SMTP with OAuth for Office 365 in FileMaker

For years we had the XOAuth2Bearer option in our CURL functions for our plugins. Any customer asking for how to use oAuth with Microsoft or Gmail got pointed to this property. Register an application with Microsoft or Google to get your client ID & secret. Then show a login screen in a browser for the user to login and grab the token. There are plenty of frameworks available and implementations in various programming languages. Once you got the access token, you can pass it to the CURL XOAuth2Bearer option. Since this is some work, we today like to show you sample scripts in FileMaker to do this:

App Setup

For the Office 365 access, please go the portal.azure.com website and login. Then go to Azure Active Directory (currently a blue pyramid icon). When you come there, you can copy your Tenant ID (an UUID) for later.

Click on the left bar on the App Registrations section and then click there to add a new registration. Pick a name for your application and pick which account types you like to use. We picked the third one for multiple organizations and personal accounts. For the redirect URI, we pick web and then put in the "http://localhost:9999/". This is what we use with the WebHook functions later to catch the answer from the authentication. The port number can be chosen freely from 1025 to 65535 and 9999 is easy to remember for our example.

After you created the application, please copy the application ID. That is the client ID (an UUID) in the scripts. For the secrets, pick second tab for client secrets and add a new client secret. Pick a nice name and a long expiration date, e.g. 24 months. Now copy the client secret, a string with random characters.

Microsoft has an article to explain registration here. Since the exact steps may vary, please be prepared to look for the buttons on a new place, if you read this in a few months.

Start Script

Let's start to get an authorization. First we setup a WebHook to catch the response later. We listen on the port we provided above, in our example 9999. We then define the WebHookReceived script to be triggered. Please do not forget to check if fmplugin privilege for FileMaker 19.2 is granted if it exists. Next we define the response to send, which may show briefly in the web viewer later.

```
If [ IsEmpty ( $$WebHooks ) ]
	# setup callback
	Set Variable [ $$WebHooks ; Value: MBS("WebHook.Create") ]
	Set Variable [ $r ; Value: MBS("WebHook.Listen"; $$WebHooks; 9999) ]
	Set Variable [ $r ; Value: MBS("WebHook.SetScript"; $$WebHooks;
Get(FileName); "WebHookReceived") ]
	#
	Set Variable [ $text ; Value: "<html><p>Request arrived.</p></html>" ]
	Set Variable [ $text ; Value: "HTTP/1.1 200 OK¶Server: MyServer
1.0¶Connection: close¶Content-Type: text/html¶Content-Length: 36¶¶" & $text ]
	Set Variable [ $text ; Value: MBS( "Text.ReplaceNewline"; $Text; 3 ) ]
	Set Variable [ $r ; Value: MBS("WebHook.SetAutoAnswer"; $$Webhooks;
$text; "UTF-8") ]
End If
```

Load login page

Next we calculate the URL to load in the web viewer. This includes the scope and we can ask for a lot of things, but here we just ask for SMTP send permissions. Once we have the right URL, we load it into the browser to show the login screen. This seems to work better on macOS when we set the custom user agent to pose as Safari.

```
Set Variable [ $clientID ; Value: Trim(Office 365 oAuth SMTP::ClientID) ]
Set Variable [ $TenantID ; Value: Trim(Office 365 oAuth SMTP::TenantID) ]
Set Variable [ $redirectURI ; Value: "http://localhost:9999/" ]
Set Variable [ $redirectURI ; Value: MBS("Text.EncodeURLComponent";
$redirectURI; "UTF-8") ]
Set Variable [ $scope ; Value: "https://outlook.office365.com/SMTP.Send" //
"openid profile offline_access https://outlook.office365.com/POP.AccessAsUser.All
https://outlook.office365.com/IMAP.AccessAsUser.All" ]
Set Variable [ $scope ; Value: MBS("Text.EncodeURLComponent"; $scope;
"UTF-8") ]
Set Variable [ $URL ; Value: "https://login.microsoftonline.com/" & $TenantID & "/
oauth2/v2.0/authorize?response_type=code&scope=" & $scope & "&redirect_uri="
& $redirectURI & "&client_id=" & $clientID & "&state=test" ]
# let web viewer be Safari
```

```
Set Variable [ $r ; Value: MBS("WebView.SetCustomUserAgent"; "web"; "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/16.1 Safari/605.1.15") ]
# Load the URL
Set Variable [ $r ; Value: MBS( "WebView.LoadURL"; "web"; $URL) ]
```

Catch response with WebHook

Our next script is the one triggered by the WebHook for the incoming request. There we pick the request and ask it for the URL components, where our plugin parses the URL into the parts. Once we got the URL parsed, we can free the request and the web hook itself. The web viewer can load the "about:blank" URL to discard the Microsoft login screen.

```
Set Variable [ $WebRequest ; Value: Get(ScriptParameter) ]
#
# we got the answer
Set Variable [ $URLComponents ; Value: MBS( "WebRequest.URLComponents"; $WebRequest ) ]
Set Field [ Office 365 oAuth SMTP::Answer ; $URLComponents ]
#
# and show full request for debugging
Set Variable [ $Text ; Value: MBS( "Text.ReplaceNewline"; MBS("WebRequest.GetRawData"; $WebRequest; "UTF-8"); 1) ]
Set Field [ Office 365 oAuth SMTP::Debug ; $Text ]
#
# free the webhook and request
Set Variable [ $r ; Value: MBS("WebRequest.Release"; $WebRequest) ]
Set Variable [ $r ; Value: MBS("WebHook.Release"; $$WebHooks) ]
Set Variable [ $$WebHooks ; Value: "" ]
#
# clear web viewer
Set Variable [ $r ; Value: MBS( "WebView.LoadURL"; "web"; "about:blank") ]
```

Query access token

After we got the URL components and pick the parameters and inside it the code field. When we have a code field, we can build the URL with the tenant ID. We make a POST request and build the data to send. This includes the code we just got, the client ID and the client secret. So we pass the code, which is only valid for client ID to the server and proof with the secret, that we are the right one to pick the access token up.

```
Set Variable [ $URLComponents ; Value: Office 365 oAuth SMTP::Answer ]
Set Variable [ $Parameters ; Value: JSONGetElement ( $URLComponents ;
"Parameters" ) ]
Set Variable [ $code ; Value: JSONGetElement ( $Parameters ; "code" ) ]
Set Variable [ $state ; Value: JSONGetElement ( $Parameters ; "state" ) ]
Set Variable [ $session_state ; Value: JSONGetElement ( $Parameters ;
"session_state" ) ]
#
# if we got a code, we continue to query the access code
If [ Length ( $code ) > 0 ]
	Set Variable [ $TenantID ; Value: Trim(Office 365 oAuth SMTP::TenantID) ]
	Set Variable [ $clientID ; Value: Trim ( Office 365 oAuth SMTP::ClientID ) ]
	Set Variable [ $clientSecret ; Value: Trim(Office 365 oAuth
SMTP::ClientSecret) ]
	Set Variable [ $URL ; Value: "https://login.microsoftonline.com/" & $TenantID
& "/oauth2/v2.0/token" ]
	Set Variable [ $redirectURI ; Value: MBS("Text.EncodeURLComponent";
"http://localhost:9999/"; "UTF-8") ]
	#
	Set Variable [ $Data ; Value: "code=" & $code & "&client_id=" & $clientID &
"&client_secret=" & $clientSecret & "&grant_type=authorization_code" &
"&redirect_uri=" & $redirectURI ]
	Set Variable [ $curl ; Value: MBS("CURL.New") ]
	Set Variable [ $result ; Value: MBS("CURL.SetOptionURL"; $curl; $URL) ]
	Set Variable [ $result ; Value: MBS("CURL.SetOptionPostFields"; $curl;
$Data; "UTF-8") ]
	Set Variable [ $result ; Value: MBS("CURL.SetOptionHTTPHeader"; $curl;
"application/x-www-form-urlencoded") ]
	Set Variable [ $result ; Value: MBS("CURL.Perform"; $curl) ]
	# Pick Result
	Set Variable [ $code ; Value: MBS( "CURL.GetResponseCode"; $curl ) ]
	If [ $result = "OK" and $code = 200 ]
		Set Variable [ $resultText ; Value: MBS("CURL.GetResultAsText";
$curl) ]
		Set Variable [ $debugText ; Value: MBS("CURL.GetDebugAsText";
$curl) ]
```

```
            Set Field [ Office 365 oAuth SMTP::CURL Debug ; $DebugText ]
            Set Field [ Office 365 oAuth SMTP::CURL Result ; $resultText ]
            Perform Script [ Specified: From list ; “Extract Access Token” ;
Parameter:    ]
        Else
            Show Custom Dialog [ "SMTP" ; "Failed to query token." ]
        End If
        Set Variable [ $result ; Value: MBS("CURL.Cleanup"; $curl) ]
End If
```

Once we got the result text, we can extract the values from the JSON. The important thing here is the access token. You may also store the expiration time and calculate the end date to get a new token in-time.

```
Set Variable [ $Result ; Value: Office 365 oAuth SMTP::CURL Result ]
# get access token
Set Variable [ $token_type ; Value: JSONGetElement ( $Result ; "token_type" ) ]
Set Variable [ $scope ; Value: JSONGetElement ( $Result ; "scope" ) ]
Set Variable [ $expires_in ; Value: JSONGetElement ( $Result ; "expires_in" ) ]
Set Variable [ $ext_expires_in ; Value: JSONGetElement ( $Result ;
"ext_expires_in" ) ]
Set Variable [ $access_token ; Value: JSONGetElement ( $Result ;
"access_token" ) ]
#
If [ Length ( $access_token ) > 0 ]
        Set Field [ Office 365 oAuth SMTP::access_token ; $access_token ]
        Show Custom Dialog [ "Got token!" ]
End If
```

Send an email

We got a token, so let's send an email. We use our SendMail functions to build an email. The SMTP server is smtp.office365.com with port 587 for Office 365. We need to enable TLSv1.2 and require encryption. On top we pass user name, but no password as we pass below the bearer token. If CURL.Perform returns OK, the email is sent.

```
# create email
Set Variable [ $EmailID ; Value: MBS("SendMail.CreateEmail") ]
Set Variable [ $r ; Value: MBS("SendMail.SetFrom"; $EmailID; Trim(Office 365
oAuth SMTP::From Email); Trim(Office 365 oAuth SMTP::From Name)) ]
Set Variable [ $r ; Value: MBS("SendMail.SetPlainText"; $EmailID; Office 365
oAuth SMTP::Email Text) ]
Set Variable [ $r ; Value: MBS("SendMail.SetSubject"; $EmailID; Office 365 oAuth
SMTP::Subject) ]
```

```
Set Variable [ $r ; Value: MBS("SendMail.SetSMTPServer"; $EmailID;
"smtp.office365.com") ]
Set Variable [ $r ; Value: MBS("SendMail.SetSMTPUsername"; $EmailID;
Trim(Office 365 oAuth SMTP::From Email)) ]
Set Variable [ $r ; Value: MBS("SendMail.AddTO"; $EmailID; Trim (Office 365
oAuth SMTP::To Email); Trim (Office 365 oAuth SMTP::To Name)) ]
#
Set Variable [ $curl ; Value: MBS("CURL.New") ]
Set Variable [ $r ; Value: MBS("SendMail.PrepareCURL"; $EmailID; $curl) ]
# Maybe use alternative SMTP port?
Set Variable [ $r ; Value: MBS("CURL.SetOptionPort"; $curl; 587) ]
# This turns TLS on and requires connection to be encrypted
Set Variable [ $r ; Value: MBS("CURL.SetOptionUseSSL"; $curl; 3) ]
# force TLS v1.2
Set Variable [ $r ; Value: MBS("CURL.SetOptionSSLVersion"; $curl; 6) ]
# put in token
Set Variable [ $r ; Value: MBS("CURL.SetOptionXOAuth2Bearer"; $curl; Office
365 oAuth SMTP::access_token) ]
# Run the transfer
Set Variable [ $r ; Value: MBS("CURL.Perform"; $curl) ]
# get debug messages
Set Field [ Office 365 oAuth SMTP::CURL Debug ;
MBS("CURL.GetDebugAsText"; $curl; "UTF-8") ]
Set Variable [ $r ; Value: MBS("CURL.Release"; $curl) ]
#
#  Cleanup
Set Variable [ $r ; Value: MBS("SendMail.Release"; $EmailID) ]
```

Please try it. We hope this works fine for you. Of course you can adapt the same code to work with other providers.
The example database will be included with future plugin downloads. Email us if you need a copy or have a question.

Scope

First there is the question about the scope of the token and how to use this for IMAP. So the scope string lists various identifiers separated by a space character and a few of those identifiers are URLs. You can lookup them in the documentation from Microsoft, but we have a few common ones:

Only SMTP:
Set Variable [$scope ; Value: "https://outlook.office365.com/SMTP.Send"

Only IMAP:
Set Variable [$scope ; Value: "https://outlook.office365.com/IMAP.AccessAsUser.All"

IMAP and SMTP:
Set Variable [$scope ; Value: "https://outlook.office365.com/SMTP.Send https://outlook.office365.com/IMAP.AccessAsUser.All"

More with POP3 and profile data and offline access:
Set Variable [$scope ; Value: "openid profile offline_access https://outlook.office365.com/SMTP.Send https://outlook.office365.com/POP.AccessAsUser.All https://outlook.office365.com/IMAP.AccessAsUser.All"

It may be good to keep this narrow and maybe only ask for SMTP most times so the user is not frightened that for sending emails you also need to read all their existing emails.

SmtpClientAuthentication disabled

Sending a test email, we may get an error:
535 5.7.139 Authentication unsuccessful, SmtpClientAuthentication is disabled for the Tenant. Visit https://aka.ms/smtp_auth_disabled for more information. [FR2P281CA0043.DEUP281.PROD.OUTLOOK.COM]

The admin may need to enable SMTP for this account.

Authenticated but not connected

For IMAP, we saw this kind of error:
A003 BAD User is authenticated but not connected.

The Exchange and Office365 servers will accept connections and authenticate users by their username and password when the user

doesn't have IMAP permissions. No error is given until the first IMAP command is issued, at which point it gives "User is authenticated but not connected".

To correct this, open the permissions for the user on the server and ensure that IMAP is selected.

Encryption trouble

Some people don't get the TLS connection due to old plugin with older SSL versions. Please use recent versions of MBS Plugins to get a recent OpenSSL with support for TLS 1.3. Please note that it is imaps://outlook.office365.com with an s attached to imap, so the port is 993 and the first data packet is encrypted. With smtp, it is without the extra s: smtp://smtp.office365.com as this uses STARTTLS, so connection starts unencrypted and then enables encryption later before sending the authentication.

Authentication unsuccessful

We saw errors like this:
535 5.7.3 Authentication unsuccessful

The token we just got is not accepted. Please check whether SMTP service is enabled for the account used.

Open Firewall

Since the Outlook application can connect to the mail server via HTTPS and their REST APIs, some clients seems to have the firewall configured to block the outlook.office365.com and smtp.office365.com domains. This can be at DNS level, so no IP is resolved. Or it can be on connection level, so the attempt to connect times out. Or it can be that the connection is happening, but then terminated within a second.

Successful request

There is an example of a CURL request, that works:
Trying 52.97.232.194:587...
Connected to smtp.office365.com (52.97.232.194) port 587 (#0)
220 ZR0P278CA0026.outlook.office365.com Microsoft ESMTP MAIL Service ready at Wed, 23 Nov 2022 09:12:07 +0000
EHLO test-mbp-3
250-ZR0P278CA0026.outlook.office365.com Hello [31.11.3.242]
250-SIZE 157286400
250-PIPELINING

```
250-DSN
250-ENHANCEDSTATUSCODES
250-STARTTLS
250-8BITMIME
250-BINARYMIME
250-CHUNKING
250 SMTPUTF8
STARTTLS
220 2.0.0 SMTP server ready
TLSv1.3 (OUT), TLS handshake, Client hello (1):
TLSv1.3 (IN), TLS handshake, Server hello (2):
TLSv1.2 (IN), TLS handshake, Certificate (11):
TLSv1.2 (IN), TLS handshake, Server key exchange (12):
TLSv1.2 (IN), TLS handshake, Request CERT (13):
TLSv1.2 (IN), TLS handshake, Server finished (14):
TLSv1.2 (OUT), TLS handshake, Certificate (11):
TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
TLSv1.2 (OUT), TLS handshake, Finished (20):
TLSv1.2 (IN), TLS handshake, Finished (20):
SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
Server certificate:
subject: C=US; ST=Washington; L=Redmond; O=Microsoft
Corporation; CN=outlook.com
start date: Jul 26 00:00:00 2022 GMT
expire date: Jul 25 23:59:59 2023 GMT
issuer: C=US; O=DigiCert Inc; CN=DigiCert Cloud Services CA-1
SSL certificate verify result: unable to get local issuer certificate (20),
continuing anyway.
EHLO test-mbp-3
250-ZR0P278CA0026.outlook.office365.com Hello [31.11.3.242]
250-SIZE 157286400
250-PIPELINING
250-DSN
250-ENHANCEDSTATUSCODES
250-AUTH LOGIN XOAUTH2
250-8BITMIME
250-BINARYMIME
250-CHUNKING
250 SMTPUTF8
AUTH XOAUTH2
334
dXNlcj1tYXJ....RMTlERkM3S0xlVmRIQQEB
235 2.7.0 Authentication successful
MAIL FROM:<user@domain.com>
```

250 2.1.0 Sender OK
RCPT TO:<other@domain.com>
250 2.1.5 Recipient OK
DATA
354 Start mail input; end with <CRLF>.<CRLF>
250 2.0.0 OK <315CA87F-XXXX-4EFA-XXX-D209056F067F@domain.com>
[Hostname=ZR0XXXXMB0XX3.CHEP278.PROD.OUTLOOK.COM]
Connection #0 to host smtp.office365.com left intact

Of course all identifying information got removed, so you won't see the true user name above.

Let us know if you have questions so far.

IMAP

If you like to change our IMAP example to use oAuth, please use script steps like this:

Set Variable [$r ; Value: MBS("CURL.SetOptionURL"; $curl; "imaps://outlook.office365.com/INBOX")]
Set Variable [$r ; Value: MBS("CURL.SetOptionXOAuth2Bearer"; $curl; "eyJ0eXAiOi...")] // your oAuth token
Set Variable [$r ; Value: MBS("CURL.SetOptionUserName"; $curl; "test@outlook.com")] // no password needed!

You remove the CURL.SetOptionPassword there and add the CURL.SetOptionXOAuth2Bearer with the token. For the URL make sure you have the imaps protocol with the s. A typo in the URL and nothing will work. Don't forget the user name, which is usually the email from the account. This may be or may not be the same as the from or sender email.

Refresh Token

We updated the example database for 13.1 to include handling of refresh tokens. The tokens by default expire within 30 to 90 minutes. So you need to refresh the token. And your admin my

Please add offline_access to the scope to enable this. e.g. use this scope:

"offline_access https://outlook.office365.com/SMTP.Send https://outlook.office365.com/IMAP.AccessAsUser.All"

That is for offline_access to get the refresh_token, the SMTP and IMAP access permissions. You may leave away one of them if you like.

Please check the script on how to refresh the token. Since refresh tokens work some time, e.g. 90 days, you may need to do the refresh regularly to get a new access token and a new refresh token.

Enable flags in Office 365 admin settings

We run into several users, where SMTP, IMAP or offline access was not checked in the options. But his is Office 365 specific, so you need to find the right check marks to allow the user.

Book keeping

Please be aware, that you need to keep track of a lot of things here. For each mail account, you need to store the current access token and the refresh token.
When you get such a token, please write down the current time in a field for each. The JSON from the server includes expires in values which provide a time in seconds for how long they are valid, so you can calculate the end timestamp and store them.

For the refresh, please store the scope used to get the original token as you need to specify the same scope again later for a refresh.

And since the client secret changes at least every 2 years, you need to store which secret belongs to which key as you may have at some point two client secrets in use in parallel.

PS: Also check if you have a fmplugin extended privilege, which if not allowed, prevents the plugin from triggering scripts.