



Delivering a B2B IAM Project

An Auth0 Integration Guide to Business-to-Business
Identity & Access Management

Table of contents

[Table of contents](#)

[Introduction](#)

[Planning](#)

[Phase 1](#)

[Workstreams](#)

[Architecture](#)

[Tenant Provisioning](#)

[Provisioning Auth0 Tenants for your Complex Organizations](#)

[Custom Domains](#)

[SDLC Support](#)

[Tenant Association](#)

[User Provisioning](#)

[Provisioning Organization Users](#)

[Provisioning Users Isolated to the Organization](#)

[Provisioning Users Shared between Organizations](#)

[User Invite](#)

[User Migration](#)

[Legacy Identity Store Proxy](#)

[User Authentication](#)

[Universal Login](#)

[Home Realm Discovery](#)

[Application Driven HRD](#)

[Home Realm Discovery through Universal Login](#)

[Email Subdomain](#)

[Identifier to Realm Map](#)

[User Choice](#)

[Username Password Authentication](#)

[Anomaly Detection](#)

[Application Integration](#)

[User Authorization](#)

[ID Token Claims](#)

[User Profile Management](#)

[Metadata](#)

[Password Reset](#)

[Account Verification](#)

[Blocked Users](#)

[Admin Portal](#)

[Branding](#)

[Universal Login](#)

- [Branding Login by Organization](#)
- [Change Password](#)
- [Custom Domains](#)
- [Email Templates](#)
- [Error Pages](#)
- [Deployment Automation](#)
- [User Logout](#)
- [User Deprovisioning](#)
- [Quality Assurance](#)
 - [Unit Testing](#)
 - [Integration Testing](#)
 - [Test Automation](#)
- [Mock Testing](#)
- [Operations \(Yvonne\)](#)
 - [Monitoring \(PHASE 1\)](#)
 - [Auth0 Status \(PHASE 1\)](#)
 - [Notifications](#)
 - [Logging \(PHASE 1\)](#)
 - [Rate Limits and other Errors](#)
 - [Backup / Restore \(PHASE 1\)](#)
 - [Email Provider Setup \(PHASE 1\)](#)
 - [Infrastructure](#)
 - [Self-Service IDP Provisioning](#)

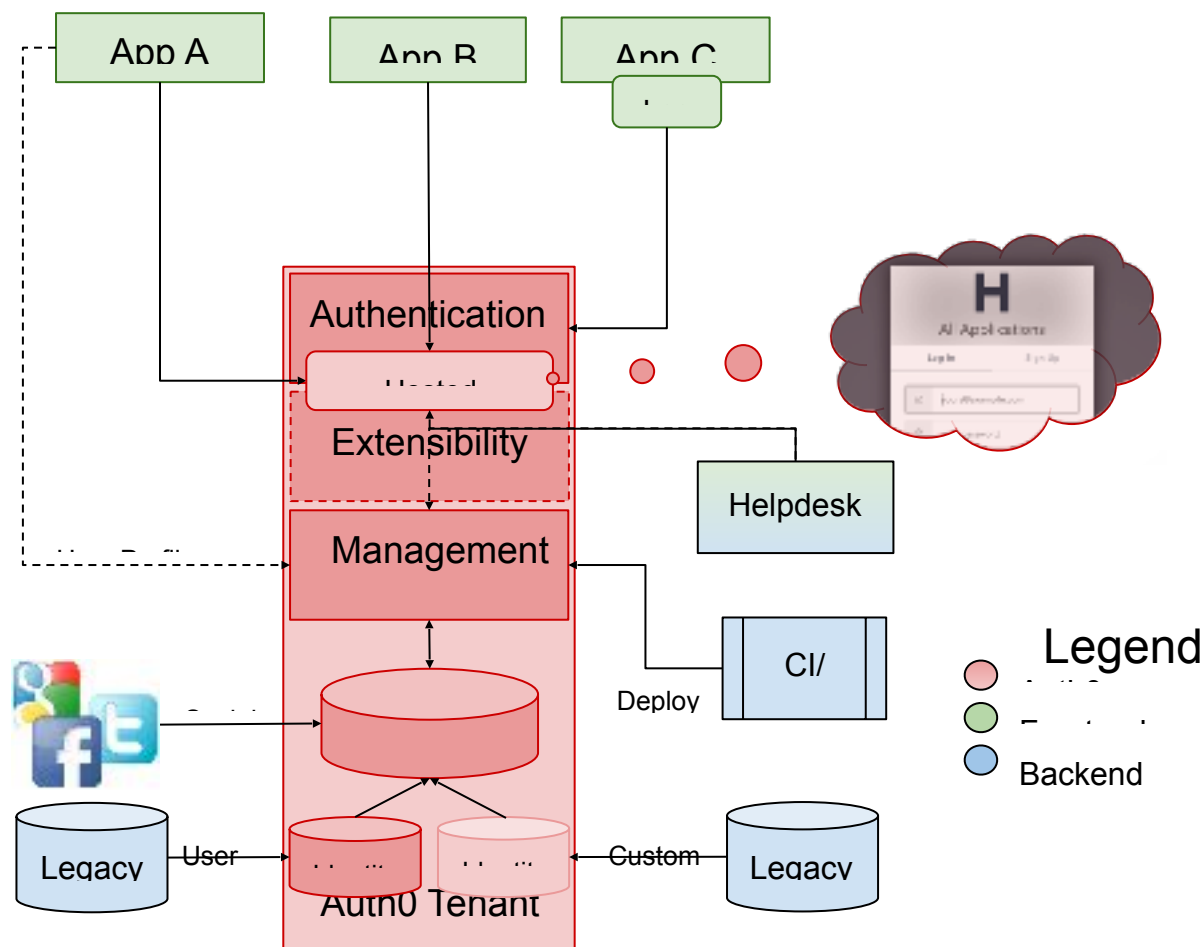
Introduction

This document is focused on integrating Auth0 within a Business-to-Business (B2B) IAM style project **based on real world customer implementation experience**. By following the advice provided, you are setting your project up for success. Customers implementing Auth0 for B2B projects, regardless of vertical or market, typically find they have a common set of goals and objectives to consider. This document sets out to distill our experience working with these customers in order to help you deliver your solution in the most effective manner.



This document is of relevance to all project stakeholders and we recommend reading it in its entirety at least once - even if you've already started your journey with Auth0.

The diagram below shows the various ways Auth0 can be integrated as part of B2B IAM project architecture. The flexibility of Auth0 allows comprehensive support for a number of use case scenarios, however not every project requires all of the capability it provides.



As can be seen (from the above diagram) there are many different things that you're going to want to consider as you embark on your journey integrating with Auth0. Knowing what, when

and how best to do this is going to help you focus on doing the right thing at the right time. To help you with this we've put together planning guidance ([here](#)) which will give you our recommended strategy for success.

Recommendations have also been provided throughout, and you'll find numerous callouts (like the one above) indicating best practices, as well as things you will need to stop and consider. This document **does not** provide an exhaustive list of Auth0 features but rather seeks to provide a generalised set of guidance and best practice recommendations, at a high-level, when you need it.



Detailed guidance regarding specific functionality and specific use case scenarios can be obtained via our online documentation at <https://auth0.com/docs/> or by engaging with the Professional Services team at Auth0 (see [here](#) for further details).

If you haven't already then we recommend you start by visiting Auth0 [here](#), and then when you're ready, head over to the planning section ([here](#)) to begin your journey!

Planning

In our experience customers who've established clear objectives, and aligned those objectives to requirements early on, have benefitted the most when it comes to integrating with Auth0. Whether working on a greenfield project, or modernizing identity and access management across an established application suite, our customers have been most successful when they plan their project using a **phased approach across multiple workstreams**. In almost all cases we've observed that our most successful customers adopt a 3 phased project structure as described below.



Identifying your primary objectives early on will help your teams focus on the specifics important to building out your solution. For instance, if your primary objective is to avoid disruption for end-users and provide continuity of service, then you shouldn't adopt a "big-bang" approach when integrating with Auth0.

Customers who **also adopt an iterative release style typically make better progress**. For example, you may have three or four applications you wish to integrate and instead of tackling them all at once, consider treating your project as a series of iterations tackling one application at a time. This way your teams can benefit from our experience and can leverage this to **help increase velocity with each iteration**.



Whilst adopting an iterative release style will improve your time to market, if you do have multiple applications and you want Single Sign On (SSO) support then our guidance [here](#) will help you understand what you need to consider.

There may also be other groups within your organization who've also been working with Auth0, and from whom you could gain first hand experience too; it's not uncommon for our customers to have disparate departments within their organisation who serve different user communities. The team at Auth0 may be able to assist you in identifying these, and doing so early on will help you when it comes to structuring your work.

Phase 1

Phase 1 focuses on integrating your application(s) with Auth0. During phase 1 you'll tackle the 10 key stages to Go-Live, across 3 key milestones, and by tackling the main risk items will address the most critical paths associated with integration. At the end of this phase you'll have working implementation that can be taken into production - or at the very least be provided as part of an early adopter or Beta program.

Phase 1 consists of a number of workstreams, with a number of topics in each. The workstreams, topics and the order in which you address each is important, so we recommend you follow the guidance prescribed. That's not to say you can't or shouldn't tackle work in parallel: User Provisioning and User Authentication, for example, could be tackled independently and at the same time, and these could both be tackled in parallel with your Branding efforts. In the majority of successful integration cases we've also found that different

teams tackle different streams, and that this can provide significant benefit: your design and development team(s) would typically tackle implementation whilst at the same time your branding team would tackle Auth0 asset customization thus reducing overall time to market.



The diagram above (click-through to navigate) provides an overall view of the planning associated with Phase 1. By Milestone 1, you will have completed major work required to integrate an application, will have addressed the most significant risk items and will also be able to provide demonstrable functionality to stakeholders too:

- Architecture ([here](#)) is the first workstream you will cover, with Tenant Provision (discussed [here](#)) being the precursor topic to all others. Other topics to address at this stage include:
 - Custom Domains ([here](#))
 - SDLC Support ([here](#))
 - Tenant Association ([here](#))
 - Provisioning Auth0 Tenants for Complex Customer Organizations ([here](#))
- User Provisioning ([here](#)) is the next workstream, and this can be done in parallel with User Authentication ([below](#)) We've found that the most successful implementations address the following topics at this stage during phase 1, however precisely what you tackle will depend on your specific requirements - i.e. you may not need User Migration, or you may already have an existing sign up mechanism that can be leveraged :
 - User Migration ([here](#))
 - User Invite ([here](#))
 - Provisioning Organization Users ([here](#))



The Auth0 Dashboard ([here](#)) in conjunction with the Delegated Administration extension ([here](#)) can be used out-of-box to provide for user provisioning and deprovisioning as described [here](#). If you require more comprehensive deprovisioning functionality - say for compliance reasons - then you should refer to the guidance provided [here](#), though we recommend you leave doing so to a later phase.

- User Authentication ([here](#)) comes next and can be done in parallel with User Provisioning ([above](#)). Topics to address at this point will include:
 - Universal Login ([here](#))
 - Home Realm Discovery ([here](#))
 - Username and Password Authentication ([here](#))
 - Anomaly Detection ([here](#))
 - Application Integration ([here](#)).
- Branding ([here](#)) can be done in parallel with User Provisioning & User Authentication, and with most customers would typically be handled by their branding team. Topics to address here include:
 - Universal Login page customization ([here](#)),
 - Branding Login by Organization ([here](#)),
 - Naming for your Custom Domain ([here](#)),
 - Change Password Page customization ([here](#)),
 - Error Page customization ([here](#)), and
 - Email template customization ([here](#); though we recommend you follow guidance [here](#) before doing so).

From here on you'll be working towards Milestone 2, and then on to Milestone 3 which will take you to production Go-Live. As you progress through the remaining work streams and topics you can start to align your Auth0 tenants with your SDLC, and you'll be steadily and progressively reducing risk as you go. You'll also have the opportunity to demonstrate further functionality to stakeholders, which will also help you to garner feedback from the rest of the business:

- Automation ([here](#)). Up 'till now you'll most likely have been working with the one Auth0 development Tenant created as part of provisioning ([above](#)). Auth0 tooling to automate deployment of assets will now allow you to utilize tenant provision for QA in preparation for your testing effort, and also production - providing you with a stable environment which can be used for demonstration and evaluation.
- Quality Assurance ([here](#)) mechanisms should now be employed to ensure any breakages due to defects or changes are detected early, and is where the Auth0 tenant provision for QA will be utilized. Topics to address here include:
 - Unit testing ([here](#)),
 - Mock testing ([here](#)) and
 - Integration testing ([here](#))

- User Profile Management ([here](#)) will address the most common cases for the changes users will want to make to their profile. We've found the most successful implementations address the following topics at this point during Phase 1, however precisely what you tackle will depend on your specific requirements (i.e. you won't need to provide for user [Metadata](#) management if you're not utilizing user metadata):
 - Metadata management ([here](#)),
 - Password Reset ([here](#)),
 - Account Verification ([here](#)),
 - Blocked Users ([here](#)), and
 - Admin Portal ([here](#))
- User Authorization ([here](#)) is - for customers who have specific access control requirements - the next thing on the agenda, and the focus for Phase 1 will be centered on how custom ID Token Claims (discussed [here](#)) can be leveraged to support this.
- User Logout ([here](#)). Eventually users will want to log out of your system and you'll need to decide exactly what this looks like. Auth0 supports several variations when it comes to user logout giving you flexibility to choose what works best for your implementation.
- Operations ([here](#)), can be addressed in parallel, though we'd recommend you setup your email provider early on as this will enable you to minimize disruption moving forward as well as allow you to quality test specific functionality not possible to do with out of box email provision. Topics to cover here will include:
 - Email Provider setup ([here](#))
 - Monitoring ([here](#))
 - Logging ([here](#))
 - Firewall configuration ([here](#))
 - Notifications ([here](#))

Congratulations! Reaching this point you are ready for Go-Live. If you've not already done so, you can align your Auth0 production tenant via deployment automation ([above](#)) and run any final QA in preparation for production release. As you move forward you'll want to keep a watch for [Notifications](#) from Auth0 which may contain important information that could impact your tenant(s) and/or project(s).


Workstreams

Architecture

<https://auth0.com/doesnotexistyet>

Understanding your application is key to understanding how Auth0 can be leveraged to meet your needs. From experience, our most successful customers start with a visualization of their proposed - or in many cases existing - architecture and then use this as a basis for reference as they progress. Understanding where your application fits within your organization is also important; Auth0 [Accounts and Tenants](#) form the basis for the grouping and structuring of Auth0


assets, and it may be that you'll need to leverage an existing Auth0 deployment in order to integrate with [SSO](#) (Single Sign On), centralized [User Profile Management](#), consolidated billing, or the like.

 *If you do have multiple applications, and you need to leverage Single Sign On (SSO), then you'll also want to checkout our guidance [here](#) before you begin.*

The value of investing time on the landscape of the architecture up-front is something that we have found pays dividends in the long run, and there are a number of things you will want to consider when looking at functionality and workflow:

- What should the URL look like when Auth0 needs to present a web page to a user?
- How can Auth0 be structured to support our SDLC (Software Development Lifecycle)?
- How can I ensure that my Auth0 Tenants are appropriately associated with my contract?
- What do I need to consider if there are other projects in my organization integrating with Auth0? Particularly projects that target their own, or a different domain of users (e.g. applications that only for employees use)?
- How can I align the structure and domain of my customers' organization with my Auth0 deployment?

Organizations often service more than one domain of user: customers, employees, and affiliates being the most frequently encountered, and typically there's little to no cross-over: employees, say, don't use the same applications as customers and vice-versa. In some cases there can also be a need to partition further within a domain: separate groups of customers, say, who use different and unconnected products. Auth0 provides a way to segregate your users and the associated collateral, and Tenant Provision (discussed [here](#)) covers this in more detail. If you need to provision an independent Tenant then you'll also want to associate this to your existing Auth0 Account as discussed [here](#), so that you can take full advantage of the benefits provided at your organization's contracted subscription level.

 *It's not uncommon for companies to have identity requirements that address multiple user communities: customers, partners, employees, etc. So be sure to consider other projects or future requirements when designing your architecture, and also check out our [B2C Welcome](#) and the [B2E Welcome](#) before you begin.*

In addition, you'll undoubtedly have an established set of processes and procedures as part of your Software Development Lifecycle (SDLC), so you'll want to check out our guidance ([here](#)) regarding Auth0 Tenant provisioning in support of that too.

For customer facing applications we typically see OpenID Connect ([OIDC](#)) as being the most frequently used protocol. OIDC makes use of web based workflows with browser URLs that are presented to the user. Out of the box, client facing URLs as part of Auth0 OIDC support are Auth0 branded, however we recommend using Auth0 [Custom Domain](#) capability to provide for consistent corporate identity and to also address potential user confidence concerns before they arise.

If your customers' organization supports the use of multiple IDPs, then it often makes sense to create a separate Auth0 (sub-)tenant for that organization; see [Provisioning Auth0 Tenants for my Complex Organizations](#) for further details. This allows you to keep the rest of your setup much simpler, by maintaining a one-to-one connection relationship between your organization and all your customer organizations within your main tenant.

Tenant Provisioning

See B2C Docs

Provisioning Auth0 Tenants for your Complex Organizations

In most cases, provisioning separate Auth0 tenants for your customers' organizations is not necessary. However, in certain circumstances this can be something that is valuable for reducing the complexity of your setup. For instance, we recommend provisioning a separate Auth0 tenant for your customers' organization as a best practice if:

- Your customers' organizations have isolated users (i.e. users don't *ever* get shared between organizations).
- You have some customer organizations that support more than one IDP. For example, you might have a customer that has their own enterprise IDP, but who also has some users that aren't in their IDP and whose credentials you will need to store. You may also have a customer who wants to provide for one or more social connections, in addition to their enterprise IDP.

If both of these are the case, then you will want to consider creating a separate Auth0 tenant for each customer that needs it. This will allow you to have a separate custom domain for them and to easily customize their login experience, including [Home Realm Discovery](#) on their login page.



Maintaining multiple Auth0 tenants can add complexity to your system and should not be done unless absolutely necessary. Make sure that you read up on our [Multitenancy](#) section before making any decision as to whether to take this approach or not.

Custom Domains

See B2C Docs

SDLC Support

See B2C Docs

Tenant Association

See B2C Docs

User Provisioning

<https://auth0.com/doesnotexistyet>

Determining how users get added is important to address early, and the decisions you make here will influence many of the decisions you'll need to make going forward. We've found there are a typical set of patterns for how users will get added to your system, and things to take note of when considering workflow design too.



Whilst Auth0 supports numerous workflows, web based workflows using Auth0 Universal Login for Sign Up are considered both industry and Auth0 best practice as they provide for optimal functionality and the best security. For further details see Auth0 documentation [here](#)

Auth0 supports user provisioning via a number of different identity providers (see [here](#) for further details). During sign up, Auth0 will also provision the Profile for the user (a.k.a. the user's Account) as described [here](#), and there are a number of things to consider when looking at functionality and workflow:

- Does a user get added to my company's domain or do they belong to/remain in their organization's domain?
- If their own domain, does a user belong to a single organization or can they belong to more than one organization?
- Should I use Auth0 as an identity store?
- Should I use my own (legacy) identity store with Auth0?
- Should I migrate user identities from my identity store to Auth0?
- Should my users sign up using their organization's identity provider?
- Should my users be invited or self register?

One of the first decisions to make when looking to provide your service(s) to other businesses is "what domain do the end-users belong to?" Based on the answer to that question, there are a couple of different approaches you will follow when provisioning those users. See [Provisioning Organization Users](#) for more information.

Auth0 provides identity storage out of the box that can be leveraged to ease the burden of storing user credentials safely and securely (see Auth0 User Invite workflow, or one of its derivatives, [here](#)). If you've already got a legacy identity store and you want to offload the burden of managing it, then Auth0's User Migration capabilities ([here](#)) provide you with a number of options to handle this. If for some reason you have to stick with your legacy identity store for now - perhaps because you've got applications which you aren't ready to migrate, or which can't be migrated - then Auth0's identity store proxying capability, described [here](#), is exactly what you need. Allowing your customers to use "bring your own identity provider" is often an attractive proposition, and when you're ready to provide it Auth0's Enterprise Connections - described [here](#) - is exactly what you'll need. Generally companies start with either a user invite flow when doing a B2B application. See [User Invite](#) for more info on what that entails.

Provisioning Organization Users

An organization should map directly to one of your business customers/partners. Each business/partner that you are working with has users who will be logging in. We are calling those end users **organization users**.

There are two different approaches to how to store your organization users:

- Isolated to the organization => Every user *belongs* to exactly one organization. It would not make sense for that user to be a part of more than one organization, and even if they were, it would make sense for them to have a separate “identity” for that other organization. See [Provisioning Users Isolated to the Organization](#) for more information. (e.g. A retail employee that works part time at two different stores has two different logins for each of those stores even if the stores both use the SaaS application.)
- Shared between organizations => In a case like this, users either create credentials in your company, or they can access other organizations instances of your application using credentials from their own organization. A simple way to look at this is that one user may be authorized to access more than one organization’s instance of the application. A user would understand that they can use the same credentials to access both instances of an application. See [Provisioning Users Shared between Organizations](#) for more information. (e.g. Some Doctors contract with multiple clinics and may need to be able to sign into each separate clinic with their same credentials)

Provisioning Users Isolated to the Organization

When users are isolated to the organization, this can provide a nice clean barrier between organizations. If there are never any users that need to access more than one organization, then this is an attractive approach.

You need to provision those users at the IDP level. Each of the organizations will have its own IDP for accomplishing this. This IDP will come in one of three flavors:

- 1) Your Auth0 Tenant is the IDP => A [Database Connection](#) in your main tenant dedicated to this organization.
- 2) Organizations bring their own IDP => You setup an Enterprise Connection for them.
- 3) Organizations with more than one IDP => You setup a new Auth0 tenant just for them and add as many IDPs (which may include a database in Auth0) as they need to that tenant, along with their own custom domain and branding.

Using Auth0 as an IDP is the recommended starting point as it’s simple to implement a user invite workflow: an administrator creates a user; a randomly generated password is created for that user, but never stored or shown to anyone; the user then receives a welcome email with a link to set their password. The only thing special about this compared to other invite flows is that the person who is creating the user will have to either select the organization ahead of time, or the system will force the organization to match that of the user doing the inviting (in situations where there is an organization administrator who belongs to that organization only). See [User Invite](#) for more information.



If you can keep a main Auth0 tenant with a 1-1 mapping between organization and connection, it will greatly simplify your login system, making it more maintainable and extendable for the future.

Provisioning Users Shared between Organizations

When sharing users between organizations you will need a way to authorize access. Since you won't know where a user might belong when authenticating, we typically recommend storing your users in a single domain - i.e. a single database connection - and then figuring out which organizations they can access through the use of user [App Metadata](#). Because of this, provisioning will often be done by starting with a [User Invite](#) workflow for the single database connection, and then `app_metadata` will be used to authorize access.

User [App Metadata](#) (a.k.a. `app_metadata`) allows information to be stored in a user's [profile](#) that can impact a user's core functionality but which a user cannot change. Let's say I'm a doctor and I belong to Clinic A and Clinic B. I might have in my `app_metadata` an `organizations` object that looks something like: `{ "organizations": ["clinicA", "clinicB"] }`, and then when attempting to log into the app for Clinic B, a rule can check that Clinic B is in the `organizations` array.



Since users are shared, you won't be able to determine who has access by isolating them to their own connection, therefore you will need to use their user [App Metadata](#) (a.k.a. `app_metadata`) to make the determination. When provisioning you will need a way to set the organizations they have access to or add a new organization to an already existing user.

User Invite

<https://auth0.com/docs/design/creating-invite-only-applications>

In most B2B scenarios, only particular individuals are allowed access to the application. As a result, it is often simpler to have an administrator provision user accounts instead of having users signup and then have an administrator approve them. This provisioning can often be done in an automated fashion when users are added to a centralized system as well.

There are three different personas who might be inviting users:

- An administrator at your company may create the users for each organization.
- An administrator from each organization may be assigned to create users.
- There may be another system responsible for creating users and that system may then create a user in Auth0.

Regardless of the audience, the technique can be similar, with the exception of the third option which would require the use of the management API and could not be done using the Delegated Administration Extension. The rest is a matter of using the right authorization model for the application.

User Invite can be accomplished in a few ways:

- using the [Delegated Administration Extension](#),
- updating a pre-existing user administration system that you've already created to use the [Management API](#), or
- creating a new application to do this using the [Management API](#).

Whether you are using the Management API or the Delegated Administration Extension (DAE), it is important to create each user with a random temporary password (using the write hook in the DAE) and *not* store that password anywhere! Then, use the Management API to send an email to the user with a link to set their password. This ensures that the only person who knows the password is the user themselves.



One of the main principles of OIDC is that no-one except the user themselves ever knows their password. If you are doing an invite flow, have your backend system randomly generate a password and then discard it and have your user reset their password before ever logging in. Do not create a temporary password and give it to them to login the first time.

User Migration

See B2C Phase 1 doc

Legacy Identity Store Proxy

See B2C Phase 1

User Authentication

<https://auth0.com/docs/user-authentication/doesnotexistyet>

In order to provide your service to your users, you must be able to identify who those users are. This process is called authentication. There are a number of ways to perform user authentication - social, username/password, passwordless - and it's often recommended that you go beyond a first factor for authenticating the user and add a second factor as well (a.k.a Multi-factor Authentication).



It's important to consider both security and user experience when designing how you will authenticate your users. Providing them multiple primary factors and/or enforcing more than one factor during authentication are ways that you can provide both.

It's important to consider both security and user experience when designing how you will authenticate your users, and so there are a number of things you will want to consider when looking at functionality and workflow:

- What do I need to do to integrate my applications with Auth0?
- Where will users enter their credentials?
- What should we be doing to keep our users' credentials safe?
- Why should we provide a way for users' to use password authentication?
- What's the benefit of providing social login for our users?
- What can we do to prevent hackers from trying to hack user accounts?
- What's the best way to minimize the upkeep on my authentication system?
- We have customers from different language backgrounds; what should I be doing to make login easy for them?
- Why would we want to migrate from our legacy identity store, and what can we do to provide our users with a first class experience as we do so?
- What do I do if I need to isolate my users by organization?
- How do I handle identifying which organization my user's belong to?
- What's the benefit of providing enterprise connections for my organizations?

Auth0 Universal Login ([here](#)) provides users with a safe and secure experience - no matter whether you chose to provide for user id/password credentials sign-in, or allow the so-called Bring Your Own Identity scenarios provided via social login. There are also brand recognition benefits to centralizing the login experience with universal login, even if you feel you will also have product specific branding requirements (see [here](#) for further details). The Auth0 UI widgets typically used with Universal Login also provide out of box support with regards to localization for users with different language requirements (as discussed [here](#)), and out of box support for Auth0 features such as [MFA](#) and [Anomaly Detection](#) enable you to put barriers in place in order to prevent hackers attempting to access user's accounts.

Providing sign-in via user id and password credentials means that you are providing a way for your users to store their credentials in your system. It gives you the control to align your application with your corporate policies for credentials (as discussed [here](#)). Auth0 provides you with a number of options in support of user id and password login and the guidance provided [here](#) will help you understand how you can leverage these. If you have an existing legacy identity store then you'll also want to see our guidance [here](#), which discusses the advantages of migrating to the safety and security of Auth0's managed identity storage. Adding social login at some point as an additional primary authentication factor provides for added flexibility, and can help you to obtain more information about your users without the need to question them force them to provide it to you since they can just give you permission to fetch the information from their social provider.

OpenID Connect (OIDC) is the most frequently used industry standard protocol when it comes to customer facing applications, and OIDC has first class citizen support in Auth0 (see [here](#) for further details). Various different approaches for integrating different applications are supported, and our guidance [here](#) provides you with the information you'll need to make an informed choice as to which you'll need to take.

Often companies need to segregate their users by organization and sometimes users can have access to more than one organization. Knowing which of these scenarios is relevant to your company will help define how to manage [Home Realm Discovery](#): whether you need to do it, when you need to do it, and how to accomplish it.

Universal Login

See B2C docs

Home Realm Discovery

<https://auth0.com/docs/home-realm-discovery/doesnotexist>

Home Realm Discovery (HRD) is the process of identifying which Identity Provider the user belongs to *before* authenticating them.

There are two main ways in which home realm discovery can occur:

- [Provide a way for the user to tell the application](#)
- [Home Realm Discovery through Universal Login](#)

Your system may need to do either or both for different applications, so it is important to understand all approaches to home realm discovery so that you can apply the ones(s) that make the most sense in your applications.



If you don't need to know ahead of time (i.e. all of your users are in a shared user pool), then you don't need to do Home Realm Discovery. You can allow users to authenticate first and then determine which organization they belong to using [app metadata](#).

Application Driven HRD

A common and effective way for determining which realm a user belongs to is when an application is branded for each organization. The organization has its own instance of the application. This copy or instance can be physically isolated (running on a separate set of servers) or virtually (running on shared servers, but presented as if it could be isolated), and is generally denoted through either a custom hostname (companyA.application1.yourcompany.com) or path (application1.yourcompany.com/companyA).



If your application already knows what connection (IDP) the user needs, then pass that along to the redirect to /authorize.

If this is the case for your application(s) then home realm discovery is a simple matter of storing the Auth0 connection name with the organization specific application configuration and sending that connection name as a parameter when redirecting the user for [Universal Login](#). Sending the connection parameter can be achieved by adding it as a query parameter when you redirect them to the authorize endpoint. For more information see the [Authentication API docs](#); however, you will generally accomplish this using the SDK for whichever language your application is written in.



If an organization needs more than one IDP, then you will have to do a second round of Home Realm Discovery once identifying their organization. This can be achieved easily with Auth0 through creating a dedicated Auth0 tenant for that organization and federating out to it. See [Multiple IDPs for a Single Organization](#)

Home Realm Discovery through Universal Login

There are three main approaches to Home Realm Discovery through [Universal Login](#):

- Discover the realm through the user's email subdomain.
- Discover the realm by looking up a user identifier in some sort of map of identifier to realm map.
- Allow the user to choose or enter their realm (or organization).

In both of the “discover the realm” approaches, you may consider doing “Identifier First Login”. This means that you present only the ability to enter an identifier first. After which you collect

the user's identifier, and then based on the identifier you either automatically redirect the user or present a way for the user to enter their password if redirection is unnecessary.



Though it is possible to implement Identifier First Login or allow a user to select their organization at the application, this can add complexity with respect to single sign on as well as complexity associated with replicating that behavior in all of your applications. Instead Auth0 recommends implementing some form of [Home Realm Discovery through Universal Login](#).

Email Subdomain

The simplest way to implement home realm discovery on the universal login page is to utilize the email subdomain of the user's identifier to map that to their Identity Provider. This, of course, only works in situations where the email subdomain will be a 1:1 mapping to an organization or at least to an Identity Provider. [Auth0's Lock widget](#) can do this for you if you are using the domain map in an enterprise connection, however if you want to build this yourself, you can, but it requires you to build a mapping of email subdomain to connection.

Identifier to Realm Map

A second, more complex alternative is to store a map of identifier's to IDP and provide a public endpoint to access that information. Then on the Universal Login page you can find the connection and redirect back to /authorize with the connection. The main drawbacks to this approach are latency, and more importantly security when it comes to identifier discovery: if you're using email addresses, this makes it much easier for someone to discover whether a particular email address is a user of yours.



Any public endpoint should have rate limiting applied to it to prevent hackers from using it to discover information and to prevent denial of service attacks.

User Choice

The other simple option is to allow your users to choose from a list, if you don't mind making public the list of organizations who use your product, or by allowing the user to enter their organization name explicitly. Once the user tells you which organization they belong to, you can redirect back to Auth0 with the connection for that organization specified.

Username Password Authentication

See B2C

Anomaly Detection

See B2C

Application Integration

See B2C

User Authorization

See B2C

ID Token Claims

See B2C doc, plus:

If you are creating different instances of your application for your customer organizations. A common practice is to create a custom claim in your ID token to represent the user's organization. Example: `context.idToken["http://yourdomain.com/claims/organization"] = "organization A";`

User Profile Management

<https://auth0.com/docs/users/concepts/overview-user-profile>

At some point you'll need to manage change to the information stored in a user's Profile. A user's profile - also known as the user's Account - is stored in Auth0, and changes to the information it contains come in many forms: self-served information update, mandatory updates concerning your organisations T's & C's, and changes due to regulatory compliance are just some of the things you'll need to handle.



*A user profile **cannot be directly accessed across multiple Auth0 tenants**, so if you're deploying multiple Auth0 tenants to production (see [here](#) for further details) then this is something you need to be aware of.*

A user's profile is populated from data supplied by an identity provider during login (see [here](#) for further details) and this is referred to as the **Normalized User Profile** (described [here](#)). By default there is one user profile created for each user identity. There are a number of things to consider when looking at user profile management:



The Normalized User Profile is updated from the identity provider during login and a limited set of information can be changed through the Auth0 Management API. Auth0 extensibility, such as Rules, can be used as an alternative to override information in the Normalised User Profile. For further information on these options see [here](#).

- What should I do if I need to store information to help customize a user's experience?
- What if I need to store user information that didn't originate from an identity provider?
- Why would I need to store user related information that a user cannot modify?
- What do I do if I need to store user related information that a user cannot modify?
- What happens if a user forgets their password?
- What should a user do if they want to change their password?
- How do I provide an administrator from a 3rd party organization the ability to manage their users?

Auth0 provides for the storage of Metadata against a user's profile, which allows for the capture of additional information - such as preference for language and/or accessibility in order to enhance the user experience. Metadata, discussed [here](#), can be used to store both information that a user can change, and also information they can't; the latter giving you the capability of associating, for example, a user profile with records in your existing systems without modifying existing implementation.

For users who forget their passwords, or who are allowed to change their password via some existing self service mechanism - or self service mechanism you have planned - Auth0's provides Password Reset functionality can be leveraged. This can be integrated with your (existing) implementation - as discussed [here](#) - and comes already incorporated with out of box Auth0 UI widgets incorporated as part as Universal Login ([here](#)).

You'll also want to make sure that you are working with a verified user account at all times, and the section [here](#) describes the mechanisms Auth0 provides for doing just that. There is also regulatory compliance to be considered (GDPR for example - <https://eugdpr.org/> - has some very specific requirements when it comes to protecting all EU citizens from privacy and data breaches) and guidance concerning that can be found [here](#).

Though Auth0 doesn't currently provide any form of centralised profile management portal out-of-the box for self serviced profile management, the Auth0 Management API can be leveraged to build your own (or utilize an already built) UI; the documentation [here](#) describes in further detail the Management API endpoint for doing this. N.B. *calls to the Management API will require use of an Access Token as described [here](#).*



Self service profile management can raise security as well as data privacy concerns. For example you may want to allow a user to change their email address, however doing so without following best practice security guidance could result in a user locking themselves out of their account; leaking Personally Identifiable Information (PII); or worse, open up a potential breach in security!

Alternatively, the Auth0 Dashboard can be used to manage aspects of a user's profile (see [here](#) for further details). Managing a user's profile via the Auth0 Dashboard is more of an administrative provision, and **should not** be used for self serviced profile management in a *production* environment. However, the interface provided by the Dashboard can be extremely useful during *development* as it provides a quick and simple way of manipulating a user's profile information.

If you need to provide a way for your customers to have an administrator that can manage their own users when they are storing those credentials in your system you can either build something yourself or use an Auth0 Extension. See [Admin Portal](#) for more information.

Metadata

See B2C Doc

Password Reset

See B2C Doc

Account Verification

See B2C Doc

Blocked Users

See B2C Doc

Admin Portal

An admin portal is an application where you can create new users, edit a user's profile, see activity about a user, etc. This application should be accessible by administrators only. Though

Auth0 provides its management dashboard, it is not advised to give access to the management dashboard to many people as there are a lot of ways someone can unintentionally break your Auth0 tenant. Instead, Auth0 provides two other options:

- 1) [Auth0 Management API](#) => With the Management API you can easily construct an application that provides administrators the ability to manage users. You can either incorporate this into an existing application that already exists for your administrators, or create a new one with a UI that matches your current applications.
- 2) [Auth0 Delegated Administration Extension](#) => This powerful and flexible extension allows you to customize a user administration experience. You can tailor this extension so that you can allow your customer admins to log in and allow them to only see and manage users within their organization.



If you are providing your own way for an administrator to manage users, you should only allow administrators to send users a change password link through email rather than allowing administrators to set passwords directly. If you must go against this recommendation and allow your administrators to set someone's password, you should force the user to change their password at their next login so that only they it (and not an administrator as well).

Branding

<https://auth0.com/doesnotexistyet>

Auth0 can be customised to provide a look and feel that aligns with your organisation's brand requirements, and your users' expectations: Universal Login, Custom Domain naming and Email Templates are a few such items that can be branded with your look and feel.



Branding Auth0 collateral provides a consistent look and feel to the user experience for your customers as well as giving them the peace of mind that they're using a product from a trusted and secure source.

Auth0 also provides support for Internationalization (I18N) and Localization (L10N) - see [here](#) for more information - and out of box collateral such as the Auth0 Lock UI widget come ready enabled for multiple language support, with built-in extensibility for adding more languages if what you need doesn't already exist (see [here](#) for further details).



Almost all applications need Internationalization and/or Localization in one form or another. Auth0 makes it easy to add, but you need to account for it up front: back-filling localization, for example, can be a painful process if left too late.

There are a number of things to consider when looking at what, and how best, to customize the various aspects of Auth0:



To provides users with helpful resources if they experience problems you should also configure a friendly name and a suitable logo, as well as the support email and support URL for your organization (see [here](#) for details).

- Can I use my own branding with Auth0 hosted pages, such as Universal Login?
- What do I do if I need to localize Auth0 hosted pages, such as Universal Login?
- If I am sharing an Auth0 tenant across customer organizations, should I add organization specific branding to their login experience?
- Can emails be customized so that they're branded and based on user preferences?
- If I'm using your hosted pages, how will my users know they're still on my domain?
- What should I do to provide for additional browser security, e.g. Extended Validation?
- Can I direct my users to our support organisation in case of errors, etc? And why would I want to? Can't Auth0 take care of that for me?

Auth0 provides tremendous flexibility when it comes to customizing and configuring the hosted pages such as Universal Login and Change Password, as discussed [here](#) and [here](#). So you can pretty much set up whatever UX look and feel you require. For many, the out of box experience - with perhaps a little alteration - is all that's required. However for others the value of their brand, and brand awareness, requires more extensive customization. This flexibility extends to not only Auth0 hosted pages, but via extensibility can also be applied to the templates Auth0 uses for email communication as discussed [here](#). Auth0 Custom Domain functionality further enhances consumer awareness, by providing users with the confidence and peace of mind when it comes to safety and security (see [here](#) for further details).

If you are sharing an Auth0 tenant across different customer organizations, providing organizations with their own domain of users, and are managing their credentials, you need to consider how the users will know which credentials they should use and trust that this is where they should enter them. See [Branding Login by Organization](#).

Whilst Auth0 provides for default information when it comes to error situations, out of box information can be somewhat cryptic as the context that can only be provided by you is missing. Auth0 error page customisation, and guidance as provided [here](#), however can mitigate that by allowing you to provide information of a more context specific nature via your own support organization.

Universal Login

See B2C Doc

Branding Login by Organization

Whether or not you need to do special customization on the Universal Login page is determined by how you plan to manage your customers' organizations. Before reading through this section, make sure you have read through the [Multitenancy](#) and [Universal Login](#) sections.

If your organization users will all be isolated from each other, than it's important to make it clear on the Universal Login page which organization the login page is for. This can be done in a couple of ways:

- Create JavaScript on the Universal Login Page that can pull resources from a CDN based on the organization presented to it.
- Create a separate tenant for the organization and use the Universal Login page to customize as desired for that organization

Change Password

See B2C Doc, plus:

If your organization users will all be isolated from each other, than it is important to make it clear on the Universal Login page which organization the change password page is for. This can be done in a couple of ways:

- Create JavaScript on the Change Password Page that can pull resources from a CDN based on the connection parameter presented to it which should tell you which organization the user is from.
- Create a separate tenant for the organization and use the Universal Login page to customize any way desired for that organization

Custom Domains

See B2C document



*If your organization users will all be isolated from each other and you ****require**** that those users are presented a login page at a custom domain, then your only option is to create a separate [Auth0 tenant for each organization](#).*

Email Templates

See B2C document

Error Pages

See B2C document

Deployment Automation

See B2C documents

User Logout

See B2C Document

If you are doing Login Federation (redirecting users to a separate IDP to login), then Logout Federation is something that you need to consider for your application. *Should* you log the user out of their IDP when they log out of your application? The answer to this depends on what your users would expect. If the application is tied closely to the organization and is a central part, then they may expect to be logged out of their IDP, if not, then it may be frustrating to them to get logged out of their IDP when they log out of your application. In most B2B applications, federated logout is not expected by the user, but if your situation is different see [Federated User Logout](#) for more information.

User Deprovisioning

See B2C Doc, plus



*Auth0 will ***not*** communicate with the upstream IDP if there is an active SSO session with Auth0, unless you force it with a `prompt=login`. If one of your customer organizations can not manage logout for those users, they may still have access after they've been decommissioned. Depending on the IDP, if Auth0 gets a token for their API, you can request information about the user from the IDP in a rule to poll whether that user should still have access or not. If you don't have this ability, you will have to provide your customer organizations a way to trigger a block or decommission of users in your system either through an API call or a UI.*

Quality Assurance

See B2C docs

Unit Testing

See B2C docs

Integration Testing

See B2C docs

Test Automation

See B2C docs

Mock Testing

See B2C docs

Operations

Operationalization requires configuring or setting up infrastructure to support the scalable, measurable and quantifiable operation that's necessary for business continuity. In Auth0, this includes configuring supporting services such as email providers, monitoring services for your deployment, detecting anomalous situations and making preparations to recover quickly and smoothly when something goes wrong in a production environment.

Establishing effective operational behaviours is something that successful customers have found pays dividends, and there are a number of things you will want to consider when looking at your workflow:

- What should I be doing to proactively detect failures?
- What do I need to know regarding how to obtain data on Auth0's operational status?
- What should I be doing about Auth0 security bulletins related to the Auth0 service
- Does Auth0 provide information regarding impending changes in the Auth0 service?
- How can I check for important notices from Auth0?
- Where should I be doing about Auth0 log data so that I can analyze it and keep it for longer than Auth0's limited data retention period?
- Can I scan Auth0 logs to determine if peak loads in my application trigger any rate limits or other errors?
- What email services should I be using use to support production volumes of email messages to users? Why can't I use Auth0 out of box email provider in my production environment?
- Why would I need to configure my firewall, and what firewall ports will I need to open for internal services that need to receive communications from Auth0 (such as custom databases, web services and email servers)?
- Will you want to provide self-service provisioning of your customer organization IDPs?
 - For guidance see the [Self-Service IDP Provisioning](#) section

Auth0 supports functionality for monitoring Auth0 service operation (as discussed in [Monitoring](#)) as well as providing information regarding Auth0 service status (discussed in [Service Status](#)). In addition, Auth0 makes available security related bulletins as well as information regarding upcoming changes to the Auth0 service via various notifications as discussed in [Notifications](#).

Auth0 logging services - discussed in [Logging](#) - provide extensive functionality for tracing and identifying operational anomalies, including restrictions encountered due to rate limiting and/or excessive loading.

Out of the box, Auth0 provides email delivery services to help you accelerate your integration. These services however are not meant for scale of use in production environments, and do not provide for any specific service level or guarantee when it comes to email delivery. Our best practice recommendation which customers typically follow involves configuring your own email service provided as discussed in [Email Provider Setup](#).

You may also need to make infrastructure configuration, as described in [Infrastructure](#), in order to support integration with Auth0. Use of Auth0 extensibility, for example, may mean that Auth0 makes calls back to your internal or even external infrastructure (e.g. if you need to make external API calls in Rules or Hooks, or via custom database scripts if you need to leverage existing legacy identity storage).

Some Auth0 customers have invested in creating a self-service portal for the admins for their customer's organizations to be able to configure the IDP for that organization. For more information see [Self-Service IDP Provisioning](#).

Monitoring

See B2C Docs

Service Status

See B2C Docs

Notifications

See B2C Docs

Logging

See B2C Docs

Email Provider Setup

See B2C Docs

Infrastructure

See B2C Docs

Self-Service IDP Provisioning

Auth0 makes it easy to configure IDPs, but depending on the number of organizations and the amount of coordination work with those organizations, it can be a time consuming process to onboard a new customer organization IDP. As a result, many of our customers have found it worth the time and effort to build a self-service portal for the administrators of their customer organizations so that they can configure their own IDP instead of having your IT department have to work with them. Auth0's Management API provides all of the necessary functionality to support your UI.



If you are creating a separate [Auth0 tenant for your more complicated organizations](#), then you can consider allowing an administrator from that organization to have access to the Auth0 Dashboard of that separate tenant. This would depend on how much trust there is with the other organization as that administrator would be able to break their login experience if they don't know how to set things up correctly.