth), along with an example. For more detailed instructions on how to set up a connection to a specific source, please see the online examples here.

1. Open the Layer2 Cloud Connector Connection Manager application (this is listed as **Start Connection Manager** in the Start page/menu). You'll see that the Connection Manager UI is divided into three parts:
    - The Connection list pane (left-hand side)
    - The Properties pane (middle)
    - The Actions pane (right-hand side)

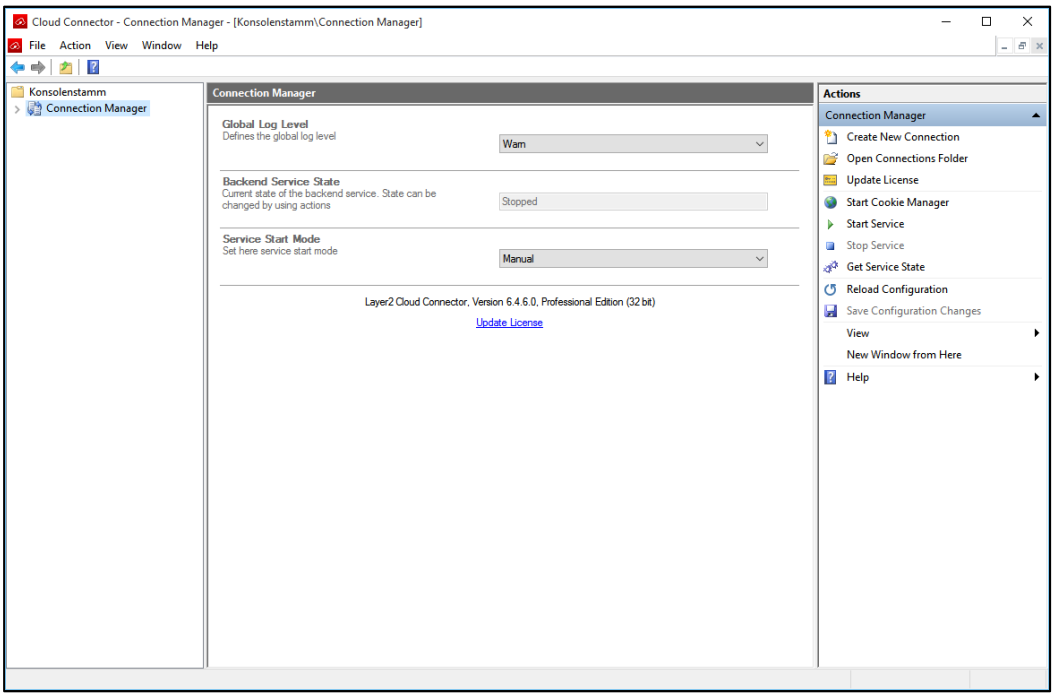


Figure 5 - Layer2 Cloud Connector Connection Manager Application

These panes and their attributes are covered in more detail in the Configuration section.

2. In the right-hand pane, click **Create New Connection**. The new connection will appear at the bottom of the Connection Manager list.
    a. You will also see there are a number of example connections that you can use as a template to start your own connection. Right-click an example connection that fits your requirements and select **Duplicate Connection** to make a new copy to edit.

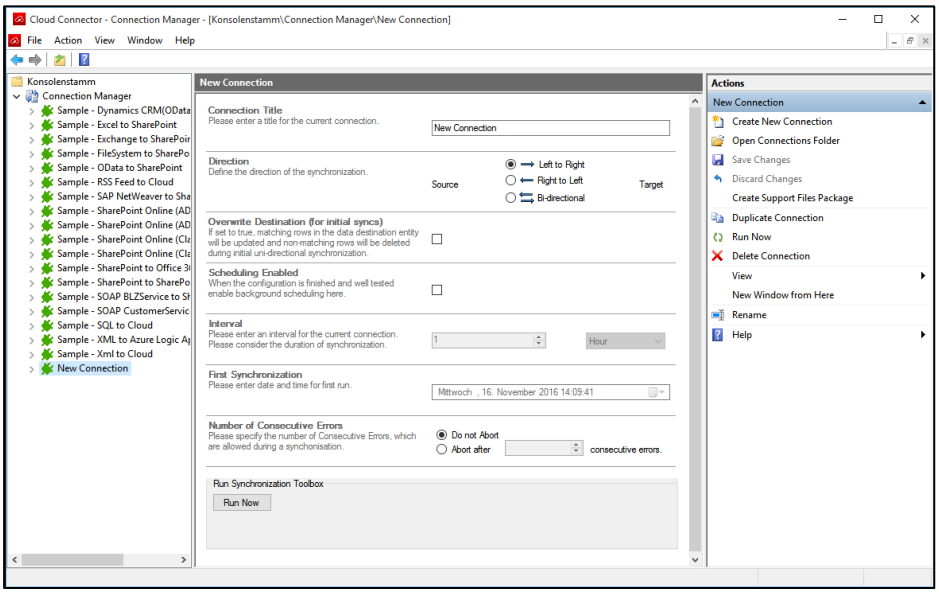


Figure 6 - Creating new connection

3. Click on the **New Connection** in the left-hand pane to see its properties.
    - **Connection Title:** Set a meaningful title for your connection.
    - **Direction:** Specify uni- or bi-directional sync of data.
    - **Overwrite Destination (for initial syncs):** [Advanced] Allows for record cleanup on target data entity for uni-directional connections.
    - **Scheduling enabled:** [Advanced] For enabling automatic background synchronization, see the Schedule Synchronization section. It is **recommended** that you do not enable this until you have a fully functional connection configured.
        - **Interval**
        - **First Synchronization**
    - **Number of Consecutive Errors:** [Advanced] Set a number of errors permitted before sync is aborted.
      See the Configuration section for a detailed description of each property.
4. Once you have set the **Connection Title**, the **Direction**, and any additional properties, click **Save the changes** in the right-hand pane to save your settings.
5. Expand your new connection and then **Data Entities** to see the objects for the Source and Target data entities for the connection.

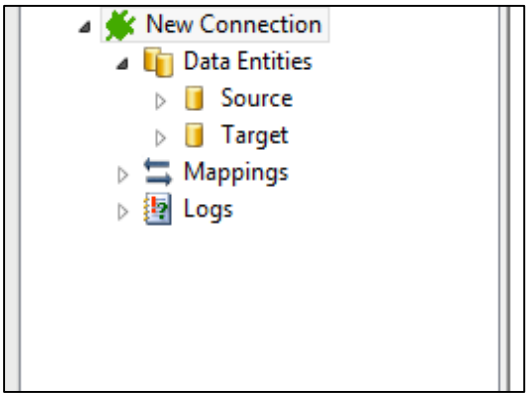


Figure 7 - New connection Data Entities

6. Click **Source** to see its properties (these may change based on the selected provider and connection type):
    a. **Data Entity Title:** Set a meaningful title for your Source data entity.
    b. **Data Provider:** Select the correct provider for the data entity. See the _Layer2_ADO_Providers section for specific information.
    c. **Connection String:** Set the necessary parameters to connect to the data entity. See the Layer2 Data Providers section for specific information.
    d. **Select Statement:** Set a provider-specific query, if required, to retrieve the right data.
    e. **Password:** This field can be used to define the password parameter for the provider.
    f. **Primary Key(s):** Usually the data entity will provide this automatically, but if not, one can be defined here.
    g. **Dynamic Columns:** [Advanced Settings] You can define additional columns here based on calculations, logic expressions, or even custom C# code. For more information, see the Dynamic Columns.
    h. **Replication Column:** [Advanced Settings] Uses a field in the data that would be the primary key and the Cloud Connector will auto-generate a GUID into it to resolve replication issues.

i. The **Advanced Settings** section can be expanded by clicking the header row.

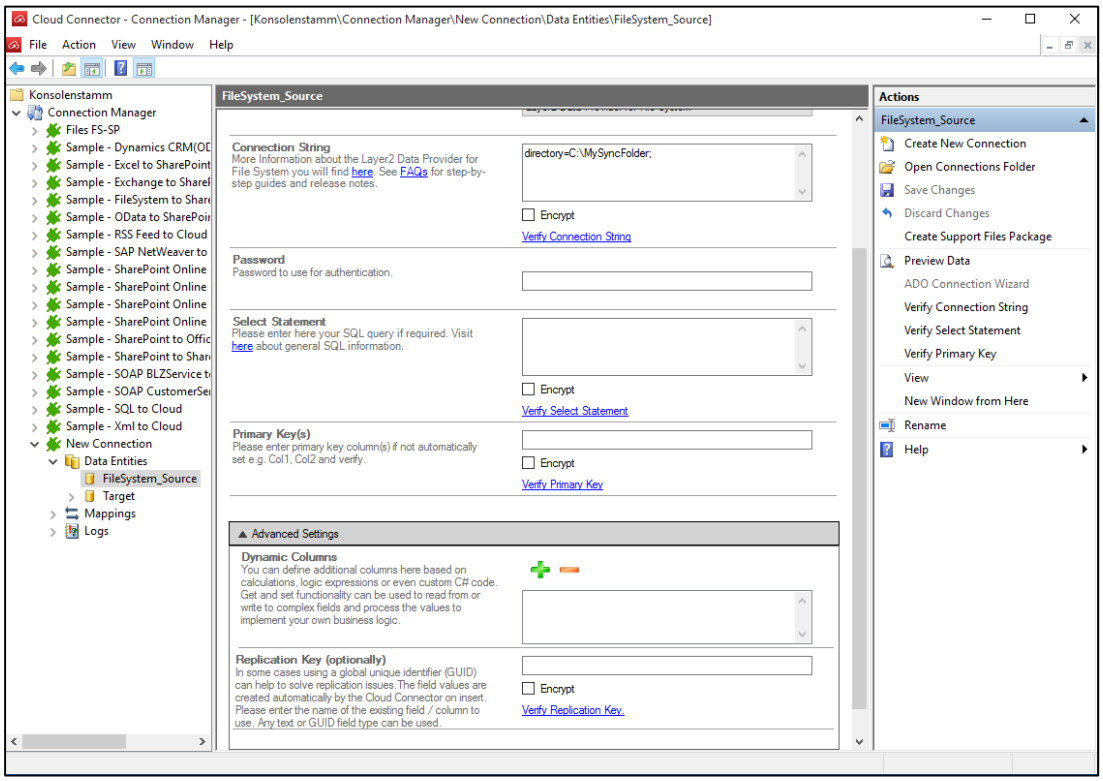


Figure 8 - Blank source Data Entity

See the Configuration section for a detailed description of each property.

Example:

For a connection that goes from a File System source to a SharePoint document library target, these are the resulting settings:

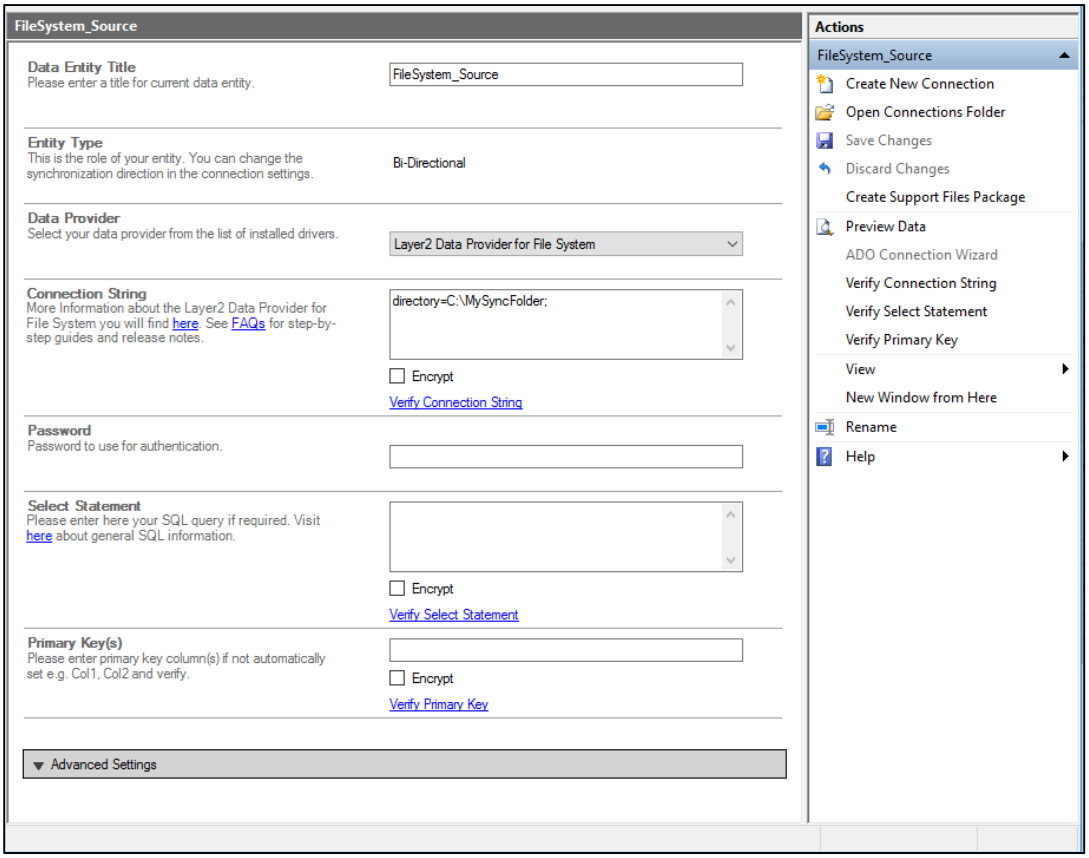


Figure 9 - Example connection for File System

7. Once you have set the **Data Entity Title**, the **Data Provider**, and other required properties, verify the fields using the **Verify *** link, and then click **Preview Data** in the right-hand pane to make sure you have a working connection. If everything is working, click **Save the changes** in the right-hand pane to save your settings.
8. Click **Target** to see its properties (these may change based on the selected provider):
    a. **Data Entity Title:** Set a meaningful title for your Target data entity.
    b. **Data Provider:** Select the correct provider for the data entity. See the Layer2 Data Providers section for specific information.
    c. **Connection String:** Set the necessary parameters to connect to the data entity. See the Layer2 Data Providers section for specific information.
    d. **Password**: This field can be used to define the password parameter for the provider.
    e. **Primary Key(s):** Usually the data entity will provide this automatically, but if not, one can be defined here.
    f. **Ignore Changes Within Target**: If you are sure that there are no data changes in the target system at all, you can speed up the synchronization by enabling this option.

g. **Dynamic Columns:** [Advanced Settings] You can define additional columns here based on calculations, logic expressions, or even custom C# code. For more information, see the Dynamic Columns section.
h. **Replication Column:** [Advanced Settings] Used for resolving replication issues.
i. **Disable Operations:** [Advanced Settings] Used to disable certain transactions during synchronization.

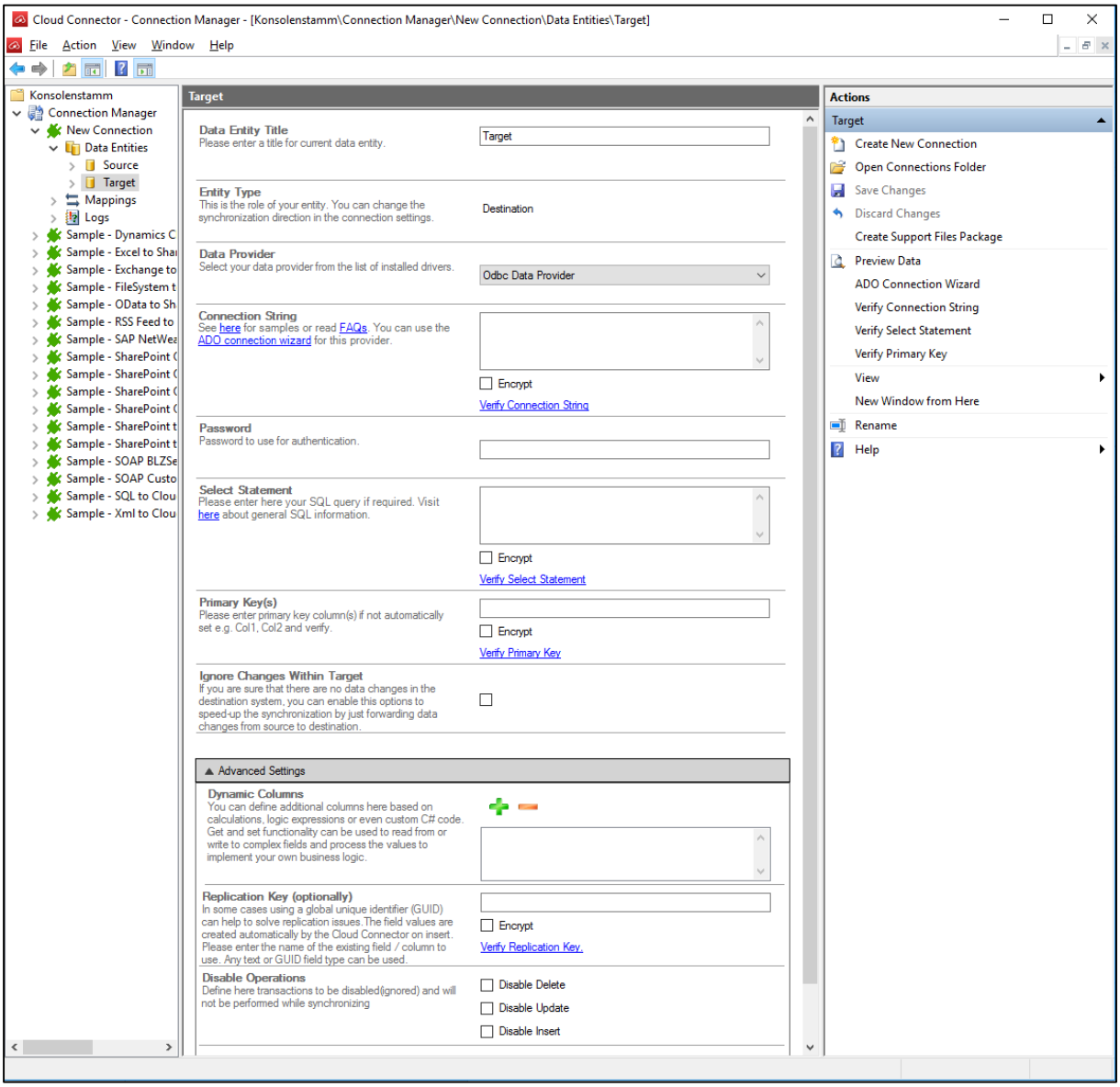


Figure 10 - Blank target Data Entity

See the Configuration section for a detailed description of each property.

Example:

For a connection that goes from a File System source to a SharePoint document library target, these are the resulting settings:

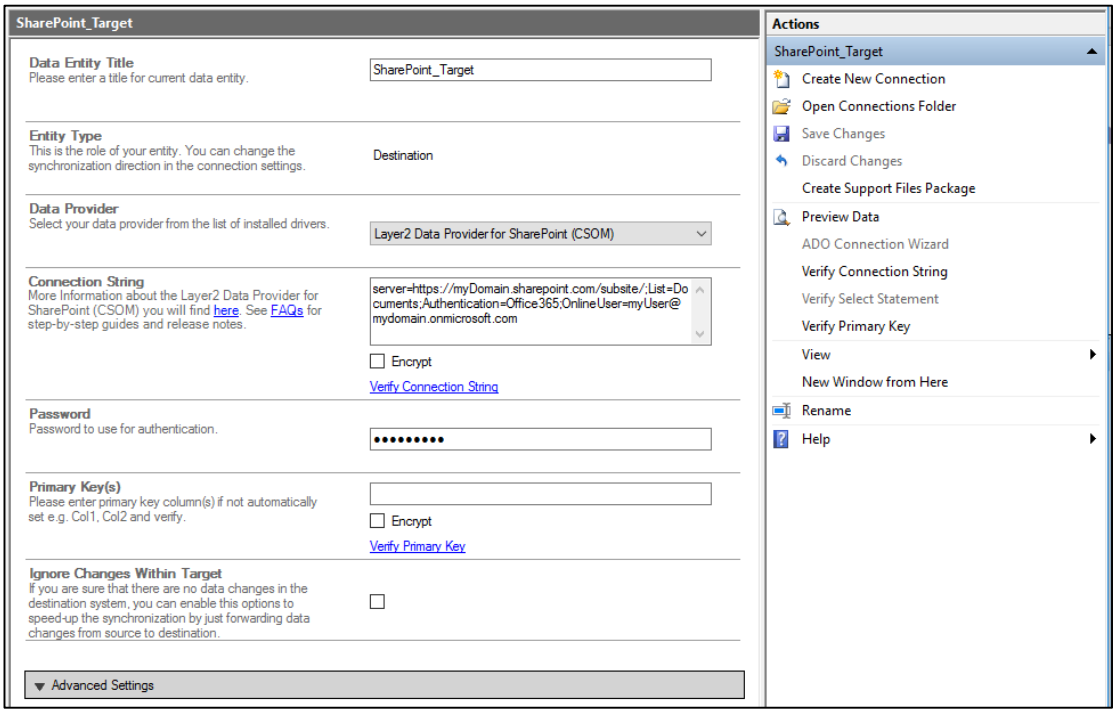


Figure 11 - Example connection for SharePoint

9. Once you have set the **Data Entity Title**, the **Data Provider**, and other required properties, verify the fields using the **Verify *** link, and then click **Preview Data** in the right-hand pane to make sure you have a working connection. If everything is working, click **Save the changes** in the right-hand pane to save your settings.
10. Click **Mappings** in the left-hand Connections pane to set the mapping of fields from the source to the target. Click **Enable Auto Mapping** to have the Connection Manager perform this for you based on identical column/field names from the data entities (otherwise, you will need to set them manually).

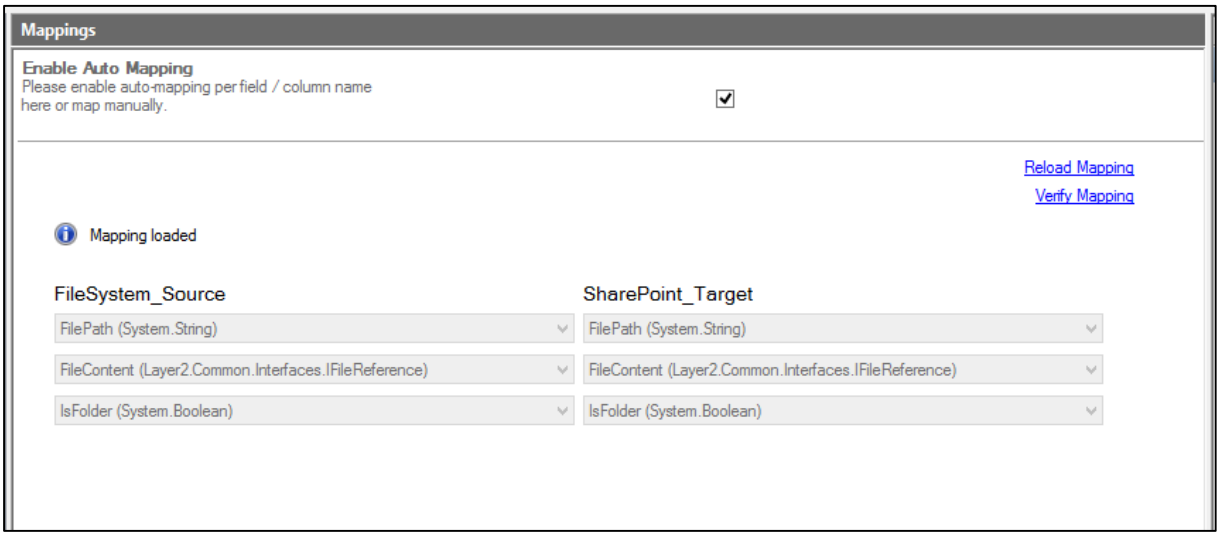


Figure 12 - Example with Auto Mapping for a file system source and SharePoint document library target

See the Configuration section for a detailed description of the mapping functionality.

Example:
Setting up manual mapping for the File System to SharePoint connection.

To add a new mapping pair, click the **green '+'**.

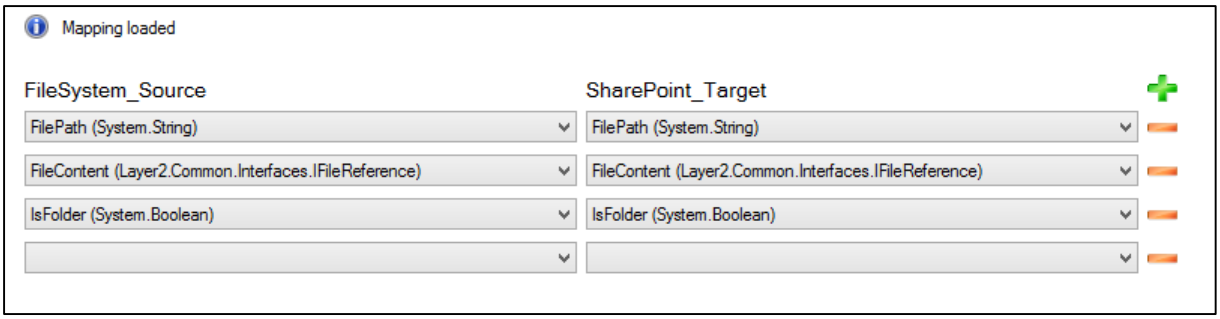


Figure 13 - Adding a new mapping pair

From the drop-downs, select the fields that are a matching pair for synchronization.

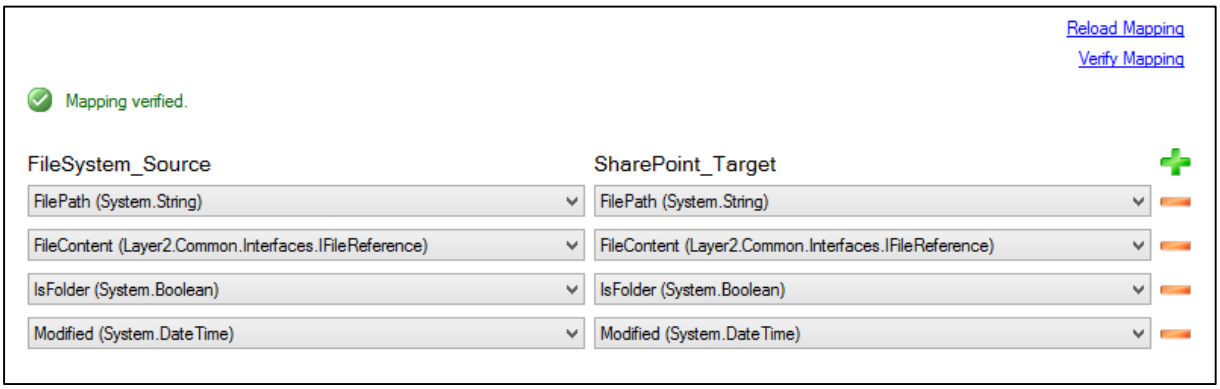


Figure 14 - Setting new mapped pair for Modified column

Add more fields as necessary by clicking the **green '+'**.

11. When done, click **Save the changes** and then click **Verify Mapping** to test.
12. Now you're ready to run the synchronization! Select your connection in the left-hand pane, and then click **Run now** in the central pane. Check the **Logs** for any warnings or errors and verify that the content from the Source was pushed to the Target (or to each other in the case of bi-directional).

    Example:

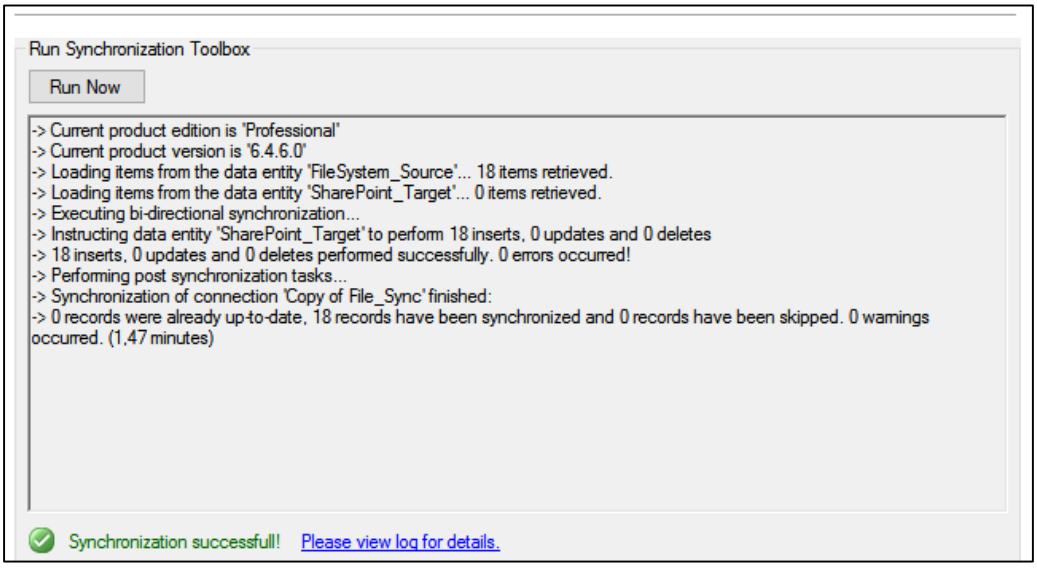


Figure 15 - Running the File System to SharePoint connection first time with sync summary data

Schedule Synchronization

Once you have your connection running as expected without errors when it is run manually, you can now set a schedule to execute automatically in background (without needing to run the Connection Manager). Below are the basics steps necessary to set up automated synchronization schedule for your connection. For more information on the scheduling properties, see the Configuration section.

1. In the Connection Manager, select the connection from the list that you wish to set up scheduling for.
2. Check the box for **Scheduling enabled** in the properties pane.

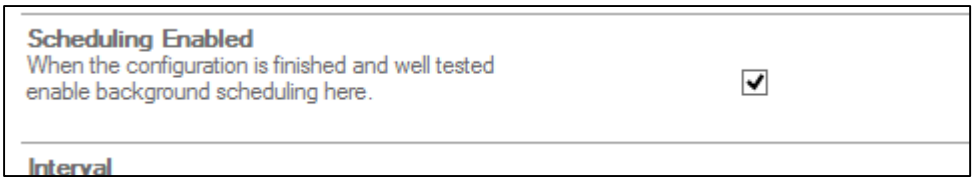


Figure 16 - Scheduling Enabled property

3. Determine the **Interval** you wish the synchronization to occur at. You can specify by "Minute", "Hour", and "Day", as well as a number value for **Interval**.
   Warning! Make sure to set an appropriate interval based on how long the synchronization of your specific connection usually takes.
4. Set the **First Synchronization** property. This will determine when the automatic synchronization will start, as well as what time subsequent runs will be performed.

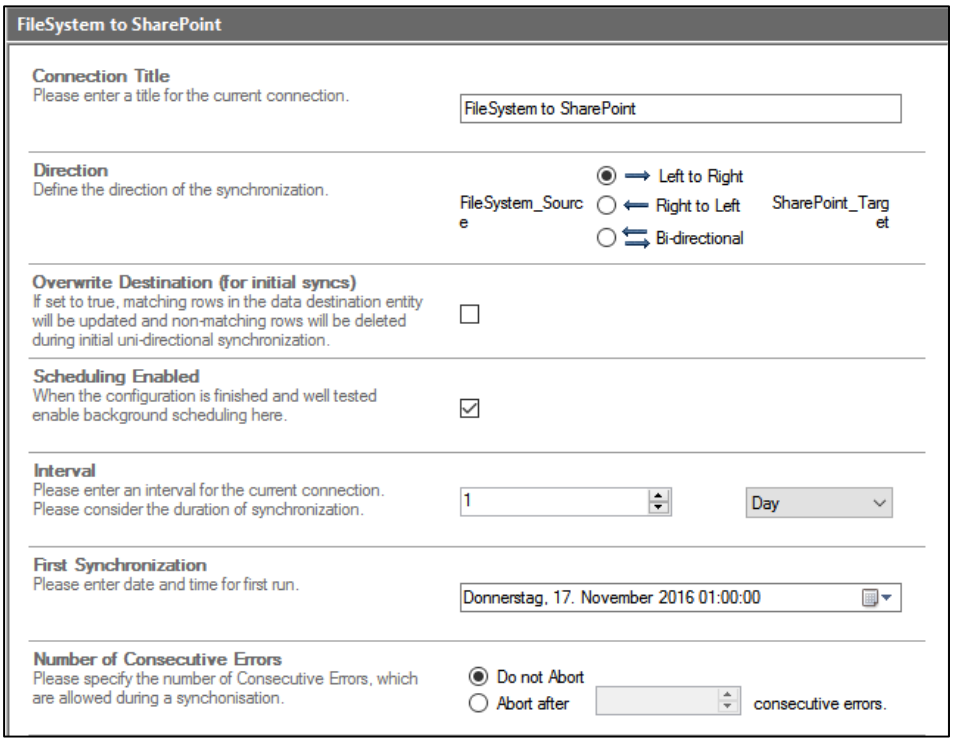


Figure 17 – Example: File System to SharePoint set to run once a day at 01:00 AM

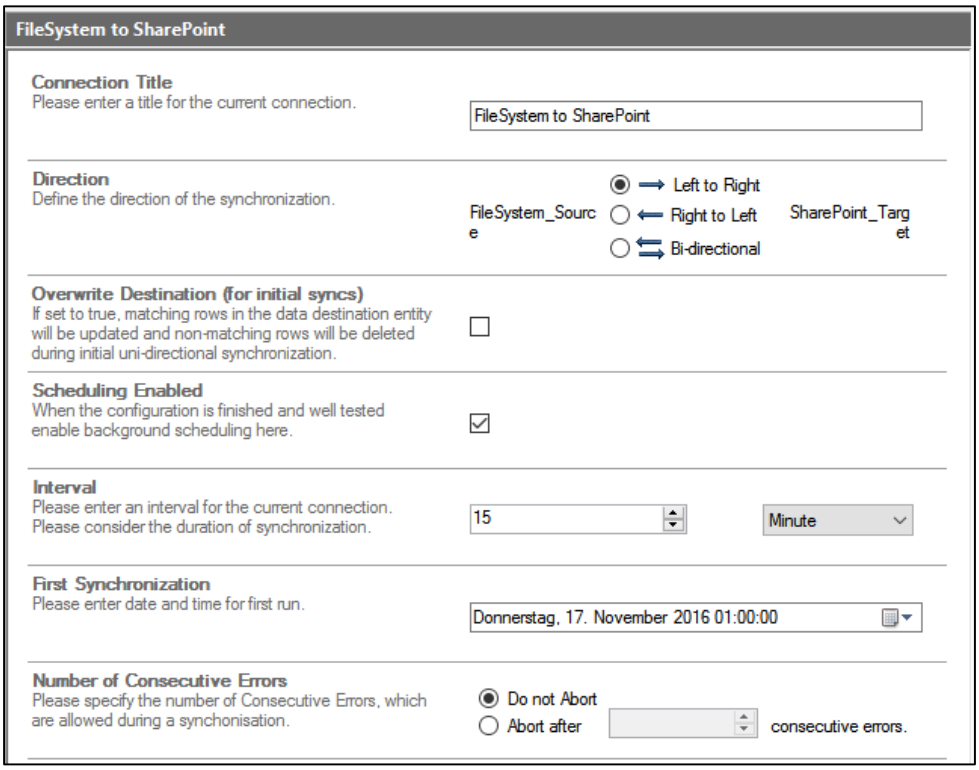


Figure 18 – Example: File System to SharePoint set to run every 15 minutes

5. Click **Save the changes** in the right-hand pane.
6. Start the "Layer2CloudConnectorService" Windows service and make sure it is **running** (the scheduled synchronization won't work without it). See the Windows Service section for additional assistance.
    a. You can start the "Layer2CloudConnectorService" from the Start menu/page option **Start Cloud Connector Service,** or from the Connection Manager itself in the right-hand Action pane.

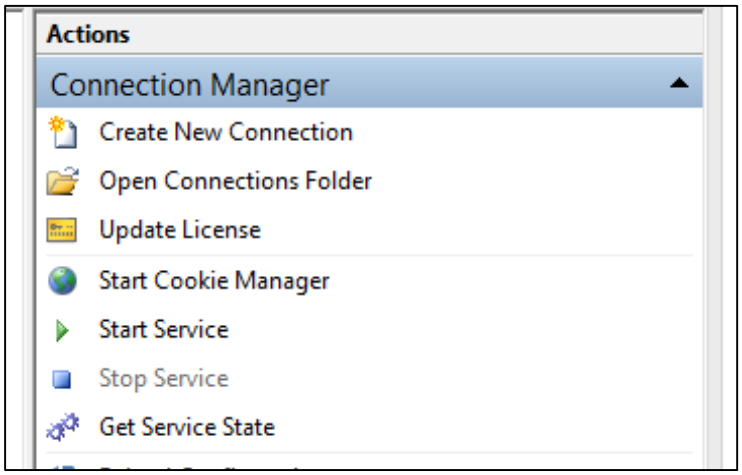


Figure 19 - Starting service in the Actions pane

b. From the root node of the Connection Manager, you can see the state of the service under the **Backend service state** property.

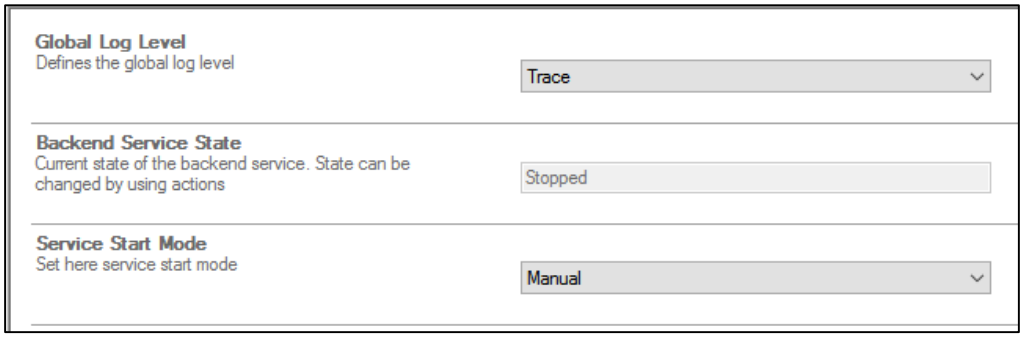


Figure 20 - Starting service from the Connection Manager Properties pane

If you have issues with getting the scheduling to run, see Troubleshooting Scheduled Sync Issues in our FAQ.

Advanced User's Guide/Technical Information

How the Layer2 Cloud Connector Works

This section will go a little deeper into the inner workings of the Layer2 Cloud Connector to provide a glance behind the scenes of the synchronization mechanism leading to a better understanding of the Layer2 Cloud Connector system overall.

The Metabase

The Layer2 Cloud Connector stores information about the synchronized data on the local disk of the machine where it is installed. This data store is called the Metabase. With the help of the Metabase the system is able to figure out which data source items have been modified since the last sync.

The Metabase is updated by the Layer2 Cloud Connector with every synchronization run. If configuration settings like the type of connection or field mappings have been changed, the Metabase is also adapted.

The Metabase is stored in the Layer2 Cloud Connector Data Directory as a binary file (see The Layer2 Cloud Connector Data Directory section). If a Metabase file is deleted, it will be re-created automatically by the Layer2 Cloud Connector during the next synchronization run, but the links between source and destination records will be lost, and it will be assumed that all records are new. That's why deleting the Metabase for a connection with two synchronous data sources can duplicate the data with the next synchronization run.

While testing synchronization connections, deleting the Metabase to allow it to be rebuilt can sometimes be a shortcut to solve issues concerning the Metabase.

However, deleting the Metabase file is sometimes necessary. For example, if the type of the ID field has changed from GUID to integer in one of the data sources, it is not possible for the Layer2 Cloud Connector to convert the IDs inside of the Metabase. In this case, deleting the Metabase will be required to clear the error.

The effect of deleting the Metabase will remove any associations between the records of the data entities. Depending on the configuration, this can lead to:

- **Duplication of the records:**
  On a bi-directional synchronization, records will be duplicated since the Cloud Connector does not know they are associated.

- **Deletion and re-adding all records:**
  This will happen on uni-directional connections, if you are using the "Overwrite Destination (for initial syncs)" option. Otherwise, the synchronization will abort with an error.

- **Updating all records:**
  There is a special case that comes into play when there is a bi-directional synchronization and the primary key of one side has been mapped to a writable field on the other side, which is usually the case with file-synchronizations. In this case, the Cloud Connector is able to determine the record-associations and rebuilds the Metabase automatically.

Auto-Backup

The Metabase is saved periodically (every hour by default) during a synchronization, so that in case of an unexpected application or service termination, already performed operations will not entirely be repeated on the next synchronization run. The interval of the auto-backup, expressed in minutes, can be specified in the registry-value:

HKEY_LOCAL_MACHINE\SOFTWARE\Layer2 GmbH\CloudConnector\AutoBackupInterval

You can disable this feature by using a negative value for the interval.

Field-Mappings and Type-Conversions

The Layer2 Cloud Connector system features a smart, tolerant conversion and comparison engine to provide field mappings between various different field types. The following table shows a digest of the more common data types and visualizes which kinds of conversions are supported and which are not. Note that for bi-directional synchronization it is necessary that conversion is supported in both directions.

Layer2 Cloud Connector Type Conversions		To				
		String	DateTime	Integer	Float	Boolean
From	String	✓	✓	✓	✓	✓
	DateTime	✓	✓	✗	✗	✗
	Integer	✓	✗	✓	✓	✓
	Float	✓	✗	✓	✓	✗
	Boolean	✓	✗	✓	✗	✓

There are two different modes to define mapping between the fields of both data sources: manual and automatic. Manual mapping is performed by choosing specific fields from both data sources and associating them as a mapped set. If automatic mapping is activated, the Layer2 Cloud Connector will consider all fields that have the same name (ignoring the case) as mapped.

Primary Keys

The Layer2 Cloud Connector requires a primary key from both data sources to identify the records. Normally, the data providers will provide a primary key, but there are cases where the primary key is not returned by the provider or where there is no primary key. If, for example, an Excel-sheet is used as a data source, there will be no primary key available. It is then necessary to manually define one or more fields as a primary key while configuring the connection.

If the data source provides a primary key, which is the most common case, it is not necessary to manually define a key. All the Layer2 data providers will return a primary key.

If the data source does not return a primary key, one of three things has happened:

- **The ADO.NET provider does not populate primary keys:**
  Some ADO.NET providers do not populate the primary keys, even though the data source has one.
- **The source has no primary key defined**:
  This case happens when the provider supports primary keys, but one is not passed to the Cloud Connector, such as when a SQL database does not have a primary key defined for the table being accessed.
- **The data source does not have a primary key at all:**
  This happens when the data source itself does not support primary keys.

In all these cases, a primary key needs to be defined in the connection configuration. Any existing field can be used as long as it is unique throughout all the records (for example an ID number, customer number, or account name). If no such field is available, it is also possible to define multiple fields as a combined primary key. The field content must be unique when combined for each item if you are using multiple fields.

Data Providers

The Layer2 Cloud Connector synchronization process goes through three distinct phases:

1. First, it connects to both data sources that are configured to be synchronized and retrieves their data.
2. Next is the synchronization phase, the Layer2 Cloud Connector inspects record by record, field by field, comparing and updating all three data sources (including the Metabase) in memory.
3. Lastly, after the synchronization is complete, the modified data will be written back to the data sources.

The data-retrieval-phase and the write-back-phase involve the data providers, which are plugins that implement a certain kind of data access. They are based on a Microsoft framework called ActiveX Data Objects (ADO). Accessing the data sources through this framework makes the Layer2 Cloud

Connector independent from how the data is actually retrieved and enables it to work with many existing ADO.NET providers.

Common ways to acquire data providers:

- Layer2 data providers are included in the distribution package and licensed with the Cloud Connector - see the Layer2 Data Providers section for the list. Some data providers are installed by default on Windows: ODBC, OLEDB, Text/CSV.
- Some are freely available directly from system or application vendors, like for Microsoft SQL Server, Oracle, or IBM iSeries.
- Some are available from 3rd-party vendors that are specialized to that business (these may need to be purchased).

See the Solutions page online to find out which data provider is necessary for a particular application.

Important – make sure the installed providers from other vendors/3rd-parties have the same bit architecture as the installed version of Cloud Connector. For example, if the 64-bit Cloud Connector is installed, the providers being used also need to be 64-bit.

Uni-directional Synchronization

Uni-directional synchronization uses one data source as the "Source" and one data source as the "Target". The data from the Source will be accessed as read-only and all detected changes to the Source will be written to the Target.

In this mode, the Layer2 Cloud Connector will perform the synchronization in the following way:

- **If two fields have different values:** the Target will get the value from the Source.
- **If a record only exists in the Source:** the record will be added to the Target.
- **If a record only exists in the Target:** the record will be deleted in the Target.

With the above process, a configuration mistake, like setting synchronization from A to B instead of from B to A, would be devastating since all the data in the Target would be deleted. Therefore, the Layer2 Cloud Connector implements following security mechanism: If a uni-directional synchronization is initially being executed and the Target contains any records, the synchronization will abort. This is the default behavior which can be disabled by selecting the **Overwrite Destination (for initial syncs)** option in the connection properties.

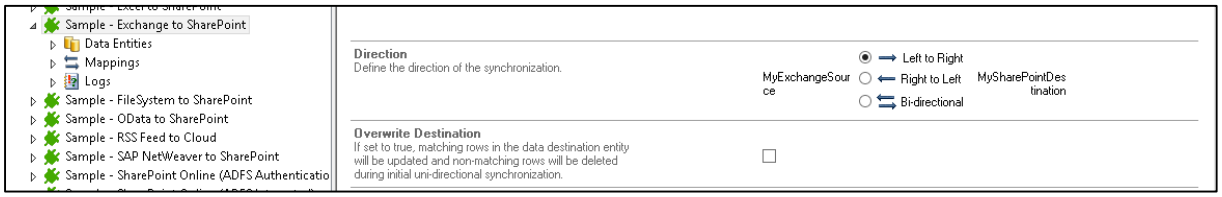


Figure 21 - Example of settings for a uni-directional synchronization

The option **Ignore Changes Within Target** can be enabled on the target data entity to speedup the uni-directional synchronization. This setting causes it not read the data from the target for the comparisons as described above but will rely on the data in the metabase. For more details, see the Ignore Changes Within Target section in the Data Entity description.

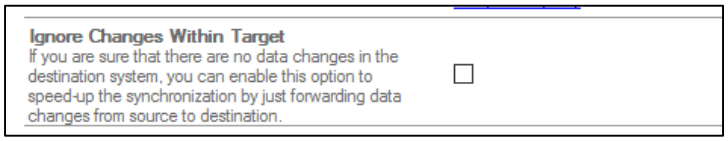


Figure 22 – The option to ignore changes within the target data source

Bi-directional Synchronization

The Bi-directional synchronization process reads from and writes to both data entities of the connection, using the Metabase to decide which side and records need to be updated.

In this mode, the synchronization will be performed in the following way:

If two fields have different values:
In this case, the Layer2 Cloud Connector checks the Metabase and determines which value has changed. If both values are changed since the last run, the Layer2 Cloud Connector detects that there is a data conflict and will use the user-defined conflict resolution strategy to determine what action should be taken. See the Conflict Resolution section for the options available.

If a record only exists in one data source:

If a record is found in the Metabase but not in another data source, it would have existed in all data sources (including the Metabase) after the last synchronization but must have been deleted in a data source between now and then. Thus the record will be deleted from the Metabase and from the other data source.
If the record cannot be found in the Metabase, it did not exist after the last synchronization and must have been added to a data source, which causes the Layer2 Cloud Connector to insert the record in the Metabase and into the other data source.

Conflict Resolution

When the Layer2 Cloud Connector encounters a conflict, meaning a field has been changed on both sides of the synchronization (Target and Source), it will check the conflict resolution strategy of the connection set by the user, which can be one of the following:

FailAbort

With this strategy, the Layer2 Cloud Connector will stop the synchronization process and report an error containing information about which fields are in conflict and their values. It would then be necessary to check the data sources directly and resolve the conflict by updating the data manually. This is the least elegant of the strategies as it halts the sync process, but it's also the safest in terms of data consistency.

This strategy is recommended for scenarios where conflicts are very unlikely but would have a big impact if they were to occur.

This conflict resolution strategy is the **default setting**.

WarnAndContinue

This strategy causes the Layer2 Cloud Connector to put out a warning into the log and continue with the synchronization process. The fields in question will not be updated, which means that the conflict remains in the system and the warning will occur with every synchronization run until it is manually resolved.

This strategy is recommended for scenarios where conflicts would have a big impact and it is applicable to have someone checking the logs regularly to manually resolve the conflicts.

WinnerLoser

With this strategy, the user defines which of the data entities is the winner (making the other one the loser) in the Mappings properties (see the Configuration section for more details on this property). When a conflict occurs, the Layer2 Cloud Connector will synchronize the winning source's value to the loser and to the Metabase.

This strategy is recommended for scenarios where one of the data entities can be defined as a master, which will always provide the determining value whenever a conflict occurs.

KeepBoth

This conflict resolution strategy is currently available for file synchronizations with the Layer2 File System Provider or the Layer2 SharePoint Provider. With this strategy, one data entity will be chosen as the "master entity". Whenever a conflict is detected, the master entity will keep its changes while the other, non-master entity will copy its changes to a new file which will have "(L2CC Mine)" appended to the file name (for example: "myFile (L2CC Mine).txt"). The changes from the master entity will then be written to the original non-master entity file (in this case, "myFile.txt").

The original non-master file will be kept up-to-date with the master entity, while the file marked with "(L2CC Mine)" will no longer be part of the synchronization process (effectively ignored). Note that the original non-master file cannot be changed as long as the "(L2CC Mine)" file exists. This will cause an error to be thrown on the next synchronization run.

To resolve the conflict, the two files on the non-master side (the original and "(L2CC Mine)") need to be merged manually into the original file and the "(L2CC Mine)" file needs to be removed. On the next synchronization run, the changes from the merge to the original file will be updated on the master entity and the conflict will be considered resolved.

Cloud Connector Components

The Layer2 Cloud Connector system consists of multiple components which are installed during setup and are described in more detail below.

The Connection Manager

The Connection Manager is a user interface for managing Layer2 Cloud Connector connections. It is integrated into the Windows operating system as a snap-in for the Microsoft Management Console (MMC) 3.0, allowing the administration of the Layer2 Cloud Connector connections by using a familiar and established interface. The Connection Manager is discussed in more detail in the Configuration section.

The Windows Service - Layer2CloudConnectorService

Layer2 Cloud Connector connections can be enabled to synchronize automatically in the background on a pre-defined schedule. This is facilitated by a Windows Service which is installed in the system as part of the Layer2 Cloud Connector: **Layer2.Data.Synchronization.Service** / **Layer2CloudConnectorService**. The service is disabled by default, and can be started using either the Connection Manager or the Windows Service Management. For additional information, see the Service Management section.

The Layer2 ADO.NET Providers

The Layer2 Cloud Connector comes with several ready-to-use ADO.NET providers to access SharePoint (2010, 2013, and SharePoint Online), the local file system, OData services, XML files, RSS feeds, Microsoft Exchange Server (on-premises or cloud-based), and SOAP web services. These providers are currently bound to be used with the Layer2 Cloud Connector or other Layer2 products. This means that they will be limited in functionality if used in absence of a Layer2 Cloud Connector license, and even with a valid license, the usage of the providers in other contexts than the Layer2 products is, while technically possible, not supported.
See the Layer2 Data Providers section for details on each included provider.

The Layer2 Cloud Connector as Console Application

In addition to using the Connection Manager or the Windows Service to start a synchronization, the Layer2 Cloud Connector provides another way to do that. The console application is available in the Start menu as **Start synchronization manually**. For client operating systems since version 8 and server operating systems since 2012, the Start synchronization manually application is available in the **Start** screen under Apps. When it is used it will then synchronize all enabled connections or specified one, and put out real-time messages about its activity. The console application is discussed in greater detail in the Console Mode section.

The Cookie Manager

The Cookie Manager offers an additional option to authenticate to Microsoft Office 365, CRM Online, or SharePoint FBA. It uses the technique of adding federation cookies to the target server connection request object. The Cookie Manager can be started from Connection Manager by clicking **Start Cookie Manager** in the right-hand pane. This is a deprecated feature and is normally not required to authenticate with supported systems.

The Layer2 Cloud Connector Data Directory

The Layer2 Cloud Connector data directory is where the system stores all files that it needs to function properly except for the binary files (libraries and executable files). This includes connection definitions, Metabase-files, logs, and the license file.

It is located in the Windows application data directory:

C:\ProgramData\Layer2 Cloud Connector

The path can be read from the environment variable **%PROGRAMDATA%**.

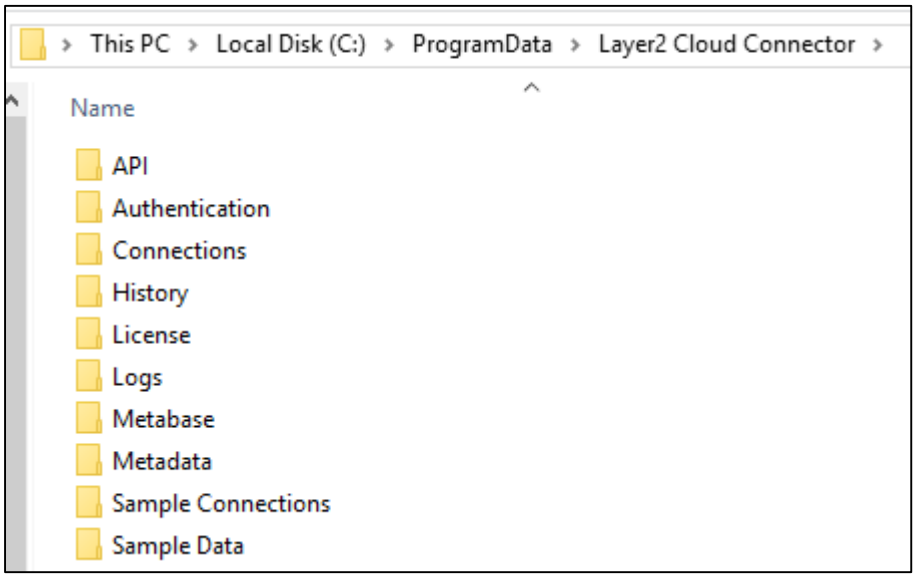


Figure 23 - Items within in the Data Directory

API

This folder contains source code of a sample ADO provider for RSS feeds to help you to create your own providers.

Authentication

This is used to store definitions for custom authentication methods. See the Authentication section for more information.

Connections

The connection definitions are stored in this folder as XML files. They're named after the connection, so for example, a definition for a connection named MyConnection will be saved in a file named "MyConnection.xml". The XML format itself is be described in detail in the Connection Definition Files section.

History

The folder contains information about the last synchronizations stored as XML files for each connection. It is used for reporting in the Connection Manager.

License

The Layer2 Cloud Connector will check this directory for a file named "productkey.xml", which will be used to license the product. If no such file is found, the Layer2 Cloud Connector will run in shareware mode. See the Licensing section for details.

Logs

This folder contains the log files that will be written during synchronization. Each connection has its own log file which is named after the connection plus the .log extension. There are two additional files: system.log file and NLog.config. System.log contains all activities of the system regardless of which connection is executed. NLog.config contains the logging configuration and is discussed further in the Logging and Alerting section.

Metabase

This folder contains all the Metabase files of the connection. Each file is named after the connection that it is storing the Metabase for. So if there is a connection named MyConnection there will be a corresponding Metabase file called “MyConnection.metabase”. The contents of these files are in binary format and it is not recommended to change them. See the Metabase section for more details.

Metadata

This folder contains metadata that is cached by the provider **Layer2 Data Provider for OData (Deprecated)**.

Sample Connections

This folder contains a backup copy of the sample connections.

Sample Data

This folder contains an XML file with sample data used in Appendix A – Examples.

Configuration

The configuration and administration of synchronization jobs for the Layer2 Cloud Connector is primarily done by using the Connection Manager. It is a snap-in for the Microsoft Management Console (MMC) and can be started by using the shortcut “Start Connection Manager” in the start menu or by adding a new snap-in to an existing MMC console configuration.

The general interface layout of the Microsoft Management Console consists of three panes:

The left-hand Connection pane provides the navigation through all the connections. For every connection there is a node which contains sub-nodes for data entities, mapping, and logs.

The middle Properties pane is the main area and contains the configuration properties which change depending on which node has been selected in the navigation pane.

The right-hand Action pane lists all the actions that are available in the context of the currently selected navigation node. These actions are also available through the Action menu at the top of the window and in the right-click context menu of the currently selected navigation node.

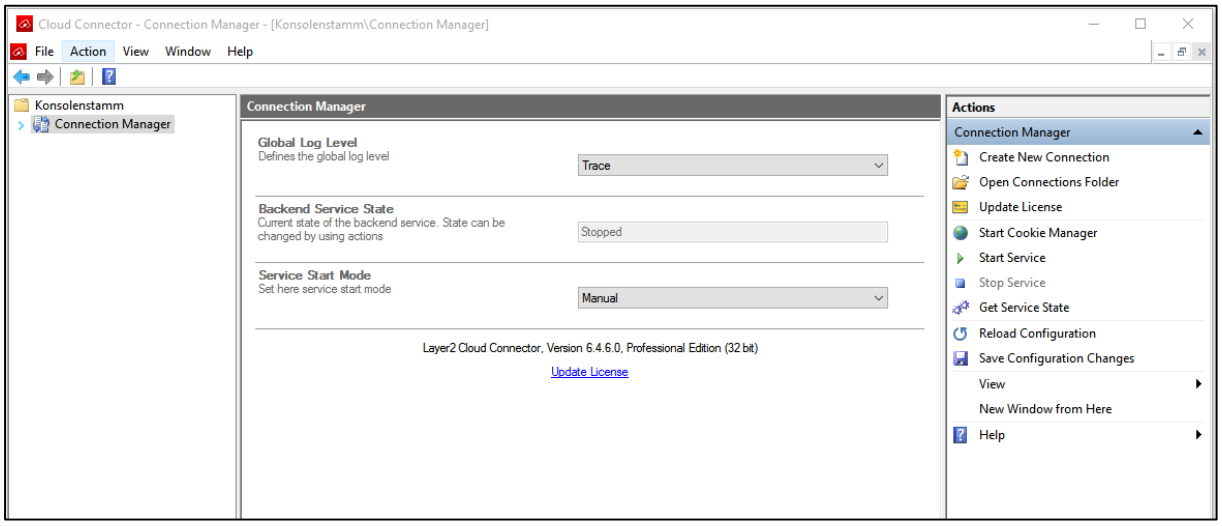


Figure 24 - View of Connection Manager properties

Global Settings

The root node is labeled **Connection Manager**. Activating this node in the navigation pane will show settings that are not specific to a connection and therefore have impact on the system as a whole.

Global Log Level

The setting defines the log threshold for the activity logging of all connections. The value configured here is the minimum severity that a log message must have to come through to the logging system. After changing the setting, changes must be saved in order to take effect, and the Layer2 Cloud Connector Service must be restarted to use new setting. There are six different severities, from most detailed to least detailed:

Trace

On this level, the most detailed log will be created, but this will also have the greatest impact on the performance of the synchronization. Messages, for example, informing about which values are compared and why they are considered equal or not will be put out to the log.

Debug

This level will not record the more detailed, technical information but it will still be verbose about what the Layer2 Cloud Connector is doing during synchronization. For example, messages about which records are modified will be written to the log.

Info

This level will restrict the log to only put out the key data about the synchronization process, such as start of the synchronization, connection attempts to the data sources, and information about how many changes have been made.

Warn

[Default] On this level there will be warnings about suspicious system conditions.

Error

All messages on this level refer to a serious problem during the synchronization and will usually cause the synchronization to stop.

Fatal

Messages on this level will be generated when unexpected errors occur, such as hardware-originated issues. These will cause the synchronization to stop.

Setting the global log level to a specific value will always include all severities below that. If, for example, the log level is set to Info, all messages with the severities Info, Warning, Error, and Fatal will get through to the log.

When the Layer2 Cloud Connector system is installed, the global log level is by default set to “Warn”. This will provide detailed information about the synchronization process while the connection is being configured and tested. It is recommended to set the log level to “Info” once the connection has gone into production use to reduce the performance impact of logging.

Backend Service State

This setting shows the current state of the Layer2 Cloud Connector Service. If there are connections scheduled for automatic background synchronization, these will only be synchronized periodically if the service is in the “Running” state. When the Layer2 Cloud Connector system is installed, the service is not started by default and the starting type is “Manual”. Background synchronization can be enabled by starting the Windows service using the **Start Service** action, which is available from the right-hand Action pane.

Service Start Mode

This setting defines what the start mode of the background service is. The options are “Automatic”, “Manual” (default), and “Disabled”. To make sure that the service will be started automatically at operating system startup, the option “Automatic” needs to be chosen in the **Service start mode** option box. Changes must be saved after making changes on global configuration section by clicking **Save Configuration Changes** in the right-hand Action pane.

Actions

Following actions are available while the **Connection Manager** node is selected:

Create New Connection

This creates a new, empty Connection node in the left-hand pane.

Open Connections Folder

Opens a Windows Explorer window for the connection configuration XML files. A quick way to access the configuration files in case a manual update needs to be done. See The Layer2 Cloud Connector Data Directory section for more details.

Update License

Opens a Windows Explorer window to locate a productkey.xml file to update the current license state. This option is also available from the central pane of the global settings. See the Licensing section for more details.

Start Cookie Manager

Starts the Cookie Manager application. See the Cookie Manager section for more details.

Start Service

Starts the back-end synchronization service. See the Service Management section for more details.

Stop Service

Stops the back-end synchronization service.

Get Service State

See Backend Service State above.

Reload Configuration

All connection configurations are saved in the data directory in XML files (see The Layer2 Cloud Connector Data Directory section for more details). If these files are changed outside of the Connection Manager a reload is required to get the changes into the Connection Manager. To reread all connections, there is an action called **Reload configuration**, which is, like all actions, available from the action-pane, the node-context-menu and the main-actions-menu.

Save Configuration Changes

Saves all changes made to the global settings.

Connection Definition

Below the root node there is one sub-node for every connection. Layer2 Cloud Connector ships with a set of sample connections as examples to help the user get started in creating their own connections. These sample connections are disabled for background synchronization.

For the selected connection, the Connection Manager will display general configuration options in the central panel.

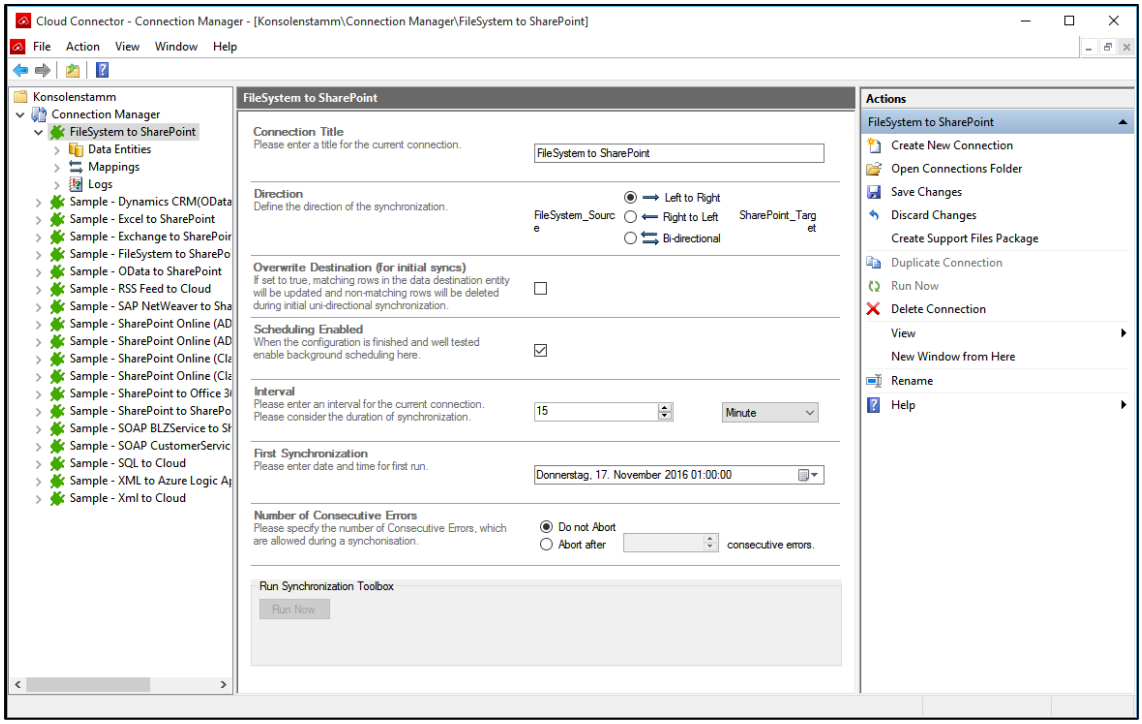


Figure 25 - View of Connection properties

Connection Title

This setting is for specifying a title for the connection. Each connection must have a unique name.

Direction

The setting defines the direction in which the synchronization will be processed. Uni-directional synchronizations can be defined as **Left to Right** or **Right to Left**, depending on which data entity should be the source and which the target. Bi-directional synchronizations will synchronize changes from both data entities to each other.

Overwrite Destination (for initial syncs)

The setting allows for cleanup of the destination data entity for the first synchronization, if the destination entity contains some pre-existing records. The setting is available only for uni-directional synchronizations.

Scheduling Enabled

If this option is enabled, the connection will be included in the automatic background synchronization. The background synchronization will synchronize the connection without needing to start the Connection Manager. This requires that the Layer2 Cloud Connector Service is running.

Interval

If background synchronization is enabled for the connection, this interval will define how often a synchronization run will be executed. This can range from a few minutes to multiple days.

First Synchronization

This defines the exact date and time when the automatic background synchronization for this connection will run for the first time (or, if it has been running before, for the next time). This is based on the local machine's time.

For example, if the connection should be configured to run every night, **Interval** would be set to 1 day whereas the next run will be set to 12:00 am. This way the synchronization will be started every night at 12:00 am.

Number of Consecutive Errors

This defines the number of errors that occur consecutively, before the synchronization is aborted. Options are **Do Not Abort** and **Abort After** which includes user defined field.

Last Synchronization

If a connection has already been executed at least once, a summary of the last synchronization run will be displayed just below the **Number of Consecutive Errors** section. Synchronization runs performed by the background service it will be displayed here as well.

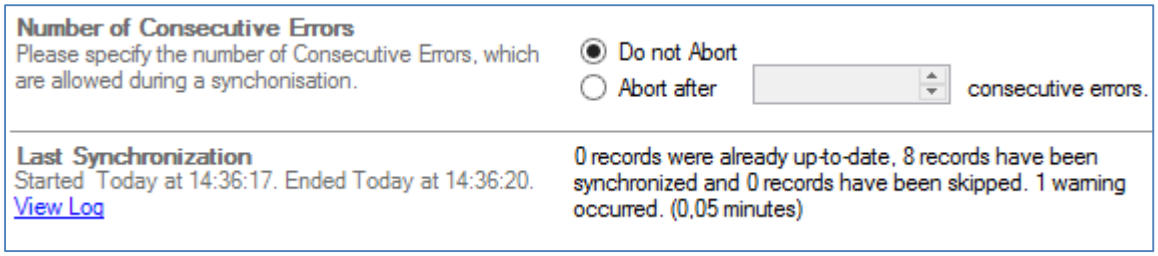


Figure 26 – Example of a Last Synchronization summary

Run Synchronization Toolbox (Run Now)

This action starts the synchronization of the currently selected connection. It can be used to test the connection or to perform one time replications, such as migration tasks. This tool box window also shows the current state of the sync, like total items updated, as well as fatal errors.

Actions

Create New Connection

This action creates a new empty connection under the root node.

Delete Connection

This will delete the currently selected connection entirely. All associated data (connection file, history, logs) will be deleted.

Save the Changes

As soon as a setting of the connection is changed, it will be active to allow saving the changes.

Discard Changes

Note, if there are unsaved changes on the current connection and another node is selected in the navigation pane, the Connection Manager will show a dialog demanding to save or discard the changes.

Create Support Files Package

This will ask for a location and create a ZIP archive with all support-related files (connection file, logs, and log configuration) associated with the current connection. Sensitive information (username/password) is automatically masked.

Duplicate Connection

With this action, the connection can be duplicated to use it as a template for a new connection.

Run Now

This action starts the synchronization of the currently selected connection. It can be used to test the connection or to perform one time replications, such as migration tasks.

Delete the Connection

This will delete the currently selected connection entirely. All associated data (connection file, history, logs) will be deleted.

Rename

With this action, the connection will be renamed as will with all related files like log, history, etc.

Data Entity

Both data sources that are participating in the synchronization, as source or target, will be referred to as a data entity. Every connection contains two sub-nodes for these data entities, which can be configured to access various data sources.

Data Entity Title

This is the name of the data entity used to identify the entity in the navigation pane and in log messages.

Entity Type

This is showing the role that this entity owns in the synchronization process and can be **Target**, **Source**, or **Bi-directional**. This value cannot directly be changed in the data entity screen and is defined by specifying a value for the synchronization direction on the connection screen.

Data Provider

Provides a selection of the ADO.NET providers which will be used to retrieve data and write changes for the current data entity. All installed providers that are compatible with the current processor architecture will be shown in the drop-down.

To see the pre-installed providers, go to the Layer2 Data Providers section. For additional 3rd-party data providers, one can usually find downloads for these on the 3rd-party's website. They usually come as an executable setup file which can be run and the provider (if it is of the correct architecture) will appear in the list of options here. If you have the Layer2 Cloud Connector open while installing a new provider, you will have to restart it before you can see the new provider in the list.

Connection String

This is the ADO connection string for the selected provider. These connection strings consist of key-value pairs separated by semicolons.

Example:

```
Url=http://MySharePointServer/MySharePointSite/; List=Links;
Authentication=Windows; User Id=MyDomain\MyUserName;
```

Connection strings are highly specific to the provider that is used. There are many examples and documentation at www.connectionstrings.com and on the Layer2 solutions page. The **Encrypt** option can be used to hide security relevant information in the connection files.

Note: In pre-6.4.6.0 versions of the Cloud Connector, the "Password=;" parameter had to be defined in Connection String field. For current versions, it is now stored in its own Password field. See Password below for more information.

For some providers, there is an action/link that starts the **ADO Connection Wizard**. See the ADO Connection Wizard action reference below for more details.

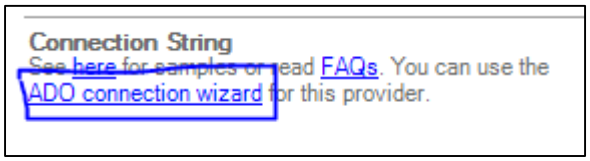


Figure 27 – ADO connection wizard highlighted in the connection string details

There is a **Verify Connection String** link below the connection string text box that is useful to quickly evaluate if the provided connection string is valid. This has the same behavior as the Verify Connection String action listed below.

Figure 28 – "Verify Connection String" action highlighted in the connection string details

Password

Here you can enter the password connection string parameter, if required by the data provider. This field masks the value for better security.

Figure 29 – Password field with masked value

Note: If using a pre-6.4.6.0 versions of the Cloud Connector, the "Password=;" parameter has to be defined in Connection String field.

You can still use the connection string to enter your password to prevent typos and specify the password parameter name (Pwd, Pass, Password, etc.), but the next time the Cloud Connector is started, the password is automatically moved into the masked password field.

Note: If both fields contain a password and they do not match, an error will be shown.

Select Statement

In this text box a data query statement can be defined. The format to be used for the query is specific to the provider selected (see the specific provider documentation to find out what is required) and is not necessary for all providers. If the provider is known the by Connection Manager to not support a

select statement, the select statement setting will not be available. The **Encrypt** option can be used to hide security relevant information in the connection files.

The select statement can be validated similar to the connection string with the **Verify Select Statement** link. This has the same behavior as the Verify Select Statement action listed below.

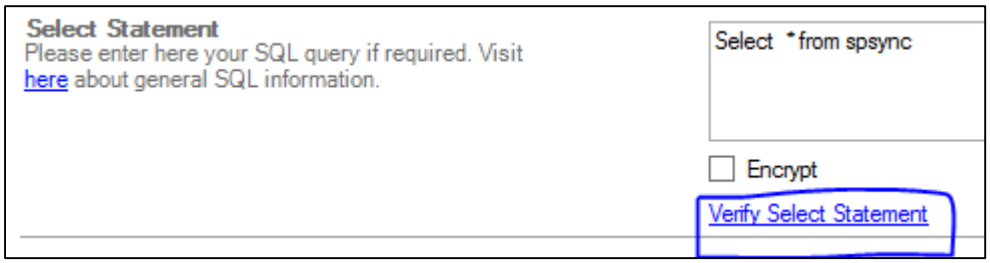


Figure 30 - "Verify Select Statement" action highlighted in the select statement details

Primary Key(s)

This setting is optional. If the data source provides a primary key, there is no need to define one explicitly. If this is not the case, the key can be defined here. Any field that is provided by the data source can be defined as a primary key. Furthermore, multiple fields defined, separated by a comma, to define a composite primary key. The **Encrypt** option can be used to hide security relevant information in the connection files.

The primary key can be validated similar to the connection string with the **Verify Primary Key** link. This has the same behavior as the Verify Primary Key action listed below.

Figure 31 - "Verify Primary Key" action highlighted in the primary key details

Ignore Changes Within Target

This setting is only available for the target data entity in a uni-directional synchronization and is optional. If enabled, it speeds up the uni-directional synchronization in that the Cloud Connector does not read the data from the target for the data comparisons but will rely on the data in the Metabase. So for example, a field value is changed in the target only, it will not be overwritten with the source value (as it normally would), because this change is not recognized. Therefore, use this option with care and only if you are sure that there are no changes made at all in the target system.

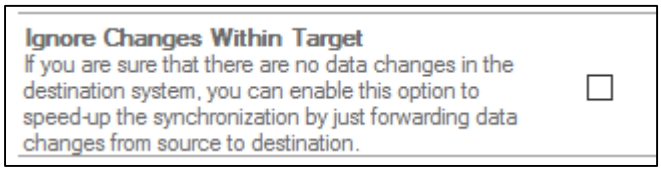


Figure 32 – Option to Ignore Changes Within Target

Replication Key

This is an optional parameter in the **Advanced Settings** section. In some cases, using a global unique identifier (GUID) can help to solve replication issues. This setting uses a field in the data that would be the primary key and the Cloud Connector will auto-generate a GUID into it. The **Encrypt** option can be used to hide security relevant information in the connection files.

The replication key can be validated similar to the primary key string with the **Verify Replication Key** link.

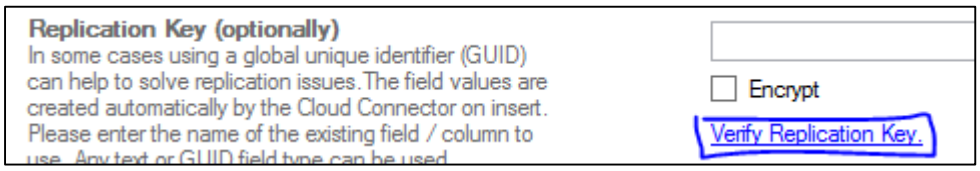


Figure 33 - "Verify Replication Key" action highlighted in the replication key details

Disable Operations

This setting in the **Advanced Settings** section allows users to define which operations/transactions are omitted during execution of a synchronization. For example, when **Disable Delete** is selected, all delete requests during synchronization will be omitted. Possible values are Insert, Update and Delete. Individual or multiple selections is possible. The setting is optional and not always available.

Dynamic Columns

This setting in the **Advanced Settings** section allows users to create new custom fields, which are added to the fields from the data source. For complete details on Dynamic Columns, see the Dynamic Columns section.

Please note that this feature is only available with the .NET 4 version of the Layer2 Cloud Connector.

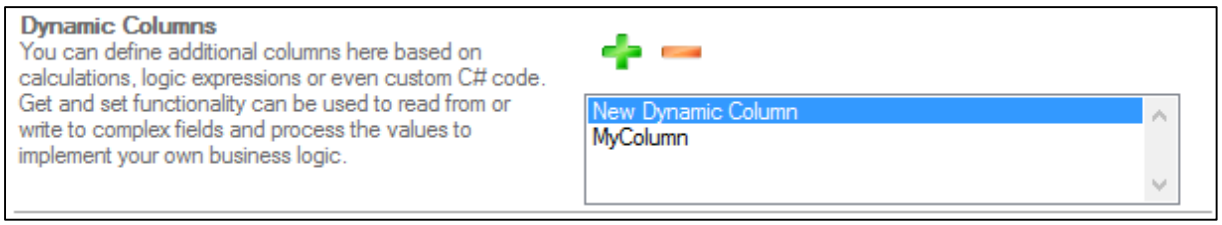


Figure 34 - "Dynamic Columns" list in the data entity settings

Actions

Create New Connection

This action creates a new empty connection under the root node.

Open Connections Folder

Opens a Windows Explorer window for the connection configuration XML files. A quick way to access the configuration files in case a manual update needs to be done. See The Layer2 Cloud Connector Data Directory.

Save the Changes

As soon as a setting of the data entity is changed, it will be active to allow saving the changes.

Discard Changes

Undo last changes, and reset data entity to last saved state.

Note, if there are unsaved changes on the current data entity and another node is selected in the navigation pane, the Connection Manager will show a dialog demanding to save or discard the changes.

Preview Data

This action is used when the data entity has been set up and there is a need to check if all the necessary fields will come back as expected. It will show a simple view of all the columns it will pull and data for the first ten records which are delivered by the entity. This action is also helpful to identify a primary key, if required.

ADO Connection Wizard

For some providers, there is an action that starts the **ADO Configuration Wizard**. This wizard leads to the creation of a connection string by showing available data sources to choose from. The selected provider must support this wizard to provide useful guidance.

Verify Connection String

The Connection Manager will try to create a connection using the specified connection string, and will show any errors that occur while doing that. If no errors occur, it will show a green "Verified" message.

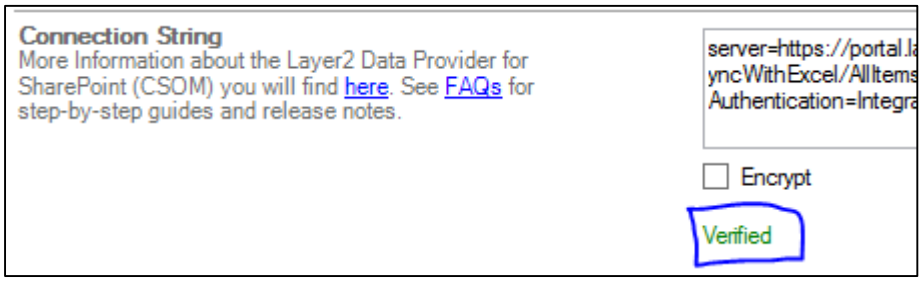


Figure 35 - Example of verifying a connection string

Verify Select Statement

Like the **Verify Connection String** action, the Connection Manager will try to create a connection using the specified connection string and select statement, and shows any errors that occur while doing that. If no errors occur, it will show a green "Verified" message.

Verify Primary Key

Like the **Verify Connection String** action, the Connection Manager will try to create a connection using the specified connection string and primary key, and shows any errors that occur while doing that. If no errors occur, it will show a green "Verified" message.

Rename

With this action, the data entity will be renamed.

Mapping

The mapping defines which fields will be compared during the synchronization. It is based on field names. In case the data entity provides different values for an internal field name and the related display name (for example, in a SharePoint list), both are shown to ease identification of the correct fields to be mapped. The internal name is shown first, with the display name shown in single quotes after it.

Enable Auto Mapping

By checking **Enable Auto Mapping**, fields with the same name will be mapped to each other. Fields of the first data entity are shown on the left side, fields of the second data entity are shown on the right side.

You can also do a manual mapping. For manual mappings, additional entries can be added by clicking the **green plus** button and existing ones can be deleted by clicking **red minus** button. If any of the data entities are not valid (e.g. invalid connection string), the mapping screen will show the error message which occurred during the connection attempt.

Conflict Resolution

In a bi-directional synchronization it is sometimes possible to have changed items in both entities. In this case there is a conflict which must be resolved during synchronization. See the Conflict Resolution section for more details.

Verify Mapping

Using the **Verify Mapping** action or link will have compared fields checked to make sure their types match and that there are no other issues (such as read-only fields that might be written to). If no errors occur, it will show a green "Mapping verified" message.

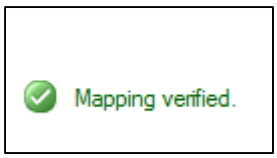


Figure 36 - Confirmation shown for verified mapping

Reload Mapping

Using the **Reload Mapping** action or link with reload all the available fields from the data entities as well as the names/types of currently mapped items.

Run Now

This action starts the synchronization of the currently selected connection. It can be used to test the connection or to perform one time replications, such as migration tasks.

Log

Shows logged messages for the current connection. You can double-click on any entry in the log to see the full details in a dialog box. Note that the Log UI shows the items from, newest to oldest, with the newest at the top of the pane. Also be aware that the Log UI does not show all logged items from a sync, nor some detailed information (such as stack traces). You will need to get the log file which will have the full record, which can be done via the Open Log File action mentioned below.

By default, the logging level is set to "Warn", which generally only shows warnings and fatal errors. If you want to increase the detail of the logging, please see the Global Log Level section for how to adjust that. For more information about logging and configuring other types of alerts (like email), see the Logging and Alerting section.

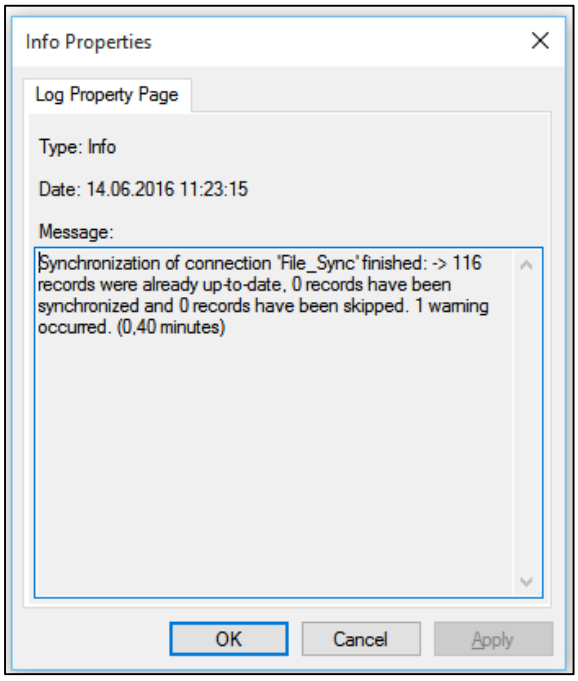


Figure 37 - Example of full text for a log entry

Open Log File

Opens the full log file for the current connection in the default program for files with the ".log"-extension.

Open Logs Folder

Opens a Windows Explorer window for the Logs folder. A quick way to access all log files for a connection (if some have been set as archived). See The Layer2 Cloud Connector Data Directory for more information.

Delete Logs

Deletes the log file for the current connection.

View

Filter the logs by selecting **View** in the right action pane and choose the required filter option.

Refresh

Refreshes the data in the logs UI to the most current information from the log file.

Export List…

Allows you to export the current information in the logs UI to a .txt file. Note that this is **not** an export of the full log file, but of the limited data shown in the Logs central pane. Use **Open Log File** to get the full dataset.

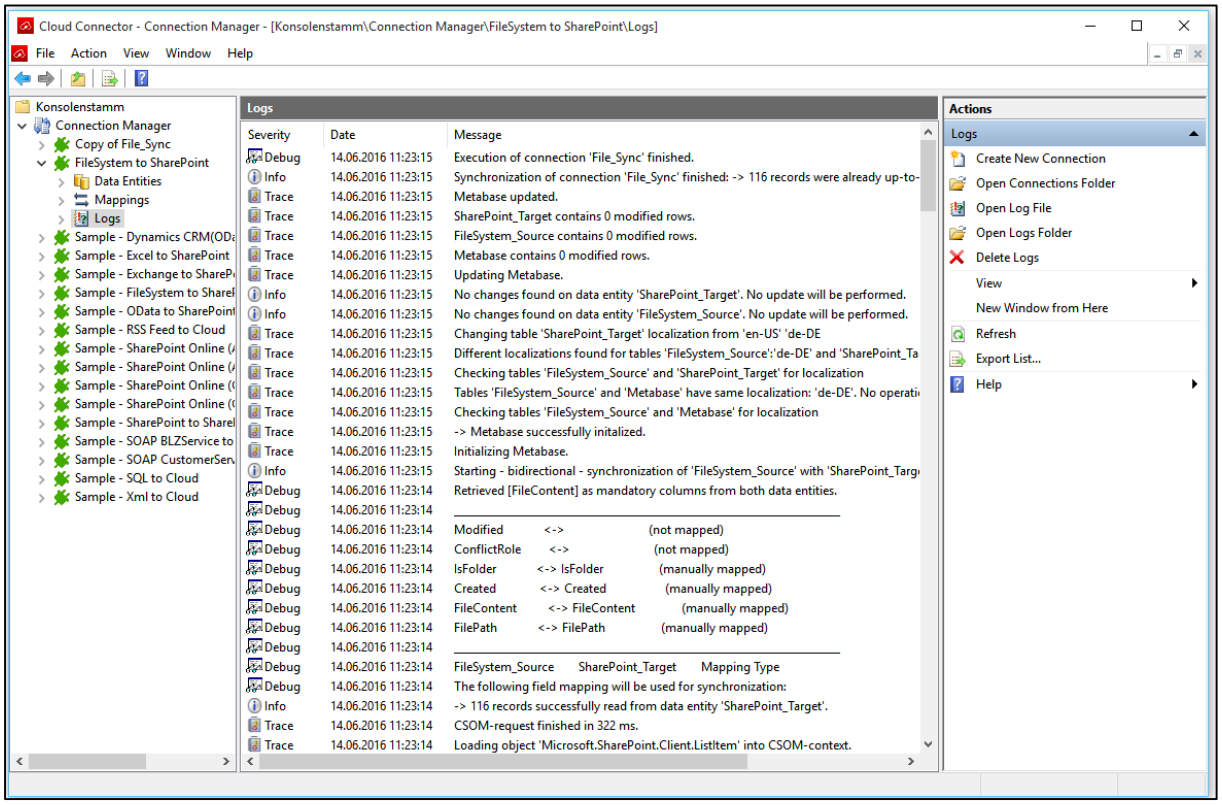


Figure 38 - Example of Logs properties

Dynamic Columns

This feature allows users to define new data columns in addition to the ones that are returned by the data entity. The values of these columns are calculated at run-time, and can be based on the content of the other columns originating from the data entity. This allows for customized conversions, translations, and formatting of the data to be synchronized.

Note: This feature is only available in the .NET 4+ version of the Cloud Connector. It is not available in .NET 3.5.

The definition of Dynamic Columns is based on the programming language C# in version 6. Either a valid C# expression or a valid method-body returning a value on all possible code-paths are acceptable. The code used includes common parts of the .NET Class-Library, but currently does not allow loading additional assemblies or namespaces. It is, however, possible to load additional assemblies through reflection.

Clicking the red '+' to create a new Dynamic Column or double-clicking an existing one will open an editor for the column, where the name and code of the dynamic column can be edited.

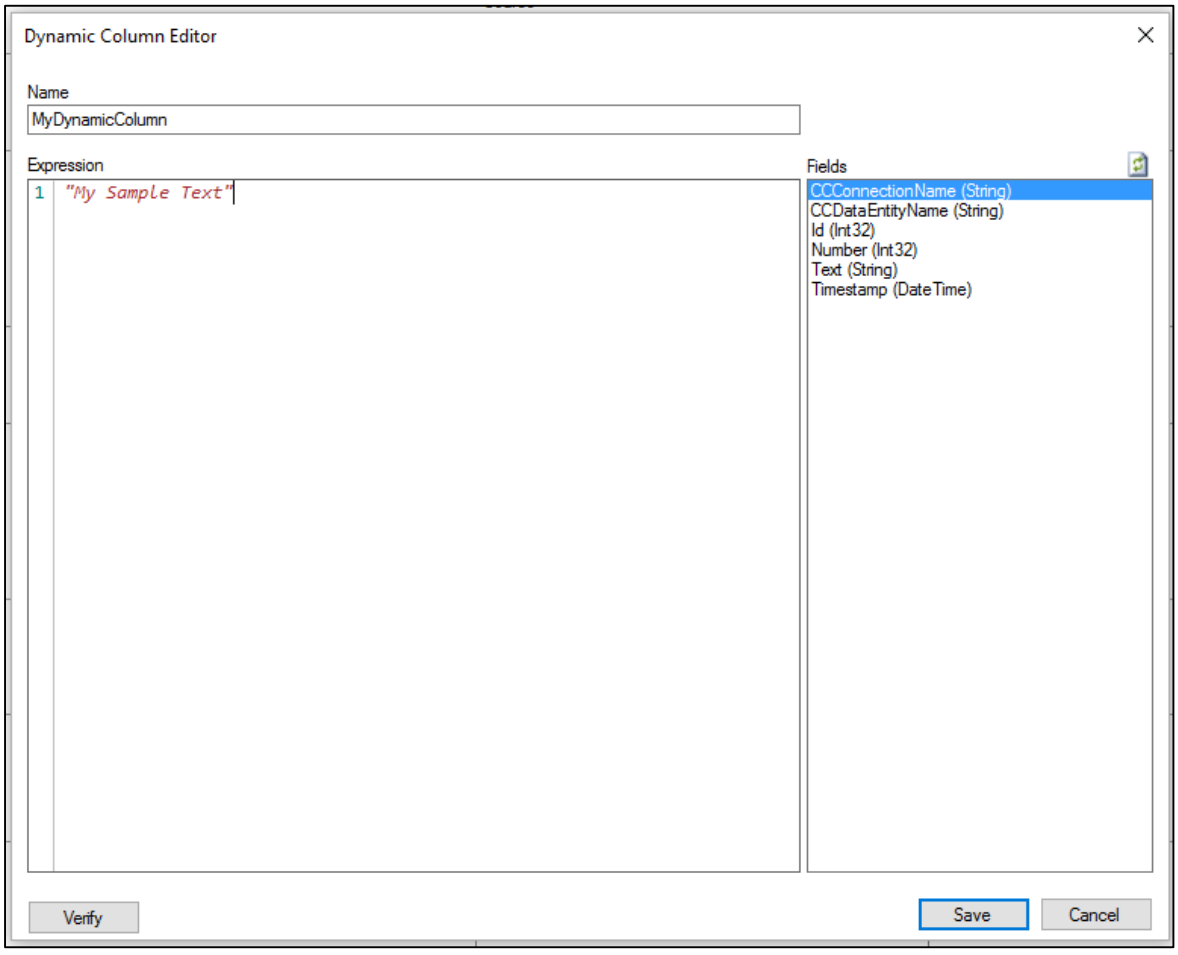


Figure 39 – The Dynamic Columns editor

A list of fields provided by the data entity is displayed on the right that can be used to add references to the fields in the dynamic column code by either drag-and-drop or double-clicking the entry. In case fields have been changed while the dialog was open, the list can be refreshed by clicking the refresh icon above it.

As long as the code is an expression that can be resolved into a value, no return statement is necessary. The data type of the column is automatically detected by the returned values. All returns need to have the same data type or the verification will report an error.

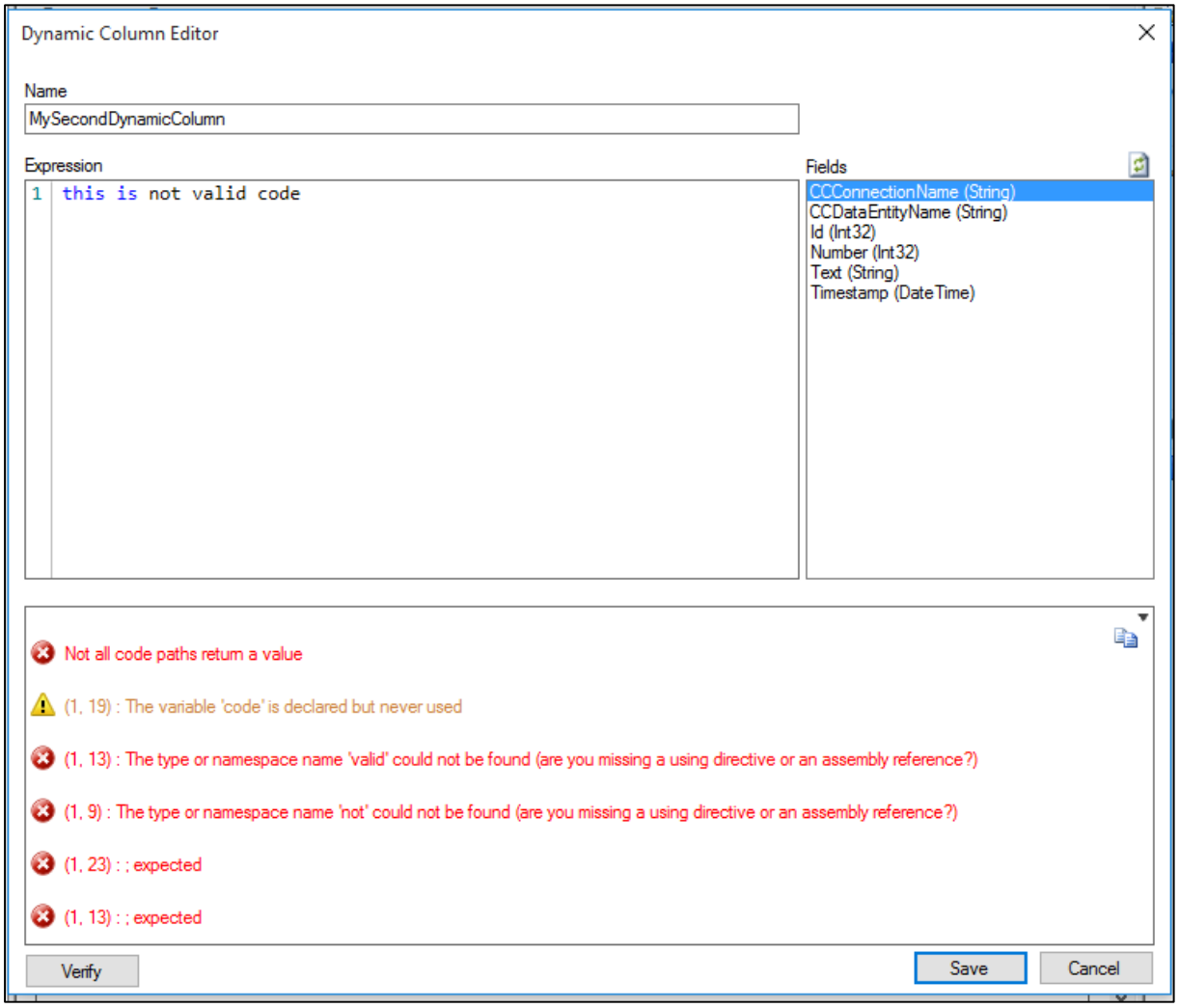


Figure 40 – A failed verification due to invalid code

Red circles with a white X are errors that need to be fixed before the Dynamic Column can be used. Yellow triangles with a black exclamation mark are warnings which do not necessarily have to be fixed, but should not be ignored in most cases as they are usually note an oversight or mistake.

With the copy symbol at the upper right of the verification window, you can copy the error messages and some additional information to the clipboard.

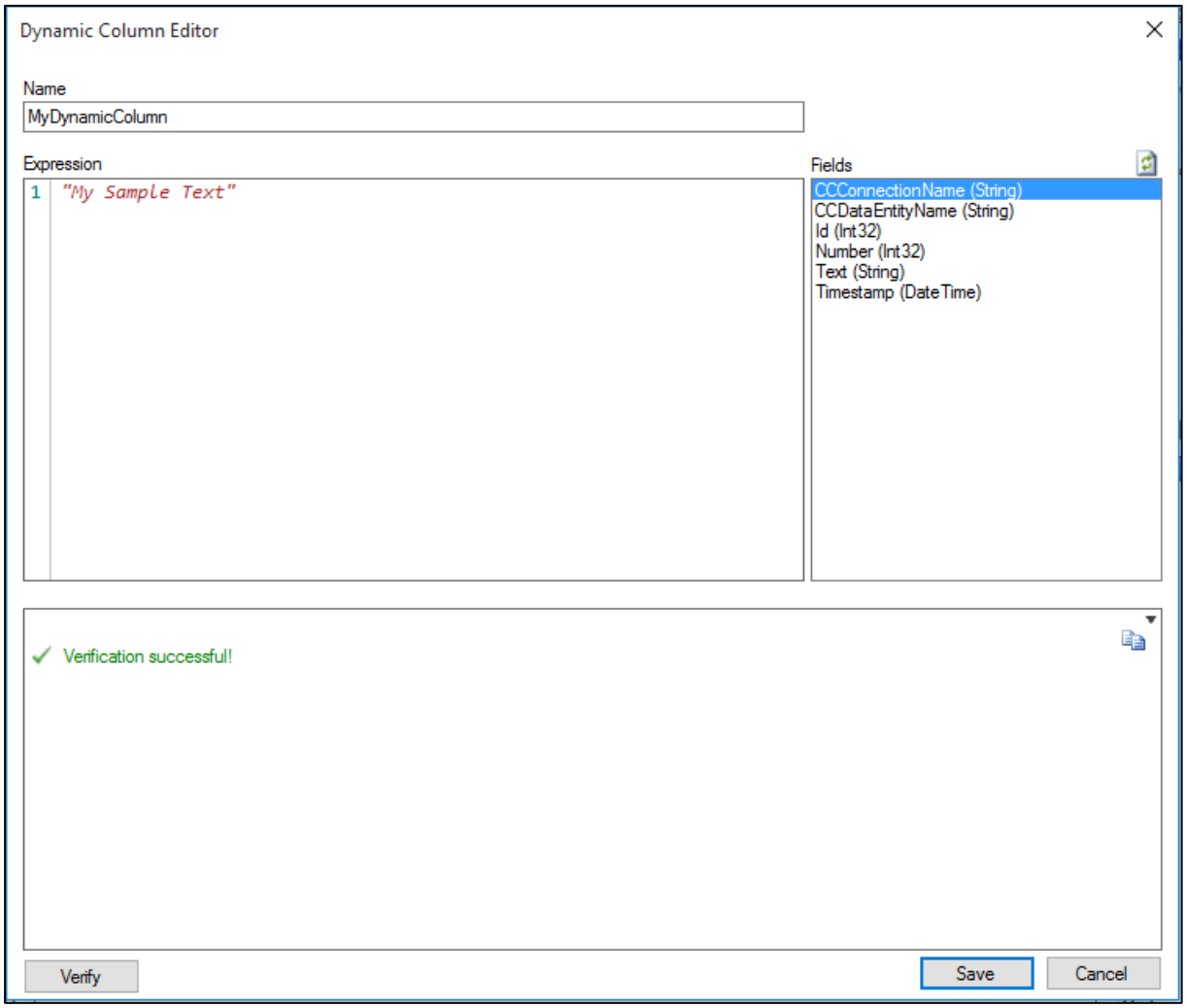


Figure 41 – The successful verification of a very simple Dynamic Column

Note: If you change the data type of the Dynamic Column after it has been mapped in the **Mappings** node, make sure that you reload and verify the mapping. This helps to avoid errors due to mismatching column types in the mapping.

Code Examples for Dynamic Columns

This section describes various possible uses for Dynamic Columns with example code.

Conditional (True or False)

This column will return `true` or `false` as boolean values based on a user-defined condition, such as "The field 'FilePath' starts with '/myFolder'".

Note: Please be careful with StartsWith, as in this case, files under "/myFolder2" would also be true.

Fields	`FilePath = "/myFolder/myFile.txt"`
Code	`FilePath.StartsWith("/myFolder")`
Result	`true : bool`

Field Combination

This column will return a combination of other fields based on the user-defined format. The data type will be `string`.

Fields	`MyFirstField  = "Test1"`
	`MySecondField = 125`
	`MyThirdField  = false`

Code	`string.Format("First: {0}, Second {1}, Third: {2}", MyFirstField, MySecondField, MyThirdField)`
Result	`"First: Test1, Second 125, Third: false" : string`

Current Date and Time

This example shows how to provide the target system with the current date and time of the host system.

Fields	-
Code	`DateTime.Now`
Result	`(Current Date and Time) : DateTime`

Field Transformation

This example shows a simple number conversion for a target system that cannot handle the source system value without adjustments.

Fields	`YearlyRevenue = 1500000`
Code	`YearlyRevenue / 12`
Result	`125000 : float`

Conditional Field Selection

In this example, the source system contains products with a regular price, a discount price, and a flag that defines if the product should use the discount. The target system should only receive a price and no information about discounts.

Fields	`RegularPrice  = 20.00`
	`DiscountPrice = 16.99`
	`UseDiscount   = true`

Code	`if (UseDiscount)`
	`{`
	`  return DiscountPrice;`
	`}`
	`return RegularPrice;`

Result	`16.99 : float`

Note: If you use a string field to hold boolean values (in SQL for example), you need to parse the string value into a boolean variable first, like:

```
bool flag;
bool.TryParse(UseDiscount, out flag);
```

Then use the flag in the if-statement instead of UseDiscount.

Lookup Translation

In the case of a migration sync from one SharePoint to another, a source list with a lookup column has to be sent to the new list in the new SharePoint. There is a similar lookup set up regarding the labels, but the IDs do not match those of the old SharePoint, leading to data issues. With a simple dynamic column that only provides and sets the lookup label without the ID, this example can solve that problem. The dynamic column has to be mapped to the Lookup column of the other data entity.

Note: This sample does not work if your lookup values contain the substring ";#" as this is a special character combination that SharePoint uses internally for the lookup field representation.

Fields	`LookupColumn  = "3;#Value"`

Code	`get`
	`{`
	`  return LookupColumn.Split(new[] { ";#" },`
	`  StringSplitOptions.RemoveEmptyEntries).Last();`
	`}`
	
	`set`
	`{`
	`  LookupColumn = value.Split(new[] { ";#" },`
	`  StringSplitOptions.RemoveEmptyEntries).Last();`
	`}`

Get-Result	`"Value" : string`
Set-Result	`LookupColumn set to "Value"`

Multi-Lookup Translation

In the case of a migration sync from one SharePoint to another, a source list with a lookup column that allows multiple values has to be sent to the new list in the new SharePoint. There is a similar lookup set up regarding the labels, but the IDs do not match those of the old SharePoint, leading to data issues. With a simple dynamic column that only provides and sets the lookup labels without the ID, this example can solve that problem. The dynamic column has to be mapped to the Multi-Lookup column of the other data entity.

Note: This sample does not work if your lookup values contain the substring ";#" or "],".

Fields	MyMultiLookup = "[3;#Value1],[4;#Value2]"

Code

```
get
{
  if (string.IsNullOrEmpty(MyMultiLookup))
  {
    return string.Empty;
  }
  // append a comma so that the split works correctly
  var multiLookupValue = MyMultiLookup + ",";
  var valueArray = multiLookupValue.Split(new [] {"],"},
                      StringSplitOptions.RemoveEmptyEntries);
  if (valueArray.Length == 0)
  {
    return string.Empty;
  }

  List<string> results = new List<string>();
  foreach (var value in valueArray)
  {
    var lookupLabel = value.Split(new[] {";#"},
                StringSplitOptions.RemoveEmptyEntries).Last();
    results.Add(string.Format("[{0}]", lookupLabel));
  }

  return string.Join(",", results);
}

set
{
  if (string.IsNullOrEmpty(value))
  {
    MyMultiLookup = string.Empty;
  }
  // append a comma so that the split works correctly
  var multiLookupValue = value + ",";
  var valueArray = multiLookupValue.Split(new [] {"],"},
                      StringSplitOptions.RemoveEmptyEntries);
  if (valueArray.Length == 0)
  {
    MyMultiLookup = string.Empty;
  }

  List<string> results = new List<string>();
  foreach (var value in valueArray)
  {
    var lookupLabel = value.Split(new[] {";#"},
                StringSplitOptions.RemoveEmptyEntries).Last();
    results.Add(string.Format("[{0}]", lookupLabel));
  }

  MyMultiLookup = string.Join(",", results);
}
```

Get-Result	`"[Value1],[Value2]" : string`
Set-Result	MyMultiColumn set to "[Value1],[Value2]" that results in both values being set in SharePoint

Identifier Transformation

In the case of a migration sync from one SharePoint to another, a source list with a lookup column has to be sent to the new list in the new SharePoint. There is a slightly different lookup set up because the labels changed to another language and the IDs do not match those of the old SharePoint, leading to data issues. Because of the different labels, the former sample will not work. With a simple resolution mapping (old ID = new ID), this example can solve that problem.

Fields

```
Old SharePoint (German) : LookupColumn = "3;#Wert"
New SharePoint (English): LookupColumn = "144;#Value"
```

Code

```
var mapping = new Dictionary<int, int>
{
  { 1, 14 },
  { 2, 18 },
  { 3, 144 },
  { 4, 26 }
};

if(string.IsNullOrEmpty(LookupColumn))
{
  return -1;
}

var id = int.Parse(LookupColumn.Split(new[] { ";#" },

                StringSplitOptions.RemoveEmptyEntries)[0]);

if (!mapping.ContainsKey(id))
{
  var errorMessage = string.Format("Unknown id: {0}", id);
  throw new Exception(errorMessage);
}

return mapping[id];
```

Result

`144 : int` (which in this sample is the Id for lookup label "Value" on the new SharePoint (English))

Identifier Translation via CSV file

This Example is a variation of the previous ID resolution via dictionary, but instead uses a CSV file to define the mapping, making it easier to maintain ID-mappings, for example with Excel.

Note: The CSV file is accessed for every record, so this method can have some performance impacts.

Fields

```
Old SharePoint (German) : LookupColumn = "3;#Wert"
New SharePoint (English): LookupColumn = "14;#Value"

CSV file content:
2;13
3;14
4;15
5;16
```

Code

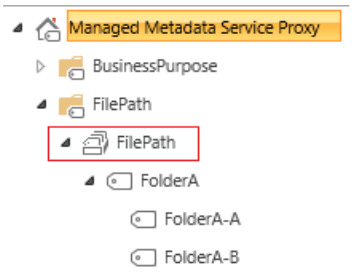
```
if(string.IsNullOrEmpty(LookupColumn))
{
  return -1;
}

var id = int.Parse(LookupColumn.Split(new[] { ";#" },

              StringSplitOptions.RemoveEmptyEntries)[0]);

var csvLines = File.ReadAllLines(@"C:\mapping.csv");

foreach(var line in csvLines)
{
  var entry = line.Split(';');
  if(entry[0] == id)
  {
    return entry[1];
  }
}

var errorMessage = string.Format("Unknown id: {0}", id);
throw new Exception(errorMessage);
```

Result

```
"14" : string
```

Setting Managed Metadata

This example shows how to set a managed metadata column in a SharePoint library to the folder structure that the file resides in. It is assumed that the metadata column is bound to a term set that reflects the folder structure, for example:



The following Dynamic Column is created on the file system-side:

Fields	`FilePath = "/FolderA/FolderA-A/myFile.txt"`

Code	`get`
	`{`
	`  var convertedFilePath =`
	`  FilePath.Substring(1).Replace("/", ";");`
	
	`  if (IsFolder == true)`
	`  {`
	`  return convertedFilePath;`
	`  }`
	
	`  var delimiterIndex = convertedFilePath.IndexOf(";");`
	`  if (delimiterIndex > 0)`
	`  {`
	`  convertedFilePath = convertedFilePath.Substring(0,`
	`  convertedFilePath.LastIndexOf(";"));`
	`  return convertedFilePath;`
	`  }`
	`  else`
	`  {`
	`  return string.Empty;`
	`  }`
	`}`

Result	`"FolderA;FolderA-A" : string, which will be resolved to the managed metadata term "FolderA-A" on the SharePoint side`

Note: In this example the FilePath value is not written back to the filesystem (because there is no set method), so in case someone changes the value of the managed metadata column on the SharePoint side, the change will be ignored in the next synchronization on the filesystem side (although it

notifies about an update). The next synchronization will then automatically correct the value on the SharePoint side back to the actual path as found on the filesystem.

Synchronizing Documents into a Flat Library

When synchronizing files and documents from a file share into SharePoint, it is sometimes necessary to ignore the existing folder structure on the file system and migrate only the given files. In this case, the "FilePath" property needs to have the folder information stripped.

The Dynamic Column serves as a custom replacement for the original FilePath field and has to be created on the file system-side:

Fields	`FilePath = "/FolderA/FolderA-A/myFile.txt"`
Code	`FilePath.Substring(FilePath.LastIndexOf('/'))`
Result	`"/myFile.txt" : string, containing only the filename itself`

In the Mappings section of the connection, map this Dynamic Column to the FilePath property of the SharePoint side.

Note: The example above is for use with uni-directional connections. If you want this to work with a bi-directional connection, you have to define `get{}` and `set{}`, and use a placeholder field in the SharePoint library to store the original folder path for later use.

Accessing Fields with Special Characters

This example shows an alternative way to access fields in case of code-unfriendly names. For the return value, it turns all characters into upper case. More text manipulation methods like ToUpper() can be found here.

Fields	`First&LastName = "Jack J. Jackson"`
Code	`var firstNameAndLastName = (string)fields["First&LastName"];` `return firstNameAndLastName.ToUpper();`
Result	`"JACK J. JACKSON" : string`

Updating Multiple Columns

This example shows how to update multiple columns through one dynamic column. In this case, it is assumed that the data source has separate fields for the first name and last name, but the other side

combines both into one field. So this will split the incoming value at the space character and provide the first/last name fields with the appropriate parts.

Note: This will not work if the full name contains a middle name.

Fields	`(other side) FullName = "Barry Benson"`

Code	`get`
	`{`
	`  return string.Format("{0} {1}", FirstName, LastName);`
	`}`
	
	`set`
	`{`
	`  if(string.IsNullOrEmpty(value))`
	`  {`
	`  FirstName = string.Empty;`
	`  LastName = string.Empty;`
	`  return;`
	`  }`
	
	`  var split = value.Split(' ');`
	`  FirstName = split[0];`
	`  LastName = split[1];`
	`}`

Result	`FirstName: "Barry" : string`
	
	`LastName: "Benson" : string`

Splitting SharePoint-Specific Fields

SharePoint has a field for links that contain an actual URL and a description. This example shows you how to create two dynamic columns that each return either only the URL or only the description.

Fields	`MyLink = "www.mysite.com;#My Homepage"`

Code (LinkUrl)	`if (string.IsNullOrEmpty(MyLink))`
	`{`
	`  return string.Empty;`
	`}`
	
	`var delimiterIndex = MyLink.IndexOf(";#");`
	
	`if (delimiterIndex < 0)`
	`{`
	`  return MyLink;`
	`}`
	
	`return MyLink.Substring(0, delimiterIndex);`

Code (LinkDescription)	`if (string.IsNullOrEmpty(MyLink))`
	`{`
	`  return string.Empty;`
	`}`
	
	`var delimiterIndex = MyLink.IndexOf(";#");`
	
	`if (delimiterIndex < 0)`
	`{`
	`  return MyLink;`
	`}`
	
	`return MyLink.Substring(delimiterIndex + 2);`

Result	`LinkUrl: "www.mysite.com" : string`
	
	`LinkDescription: "My Homepage" : string`

Reading Parameters from XML Document

When you have XML Documents with data and parameters, the Layer2 Cloud Connector can help you not only retrieve and integrate the XML data, also the parameters can be read and synchronized to any other data target.

Fields	`FileName = "myXmlDocument"`

Code	`var assemblyStrongName = "System.Xml, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089";`
	
	`var filePath = @"C:\temp\Sync\" + FileName + ".xml";`
	
	`var xmlAssembly = Assembly.Load(assemblyStrongName);`
	
	`var xmlDocType = xmlAssembly.GetType("System.Xml.XmlDocument");`
	`var loadMethod = xmlDocType.GetMethod("Load", new[] { typeof(string) });`
	`var selectNodeMethod = xmlDocType.GetMethod("SelectSingleNode", new[] { typeof(string) });`
	
	`var nodeType = xmlAssembly.GetType("System.Xml.XmlNode");`
	`var innerTextProperty = nodeType.GetProperty("InnerText");`
	
	`var xmlDocInstance = System.Activator.CreateInstance(xmlDocType);`
	`loadMethod.Invoke(xmlDocInstance, new[] {filePath});`
	`var node = selectNodeMethod.Invoke(xmlDocInstance, new[] {"/item/Title"});`
	
	`return innerTextProperty.GetValue(xmlDocInstance);`

Result	`Title: "My Title Parameter" : string`

Licensing

The licensing model of the Layer2 Cloud Connector provides three different product editions, which are explained in more detail below. The product is licensed per local installation; each server that is running the Layer2 Cloud Connector requires its own license key.

Shareware

This license will automatically be applied if there is no license file found inside of the License directory or if a previous license is invalid for any reason. It restricts the Layer2 Cloud Connector to:

- Only synchronize a maximum of 25 records.

The Layer2 Cloud Connector will read all records from both data sources and synchronize them completely, but when writing back, only the first 25 records will be written, after that the Layer2 Cloud Connector stops the synchronization and puts out a warning.

Personal

With this license, the Layer2 Cloud Connector can be used to synchronize Microsoft SharePoint content without any limitations regarding the record count. The following restrictions apply:

- SharePoint (including Office 365 and OneDrive for Business) must be used as one of the data entities in the connection.
- Only one connection is allowed. For example, between a specific SharePoint library and a specific file share, or between a specific SQL DB data query and an Office 365 list.

Professional

With this license, one can have any number of connections synchronizing any number of records between any supported providers/data sources. For example, Dynamics CRM to SQL Database, SharePoint 2010 to SharePoint Online, or OneDrive for Business to a file share.

SharePoint App Store License

The Layer2 Cloud Connector can also be bought through the SharePoint marketplace. In this case, the license token is stored on the SharePoint instance. The license will be specific to the SharePoint instance and with this license, unlimited number of connections to that SharePoint instance can be created. This license mode can be used together with other local licenses specified in this section. The license can be “Paid” or “Trial”. In every synchronization, the license will be verified against the “Office store verification service”. If the verification fails because of connection problems to the verification service or the trial license expires, the synchronization for this connection will be aborted.

Currently, for technical reasons, there is only a free shareware version available via the Microsoft App Store.

Installing a License

There are two ways to install a license key for the Layer2 Cloud Connector: manually placing the file into the License folder or using the **Update License** action in the Connection Manager UI.

Method One – Manual

1. The license XML file (productkey.xml) will be provided by the Layer2 Sales team by email. Note: Do not modify the signed file in any way. It will invalidate it.
2. Copy (do NOT move) the attached file into the License folder in the Cloud Connector data directory.
    - C:\Documents and Settings\All Users\Layer2 Cloud Connector\License\ OR
    - C:\ProgramData\Layer2 Cloud Connector\License\ (Vista or higher)
3. Restart the Layer2 Cloud Connector Connection Manager and verify the version has been updated to the correct license that was purchased. If it still says "Shareware", then the license was not recognized. Some possible reasons why the license was not recognized are that the license does not match the installed software version, that the file was modified, or file corruption.

Method Two – Using the Update License Action

1. The license XML file (productkey.xml) will be provided by the Layer2 Sales team by email. Note: Do not modify the signed file in any way. It will invalidate it.
2. Copy the attached file to the local machine.
3. Open the Cloud Connector Connection Manager.
4. Select the **Update License** action in the right-hand pane or click the link in the Connection Manager node.

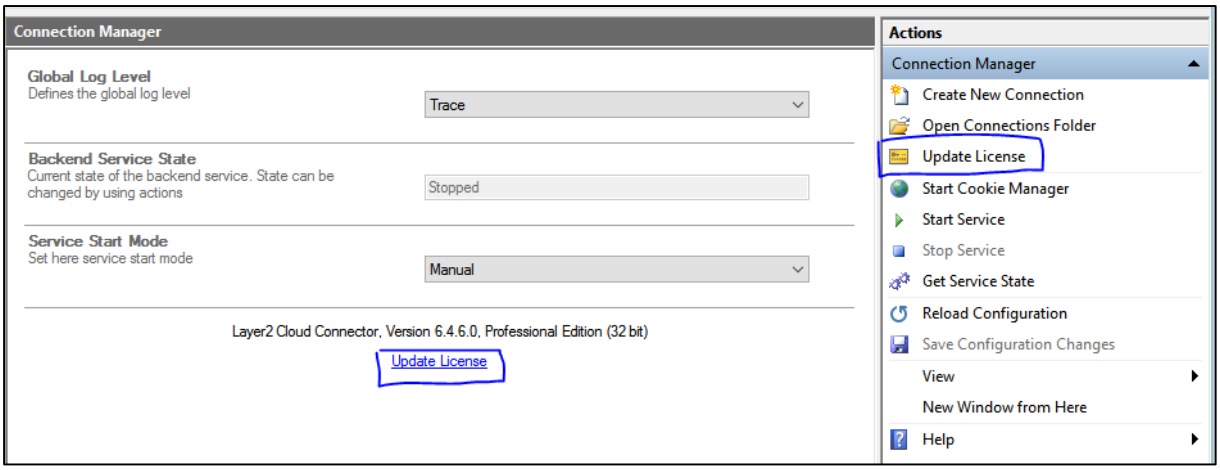


Figure 42 - Location of the Update License action

5. Select the productkey.xml file and click **Open**.
6. Verify the version has been updated to the correct license that was purchased.

Connection Definition Files

All connections will be saved as an XML file (connection name plus .xml extension) on the local machine hosting the Layer2 Cloud Connector. These files can be found in The Layer2 Cloud Connector Data Directory under Connections.

Generally speaking, these files are created automatically by the Connection Manager when you save a new connection. However, as they are simple XML, they can be programmatically generated if you require multiples of files.

<connection>

The root node of the XML file is the `<connection>` node. This node can have the following connection attributes:

enabled

This can be either true or false. It specifies if the connection is enabled for automatic background scheduling and it is mandatory.

interval

This defines the synchronization interval in minutes. This attribute is mandatory although it is ignored if **enabled** is set to "false".

overwriteDestination

This attribute is optional. It can be used to tell the Layer2 Cloud Connector to delete existing records in the synchronization target on an initial uni-directional synchronization and therefore deactivates the security mechanism that is preventing that. (See the Uni-directional Synchronization section.)
It can be either true or false. If omitted, the default value will be "false".

version

This is an internally used attribute which is used by the Layer2 Cloud Connector to differentiate between versions of the connection definition file format. Currently, this should always be set to 1.4.

firstRun

This is a date and time information, defining when the automatic background synchronization will be run for the first time. This setting is only influential if it is set to a future date. The Layer2 Cloud Connector will wait until this specific date and time to start the synchronization for the first time allowing exact timing of the synchronization runs. Different date and time formats can be used here, even localized to the system that runs the Layer2 Cloud Connector. One format which is always accepted is mm/dd/yyyy hh:MM.

Two sub-nodes need to be enclosed in the `<connection>` root node: `<dataEntities>` and `<fieldMappings>`.

<dataEntities>

The `<dataEntities>` node must contain exactly two `<dataEntity>` sub-nodes. A `<dataEntity>` is defined using the following attributes:

name

This sets a name for the data entity. It is mandatory and both entities need to have different names.

type

The type of the entity can have one of three distinct values: source, destination, or bi-directional. The types of both entities are defining the synchronization direction. If one entity is bi-directional, the other one must be bi-directional too. If one of the entities is typed as being destination, the other entity must have the source type.
This attribute is optional. If omitted, both entities will be considered bi-directional.

provider

This identifies the ADO provider that is used for this data entity. This attribute is expecting the invariant name of the provider, which is a unique name to identify a specific ADO provider. This attribute is mandatory.

connectionString

The connection string is the provider-specific configuration information to connect to that data source. For more information on connection strings, see the Configuration: Data Entity section. This attribute is mandatory.

selectStatement

This attribute is optional and is necessary for some providers. It defines a query for selecting the data. For more information on select statements, see the Configuration: Data Entity section.

primaryKey

This attribute is optional and can be used to define a primary key, if it is not automatically returned by the provider. For more information on primary keys, see the Configuration: Data Entity section.

replicationKey

This attribute is optional and can be used to define a GUID as primary key. This value will overwrite any setting returned by the provider or defined in the primaryKey field. For more information on replication keys, see the Configuration: Data Entity section.

disabledOperations

The attribute allows users to define which operations/transactions are omitted during execution of a synchronization. The setting is optional. For more information on disabled operations, see the Configuration: Data Entity section.

Optional Encryption Flags

isConnectionStringEncrypted
isSelectStatementEncrypted
isPrimaryKeyEncrypted
isReplicationKeyEncrypted

These flags are used internally to specify that the contents are encrypted.

The <dataEntity> nodes may also contain optional <dynamicColumns> nodes, which are explained below in their own section.

<dynamicColumns>

The `<dynamicColumns>` node is optional and contains one or more `<dynamicColumn>` sub-nodes. The content of the `<dynamicColumn>` node is an expression (the code executed when the dynamic column is read) encased in a CDATA block. A `<dynamicColumn>` is defined using the following attributes:

name

This mandatory attribute defines the name of the Dynamic Column. It must be unique in the context of the data entity (for example, two Dynamic Columns under two data entities can share the same name).

<fieldMappings>

The second node that is always contained inside the connection node is the `<fieldMappings>` node. It contains the definition of mappings between the fields of both data entities. The `<fieldMappings>` node can only have one attribute which is optional: `<autoMapping>`. This attribute can be either true or false and defines if fields in both data entities that have the same name will automatically be considered as mapped. The default value, if the attribute is omitted, will be false.

Inside of the `<fieldMappings>` node there might be an arbitrary number of `<fieldMapping>` (singular) sub-nodes defining a single mapping. Every `<fieldMapping>` node has exactly two sub-nodes both named `<field>`. These two subnodes define the fields that should be mapped. Each field node must have two attributes: name and entity.

Entity

Entity is the name of the data entity the field belongs to. This information is not case sensitive.

Name

Name specifies the field name. This information is not case sensitive.

In addition, there might be an optional attribute on the `<fieldMapping>` node called isMapped. It can be true or false, and it can be used in case of an automatic mapping to explicitly un-map two fields that have the same name but should be excluded from the synchronization.

Console Mode

The Layer2 Cloud Connector service can be started in console mode. This can be used to:

- Execute connections on-demand.
- Execute several connections with dependencies, one after another.

During installation the setup will create a Start menu shortcut which read **Start synchronization manually** which starts the Layer2 Cloud Connector in console mode. The console application is identical to the Windows service binary. It's located in the program directory as Layer2.Data.Synchronization.Service.exe.

Starting this executable on a command prompt will result in an error message stating that a service cannot be started directly. To make the service run as a console application, it needs command line arguments. If the executable is started with the command line option `-console`, it will start and synchronize all existing connections that are scheduled to run, exactly like the background synchronization service would do it.

If the connections are not scheduled to run, they can be forced to be synchronized by using the additional option `-force`. Connections which are not enabled for background synchronization will never be synchronized even if the force-option is used. Also it is possible to synchronize a specific connection in console mode using the `-execute` argument:

```
Layer2.Data.Synchronization.Service.exe -execute "<connection name>"
```

Logging and Alerting

The Layer2 Cloud Connector creates a variety of messages about its activities during synchronization. These messages will at first be filtered by the current global log level. Any message that gets through this filter will be handed to the logging subsystem NLog. NLog is a free logging framework used by the Layer2 Cloud Connector system to manage its logging activities.

By default, there will be one log file created for each connection. These files will be saved under the Logs subdirectory under the Layer2 Cloud Connecter Data Directory and they are named after the connection they belong to plus the .log extension. You can also see some logging details in the Cloud Connector UI, see the Log section for more information.

Also located in the Logs folder is the logging configuration file, NLog.config. This file contains all logging specific configuration settings and defines different log formats for the Connection Manager and the background Windows Service log files. In the log configuration, writing the log to a .log file

(which is viewable with a simple text editor) is enabled by default. There are also sample configurations for using the Windows Event Log or databases. You can uncomment the desired configuration to log to different targets. To use a database target, a connection string must be edited and a target table must be created. The required SQL script to create an appropriate table is also placed in the configuration file as a comment. Since some log entries contain prohibited characters for SQL, the insertion of some entries may fail.

More detailed information about how to configure NLog can be found in the NLog documentation at https://github.com/nlog/nlog/wiki.

Windows Event Log Configuration

To configure the Layer2 Cloud Connector for logging into the Windows Event Log, a target section in the log configuration file needs to be created, similar to the following:

```
<target xsi:type="EventLog"
      name="EventLogTarget"
      layout="${message}"
      source="Layer2 Cloud Connector"
      log="Application" />
```

xsi:type

Defines the type of target to log to. For the Windows Event Log this needs to be “EventLog”, but there is a variety of different target-types to use. Please see the NLog documentation for a complete reference.

name

This attribute contains the name of the target to reference it in the rules configuration.

layout

This defines the layout of the log message to be written to the event log. It can contain several placeholders of the format ${placeholderName}. Please see the NLog documentation for a complete reference.

source

The name of the source to appear in the Windows Event Log.

log

The name of the event log to be written to.

Additionally, to make the new target work, it is necessary to define a logging rule in the configuration similar to the following.

```
<rule minlevel="Warn"
      name="*"
```

```
        writeTo="EventLogTarget" />
```

Email Alert Configuration

To configure the Layer2 Cloud Connector for sending emails for specific log messages, a target section in the log-configuration file needs to be created, similar to the following.

```
<target xsi:type="Mail"
      name="MailTarget"
      subject="Synchronization Error"
      body="${message}"
      to="me@myCompany.com"
      smtpUserName="cloudConnector@myCompany.com"
      smtpPassword="myPassword"
      smtpAuthentication="Basic"
      smtpServer="mail.myCompany.com" />
```

xsi:type

Defines the type of target to log to. For mail alerts this needs to be “Mail”, but there is a variety of different target-types to use. Please see the NLog documentation for a complete reference.

name

This attribute contains the name of the target to reference it in the rules configuration.

subject

This defines the subject of the mail alert. It can contain several placeholders of the format ${placeholderName}. Please see the NLog documentation for a complete reference.

body

This defines the body-text of the mail alert. It can contain several placeholders of the format ${placeholderName}. Please see the NLog documentation for a complete reference.

This attribute contains the email addresses of the recipients, separated by semicolons.

smtpUserName and smtpPassword

These settings specify the account on the mail server to be used to send the email alert.

smtpAuthentication

Defines the authentication mode to use when authenticating against the SMTP mail server. It can be one of the following.

- **None:** Anonymous access to the mail server. Username and password is not necessary with this setting.
- **Basic:** Basic username and password authentication.
- **Ntlm:** NTLM-Authentication. Username and password is not necessary with this setting.

Additionally, to make the new target work, it is necessary to define a logging rule in the configuration similar to the following.

```
<rule minlevel="Error"
name="*"
writeTo="MailTarget" />
```

Service Management

Part of the Layer2 Cloud Connector system is a Windows service which enables the Layer2 Cloud Connector system to synchronize according to the configured schedule, even if the actual Connection Manager is not running.

This Windows service is registered in the Windows system during installation and will by default be in the "Stopped" state. It is possible to start and stop the service, as well as changing the startup type, from the Connection Manager (see the Configuration section for details), but also the service can be administrated like any other Windows service through the "**Services" MMC-snap-in**.

By default, the Layer2 Cloud Connector service is set up to use the "local system account" on the host machine as the logon account. However, there can be scheduled sync issues if this is left as the default account, as the SYSTEM account may not have access to all data entities. To get around this, the Logon account should be changed to a service account that does have access to the data entities. You can change this in the properties of the Service.

Automatic Fields

The Layer2 Cloud Connector adds some fields to every data-entity result, regardless of the provider in use. These fields are referred to as automatic fields. The name of an automatic field always starts with the prefix CC. Currently the Layer2 Cloud Connector adds the following automatic fields:

CCConnectionName
This field contains the name of the Layer2 Cloud Connector Connection.

CCEntityName
This field contains the name of the Layer2 Cloud Connector data entity.

These can be used in the case where one would execute several different synchronizations with the same target to differentiate between data from a specific connection or data entity.

Layer2 Data Providers

Along with the Layer2 Cloud Connector system, several ADO providers are included in the installation which enable the Layer2 Cloud Connector to connect to many different data entities.

Layer2 Data Provider for SharePoint

The Layer2 SharePoint Provider connects to Microsoft SharePoint 2010, Microsoft SharePoint 2013, SharePoint Foundation and Microsoft SharePoint Online (Office 365 or OneDrive for Business) to retrieve data from and write data to SharePoint lists, calendars, contacts, tasks, and document libraries. The provider does not support a select statement, although querying can be done by setting up an appropriate SharePoint list view. See also the SharePoint (CSOM) Provider Specifications page on the web.

Connection String

A typical connection string for the SharePoint provider looks like this:

Url=<SharePointSiteUrl>;List=<ListName>;Authentication=<auth>

Here is the full list of connection string parameters for this provider:

Url

This is the URL of the SharePoint site-collection where the list or document library is located or the full URL to the view in the SharePoint web-site. When a full URL is used then the list, folder and view information is automatically retrieved from it. Otherwise the list-name must be specified separately. The URL is mandatory.

Authentication

This setting specifies how the SharePoint-Provider authenticates against the target server. See the Authentication Methods section for details.

This setting is optional. If not provided, it will be Anonymous.

User

This part of the connection string specifies the username for the account which is used to authenticate against SharePoint. It needs to be specified for several authentication methods. See the Authentication Methods section for details.

Password

This defines the password for the account which is used to authenticate. It needs to be specified for several authentication methods by typing it into the Connection String or Password field. See the Authentication Methods section for details.

List

The List setting specifies the SharePoint list, calendar, or document library to be used as the data entity. This could be the internal name, the display name or the ID of the list or document library. This setting is mandatory when the full URL to the list is not used in the connection string.

View

The View setting is optional and can be used to define a specific set of sub-data of a list or document library to be synchronized. The view can be created and configured on the SharePoint portal as usual and then the view can be specified in the connection string. The View setting will accept the name of the view as well as the URL of the view's .aspx-site or just the name of the view's .aspx-site.

Folder

With this optional setting, it is possible to define a folder-path which should be used as the root for the synchronization.

BatchReadItems

This setting is optional and specifies how many SharePoint items will be retrieved together from the server in one read request. By default, SharePoint allows reading up to 5000 items.

BatchWriteItems

This setting is optional and specifies how many SharePoint items will be committed together to the server in one update request. By default, SharePoint allows committing up to 50 items together during one update request.

Scope

This setting is optional and specifies how the items on a list with folders will be read. The value "Recursive" tells that all items will be read including subfolders, and that folders are not listed in results. The value "RecursiveAll" gets all items plus folders. Default value is "RecursiveAll".

License

This setting is used to define the licensing method for the connection. By default it is "onpremise", but it can also be set to "SPAppStore" which tells the connection to use a license from the SharePoint application store. This setting is optional. If "SPAppStore" is used and any error occurs during the license verification, synchronization will stop and will not be performed in shareware mode.

AppWebUrl

Parameter is mandatory for and only used when **License** is set to "SPAppStore". The parameter value must be set to the unique URL of the Layer2 Cloud Connector application web on destination SharePoint instance.

TestLicensePath

Parameter is only used when License is set to "SPAppStore". If the SharePoint App Store license is a test license, then a valid local test license must exist to ensure it is a test environment. This parameter holds the path to the local test license.

AutoRenaming

This setting is optional and can either be set to "true" to enable it or "false" to disable it. If it is not specified, the provider will consider it enabled (default value is "true"). Various restrictions concerning item names (forbidden characters, prefixes, etc.) will be avoided by escaping them.

For a detailed description of the renaming logic, see the AutoRenaming section.

AutoZipping

This setting is optional and can either be set to "true" to enable it or "false" to disable it. If it is not specified, the provider will consider it enabled (default value is "true"). If the SharePoint blocks a file upload due to forbidden file extensions, this feature will try to upload a ZIP archive of the file in question. Its name will be appended by ".l2cc.zip" to distinguish it from regular archives.

AutoZippingExceptions

This setting is optional and expects a comma-separated list of file extensions which should be excluded from the auto-zipping feature. If these extensions are blocked by the SharePoint, uploading a file with one of them will throw an error. By default, the following extensions are excluded: pdf, xlsx, doc, docx, jpeg, and jpg.

Managed Metadata Handling

Managed Metadata can be synchronized between mapped fields in SharePoint lists and libraries, if one of the following two prerequisites is matched:

- The fields are bound to the same term set (same SharePoint farm).
- The fields are bound to different term sets (possibly different SharePoint farms) but the terms and the hierarchy in the two term sets are identical.

In the case that Managed Metadata from a list or library is synchronized to non-SharePoint data sources, the values are exported as described in the following examples:

TermSet *Regions*
EMEA
Europe
Germany
Italy
Spain
Middle-East
USA
North America
Washington
Massachusetts
South America

Managed Metadata field is bound to "Regions".
Example 1:
Field has value "Washington"
Value is exported as "USA;Nort America;Washington"

Example 2:
Multiple values are allowed and the field has two values set: "Italy;South America"
Value is exported as "EMEA;Europe;Italy|USA;South America"

To import Managed Metadata values from a non-SharePoint data source into a SharePoint list or library, the values must be in the exact format as described in Examples 1 and 2 above, and must match the bound term set hierarchy and terms of the Managed Metadata field that is being synchronized to. Note that the Layer2 Cloud Connector does not currently support synchronization of Managed Metadata between external sources and the SharePoint Term Store. Please see the Layer2 Taxonomy Manager tool for this.

User/Lookup Handling

For both user and lookup fields, the SharePoint provider can handle “only ID” (e.g., 12), “only value” (e.g., myLookupValue), and full values (e.g., 12;#myLookupValue). Please note that giving only the value will require it to be unique in the target system and comes with a slight impact on synchronization performance.

For lookup fields, you can explicitly map the part of the lookup you wish to use:

- The original lookup field (delivers the full value, e.g. *1;#myLookupValue*)
- The *<lookpField>_idOnly* field (delivers only the id, e.g. *1*)
- The *<lookupField>_valueOnly* field (delivers only the value, e.g. *myLookupValue*).

The latter *_valueOnly* field is required if the values contain numeric values as it’s otherwise not possible to determine whether the passed value is an id or a value.

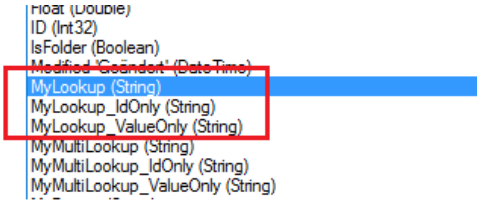
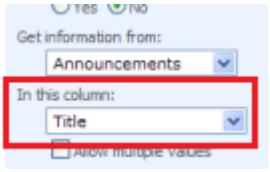


Figure 43 - Example of how the lookup fields are shown in the mapping

Please make sure that when you synchronize from SharePoint to SharePoint that you specify the same source column in the lookup field definition for both sides. So, for example, “Title” should be used on both sides or “ID” on both sides.



Layer2 Data Provider for File System

The Layer2 File System provider connects to local or remote file system resources to retrieve files. Server file shares are also supported. It is typically used to synchronize local files with SharePoint or Office 365 / OneDrive for Business document libraries (including the files) or lists (metadata only). The provider supports both read and write.

The provider does not support select statements. For querying, connection string parameters can be used.

Connection String

As long as a complete select statement is used, the connection string can be left empty, otherwise it usually looks like this:

directory=<folderPath>;

Here is the full list of connection string parameters for this provider:

Directory

This parameter is optional if the select statement states a directory in it's FROM part, otherwise it is mandatory. It is the path for the files to be retrieved. The value can be a local path or a network share in UNC format. If both this and the select statement FROM part are used, the select statement has a higher priority and a warning will be logged.

Note: To avoid errors, it is recommended for network shares that you use the device name (UNC) or its IP address instead of a mapped drive letter.

Authentication

This setting specifies how the File System-Provider authenticates against the target server. See the Authentication Methods section for details.

User

This part of the connection string specifies the username for the account which is used to authenticate. It needs to be specified for several authentication methods. See the Authentication Methods section for details.

Password

This defines the password for the account which is used to authenticate. It needs to be specified for several authentication methods by typing it into the Connection String or Password field. See the Authentication Methods section for details.

FilenameFilter

This is an optional search pattern for filtering files by name or extension. The parameter accepts standard windows file search patterns like these: * ?

Note that only one filter can be set with this parameter.

FileReadMode

This is an optional setting for specifying how to read files. Options are:

- **Recursive (Default):** Gets the content of all folder and subfolder recursively.
- **Flat:** Gets the files only in the specified folder excluding all subfolder contents.

FilesToInclude

This is an optional setting for reading hidden or protected system files. The parameter has three possible values:

- **Visible (Default):** Hidden or protected system files will be excluded.
- **Hidden:** Only protected system files will be excluded.
- **All:** All files regardless from its attribute values will be included.

IncludeFolders

This setting is optional. If it is not specified, the provider will read all folders as records for synchronization by default. If this is not applicable, for example to create a flat list of all files in the folder-hierarchy, it can be set to “false”.

IncludeDirectorySize

This setting is optional. If it is not specified, the provider will return 0 for all directory-sizes. This is due to optimization: To determine the size of a folder, it is necessary to traverse through the whole directory-tree and sum up the file-sizes. This can be very slow on a large number of files and is in many cases unnecessary. To retrieve the actual sizes, set this to “true”.

Select Statement

The File System Provider supports a SQL-like syntax to filter the files and folders found in the given directory as described below.

```
SELECT [Fields] FROM [Directory] WHERE [Filter]
```

The list of [Fields] can either contain a wildcard (*) or a comma-separated list of field names. Furthermore, it is possible to rename the fields by using the SQL alias-syntax.For example, `SELECT FileExtension AS ext` would select the contents of the file extension field and populate it as a field named ‘ext’ in the result.

The FROM part of the select statement currently supports exactly one directory path. This is optional if the connection string contains the ‘Directory’ parameter, otherwise it is mandatory. If both are given, the select statement has a higher priority and a warning will be logged.

The WHERE clause supports various elements known from SQL to build a complex conditional filter:

- Conditional operators: <, >, <=, >=, =, <>, LIKE, IN
- Logical operators: NOT, AND, OR
- Brackets can be used to change precedence

Various wildcards can be used with conditional operators. LIKE supports “%” (any characters) and all other conditional operators support “?” (exactly one character) and “*” (like %).

Example filter only includes files/folders in specific folders:

```
WHERE FilePath IN ('/myFolderA/*', '/myFolderC/myFolderD/*')
```

Example filter excludes files/folders in specific folders:

```
WHERE NOT FilePath IN ('/myFolderA/*', '/myFolderC/myFolderD/*')
```

Example for a simple filter to get all txt files:

```
WHERE FileExtension = '.txt'
```

Example for a complex filter to get all files created before the 1st of February 2016 with 'customer' in their file names:

```
WHERE Filename = '*customer*' AND Created < '2/1/2016'
```

Example filter selects all files starting with the letter 'A':

```
WHERE Filename LIKE 'A%'
```

Example filter selects all files starting with the letter 'A' and ending with 'CDE':

```
WHERE Filename = 'A*CDE'
```

Example filter selects all files bigger than 1 GB:

```
WHERE Size > 1 GB
```

Example filter excludes all files with extension .aspx:

```
WHERE NOT FileExtension LIKE ".aspx"
```

Supported units for file-sizes are:

- KB = Kilobytes
- MB = Megabytes
- GB = Gigabytes

If the "IncludeFolders" parameter is "true", which is the default, all folders that contain files matching the filter, will implicitly be included in the result, even if they do not match the filter. This is to make sure files will never be returned without the respecting folders. If only files should be returned, the "IncludeFolders" parameter needs to be set to "false".

Layer2 Data Provider for XML

The Layer2 Xml Provider retrieves data from XML sources. It is typically used to connect to any XML-based data sources, files, or web requests. It could also be used to integrate systems or applications that do not offer a specific data provider. Data can be exported as an XML file or exposed as XML via HTTP to connect to those kinds of systems. The provider is read-only.

The data is fetched using XPath expressions. See the XML Provider Specifications on the web for more information.

Connection String

A typical connection string for the XML provider looks like this:

Url=<XmlPath>;

Here is the full list of connection string parameters for this provider:

URL

The path to the XML file which can be a URL or a local file path. This field is mandatory. Aliases are: Url, DataSource, Data source, Web, Host, Server, Address.

Authentication

This setting specifies how the XML-Provider authenticates against the target server. See the Authentication Methods section for details.

User

This part of the connection string specifies the username for the account which is used to authenticate against the server. It needs to be specified for several authentication methods. See the Authentication Methods section for details.

Password

This defines the password for the account which is used to authenticate. It needs to be specified for several authentication methods by typing it into the Connection String or Password field. See the Authentication Methods section for details.

Select Statement

The select statement could look like the following example, which requests the title, language, and price of each book in the bookstore XML structure:

```
Select title, title/@lang, price from /bookstore/book
```

The “select” and “from” are fixed keywords (just like SQL select statements). The other parts are XPath expressions which extract the relevant portions out of the XML. Referring to the example above, `/bookstore/book` is the start node of the select, `title` gets the content of the title node (sub-node of book), and `title/@lang` selects the lang attribute of the title node (see figure 35).

Title	Title/@lang	Price
Harry Potter	eng	29.99
Learning Xml	eng	39.95

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
    <book>
      <title lang="eng">Harry Potter</title>
      <price>29.99</price>
    </book>
    <book>
      <title lang="eng">Learning XML</title>
      <price>39.95</price>
    </book>
</bookstore>
```

Figure 44 - Example XML file

Layer2 Data Provider for RSS

The Layer2 RSS Provider retrieves data from RSS (known as Really Simple Syndication) feeds. As RSS is a very common format for data exchange, this can be used to connect many systems. For example, it is used for news feeds. The connector is read-only.

The provider does not support select statements. It gets all mandatory and additional (non-standard) columns from the feed.

Connection String

A typical connection string for the RSS provider looks like this:

Url=<FeedUrl>; FeedType=<FeedFormat>;

Here is the full list of connection string parameters for this provider:

URL

This is the URL of the RSS feed. This information is mandatory.

FeedType

This is the format information of the RSS feed. Possible values are RSS, Atom or RDF. The content will be based on this parameter.

Source

Creates an additional column named “Source”, if it does not already exist, and fills the content with the given value. If the column is present, the values will be overwritten by the given value. This is useful when gathering RSS feeds from different sources.

Authentication

This setting specifies how the RSS provider authenticates against the target server. See the Authentication Methods section for details.

User

This part of the connection string specifies the username for the account which is used to authenticate against the server. It needs to be specified for several authentication methods. See the Authentication Methods section for details.

Password

This defines the password for the account which is used to authenticate. It needs to be specified for several authentication methods by typing it into the Connection String or Password field. See the Authentication Methods section for details.

Layer2 Data Provider for Exchange

The Layer2 Exchange Provider connects to Microsoft Exchange servers, on-premises or cloud-based (Exchange Online, Outlook.com). It can read and write contacts, appointments, emails, notes, and tasks. This provider is often used to mobilize line-of-business data from SQL or ERP/CRM, and make it available for traveling users. It can also be used for aggregating calendars or tasks from several systems and keep it sync.

See also the Exchange Provider Specification on the web.

Connection String

A typical connection string for the Exchange provider looks like this:

User=<UserName>;

Additionally, the password needs to be typed into the Connection String or Password field.

With this connection string, the provider will automatically discover the exchange-server URL by the given user-name and connect with the given credentials.

Here is the full list of connection string parameters for this provider:

URL

This is the URL of the Exchange server. This can be either Exchange Online or on-premises. This setting is optional. If it is not specified, the provider will attempt auto-discovery by using the specified user-name or mailbox-user. Auto-discovery usually takes an additional 4-5 seconds.

Version

This setting specifies the Exchange version which will be assumed while processing requests on the Exchange server. If the version configured by this setting is lower than the actual Exchange version, some properties that have been introduced in a newer Exchange version may not be available. If it is higher than the actual version, errors might be produced if properties are accessed which are not available in the actual Exchange server version.

This setting is optional and in most cases, does not need to be specified at all. If it is not specified, the provider will attempt to detect the server-version automatically.

Possible values are:

- Exchange2007_SP1
- Exchange2010
- Exchange2010_SP1
- Exchange2010_SP2
- Exchange2013
- Exchange2013_SP1

User

This parameter is **mandatory**. The name of the user to authenticate with against the Exchange server. If the user is not associated with a mailbox on the Exchange server, the **Email** parameter will be mandatory.

Password

This parameter is **mandatory**. The password of the user to authenticate with against the Exchange server. It needs to be typed into the Connection String or Password field.

Email

This setting defines the mailbox which should be accessed. It is optional and if not specified, the provider will access the mailbox of the user specified with the **User** parameter. If the **Email** parameter is specified, the user which is used for authentication (specified with the **User** parameter) must have permission to access the mailbox account. Based on the **Impersonation** parameter, this either means delegated access or permission to impersonate.

Impersonation

When using the **Email** parameter to access a different mailbox, you can either do so via Impersonation (Impersonation=true) or Delegation (Impersonation=false). This setting is optional and defaults to “false”.

Recursive

This setting is used to define whether the provider should traverse through all subfolders to gather the items (Recursive=true) or only return items which are direct children of the specified folder (Recursive=false). This setting is optional and defaults to “false”.

NamingScheme

When synchronizing specific item-types, such as contacts or appointments, to another system, fields with the same meaning may have different names in both systems. To minimize the effort for setting up synchronizations, the Exchange provider supports naming-schemes which effectively rename the resulting columns to match the names of the target system, which enables automapping-functionality for these scenarios. Currently, there is one naming-scheme “SharePoint”, which will rename the Exchange-columns for contacts and appointments to match the corresponding SharePoint fields. This setting is optional.

BatchReadItems

With this parameter it is possible to define the maximum number of items that will be read in one request from the Exchange server. This setting is optional, defaults to 1000, and normally does not need to be adjusted. When working with large amounts of data, performance may be enhanced by tweaking this setting.

TrustAllSSL

This parameter can be used to make the provider bypass the validation of certificates when connecting through HTTPS. This setting is only for testing purposes; it is NOT recommended that this be used in a production environment due to the security risk. The setting is optional and defaults to “false”.

Queries

The Exchange provider supports a SQL-like syntax to query the items from the Exchange server as described below.

```
SELECT [Fields] FROM [Folder] WHERE [AQS-filter]
```

The list of fields can either contain a wildcard (*) or a comma-separated list of Exchange-property names. Furthermore, it is possible to rename the fields by using the SQL alias-syntax, i.e. SELECT Companyname AS cmp, would select the contents of the property Companyname and populate it as a field named cmp in the result.

The folder part specifies the Exchange folder to select the items from. It is defined as a folder path separated by slashes (/) while the first folder must always be one of the Exchange root folders as seen below:

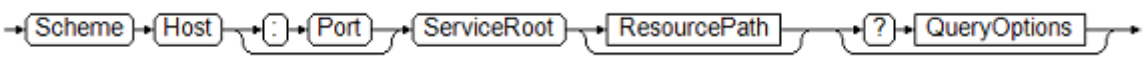
Calendar	- Root folder for calendars.
Contacts	- Root folder for contacts.
DeletedItems	- Deleted items folder.
Drafts	- Drafts folder.
Inbox	- Mail inbox.
Journal	- Journal items.
Notes	- Notes folder.
Outbox	- Mail outbox.
SentItems	- Mail sent items.
Tasks	- Tasks folder.
MsgFolderRoot	- Root folder for public folders.
PublicFoldersRoot	- Root folder for public search folders.
SearchFolders	- Root folder for search folders.

The filter part, which starts with the keyword WHERE, is optional and contains a filter-query in the Advanced Query Syntax. For more information about AQS, please refer to the Microsoft documentation.

Layer2 Data Provider for OData

The Layer2 OData provider retrieves data from OData sources. The Open Data Protocol (OData) is a Web protocol for querying and managing data. OData is being used to expose and access information from a variety of sources including, but not limited to, relational databases, file systems, content management systems, and traditional web sites. Examples of popular applications supporting OData are Microsoft Dynamics, SAP via NetWeaver, Microsoft SharePoint, and Visual Studio TFS. Also MS-SQL databases might be populated through this protocol.

OData requests are presented as a single URL. The parts of an OData URI are separated to different fields, e.g., the select statement of the OData provider refers to the Query Options section of an OData URL.



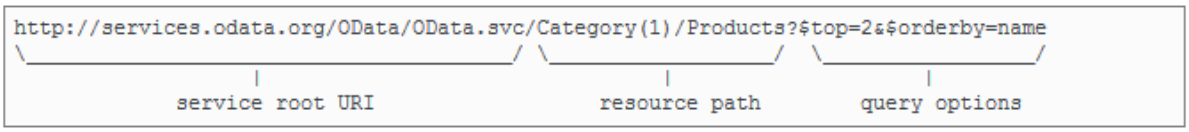


Figure 45 - OData URI construction

The Layer2 Cloud Connector comes with two different versions of the Odata provider. One is based on the .NET 3.5 framework and only supports OData up to version 2. The other one uses the .NET 4.5 framework and supports all current OData-Versions, V1 to V4.

Accordingly, installing the .NET 3.5 version of the Layer2 Cloud Connector will install the .NET 3.5 Odata provider and installing the .NET 4.5 version will install the .NET 4.5 Odata provider.
Since the .NET 3.5 Odata provider's functionality is more limited than the .NET 4.5 Odata provider, it is recommended to install the .NET 4.5 version of the Layer2 Cloud Connector, if possible.

Also when using the .NET 4.5 version of the Layer2 Cloud Connector, it is possible to switch to the .NET 3.5 Odata provider by choosing the provider **Layer2 Data Provider for OData (Deprecated)**.

Odata Providers	OData-Provider for .NET 3.5	OData-Provider for .NET 4.5
OData-Protocol V1	✓	✓
OData-Protocol V2	✓	✓
OData-Protocol V3	✗	✓
OData-Protocol V4	✗	✓
Atom-Format	✓	✓
Json-Format	✗	✓
Entity-Inheritance	✗	✓
SQL-Style Selects	✗	✓
Batched Writing	✗	✓
Collection URL´s	✗	✓
HTTP/PATCH support	✗	✓
HTTP E-Tag support	✗	✓

Figure 46 – Odata provider comparison

OData Specifications V1-V4

There are currently four versions of the OData-specification. The full specifications can be found at http://www.odata.org/.

Formats – ATOM and JSON

The OData-protocol can provide the data in either the *Atom Syndication Format (ATOM)* or *JSON*. Both formats are supported. If a server also supports both formats, JSON will be given preference.

Specifying the URL

The URL which is specified for the provider can point to the root of an OData-service or to an OData-collection directly. In the latter case, the provider will determine the root URL automatically from the collection-response.

Selecting Data

Select statements can be defined by using either the OData query syntax or a SQL-style select statement. Note that the .NET 3.5/Deprecated version of the Odata provider only supports the classic Odata query syntax and not the SQL-style syntax.

The OData query syntax is the style that uses query options normally added as part of an OData URL, which start after the question mark as defined in the specification. The keywords `$value`, `$inlinecount`, `$count`, and `$format` are not supported.

The SQL-style query is a query using the SQL-like-syntax:

```
SELECT [fields] FROM [source] WHERE [filter]
```

The provider will translate the query to a corresponding Odata query. The source is Odata collection parameter that was used in the connection string. If you use the SQL-style query, you do not need to define the "Collection" parameter in the connection string (as it will be in the select statement).
The filter expression supports all functions (like `endswith()`, `startswith()`, `indexof()`, etc.) that can be used in an Odata query. Also the SQL TOP keyword is supported. ORDER BY clauses are not supported, but also not necessary, since the order of records does not affect the synchronization.

Batching and Paging

The Odata provider supports the batching of operations sent to an OData-service as well as the paging of results returned. Paging is always driven by the server, which decides how many records it will return in one page and the provider will read all the returned pages accordingly.
When the provider sends operations (such as inserts, updates, and deletes) to the OData-service, it will, by default, attempt to send the operations as a batch to the OData batch-endpoint `$batch`. If the request fails because the server does not provide a batch endpoint, all operations will be sent

one-by-one. By default, the provider will create batches of 500 items. The batch-size can be configured by specifying the "*BatchSize=;*" parameter in the connection string.

Entity-Inheritance

The provider supports inheritance of entity- and complex-types defined in the metadata-description of the OData-service.

Specifying the Entity-Type

In most cases, the provider can automatically detect the entity type. There are some rares case when this is not possible:

- The URL points to a collection (i.e. has not been specified explicitly with the "*Collection=;*"property)
- The format is ATOM
- The data source is empty

In this case, the provider will return an error stating that the entity-type needs to be specified explicitly. Specifying the entity-type can also be useful, especially if the collection does contain different types of entities.

The Update HTTP-Verb

There are different HTTP-verbs which both are frequently used to indicate an update to an OData-service: MERGE and PATCH. The provider supports both verbs and tries to detect automatically which one is expected by the server. By default, it tries to use the MERGE-verb for services with OData versions V1-V3 and the PATCH-verb for services with version V4. If the default verb is rejected, the provider will use the alternative. To prevent this additional step, the verb can also be defined explicitly as part of the connection-string.

The Entity Tag (etag)

The entity tag is a HTTP-header which specifies the version of a HTTP-resource. It is commonly used to manage caching of web-resources. Some OData-services provide an etag-header and also expect that the provided tag is sent on subsequent update-requests, which enables the server to make sure the entity has not been modified since the last read-request.
The entity-header is populated as a read-only field in the result table.

Reading SharePoint Lists

Microsoft SharePoint also provides access to all content (Webs, Libraries, Lists, etc.) via Odata. The SharePoint OData endpoint can be found at siteCollection/_api. If, for example, the site collection is at https://mySharepoint.com/siteCollection, the OData-endpoint would be at https://mySharepoint.com/siteCollection/_api. To access list or library contents the URL would be: https://mySharepoint.com/siteCollection/_api/web/lists('8E6C08E4-AD36-470D-AB8B-276306D88A10')/items, for a list with the given ID.

Reading from SAP NetWeaver

SAP NetWeaver requires a csrf-token header to be initially fetched and sent with all subsequent data modification requests (insert, update, delete). To enable support for the csrf token, the keyword *SAPCsrfHeader* must be added as part of the authentication type in the connection string (for example:

```
Authentication=Windows,SAPCsrfHeader;
```

This means that Windows authentication is used and the csrf token is supported).

Connection String

A typical connection string for the OData provider looks like this:

Url=<DataServiceRootUri>; Collection=<ResourcePath>; Authentication=<AuthenticationType>;

or

Url=<FullCollectionUri>; Authentication=<AuthenticationType>;

Here is the full list of connection string parameters for this provider:

URL

This can be either the service root URL for the OData source or the full URL to an ATOM or JSON representation of an OData-collection. This parameter is **mandatory**.

Collection

This parameter defines the OData-collection that is to be accessed as a URL relative to the root of the OData-service, which in many cases it is just the name of the collection.
If the URL defined with the URL connection-string parameter already points to a collection or if the collection is specified in the FROM-part of a SQL-style select, this parameter should not be specified.
This parameter is optional in .NET 4.5 version of the provider, but mandatory for .NET 3.5/Deprecated version.

Authentication

This parameter specifies how the OData provider authenticates against the target server. See the Authentication section for details. In case of an SAP Netweaver connection, the authentication parameter needs to also contain the *SAPCsrfHeader* keyword.
This parameter is optional.

User

This part of the connection string specifies the username for the account which is used to authenticate against the server. It needs to be specified for several authentication methods. See the Authentication Methods section for details.

Password

This defines the password for the account which is used to authenticate. It needs to be specified for several authentication methods by typing it into the Connection String or Password field. See the Authentication Methods section for details.

EntityType

This parameter can be used to explicitly define the type of the OData-entity, either if it could not be determined automatically or if the collection can contain multiple different entity-types.
This parameter is optional.

BatchSize

Specifies the number of operations which will be sent in a single batch. The default is 500.
This parameter is optional.

Batching

Can be used to explicitly turn batching of operations sent to the OData-server off or on. It accepts the values “Auto”, “True”, or “False”. If it is not specified, it defaults to auto. In auto mode, operations will be sent as a batch to the OData *$batch*-endpoint and, if that fails, all operations will be sent one-by-one in single requests.
If the server does not support batching, it can be explicitly turned off with *Batching=False*, to prevent the provider from executing the failing batch-request.
On the other hand, the provider might try to commit operations in single requests, even if batching is supported, if the batch-request fails due to an unrelated error.
In this case, batching can be explicitly turned on with *Batching=True*, to prevent the provider from unnecessarily retrying with single-requests.
This parameter is optional.

UpdateVerb

Defines the verb, which should be used to indicate update-operations to the OData-server. This can be any text, but typically only *MERGE*, *PATCH* or *Auto* would be used.
The default is *Auto*, which will first try the HTTP/MERGE verb for OData V1-V3 and fallback to HTTP/PATCH if that fails. For OData V4 it will first try HTTP/PATCH and then fallback to HTTP/MERGE. To prevent the provider from performing unnecessary failing requests, the verb can be defined explicitly with this parameter.
This parameter is optional.

IEEE754Compatible

This parameter specifies if the JSON format that is sent to the service should be IEEE754 compatible. The default value is “auto” which will first try “ExplicitOn” and then “ImplicitOn”. If the service accessed is not IEEE754-compatible, this parameter should be set to “ExplicitOff”. If the service accessed cannot handle the IEEE754 content type argument, but still expects IEEE754-compatible

content, this parameter should be set to “ImplicitOn”. If the service accessed is not IEEE754-compatible and cannot handle the content type argument, this parameter should be set to “ImplicitOff”.
This parameter is optional.

DateTimeFormat

This parameter specifies the JSON format that is used to send dates. The default-value is *yyyy-MM-ddTHH:mm:ssZ* (for example, *2016-01-01T10:10:10Z*). For connections using the SAPCsrfHeader authentication, the default-value is *yyyy-MM-ddTHH:mm:ss*.
This parameter is optional.

Timeout

This parameter allows for timeout values to be specified for situations where the default timeout is not long enough for the data pull. It is defined in seconds. This parameter is optional.

Layer2 Data Provider for Office 365 Groups

The Layer2 Office 365 Groups provider connects to Webhooks of Microsoft Office 365 Groups so that it can write a “Card” to it after each synchronization. The card contains information about changes in the source data entity that were made since the last synchronization run. It can, for example, be used to regularly notify Office 365 Group members about changes that have been made in a certain local or cloud-based business system like SQL, ERP/CRM, etc., as supported by the Layer2 Cloud Connector.

The Layer2 Data Provider for Office 365 Groups cannot be used to sync documents in an Office 365 Groups document library. You can use of the Layer2 Data Provider for SharePoint to do this.

Group Cards can only be written to but not read from Office 365 Groups using the Cloud Connector.

The Layer2 Office 365 Groups provider does not manipulate any data but just writes cards to an Office 365 Group. Therefore, it is best practice to define the connection as uni-directional (although bi-directional also works).

Card Reference

The card that will post the changes contains the following elements:

CardTitle	The title of the card. The content of this field needs to be specified in the connection string as the “CardTitle=;” parameter.
Summary	The summary contains information about the changes that were made since the last synchronization run. The format is:

	x items inserted, y items modified and z items removed in <Office 365 data entity name[1]>.
Section	A section is created per item that has been changed. The title gives information about the data record and the kind of change. The format will be similar to the following: *Item '<record-key>' was added to '<Office 365 data entity name>'.*
Section-Details	Each section can contain detailed information about the changes made to the data record. The information is shown as table of name/value pairs, showing the mapped fields (see below) and their new values.

Connection String

A typical connection string for the Office 365 Groups provider looks like this:

WebHookUrl=<url of the webhook to the Office 365 group>; CardTitle=MyCardTitle

Here is the full list of connection string parameters for this provider:

WebHookUrl

The WebHookUrl is created in the Office 365 Group you want the Cloud Connector to send cards to. When opening the Office 365 Office group, you can find an option at the top menu bar called **Connectors**. Click on that to start configuring your Webhook Connector.

You need to create a connector by choosing the "Incoming Webhook" Connector and then click **Add**.

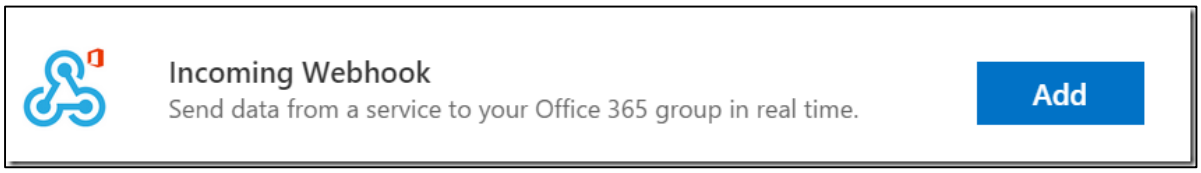


Figure 47 –"Incoming Webhook" Connector in Office 365 Groups

To configure the Incoming Webhook, you need to give it a name and optionally add an image for the Connector. Once that is done you need to click on **Create** to create the Webhook. When it is created you will see an input box with a URL in it. This is the Webhook and you need this URL in the

[1] No data is changed in the Office 365 Group (just a card sent), however the group is treated as target data entity and "as-if" data had been changed there during the synchronization.

connection string of the Cloud Connector. The URL is unique to this Group and Webhook. If you remove the Webhook, you will disable it.

Note: The Webhook URL will validate even if it is incorrect as it cannot be checked without sending a card to post.

CardTitle

This parameter sets the text that is written as title of the card. It is recommended to use the connection name so that you can immediately see which connection sent this card.

SkipDetails

When this parameter is set to “true”, no sections are created, but just the CardTitle and the Summary information. This is useful if many data records have been changed (for example, due to a mass update in your business system) and you do not want to get every change detail but just the summary of the synchronization This parameter is optional and defaults to “false”.

Select Statement

The select statement defines which fields are offered for mapping and, if mapped, what will be shown in the section details.

The format for the select statement is this, where `[FIELDS]` is a comma-separated list of the fields:

```
SELECT [FIELDS]
```

Fields must be explicitly defined; the wildcard * is not supported.

Note: Only changes for fields that are mapped will be tracked. So in case a field of a record in your business system has changed and this field is not mapped (and therefore not part of the synchronization), the change will not be tracked in the card at all.

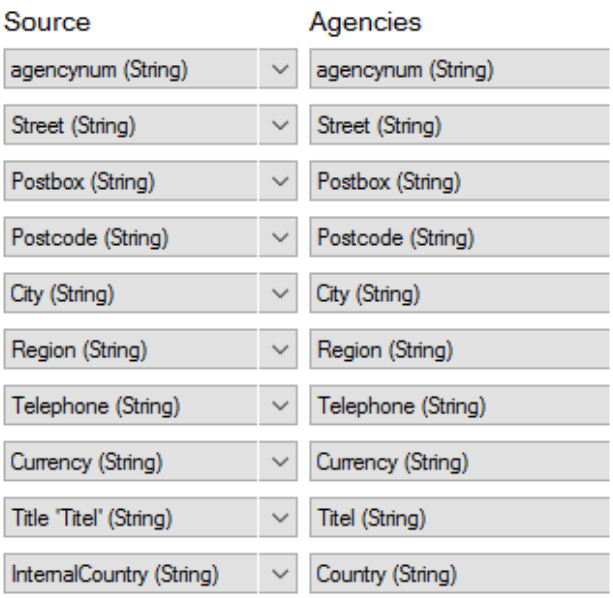
Primary Key(s)

You need to specify the field(s) that are used to create the Section title. If using a GUID- or counter-type of primary key, it might be a good idea to add another detailed column (for example, “Agencynum, Title”) to have human-readable information in the card.

Example:

Synchronization from an Agency list in SharePoint to Office 365 Group

Connection string	WebHookUrl=<MyWebHookUrl>;CardTitle=O365Groups-SP Agencies;

Select statement	select agencynum, Titel, Street, Postbox, Postcode, City, Region, Telephone, Language, Currency, Country
Primary Key(s)	Agencynum, Title
Mapping	

Example resulting Group Card for the initial synchronization:

- **O365Groups-SP Agencies** has been set via CardTitle parameter in the connection string
- **SPO Agencies** is the name of the created Webhook
- **SharePoint logo** has been set when the Webhook was created
- **AgencyList** is the name of the source data entity

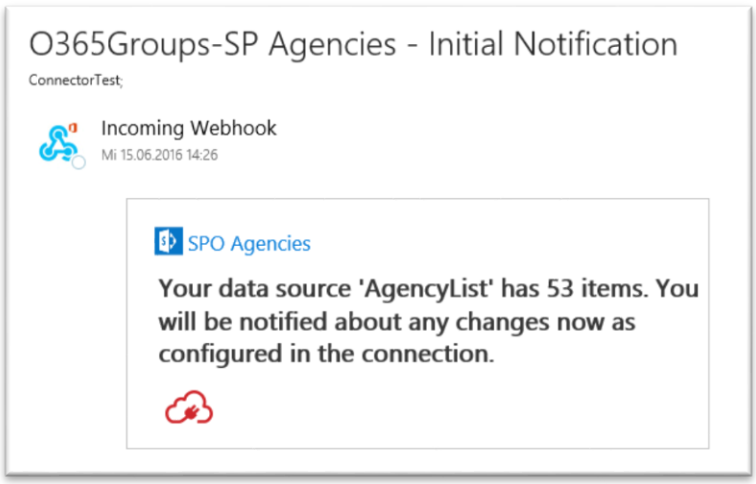


Figure 48 - Example card output for initial synchronization

Example resulting Group Card for subsequent synchronization:

- The detail fields are shown in the order as specified in the select statement
- Detail fields are only shown, if changed

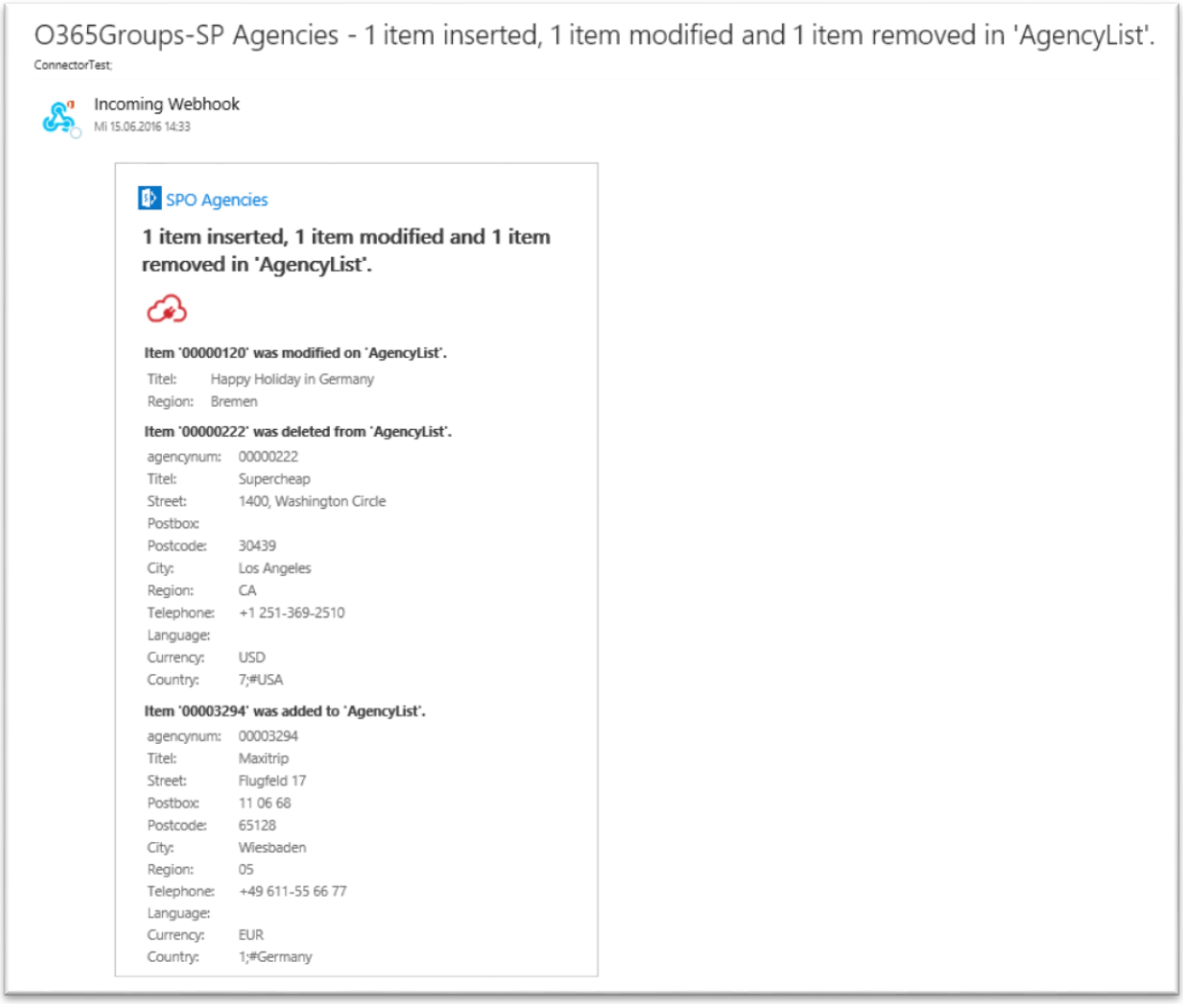


Figure 49 - Example card after a synchronization with changes

Layer2 Data Provider for Microsoft Flow and Logic Apps

The Layer2 Microsoft Flow and Logic Apps provider connects to the Request trigger API of a Flow or an Azure Logic App. It is used to write inserted, changed, and deleted data (change notifications) to the Request trigger of the Flow or the Logic App after each synchronization. The change notifications contain information about changes in the source data entity that were made since the last synchronization run. The information is sent as a JSON payload and if the related JSON schema has been pasted to the Request trigger, it can be used in further actions or conditions of the flow.

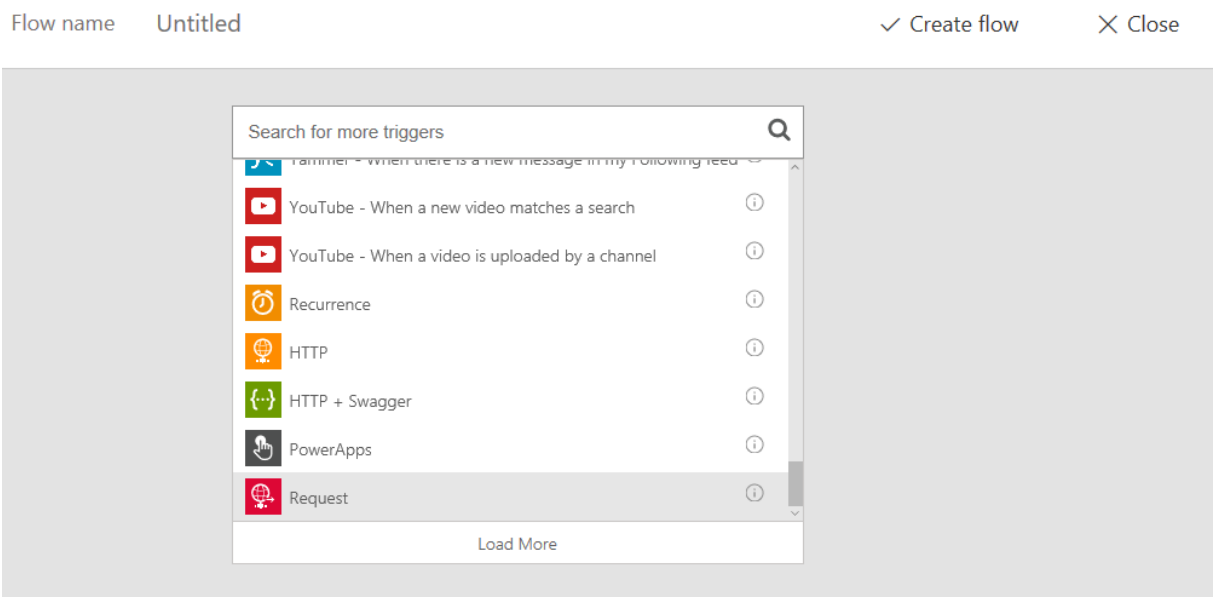
Change notifications can only be written to but not received from Flow or Logic Apps using the Cloud Connector.

The Layer2 Microsoft Flow and Logic Apps provider does not manipulate any data but just writes change notifications to a Flow or Logic App in order to trigger the flow. Therefore, it is best practice to define the connection as uni-directional (although bi-directional does work, but does not change anything).

How to Setup a Microsoft Flow that is Triggered by the Cloud Connector

To use this provider to trigger a Flow, you just need to sign up with an email address. To create a Flow that receives the Change Notifications from the Cloud Connector, follow these steps:

1. On the Flow landing page, sign in (or sign up, if not yet done).
2. Click “My flows” on the top menu bar.
3. On the “My flows” page, click “Create from blank”
4. The first item you need to create is the trigger. This is the event that will start your Flow. Select the “Request” trigger:



5. Paste the JSON schema from the Cloud Connector (explained in section Create and Get the JSON Schema below) into the **Request Body** field:

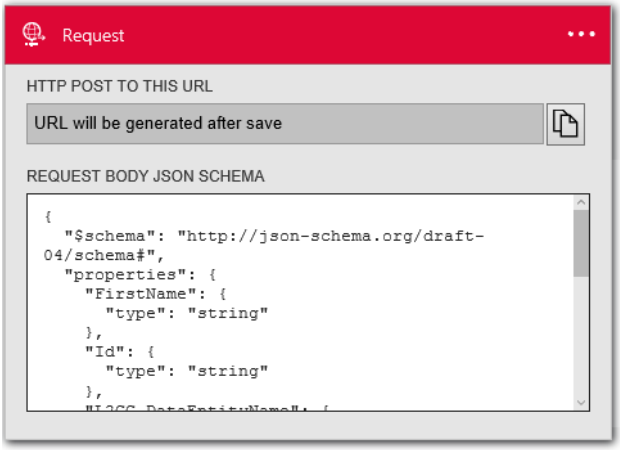


Figure 50- Example request body JSON Schema in the API UI

6. Add an action (for example “Send an email”) and fill in the required fields:

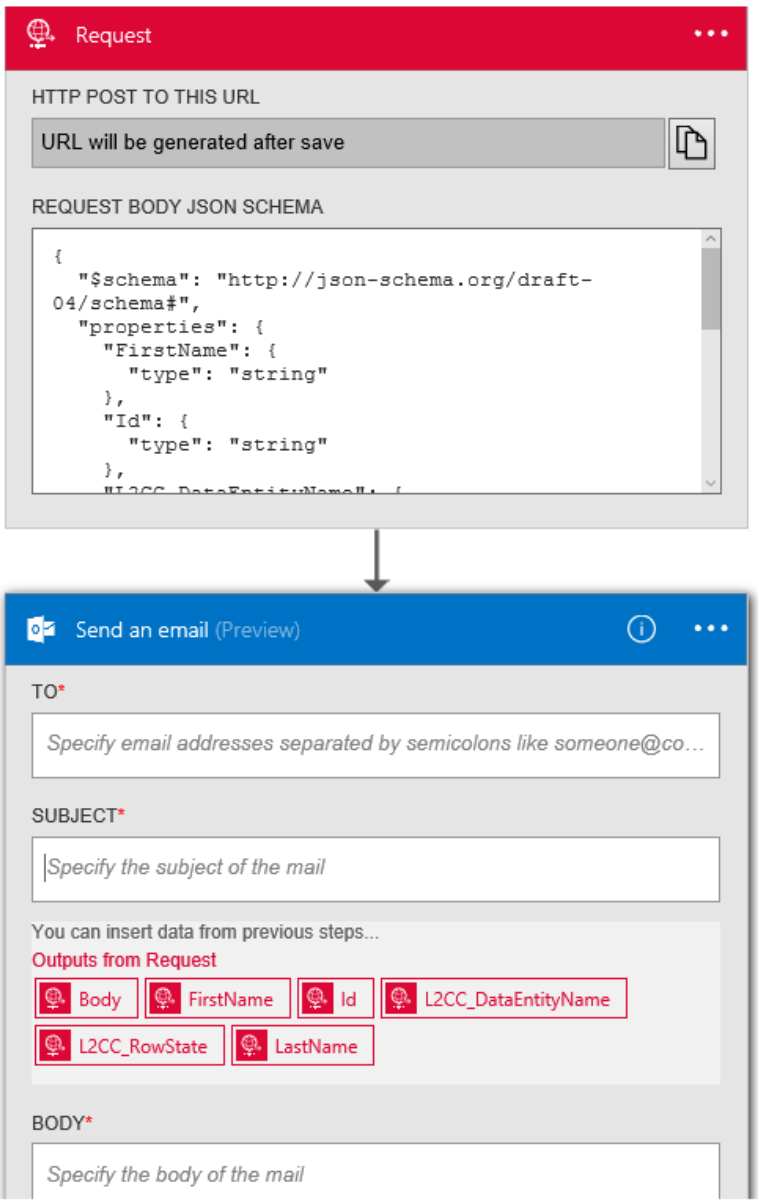


Figure 51 - UI for adding Send an Email action

As you can see, the fields from the JSON schema are shown to be used (shown under “Outputs from Request”.

7. Add a name for the Flow (at the top).
8. Save the Flow and copy the URL that has been created for the trigger

Figure 52 - Example Flow Trigger URL

How to Setup an Azure Logic App that is Triggered by the Cloud Connector

To use this provider to trigger a Logic App, you need an Azure subscription. To create a Logic App that receives the Change Notifications from the Cloud Connector, follow these steps:

1. On the Azure portal dashboard, select **New**.
2. In the search bar, search for 'logic app', and then select **Logic App**. You can also select **New, Web + Mobile**, and select **Logic App**.
3. Enter a name for your Logic App, select a **Location**, **Resource Group**, and then click **Create**. If you select Pin to Dashboard the Logic App will automatically open once deployed.
4. After opening your Logic App for the first time you can select from a template to start. For now, click **Blank Logic App** to build a new Logic App.

5. The first item you need to create is the trigger. This is the event that will start your Logic App. Select the “Request” trigger:

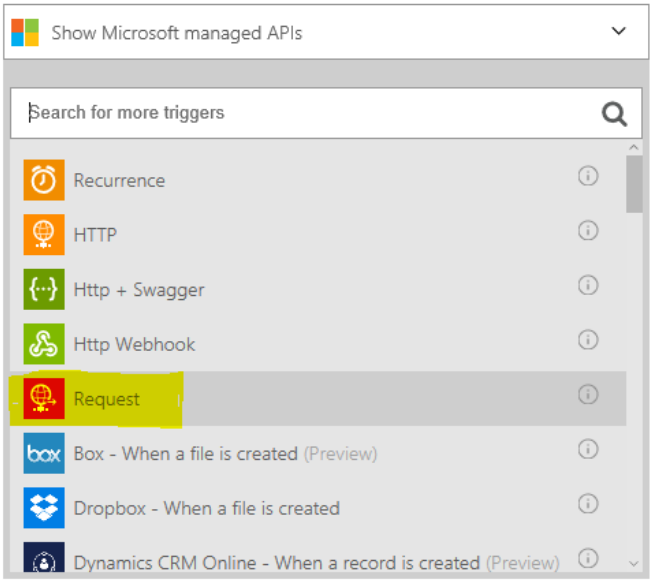


Figure 53- Logic API triggers, with 'Request' highlighted

6. Paste the JSON schema from the Cloud Connector (explained in section Create and Get the JSON Schema below) into the **Request Body** field:

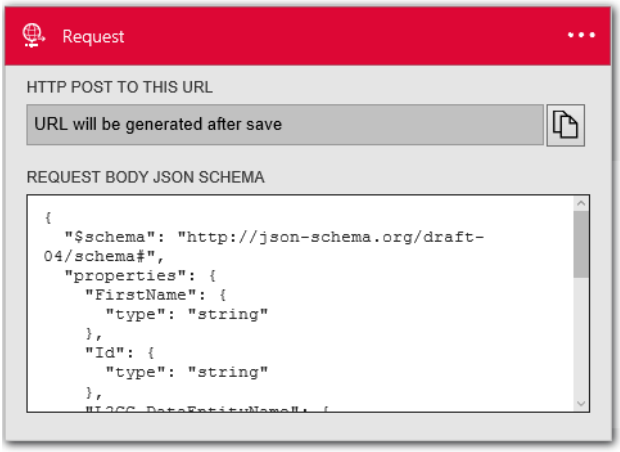


Figure 54- Example request body JSON Schema in the API UI

7. Add an action (for example “Send an email”) and fill in the required fields:

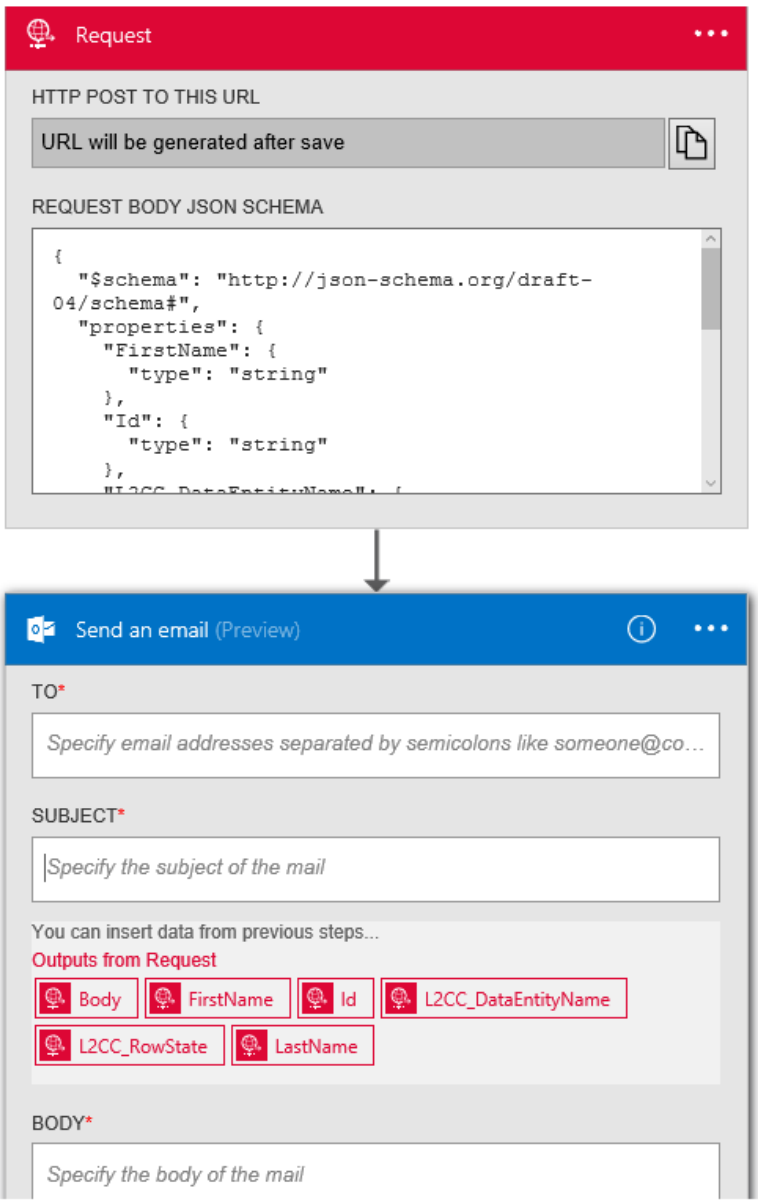


Figure 55 - UI for adding Send an Email action

As you can see, the fields from the JSON schema are shown to be used (shown under “Outputs from Request”.

8. Save the Logic App and copy the URL that has been created for the trigger:

Figure 56 - Example Logic App Trigger URL

Change-Notification Payload

Each of the notifications that will be sent contain information about a changed record in the source system. The information is sent as JSON payload and the Cloud Connector provides a feature to get and copy the JSON schema of the notification in order to paste it into the Flow's or Logic App's **Request** trigger.

Connection String

A typical connection string for the Flow and Logic Apps provider looks like this:

WebHookUrl=<url of the flow or logic app trigger>

Here is the full list of connection string parameters for this provider:

WebHookUrl

The WebHookUrl is created in the Flow or Logic App you want the Cloud Connector to send notifications to.
After saving the Flow or Logic App, the URL of the Request trigger action will be displayed (see above). This URL is the Webhook and you need to use this URL in the connection string of the Cloud Connector.

Note: The Webhook URL will validate even if it is incorrect as it cannot be checked without sending a notification to post.

SkipNotifications

In case of an initial sync or in case of any data updates that should not be sent to the Flow or Logic App, you can specify the parameter *SkipNotifications=true*. Default value is "false", so that change notifications for inserted, deleted, and updated records are sent. However, you can suppress them using this connection string parameter, if desired.

Primary Key(s)

You must specify the field(s) that are used to identify the records on the target (Flow or Logic App) side.

Select Statement

The select statement defines which fields are offered for mapping and of what data type they are.

The format for the select statement is a comma-separated list of the fields and optionally their data types (shown in brackets, because of being optional):

```
SELECT FIELDNAME1[:DATATYPE], FIELDNAME2[:DATATYPE]
```

Datatypes that can be specified are the following:

- integer
- long
- float
- decimal
- double

The above datatypes will be sent as JSON type *number* in the notification.

- string
- byte
- datetime

The above datatypes will be sent as JSON type *string* in the notification.

- boolean

The above datatype will be sent as JSON type b*oolean* in the notification.

If no datatype is specified, the default type *string* will be used.

Example Select Statement:

Select Id:integer, CreditLimit:float, Name, ExistingCustomer:boolean

In this example, the field “Name” will be of type string, as no explicit datatype has been specified.

Fields must be explicitly defined; the wildcard * is not supported. In case no data type has been defined, the default of *string* is used.

Note: Only changes for fields that are mapped will be sent with the change notification. So in case a field of a record in your business system has changed and this field is not mapped (and therefore not part of the synchronization), the change will not trigger a change notification.

Create and Get the JSON Schema

In the data entity with the Flow and Logic Apps provider, there is a section **Json Schema**. After clicking “Show” (marked yellow in the image below), the schema is created based on the select statement and is shown in a popup window. You can now copy the schema or save it to disk. This is the schema you need to paste into the JSON Schema Body of the Flow or Logic Apps Request Trigger (see How to setup an Azure Logic App that is Triggered by the Cloud Connector).

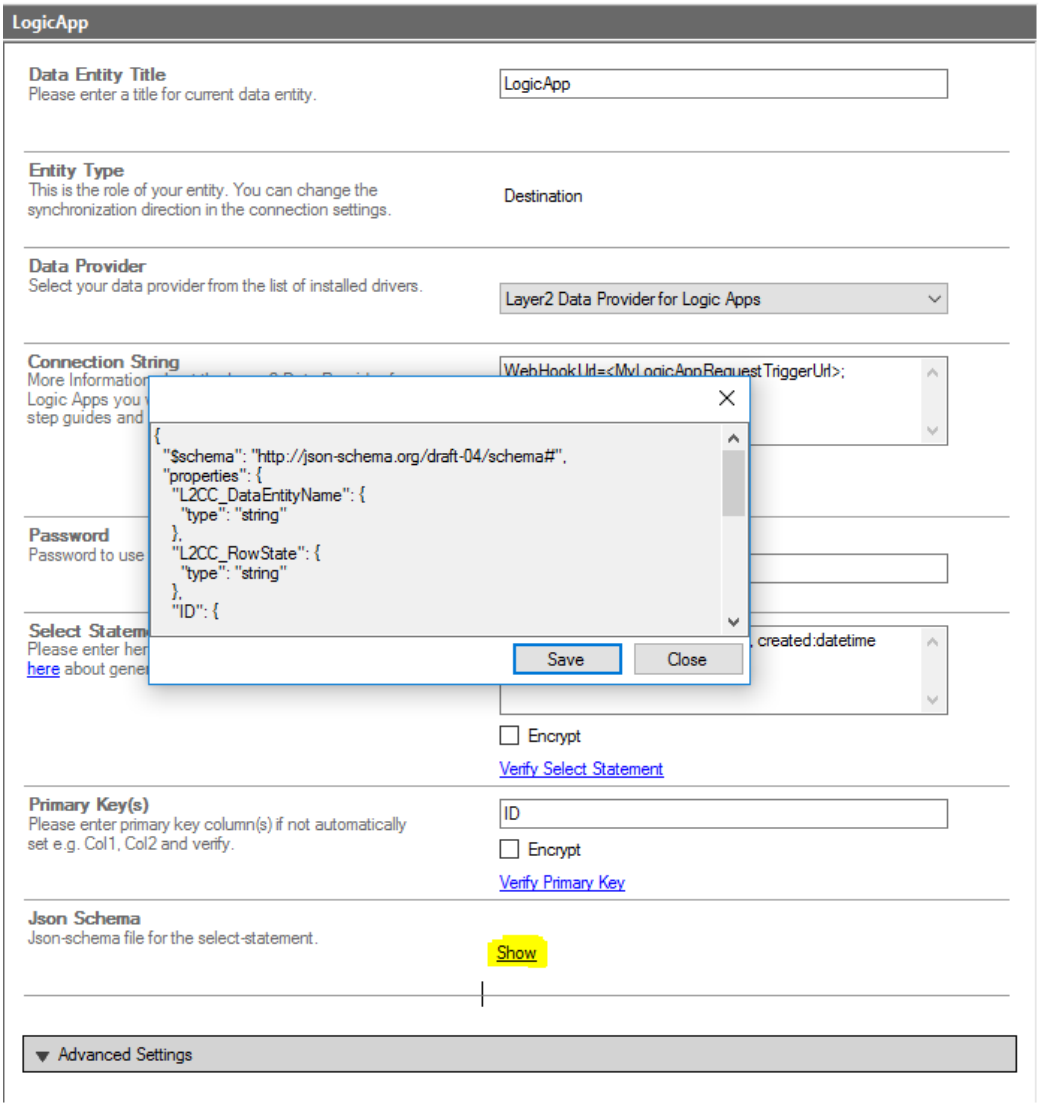
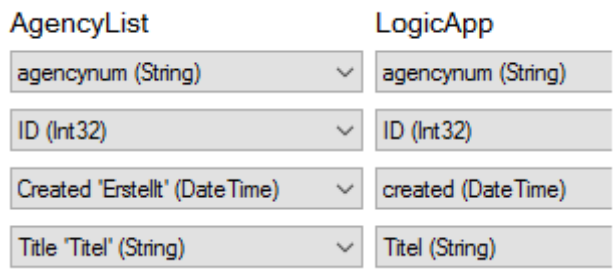


Figure 57 - Show link for the Json Schema option for the Flow and Logic App Provider

Example:

Synchronization from an Agency list in SharePoint to a Logic App.

Connection string	WebHookUrl=<MyLogicAppRequestTriggerUrl>;
Select statement	select ID:integer, agencynum, Titel, created:datetime

Mapping	

Example of the resulting change notification that is sent to the Logic App for a changed record – L2CC_DataEntityName and L2CC_RowState are always sent:


```
{
  "headers": {
    "Connection": "Close",
    "Expect": "100-continue",
    "Host": "prod-05.westeurope.logic.azure.com",
    "Content-Length": "307",
    "Content-Type": "application/json"
  },
  "body": {
    "L2CC_DataEntityName": "AgencyList",
    "L2CC_RowState": "added",
    "ID": 225,
    "agencynum": "00000001",
    "Titel": "Nummer1",
    "created": "13.06.2016 13:46:58"
  }
}
```


Possible values for L2CC_RowState are “added”, “deleted”, and “modified” – the row state can be used to determine which step to execute next in the Logic App.

You can find a detailed example for the whole setup process in Appendix A. See the **Start an Azure Logic Apps Workflow on Local XML Data Changes section.**
Note: The configuration and process to trigger a Flow is quite similar, just the UI is a little bit simpler for Flow.

Layer2 Data Provider for Microsoft Teams

The Layer2 Microsoft Teams provider connects to Webhooks of Microsoft Teams channels so that it can write a "Chat entry" to it after each synchronization. The chat entry contains information about changes in the source data entity that were made since the last synchronization run. It can, for example, be used to regularly notify team members about changes that have been made in a certain local or cloud-based business system like SQL, ERP/CRM, etc., as supported by the Layer2 Cloud Connector. As the chat entries are written into specific channels, it is possible to target change notifications to special interest groups that watch specific channels.

The Layer2 Data Provider for Microsoft Teams cannot be used to sync documents in a Microsoft Teams document library. However, you can view the Team files in SharePoint and use of the Layer2 Data Provider for SharePoint to synchronize files to this library.

Chat entries can only be written to but not read from Microsoft Teams Channels using the Cloud Connector.

The Layer2 Microsoft Teams provider does not manipulate any data but just writes chat entries to a Microsoft Teams channel. Therefore, it is best practice to define the connection as uni-directional (although bi-directional also works).

Chat Entry Reference

The chat entry that will post the changes contains the following elements:

Summary	The summary contains information about the changes that were made since the last synchronization run. The format is: *x items inserted, y items modified and z items removed in <Office 365 data entity name[2]>.*
Section	A section is created per item that has been changed. The title gives information about the data record and the kind of change. The format will be similar to the following: *Item '<record-key>' was added to '<Office 365 data entity name>'.*

[2] No data is changed in the Microsoft Teams (just a chat entry sent), however the team is treated as target data entity and "as-if" data had been changed there during the synchronization.

Section-Details	Each section can contain detailed information about the changes made to the data record. The information is shown as table of name/value pairs, showing the mapped fields (see below) and their new values.

Connection String

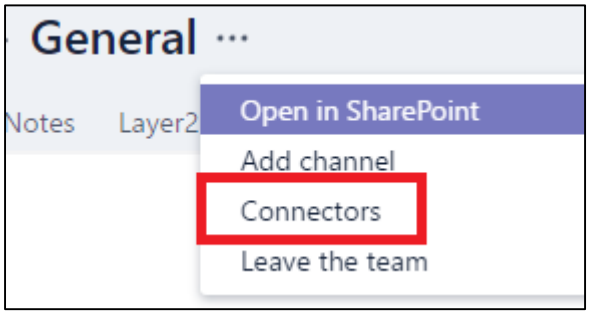
A typical connection string for the Microsoft Teams provider looks like this:

WebHookUrl=<url of the webhook to the Microsoft Team>;

Here is the full list of connection string parameters for this provider:

WebHookUrl

The WebHookUrl is created in the Microsoft Teams channel you want the Cloud Connector to send chat entries to.
When opening the Microsoft Teams channel, you can find an option at the top menu bar called **Connectors**. Click on that to start configuring your Webhook Connector.



You need to create a connector by choosing the "Incoming Webhook" Connector and then click **Add**.

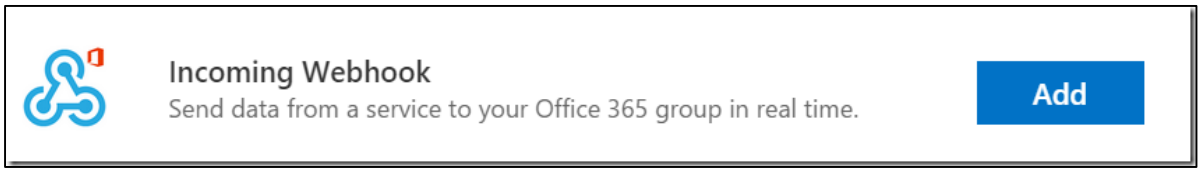


Figure 58 –"Incoming Webhook" Connector in Office 365 Groups

To configure the Incoming Webhook, you need to give it a name and optionally add an image for the Connector. Once that is done you need to click on **Create** to create the Webhook. When it is created you will see an input box with a URL in it. This is the Webhook and you need this URL in the connection string of the Cloud Connector. The URL is unique to this Team and Webhook. If you remove the Webhook, you will disable it.

Note: The Webhook URL will validate even if it is incorrect as it cannot be checked without sending a card to post.

SkipDetails

When this parameter is set to "true", no sections are created, but just the Summary information. This is useful if many data records have been changed (for example, due to a mass update in your business system) and you do not want to get every change detail but just the summary of the synchronization This parameter is optional and defaults to "false".

BatchSize

This parameter is optional and defaults to 10. This means that up to 10 changes are listed in one single chat entry. So in case you have 32 changes, the Cloud Connector would send 4 change chat entries (3 entries with 10 items listed each and another one with 2 items listed).

> **Note**:
> In case you receive an error like this: "*Error while writing data to entity 'Target': Could not write Office 365 card. Details: Target responded with 'Microsoft.Griffin.Connectors.Providers.Common.MessageDelivery.MessageDeliveryException : RequestEntityTooLarge*" during the sync, try to reduce the batch size.

Select Statement

The select statement defines which fields are offered for mapping and, if mapped, what will be shown in the section details.

The format for the select statement is this, where `[FIELDS]` is a comma-separated list of the fields:

```
SELECT [FIELDS]
```

Fields must be explicitly defined; the wildcard * is not supported.

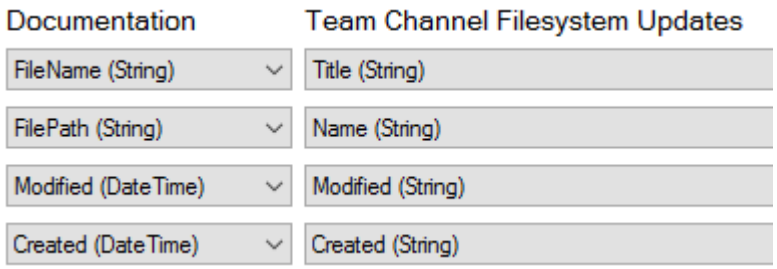
Note: Only changes for fields that are mapped will be tracked. So in case a field of a record in your business system has changed and this field is not mapped (and therefore not part of the synchronization), the change will not be tracked in the chat entry at all.

Primary Key(s)

You need to specify the field(s) that are used to create the Section title. If using a GUID- or counter-type of primary key, it might be a good idea to add another detailed column (for example, "Agencynum, Title") to have human-readable information in the card.

Example:

Synchronization from a file system to Microsoft Teams Channel

Connection string	WebHookUrl=<MyWebHookUrl>;
Select statement	select Name, Title, Created, Modified
Primary Key(s)	Name
Mapping	

Example resulting Chat Entry for the initial synchronization:

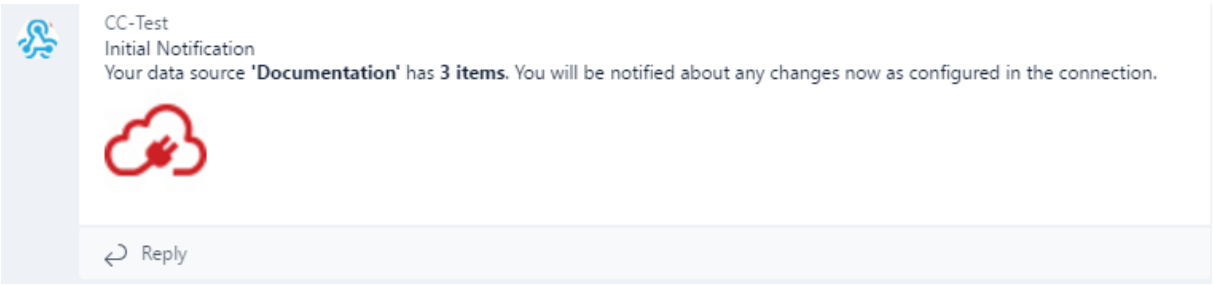


Figure 59 - Example chat entry for initial synchronization

Example resulting Chat Entry for subsequent synchronization:

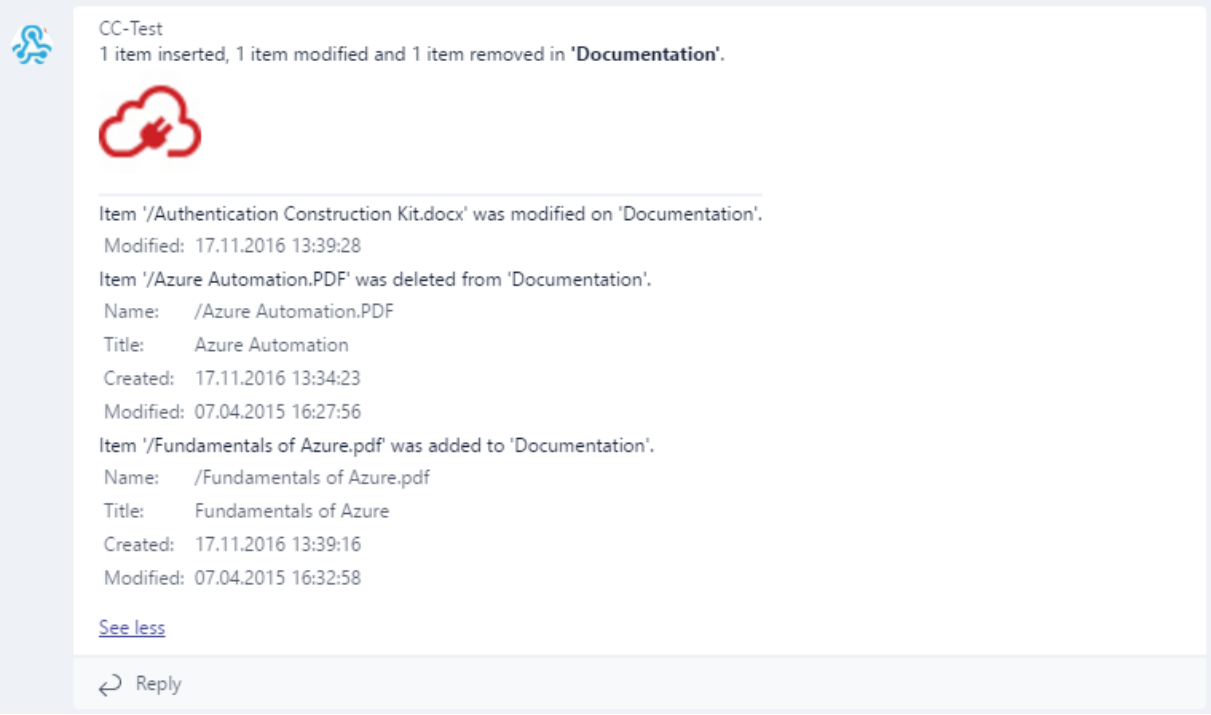


Figure 60 - Example chat entry after a synchronization with changes

Layer2 Data Provider for SOAP Web Services

The Layer2 Data Provider for SOAP Web Services connects to SOAP web services based on their WSDL. It only supports read operations.

Connection String

A typical connection string for the SOAP provider looks like this:

Url=<Url of the web service >;User=<UserName>; Protocol=<SOAP protocol version>;WsdlQuery=<wsdl | singlewsdl>

Additionally, the password needs to be typed into the Connection String or Password field.

With this connection-string, the provider will automatically discover the WSDL by the given URL and connect with the given credentials.

Full list of connection string parameters for the provider:

URL

This parameter is **mandatory**. It is the URL of the web service.

User
The name of the user to authenticate with against the web service. This setting is optional.

Password
The password of the user to authenticate with against the web service. It needs to be typed into the Connection String or Password field. This setting is optional.

Protocol
This setting specifies the SOAP protocol version to use. It is optional. If it isn't set, the protocol SOAP1.2 is used as default. You can determine the protocol from the namespace definitions in the web service's WSDL.

WsdlQuery
This setting specifies how the Wsdl of the web service is queried. It is optional. If it isn't set, the query "?wsdl" is used as default. If "singlewsdl" is set, the query "?singlewsdl" is used to query for the full single-file Wsdl without any external Xsd-references.

Queries
The Layer2 Data Provider for SOAP Web Services supports a slightly different syntax than SQL to query the items from the web service as described below.

```
SELECT [Fields] FROM [Operation]([Parameters])
```

The list of fields can either contain a wildcard (*) or a comma-separated list of property names returned by the web service method. Furthermore, it is possible to rename the fields by using the SQL alias-syntax, i.e. SELECT Companyname AS cmp, would select the contents of property Companyname and populate it as a field named cmp in the result.

The operation part specifies the operation to use for item retrieval.

The parameters part, which is enclosed in parentheses, is optional and contains a comma-separated list of the parameters the operation expects.

Restrictions
The Layer2 Data Provider for SOAP Web Services supports web services only under the following conditions:

- The WSDL has to be compliant to WS-I Basic Profile.
- The web services require either no authentication or windows authentication (username and password). Custom authentication mechanisms (i.e. session-id or token based) are not supported.
- Operation parameters are of simple type (i.e. string, integer, etc.). Parameters of complex types defined in the WSDL or of type "any" are not supported.

- The result returned from the operation does not contain multi-occurrence properties. If it does, these properties will be skipped (omitted) and a warning will be raised.

Authentication

This section describes the details regarding the authentication used by our data providers against various target systems.

Authentication Sequence

Instead of just one authentication method, multiple methods can be used in a sequence. Later methods will use all authentication data provided by former methods. Currently, this feature is only in use with the SapHeader method. The following sample shows how to define a sequence:

Authentication=ADFS,SapHeader

Authentication Construction Kit

The Cloud Connector comes with many pre-built authentication methods that cover the majority of user cases (see the Authentication Methods section for what those are and how to use them). The Authentication Construction Kit is intended for advanced users that have authentication needs not covered under the standard pre-built methods.

With the Authentication Construction Kit feature, any browser-based authentication can be added to the Cloud Connector. All you need is an analysis of the authentication process used by your data source (such as with a Fiddler trace) and the guidance given in this section. Based on the analysis, you can write an XML-based file with a series of actions to mimic the process.

Authentication Directory

This folder can be found under the Layer2 Cloud Connector Data Directory. It contains authentication files, a folder for regular expression files, and a template folder for reusable XML-fragments. If access to the file system is not possible, the default authentication configurations will be used, but the authentications cannot be customized.

Authentication File

These files describe the actions taken to mimic the process necessary to authenticate against a target system.

```
<?xml version="1.0" encoding="utf-8" ?>
<Authentication>
  <Actions>
    <Store />
    <Request />
    <Log />
    ...
  </Actions>
```

```
  <RegularExpressions>
    <RegularExpression />
    ...
  </RegularExpressions>
</Authentication>
```

Figure 61 – Example authentication file

Regular expressions can either be defined inline, inside of an authentication file, or globally, in regular expression files in the "RegularExpressions" directory. Using an Inline definitions of regular expressions increases the portability because the authentication files can be distributed without any regular expression files. However, defining regular expressions globally enhances reuse. For global regular expressions, see the Regular Expression File section below.

Regular Expression File

These files are used to store necessary global regular expression definitions.

```
<?xml version="1.0" encoding="utf-8" ?>
<RegularExpressions>
  <RegularExpression />
  ...
</RegularExpressions>
```

Figure 62 - Example regular expression file

Regular expression nodes need a name and a regular expression, which contains at least one match-group. If an expression contains XML-specific characters, it has to be encased in a CDATA block.

```
<RegularExpression Name="Url">
  <Expression>
<![CDATA[^(?<ProtocolHost>(?<Protocol>https?:\/\/)?(?<Host>[\da-z\.-
]+)?)?(?<Path>\/[\/\w \.-]*)*]]>
  </Expression>
</RegularExpression>
```

Figure 63 - Example regular expression definition

Actions

All attributes and sub-nodes of actions support placeholders. If a value contains XML-specific characters, it has to be encased in a CDATA block.

Condition Attribute

All actions have a common attribute called "Condition". Depending on the given value, the action is executed or not.

No condition	`Action is executed`
Empty	`Action is NOT executed`
Non-empty	`Action is executed`

Empty with exclamation mark	`Action is executed`
Non-empty with exclamation mark	`Action is NOT executed`

```
<Log Message="This message will be logged." />
```
Figure 64 – Example for no condition

```
<Log Condition="[NonEmptyVariable]" Message="This message will be
logged." />
```
Figure 65 – Example for non-empty condition

```
<Log Condition="[NonExistingVariable|]" Message="This message will NOT be
logged." />
```
Figure 66 – Example for empty condition

```
<Log Condition="![NonEmptyVariable]" Message="This message will NOT be
logged." />
```
Figure 67 – Example for non-empty condition with exclamation mark

```
<Log Condition="![NonExistingVariable|]" Message="This message will be
logged." />
```
Figure 68 – Example for empty condition with exclamation mark

<Request>

This sends out a web request and stores the response as “LastResponse”. It also adds all received cookies to “Cookies”. All requests need a URL as a target. By default, the request will be executed using the HTTP/POST verb if a body is set and HTTP/GET if no <Body> is specified. The verb can also explicitly be specified using the “Method”-property. It is also possible to define headers for the request as shown in the example.

```
<Request>
  <Headers>
    <Header Name="Content-Type" Value="application/soap+xml; charset=utf-
8" />
  </Headers>
  <Url>[Settings.Url]</Url>
  <Body>
<![CDATA[UserName=[Settings.User]&Password=[Settings.Password]&AuthMethod
=FormsAuthentication]]>
  </Body>
</Request>
```
Figure 69 - Example <Request> action

AutoRedirect

The request automatically tries to follow redirects, but this behavior can be disabled by setting “AutoRedirect” to “false” (as seen in the sample below).

```
<Request AutoRedirect="false">
```

```
  <Url>[Settings.Url]</Url>
</Request>
```

Figure 70 - Example of a <Request> without auto-direct

NTLM

The request automatically tries to handle NTLM authentication with default credentials. You can specify the credentials (User, Password, and Domain [optional]) via sub-nodes (as shown in the sample below) or disable this feature completely by setting "HandleNTLM" to "false".

```
<Request User="[Settings.User]" Password="[Settings.Password]"
Domain="myDomain">
  <Url>[Settings.Url]</Url>
</Request>
```

Figure 71 - Example <Request> with custom credentials for NTLM authentication

<Store>

This action needs a name and a value. The value is stored as a named variable for later usage.

```
<Store Name="Url_ProtocolHost" Value="[Settings.Url:Url.ProtocolHost]" />
```

Figure 72 - Example <Store> action

Some special variables are automatically created for specific uses:

- "Settings" holds all connection string parameters
- "Output" holds "Output.Headers" and "Output.Cookies" which are used to determine the headers and cookies used after the authentication process is done
- "CrmIdentifier" helps identifying the CRM realm for some authentications
- "AuthenticationData" holds data from previous authentications in the current sequence
- "Persisted" holds values that should be saved longer than one authentication execution

<Log>

This action writes to the Cloud Connector log and is mostly for debugging purposes. It requires a message and optionally a log level, which can have one of the following values: Trace, Debug, Info, Warn, Error, or Fatal.

```
<Log LogLevel="Trace" Message="Step 3 - MS authentication via SAML token
from [LastResponse.Url]" />
```

Figure 73 - Example <Log> action

<Browse>

This action was primarily implemented for authentication processes that require input via Browser and then send a response to a callback (such as OAuth). It opens the given URL in a browser-like popup and starts a HTTP listener on the given port.

```
<Browse Url="http://mysite.mydomain" Port="8910" />
```

Figure 74 - Example <Browse> action

Event Handlers

Event handlers are an optional part of the authentication file. They will execute their actions whenever the associated event is triggered.

```
<?xml version="1.0" encoding="utf-8" ?>
<Authentication>
  <EventHandlers>
    <EventHandler Event="BeforeAction" Action="Request">
      <Log Message="Next action is a request." />
    </EventHandler>
  </EventHandlers>
  <Actions>
    <Store />
    <Request />
    <Log />
    ...
  </Actions>
  <RegularExpressions>
    <RegularExpression />
    ...
  </RegularExpressions>
</Authentication>
```

Figure 75 – Example file with an event handler which is logging a message before each request action

Scope Identification

Each authentication needs a scope identifier for internal purposes such as caching. If there is no scope identification block in an authentication file, the default scope identifier is used.

```
<?xml version="1.0" encoding="utf-8" ?>
<Authentication>
  <ScopeIdentification>
    <Store Name="Output.ScopeIdentifier" Value="[Settings.User];Custom" />
  </ScopeIdentification>
  <Actions>
    <Store />
    <Request />
    <Log />
    ...
  </Actions>
  <RegularExpressions>
    <RegularExpression />
    ...
  </RegularExpressions>
</Authentication>
```

Figure 76 – Example file with a custom scope identifier

Placeholders

These follow a simple pattern: [Name-Transformation-Default] or [Name-Default-Transformation]

The placeholder always starts with the opening placeholder token '[', then always expects a name of a variable. After that a default value (indicated by '|'), one of several transformations can be specified and the placeholder ends with ']'. Default value and transformations are both optional.

```
[LastResponse.Body:myRegularExpression|http://www.sample.com]
```

Figure 77 - Example Placeholder. Returns the action of the first form in the body of the last response or 'http://www.sample.com' if that cannot be done

Name

This can either be something previously stored with the store action or one of these default variables: "Settings" (connection string components, e.g. Settings.User), "LastResponse" (e.g. LastResponse.Headers.Location) or "Cookies". LastResponse and Cookies are only available after the first request action.

Default Value

This is indicated by '|' and specifies a value that is used if the variable or transformation before it returns nothing.

Transformations

These are indicated by various characters, but they all perform a change the value.

Trim

This is indicated by ';Trim' and removes any unimportant leading or trailing whitespace characters.

Example:

```
[Settings.Url;Trim]
```

Url Decode

This is indicated by ';UrlDecode' and will decode URL-encoded strings.

Example:

```
[LastResponse.Headers.Custom;UrlDecode]
```

Url Encode

This is indicated by ';UrlEncode' and will URL-encode values which might be required by some systems.

Example:

```
[Settings.User;UrlEncode]
```

Regular Expression

This is indicated by ':' followed by the name of a regular expression and optionally the name of a named group inside that expression. These regular expressions are executed with the SingleLine option.

Example of a regular expression transformation with named group:

```
[Settings.Url:Url.ProtocolHost]
```

Example of a regular expression transformation *without* named group:

```
[LastResponse.Body:MyRegularExpression]
```

Extract Input Value

This is indicated by ';extractInputValue' followed by the name or id of the desired input element. As the name suggests, this will return the value of said input.

Example:

```
[LastResponse.Body;extractInputValue.Wresult]
```

Extract Form Action

This is indicated by ';extractFormAction ' followed by the name or id of the desired form element. As the name suggests, this will return the action of said form.

Example:

```
[LastResponse.Body;extractFormAction.Login]
```

Authentication Methods

The Layer2 ADO Providers provide several different methods to authenticate against the system which they access. This section describes the different provided authentication methods. Only the Layer2 Exchange Provider has its own authentication mechanisms (See the Layer2 Data Provider for Exchange section for details).

All authentications implemented with the authentication construction kit have a fallback for compatibility reasons. To use the fallback, you just have to add a "_legacy" to the method name. For example: ADFS_legacy

IntegratedWindows

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

With this authentication method, the Layer2 Cloud Connector uses the current account to log in on the target server which must have been configured to allow integrated authentication. If the

provider is used by the Windows service in the context of an automatic background synchronization, the user that will be used to perform integrated authentication is the service account which is by default, usually a system account like NetworkService. In this case, either the Windows-authentication method should be used or the Windows service needs to be configured to run in the context of a specific account which has the required permissions on the target server.

Windows

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider, FileSystem-Provider, SOAP-Provider)

This authentication method is using a Windows domain-account to authenticate against the target server.

Related connection-string settings:

User

This part of the connection string specifies the username for the account which is used to authenticate in the format: DOMAIN\username.

Password

This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

Anonymous

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider, SOAP-Provider)

If the target server is configured to support it, this method can be used to connect to the server without any authentication.

Office365

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

This is the default authentication method to access Microsoft Office 365 instances and should work in most cases. When using the .NET 4 edition of the Cloud Connector this will also work for ADFS-connected SharePoint sites.

Related connection-string settings:

User

This part of the connection string specifies the username for the account which is used to authenticate.

Password

This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

SecureTokenService

(Only used in the legacy version)

This setting is optional and should not be specified in most cases. It defines the URL of the secure token service which is used for authentication. In most cases, this should be https://login.microsoftonline.com/extSTS.srf, which is the default.

SignInUrl

(Only used in the legacy version)

This setting is optional and should not be specified in most cases. It is the site collection relative URL which is used to sign in after the authentication token has been retrieved from the secure token service. If omitted, it will by default be /_forms/default.aspx?wa=wsignin1.0.

Realm

(Only used in the legacy version)

This setting needs to be specified, if the URL which is used to access the SharePoint Online instance is not the default URL. SharePoint Online default URLs have the format https://myCompany.sharepoint.com. This URL is used in two different contexts:

First it is used to identify the SharePoint instance to the secure token server (STS), in this context, the URL is called a Realm.

Second, it is used to locate and access the SharePoint instance, for example in a browser, as a normal URL. If a different URL then the SharePoint Online default URL has been established to access the SharePoint Online instance, the URL will be, for example, https://mySharepoint.myCompany.com, but the realm will still be https://myCompany.sharepoint.com. In this case, the Layer2 Cloud Connector will no longer be able to infer the realm from the URL and the realm would need to be defined explicitly through this setting.

Office365viaGoDaddy

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

This is a custom variation of the Office365 authentication for those who are using godaddy.com to authenticate against their SharePoint Online.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

OnlineUser

This part of the connection string specifies the username for the account which is used to authenticate.

Password
This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

LandingPage
This setting is optional and should not be specified in most cases. It defines the URL to the login form or landing page for non-authenticated users, if it is not possible to detect that automatically.

Office365viaOkta

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

This is a custom variation of the Office365 authentication for those who are using okta.com to authenticate against their SharePoint Online.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

OnlineUser
This part of the connection string specifies the username for the account which is used to authenticate.

Password
This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

LandingPage
This setting is optional and should not be specified in most cases. It defines the URL to the login form or landing page for non-authenticated users, if it is not possible to detect that automatically.

Office365viaPing

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

This is a custom variation of the Office365 authentication for those who are using the SSO service "Ping" to authenticate against their SharePoint Online.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

OnlineUser
This part of the connection string specifies the username that helps SharePoint Online identifying you as a Ping user. You would use this username on the Microsoft Online login page to be redirected to Ping.

User
This defines the username of the account used to login via Ping.

Password
This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

LandingPage
This setting is optional and should not be specified in most cases. It defines the URL to the login form or landing page for non-authenticated users, if it is not possible to detect that automatically.

Office365withSPRedirect

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

This is a custom variation of the Office365 authentication for cases where the authentication process is redirected to a second SharePoint before receiving authentication cookies from the target SharePoint. A good indication of this scenario is that the login page (login.microsoftonline.com) has a different SharePoint as “wreply” argument.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

User
This part of the connection string specifies the username for the account which is used to authenticate.

Password
This defines the password for the account which is used to authenticate. It needs to be typed into the separate Password field.

LandingPage
This setting is optional and should not be specified in most cases. It defines the URL to the login form or landing page for non-authenticated users, if it is not possible to detect that automatically.

OAuth2

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

This is a generic authentication for the OAuth2 standard. It requires a registered app in the target system. The details may vary, but when asked for a redirect URL, it always needs to be http://localhost:myPort (for example: http://localhost:8910), where myPort is an available port on the system that is executing this authentication method.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

ClientId
This parameter is mandatory and is used to identify the registered app in the target system.

ClientSecret
If the registered app provides a client secret, this is mandatory. Otherwise, this is irrelevant.

Scope
Depending on the target system, this is mandatory or irrelevant. It is used to state the requested permissions.

Example:

Scope=https://www.googleapis.com/auth/drive https://www.googleapis.com/auth/calendar;

AuthorizeEndpoint
This mandatory parameter has to be set to the target system URL that is handling the first part of an OAuth2 token request.

Example: https://login.microsoftonline.com/common/oauth2/authorize

TokenEndpoint
This mandatory parameter has to be set to the target system URL that is handling the second part and refreshing of an OAuth2 token request.

Example: https://login.microsoftonline.com/common/oauth2/token

Port
If the app registration asked for a redirect URL, this needs to be set to the same port. Otherwise, this is optional and will be set to an available port by default.

OAuth2AzureAD

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

This is a variation of the OAuth2 method, optimized for the Azure Active Directory.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

ClientId
This parameter is mandatory and is used to identify the registered app in the Azure Active Directory.

Port

As the Azure Active Directory app registration requires a redirect URL, this parameter is mandatory and has to be set to the name port as in the redirect URL.

ResourceUri

The Azure Active Directory uses this parameter to identify the target application of the access token.

Example (OneDrive for Business): https://layer2-my.sharepoint.com/

AuthorizeEndpoint

This parameter is optional and only has to be given if the Azure Active Directory does not use the default endpoints. This endpoint handles the first part of an access token request.

Example: https://login.microsoftonline.com/common/oauth2/authorize

TokenEndpoint

This parameter is optional and only has to be given if the Azure Active Directory does not use the default endpoints. This endpoint handles the second and refreshing parts of an access token request.

Example: https://login.microsoftonline.com/common/oauth2/authorize

AzureSP

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

This is an authentication for Azure-hosted On-Premises SharePoint using the Azure Active Directory for credentials management.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

User

This part of the connection string specifies the username for the account which is used to authenticate.

Password

This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

IECookie

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

If an authentication cookie has been created using the Cookie Manager, this authentication method can be configured to authenticate by using the cookies. (See the Cookie Manager section for details)

SharePoint_FBA

(SharePoint-Provider)

This uses the forms-based authentication provided by SharePoint, but won't work if this feature is not enabled on the target system.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

User

This part of the connection string specifies the username which is used to authenticate.

Password

This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

ADFS

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

It is the authentication method for accessing Office365 using ADFS. ADFS is the authentication method to be used when your organization redirects you to a custom login page in a browser when accessing Office 365.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

User

This part of the connection string specifies the local Windows domain username in the format: DOMAIN\Username.

Password

This defines the password of the Windows domain account, which is used to authenticate. It needs to be typed into the Connection String or Password field.

OnlineUser

Specifies the users online ID in the format: userid@company.com.

Office365UserRealm

It is used to query online user ID information. Parameter is optional and should not be specified in most cases. The default value is:

https://login.microsoftonline.com/pp910/GetUserRealm.srf

WsTrustVersion
This setting is optional and used to set the WS-Trust version which defines the message format of the Secure Token Server authentication token. The possible values are “WSTrustFeb2005” or “WSTrust13”. Default value is “WSTrustFeb2005”.

AdfsEndpointUrl
This is used to define the local ADFS server endpoint URL for issuing an ADFS token. By default, it is queried from the service at the Office365UserRealm by the online user ID. This parameter is optional.

ADFSOnPremise

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

This is the authentication method for accessing On-Premises-Sharepoint servers using ADFS.

Related connection-string settings:

User
This part of the connection string specifies the username.

Password
This defines the password of the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

WsTrustVersion
This setting is optional and used to set the WS-Trust version which defines the message format of the Secure Token Server authentication token. The possible values are "WSTrustFeb2005" or "WSTrust13". Default value is "WSTrustFeb2005".

AdfsEndpointUrl
This is used to define the ADFS server endpoint URL for issuing an ADFS token. This settings is mandatory for On-Premises ADFS.

ADFSIntegratedWindows

(OData-Provider, SharePoint-Provider, XML-Provider, RSS-Provider)

It is the authentication method for accessing Office365 using ADFS with current user credentials.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

OnlineUser
Specifies the user’s online ID in the format: userid@company.com.

Office365UserRealm

It is used to query online user id information. Parameter is optional and should not be specified in most cases. Default value is:

https://login.microsoftonline.com/GetUserRealm.srf

WSTRUSTVERSION

This setting is optional and used to set the WS-Trust version which defines the message format of the Secure Token Server authentication token. The possible values are "WSTrustFeb2005" or "WSTrust13". Default value is "WSTrustFeb2005".

ADFSENDPOINTURL

This is used to define the local ADFS server endpoint URL for issuing an ADFS token. By default, it is queried from the service at the Office365UserRealm by the online user id. This parameter is optional.

DynamicsCrmOnline

(OData-Provider, XML-Provider, RSS-Provider)

This is the authentication method for accessing data on a Microsoft Dynamics CRM Online Instance, preferably by using the OData provider, but it can also be used for XML content on the CRM server or RSS feeds.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

User

This part of the connection string specifies the Office365 username for the account which is used to authenticate.

Password

This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

DynamicsCrmOnlineLiveId

(OData-Provider, XML-Provider, RSS-Provider)

This is the authentication method for accessing data on a Microsoft Dynamics CRM Online Instance using Windows Live ID authentication.

Related connection-string settings:

User

This part of the connection string specifies the Windows Live ID which is used to authenticate.

Password
This defines the password for the account which is used to authenticate. It needs to be typed into the separate Password field.

DynamicsCrmAdfs
(OData-Provider, XML-Provider, RSS-Provider)

This is the authentication method for accessing data on a Microsoft Dynamics CRM Online Instance using ADFS authentication.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

User
This part of the connection string specifies the Windows Live ID which is used to authenticate.

Password
This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

Onlineuser
Specifies the users online ID in the format: userid@company.com.

Office365UserRealm
It is used to query online user id information. Parameter is optional and should not be specified in most cases. The default value is:

https://login.microsoftonline.com/GetUserRealm.srf

NextCRMOnline
(OData-Provider, XML-Provider, RSS-Provider)

This is a custom authentication for the NextCRM system, but it also works with various other CRM providers (online, as well as on-premises).

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

User
This part of the connection string specifies the username which is used to authenticate.

Password
This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

NextCRMOnlineIntegrated
(OData-Provider, XML-Provider, RSS-Provider)

This is the integrated version of the NextCRMOnline authentication. It uses the current user instead of connection string settings.

This authentication is implemented with the authentication construction kit.

AdfsOnPremiseNtlm
(OData-Provider, XML-Provider, RSS-Provider)

This is for ADFS On-Premises authentications where an NTLM handshake happens, but instead of being integrated, it does require user/password credentials.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

User
This part of the connection string specifies the username which is used to authenticate.

Password
This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

xRMLive
(OData-Provider, XML-Provider, RSS-Provider)

It is the authentication method for accessing data on a Microsoft Dynamics CRM Online Instance using xRMlive.com as a CRM provider.

Related connection-string settings:

User
This part of the connection string specifies the Windows Live ID which is used to authenticate.

Password
This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

SignInUrl

This is used to request xRM authentication cookies based on the SAML token acquired from the STS. This parameter is optional and should not be specified in most cases. The default value is:

https://auth-2013.xrmlive.com/

Realm

This is the host address used to connect to xRMLive. This parameter is optional and should not be specified in most cases. Default value is acquired automatically.

SecureTokenService

This is used to exchange authentication information with the xRMLiveSecure Token Service (STS). This parameter is optional and should not be specified in most cases. Default value is acquired automatically.

MSOPartner

(SharePoint-Provider)

This is a custom authentication for Microsoft Online partner systems, where the login is handled by a custom login page (e.g. 'login.partner.microsoftonline.cn') instead of login.microsoftonline.com.

Related connection-string settings:

User

This part of the connection string specifies the username which is used to authenticate.

Password

This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

Realm

This parameter is optional and should not be specified in most cases. It is the authentication interface of the SharePoint, where the Microsoft Online security token is converted into rtFa and FedAuth cookies. Default value is:

(Url-Authority) + /_forms/default.aspx?wa=wsignin1.0

RMUnify

(SharePoint-Provider)

This is a custom authentication for RM Unify, which uses a combination of STS-provided SAML-Bearer-Token and other cookie-based tokens.

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

User
This part of the connection string specifies the RM Unify username which is used to authenticate.

Password
This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

PortalUrl
This is a customer-specific URL used to access the RM Unify portal.

Example Value:

http://ccs.rmunify.com

SignInUrl
This is used to provide Microsoft Online with the required information for association with the used SharePoint instance. In most cases it can be acquired by manually logging into the RM Unify portal and searching for a SharePoint tile. ("MySite" for example).

Example Value:
https://login.microsoftonline.com/login.srf?wa=wsignin1.0&wp=MBI_KEY&wreply=https:%2F%2Fcastlecourtcom-my.sharepoint.com%2F_layouts%2FMySite.aspx&whr=castlecourt.com&CBCXT=out

ExactOnline
(OData-Provider, XML-Provider, RSS-Provider)

This is a custom authentication to log into the ERP system Exact Online (exactonline.com).

This authentication is implemented with the authentication construction kit.

Related connection-string settings:

User
This part of the connection string specifies the username which is used to authenticate.

Password
This defines the password for the account which is used to authenticate. It needs to be typed into the Connection String or Password field.

Important – ExactOnline gives different login portals for customers in different regions. The authentication method is set to US by default (start.exactonline.com). If you are using ExactOnline for another region (for example, start.exactonline.nl), you will need to modify the two login site URLs

in the authentication file to the correct URL (see the Authentication Directory section above for more information on where this file is).

SAPHeader

(OData-Provider, XML-Provider, RSS-Provider)

This is an authentication that re-enables writing access to SAP systems after the addition of the cross-site request forgery token. It has to be attached to an existing authentication as it won't work by itself.

This authentication is implemented with the authentication construction kit.

Example:

Authentication=Windows;SAPHeader

AutoRenaming

This is a feature of the Layer2 Data Provider for SharePoint that is enabled by default and can optionally be disabled in the connection string. If enabled, file and folder names that violate SharePoint naming restrictions are automatically renamed so that they can be uploaded.

It is not recommended to change the AutoRenaming settings for a connection between synchronizations as this can lead to duplication of files and folders that have been renamed previously and will now be handled differently.

Renaming includes:

- Escaping forbidden characters, character sequences, prefixes and suffixes for files and folders
- Shortening file names that violate the length restrictions
- Ensuring that the renaming result is still a unique file path

Folders are escaped in a way that allows un-escaping the name back to the original name on the source system. This is required in order to avoid files being copied to wrong folders during synchronization.

Files are just escaped and cannot be un-escaped to the original name. For the synchronization process, the information about the mapping "source-system file name" ⇔ "auto-renamed SharePoint file name" is maintained in the Metabase.

Note: This means losing the Metabase (for example by manually deleting it) will have the following effects:

- Files will be duplicated by the uniqueness check due to the association between un-escaped source and escaped target being lost (the “new” escaped file is not associated with the previously created escaped file)
- Folders will not be duplicated by the uniqueness check, but will throw errors instead as they are not renamed (to prevent folder structure changes)

Escaping of File and Folder Names

Escaping is done for files as well as folders. Which characters and sequences are escaped depends on the SharePoint version.

All SharePoint Versions

File and folder names may not end with any of the following strings:

- .files
- _files
- -Dateien
- -fichiers
- _bestanden
- -filer
- _file
- _archivos
- _arquivos
- _tiedostot
- _pliki
- _soubory
- _elemei
- _ficheiros
- _dosyalar
- _datoteke
- _fitxers
- _failid
- _fails
- _bylos
- _fajlovi
- _fitxategiak

The escaping process for those suffixes differs between folders and files.

Folders

The first two entries are escaped as follows:

-Dateien	+Dateien+
.files	+f+files+

_files	+g+files+

All other suffixes in the list are escaped by omitting the leading character and then adding a '+' at the beginning and the end.

Examples:
A folder named "myFolder.files" will be renamed to "myFolder+f+files+" in SharePoint.
A folder named "myFolder-fichiers" will be renamed to "myFolder+fichiers+" in SharePoint.

Files

For files only the first character of a forbidden suffix is replaced with a + character.

Example: A file "myFile.files" will be renamed to "myFile+files" in SharePoint.

SharePoint Online and SharePoint 2016

Folders

For SharePoint Online the following characters and sequences are escaped in folder names:

Character or Sequence	Escaped to	Sample
#	+1+	"my # folder" ⇔ "my +1+ folder"
%	+2+	"my % folder" ⇔ "my +2+ folder"
*	+4+	"my * folder" ⇔ "my +4+ folder"
\	+7+	"my \ folder" ⇔ "my +7+ folder"
:	+8+	"my : folder" ⇔ "my +8+ folder"
<	+9+	"my < folder" ⇔ "my +9+ folder"
>	+a+	"my > folder" ⇔ "my +a+ folder"
?	+b+	"my ? folder" ⇔ "my +b+ folder"
|	+c+	"my | folder" ⇔ "my +c+ folder"
"	+d+	"my " folder" ⇔ "my +d+ folder"
..	+.+	"my .. folder" ⇔ "my +.+ folder"

+	++	“my + folder” ⇔ “my ++ folder”

Prefix	Escaped to	Sample
<whitespace>	+h+	“ my folder” ⇔ “+h+my folder”
~	+i+	“~my folder” ⇔ “+i+my folder”

So for example, if a folder named “my # folder” is uploaded from a file system to SharePoint, it is named “my +1+ folder” on SharePoint as # is a forbidden character. When reading from SharePoint, the folder name is un-escaped to “my # folder” again.

Files

For files, the forbidden characters and sequences are replaced with a single + character.
Example: A file “my # File.txt” will be renamed to “my + File.txt” in SharePoint.

The only forbidden file prefix (whitespace) is also replaced with a single + character.

As mentioned before, the relation between auto-renamed file name and original file name is maintained in the Metabase as renamed file names cannot be un-escaped.

SharePoint 2010 and SharePoint 2013

Folders

Character or Sequence	Escaped to	Sample
~	+0+	“my ~ folder” ⇔ “my +0+ folder”
#	+1+	“my # folder” ⇔ “my +1+ folder”
%	+2+	“my % folder” ⇔ “my +2+ folder”
&	+3+	“my & folder” ⇔ “my +3+ folder”
*	+4+	“my * folder” ⇔ “my +4+ folder”
{	+5+	“my { folder” ⇔ “my +5+ folder”
}	+6+	“my } folder” ⇔ “my +6+ folder”
\	+7+	“my \ folder” ⇔ “my +7+ folder”
:	+8+	“my : folder” ⇔ “my +8+ folder”

<	+9+	“my < folder” ⇔ “my +9+ folder”
>	+a+	“my > folder” ⇔ “my +a+ folder”
?	+b+	“my ? folder” ⇔ “my +b+ folder”
|	+c+	“my | folder” ⇔ “my +c+ folder”
“	+d+	“my “ folder” ⇔ “my +d+ folder”
..	+.+	“my .. folder” ⇔ “my +.+ folder”
+	++	“my + folder” ⇔ “my ++ folder”

Prefix	Escaped to	Sample
.	+e+	**“.my folder” ⇔ “+e+my folder”**
<whitespace>	+h+	**“ my folder” ⇔ “+h+my folder”**

Files

For files the forbidden characters, sequences and prefixes are replaced with a single + character.

Example: A file “my # File.txt” will be renamed to “my + File.txt” in SharePoint.

As mentioned before, the relation between auto-renamed file name and original file name is maintained in the Metabase as renamed file names cannot be un-escaped.

Shortening of File Names

SharePoint restricts the length of these items:

- Full URL
- File name
- Folder name

These restrictions are version-specific.

In case a folder violates these restrictions, an error is raised so that you can identify the conflicting folder and rename it to a shorter name.

In case a file violates these restrictions, the Cloud Connector shortens the file name step-by-step until it conforms to the restrictions, if possible. If this is not possible, an error is raised so that you can identify the conflicting file and rename it.

The relation between the shortened and the original file name is maintained in the Metabase.

Ensuring a Unique File Name

If a file name needs to be escaped or shortened, the Cloud Connector ensures that the resulting new file name is unique. In case a file with the new file name already exists, it appends a number.

The relation between the modified and the original file name is maintained in the Metabase.

Logging

When a file or folder name has been renamed during the synchronization process, this is logged in the log files with log level *Debug*. Here is an example of what the messages look like:

NamingEnforcer: The folder path '/one +9+ two' was unescaped to '/one < two'.

Support

Online FAQs

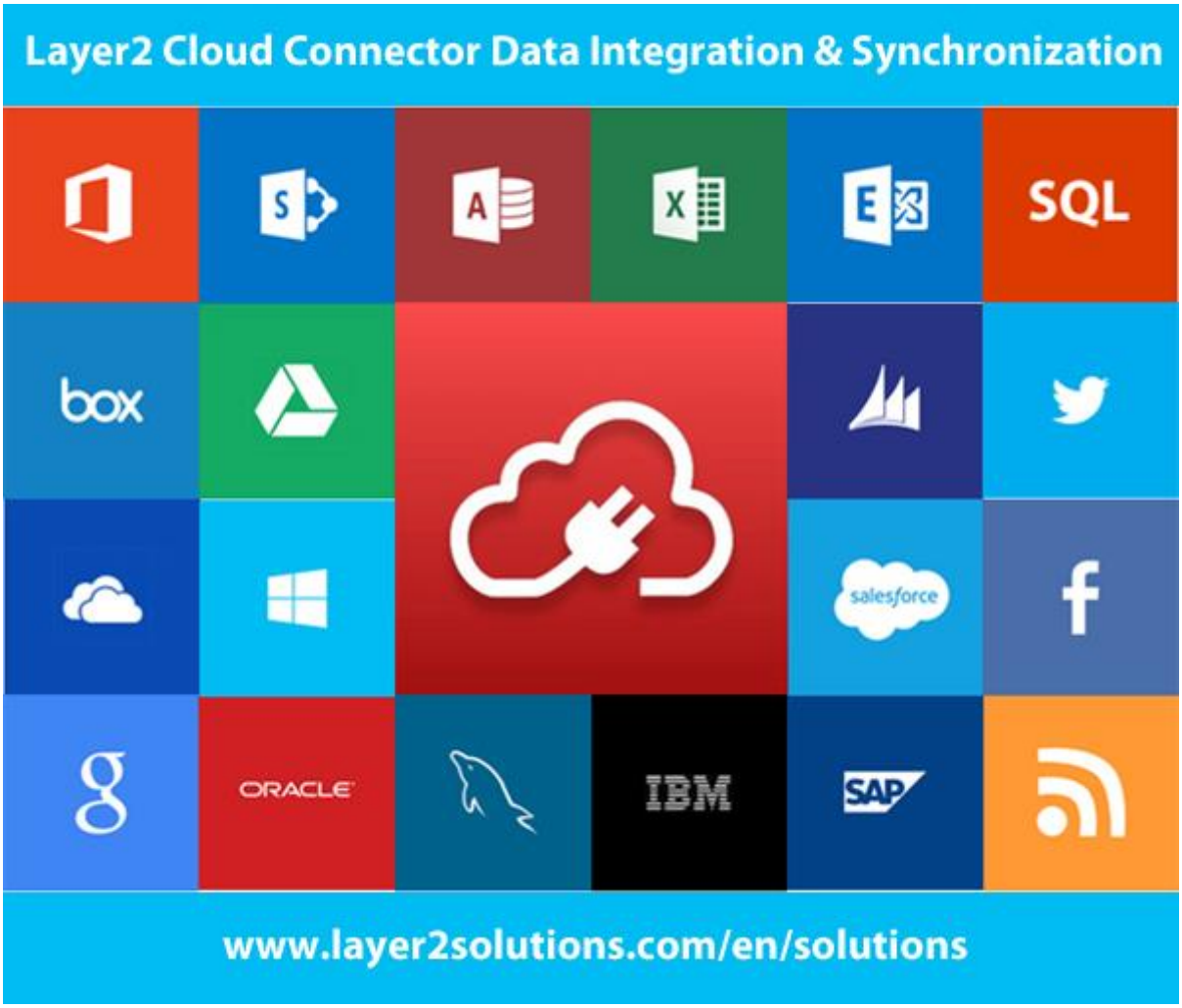
Please find answers to frequently asked questions online:

http://www.layer2solutions.com/en/community/FAQs/general/Pages/default.aspx

http://www.layer2solutions.com/en/community/FAQs/cloud-connector/Pages/default.aspx

Common Scenarios

You will find common scenarios and supported systems/applications online here:
http://www.layer2solutions.com/en/solutions/Pages/default.aspx



Trial

Please visit the product home page to register for trial. You will then receive instructions on how to download and install a Shareware Edition. If you are interested in evaluating the application with full features, please contact sales@layer2solutions.com to receive a time-limited license key.

Ordering

Please visit the product home page to order online. For specialty orders (such as volume packages) please contact sales@layer2solutions.com for a detailed quote.

Software Assurance

License holders who optionally acquire Software Assurance (SA) benefit from future improvements and new features of the licensed product. Software Assurance enables you to migrate your software from a lower-level software version to a higher-level version or from one server to another. It also makes available maintenance, updates and upgrades, and minor and major releases.

The Software Assurance is valid per one license (server) for one year from the date of product license purchase. It can be renewed after expiring. Additional services, which may be required for updating or upgrading, are not included.

Upgrade

When a new version of Layer2 Cloud Connector is released, we will announce the changes in the change log on the version history page. Please take a look at the release notes online before installation. Contact sales@layer2solutions.com to request a license upgrade, if required.

Before running the installer, make sure that the Layer2 Cloud Connector Connection Manager is closed and the background service is stopped. The installer will update the existing components. During the upgrade, all existing configuration-settings and connected data sets will be preserved.

The following may be changed due to the Upgrade:

- The service will be stopped if it was still running and will reset to “Manual” start mode. It may also have any custom logon accounts reset to the local system account.
- The logging will revert to “Warn” level.
- The Nlog.config file may be overwritten.

If you had changed those before, you will need to change them back to your settings after the update is complete.

Migration

In the case that you need to migrate your existing Cloud Connector installation to a new server or newer operating system, please see the instructions below to execute the migration successfully.

Part 1 - Installation and Configuration

Download and install the latest Cloud Connector version on to your new server. If you were using any 3rd-party providers or ODBC configurations, install and configure the necessary drivers. Make sure the provider architectures (32- or64-Bit) match your installed version of the Cloud Connector, as they may have changed with the new installation (see the Data Providers section for more information).

Then contact sales@layer2solutions.com to request a new license file with the new server name and correct version (if upgrading from an older version). Your old license will become invalid when the new one is generated. Install the new license file per the instructions provided by Sales or see Installing a License.

Part 2 - Migrating Connections

For each connection, you will need to migrate the associated connection definition file and the Metabase files to the new server. Those can be found in the “Connections” and “Metabase” directories under one of these paths:

- C:\ProgramData\Layer2 Cloud Connector (for Windows Vista or higher)
- C:\Documents and Settings\All Users\Layer2 Cloud Connector

The path can be read from the environment variable %ALLUSERSPROFILE% (or %PROGRAMDATA% for Windows Vista and higher).

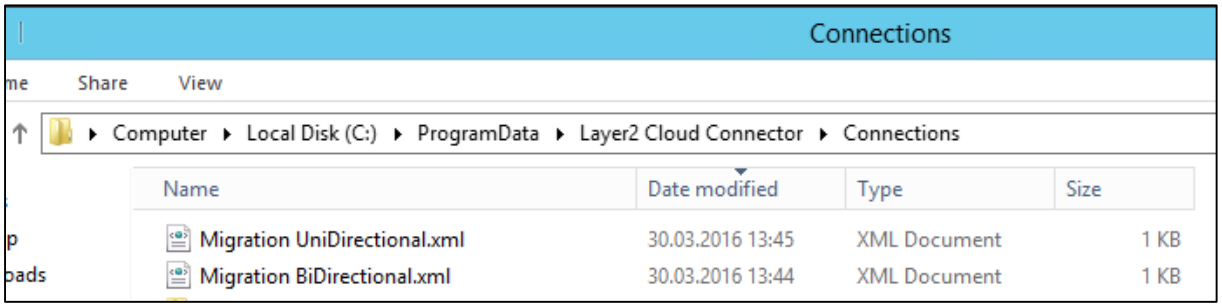


Figure 78 - Example location of connection definitions

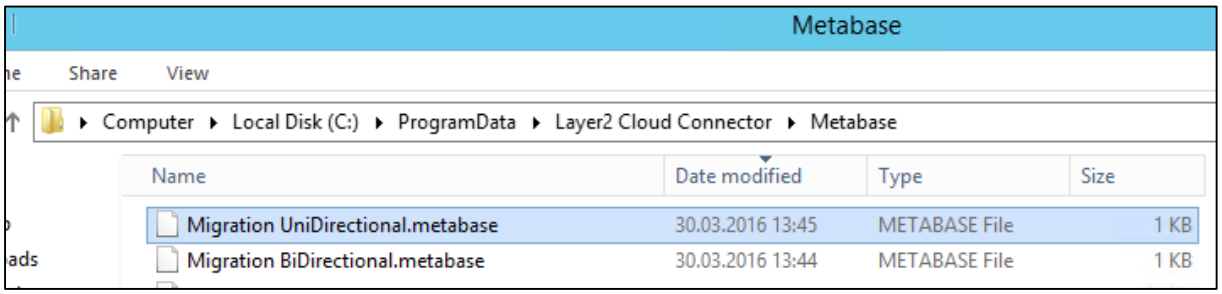


Figure 79 - Example location of Metabase files

You can also move the log and history files, but those are not mandatory for a successful migration.

IMPORTANT - If you had any custom alerts/notifications set up in the Nlog.config file, then that file will also need to be migrated to the Logs folder on the new server to preserve your settings.

Migrating the connection content itself is as simple as copy and pasting the mentioned files to the same folders on your new system.

Part 3 – Validating the Connections

As a last check, go through and make sure that all your connections are still working as intended by launching a manual synchronization run for each migrated connection. Once this has been completed, then you can enable scheduling/start the Layer2 Cloud Connector Service to have the jobs run automatically (if appropriate).

After performing all three parts, your migration is done. If you encounter issues along the way or errors during the connection validation step, contact support@layer2solutions.com for assistance.

Contact

In case of general questions about the Layer2 Cloud Connector, contact sales@layer2solutions.com.
If you have technical issues or errors with a connection, please contact support@layer2solutions.com.

Appendix A – Examples

Start an Azure Logic Apps Workflow on Local XML Data Changes

In this example, we want to start an Azure Logic Apps workflow on local XML data changes.

Note: This sample works similar with a Microsoft Flow instead of an Azure Logic App.

Please copy your Layer2 Cloud Connector sample connection and adapt it as required. The sample XML file that is used here can be found folder *C:\ProgramData\Layer2 Cloud Connector\Sample Data* so that you can easily replay this example.

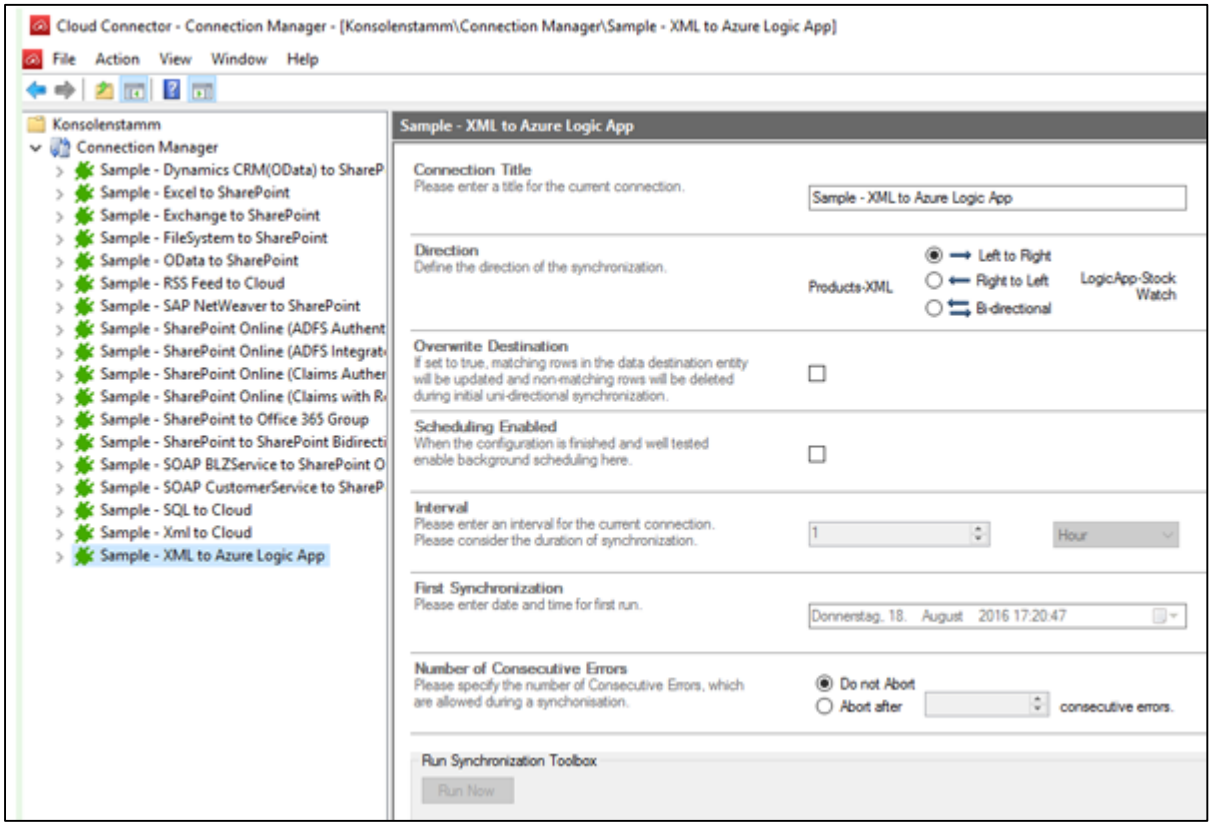


Figure 80- Example connection to adapt for evaluation

Please configure your data source first. In this case, the example data source is a local XML file, but can be any supported data source. The Layer2 Data Provider for XML is used to query an XML file via XPath. Please note the primary key (column with unique values) in the result set. It is required for the data sync. You can also make use of several columns that are unique together, e.g. Col1, Col2, Col3, to make a combination primary key

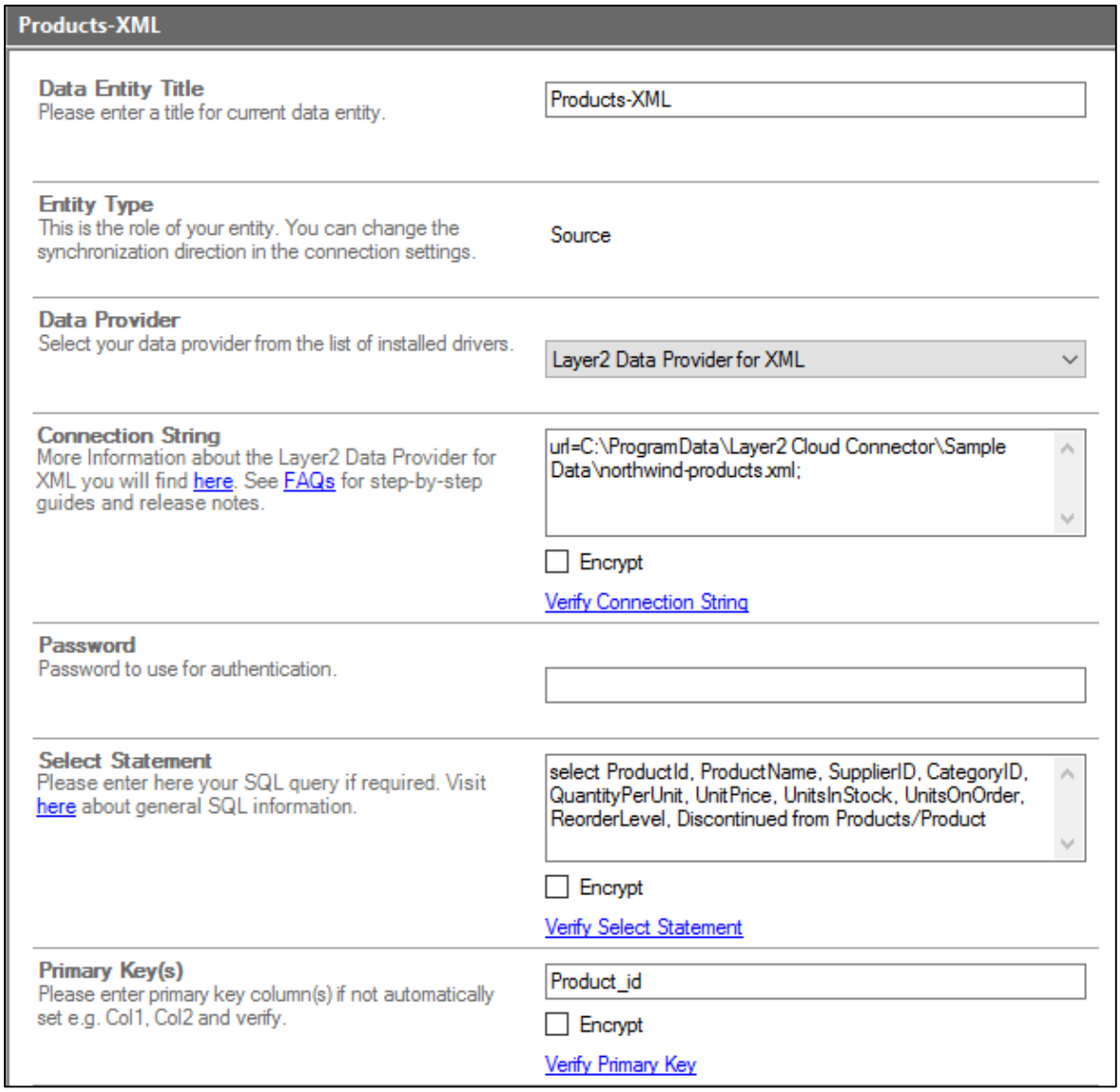


Figure 81 - Example data source using an XML file.

Next configure the Azure Logic Apps data destination. Please select the Layer2 Data Provider for Azure Logic Apps for this. You will need a Web Hook Url for this (see below how to get it). Please also provide a symbolic SQL-like data query for data mapping and to declare data types, and a primary key. The SQL query is also used to generate and display a JSON schema of your data items required for the Azure Logic App.

You can use a notation similar to this:
Select ProductID: integer, ProductName, SupplierID:integer, UnitPrice:float

If you don't add any data type, “string” is assumed by default. Please take a look into the User's

Guide for more data types.

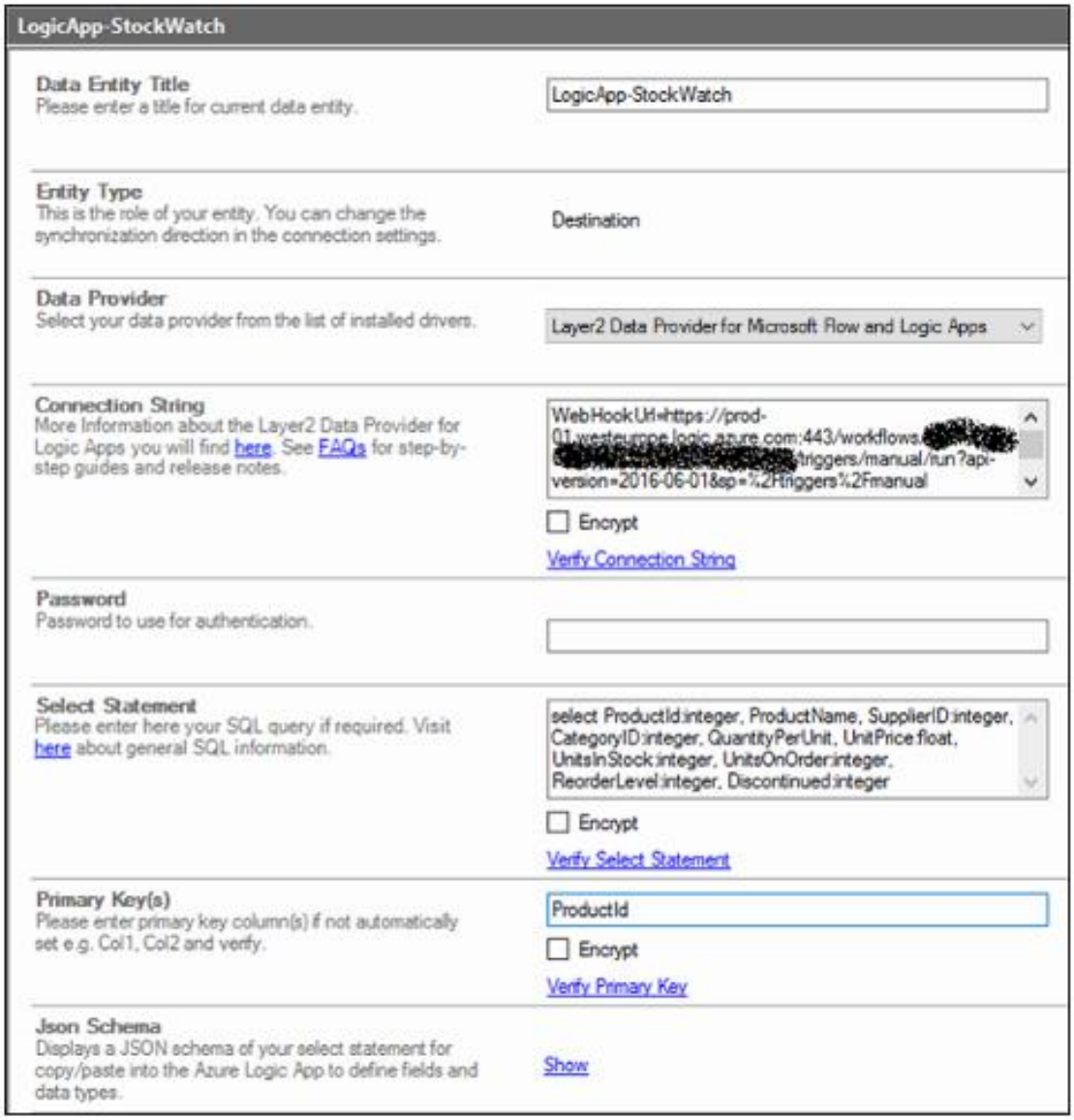


Figure 82 - Example configuration of a Azure Logic App data destination

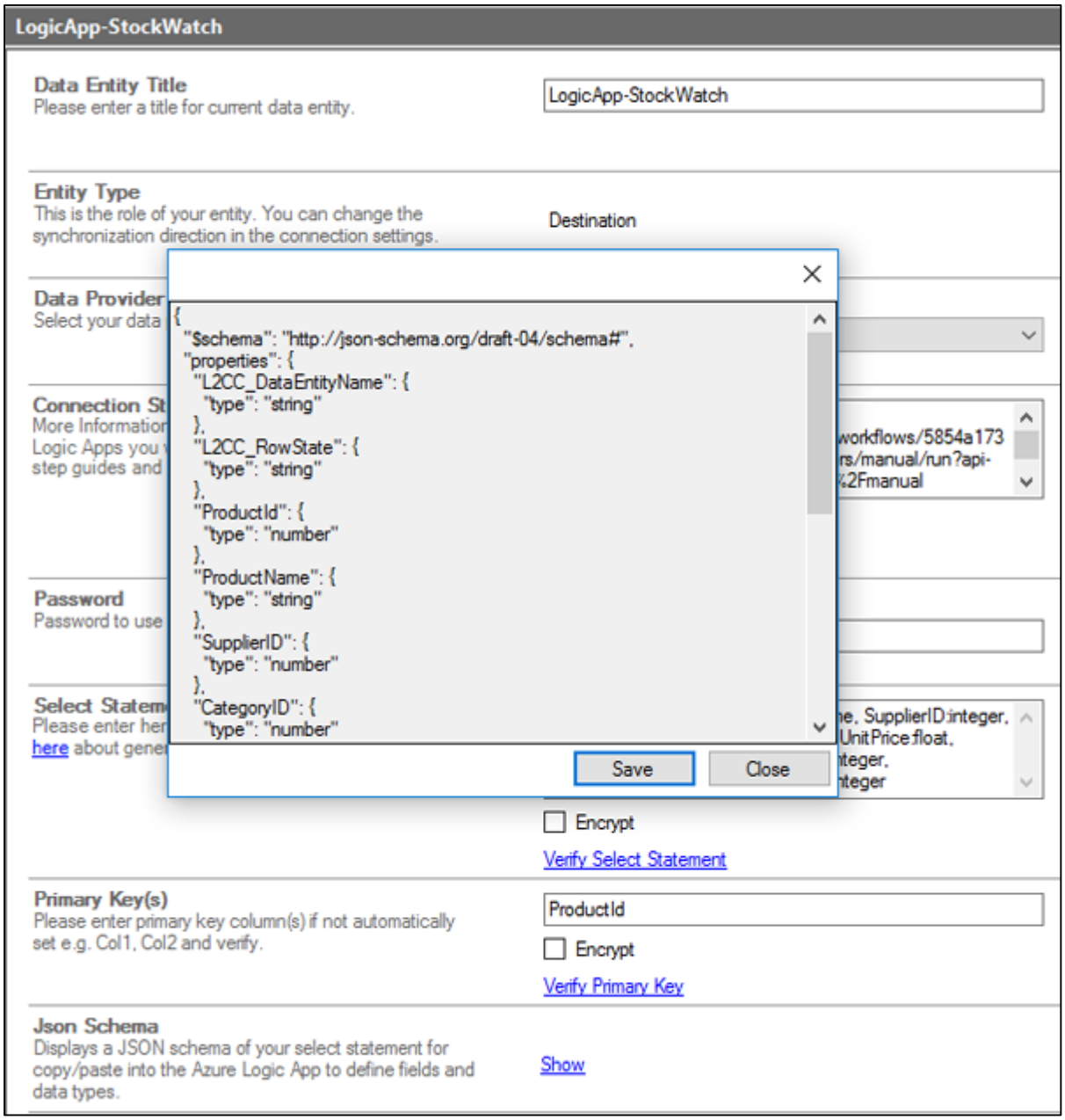


Figure 83 - The Layer2 Cloud Connector provides the JSON schema of the data source as required in Azure Logic Apps to connect to the data source and map fields

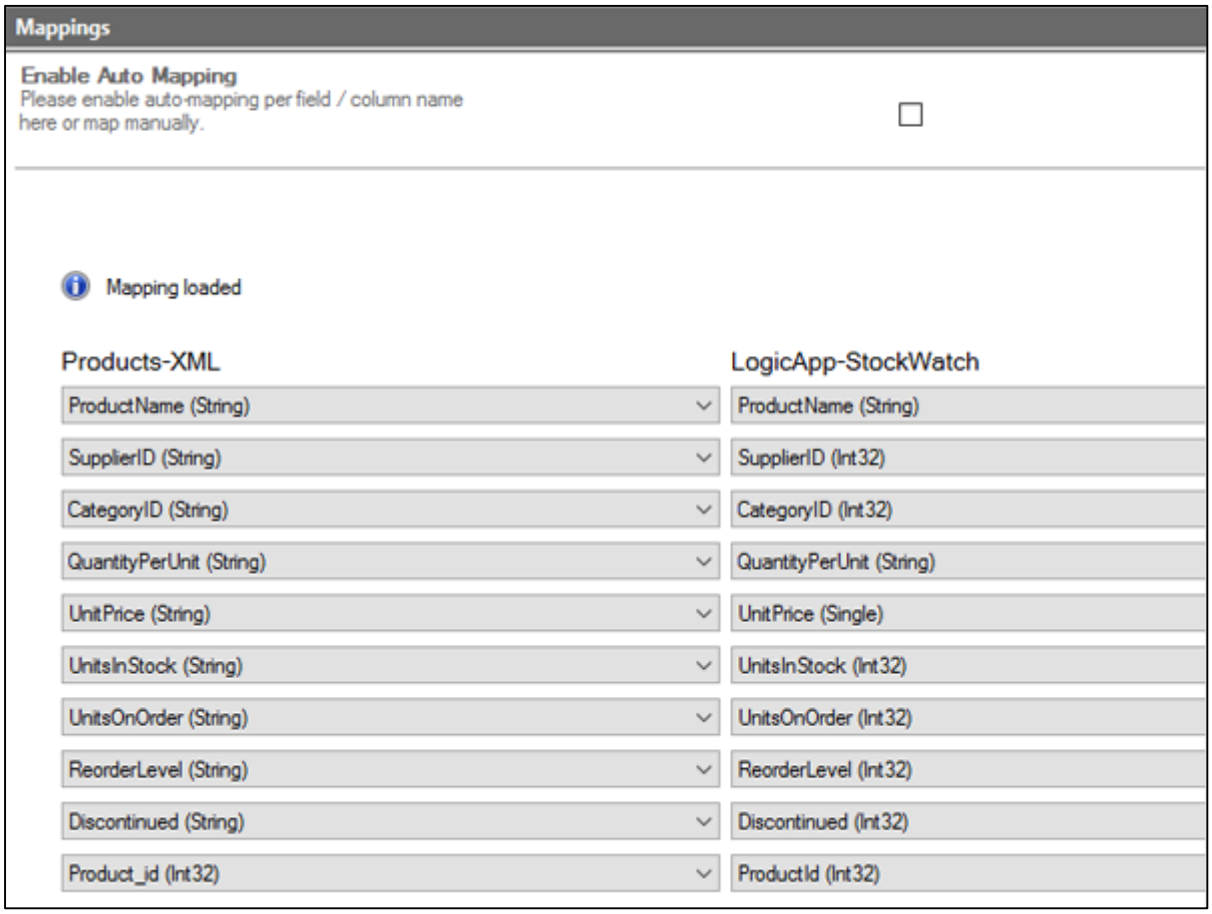


Figure 84 - Example mapping between XML and Logic App data entities

Next create your Azure Logic App as follows:

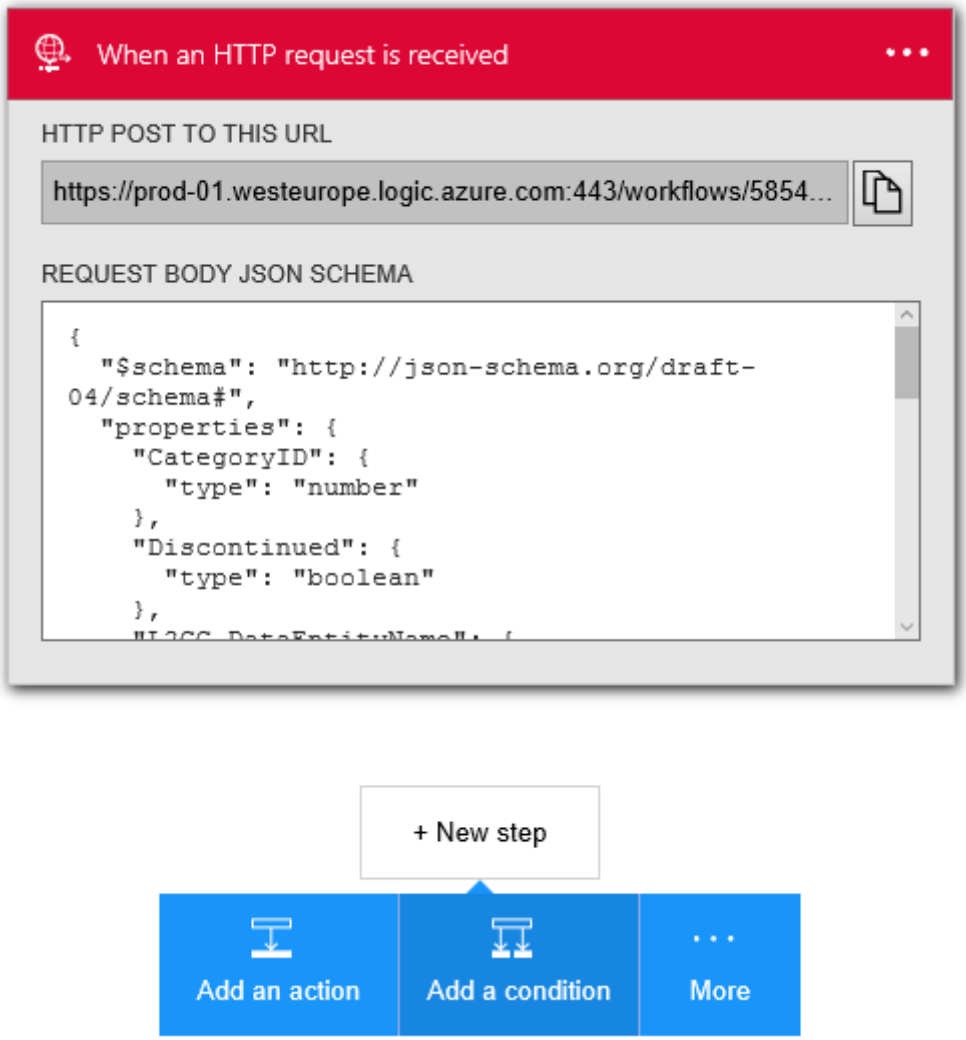


Figure 85 - Configure the Azure Logic Apps HTTP POST end-point

You will find the Webhook URL in the definition of the HTTP Post. Also paste your JSON schema to the Request Body section.

Now you are ready to add some specific logic. This example will send an email notification via Azure Logic App if the number of units in stock is below the reorder level in the local data source.

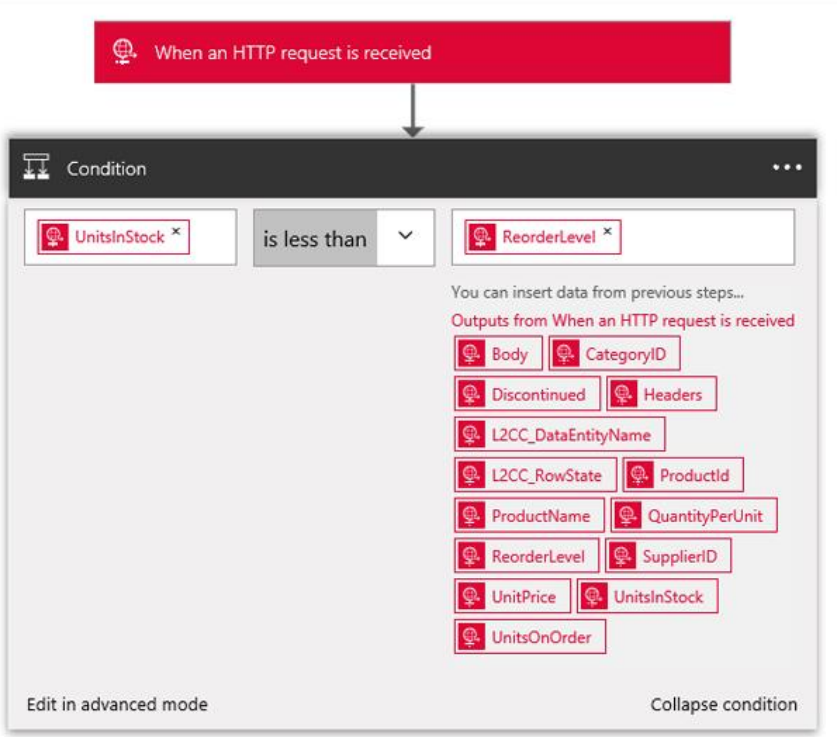


Figure 86 - Apply a logical expression in Azure Logic Apps based on the changed local data record

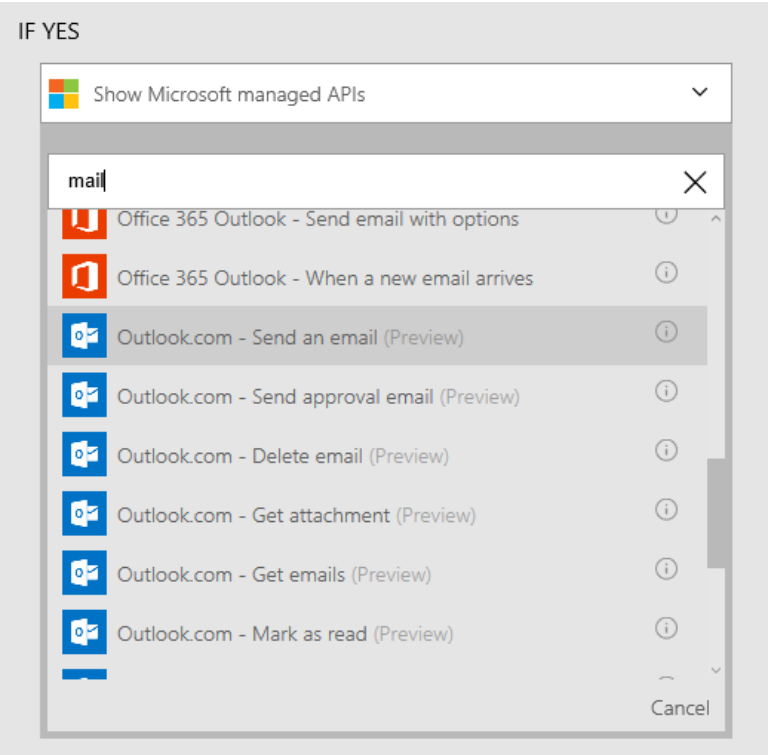


Figure 87 - Select the "Send an email" action from the actions offered by Azure Logic Apps

You can configure the email based on the local data source record as provided by the Layer2 Cloud Connector.

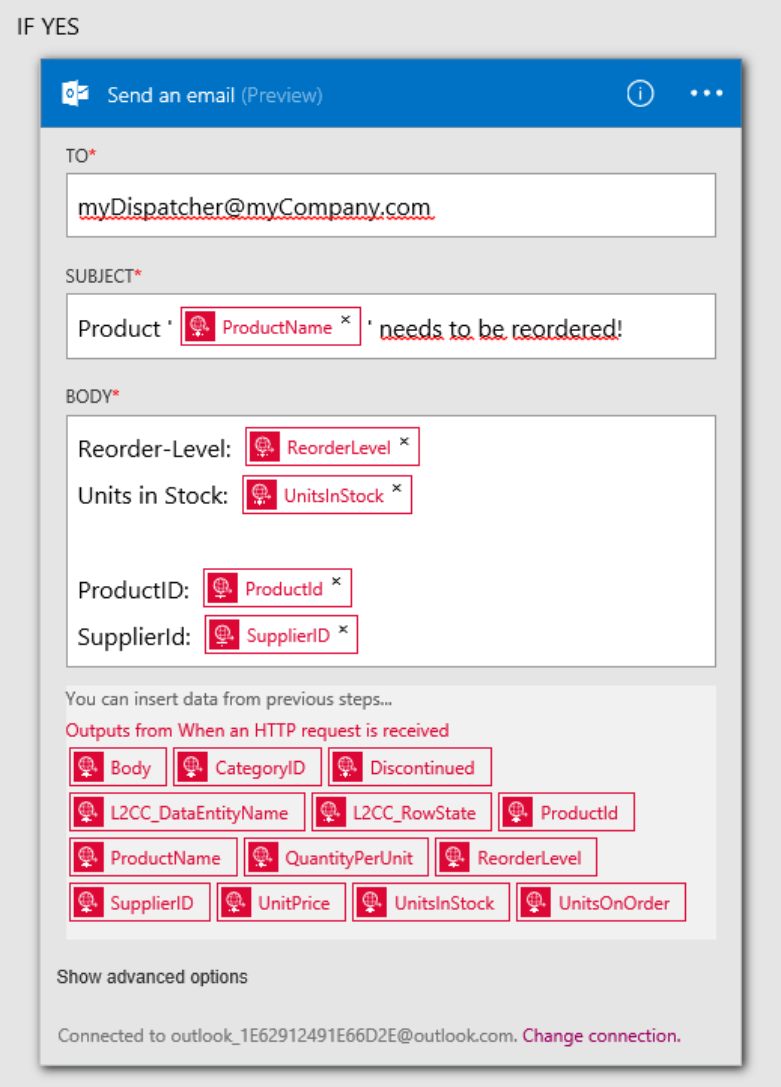


Figure 88 - Example email configuration to send notifications in Azure based on local data changes

You are now ready to run your Layer2 Cloud Connector connection: manually using the **Run Now** button, on-demand using the command line, or scheduled using the Windows service. If your condition is true, you will receive the following email from your connected Azure Logic App:

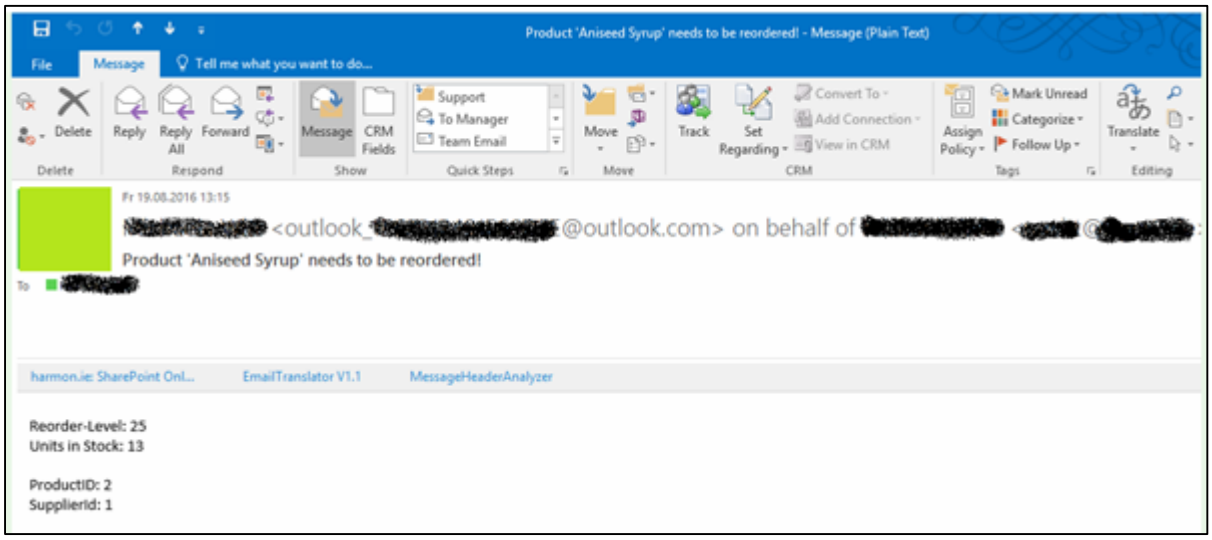


Figure 89 - Example mail received when the condition is "true" for the changes

If you did not receive emails as expected from Azure, check your SPAM folder.