How FileMaker Developers Can Extend Authentication Options With New Additional OAuth2 Identity Providers In The FileMaker Platform

—By—

Wim Decorte and Steven H. Blackwell

Version 1.0 October 2019

CONTENTS

EXECUTIVE SUMMARY

Since the advent of External Authentication based on the oAuth2 Protocol that Claris International, Inc. introduced in FileMaker® Server 16, some developers have searched for ways to expand the number of Identity as a Service (IDaaS) Providers that they could employ to provide authentication for access to hosted FileMaker Pro files. FileMaker Server allowed for three such providers, only one of which supported group structures.

Based on some work done at a FileMaker Platform customer with assistance from Claris Consulting Engineers and Product Development Engineers[1] the authors—both long-time FileMaker Server and Security experts—experimented with and devised methods to expand the universe of IDaaS Providers. **We are pleased to advise both developers and customers that we have thoroughly documented the requirements for making the underlying mechanism work, and we have at least five new providers working.** And we believe that there are more to come. This document will allow you to evaluate whether your preferred IDaaS provider can be made to work with the current versions of the FileMaker platform. In a subsequent Addendum to this White Paper to be released at a latter date, we hope to provide some Case Studies about implementation of this feature at actual FileMaker customer locations.

This White Paper explains the reasons for wishing to expand these services and the importance of Federated Identity Management and the Zero Trust Model. It also provides detailed and specific instructions, complete with illustrations, of what configuration changes Developers and Administrators need to make using Okta as an example IDaaS. *We hope—and dare to expect—this means, among other items, that the FileMaker Platform can now be used in organizations where heretofore it was disqualified due to lack of support of these Authentication options.*

---

[1] We are grateful to a number of our colleagues at Claris International, Inc. (formerly FileMaker, Inc.). for their assistance in the preparation of this White Paper and for their work at the customer site in investigating this process. Their names are listed on Page 32.

HOW FILEMAKER DEVELOPERS CAN EXTEND AUTHENTICATION OPTIONS WITH NEW ADDITIONAL OAUTH2 IDENTITY PROVIDERS IN THE FILEMAKER PLATFORM

Version 1.0

FileMaker® Server 16, introduced in April of 2017, brought with it the capability to use three oAuth2 Identity Providers to authenticate Identity Assertions made in attempts to access hosted FileMaker® Pro files. Google, Amazon, and Microsoft® Azure AD were the three Identity Providers. Since then various members of the developer community have sought ways to expand beyond those three providers to employ other Identity as a Service (IDaaS) options.

We are pleased to advise that we have succeeded in an effort to expand the available Identity Provider services. We were aided in this endeavor to a substantial degree by assistance from Claris International, Inc. (formerly FileMaker, Inc.) working in a customer environment. This ratifies to a substantial degree the maxim postulated by well-known business author Tom Peters back in the 1970's that *"Innovation doesn't come from the R&D Department; it comes from the customer base."*

Why might we have been interested in expanding the list of available Identity Providers? Several reasons come to mind:

- A client may have already subscribed to such a service and wants to expand its use to the FileMaker Platform.
- We may wish to leverage various Multi-Factor Authentication (MFA) and Single Sign On (SSO) capabilities of such services to provide a more user-friendly and secure Identity and Access Management (IAM) for users. This will enable ease of administration of IAM for data owners and data security managers, or allow self-service for users to manage their passwords and other factors.
- We may wish to use the IDaaS Provider's pre-built reporting tools and Threat Analysis and Warning features.
- We may also wish to leverage password-free authentication options not otherwise available.
- Such a process can allow for geo-fencing, context-sensitive credentials challenges, and device monitoring.
- An expansion of IDaaS options also facilitates employment of the ***Zero Trust Model of Security***.

OVERVIEW

In this White Paper we will cover a number of specific items related to this topic:

- Federated Identity
- Applicability To FileMaker Platform
- How To Make It Work
- How To Extend To Other IDaaS Services
- Some Caveats To Consider

Expansion of IDaaS options for the FileMaker Platform provides a number of significant advantages to FileMaker developers and end-user customers. Among other items it enhances the capabilities of the Platform to meet varied requirements related to Identity and Access Management (IAM) in a variety of organizations. Likewise, it facilitates adoption of Federated Identity Management (FIdM) by commercial, not-for-profit, and public-sector organizations wishing to employ modern IDaaS systems. Additionally, it facilitates adoption of **Zero Trust Model**[2] for Information Security. Allow us to expand a bit on this important concept.

Users and Data and Systems are everywhere…

Businesses use more and more cloud-based services and Software-as-a-Service (SaaS) applications. In practical terms that means that the business data are in more places than ever before. Access to that data is governed by these separate systems, and that creates a challenge around uniform and centralized user ***Authentication*** and ***Authorization***.

The challenge is to authenticate user identity assertions and then have that authentication be passed to those numerous data sources so that the user will have the proper privileges in the proper application. Ideally there should be only one central place where identities are created and managed. This consolidation of Identities reduces the Attack Vector surface hackers can use to compromise Identities. It also facilitates easier provisioning and deprovisioning of Accounts in organizations. When duplicate identities exist, there is always the risk of mismatched privileges, maintenance hassles and possibly more attack points that can be compromised.

Traditional network perimeters[3] no longer exist, and ***Identity*** is the top attack vector against digital assets.

---

[2] https://fmforums.com/files/file/105-federated-identity-management-zero-trust-and-the-filemaker-platform/

[3] Over 150,000 Cloud applications. Over 8 billion mobile devices world-wide. Over 50 billion IoT devices world-wide.

Thus, *Identity Is the New Security*. A connection from a particular network **must not** determine user's access to services. *Users must be challenged and verified.* The three **Core Principles of Zero Trust** describe this:

- What do we know about you?
- What do we know about your device?
- All access to all services must be authorized once authentication is verified.

This means we ***verify the user*** with a variety of challenges. Then we ***validate the device.*** Then we ***limit access and privileges*** based on these challenges. This process needs to be ***adaptive*** and ***behavior-based***, changing with circumstances and may therefore require machine learning.

WHAT IS FEDERATED IDENTITY MANAGEMENT (FIDM) AND WHY IS IT IMPORTANT TO THE FILEMAKER COMMUNITY?

At base, Federated Identity Management (FIdM) is a means for *linking a person's electronic identity and related attributes across multiple, distinct systems.* ***FileMaker Server's use of oAuth2 capabilities enables it to use Federated Identity Management.***

> Federated identity provides a secure, standard, internet-friendly way to share identity among multiple organizations and applications. Users sign on once with a standard network login or hosted authentication service. When they click a web application link, their identity is transparently and securely shared with the application, removing the login requirement. Since the organization authenticates the user and the application provider can verify the authenticity of the provided federated identity, application passwords are not needed and users enjoy "click-and-work" access to applications.
>
> Identity federation is a huge win for users, IT and the business alike. Users love federation because…[it] enables them to use web applications as easily as internal applications while freeing them from remembering (and resetting) a battery of passwords. IT loves federation because it simultaneously enhances security and reduces the support burden, especially at the help desk. Business leaders love federation because it accelerates application and data sharing with customers, business partners, vendors and subsidiaries while decreasing risk and increasing regulatory compliance.[4]

---

[4] *SAML 101,* p. 6. (Ping Identity. Denver, CO. April 2017. www.pingidentity.com) https://www.pingidentity.com/content/dam/pic/downloads/resources/white-papers/en/saml-101-white-paper.pdf See also a webinar presented by Pamela Dingle, at the time, Principal Technical Architect at Ping, now Director of Identity Standards at Microsoft. *Emerging Standards on the Identity Landscape.* https://www.pingidentity.com/en/resources/webinar-replays/emerging-standards-on-the-identity-landscape-replay.html

FIdM is not the same thing as Single Sign-On; however, Single Sign-On (SSO) can be an attribute of Federated Identity. **The core purpose of FIdM is to enable *portability* of identity information across autonomous security domains.** This eliminates the need for new Account registration in multiple domains with the resulting redundant log-ons. This enables an organization to identify and authenticate a user once and then use that information across multiple systems.

It is very important to distinguish among several terms related to IAM that are frequently interchangeably used, thus promoting confusion. First, ***Authentication*** is distinct from ***Authorization***. The former refers to a process for validating Identity Assertions. *Are you who you say you are, and how do we know that?* Then, once we have validated your *Assertion*, we grant you *Authorization* to access various digital assets, *e.g.* a FileMaker Pro hosted database, with *specific rights and privileges*.

❖❖We have documented the underlying mechanism that FileMaker Server uses and have identified at least five major IDaaS providers whose oAuth2 capabilities we can leverage to permit their use in the FileMaker Platform environment. We have tested and verified **Ping Federate,**[5] **Okta,**[6] **OneLogin,**[7] **Auth0,**[8] **and MiniOrange**[9] as working with FileMaker Server. We are testing others. ***If the provider supports the OAuth2 protocol and can support the exchange of information about the user in the way that FileMaker Server expects, then we are hopeful it will work with FileMaker Server.***

❖Keep in mind that in this White Paper all our discussions and descriptions are focused on ***Authentication***, namely: *who are you, are you who you claim to be, and how do we know that.* The ***Authorization*** process remains the same; it is entirely controlled by the FileMaker Pro Privilege Set. Using an OAuth provider does not change the authorization you use in your FileMaker Pro solution.

❖*Authentication* is distinct from *Single Source Log-on* or *Single Sign-on* (SSO) as previously noted. SSO is a method to allow authentication to be passed from one source to another. In the FileMaker Platform environment, authenticating a user by an external process (rather than by an internal Account) may employ SSO; conversely, it may not do so, nor is it required to use SSO to achieve authentication.

❖We must also carefully distinguish an **Identity Provider** from a **Service Provider**. This process is complicated by the fact that some IDaaS processes can be both at the same time. An *Identity Provider* is just that: it contains the elements needed to establish a valid authentication of an Identity Assertion. A *Service Provider* acts as a broker. It queries an Identity Provider such as Active Directory for valid assertions and then passes the results of that process along to the system requesting the authentication

---

[5] www.pingidentity.com/en.html

[6] www.okta.com

[7] www.onelogin.com

[8] www.auth0.com

[9] www.miniorange.com

for purposes of granting authorized access. The Service Provider delegates the authentication to the Identity Provider.

The Okta example we will show however uses Okta's internal user repository; so, Okta acts as an Identity Provider at the outset.

HOW TO MAKE IT WORK

If you are looking for information on how to configure the three "traditional" OAuth providers, you may wish to consult an earlier White Paper[10] about configuring FileMaker Server to use OAuth2 that we released when FileMaker Server 16 deployed.

So, what did we actually do to enable other IDaaS providers to work with FileMaker Server? What is needed to make this happen?

To avoid confusion: this mechanism only works with providers supporting OAuth and OpenID Connect; it does not work with providers who only support SAML. SAML-based authentication currently is not supported in the FileMaker platform.

We have been able to make the OAuth flow work with both FileMaker Server 18 and 17 and FileMaker Pro Advanced 17 and 18. We have also successfully made this work with WebDirect™ with several major browsers. It feels like it should work with FileMaker Pro Advanced 16 as well, but we have not been able to make it work.

For this section we chose **Okta** as an example of how to accomplish this extensibility. Okta is a Forrester Wave™ Leader[11] for IDaaS; and, we have come across it at various clients, so it makes for a very good test scenario.

---

[10] https://fmforums.com/files/file/91-oauth-identity-providers/

[11] https://www.okta.com/resources/analyst-research-forrester-wave-leader-identity-as-a-service/

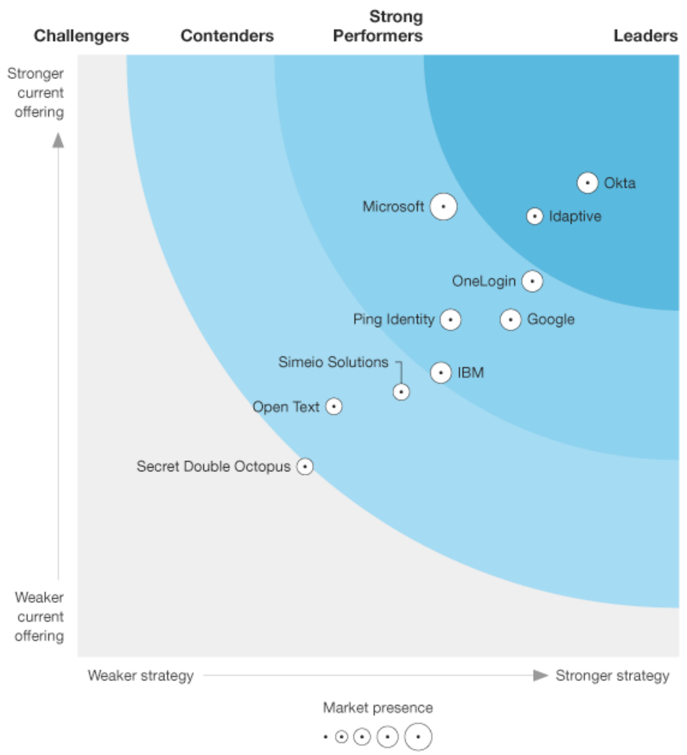


Figure 1. Okta is a Forrester Wave Leader.

To understand how the Provider's configuration needs to be set up, we must first understand what FileMaker Server expects to receive at the end of the authentication exchange and how FileMaker Server uses that information to grant access to a hosted file. And for that we need to take a look at how the OAuth 2 protocol works. In particular we are interested in Open ID Connect[12] (OIDC), which is the simplest way to implement this. From the OIDC web site:

> "OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner."

That profile information is important because it is the piece of information that will contain the user's name and the user's group information, and that group information will be matched against the groups that exist in the FileMaker file to assign the user to the correct Privilege Set.

As with most things, there is a *very specific* language that is used to describe the various parts of an OIDC exchange and its participants. We will highlight specific terms that you will likely encounter.

FileMaker Server can work with a pure OAuth non-OIDC authentication flow as well but since most IDaaS providers support OIDC we have limited the scope of this

---

[12] https://openid.net/connect/

document to OIDC-enabled configurations. We will provide a pure OAuth example in a separate White Paper at a future date.

The IDaaS provider provides you with some configuration details to be used in FileMaker Server setup. These include:

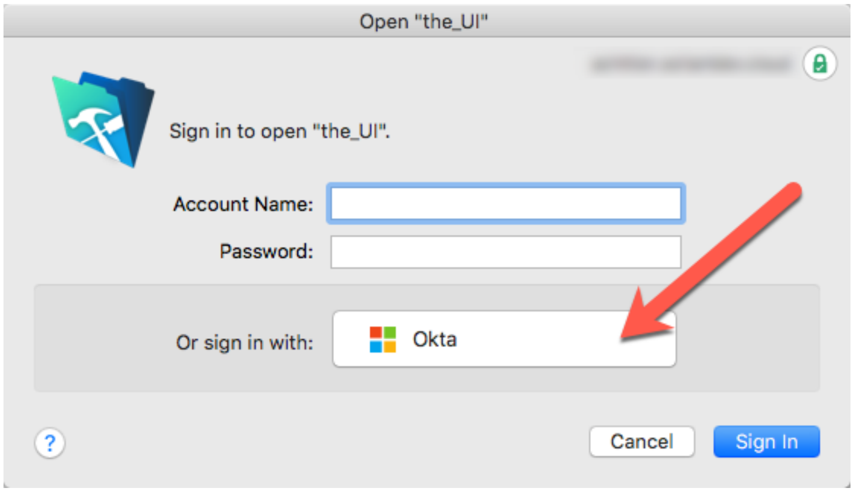
- The **Authentication URL** (where FileMaker Server will send the authentication request)
- The **Client ID** and **Client Secret** (two pieces of information that identify something that exists at the IDaaS provider. Most of them call this an "application")
- The **Token URL** (where FileMaker Server will send the request to for the user's profile)

Similarly, FileMaker Server provides a URL for us to use in the IDaaS Provider's configuration of the "application."

- The **Redirect URL** (where the IDaaS provider will send the result of the authentication)

There are a few different OAuth flows[13] and depending on your IDaaS provider you may have to select explicitly the proper one. FileMaker Server uses the "Authorization code flow." The whole exchange goes as follows:

1. User selects a file from a FileMaker Server that is properly configured, and clicks on the OAuth provider's button on the FileMaker login dialog:



Open "the_UI"

Sign in to open "the_UI".

Figure 2. FileMaker Login Dialog.

2. FileMaker Server calls the authentication URL for the IDaaS provider together with a random string called the "state." The state will be included in the IDaaS' response to FileMaker Server so that it can verify that the

---

[13] These flows are often referred to as "grant types." See https://oauth.net/2/grant-types/

response is from the IDaaS provider and not from some impersonator. In Open ID terms, FileMaker Server is the “relying party”: the party making the request. The IDaaS provider is the “OpenId Provider.”

3. The user’s browser will show the login page for the provider.

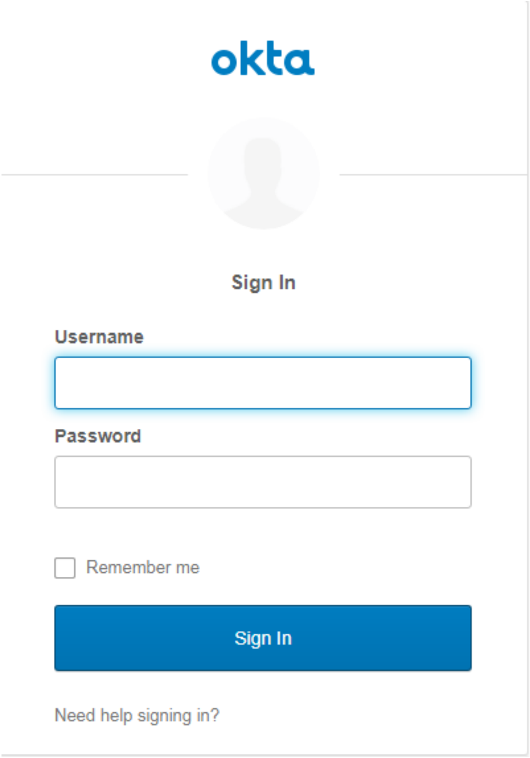


Figure 3 . Okta Gateway Page.

4. The user authenticates at the IDaaS by whatever means are set up at the IDaaS. This could be simply typing in a username and password; it could be inserting an ID card into a reader, providing a fingerprint, or a combination of factors.

5. The IDaaS sends a one-time code to FileMaker Server’s redirect URL if the authentication was successful, and it includes the “state” as previously described.

6. FileMaker Server receives the code and verifies the “state” and calls the IDaaS’ Token URL to exchange the code for a set of tokens. FileMaker Server also includes the Client ID and the Client Secret for the IDaaS “application” so that the IDaaS knows what to include in the user’s profile: namely what groups the user belongs to for the “application” identified by that Client ID and Secret.

7. The IDaaS provider verifies the code and sends a JSON response to FileMaker Server that contains a set of tokens. One of these tokens is the

ID Token (also known as *id_token*), a JSON Web Token (JWT)[14] encoded string that contains the user's profile including the user's email and the list of groups. The content of the JWT are called "claims," as in "what information about the user do you claim?" *Claims are the most crucial part of the IDaaS configuration to make this whole setup work.*

8. FileMaker Server reads the id_token and extracts the user's email address and the list of groups. FileMaker evaluates whether one of the Groups matches the name of a group as set up in the security schema of the FileMaker file and allows access to the file if there is a matching group.

Note that steps 3 and 4 may not happen if the user is already authenticated by the IDaaS in the browser's session. In that case, typically a blank web page will be shown, and the flow jumps straight to step 5.

The set of tokens that FileMaker Server receives in step 7 from the provider looks like this:

```
{
    "access_token": "NWE4Nzg2ZDEtNzQyMS00ZDViLThjMjctMGQwNjlmZjU5MWNkBGjFElT7CWzl0d....",
    "expires_in": 3600,
    "id_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IkpSY080bnhzNWpnYzhZZE43STJoTE80V...",
    "token_type": "Bearer",
    "refresh_token": "897987AGBEtNzQyMS00ZDViLThjMjctMGQwNjlmZjU5MWNkBGjFElT7CWzl0d...."
}
```

FileMaker Server is mostly interested in the **id_token**. That token, when decoded[15] contains "claims" information in this format:

```
{
  "sub": "00u5t60iloOHN9pBi0h7",
  "ver": 1,
  "iss": "https://{yourOktaDomain}",
  "aud": "0oabskvc6442nkvQO0h7",
  "iat": 1514497781,
  “email”: “wdecorte@soliantconsulting.com”
  "exp": 1514501381,
  "jti": "ID.kAe1asSO03M0jZtcfGdkiXl0QoKDq9hyw8MUQMT4l-Y",
  "amr": [
    "pwd"
  ],
  "idp": "00o5t60il3UzyIe5v0h7",
  "nonce": “1234568”
  "auth_time": 1514496350,
  "groups": [
    "WestCoastDivision".
    “Manager”
  ]
}
```

---

[14] JSON Web Token - https://tools.ietf.org/html/rfc7519

[15] There are some good online tools to help with decoding JWT for debugging purposes: https://jwt.io/ and https://www.jsonwebtoken.io/

Most of these claims are irrelevant for the task at hand, but if the token does not contain the claim for “groups” then FileMaker Server will not be able to match the authenticated user to its own group accounts, and the user will not get access to the FileMaker file. The claim for “email” or “unique_name” is used for the Get(AccountName) function.

With some IDaaS providers you need specifically to add a claim for the email, some include it by default. Typically, it is easy to make sure that “email” is included.

But if you run into problems configuring your chosen IDaaS, it will largely be because of the “groups” claim. ***Not all of the IDaaS providers provide an id_token with the groups claim just named “groups” or with the groups data in the expected array format.***

The reason for this is that “groups” is not a standard claim. It is widely used though and the Open ID Connect standard recommends that all additional (non-standard) claims use name-spacing to avoid naming conflicts.[16] What this means in practice is that the id_token from some providers will look like this:

```
{
  "http://yourdomain/claims/groups": [
    "fm_group_1"
  ],
  "email": "wdecorte@soliantconsulting.com",
  "email_verified": true,
  "iss": "https://some_privider.com/",
  "sub": "auth0|5d78f8d7ef51960e39cce361",
  "aud": "YowRSjyKx37apnUlFPbn8hUJHWfB3z50",
  "iat": 1568227905,
  "exp": 1568263905,
  "c_hash": "I_iHUdwzP9883GFGGDdYzQ",
  "nonce": "f8jaskrkn97"
}
```

with the word “groups” preceded by a chosen prefix. *If the provider insists on following the OpenID recommendation, then that name-space prefix needs to be included in the FileMaker Server configuration that we will discuss later in this document.*

If the provider cannot be configured to include a groups claim then you can still use that provider for ***individual*** Account authentication, similar to what we can do with Google and Amazon accounts. This means that you need to create an account in the FileMaker file for each individual user.

Auth0 for example enforces the recommendation.

---

[16] https://openid.net/specs/openid-connect-core-1_0.html#AdditionalClaims

Now that we know how the authentication and access-granting flow works, we will turn to Okta and see how we need configure it.

Configuring the Okta Identity Provider

In this section we will run through the configuration options that need to be made on the Okta side.

For this test scenario, we are using users and groups that exist in Okta's own "directory service" but of course these could come from any Identity Provider that Okta can work with: classic providers such as Active Directory, any LDAP directory, but also a myriad of other providers in the social realm (Facebook, LinkedIn, Google, …) and more general connectors to providers through OpenID or SAML.

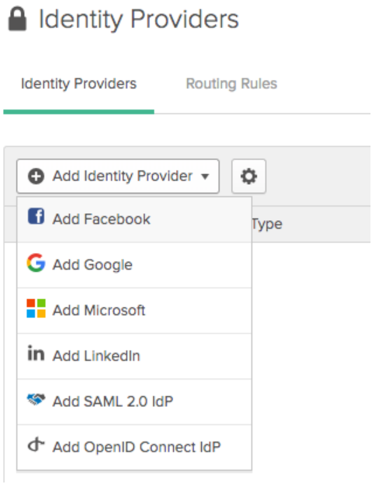
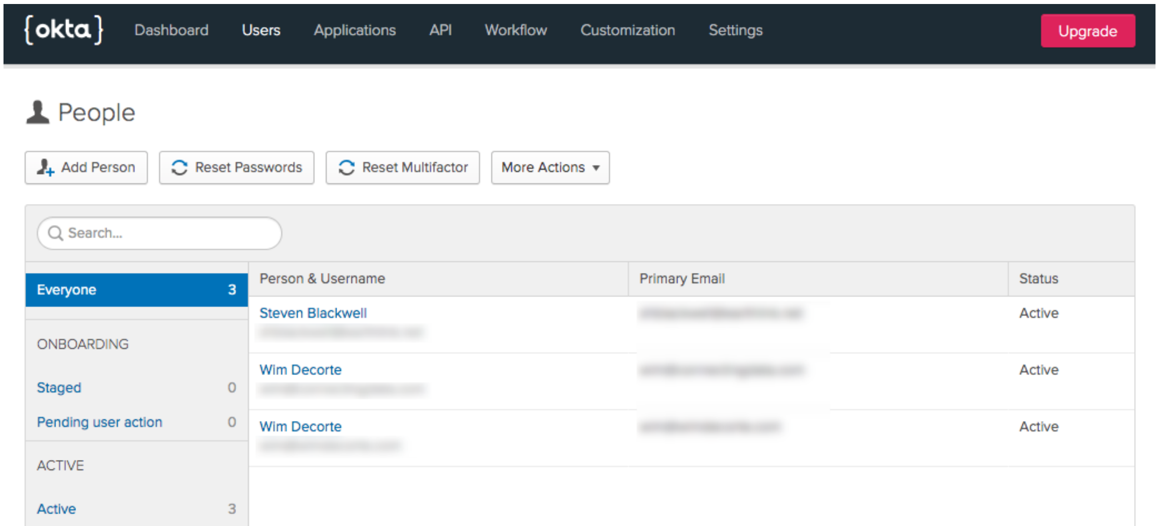


Figure 4. Adding Identity Providers to Okta.

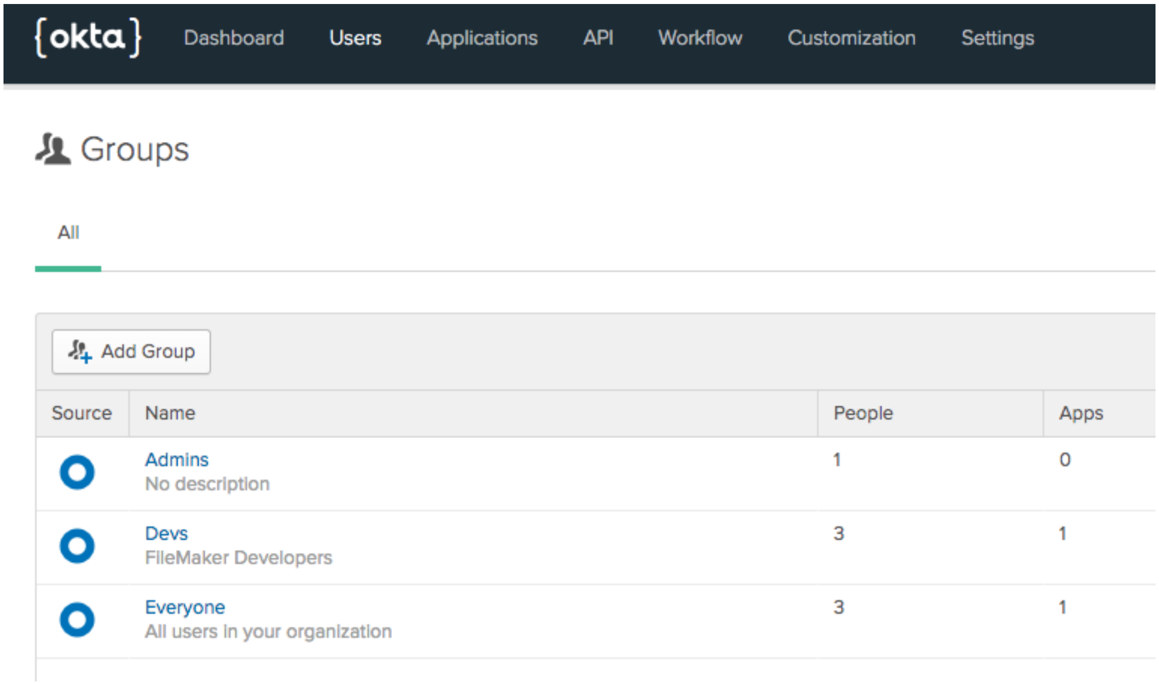
For this demo we have these Accounts in Okta:



Person & Username	Primary Email	Status
Steven Blackwell [redacted]	[redacted]	Active
Wim Decorte [redacted]	[redacted]	Active
Wim Decorte [redacted]	[redacted]	Active

Figure 5. Accounts in Okta.

And these Groups:



Source	Name	People	Apps
○	Admins No description	1	0
○	Devs FileMaker Developers	3	1
○	Everyone All users in your organization	3	1

Figure 6. Groups in Okta.

As mentioned, it all begins with setting up an "application" in the Identity Provider. Log into your Okta Console and from the Applications section add a new application. You may be able to reconfigure any existing application that you already have with the settings further down in this section.

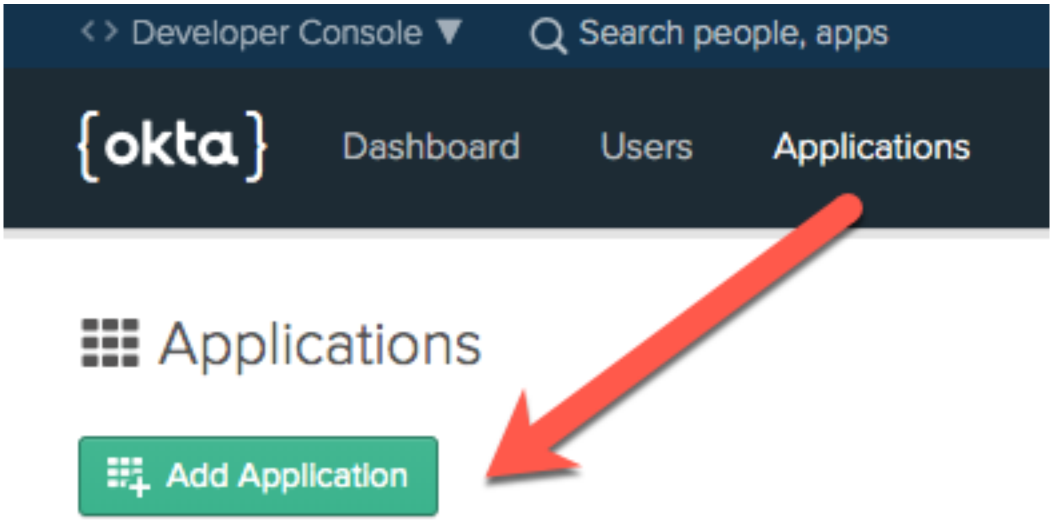
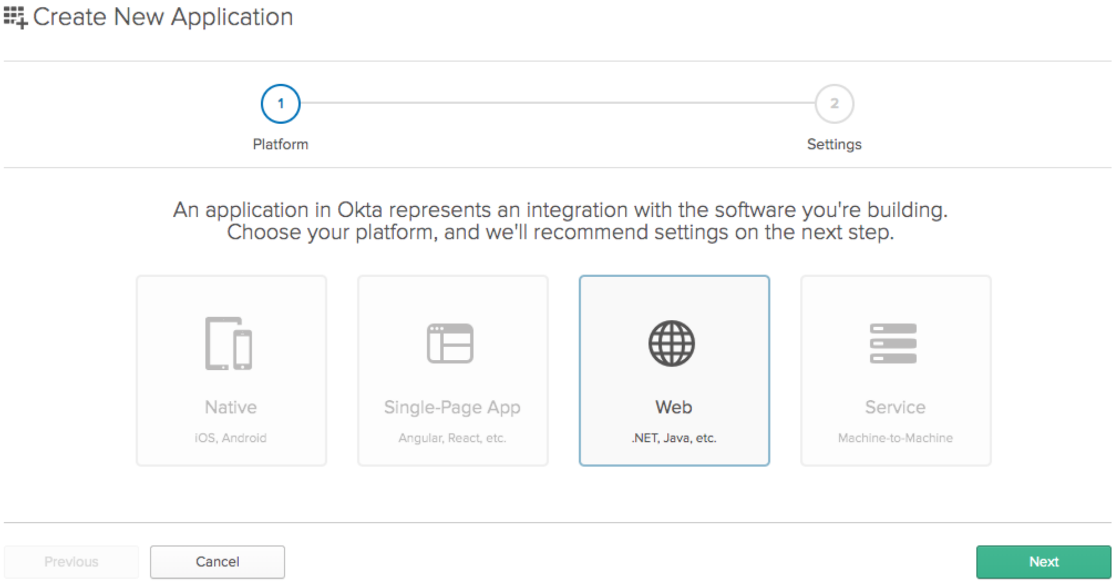


Figure 7. Adding an Application to Okta.

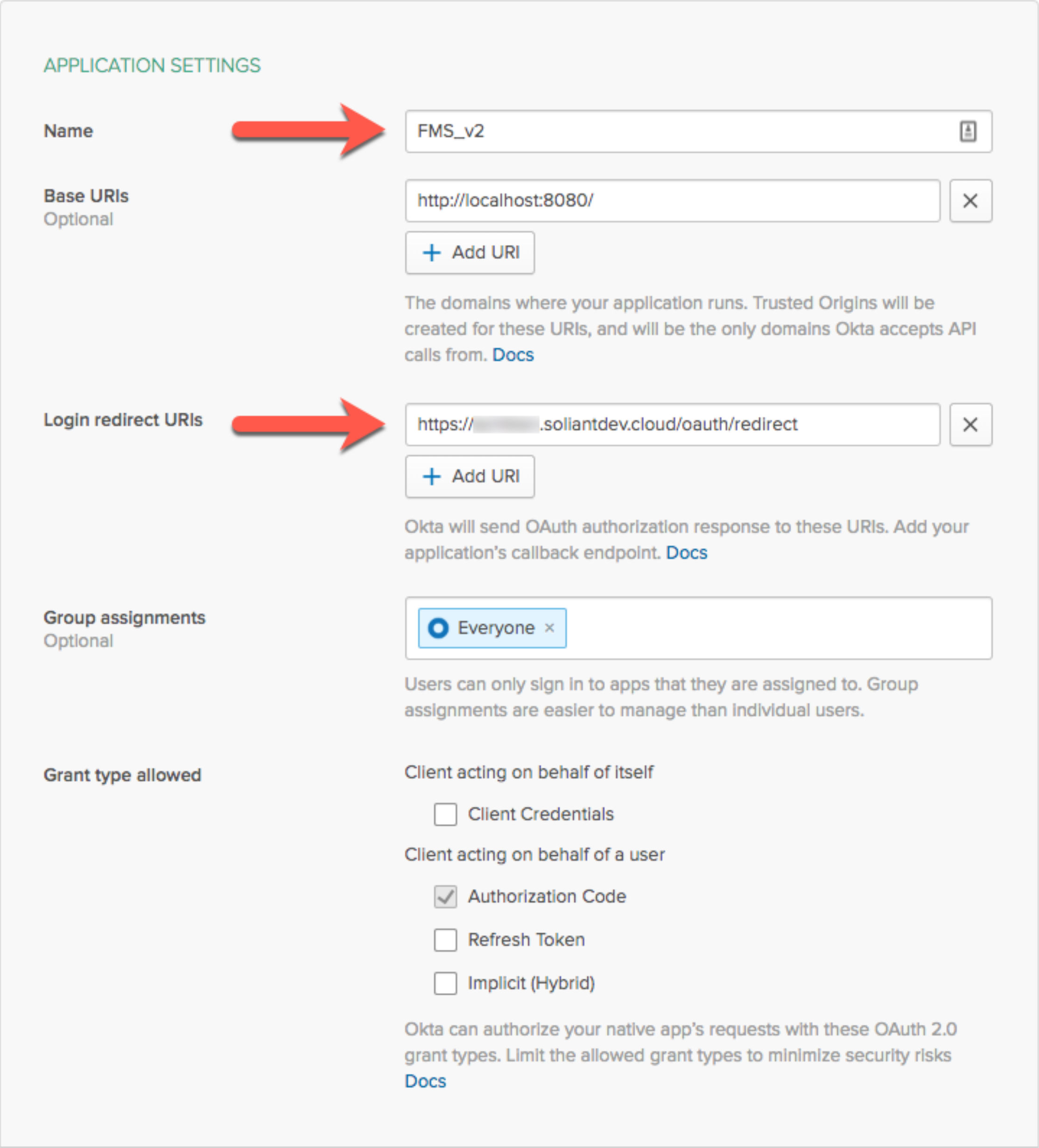
There are a few different options to choose from; choose *"Web"*.



An application in Okta represents an integration with the software you're building. Choose your platform, and we'll recommend settings on the next step.

Figure 8. Adding an Application to Okta.

In the configuration options, name your application and enter the FileMaker Server redirect URL:



APPLICATION SETTINGS

Name: FMS_v2

Base URIs Optional: http://localhost:8080/

The domains where your application runs. Trusted Origins will be created for these URIs, and will be the only domains Okta accepts API calls from. Docs

Login redirect URIs: https://[redacted].soliantdev.cloud/oauth/redirect

Okta will send OAuth authorization response to these URIs. Add your application's callback endpoint. Docs

Group assignments Optional: Everyone ×

Users can only sign in to apps that they are assigned to. Group assignments are easier to manage than individual users.

Grant type allowed

Client acting on behalf of itself

- ☐ Client Credentials

Client acting on behalf of a user

- ☑ Authorization Code
- ☐ Refresh Token
- ☐ Implicit (Hybrid)

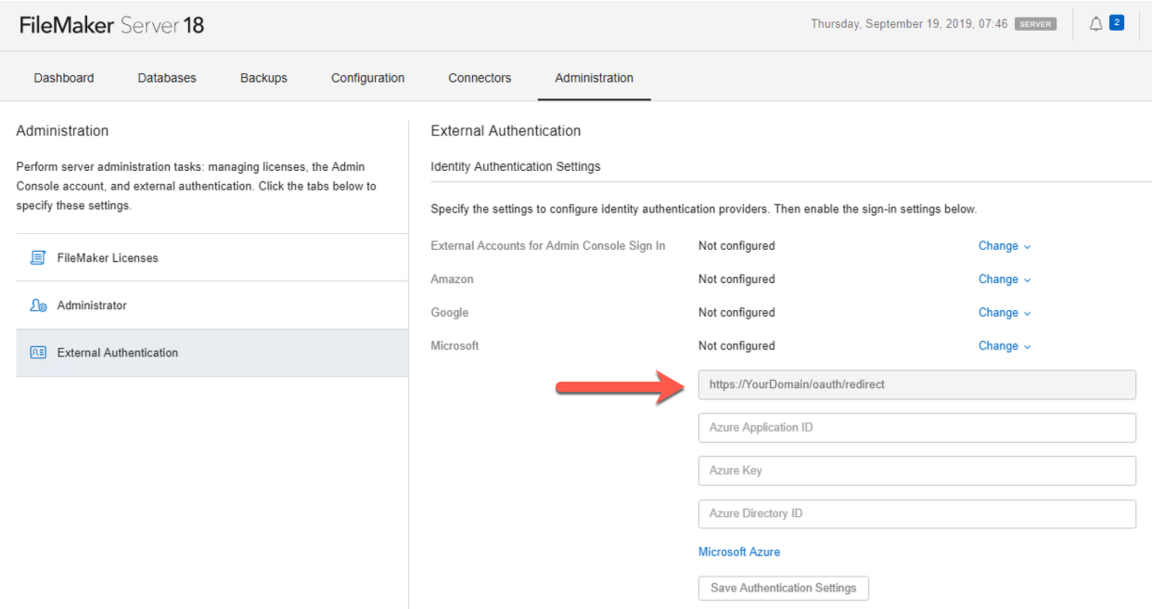
Okta can authorize your native app's requests with these OAuth 2.0 grant types. Limit the allowed grant types to minimize security risks Docs

Figure 9. Okta Application Settings.

The FileMaker Server redirect URL is always in this format:

https://<the DNS name of your server>/oauth/redirect

You can always check that by clicking "change" next to one of the three traditional OAuth providers in the FileMaker Server admin console.



Administration

Perform server administration tasks: managing licenses, the Admin Console account, and external authentication. Click the tabs below to specify these settings.

External Authentication

Identity Authentication Settings

Specify the settings to configure identity authentication providers. Then enable the sign-in settings below.

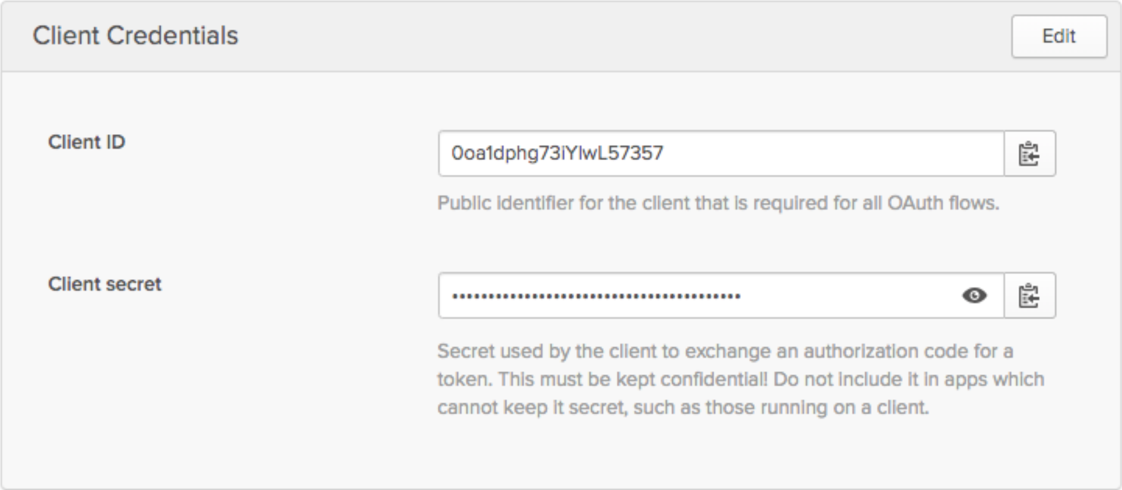
External Accounts for Admin Console Sign In: Not configured
Amazon: Not configured
Google: Not configured
Microsoft: Not configured

https://YourDomain/oauth/redirect
Azure Application ID
Azure Key
Azure Directory ID

Figure 10. The FileMaker Server Redirect URL format.

Note that when using any external OAuth provider (the traditional three or any supplemental one that you configure), your FileMaker Server needs to be reachable through public DNS on HTTPS (port 443 or a custom port of your choosing).

The other Okta application settings can remain as they are for now. When you save the settings, you will end up on the General tab of your Application. At the bottom of that page Okta will list the Client ID and Client Secret for the application.



Client ID: 0oa1dphg73iYlwL57357

Public identifier for the client that is required for all OAuth flows.

Client secret: ••

Secret used by the client to exchange an authorization code for a token. This must be kept confidential! Do not include it in apps which cannot keep it secret, such as those running on a client.

Figure 11. Client ID and Client Secret.

Keep a note of these two as they will be needed when we configure FileMaker Server.

At the top of the page, verify that the green icon is highlighted to indicate that OpenID Connect is enabled for the application:

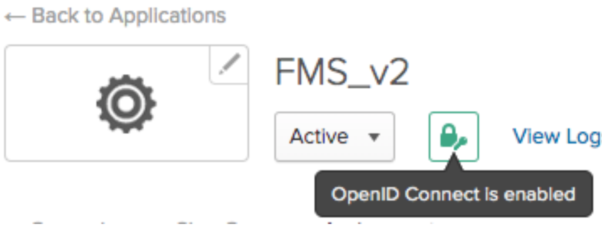


Figure 12. OpenID Enabled.

At this point we want to allow our existing "Devs" group to use this application. Still on the application, select the "Assignments" section, click the green "Assign" button and select "Assign to Groups" as shown:

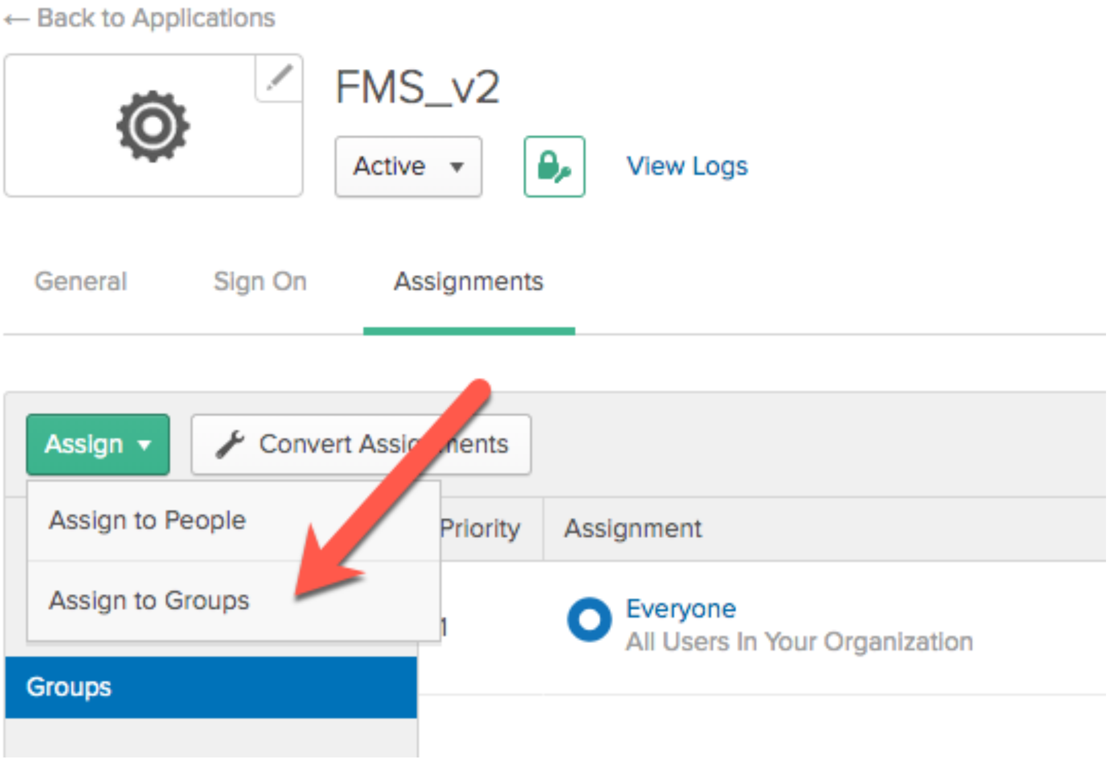


Figure 13. Assign to Groups.

Okta will show your available groups, select "Devs" to complete this part of the process.

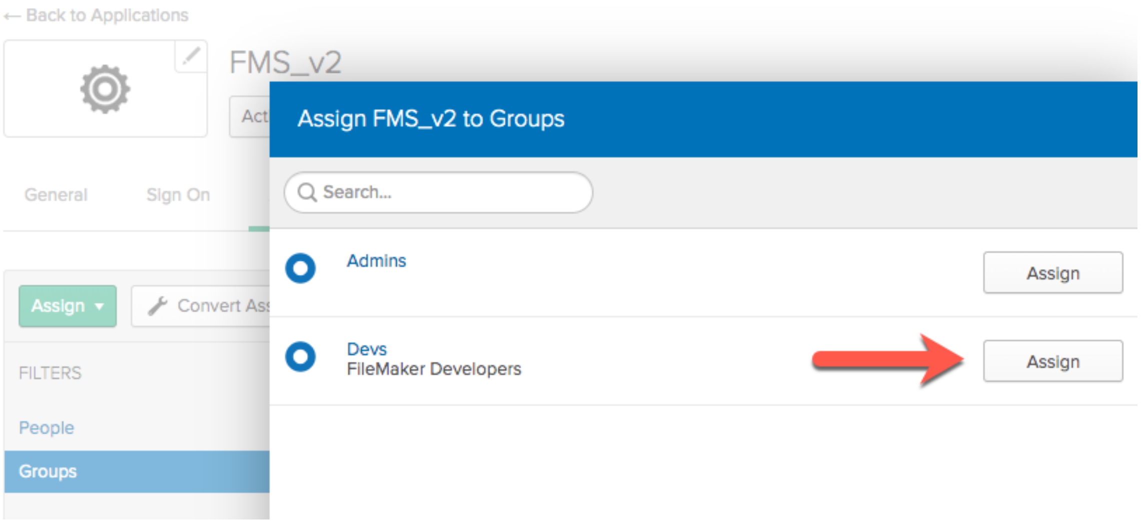


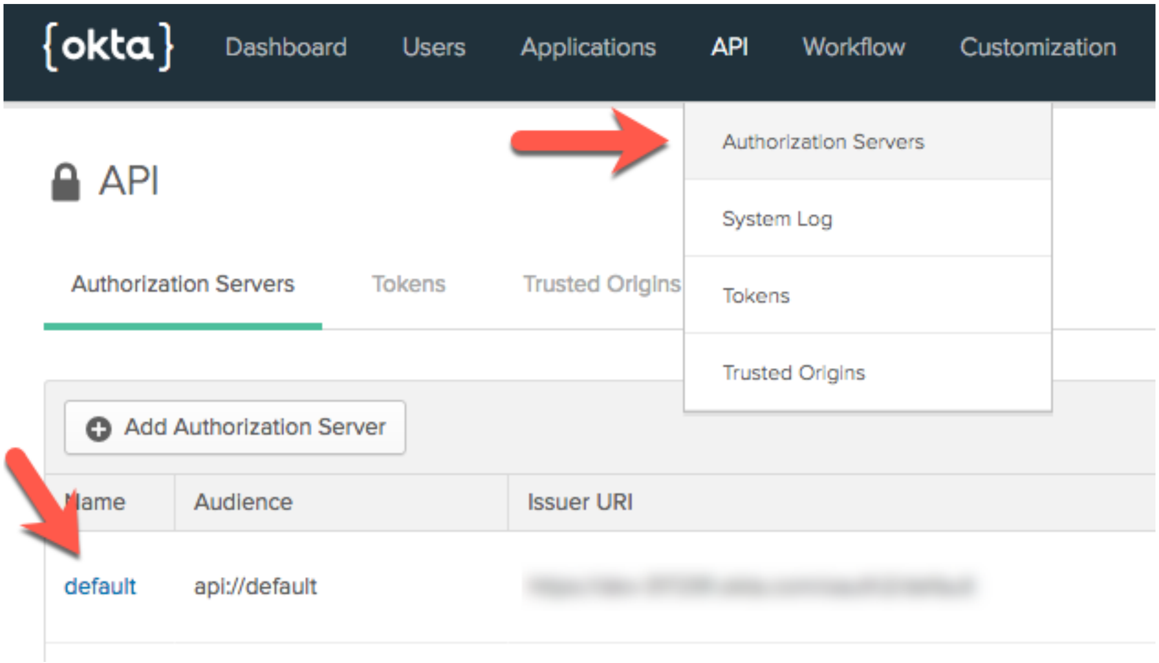
Figure 14. Select a Group.

Up to this point, the process has been fairly standard, and all providers have small variations in how to set up an application and assign users or groups to it, but for the most part it is straightforward.

The next part varies a lot more by provider, this is the part where we need to make sure that the id_token that is returned after a successful authentication by Okta contains the information that FileMaker Server needs. The steps below are specific to Okta and will be completely different for other providers, but the concept is the same for all of them.

The required configuration is done through "*scopes*" and "*claims*." A **scope** is a set of attributes that describes a user, each of those attributes is called a claim. Every OpenID Connect exchange requires the "*openid*" scope, which will return the basic claims that make the exchange work. Since we want to include group data we need to add a groups scope and a groups claim.

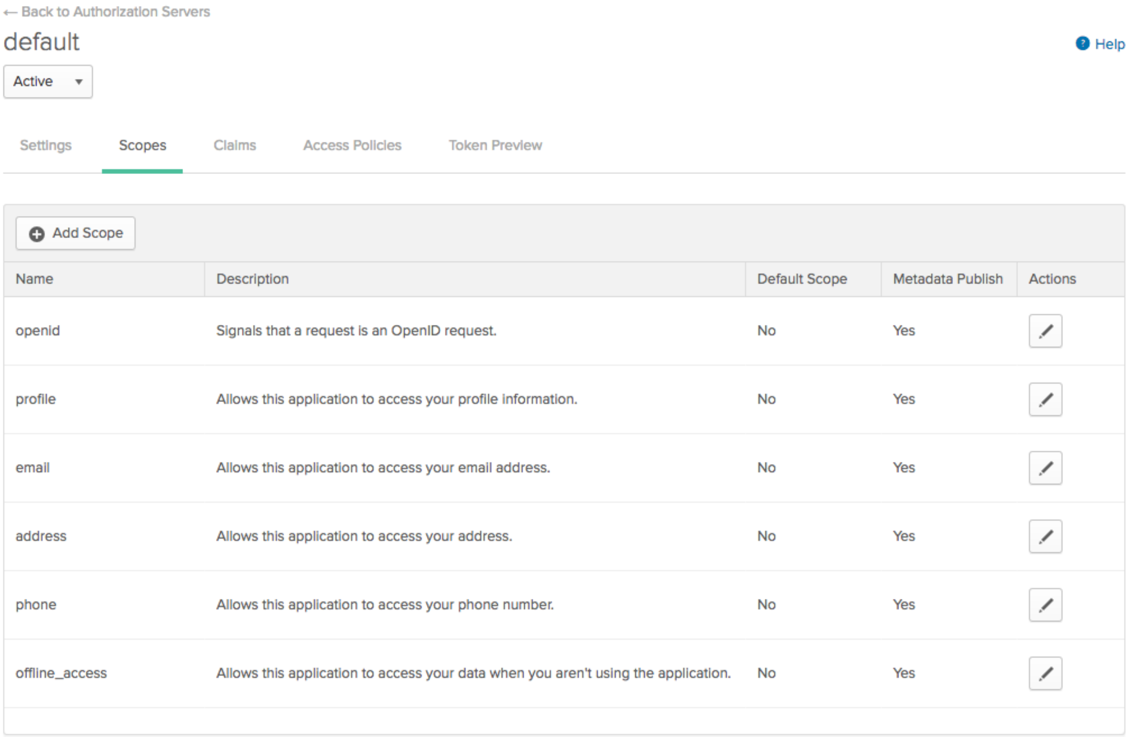
In Okta, select API > Authorization Servers and select the Default server.



Name	Audience	Issuer URI
default	api://default	[redacted]

Figure 15. Scopes and Claims.

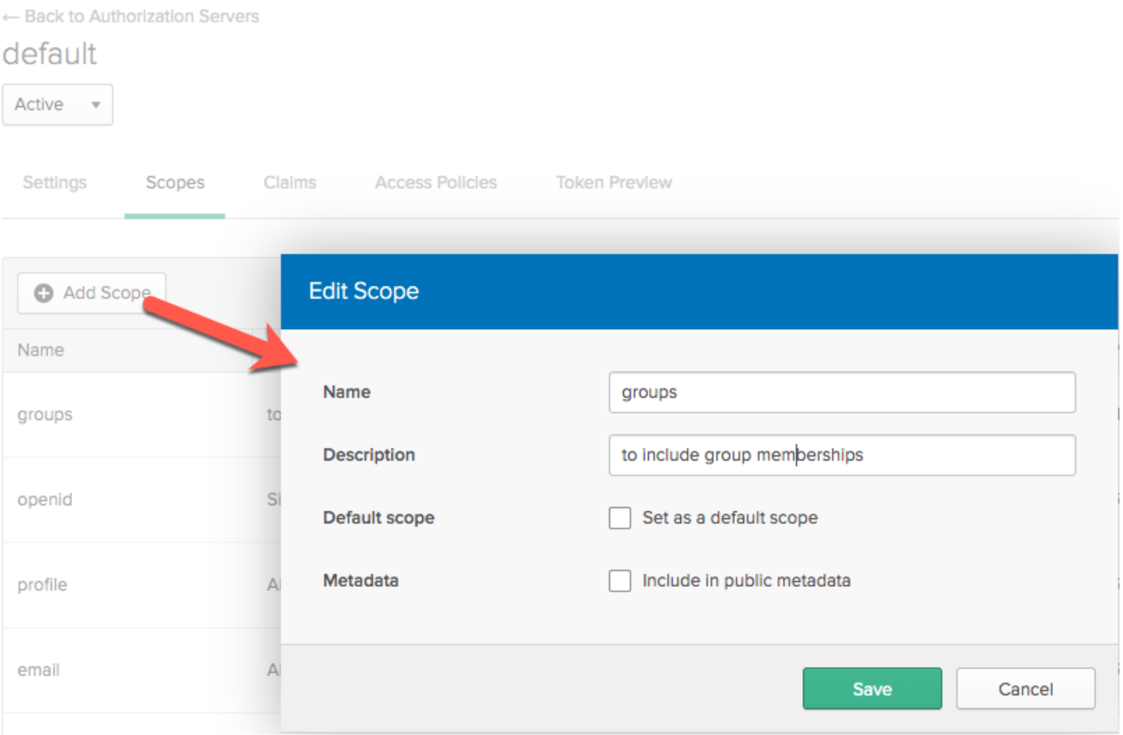
Under “scopes” you see what scopes Okta offers by default:



Name	Description	Default Scope	Metadata Publish	Actions
openid	Signals that a request is an OpenID request.	No	Yes	✎
profile	Allows this application to access your profile information.	No	Yes	✎
email	Allows this application to access your email address.	No	Yes	✎
address	Allows this application to access your address.	No	Yes	✎
phone	Allows this application to access your phone number.	No	Yes	✎
offline_access	Allows this application to access your data when you aren't using the application.	No	Yes	✎

Figure 16. Default Scopes.

The scope labeled “*email*” is one that we need for FileMaker Server to identify the user. But there is no “groups” scope so we will need to add one:



Name groups

Description to include group memberships

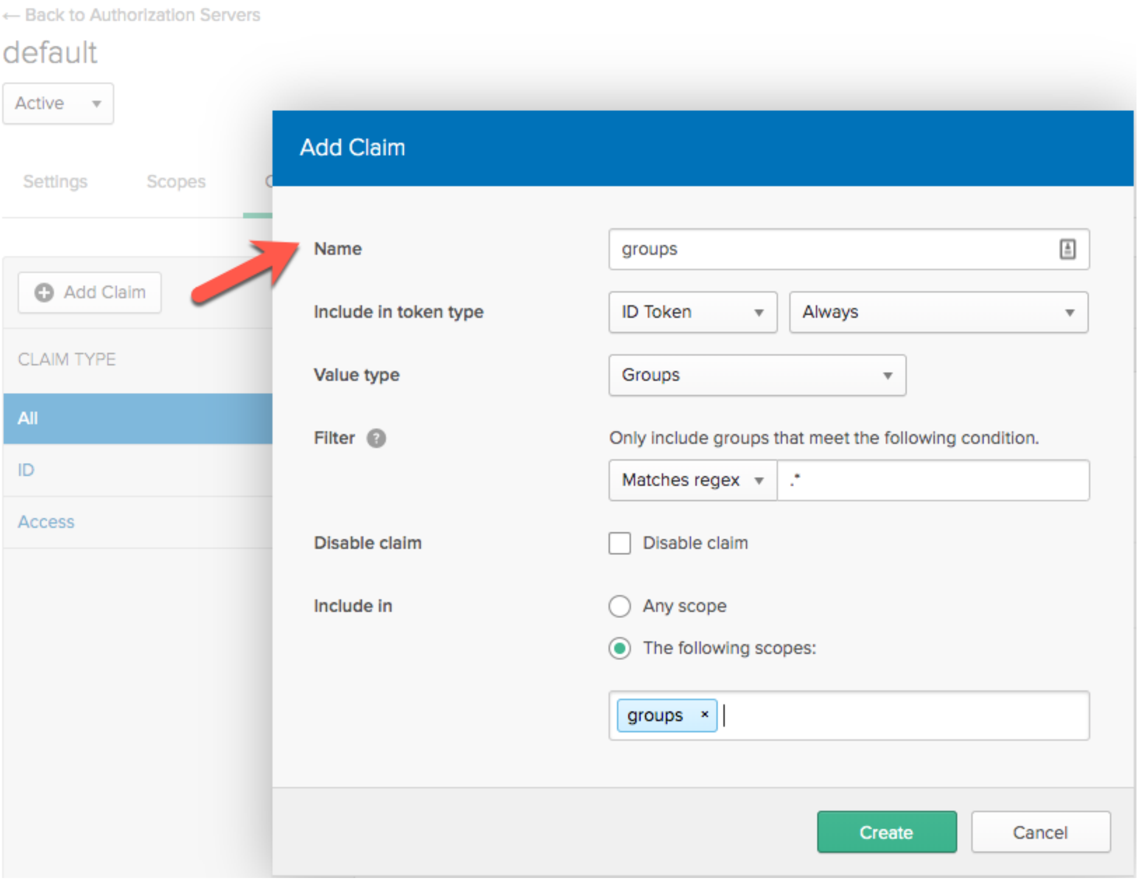
Default scope ☐ Set as a default scope

Metadata ☐ Include in public metadata

Figure 17. Adding Groups Scope.

And then under “claims” (*Figure 18*), we will add a claim for the Groups data itself and link the claim to the scope. The relevant settings are:

- **Include in token type:** this must be set to ID Token (which determines the content of id_token) because that is where FileMaker Server will look for the data.
- **Value Type:** is set to ***groups*** because that is the data we want to include and to make sure we get all the group names, we set the filter to a regular expression that will return everything (“.*”).
- **Include in:** this is where we link this claim to the groups scope we created earlier.



Name: groups

Include in token type: ID Token, Always

Value type: Groups

Filter: Only include groups that meet the following condition.

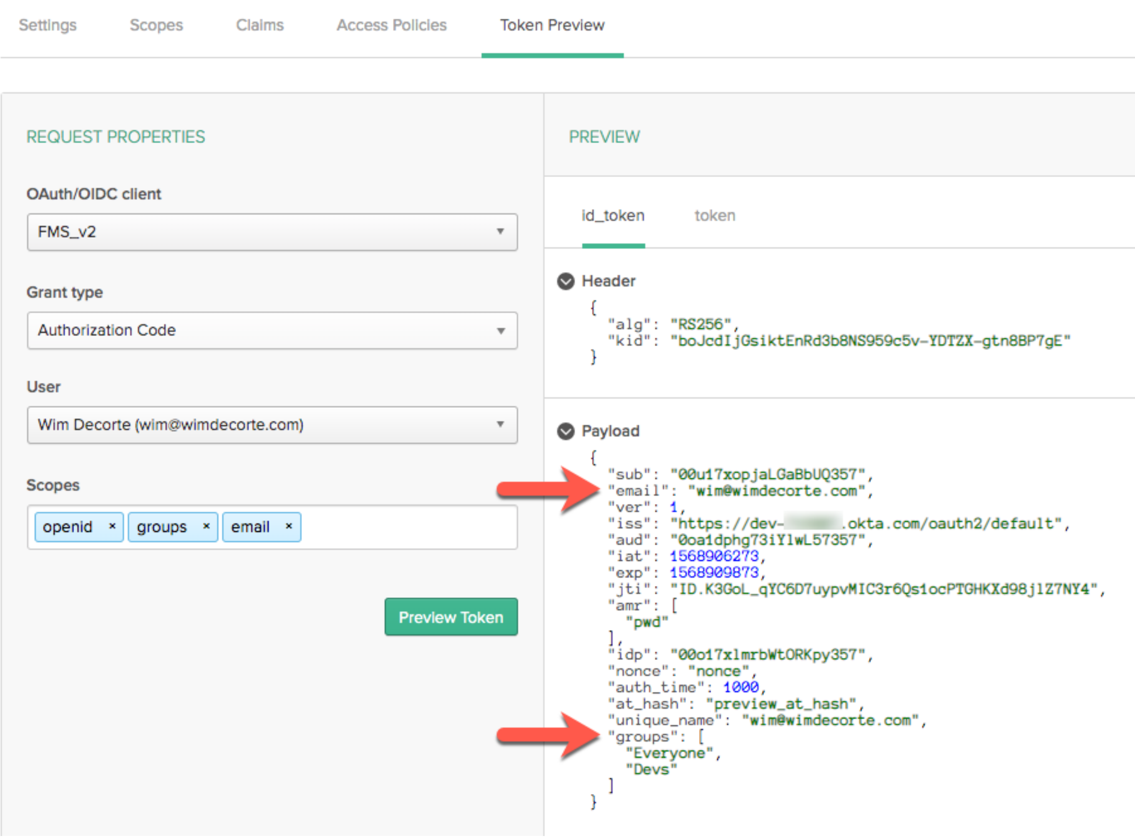
Matches regex .*

Disable claim: ☐ Disable claim

Include in: ○ Any scope ● The following scopes: groups ×

Figure 18. Adding Claims.

One of the nice features in Okta is that we can test the configuration immediately to see what token it will produce. For that, go to the “Token Preview” section and select your application, set Grant Type to “Authorization Code” (as you remember from the flow, steps 5 and 6 indicate that an authorization code exchange is happening). Next select a user that belongs to the Devs group that was assigned to the application and finally add “openid”, “groups” and “email” to the scopes. The Payload under *id_token* on the right-hand side shows what FileMaker Server will receive. And it does contain the email and the list of groups in exactly the format that we know will work.



OAuth/OIDC client
FMS_v2

Grant type
Authorization Code

User
Wim Decorte (wim@wimdecorte.com)

Scopes
openid × groups × email ×

Header

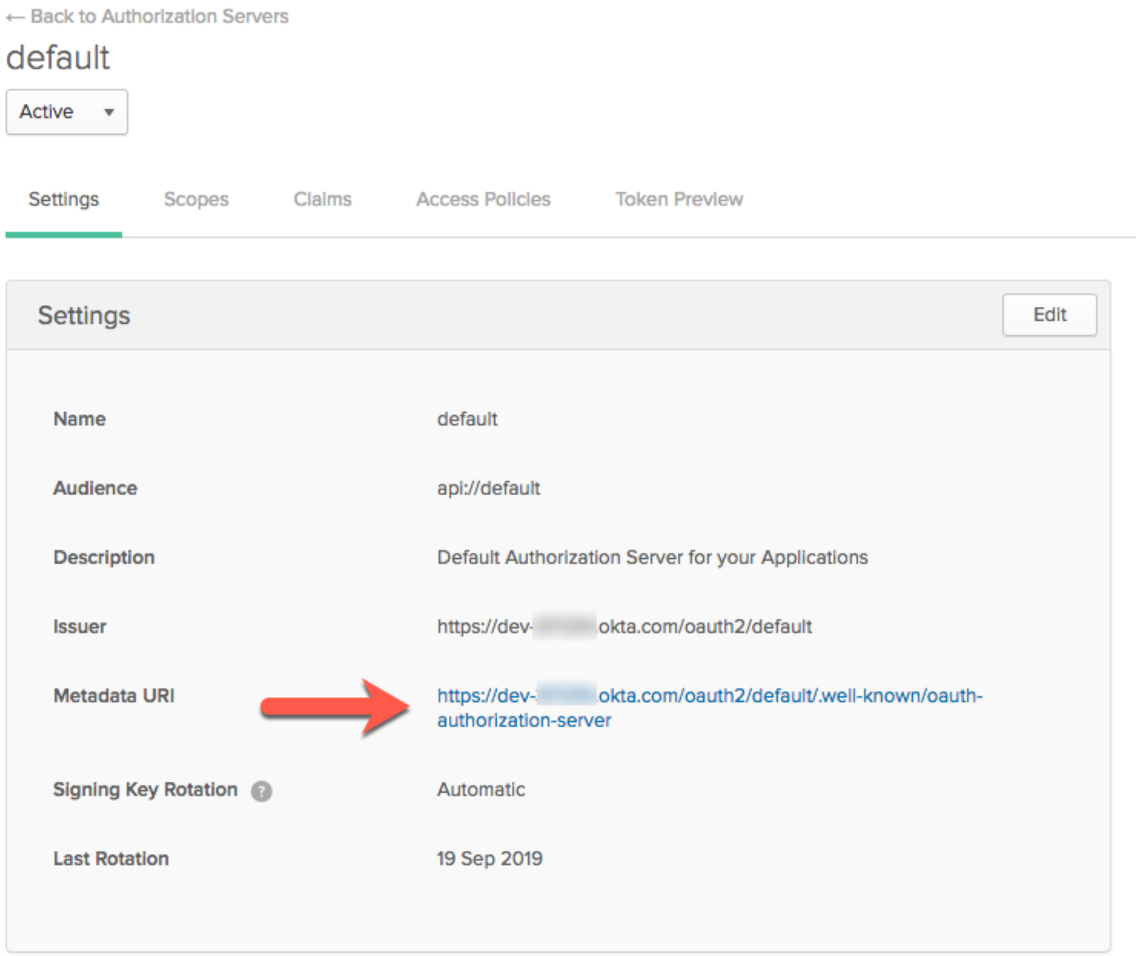
```
{
  "alg": "RS256",
  "kid": "boJcdIjGsiktEnRd3b8NS959c5v-YDTZX-gtn8BP7gE"
}
```

Payload

```
{
  "sub": "00u17xopjaLGaBbUQ357",
  "email": "wim@wimdecorte.com",
  "ver": 1,
  "iss": "https://dev-[redacted].okta.com/oauth2/default",
  "aud": "0oa1dphg73iYlwL57357",
  "iat": 1568906273,
  "exp": 1568909873,
  "jti": "ID.K3GoL_qYC6D7uypvMIC3r6Qs1ocPTGHKXd98j1Z7NY4",
  "amr": [
    "pwd"
  ],
  "idp": "00o17xlmrbWtORKpy357",
  "nonce": "nonce",
  "auth_time": 1000,
  "at_hash": "preview_at_hash",
  "unique_name": "wim@wimdecorte.com",
  "groups": [
    "Everyone",
    "Devs"
  ]
}
```

Figure 19. The Payload Result.

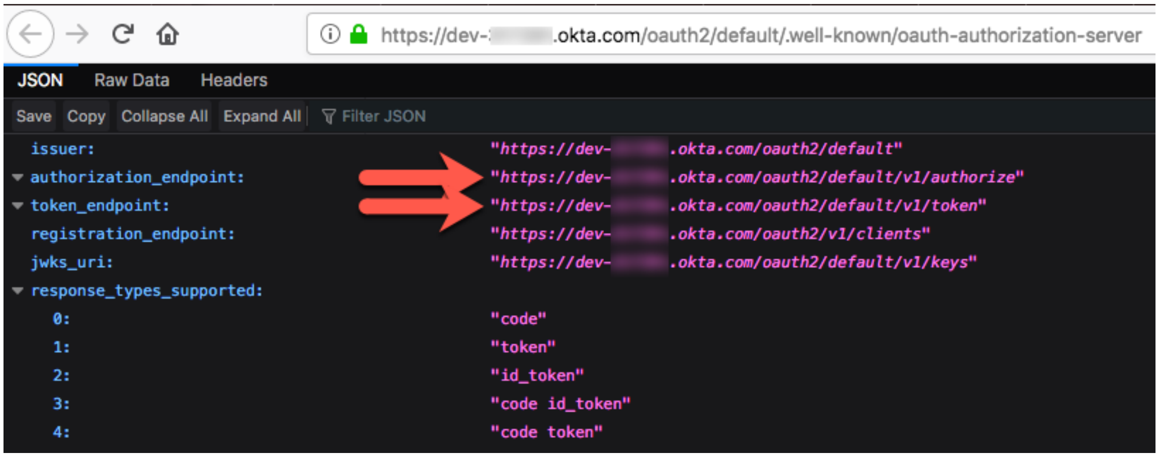
While we are on the Authentication Server settings, we can grab the URLs for the *Authentication Request* and *Token Request*, we will need them to configure FileMaker Server. From the "Settings" section, click on the link for the metadata:



Name	default
Audience	api://default
Description	Default Authorization Server for your Applications
Issuer	https://dev-[redacted]okta.com/oauth2/default
Metadata URI	https://dev-[redacted]okta.com/oauth2/default/.well-known/oauth-authorization-server
Signing Key Rotation	Automatic
Last Rotation	19 Sep 2019

Figure 20. Authentication Request and Token Request.

That link will display some json information about the authentication server and from it we will note the two URLs:


https://dev-[redacted].okta.com/oauth2/default/.well-known/oauth-authorization-server
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON

```
  issuer:                      "https://dev-[redacted].okta.com/oauth2/default"
▼ authorization_endpoint:      "https://dev-[redacted].okta.com/oauth2/default/v1/authorize"
▼ token_endpoint:              "https://dev-[redacted].okta.com/oauth2/default/v1/token"
  registration_endpoint:       "https://dev-[redacted].okta.com/oauth2/v1/clients"
  jwks_uri:                    "https://dev-[redacted].okta.com/oauth2/default/v1/keys"
▼ response_types_supported:
    0:                         "code"
    1:                         "token"
    2:                         "id_token"
    3:                         "code id_token"
    4:                         "code token"
```

Figure 21. JSON Information.

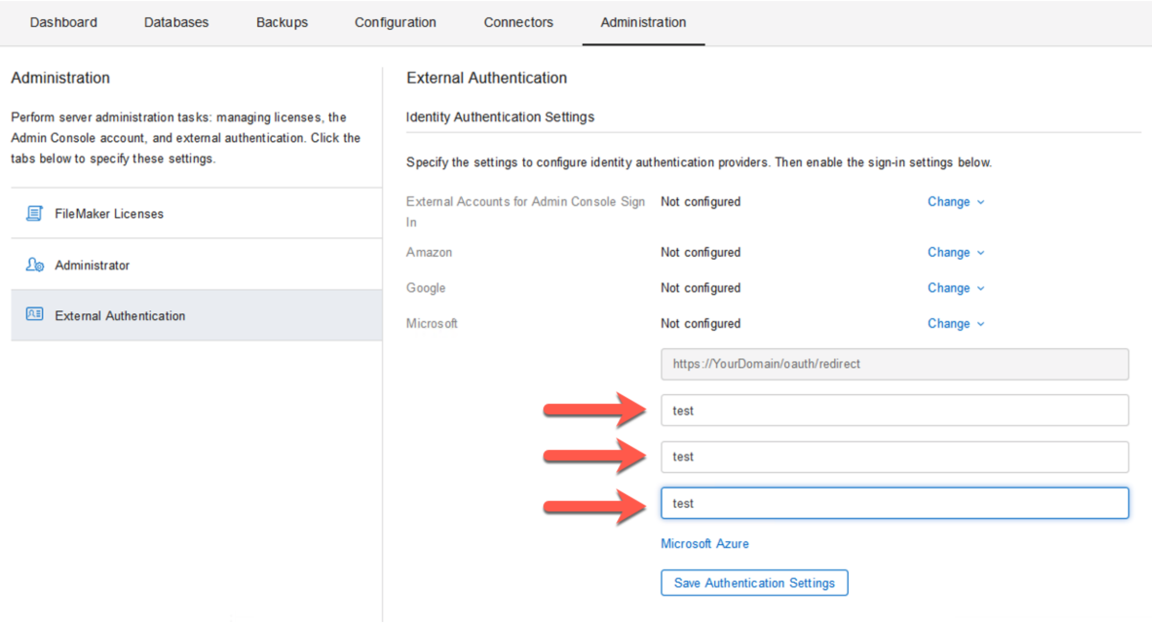
At this point we know that Okta is configured correctly to authenticate users for the application we created and that it will return a token that FileMaker Server expects. Next, we need to configure FileMaker Server itself to participate in this authentication flow.

Configuring FileMaker Server

Note: the steps below involve stopping the database server, so only do this in a production machine when it will not inconvenience your users!

The FileMaker Server Admin Console allows us to configure the three traditional OAuth providers but there is no UI to add other ones. Since the Microsoft Azure AD provider is the only one of those three pre-configured ones that contains groups, we will use that one as the basis for our setup.

Click "change" next to the Microsoft entry and type in some meaningless text for each of the provided spaces. Our only intent here is for FileMaker Server to generate the necessary configuration entries.



Administration

Perform server administration tasks: managing licenses, the Admin Console account, and external authentication. Click the tabs below to specify these settings.

External Authentication

Identity Authentication Settings

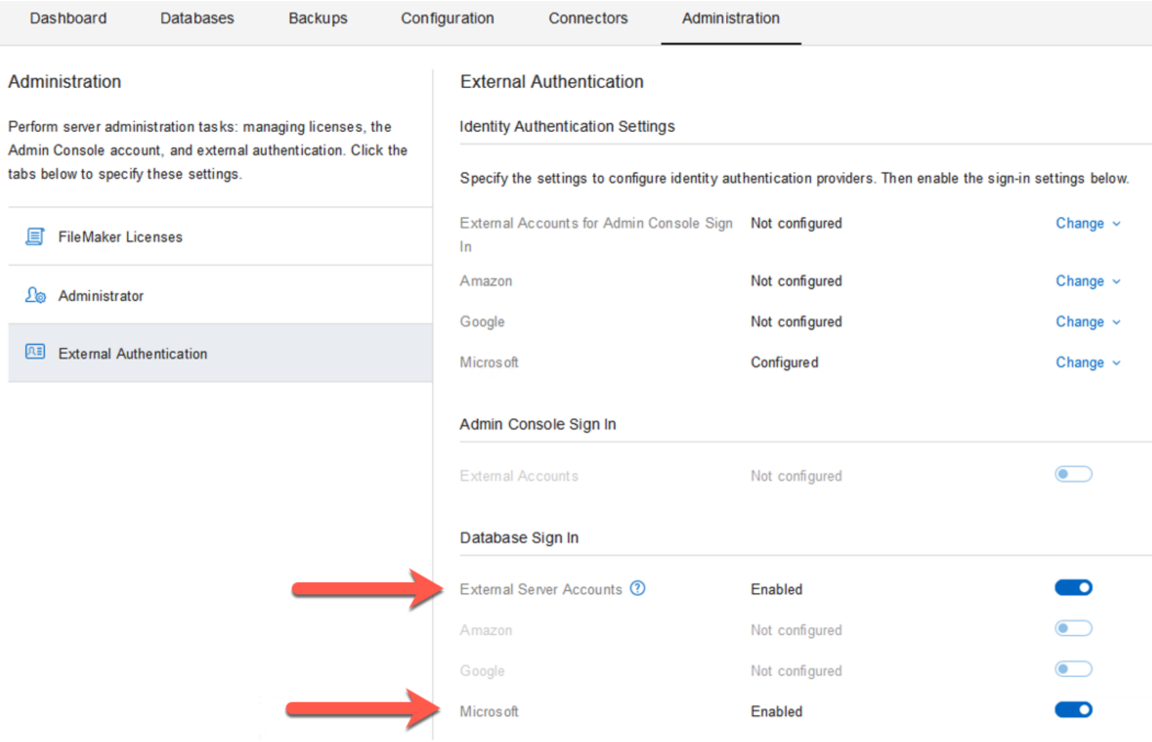
Specify the settings to configure identity authentication providers. Then enable the sign-in settings below.

External Accounts for Admin Console Sign In: Not configured
Amazon: Not configured
Google: Not configured
Microsoft: Not configured

https://YourDomain/oauth/redirect
test
test
test

Figure 22. Configuring the FileMaker Server External Authentication.

Save the settings and in the bottom section enable "FileMaker and External Accounts" and the Microsoft entry:



Administration

Perform server administration tasks: managing licenses, the Admin Console account, and external authentication. Click the tabs below to specify these settings.

External Authentication

Identity Authentication Settings

Specify the settings to configure identity authentication providers. Then enable the sign-in settings below.

External Accounts for Admin Console Sign In: Not configured
Amazon: Not configured
Google: Not configured
Microsoft: Configured

Admin Console Sign In

External Accounts: Not configured

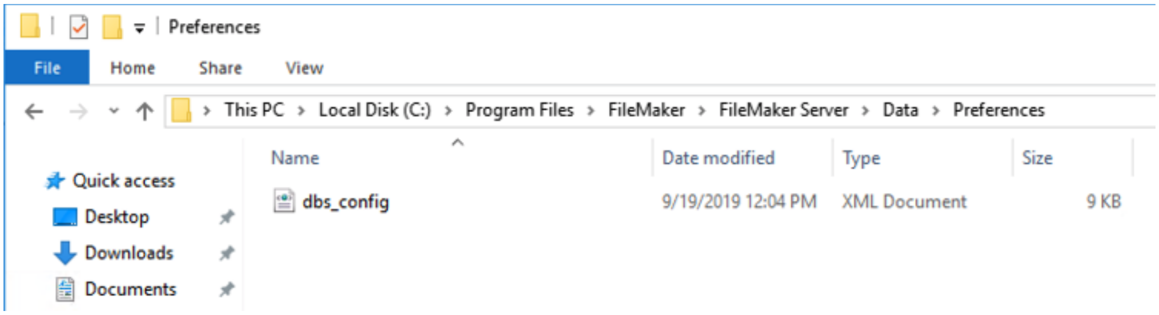
Database Sign In

External Server Accounts: Enabled
Amazon: Not configured
Google: Not configured
Microsoft: Enabled

Figure 23. Saving the New Settings.

Close all the hosted files and stop the database server. Stopping the database server is crucial, otherwise all configuration changes discussed below will **not** be saved.

Using Finder on macOS or Windows Explorer on Windows, navigate to the FileMaker Server folder, and go to the Preferences folder inside the Data folder. There is a file named **dbs_config.xml** that we will edit:

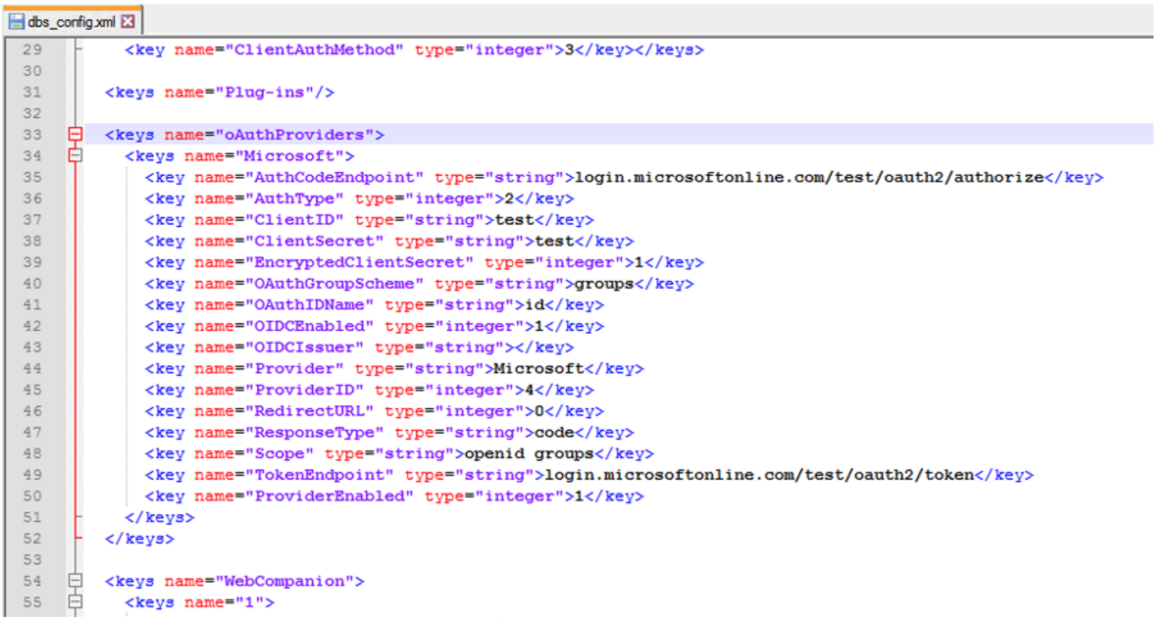


This PC › Local Disk (C:) › Program Files › FileMaker › FileMaker Server › Data › Preferences

Name	Date modified	Type	Size
dbs_config	9/19/2019 12:04 PM	XML Document	9 KB

Figure 24. Locating the XML file.

Open that file with a **text editor—not with a word processor**— and scroll down in that file until you see the section named "**oAuthProviders**" as shown in *Figure 25.*



```
    <key name="ClientAuthMethod" type="integer">3</key></keys>

  <keys name="Plug-ins"/>

  <keys name="oAuthProviders">
    <keys name="Microsoft">
      <key name="AuthCodeEndpoint" type="string">login.microsoftonline.com/test/oauth2/authorize</key>
      <key name="AuthType" type="integer">2</key>
      <key name="ClientID" type="string">test</key>
      <key name="ClientSecret" type="string">test</key>
      <key name="EncryptedClientSecret" type="integer">1</key>
      <key name="OAuthGroupScheme" type="string">groups</key>
      <key name="OAuthIDName" type="string">id</key>
      <key name="OIDCEnabled" type="integer">1</key>
      <key name="OIDCIssuer" type="string"></key>
      <key name="Provider" type="string">Microsoft</key>
      <key name="ProviderID" type="integer">4</key>
      <key name="RedirectURL" type="integer">0</key>
      <key name="ResponseType" type="string">code</key>
      <key name="Scope" type="string">openid groups</key>
      <key name="TokenEndpoint" type="string">login.microsoftonline.com/test/oauth2/token</key>
      <key name="ProviderEnabled" type="integer">1</key>
    </keys>
  </keys>

  <keys name="WebCompanion">
    <keys name="1">
```

Figure 25. The "OAuth Providers" Section of the dbs_config.xml file.

Be careful making changes to this file beyond what we describe below, as it may invalidate your FileMaker Server settings overall. Also do not modify this file while FileMaker Server is still running, or your changes will be lost.

The settings that you need to modify are highlighted in *Figure 26*. This is where all the configuration options and concepts that we have discussed come together.

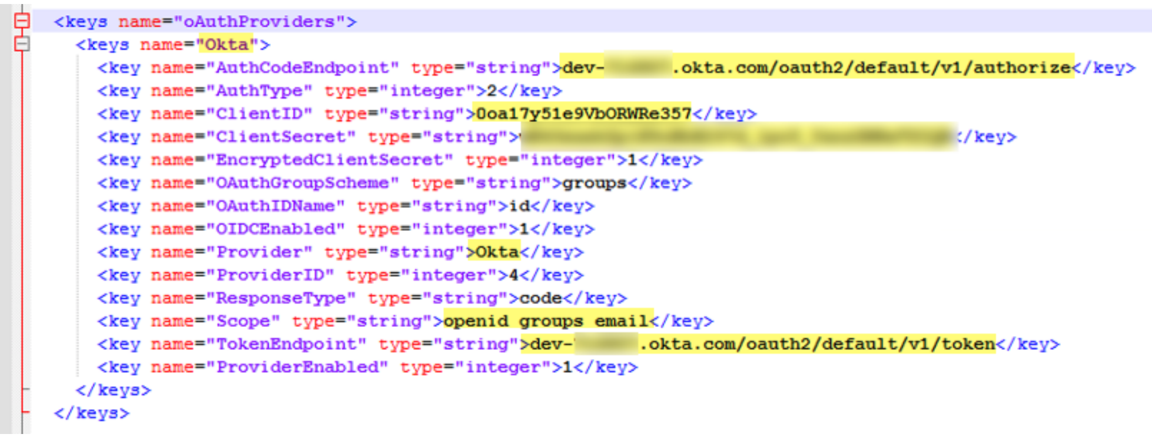
There are a few settings that **absolutely must not be modified**, like the **ProviderID**; that one must remain "4." **ResponseType** should always be "code" and **AuthType** must be "2". If your provider does not support OpenId Connect then there are a few more settings that need to be configured including **OIDCEnabled** set to 0, and a valid **ProfileEndpoint** and **OAuthIDName.** The need for these extra configuration settings is why we recommend using OpenId Connect if it is supported by your provider.

There are also subtle differences between FileMaker Server 17 and 18 in what keys exist in that section. For instance, FileMaker Server 17 has a "logo" key that does not exist in 18. You cannot change the logo in any case; the Microsoft logo will always be shown. It will require an update to FileMaker Server to resolve this. Hopefully something that we will see in the next version of FileMaker Server.

From top to bottom the required changes are:

- The **AuthCodeEndPoint** is where FileMaker Server will send the initial authentication request (step 2 on page 7). The URL is outlined in the Okta documentation[17] and we took a note of it while we configured the authentication server (see figure 21 on page 23).

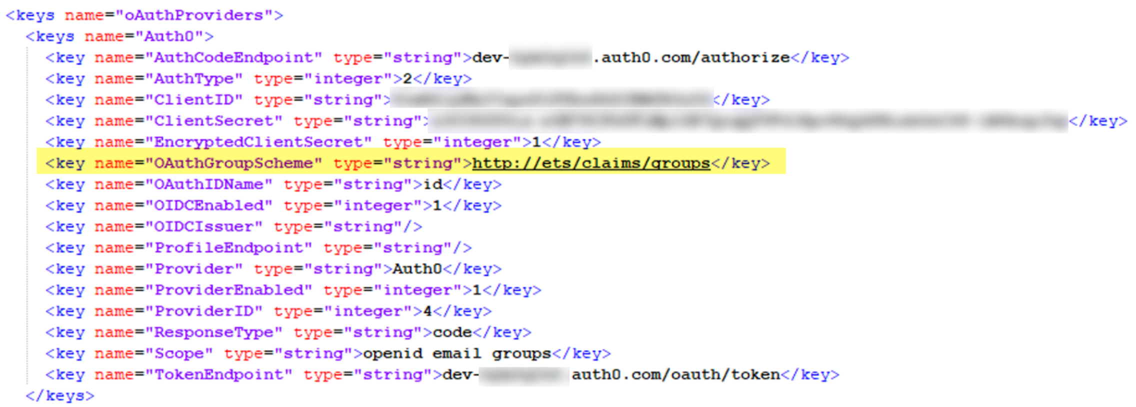
- **ClientID** is from the Okta application (see *Figure 11* on page 15)
- **ClientSecret** is from the Okta application (see *Figure 11* on page 15)
- **Provider** can be any string you want
- **Scope**: use the scopes that you tested with earlier (see *Figure 17* on page 19)
- **TokenEndPoint** is the 2nd URL that we retrieved from Okta's authentication server's metadata (see figure 21 on page 23)



```
<keys name="oAuthProviders">
  <keys name="Okta">
    <key name="AuthCodeEndpoint" type="string">dev-[redacted].okta.com/oauth2/default/v1/authorize</key>
    <key name="AuthType" type="integer">2</key>
    <key name="ClientID" type="string">0oa17y51e9VbORWRe357</key>
    <key name="ClientSecret" type="string">[redacted]</key>
    <key name="EncryptedClientSecret" type="integer">1</key>
    <key name="OAuthGroupScheme" type="string">groups</key>
    <key name="OAuthIDName" type="string">id</key>
    <key name="OIDCEnabled" type="integer">1</key>
    <key name="Provider" type="string">Okta</key>
    <key name="ProviderID" type="integer">4</key>
    <key name="ResponseType" type="string">code</key>
    <key name="Scope" type="string">openid groups email</key>
    <key name="TokenEndpoint" type="string">dev-[redacted].okta.com/oauth2/default/v1/token</key>
    <key name="ProviderEnabled" type="integer">1</key>
  </keys>
</keys>
```

Figure 26. Changing Values For New Provider.

If your chosen provider requires name-spacing the groups claim, then there is one additional key to update: **OAuthGroupScheme**. In that key's value include the complete name-spaced string that you have configured on the provider's side. This is our Auth0 configuration as an example:



```
<keys name="oAuthProviders">
  <keys name="Auth0">
    <key name="AuthCodeEndpoint" type="string">dev-[redacted].auth0.com/authorize</key>
    <key name="AuthType" type="integer">2</key>
    <key name="ClientID" type="string">[redacted]</key>
    <key name="ClientSecret" type="string">[redacted]</key>
    <key name="EncryptedClientSecret" type="integer">1</key>
    <key name="OAuthGroupScheme" type="string">http://ets/claims/groups</key>
    <key name="OAuthIDName" type="string">id</key>
    <key name="OIDCEnabled" type="integer">1</key>
    <key name="OIDCIssuer" type="string"/>
    <key name="ProfileEndpoint" type="string"/>
    <key name="Provider" type="string">Auth0</key>
    <key name="ProviderEnabled" type="integer">1</key>
    <key name="ProviderID" type="integer">4</key>
    <key name="ResponseType" type="string">code</key>
    <key name="Scope" type="string">openid email groups</key>
    <key name="TokenEndpoint" type="string">dev-[redacted]auth0.com/oauth/token</key>
  </keys>
```

Figure 27. Adding name-spacing to the groups claim.

---

[17] For the correct URL syntax: https://developer.okta.com/docs/reference/api/oidc/#_1-single-sign-on-to-okta

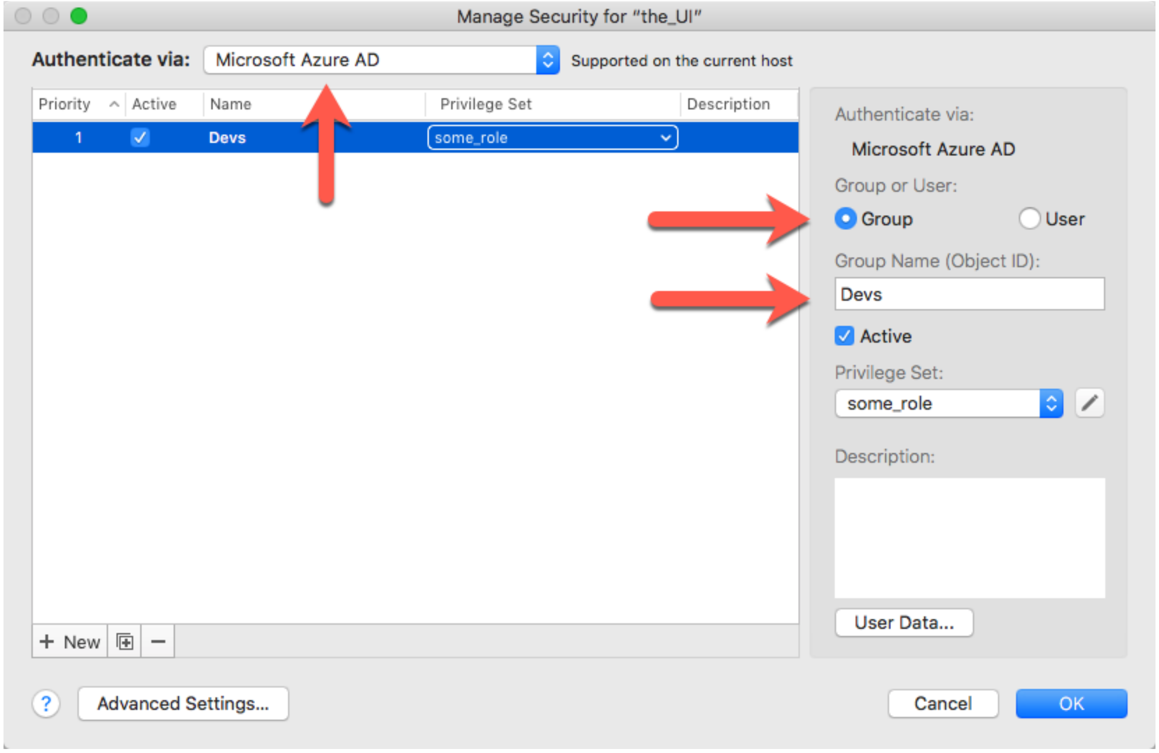
Save the edits and start FileMaker Server again. Open the xml file again to verify that the edits are still there and have not been overwritten by FileMaker Server on restart.

In the FileMaker Server Admin Console, the interface will revert back to show that no OAuth providers have been configured, but that is entirely normal.

Configuring the FileMaker Pro file

The final part is to make the required changes in the FileMaker file. There we need to create groups that match the relevant groups that exist in Okta.

In "Manage Security", select "Microsoft Azure AD" as the provider and add a group account that matches ***exactly*** the name of the group as it exists in Okta. This is shown below.



Manage Security for "the_UI"

Authenticate via: Microsoft Azure AD

Supported on the current host

Priority	Active	Name	Privilege Set	Description
1	☑	**Devs**	some_role	

Authenticate via: Microsoft Azure AD

Group or User: ● Group ○ User

Group Name (Object ID): Devs

☑ Active

Privilege Set: some_role

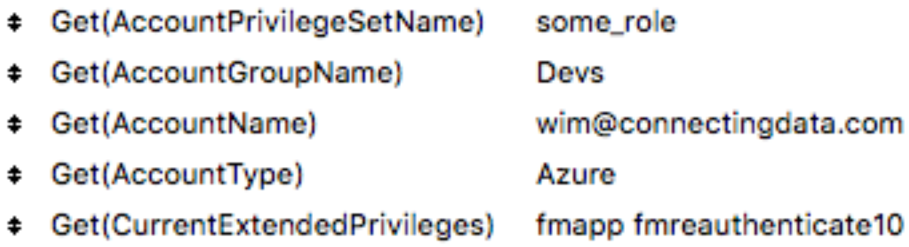
Description:

Figure 28. Configuring the FileMaker Pro file.

If you are working with an IDaaS provider that does not provide the group information or enforces name-spacing on the group claim then you can still create "User" accounts in this section for each individual user.

When a user now logs into this FileMaker file hosted on our configured FileMaker Server and clicks the provider's button, the whole flow as described on page 7 completes successfully.

The Get() functions that provide information about the user's Account and Privilege Set report this:



Get(AccountPrivilegeSetName)	some_role
Get(AccountGroupName)	Devs
Get(AccountName)	wim@connectingdata.com
Get(AccountType)	Azure
Get(CurrentExtendedPrivileges)	fmapp fmreauthenticate10

Figure 29. Account Meta-Data.

The Get(AccountName) is taken from the "email" value in the *id_token*.[18] The list of groups in that *id_token* is matched with the groups in the FileMaker file, and the match was "Devs" which goes into Get(AccountGroupName). The Get(AccountType) is hard-wired to report "Azure" since we are using the Azure AD section of the configuration.

In conclusion: if the IDaaS provider can be configured to return "email" and "groups" claims in the id_token, then you can use that provider with FileMaker Server to do Group-based authentication to your FileMaker apps.

CAVEATS AND "GOTCHAS"

There are a number of considerations that developers and server administrators may wish to bear in mind about the use of oAuth2 providers:

- Integration with the providers requires internet access to and from the FileMaker Server, and that is not always a given in secure deployments. In addition the server needs a custom SSL certificate that covers the fully qualified domain name (FQDN) for the FileMaker Server as set up in DNS that will guide the OAuth provider back to the FileMaker Server machine through the "Allowed Return URL."
- The "client secret" key may have an expiration date. The key will have to

---

[18] Or the "unique_name" claim if there is no "email" claim in the id_token.

be renewed in time; otherwise, it will block the authentication from FileMaker.

- OAuth provider buttons will show up for all files hosted on the configured server. It does not matter that the FileMaker file itself does not use any of them.
- In many instances the IDaaS Provider allows saving of the credentials in the user's browser. Developers and administrators must weigh any risks associated with this. Some providers allow you explicitly to disable this feature.

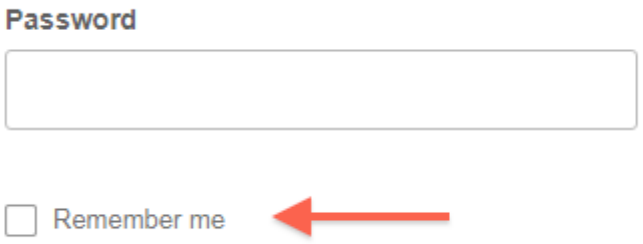
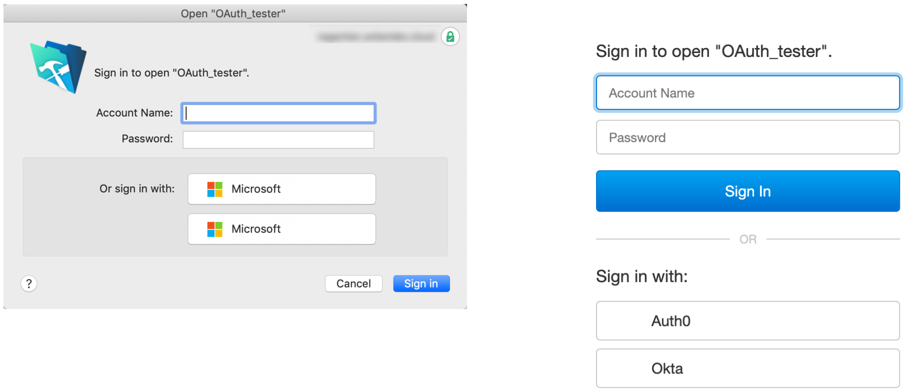


Figure 30. Saving Credentials on Provider Gateway Page.

- There are some configuration options that we cannot update such as the provider's logo: it will always be the Microsoft logo when using FileMaker Pro Advanced and no logo when using WebDirect.
- There are subtle differences between FileMaker Pro Advanced 17 and 18; 17 will show the name of the configured provider but 18 just shows "Microsoft" regardless of what name was used in the dbs_config.xml file.
- It may be obvious but this whole setup only works for files hosted on FileMaker Server. It does not work on FileMaker Cloud since you do not have access to the *dbs_config.xml* file to make the required changes. It should work on FileMaker Cloud for AWS.
- You can add multiple OAuth providers to the dbs_config.xml file. However, since they will all be based on the Microsoft Azure AD "template" the buttons on the FileMaker Pro interface will be confusing as they will all display the Microsoft name and logo. WebDirect does show the names you used in the xml file. This should not typically be an issue on production servers since we expect that only one provider will be used. But it may be an issue on development and test servers.


Sign in to open "OAuth_tester".

Sign in to open "OAuth_tester".

Figure 31. Multiple OAuth providers.

- As mentioned earlier in this document, the Get(AccountType) will always return "Azure" and not the name of your chosen IDaaS provider. See *Figure 29* on Page 28.
- The structure of the id_token is the key to a successful implementation. As mentioned, some providers cannot provide the groups data in the right format.
- The JWT token (see page 9) that is returned by the provider must be in JWS[19] format, not JWE.[20] This means that the token must be *encoded* but *not encrypted*. And the algorithm for the encoding must be HS256. Some providers offer different choices for the algorithm.

In summary, in this White Paper the authors have:

- Revealed a mechanism for extending the oAuth2 based Authentication process to at least five additional IDaaS Providers beyond the original three provided at the release of FileMaker® Server 16.
- Provided a detailed explanation using Okta as an example of how to configure the IDaaS Provider, FileMaker Server, and FileMaker Pro to allow this process to occur.
- Explained the need for Federated Identity Management and the benefits of using it.
- Indicated the hope that these five new IDaaS Providers will enable broader use of the FileMaker Platform including in places where it was previously disqualified.

---

[19] https://tools.ietf.org/html/rfc7515

[20] https://tools.ietf.org/html/rfc7516

ABOUT THE AUTHORS

WIM DECORTE is a Senior Technical Architect at Soliant Consulting, a FileMaker Business Alliance Platinum Member company. He is a leading expert on FileMaker Server, FileMaker Platform integration, and IT infrastructure issues. He is the author of numerous White Papers, Technical Briefs, and BLOG posts (https://www.soliantconsulting.com/author/wdecorte/).

STEVEN H. BLACKWELL is a FileMaker Business Alliance Platinum Member Emeritus. He is the author of *FileMaker Security: The Book* as well as numerous White Papers and Technical Briefs about FileMaker Platform Security. He is also the creator of the FileMaker Security BLOG (https://fmforums.com/blogs/blog/13-filemaker-security-blog)

ACKNOWLEDGMENTS

The authors appreciate the assistance and information we received from various persons at Claris International, Inc. Their work was the impetus for this White Paper. The FileMaker Developer Community and the customer base both owe them a large debt of gratitude.

Claris, Inc. (formerly FileMaker, Inc.)—

Ronnie Rios, Senior Consulting Engineer

Fan Zhao, Software Engineer

Darrin Quick, Senior Business Account Manager

Sangita Bannerjee, Manager, Product Management

Rick Kalman, Director of Product Management