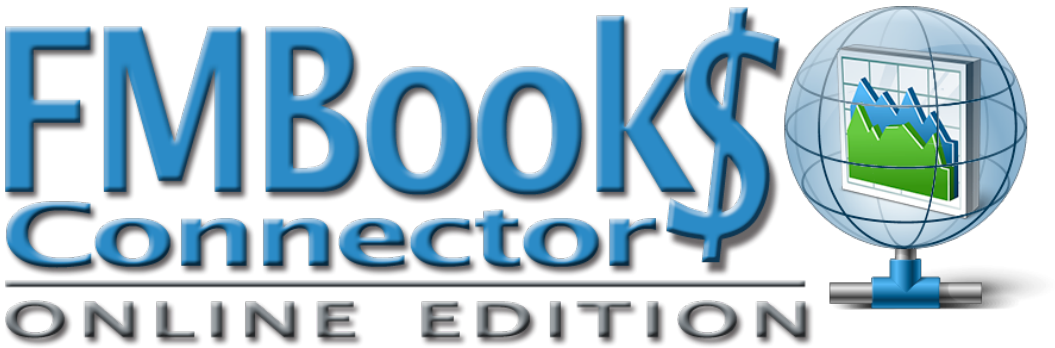
Productive Computing

Creating Efficiency Through Automation



Migration Guide

FM Books Connector to FM Books Connector Online

February 21, 2017

950 Boardwalk, Suite 205, San Marcos, CA 92078 • (760) 510-1200 • www.productivecomputing.com

This migration guide is intended to provide guidance on migrating your scripts from using the FM Books Connector plug-in integrated to communicate with QuickBooks Online ("QBOE") to using the FM Books Connector Online plug-in features. This migration guide only applies if your solution is communicating with an Intuit® QuickBooks Online company; integrations of the FM Books Connector with Intuit® QuickBooks desktop editions do not require migration.

System Requirements

The FM Books Connector Online plug-in requires the following client system components:

- .NET 4.5 Framework (Windows only)
- Windows 7, 8, 8.1, or 10
- Mac OS 10.10 - 10.12
- FileMaker Pro 13- 15 (32-bit) and FileMaker 14- 15 (64-bit)

The plug-in will no longer require the Visual C++ 2008 Redistributable Package.

Authorization

Instead of requiring the Intuit® QuickBooks Online Connector application, FM Books Connector Online will use a new session management format called "OAuth". There are no installers required to use OAuth. As of Version 2, the plug-in uses the PCQO_BeginSession function to perform the OAuth process as follows:

> 1) PCQO_BeginSession launches an in-app web browser window that brings up the Intuit authorization page. The user is then directed to log in to the desired company and authorize the FM Books Connector Online plug-in to exchange data with the company.
>
> 2) Once authorization has been granted, the plug-in closes the in-app browser window and then tests the connection "keys" it received, verifying that it can exchange data with the company.
>
> 3) If verification is successful, the PCQO_BeginSession function returns "0" and the plug-in is now able to execute requests and process responses.

Calling PCQO_EndSession will terminate the authorized session, and communicating with the company at that point will require acquiring authorization once again via PCQO_BeginSession. Sessions are unique to each call of PCQO_BeginSession; a session generated by one PCQO_BeginSession request is different from a session generated by another PCQO_BeginSession request, and if connecting to the same company, the second call invalidates the first.

There is a limitation on connection that prevents a given application (in this case, the FM Books Connector Online plug-in) from connecting to a company more than once. For example, this could be the case if there are more than one accountants or financial experts in a company that need to use the plug-in to push invoice and/or customer data to the QuickBooks Online Company. To get around this, the FM Books Connector Online plug-in comes equipped with a pair of functions to handle sharing session information with other instances of the plug-in: PCQO_SSaveSessionInfo and PCQO_SLoadSessionInfo(SessionInfo).

- PCQO_SSaveSessionInfo will generate an encoded text string that contains everything the plug-in knows about the connection it currently has with QuickBooks Online. This is best called

immediately after a successful PCQO_BeginSession call, and saved to a field in the solution's persistent Preferences section, if applicable.

- PCQO_SLoadSessionInfo takes the encoded text string from PCQO_SSaveSessionInfo, decodes it, and recreates the session connection to the QuickBooks Online. The function also tests the connection, similar to the verification process of PCQO_BeginSession, and if the verification is successful, the function returns "0".

This authorization process is the replacement for the standard QuickBooks Online Connector authorization process that the FM Books Connector uses (which is initiated by calling "PCQB_BeginSession("QBOE" ; "")").

Requests and Responses

Requests and responses in the FM Books Connector Online plug-in handle exactly the same as the FM Books Connector plug-in: first build a request using the various "Rq" functions, execute the request with "PCQO_RqExecute", and then open and process the response using the various "Rs" functions. For specific details on how each function works, please refer to the FM Books Connector Online Functions Guide.

Functions

Due to the inherent difference in the underlying structures of the FM Books Connector and FM Books Connector Online plug-ins, the plug-in functions will not map when replacing the FM Books Connector plug-in with the FM Books Connector Online plug-in. We advise to keep the original FM Books Connector plug-in installed while performing the migration, so that your scripting will retain the original plug-in function calls, and reference those when replacing the functions with those from the FM Books Connector Online.

The following table determines a mapping of functions from FM Books Connector to FM Books Connector Online (functions in **green** are new to the FM Books Connector Online plug-in, functions in **blue** are functions that are only found in FM Books Connector):

FM Books Connector	FM Books Connector Online
PCQB_GetOperatingMode	PCQO_GetOperatingMode
PCQB_Register(ServerName ; ServerPort ; ServerPage ; LicenseID)	PCQO_Register(ServerName ; ServerPort ; ServerPage ; LicenseID)
PCQB_Version(Type)	PCQO_Version(Type)
PCQB_BeginSession(CompanyFile ; ShareMode ; optHideFMWindow)	PCQO_BeginSession
PCQB_EndSession	PCQO_EndSession
PCQB_Browse	
PCQB_GetCurrentFileName	
PCQB_GetCurrentFileVersion	
PCQB_GetCurrentQBVersion	
PCQB_RqNew(MessageType ; optAttributes)	PCQO_RqNew(Operation ; EntityType)

PCQB_RqAddFieldWithValue(QBFieldName ; Value ; optAttributes)	PCQO_RqAddFieldWithValue(FieldName ; Value)
PCQB_RqAddRelatedRecord(Type ; optValue ; optAttributes)	PCQO_RqAddRelatedRecord(Type)
PCQB_RqCloseRelatedRecord	PCQO_RqCloseRelatedRecord
PCQB_RqExecute(optHideFMWindow ; optFilter ; optPath)	PCQO_RqExecute
PCQB_RqUseXML(XMLText ; optIsPath)	
PCQB_RsOpenFirstRecord	PCQO_RsOpenFirstRecord
PCQB_RsOpenNextRecord	PCQO_RsOpenNextRecord
PCQB_RsGetFirstFieldValue(QBFieldName)	PCQO_RsGetFirstFieldValue(FieldName)
PCQB_RsGetAttribute(AttributeName)	
PCQB_RsGetResponseAttribute(AttributeN	
PCQB_RsOpenFirstRelatedRecord(Type)	PCQO_RsOpenFirstRelatedRecord(Type)
PCQB_RsOpenNextRelatedRecord	PCQO_RsOpenNextRelatedRecord
PCQB_RsUseXML(XML ; optIsPath ;	
PCQB_SGetXML(Type ; optPath)	PCQO_SGetXML(Type)*
PCQB_SGetStatus	PCQO_SGetStatus
PCQB_SFormatString(String ; qbType)	
PCQB_ValidateRequests(bParam)	
	PCQO_SCountEntities(EntityType)
	PCQO_SSaveSessionInfo
	PCQO_SLoadSessionInfo(SessionInfoStrin
	PCQO_RqUseJSON(JSON)
	PCQ_RsUseJSON(JSON)
	PCQO_SGetJSON(Type)

*The PCQO_SGetXML(Type) function is deprecated as of FM Books Connector Online version 2.0.0.0. The plug-in now uses JSON to store data internally, and calling PCQO_SGetXML will return a JSON object instead of an XML document. Please use the PCQO_SGetJSON function to get a text representation of the Request or Response object(s) instead.

Notable Differences:

1) PCQO_RqNew accepts two parameters: an operation, and an entity. The operation is either “Create”, “Update”, “Query”, or “Delete”. The entity is the type of record that the operation will apply to, such as “Customer”, “Invoice”, “Purchase”, and others. A list of valid entities can be found in Intuit’s supporting documentation. Further details about the parameters for PCQO_RqNew can be found in the Functions Guide.

2) The “Z Functions” from the FileBooks Link migration have not been implemented into the FM Books Connector Online plug-in, due to the new architecture of communicating with the QuickBooks Online Company, and that it does not support the qbXML format any longer. The new plug-in is able to convert requests and responses in memory to XML, but cannot accept XML for requests or responses at this time. The use of XML should be strictly for troubleshooting or external parsing purposes.

3) FM Books Connector allows the ability to modify attributes, which specifically refers to the “attributes” found within an XML tag. As the FM Books Connector Online does not use XML to handle requests and responses, attribute fields are set through the

PCQO_RqAddFieldWithValue function, and read through the PCQO_RsGetFirstFieldValue function. These fields are the “attributes” of the entity that the request or response is working on. Refer to the Entity Services Reference located at the link below for a table of attributes/ fields for entities.

https://developer.intuit.com/docs/api/accounting

Scripting

The way that requests are generated in the FM Books Connector Online plug-in is primarily similar to those created by the FM Books Connector plug-in, with a few exceptions:

1) Working with entity lines requires “nested related records”, for specifying general line information and more specific type-related information

2) Field names are based off of the new Intuit® Partner Platform structure for entities, and can be referenced in their supporting documentation.

The following is a comparison of scripts, showing both an example of similarity between FM Books Connector and FM Books Connector Online, and an example of differences between FM Books Connector and FM Books Connector Online.
These scripts are pulled from the FM Books Connector demo file for use with QuickBooks Online, and the FM Books Connector Online demo file.

Case 1: Similarity in Script Structure

```
#Prep Customer info (Request)
Go to Layout [ “Main” (Main) ]
Go to Object [ Object Name: "Push" ]
Set Variable [ $$Result ; Value:PCQB_RqNew("CustomerAdd" ; "" ) ]
Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "Name" ; Main::gPush_Customer Company ) ]
Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "CompanyName" ; Main::gPush_Customer Company ) ]
Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "FirstName" ; Main::gPush_Customer First Name ) ]
Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "LastName" ; Main::gPush_Customer Last Name ) ]
Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "Phone" ; Main::gPush_Customer Phone ) ]
Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "Email" ; Main::gPush_Customer Email ) ]
#Execute
Set Variable [ $$Result ; Value:PCQB_RqExecute ]
```

To create a new Customer in QuickBooks Online through the FM Books Connector, you could structure your script like so:

```
#Prep Customer info (Request)
Go to Layout [ “Main” (Main) ]
Go to Object [ Object Name: "Push" ]
Set Variable [ $$Result ; Value:PCQO_RqNew( "Create" ; "Customer" ) ]
Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "DisplayName" ; Main::gPush_Customer Company ) ]
Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "CompanyName" ; Main::gPush_Customer Company ) ]
Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "GivenName" ; Main::gPush_Customer First Name ) ]
Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "FamilyName" ; Main::gPush_Customer Last Name ) ]
Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "PrimaryPhone" ; Main::gPush_Customer Phone ) ]
Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "PrimaryEmailAddr" ; Main::gPush_Customer Email ) ]
#Execute
Set Variable [ $$Result ; Value:PCQO_RqExecute ]
```

With the FM Books Connector Online, the same script would be structured like so:

With the exception of field names, the structure for pushing a customer via the FM Books Connector Online is the same; start the request with PCQO_RqNew, populate the fields using PCQO_RqAddFieldWithValue, and then execute the request with PCQO_RqExecute.

Case 2: Differences in Script Structure

An example of differences between scripting for the FM Books Connector and scripting for the FM Books Connector Online can be seen below.

```
#Prep Invoice  info (Request)
Set Variable [ $$Result ; Value:PCQB_RqNew("InvoiceAdd" ; "" ) ]
Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "CustomerRef::ListID" ; Main::gPush_Customer ListID ) ]
Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "TxnDate"; Main::gPush_Invoice TxnDate) ]
# If the invoice has a reference number (Invoice ID, RefNumber, etc.), push it; otherwise,
# don't include this function, so that QuickBooks will generate its own Invoice RefNumber.
If [ not IsEmpty(Main::gPush_Invoice RefNumber) ]
    Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "RefNumber" ; Main::gPush_Invoice RefNumber ) ]
End If
#
Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "DueDate"; Main::gPush_Invoice DueDate) ]
Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "ShipDate"; Main::gPush_Invoice ShipDate) ]
Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "Memo"; Main::gPush_Invoice Memo) ]
#Prep Line Items (Request)
Go to Field [ Inv LI Push::InvoiceLineDesc ]
Go to Portal Row [ Select; First ]
Loop
    Set Variable [ $$Result ; Value:PCQB_RqAddRelatedRecord( "InvoiceLineAdd" ; "" ) ]
    Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "ItemRef::FullName" ; Inv LI Push::InvoiceLineItemNo )
    ]
    Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "Desc" ; Inv LI Push::InvoiceLineDesc) ]
    Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "Quantity" ; Inv LI Push::InvoiceLineQuantity ) ]
    Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "Rate" ; Inv LI Push::InvoiceLineRate ) ]
    If [ not IsEmpty ( Inv LI Push::InvoiceLineClassRefFullName ) ]
        #Push class only if it's included in the FileMaker file.
        Set Variable [ $$Result ; Value:PCQB_RqAddFieldWithValue( "ClassRef::FullName"; Inv LI Push::
        InvoiceLineClassRefFullName ) ]
    End If
    Set Variable [ $$Result ; Value:PCQB_RqCloseRelatedRecord ]
    Go to Portal Row [ Select; Next; Exit after last ]
    Exit Loop If [ not IsValid(Inv LI Push::InvoiceLineQuantity) ]
End Loop
#Execute
```

To pull an invoice and its related line items through the FM Books Connector, the script would be structured like so:

The following is the same invoice push written using the FM Books Connector Online plug-in:

```
#Prep Invoice  info (Request)
Set Variable [ $$Result ; Value:PCQO_RqNew( "Create" ; "Invoice" ) ]
Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "CustomerRef::Value" ; Main::gPush_Customer ListID ) ]
Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "TxnDate"; Main::gPush_Invoice TxnDate)
# If the invoice has a reference number (Invoice ID, RefNumber, etc.), push it; otherwise,
# don't include this function, so that QuickBooks will generate its own Invoice RefNumber.
If [ not IsEmpty(Main::gPush_Invoice RefNumber) ]
     Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "DocNumber" ; Main::gPush_Invoice RefNumber ) ]
End If
#
Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "DueDate"; Main::gPush_Invoice DueDate) ]
Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "ShipDate"; Main::gPush_Invoice ShipDate) ]
Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "CustomerMemo" ; Main::gPush_Invoice Customer Memo ) ]
Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "PrivateNote" ; Main::gPush_Invoice Private Memo ) ]
#
# Tax Line information
# ...
# End Tax Line information
#
#Prep Line Items (Request)
Go to Field [ Inv LI Push::InvoiceLineDesc ]
Go to Portal Row [ Select; First ]
Loop
     Set Variable [ $$Result ; Value:PCQO_RqAddRelatedRecord( "Line" ) ]
     Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "Description" ; main_INV LI Push::InvoiceLineDesc) ]
     Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "DetailType" ; main_INV LI Push::InvoiceLineDetailType )
     ]
     Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "Amount" ; main_INV LI Push::InvoiceLineAmount ) ]
     # Currently, the Demo File will only push lines as "SalesItemLineDetail" records. Other detail types are
     supported by QBO:
     # - SalesItemLineDetail, SubtotalLineDetail, DiscountLineDetail, DescriptionOnly
     If [ main_INV LI Push::InvoiceLineDetailType = "SalesItemLineDetail" ]
          Set Variable [ $$Result ; Value:PCQO_RqAddRelatedRecord( "SalesItemLineDetail" ) ]
          Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "ItemRef::Value" ; main_INV LI
          Push::InvoiceLineItemNo ) ]
          Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "Qty" ; main_INV LI Push::InvoiceLineQuantity ) ]
          Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "UnitPrice" ; main_INV LI Push::InvoiceLineRate ) ]
          Set Variable [ $$Result ; Value:PCQO_RqAddFieldWithValue( "TaxCodeRef::Value" ; main_INV LI Push::
          InvoiceLineSalesTaxCodeRefFullName ) ]
          Set Variable [ $$Result ; Value:PCQO_RqCloseRelatedRecord ]
     End If
     Set Variable [ $$Result ; Value:PCQO_RqCloseRelatedRecord ]
     Go to Portal Row [ Select; Next; Exit after last ]
     Exit Loop If [ not IsValid(main_INV LI Push::InvoiceLineQuantity) ]
End Loop
#Execute
Set Variable [ $$Result ; Value:PCQO_RqExecute ]
```

When dealing with Line entities (Invoice lines, Payment lines, Tax lines, etc.), the Line entity is separated into two parts: a header, and a detail. The header holds common information, such as the description of the line, the numerical currency amount of the line, and a DetailType that indicates what specifically the line is. Additionally, Line entities may have linked transactions (for example, a Payment Line may have a Linked Transaction field that contains the QB ID of an Invoice to which the payment was applied). To properly set line information in a request, a developer must add a related Line record to the parent entity, fill out the necessary Line Header information, declare a Line detail type, and then add a related record of that detail type to the Line record and fill out the necessary detail fields and close out of both the line detail record and the line record itself. To pull information from related Line records, it would work similarly: open the main Line record, gather the Line Header data, then open the related Line Detail record by its Detail Type and gather the detail information, before moving on to any remaining Line records.

Further details and explanations of how Lines are handled can be found in the Accounting API documentation:
https://developer.intuit.com/docs/0025_quickbooksapi/0050_data_services/020_key_concepts/0700_other_topics

Additional Support

If you require additional support in your migration from FM Books Connector to FM Books Connector Online for integration with QuickBooks Online, or would like to start a new integration between FileMaker and Intuit® QuickBooks Online, please contact us via the avenues listed below.

Phone: 760-510-1200
Email: support@productivecomputing.com
Forum: www.productivecomputing.com/forum

Please note assisting you with implementing this plug-in (excluding registration) is billable at our standard hourly rate. We bill on a time and materials basis billing only for the time in minutes it takes to assist you. We will be happy to create your integration scripts for you and can provide you with a free estimate if you fill out a Request For Quote (RFQ) at www.productivecomputing.com/rfq. We are ready to assist and look forward to hearing from you!