# A new way of working with Azure File Share from X++

Azure File Share is a popular integration medium for Dynamics 365 Finance and Operations (D365FO). Recently, Microsoft upgraded the connection library that allows communication from X++ to it. In this post, I will describe and demonstrate how to implement an integration using the new libraries.

# Original deprecation notice

The original deprecation notice was posted quite some time ago:

Migration from deprecated libraries – WindowsAzure.Storage and Microsoft.Azure.Storage to Azure.Storage.Blobs

<https://learn.microsoft.com/en-us/dynamics365/fin-ops-core/fin-ops/get-started/removed-deprecated-features-platform-updates#migration-from-deprecated-libraries--windowsazurestorage-and-microsoftazurestorage-to-azurestorageblobs>

However, up to 10.0.43, Microsoft didn’t provide a DLL part that works with File Share; this has been fixed in the upcoming 10.0.43 release.

# How to check that you have the correct library

Before starting, you need to check the Reference node in AOT and find the Azure.Storage.Files.Shares reference.

A screenshot of a computer

Description automatically generated

# X++ usage example

External integration https://github.com/TrudAX/XppTools?tab=readme-ov-file#devexternalintegration-submodel is an open-source module that supports various integration scenarios. For working with Azure File Share, it uses the following class that implements all communication logic: DEVIntegAzureStorageConnectionHelper. https://github.com/TrudAX/XppTools/blob/master/DEVTutorial/DEVExternalIntegration/AxClass/DEVIntegAzureStorageConnectionHelper.xml

# Testing integration

To test a new library, I used the following example:

**Business Scenario:** The external party will send us sales orders in JSON format to Azure File Share. We want to import these files into D365FO and create sales orders.

## File share creation

I created a File Share that contains Incoming and Archive folders.

A screenshot of a computer

Description automatically generated

Then, I attached this directory locally (to simplify the test process) and put some files there that contain both correct and incorrect data.

A screenshot of a computer

Description automatically generated

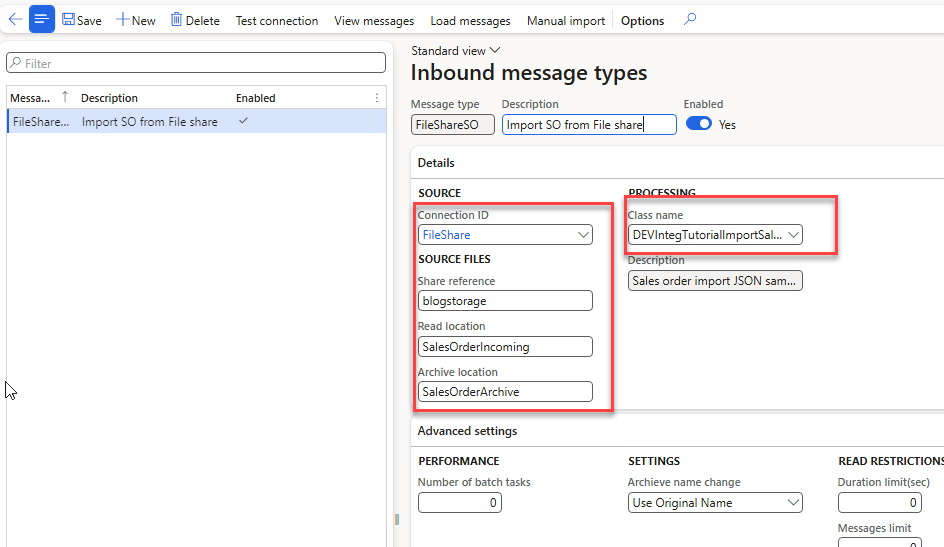
## D365FO setup

In **External integration – Connection type** I created a new link to the created Azure File Share

A screenshot of a computer

Description automatically generated

And then I created a link to this file share in **External integration - Inbound message types**. The settings below mean the system loads files from the SalesOrderIncoming folder and passes the loaded data to the DEVIntegTutorialImportSalesTableJSON class that performs processing.



After running the Inbound message processing, I got three new records in the Sales Orders staging table. Two were processed successfully and contained links to the created Sales Orders. One has an error with a non-existing customer that the user is supposed to analyze and fix.

A screenshot of a computer

Description automatically generated

A more detailed description of the process can be found in the original blog post on How to implement file-based integration in Dynamics 365 FinOps using X++: <https://denistrunin.com/xpptools-fileintegledger/>. Feel free to post any questions or comments