Multiple Organization Architecture (Multitenancy): Isolated Users by Organization

An Auth0 Integration Guide to Architectures that must Accommodate Application Instances for Multiple Organizations or Brands

Table of contents

Introduction

This document will cover the Isolated Users by Organization use case as described in the Multiple Organization Architecture (Multitenancy): Overview document.

If there will be no user sharing between organizations, then you can gain benefit by segregating your users into their own database connection, enterprise connection and/or custom social connection. There are a few different approaches that can be taken in this situation, so before we dive into any solution, let's figure out where your application lands.

*If **all** of your applications know their organization ahead of time and **all or most** of your organizations have more than one IDP associated with them. You may want to give each organization its own Auth0 tenant. Similar to Create Separate Auth0 Tenant for each Organization.*

We will use the Travel0 Corporate Booking application that is described in the introduction in the overview document. Travel0 would be the Auth0 customer.

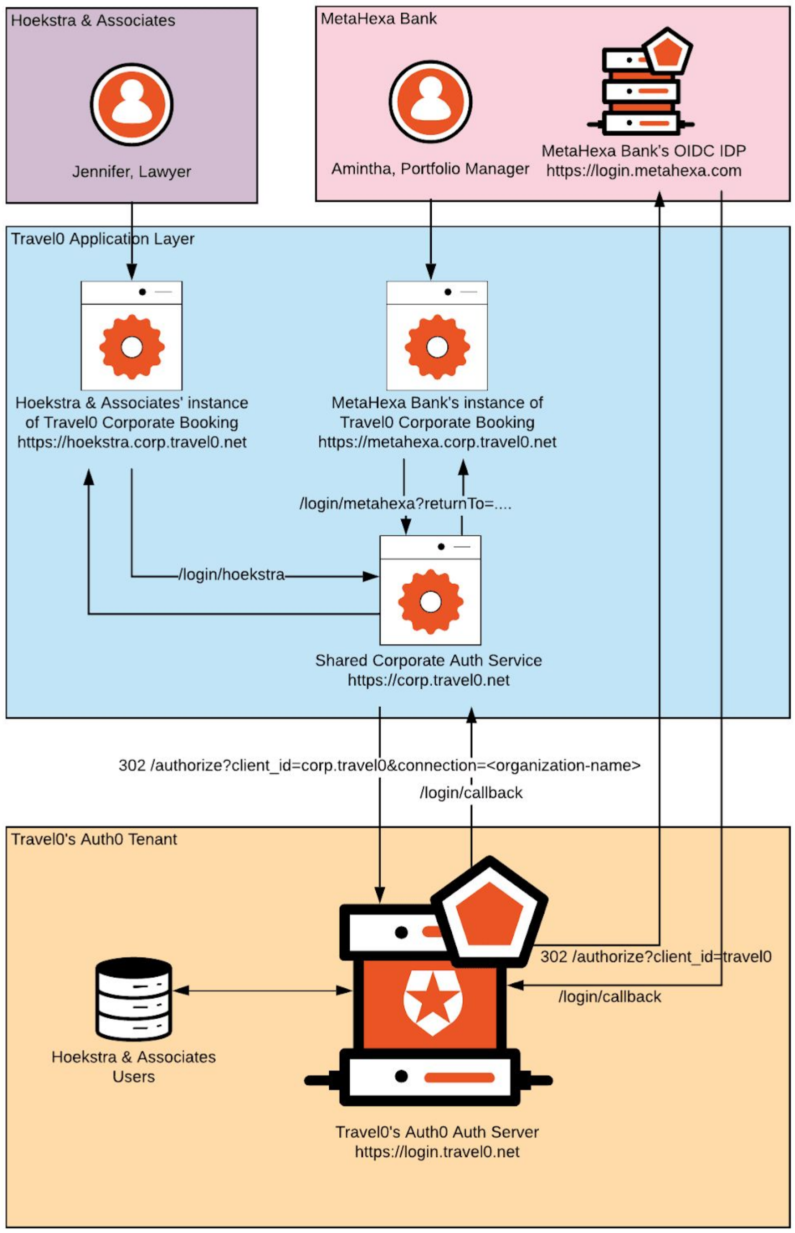
Most applications that have isolation per organization have three different kinds of organizations:

- Organizations that either don't have or don't know how to use their own IDP. These organizations tend to be smaller organizations that don't have an IT department to configure SSO with the organization's IDP. The example organizations from our Travel0 example that fit into this category are Jennifer Zero Law and Gupta & Smith Law.
- Organizations that would prefer to configure their IDP for your application so their employees don't have to create a new set of credentials for your application. Most organizations fall into this category. Our example organization is MetaHexa Bank.
- Organizations that need to have multiple different authentication options for their organization. Examples of this include larger organizations that have multiple acquisitions or frequently acquire new companies; organizations like schools that might have staff and parents logging into the same application; or organizations that invite partners or customers (a.k.a B2B2C) to log into their application instance. The example organization from Travel0 is University0.

In the first two types of organizations the solution tends to be fairly simple. These organizations are considered Single IDP organizations and how you approach it is almost always the same. See Single IDP Organizations for more detailed information. Those organizations that are more complex and have more than one IDP for the organization tend to cause a higher order of complexity and there are a few different ways to approach it to keep the complexity to a minimum. If your company is like most, this set of customers either doesn't exist or is in the minority. See Multiple IDP Organizations for more detailed information.

Single IDP Organizations

For single IDP organizations you can keep your system fairly simple. In this scenario you have exactly one Auth0 connection for every organization. This keeps your authentication simple as each application can send the organization information to the Auth0 Tenant (Authorization Server) and it can then either collect credentials, or forward the request on to the appropriate IDP to handle the request.


Hoekstra & Associates
Jennifer, Lawyer
MetaHexa Bank
Amintha, Portfolio Manager
MetaHexa Bank's OIDC IDP
https://login.metahexa.com
Travel0 Application Layer
Hoekstra & Associates' instance of Travel0 Corporate Booking
https://hoekstra.corp.travel0.net
MetaHexa Bank's instance of Travel0 Corporate Booking
https://metahexa.corp.travel0.net
/login/metahexa?returnTo=....
/login/hoekstra
Shared Corporate Auth Service
https://corp.travel0.net
302 /authorize?client_id=corp.travel0&connection=<organization-name>
/login/callback
Travel0's Auth0 Tenant
302 /authorize?client_id=travel0
/login/callback
Hoekstra & Associates Users
Travel0's Auth0 Auth Server
https://login.travel0.net

From your applications' point of view, it doesn't need to know how credentials are stored - i.e. the application behaves the same whether the users are stored in a database connection, like Jennifer Zero Law from our Travel0 example, or in an enterprise connection like MetaHexa Bank from the Travel0 example. However, there are some unique things you need to do with how you setup your Auth0 tenant and supporting code for each different type. See Database Organizations and Enterprise IDP Organizations for more details.

We will be continuing to use the Travel0 Corporate Booking application for an example. This application should be what you equate to your application and Travel0 should be the example for your company. For more details you can see the introduction in the Multiple Organization Architecture (Multitenancy): Overview document.

Database Organizations

Many B2B applications support organizations that either don't have an employee IDP, or are small enough that they don't have an IT department or person who knows enough about IAM that they can configure it to work with the application. These organizations need to have their users get a new username and password for your application (or suite of applications). We are representing this as Hoekstra & Associates.

Let's now walk through each IAM workstream to figure out how we should address this type of organization.

Architecture

A few requirements to consider for these users:

- The organization may have different requirements or desires around password complexity.
- If the organization leaves, you want to easily decommission the users.
- You want to limit error messages by limiting opportunity to accidentally attempt to log into the wrong organization (example: prefer a "wrong username/password" error over "forbidden to access this organization" error).
- Make sure error messages are consistent across organizations

The best practice to address these requirements is to create exactly one database connection for each organization. We will now see how this plays out with each of the other workstreams.

Provisioning

There are two aspects of provisioning for this use case. The first is the provisioning of the organization itself. As in, what you need to do every time you add a new organization. The second is the provisioning of new and existing users for that organization.

Provisioning the Organization

When provisioning an organization you will need to do the following:

- Add the organization to your application configuration or database.
- In your Auth0 tenant:
    - Create a new Database connection for this organization
    - [Optional] Provision the first user as an administrator for the organization
- Either ensure you named the database connection with a consisting naming convention so that you can construct it using the organization name, or add the exact connection name to your application configuration or database. This will allow you to either

construct the connection name on the fly to pass to the /authorize endpoint, or look it up in configuration.
- Configure the branding on the Universal Login Page

When migrating from an existing application that already has an organization defined and users are already using the system you will need to do the following:
- Mark this tenant as migrated to Auth0 in your application configuration
- Create a new Database connection for this organization and setup migration for the users. (Make sure you have a way to note administrators).
- Configure the branding on the Universal Login Page

Provisioning new Organization Users

If there will be a large number of organizations and organization users that are using a database connection, it can be a scalability problem to manage the addition of those users. Since you are greatly limiting who is allowed to have access, it is common to have an administrator add the users. This Administrator can either be an employee of your company, or one of the organization users that has been given this privilege. The latter is often done for scalability reasons, but requires you to provide the organization with a tool for adding users.

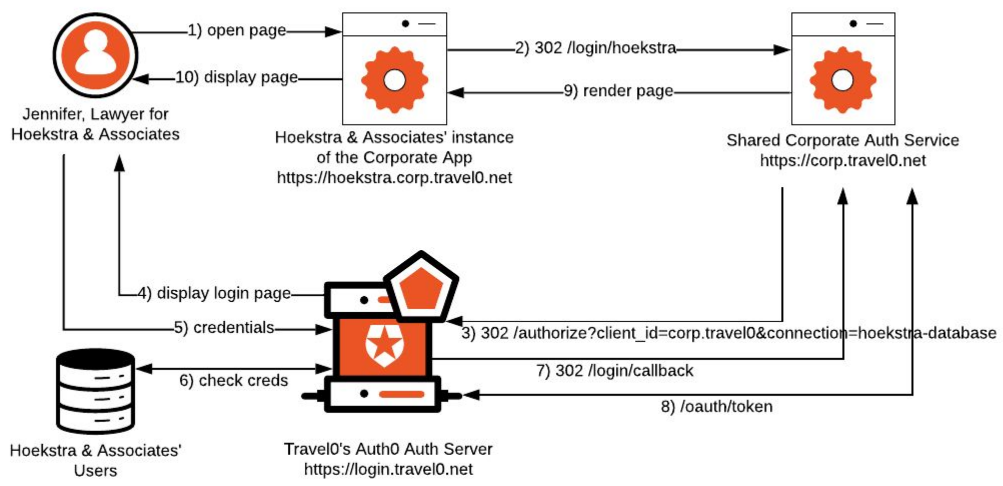
When adding a new organization user you will need to do the following:
- The Administrator will add the user to the system by entering their information into a provisioning tool.
- This tool should create a random password that is never shared with anyone
- This tool will email the user a link for setting their initial password

Authentication

Authenticating database users will require collecting credentials. This means that you will want to do this at the Auth0 Tenant (Authorization Server) as that removes any requirement for each application to protect the user's password, and also enables the Auth0 Tenant (Authorization Server) to maintain a single sign-on session for the user - thus reducing the number of times the user has to enter their credentials. This requires taking advantage of the Universal Login Page.

If your application is sharing a common domain with all of the different organization instances, then the best practice is to have a centralized application that initiates login and manages the callback. This will simplify your configuration and provide you with an application that can handle common errors and common activities (like forcing password resets, email verification, progressive profiling, etc). Let's look at our Travel0 example again. For the Hoekstra & Associates organization, the Travel0 Corporate Booking URL is https://hoekstra.corp.travel0.net. All organizations with this setup will share a common sub-domain “corp.travel0.net”. Therefore a service can be created that is hosted at https://corp.travel0.net which can manage the authentication and authorization for all organizations because it has the power to set a cookie at .corp.travel0.net that can be read at https://hoekstra.corp.travel0.net. This allows it to manage the session for Hoekstra & Associates' instance of Travel0 Corporate Booking and can therefore manage the authentication/authorization safely to minimize the complexity at Hoekstra & Associates' instance of Travel0 Corporate Booking.

Authentication Flow


1) open page
10) display page
Jennifer, Lawyer for Hoekstra & Associates
2) 302 /login/hoekstra
9) render page
Hoekstra & Associates' instance of the Corporate App https://hoekstra.corp.travel0.net
Shared Corporate Auth Service https://corp.travel0.net
4) display login page
5) credentials
3) 302 /authorize?client_id=corp.travel0&connection=hoekstra-database
7) 302 /login/callback
6) check creds
8) /oauth/token
Hoekstra & Associates' Users
Travel0's Auth0 Auth Server https://login.travel0.net

1) Jennifer from Hoekstra & Associates opens a link to Hoekstra & Associates' instance of Travel0 Corporate Booking at https://hoekstra.corp.travel0.net
   a) NOTE: If Jennifer already has a session cookie with https://corp.travel0.net, then Jennifer is logged into the system and we exit here.
2) The corporate application redirects Jennifer to the Shared Corporate Auth Service at https://corp.travel0.net/login/hoekstra
3) The Shared Corporate Auth Service generates a unique state and session key and stores the organization Jennifer belongs to as well as any deep link returnTo path that was sent to it. It then redirects Jennifer to Travel0’s Auth0 Auth Server at https://login.travel0.net. It passes the following parameters:
   a) redirect_uri: https://corp.travel0.net/login/callback
   b) response_type: code
   c) state: the unique state generated in this session. It should be a nonce
   d) client_id: the client ID for the application created for Travel0 Corporate Booking
   e) connection: hoekstra-database
   f) scope: openid profile ...
      i) NOTE: can add additional OIDC scopes here depending on the information needed about the user

4) https://login.travel0.net/authorize redirects to https://login.travel0.net/login to collect credentials from the user.
    a) This is where the Universal Login Page is displayed. This can be configured to call an API on your service to collect organization specific logos and colors for displaying on the login form.
5) The user enters their credentials in the form and clicks login.
6) The Travel0 Auth0 Auth Server checks the credentials and executes rules
7) On successful credential check and rules execution, the user is redirected back to the redirect_uri: https://corp.travel0.net/login/callback with the state passed in at step 3 as well as a code.
8) The Shared Corporate Auth Service will then validate that the state matches what it sent in, and it will call Travel0's Auth0 Auth Server at (https://login.travel0.net/oauth/token) with the code and its client secret to get the ID token. It will then use the ID token to generate a session for https://*.corp.travel0.net
9) The Shared Corporate Auth Service will use the returnTo information in the session to redirect the user back to the original application instance, landing them at the appropriate location.
10) The application instance will display the originally requested page to the user

NOTE: this seems somewhat complex, but most of this code is fairly simple and will be encapsulated within the SDKs for your technology stack.

Authorization

Authorization for multiple organization applications, when the user is isolated to a particular organization, is almost identical to any standard authorization approaches in Auth0.

There are three different aspects of authorization to consider. Depending on your application you will be affected by one or more of these aspects depending on how many applications you have and how those applications work.

1) ID Tokens often need to carry authorization information in them. This is often so you can present a user interface in a way so that the users don't even have the ability to attempt something they are not permitted to do. And also to provide authorization information to the backend of an application so it can restrict the user from bypassing frontend controls.
2) If you have a shared API, then you will want to generate an access token. This access token can carry authorization information about the user so that the API can apply the appropriate level of access control.
3) You may want to apply some coarse grained access control at your Auth0 tenant in rules. Some Examples:
    a) Restrict access to a certain API audience, so a user can not get an access token for a particular API audience.
    b) Restrict login to users who have verified their email address

c) Block access to users from a particular IP address

ID Token Claims

For the most part, claims will need to be added as per our general ID Token Claims best practices.

The audience of a token should always be respected. For ID Tokens, this audience is the Application (client) defined in Auth0. The only consumers of that ID token should be the application that generated the request to authorize. For applications that have both a frontend and a backend (some aspects of the application run in the browser), and the backend is dedicated to the frontend, this can be considered "one" application and therefore the ID token can be consumed by both. You should not send an ID token as an access token to a separate API. See our docs guidance on using access tokens to call APIs.

In addition to the claims you need for permissioning, you will almost certainly want to add the user's organization for validation at the application that the organization matches the organization requested.

Example:

```
context.idToken['http://travel0.net/claims/organization'] = context.connection.split('-')[0];
```

Shared API Access

If you are going to be creating an API and using the audience parameter to generate a JWT access token. In addition to other claims you are adding to your access token for access control decisions by the API, you will also need to communicate the organization that the user belongs to. There are two main approaches for this:

1) Create a unique API audience for each organization
2) Add the organization as a custom claim to the access token

Approach	**Pros**	**Cons**
Unique API Audience	• OOTB support for machine-to-machine access for a single organization • Audience is a standard claim in the access token • Refresh tokens lock in the	• Have to automate the creation of an API for every organization

	organization	
Custom Claim	● Simpler Auth0 Tenant Configuration	● Custom code needed in a rule to add the organization to the access token

If you go with the organization as a custom claim, this is easy to do when you have a single connection per organization as the organization name can be part of the connection itself.

Coarse Grained Access Control in Rules

Access control is the responsibility of the application and/or API's in your system. If you attempt to do all access control in your centralized Authorization Server, you will quickly run into a complex control system that is difficult to maintain and understand. However, there are some coarse grained access control opportunities that you can easily apply in a rule and remove the need for every application to apply the same restrictions. Some examples of these are:

- Restrict access to a certain API audience, so a user can not get an access token for a particular API audience.
    - If you are creating a custom API Audience for each organization it is important to create a rule to ensure that the user logging in belongs to the organization that matches the API audience.

Branding

Branding in this environment can be more complex than in a standard environment where you don't have to deal with multiple organizations. However, you will likely want to include the user's organization in most, if not all, of the pages and emails that are displayed to that user by your Auth0 tenant.

The places where you will likely want to customize per organization:

- Universal Pages:
    - Login Page
    - Change Password Page
- Email Templates

Universal Login Page

When using a database connection per organization inside the same Auth0 tenant, you will need the Universal Login Page to look different for each organization that logs in. Generally most companies simply create a generic form that is the same for each organization, but includes a logo, title, and potentially some color changes. This gives each organization an understanding that the credentials they are logging in with are within the context of this application and their own organization without requiring a large amount of customization work for each organization.

Change Password Page

In a similar fashion to the Universal Login Page, you can also customize the Change Password Page. However customizing the Change Password Page is slightly more complex because you don't have access to the connection name. To get around this, you simply need to append the connection to the URL in the Change Password Email template. Then you can get the connection name by parsing the hash on the Change Password Page. After that, it is the same as the Universal Login Page, you can use the connection name to call an API to get logo, color and title information to display to the user when prompting them to change their password. See Customizing the Change Password Page for more specific details on how to customize the change password widget. See the Branding Email Templates section to see how to customize the url.

Email Templates

In the Auth0 email templates, you have the ability to use Liquid syntax to customize your email templates. Though this does give you a flexible HTML environment where you can really customize what you want, it doesn't give you the ability to call out to an external service to get configuration information. As a result, the main option is to keep things simple, as in, don't make the email itself branded to the organization, but instead have the look and feel generic for all users. You do have access to connection.name, so you can do a large chain of if-then-else statements to customize the email, however this could be cumbersome to maintain and could get quite large depending on the number of organizations that you will have. It also requires that you have a step in your provisioning to edit and push the latest version of the email templates.

If you decide you want to attempt the branding in the email template, you can store some information in the user's metadata to display in the template. However, you need to be careful about potential size limitations in both metadata and in the email template itself before going down this path to ensure your system is scalable.

One area that you will want to customize is the change password email. You will want to customize the URL by appending the connection name, example:

```
Click <a href="{{ url }}connection={{ connection.name }}">here</a> to set
your password.
```

This will ensure that the url has the connection name in it, and that connection name will then be accessible on the change password page if you want to brand that page by organization.

Profile Management

Profile Management for multiple organization applications is primarily the same as other profile management. See Profile Management for some general advice on profile management.

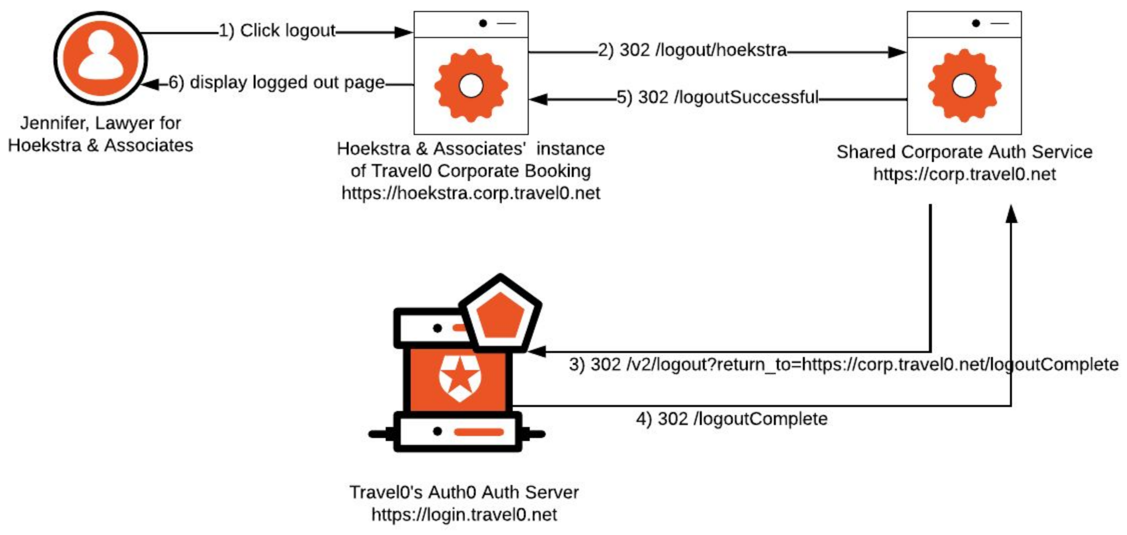
There are a few things to consider for this particular approach:

- You may not want users to be able to change their email address
- Account recovery (password reset) needs specific organization branding since the database is organization specific.
- Email verification is considered a MUST instead of a SHOULD
- Blocking Users and Deprovisioning is a MUST instead of a SHOULD

In your application you may have a set of user specific fields that you provide self-service management to users over. This may include preferences or maybe some identifying information so you can better serve the customer. However, if you are providing an invite-only service where an administrator is tightly managing user access, you will need to restrict changes to fields that are often owned by the end user. One particular example of this is the email address. You may also restrict username depending on your company's preference. If you are using username at all. The main example of why this matters, is you don't want a user putting their person email address in the system and having company specific emails going to that email address. Usernames may be restricted so that usernames can be known and controlled without worry of someone using an obscure username that no one realises belongs to a particular person or an inappropriate username.

Account recovery involves the user clicking a link that lands them on a change password page. It will be important that this change password page is branded for the organization so the user knows where they are. See Branding the Change Password Page for more details.

Logout


1) Click logout
6) display logged out page
2) 302 /logout/hoekstra
5) 302 /logoutSuccessful
Jennifer, Lawyer for Hoekstra & Associates
Hoekstra & Associates' instance of Travel0 Corporate Booking https://hoekstra.corp.travel0.net
Shared Corporate Auth Service https://corp.travel0.net
3) 302 /v2/logout?return_to=https://corp.travel0.net/logoutComplete
4) 302 /logoutComplete
Travel0's Auth0 Auth Server https://login.travel0.net

Logout for this scenario is almost identical to the logout of any other system. It will have the same level of complexity as described in our standard Logout documents. This scenario does add one additional level of complexity. In this scenario, if you are using a centralized Authorization Server (as described in Authentication) to manage your login and application session, then you will need to redirect to that service for logout as well so the session can be eliminated before redirecting to your Auth0 tenant to logout from there. This also keeps your Auth0 tenant configuration simpler because you don't need to register a separate logout URL for all organizations.

BEST PRACTICE

*If you do **not** wish to remove the user's SSO session when logging them out of the application itself, then the Shared Corporate Auth Service can simply redirect back to the Organization Specific Corporate App at a page that can tell them they successfully logged out. However, you should communicate to the user that they are still logged into the Auth0 Tenant (Authorization Server) and if they click login, they will not be challenged for credentials. You may need to also provide a separate logout or ask when they click the first logout button to give them the option for global logout or just local logout.*

Logout Flow

1) Jennifer clicks logout

2) Hoekstra & Associates' Instance of Travel0 Corporate Booking redirects to the Shared Corporate Auth Service at https://corp.travel0.net/logout/hoekstra.
    a) The Shared Corporate Auth Service stores the organization name in the server side session
    b) The Shared Corporate Auth Service removes everything else from the server side session about the user
3) The Shared Corporate Auth Service redirects the user to Travel0's Auth0 Auth Server at https://login.travel0.net/v2/logout?return_to=https://corp.travel0.net/logoutComplete
    a) Travel0's Auth0 Auth Server removes the user's SSO session
4) Travel0's Auth0 Auth Server redirects the user back to the Shared Corporate Auth Service at https://corp.travel0.net/logoutComplete
5) The Shared Corporate Auth Service redirects the user back to Hoekstra & Associates' Instance of Travel0 Corporate Booking
6) Hoekstra & Associates' Instance of Travel0 Corporate Booking renders a page to let Jennifer know she successfully logged out, with a button to log back in if desired.

Enterprise IDP Organizations

Many B2B applications want to provide organizations with the ability to bring their own IDP to this application. This has a lot of benefits including:

- Your company is not responsible for protecting user credentials for this organization
- The organization may already have roles setup
- The organization already has tools for managing user creation and removal that they can leverage for your application
- Users of those organizations like that they don't need to store or remember a new set of credentials
- The user can log in automatically if they already have an SSO session with their IDP.

In this section we will be addressing the case where an organization has a single IDP that they want to use.

Let's now walk through each IAM workstream to figure out how we should address this type of organization.

Architecture

A few requirements to consider for these users:

- The organization would like to configure their application to use their own IDP instead of storing credentials in your system.
- If the organization leaves, you want to easily decommission the users.
- The users of this organization do not want to have to enter credentials when they already have a live SSO session with their IDP.

A common practice to address these requirements is to configure a single enterprise connection for this organization. In that connection, you should be able to map any attributes to the appropriate user fields such that they can be normalized for use in rules.

Provisioning

Provisioning should be done at the organizations IDP instead of at your Auth0 tenant. This simplifies your interface. If the organization is going to limit who in their organization has access to your application, then you will need to provide a way for the organization to map a particular attribute in their IDP to a normalized user field that you can check for in a rule. It is important that you have a consistent easy to understand way to do this so that you don't have a custom rule for every organization.

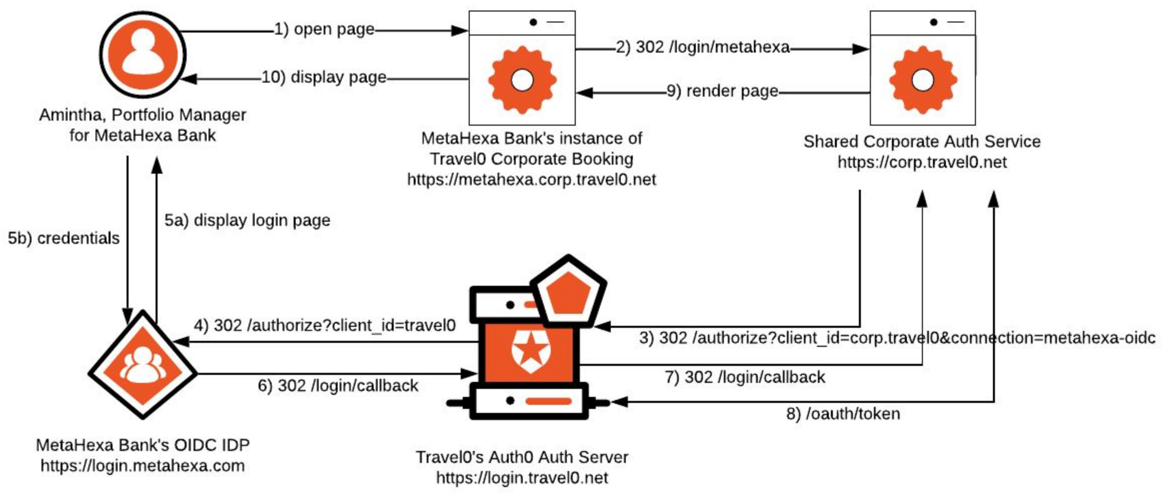
Let's use our Travel0 example again. From earlier you may recall that MetaHexa Bank uses Travel0 for corporate travel and they have their own OIDC IDP. In their OIDC IDP, they would create an attribute called "CanAccessTravel0". Then in the OIDC IDP connection you would create a mapping from "CanAccessTravel0" to "hasCorporateAccess". Then, in a rule, you can check for user.hasCorporateAccess, and if they don't you can throw an UnauthorizedError to deny them access. This is an example of white-list access, you could also do the opposite, only reject people who have "NoAccessToTravel0". Obviously you can name these anything you want, but the point is to keep it simple so that you can apply the same rule to all users to simplify your system.

Authentication

Authenticating users with a federated IDP requires that you redirect those users to the IDP to collect their credentials. The good news is that this interaction (from your application's point of view) is identical to the interaction for validating a user with credentials stored in your Auth0 tenant. Auth0 will take care of the additional redirection and negotiation with the federated IDP.

As with the database users, we will use a centralized auth service for initiating the authentication request. See the database section for more details.

Authentication Flow


1) open page
10) display page
Amintha, Portfolio Manager for MetaHexa Bank
2) 302 /login/metahexa
9) render page
MetaHexa Bank's instance of Travel0 Corporate Booking https://metahexa.corp.travel0.net
Shared Corporate Auth Service https://corp.travel0.net
5b) credentials
5a) display login page
4) 302 /authorize?client_id=travel0
6) 302 /login/callback
3) 302 /authorize?client_id=corp.travel0&connection=metahexa-oidc
7) 302 /login/callback
8) /oauth/token
MetaHexa Bank's OIDC IDP https://login.metahexa.com
Travel0's Auth0 Auth Server https://login.travel0.net

1) Amintha from MetaHexa Bank opens a link to MetaHexa Bank's Instance of Travel0 Corporate Booking at https://metahexa.corp.travel0.net
   a) NOTE: If Amintha already has a session cookie with https://corp.travel0.net, then Amintha is logged into the system and we exit here.
2) MetaHexa Bank's Instance of Travel0 Corporate Booking redirects Amintha to the Shared Corporate Auth Service at https://corp.travel0.net/login/metahexa
3) The Shared Corporate Auth Service generates a unique state and session key and stores the organization Amintha belongs to as well as any deep link returnTo path that was sent to it. It then redirects Amintha to Travel0's Auth0 Auth Server at https://login.travel0.net. It passes the following parameters:
   a) redirect_uri: https://corp.travel0.net/login/callback
   b) response_type: code
   c) state: the unique state generated in this session. It should be a nonce.
   d) client_id: the client ID for the application created for Travel0 Corporate Booking
   e) connection: metahexa-oidc
   f) scope: openid profile ...
      i) NOTE: can add additional OIDC scopes here depending on the information needed about the user
4) https://login.travel0.net/authorize redirects to https://login.metahexa.com/authorize, sending it similar parameters to delegate authentication to MetaHexa Bank's OIDC provider
5) Step 5 will almost always be completely skipped as Amintha will more than likely already have an SSO session with MetaHexa Bank's IDP to do her other work. Otherwise:
   a) The IDP will display a login page to Amintha
   b) Amintha will enter her credentials and they will be validated

6) On successful validation MetaHexa Bank's OIDC IDP will redirect back to Travel0's Auth0 Auth Server at https://login.travel0.net/login/callback with a code and state
    a) The state will be validated
    b) The code will then be exchanged for an ID token at https://login.metahexa.com/oauth/token which it will validate.
7) On successful validation of the ID token, the user is redirected back to the redirect_uri: https://corp.travel0.net/login/callback with the state passed in at step 3 as well as a code.
8) The Shared Corporate Auth Service will then validate that the state matches what it sent in, and it will call Travel0's Auth0 Auth Server at (https://login.travel0.net/oauth/token) with the code and its client secret to get the ID token. It will then use the ID token to generate a session for https://*.corp.travel0.net
9) The Shared Corporate Auth Service will use the return to information in the session to redirect the user back to MetaHexa Bank's Instance of Travel0 Corporate Booking, landing them at the appropriate location.
10) MetaHexa Bank's Instance of Travel0 Corporate Booking will display the originally requested page to the Amintha

See Enterprise Connections for more details on how to configure the enterprise connection.

Authorization

With authorization there are two pieces:
1) How will you determine what a person is allowed to do? As in, where will this information come from?
2) How will your Auth0 Tenant (Authorization Server) tell your applications and/or APIs what the person is allowed to do?

The second consideration is covered in this Authorization section and should be the same for all scenarios. As in, it shouldn't matter whether the user is a database user or an enterprise connection user, the claims and/or audience should be consistent and it shouldn't matter where the user authenticated.

For the first consideration, that will depend on where the user comes from. With a federated IDP each IDP may have different information that needs to be mapped to the user. This can be done during the configuration of the IDP when creating the Enterprise connection.

For example, you might add this configuration to the mappings section of a SAML enterprise connection:

```
{
 "user_id": [
   "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier",
   "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn",
   "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"
```

```
  ],
  "email":
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress",
  "name": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name",
  "given_name": [
    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname",
    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"
  ],
  "family_name":
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname",
  "groups": "http://schemas.xmlsoap.org/claims/Group"
}
```

Each of these fields will end up with a value in the user object in a rule. This will allow you to map something to any field and then access it in a rule using user.field_name.

For OIDC connections you will implement a method in the custom social connection that will map the decoded ID token to a user profile.

Branding

Branding is fairly non-existent for enterprise connections because most of the emails and hosted pages are part of the database configuration: change password email and page, verify email, login page, anomaly detection email, etc. This is one of the benefits of encouraging your customers to use their own IDP.

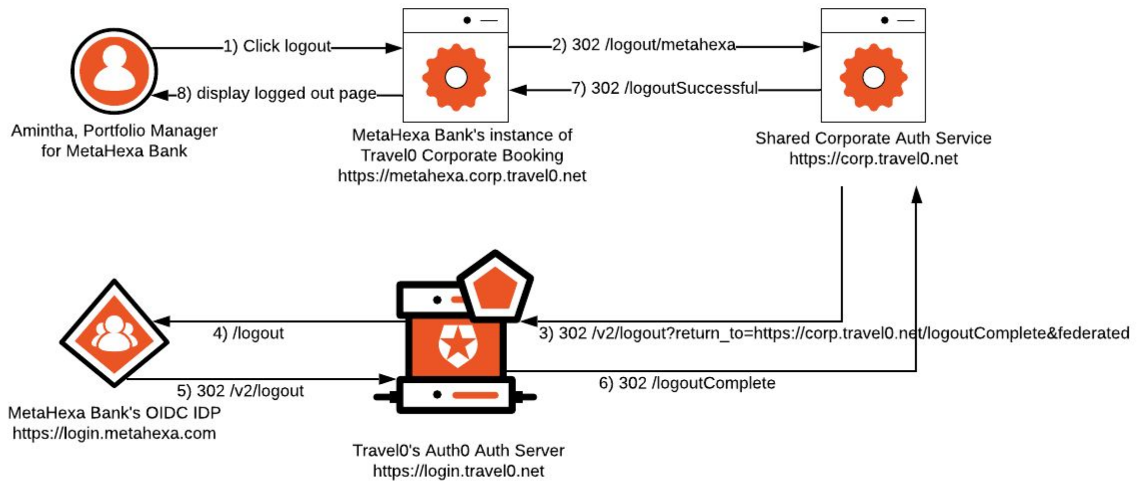
Profile Management

Profile Management is also fairly non-existent depending on how you set things up. If your authorization information can be handled by the IDP and you can use simple mappings, then profile management will be handled at the organization's IDP instead of in your system.

If you must manage roles in your system instead of at the organization IDP, then you must manage the fact that a user does not exist in your system until they log in through the IDP. If you must pre-register a user for a particular role, that information will need to be stored somewhere outside of their Auth0 profile and then accessed and copied in as part of a rule during their first login.

Deprovisioning will have to be done in your system. You will need to provide the organizations with a way to deprovision users from your system. Auth0 does not communicate with the upstream IDP except when the Auth0 SSO session expires. This amount of time is almost certainly going to be too long for most scenarios and the administrator of the organization will need a way to block or delete users until that time expires.

Logout


1) Click logout
8) display logged out page
2) 302 /logout/metahexa
7) 302 /logoutSuccessful
Amintha, Portfolio Manager for MetaHexa Bank
MetaHexa Bank's instance of Travel0 Corporate Booking https://metahexa.corp.travel0.net
Shared Corporate Auth Service https://corp.travel0.net
4) /logout
5) 302 /v2/logout
3) 302 /v2/logout?return_to=https://corp.travel0.net/logoutComplete&federated
6) 302 /logoutComplete
MetaHexa Bank's OIDC IDP https://login.metahexa.com
Travel0's Auth0 Auth Server https://login.travel0.net

Logout for this scenario is more complex than for the database organizations, in that you have several options for how to manage it:

1) You can have logout of the application result in logout of just the application and nothing else. This is something you might do if you have several other corporate applications that the user may also need to be logged into. This does result in the user automatically getting back in if they click the login button, which may be odd for them.
2) You can have logout of the application result in logout from Travel0's Auth0 Auth Server as well, but **not** from the organization IDP. This will also result in the user automatically logging back into the app if they click login.
3) You can have the logout propagate to the organization IDP as well, thus requiring the user to enter credentials the next time their session expires in other applications.
4) You can have the logout also propagate to all other applications and force logout.

You need to decide which of these options to implement with the further option of prompting the user to choose which of those behaviors they want when they click the logout button.

*If you do **not** wish to remove the user's SSO session when logging them out of the application itself, then the Shared Corporate Auth Service can simply redirect back to the Organization Specific Corporate App at a page that can tell them they successfully logged out. However, you should communicate to the user that they are still logged into the Auth0 Tenant (Authorization Server) and if they click login, they will not be challenged for credentials. You may need to also provide a separate logout or ask when they click the first logout button to give them the option for global logout or just local logout.*

Logout Flow

This logout flow demonstrates option 3, propagating the logout all the way to the organization's IDP.

1) Amintha clicks logout
2) MetaHexa Bank's Instance of Travel0 Corporate Booking redirects to the Shared Corporate Auth Service at https://corp.travel0.net/logout/metahexa.
    a) The Shared Corporate Auth Service stores the organization name in the server side session
    b) The Shared Corporate Auth Service removes everything else from the server side session about the user
3) The Shared Corporate Auth Service redirects the user to Travel0's Auth0 Auth Server at https://login.travel0.net/v2/logout?return_to=https://corp.travel0.net/logoutComplete&federated
    a) Travel0's Auth0 Auth Server removes the user's SSO session
4) Travel0's Auth0 Auth Server redirects the user to MetaHexa Bank's OIDC IDP at https://login.metahexa.com/logout?returnTo=https://login.travel0.net/v2/logout
5) MetaHexa Bank's OIDC IDP removes the user's session at the IDP and redirects back to Travel0's Auth0 Auth Server at https://login.travel0.net/v2/logout
6) Travel0's Auth0 Auth Server redirects the user back to the Shared Corporate Auth Service at https://corp.travel0.net/logoutComplete
7) The Shared Corporate Auth Service redirects the user back to MetaHexa Bank's Instance of Travel0 Corporate Booking
8) MetaHexa Bank's Instance of Travel0 Corporate Booking renders a page for the user to let them know they successfully logged out, with a button to log back in if desired.

Multiple IDP Organizations

If you have organizations that need you to support more than one IDP (either multiple enterprise connections, or a username/password connection and an enterprise connection), this has the potential to complicate the clean architecture described in the single IDP section. The goal would be to avoid this situation to limit complexity in your Auth0 configuration and eliminate potential maintenance headaches in the future.

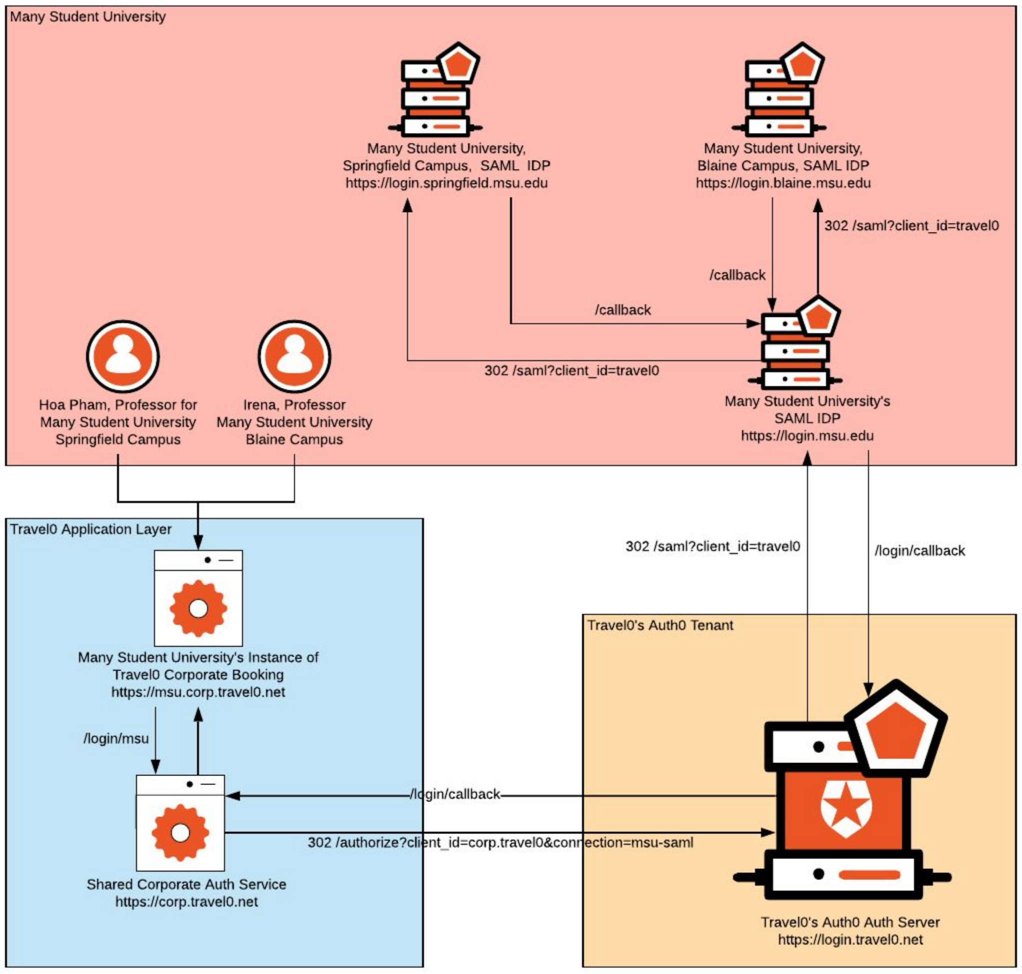
There are three ways to avoid complicating your Auth0 tenant. We will present them in order from simplest to most complex:

1) Have the organization utilize their existing IDP to simplify the IDP's they present to you. See Use Their Existing IDP Provider to Create a Single Federation IDP
2) You can split the organization at your application level. See Split the Organization.
3) You can create a separate Auth0 tenant for each of these organizations. See Create Separate Auth0 Tenant for the Organization.

If none of those options are possible, then you are stuck with complicating your Auth0 tenant to handle these organizations. See Handle Organizations with Multiple IDPs in a Single Auth0 Tenant.

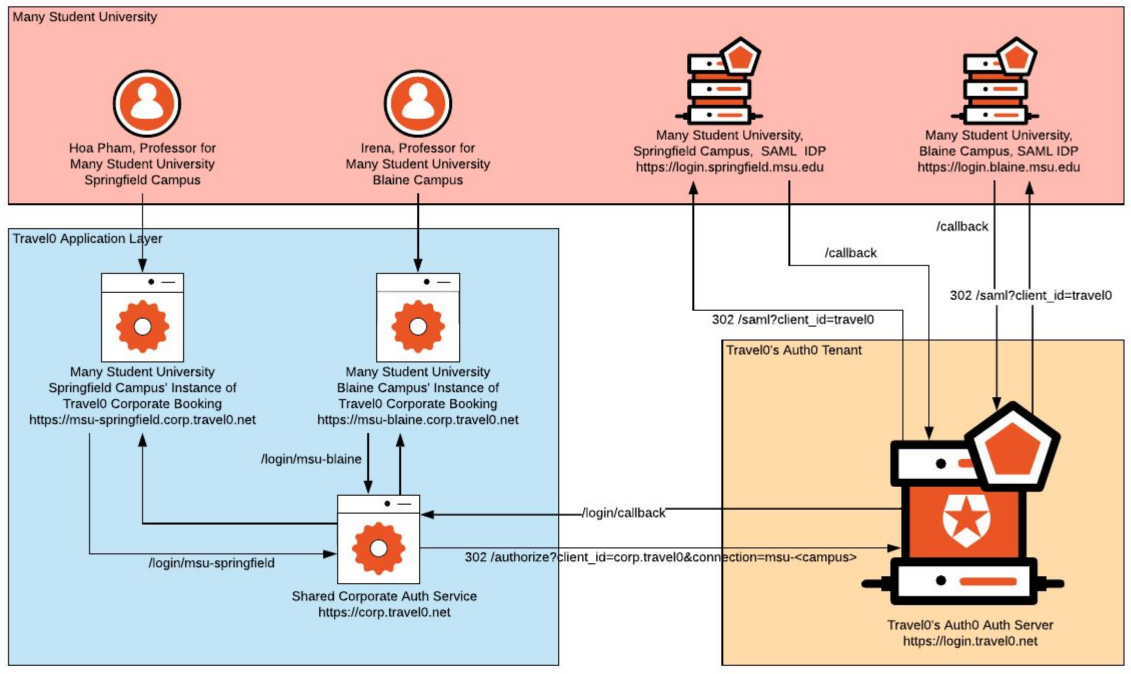
Use Their Existing IDP Provider to Create a Single Federation IDP

This solution is quite simple. The idea here is that you ask the organization to change from being multiple IDP to a single IDP by managing on their end, utilizing one of their existing IDPs. We will use University0 as our example.


Many Student University
Many Student University, Springfield Campus, SAML IDP https://login.springfield.msu.edu
Many Student University, Blaine Campus, SAML IDP https://login.blaine.msu.edu
302 /saml?client_id=travel0
/callback
/callback
302 /saml?client_id=travel0
Hoa Pham, Professor for Many Student University Springfield Campus
Irena, Professor Many Student University Blaine Campus
Many Student University's SAML IDP https://login.msu.edu
Travel0 Application Layer
302 /saml?client_id=travel0
/login/callback
Many Student University's Instance of Travel0 Corporate Booking https://msu.corp.travel0.net
Travel0's Auth0 Tenant
/login/msu
/login/callback
302 /authorize?client_id=corp.travel0&connection=msu-saml
Shared Corporate Auth Service https://corp.travel0.net
Travel0's Auth0 Auth Server https://login.travel0.net

Split the Organization

Splitting the organization at your application level will keep your authentication/authorization considerably simpler. Oftentimes it will make sense from the application level as well and keep everything simpler overall. If this is a possibility it is encouraged to follow this architecture. Again, we will use University0 from the example.


Many Student University
Hoa Pham, Professor for Many Student University Springfield Campus
Irena, Professor for Many Student University Blaine Campus
Many Student University, Springfield Campus, SAML IDP https://login.springfield.msu.edu
Many Student University, Blaine Campus, SAML IDP https://login.blaine.msu.edu
Travel0 Application Layer
Many Student University Springfield Campus' Instance of Travel0 Corporate Booking https://msu-springfield.corp.travel0.net
Many Student University Blaine Campus' Instance of Travel0 Corporate Booking https://msu-blaine.corp.travel0.net
/login/msu-blaine
/login/msu-springfield
Shared Corporate Auth Service https://corp.travel0.net
/callback
/callback
302 /saml?client_id=travel0
302 /saml?client_id=travel0
Travel0's Auth0 Tenant
/login/callback
302 /authorize?client_id=corp.travel0&connection=msu-<campus>
Travel0's Auth0 Auth Server https://login.travel0.net

Create Separate Auth0 Tenant for the Organization

This section is not complete. It will be coming soon!

Handle Organizations with Multiple IDPs in a Single Auth0 Tenant

This section is not complete. It will be coming soon!