



Migrating Application Authentication to Azure AD B2C in a Hybrid Environment

August 2018

Contents

Solution Overview.....	4
Business value.....	4
Security & Reliability	4
Friction-Free Customer Experience	5
Ease of Development and Deployment of New Apps.....	5
Technical Steps to Planning a Migration to Azure AD B2C	6
Identifying the Right Set of User Attributes to Store in Azure AD B2C.....	6
Creating Custom Attributes	7
Planning for Cost Efficiency	9
Azure AD B2C Pricing	9
Cost of Ancillary Services	9
User Migration Planning	9
Cleaning Existing User Data	9
Migrating Social Logins	10
App Migration vs User Migration.....	10
Migration Strategies	10
Using Azure AD B2C as Authentication Middleware	10
Bulk User Migration	11
Just-In-Time User Migration	12
Maintaining Consistency Between Two User Repositories	13
Handling Password Migration.....	15
Considerations When Operating in A Hybrid Identity Environment	16
Breaking Single Sign-On (SSO) Across Apps	16
Advanced Threat Protection Capabilities Aren't Realized	16
Duplication of Compliance Work	16
On-going Dependency on Legacy Authentication Provider	16
Planning for Updates to Apps	16
Updating the App	17
Maintaining the App	17
Compliance and Data Residency.....	18

GDPR Compliance	18
Data Residency.....	18
Localization / Customization.....	18
Staffing for the Right Roles and Responsibilities	19
Application Development	19
Policy Development	20
User Maintenance.....	20
Skills Analysis.....	20
Testing Your Identity Solution.....	20
Functional Testing.....	21
Usability Testing.....	21
Policy Testing	21
Load Testing	21
Federation Testing	21
Architect for robustness to failure for scenarios that are adjunct to core auth.....	21
Roll-Out Planning	22
Engage Azure Identity Support	22
Communication Planning to End Users.....	22
Roll-Out Best Practices.....	22
Solution support and analytics	23
Reporting and Audit.....	23
Sign-In Reports.....	23
Audit Reports	23
Usage Reports	24
Getting support from Microsoft and the Community	Error! Bookmark not defined.

Solution Overview

Today's business requires that applications interact with users on a personal level. Historically, it has been the responsibility of the application creator to maintain the identity of the users, including the storage and security of personally identifiable information (PII). Modern Consumer Identity and Access Management (CIAM) platforms such as Azure Active Directory B2C (Azure AD B2C) enable application developers to separate the authentication aspect from the application content, increasing availability and security, while lowering infrastructure and dev-ops costs.

However, moving from an existing authentication architecture to using Azure AD B2C as a claims-based identity provider may create a hybrid identity, either temporarily or for a longer duration. A hybrid identity environment exists when different applications use two different identity providers or authentication mechanisms to authenticate the same set of users. This situation may exist during the migration of applications and users from one system to another. It may also be a design choice that can support legacy applications, while simultaneously allowing new behavior such as mobile application authentication.

In this solution guide, we will focus on the different aspects of putting together a migration plan for moving your first app into the cloud in a hybrid identity environment. The actual implementation of user and app migration is out of scope for this document as the topic has been extensively covered in other guides, to which there are links throughout this paper. For links to additional migration deployment guides and samples, please refer to the Additional Resources section at the end of this paper.

Business value

Some of the top business reasons we get from customers look to move to a CIAM service include:

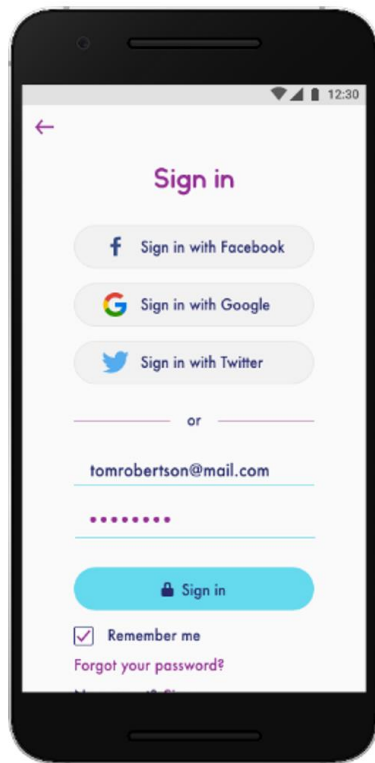
- 1) I want to provide reliable and secure SSO access to my customer-facing apps (and I don't want to be in the identity business)
- 2) I want to add social identities to my web and mobile apps
- 3) I need to transition my existing customer accounts to a new cloud-based solution and provide a smooth and simple user experience
- 4) I want the access to my apps to be protected from advanced threats
- 5) I need to comply with industry regulation and national data protection laws

While the prospect of planning out a multi-app, multi-user store migration may be daunting at first, there are significant advantages to such a strategy.

Security & Reliability

Azure Active Directory (Azure AD) is the largest single enterprise cloud identity provider in the world with over 14.2 million organizations hosting over 1 billion identities. Using the same infrastructure for your customers means that you get out-of-the-box security features such as protection for your customers' identities, additional security layers (MFA), protection of your application frontends from brute force and other attack vectors, as well as a 99.9% availability SLA. Additionally, the platform guarantees data residency and has all the built-in functionality you need to make your app compliant with the General Data Population Regulation (GDPR).

Friction-Free Customer Experience



With the proliferation of mobile and web apps, customers expect seamless, fun, and highly polished interactions with the apps they use the most. Azure AD B2C provides several ways for you to create rich, friction-free experiences for your customers. These include:

- The choice of whether to match each app's identity experience to the app's branding, or to create a single identity experience across your entire family of apps to increase overall corporate brand recognition
- Self-service sign-in, sign-up as well as profile and password management or reset experiences that can be deployed with little to no need to write new code
- "Bring-your-own-identity" using several different social identity providers such as Facebook, Google, WeChat, QQ and more
- The ability to enhance user profiles with media and detailed metadata
- Massively scalable cloud storage for the rest of your app's user data storage needs

Ease of Development and Deployment of New Apps

As your enterprise continues to develop new apps, having a standards compliant CIAM platform that abstracts authentication away from the app itself ensures you can get to market more quickly, and not worry about your security posture weakening over time as new threats come online. Additionally, using a centralized cloud identity platform enables you to offer single sign-on to users across your properties, and allows you to draw rich user and app usage insights to improve how you engage with your customers.

Technical Steps to Planning a Migration to Azure AD B2C

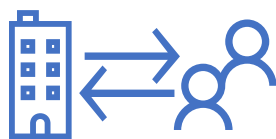
Identifying the Right Set of User Attributes to Store in Azure AD B2C

The user directory in Azure AD B2C is essentially a flat table of users and identity data about the users. These attributes are common pieces of identity information that you may wish to know about a user.

When migrating the first of several apps to Azure AD B2C, it is important to determine the right set of user attributes to be stored in the Azure AD B2C database. Specifically, the first few apps may not always encompass the entire set of user attributes subsequent apps use.

<input type="checkbox"/>	NAME	DATA TYPE	DESCRIPTION	ATTRIBUTE TYPE
	City	String	The city in which the user is located.	Built-in
	Country/Region	String	The country/region in which the user is located.	Built-in
<input checked="" type="checkbox"/>	Display Name	String	Display Name of the User	Built-in
	Email Address	String		Built-in
<input checked="" type="checkbox"/>	Given Name	String	The user's given name (also known as first name).	Built-in
	Job Title	String	The user's job title.	Built-in
	Postal Code	String	The postal code of the user's address.	Built-in
	State/Province	String	The state or province in user's address.	Built-in
	Street Address	String	The street address where the user is located	Built-in
<input checked="" type="checkbox"/>	Surname	String	The user's surname (also known as family name or last name).	Built-in

Out of the box, Azure AD B2C has several of the most common user-identifying attributes available for you to choose from. These attributes are called 'Built-in' attributes. Additionally, you can create up to a hundred custom attributes that are specific to the set of apps that your customers will interact with.



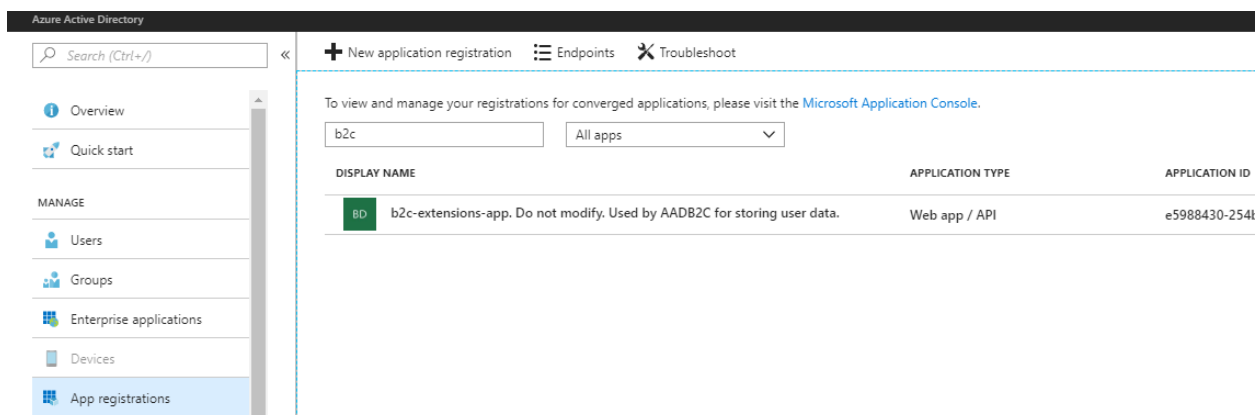
Decide Which Attributes Will Be Used Across All Apps: While planning to migrate authentication to Azure AD B2C, you need to determine what attributes you wish to store as part of the user's profile in Azure AD B2C, and what attributes you wish to keep within the app itself. It is a good idea to store a few, well-chosen attributes related to the user's identity instead of using Azure AD B2C as a substitute for a Customer Relationship Management (CRM) provider. This ensures your authentication tokens are easier to manage, and more complex user profile attributes can be stored in a CRM that supports richer querying and is a more purpose-built database system. This is important, as all apps will use the same user datastore to authenticate against.

In general, login information and data about the identity of the user should be stored within the Azure AD B2C user object. Most other data that isn't identity related should be stored by the app.

Type of Data	Where it Belongs	Why
Username, password, email addresses, phone numbers, membership identifiers	Azure AD B2C	Potentially required for core login/profile management or password reset
Consent markers for privacy policy, end user license agreements	Azure AD B2C	You may need to block access to your app/site based on these responses
Credit card information, SSN, medical records or other data regulated by industries/compliance bodies	CRM or custom storage solutions compliant with industry regulations	Storage of sensitive information is often governed by local or industry laws. Separating it from your core identity system gives you greater control over managing compliance requirements for your business
Marketing or communication preferences, user behaviors and insights	CRM/marketing tools	While you can build marketing and behavior insights on top of Azure AD B2C data, the relatively flat data storage structure for a user in Azure AD B2C may not be optimal for the creation and analysis of behaviors, trends and insights across your entire set of products or services

Creating Custom Attributes

Built-in attributes are stored within the user directory and cannot be deleted or modified. For instance, you cannot delete the Name or Email Address field from the Azure AD B2C user database schema though you can choose not to use them. Custom attributes on the other hand, are stored as fields that are tied to the app that was used to create the custom attribute. For all custom attributes that are created through the Azure Portal, Azure AD B2C uses a specific first party Graph App:



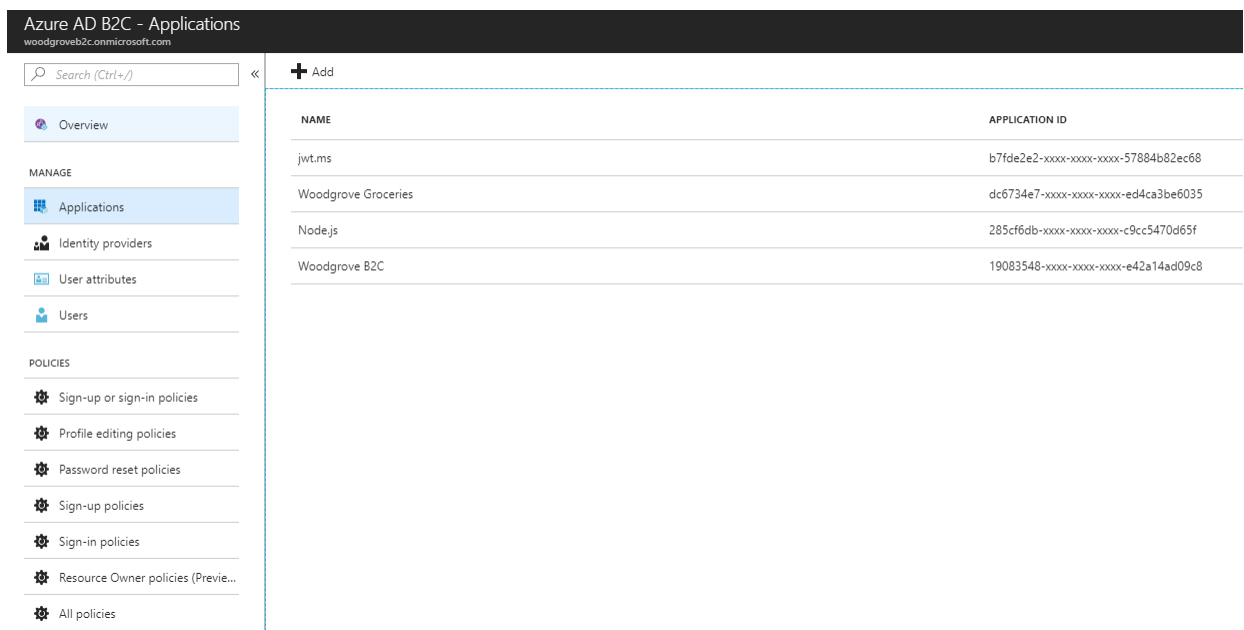
If you use the Graph API surface to inspect a custom attribute that is created through the Azure AD B2C portal, you will observe that the attribute is stored in the following format:

Example of custom attribute as seen by graph api call:

```
{
  "extension_e5988430254..._AgreedToTermsOfService": true
}
```

In this example, the “extension_” identifies the attribute as being a custom created attribute, and the “e5988xxx” value reflects the AppID of the application that was used to create the attribute. Every Azure AD B2C tenant has an app that is created by default called the ‘B2C_Extensions’ app. This is the default app that is used to create and configure custom attributes through the Azure Portal. If the B2C_Extensions app or any app you create to store custom attributes is deleted, it will cause all attributes stored by that app to be deleted as well.

Registering Multiple Apps to Run on an Azure AD B2C Tenant



NAME	APPLICATION ID
jwt.ms	b7fde2e2-xxxx-xxxx-xxxx-57884b82ec68
Woodgrove Groceries	dc6734e7-xxxx-xxxx-xxxx-ed4ca3be6035
Node.js	285cf6db-xxxx-xxxx-xxxx-c9cc5470d65f
Woodgrove B2C	19083548-xxxx-xxxx-xxxx-e42a14ad09c8

All applications that use Azure AD B2C for authentication must have their own app registration within the tenant. A tenant can support up to 250 apps.

When an application calls Azure AD B2C to authenticate a user, the application needs to provide the name of the policy being used, its own ApplicationID, and a ‘reply URL’ to which the user must be sent once authentication is complete. Multiple applications can call the same policy. Additionally, during the registration of an app, the app can also specify a list of reply URLs belonging to the same domain, any of which the user can be sent to after authentication. If the authentication call specifies a reply URL that was in the list of URLs specified in the app registration, the user will be redirected to that URL. Each application supports one domain, and multiple reply URLs may be specified if they remain in the same domain.

Single sign-on configuration ⓘ

Tenant	Application	Policy
Disabled		

Single Sign-On (SSO) allows for a single authentication action to enable access to multiple applications. Azure AD B2C supports SSO for any application within the tenant based on the policy. SSO can be enabled across the entire tenant, across all applications that use a specific policy, for a single application, or

disabled entirely. More details about SSO can be found here: <https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-token-session-ssso>

The only time a separate AAD B2C tenant would be needed is in the case of a business requirement that dictates a separation of users – for example, a company that maintains users in different geographic users with different compliance requirements may want to separate users by tenant.

Planning for Cost Efficiency

Azure AD B2C Pricing

Azure AD B2C has a billing structure that allows you to kick-start your proof of concept or pilot at little to no cost, and then has a tiered pricing structure as the number of enrolled users and authentications grows. The latest pricing information can be found here: [Azure AD B2C Pricing](#)

Billing is done based on the number of stored users, the total number of authentications, and the use of built-in Azure multi-factor authentication.

It's worth carefully assessing each of these vectors while planning your deployment to ensure that you have a cost-efficient solution. You may realize additional costs to support customized UI content or other policy requirements.

Cost of Ancillary Services

While developing a fully functional customized authentication experience for your customers, you may require other resources. If you develop customized content for the different authentication pages, you need to host the content on a publicly facing server – one option being Azure Blob Storage. Based on the verification requirements for users, web service calls may be utilized to verify data or incorporate more information into the user profile – this may be done using Azure Functions or using a web API hosted by Azure. Regardless of whether these additional requirements are hosted on-premises or in the cloud, they need to be included in the cost analysis. You can find pricing for the full range of Azure services using the [Azure Pricing Calculator](#).

User Migration Planning

Cleaning Existing User Data

As part of the migration of your existing user base, you can take this time to clean up accounts and data that may no longer be worthwhile to keep.

Junk Accounts



Ensure that obviously spurious accounts with junk email addresses, login identifiers, or data are removed in a pre-processing step. Consider using one of several third-party email list cleaning services that can quickly flag junk accounts that have crept into your system.

Duplication



Simple heuristics such as matching on email addresses, cellphone numbers or other relevant identifiers in your database can help de-duplicate user accounts. While you may not be able to arbitrarily remove a potential duplicate account, you can add a flag to the user entry and design a workflow that asks the user to merge a potential duplicate account the next time they access your service. This also serves the purpose of getting you more complete and consistent user records.

Aged accounts

If your legacy system has timestamps corresponding to the last time a user accessed your services, you might consider either leaving those accounts behind entirely, or flagging them for future deletion as you migrate.

Migrating Social Logins

When you plan to migrate your identity provider to Azure AD B2C, you may also need to migrate users with social identities. Refer to our online documentation titled [Azure Active Directory B2C: Migrate users with social identities](#) to learn more about how to do this.

App Migration vs User Migration

A complete solution will involve migrating both applications and users to Azure AD B2C. You likely want to handle a migration without significant impact to your end users. As you migrate applications to use Azure AD B2C for authentication, you want your other existing applications to continue working, using your existing authentication infrastructure.

One challenge is keeping user identity data synchronized between Azure AD B2C and your existing identity store. A potential solution is to separate the migration of applications from the migration of users.

Migration Strategies

Using Azure AD B2C as Authentication Middleware

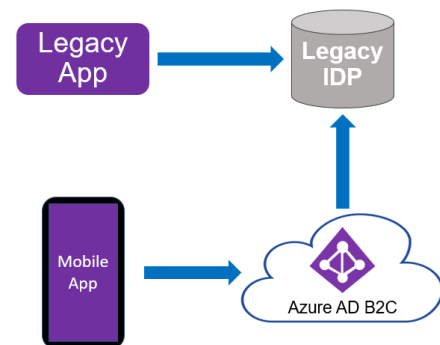
You can choose this strategy when all the following apply:

- You want to migrate applications separate from migrating users
- You want to maintain just one active user repository

One strategy which supports modern authentication for new and migrated apps, while maintaining your current infrastructure as-is, is to use Azure AD B2C as a middleware for authentication.

While this is not a true user migration strategy, it can provide an important stepping stone to a complete migration solution. This strategy does not migrate your user accounts into Azure AD B2C and will continue to require that your current authentication infrastructure remains in place.

The implementation will differ depending on your current identity provider. If your current infrastructure uses a standards-based identity provider, Azure AD



B2C can federate with your existing provider. The following standards are supported by Azure AD B2C for federation:

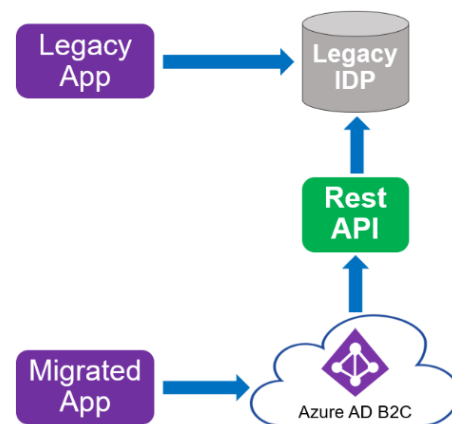
- OpenID Connect
- OAuth 1.0
- OAuth 2.0
- SAML
- WS-Fed

If your current infrastructure provides another authentication mechanism, such as storing credentials in a database, you can create an API to allow Azure AD B2C to interact with your existing user directory.

A benefit of this strategy is that Azure AD B2C provides authentication options for apps and devices that may not support the older authentication infrastructure. For instance, it may be vastly more difficult to authenticate a mobile application against a SAML provider. Since Azure AD B2C supports federation with SAML providers, the application can utilize B2C to authenticate, and Azure AD B2C will interact with the SAML provider using API calls.

There are several workflows to consider. You need to provide a method to create a new account, sign-in to an existing account, edit the profile for an account, and change the password. The implementation for these workflows will depend on whether you are using federation or integrating with a Rest API.

This table provides an example of the user and logic flow for a successful sign-in.



Federated Identity Provider	Rest API interface
<ul style="list-style-type: none"> • User starts sign-in process for Migrated App • App redirects user to Azure AD B2C • Azure AD B2C redirects user to federated / legacy identity provider • User enters credentials into legacy identity provider • User is returned with a legacy authenticated session to Azure AD B2C • Azure AD B2C creates JWT and returns user in an Azure AD B2C authenticated session • User is authenticated in the Migrated App 	<ul style="list-style-type: none"> • User starts sign-in process for Migrated App • App redirects user to Azure AD B2C • User enters credentials into Azure AD B2C • Azure AD B2C calls Rest API to test credentials • Rest API tests credentials against the existing identity provider • Rest API returns success result to Azure AD B2C • Azure AD B2C creates JWT and returns user in an Azure AD B2C authenticated session • User is authenticated in the Migrated App

Bulk User Migration

You can choose this strategy when any of the following apply:

- You have access to user passwords in plain-text or reversible encryption
- You want to force users to use updated password requirements and change their passwords
- You only have one or two applications to migrate

One common strategy is to use a migration script to migrate all the existing users at once. This can be done before or after migrating applications to use Azure AD B2C for authentication.

New or updated apps will use Azure AD B2C as the authentication source, and other existing applications would continue to use the legacy authentication system. If the user credentials are not stored in an accessible manner, then users would be required to reset their password on the next authentication attempt.

This script has no interaction with the existing application. It may read directly from the database containing user identity information, or it may interact indirectly through a new or existing API. By design, the pre-migration script will be discarded after the migration is completed. This code may be responsible for gathering data from several sources about the user to create a complete identity object to migrate. In addition, this code may update the existing database to update the key used to reference user identity.

The migration script should use some persistence method to determine which users have been migrated so that it can be run again. As well, the script should be written to back off in the case a security throttling limit is reached.

This strategy is straightforward to implement and may be the best strategy if there are only one or two applications to modify and cut over.

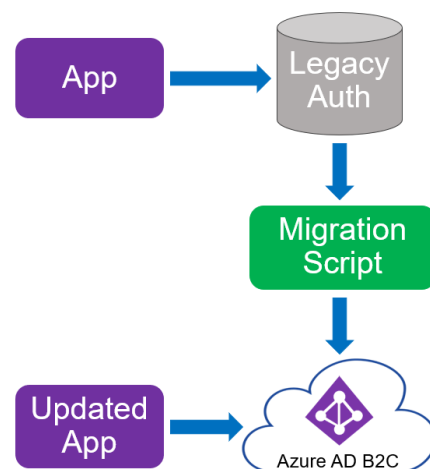
This strategy has challenges when looking at user management between the legacy authentication system and Azure AD B2C. Once the migration is completed, the user stores may diverge as users are created, passwords are changed, or other identity information is updated. Customers may find this confusing as additional applications are switched to use Azure AD B2C for authentication. So, this option works best when there are a small number of apps that are mutually independent from the perspective of the customer.

A sample migration tool is provided here: [Azure Active Directory B2C: User migration](#).

Just-In-Time User Migration

You can choose this strategy when all the following apply:

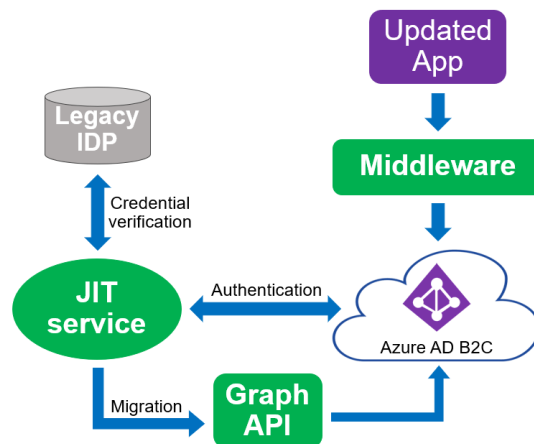
- Your user's passwords are stored in a secure manner
- You want your users to be able to continue using their existing passwords



In a Just-In-Time user migration, you create a service which can verify user credentials against the existing Identity Provider. This allows a migration of users, including their current password, to Azure AD B2C.

This strategy provides the best user experience, as they do not have a disruption to their authentication experience. This strategy is very similar to the “Using Azure AD B2C as Authentication Middleware” described above, especially if you are interacting with a non-federated identity provider.

Following is a sample for JIT Migration that you may be able to adapt: <https://github.com/Azure-Samples/active-directory-b2c-advanced-policies/tree/master/Migration-JIT-sample>



The user and logic flows for a Just-In-Time migration are described below. There are several methods of determining if a user needs to be migrated or updated that you may want to implement based on your business requirements.

Just-In-Time User Migration

- User starts sign-in process for Migrated App
- App redirects user to Azure AD B2C
- Azure AD B2C calls JIT service with credentials
- JIT service verifies credentials against legacy identity provider
- JIT service calls Graph API to create or update user in Azure AD B2C
- JIT service returns login success result to Azure AD B2C
- Azure AD B2C creates JWT and returns user in an Azure AD B2C authenticated session
- User is authenticated in the Migrated App

Maintaining Consistency Between Two User Repositories

You can choose this strategy when all the following apply:

- You want to migrate users and apps at the same time
- You have a business need to maintain two active user repositories

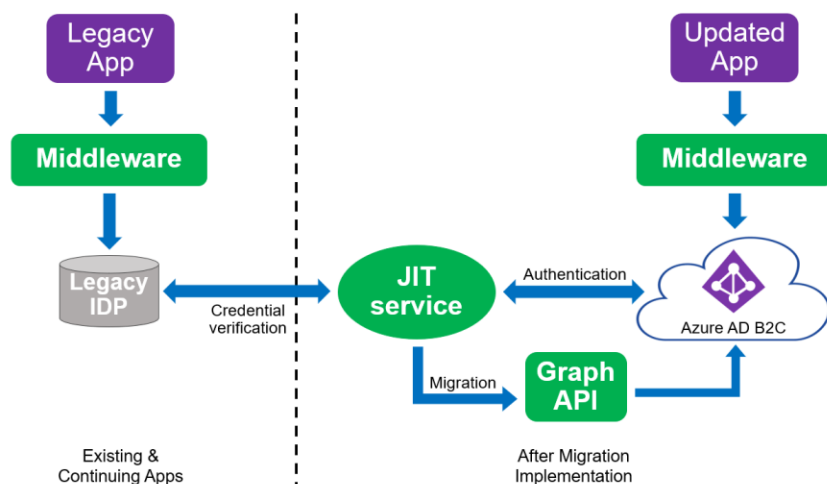
There may be a requirement to maintain both Azure AD B2C and a legacy identity provider at the same time. To handle this scenario properly, you must decide which user repository will be the authoritative source. This decision will drive further implementation details.

One implementation of this strategy combines a just-in-time service with Azure AD B2C to produce a comprehensive authentication and user migration flow. As applications are updated or new ones are added, they authenticate with the Azure AD B2C tenant. This example uses the legacy identity provider as the primary source of authentication.

When a user of the new or updated application first authenticates, the policy gathers the user credentials. These credentials are passed to the JIT migration service using a web service call. The JIT service tests the credentials against the existing legacy authentication service. If the credentials are correct, the user migration status is queried. If the user has not been migrated, a call is made to the Graph API to create the user object. If the user has been migrated, there may be updates that are required to migrate to Azure AD B2C for the user. Finally, the call to the JIT service completes, and Azure AD B2C completes the authentication based on the result of the JIT service. A set of claims is returned to the application and the user can continue with the application in a signed-in status.

This design will let one or more apps utilize the features of Azure AD B2C, while maintaining an existing legacy system. The apps that use Azure AD B2C will realize the benefits of SSO. Once the remaining apps have been migrated to use Azure AD B2C, the policies can be updated to perform the authentication process in Azure AD B2C and no longer call the JIT migration service. At this point, you may choose to migrate the remaining user accounts and require the passwords to be reset, or you may choose to discard the accounts due to age and inactivity.

The table below provides some details on the different user authentication workflows that need to be considered.



User Workflow	Azure AD B2C as primary	Legacy IDP as primary
Sign-In	<ul style="list-style-type: none"> Updated apps authenticate against Azure AD B2C Legacy apps authenticate against Legacy IDP 	<ul style="list-style-type: none"> Updated apps authenticate against Azure AD B2C Azure AD B2C may use Federation or Rest API against the Legacy IDP for authentication Legacy apps authenticate against Legacy IDP
Sign-Up / New User	<ul style="list-style-type: none"> Azure AD B2C handles new user sign-up Azure AD B2C must create new user accounts in Legacy IDP 	<ul style="list-style-type: none"> Sign-Up is disabled in Azure AD B2C or forwarded to Legacy IDP Legacy IDP controls sign-up experience Legacy IDP or sync process must create/copy new users to Azure AD B2C

Edit Profile / Reset Password	<ul style="list-style-type: none"> • Azure AD B2C handles profile edits & password changes • Azure AD B2C updates Legacy IDP profile and password when it is changed via Rest API call • All apps (including apps authenticating against Legacy IDP) must call Azure AD B2C profile edit & password reset flows 	<ul style="list-style-type: none"> • Profile Edit and Password Reset are disabled in Azure AD B2C – user is redirected to Legacy IDP flow • Legacy IDP or sync process must call Graph API to update users in Azure AD B2C • Sync may be done during log-in process as part of JIT migration
--------------------------------------	--	---

Handling Password Migration

Passwords may be stored in a raw format, or with a reversible encryption applied. Many systems store just the hash of the password or store a salted and hashed password. If the password is not programmatically available, the migration of user data will not be able to port over the existing password. Determining whether you wish to migrate passwords may consider many factors, such as:

- Are the existing passwords guaranteed to meet the desired password complexity?
- How have the passwords been stored in the current identity infrastructure?
- Are there already institutional rules in place requiring password resets?
- What are the risks in requiring users to reset or change their password?
- User data, and specifically passwords, can be stored in several manners.

Following are a few password migration strategies that can be employed based on your business use-case and depending on whether you are pre-migrating all users, or migrating them to Azure AD B2C only when they log in (JIT):

- **Password Reset:** Enforcing a password reset when the user first logs in using Azure AD B2C is the easiest, and perhaps most secure way to migrate a user to the cloud. This ensures that the new password meets your latest security and complexity requirements and ensures that the user provided accurate account recovery email addresses in the first place.
- **Migrate Passwords As-Is:** If you have access to the user's password in clear-text, the passwords can be written to Azure AD B2C using Graph. However, this is typically quite rare.
- **Migrate just-in-time:** Please refer to the section above on Just-In-Time .
- **Migrate just-in-time, but force reset on next login if password doesn't meet security standards:** The migration can also be used to update or improve your password complexity requirements. Azure AD B2C allows you to migrate weak passwords into the system, but then enforce more stringent password complexity rules when the user attempts to change their password. This can be used to good effect by writing a flag to user accounts that have been migrated with a weak password.

Considerations When Operating in A Hybrid Identity Environment

Breaking Single Sign-On (SSO) Across Apps

All apps that are moved to Azure AD B2C can be configured to have SSO. However, your users may be faced with a situation where the non-migrated apps do not have an SSO experience with apps that are migrated, as the tokens and cookies used by each app are different.

Advanced Threat Protection Capabilities Aren't Realized

Azure AD B2C has a variety of security controls in place to protect against authentication threats. In addition, you may define multi-factor authentication requirements in the policy, whether that be for every authentication, or just to access the most critical areas of your application. However, these protections will only be in place for the applications using Azure AD B2C. Until the last application is migrated over from the existing authentication infrastructure, the security benefits of the upgrade will remain unrealized. Any architectural vulnerability points in the legacy system are still exposed, albeit with fewer points of entry.

Duplication of Compliance Work

Regulations such as GDPR can be the business motivation to eliminate on-premises user storage entirely. However, to be compliant, you may need to replicate compliance efforts across both the legacy system as well as Azure AD B2C if the legacy system needs to be persisted.

On-going Dependency on Legacy Authentication Provider

Often, keeping the legacy authentication provider running is expensive, or the provider may be out of support. Until all users and apps have been migrated to the cloud, the legacy provider may need to be kept operational, resulting in the added cost of concurrently maintaining two identity providers.

Planning for Updates to Apps

When examining the applications in your environment that are used by customers, there are several questions to ask:

- How do the applications authenticate – is it the same for all or are there differences?
- How many applications are used by customers?
- Are these applications Line of Business applications that have been written by your company, or are they purchased from external vendors?
- If the applications are LOB apps, do you have the source code?
- If the applications are purchased, are there supported updates?
- What languages are the applications written in?
- Are you using an industry standard authentication protocol?

Depending on the source of the application and the usage, there are several options:

- Develop updates for the application
- Maintain the application in the current state
- Retire the application
- Replace the application
- Purchase or obtain upgrade for the application

Updating the App

Often, for business reasons, cost, or since the app was developed a long time ago, you may find yourself in a situation where you need to decide whether it's worth updating the app as part of your identity migration. There are pros and cons to both approaches:

Pros	Cons
<ul style="list-style-type: none">• Upgrade to a standards compliant authentication protocol if legacy app wasn't using one• Centralized authentication and identity management helps move away from custom app-specific identity implementations• Can architect towards a single sign-on experience across all customer-facing apps so that users don't have to login multiple times	<ul style="list-style-type: none">• May be more expensive in the short term• In-house skillset to update existing source-code may be hard to find• Might break existing SSO with your legacy system

The difficulty in updating an application will mostly depend on the type of authentication it is currently using. An application that uses an OIDC compliant endpoint will be simple to update. An application that handles authentication in a co-mingled manner with the business logic is going to be much more difficult to update.

When updating the application, it is best to use middleware authentication libraries. This will separate the authentication logic from the business requirements, and reduce further maintenance related to authentication.

Maintaining the App

Bypassing the Azure AD B2C Authentication Experience

Customers who are either unable to update the existing app's authentication experience or have strong reasons to not redirect the user to an external authentication experience might choose to bypass the Azure AD B2C authentication experience entirely. In this scenario, you accept the user's username and password natively within your app without redirecting to Azure AD B2C, and then [pass the credentials to B2C using a third-party library via an app to service call](#). The [ROPC authentication flow](#) that Azure AD B2C supports works for mobile, desktop and web apps. ROPC for server to server calls are not supported.

There are certain benefits to this approach such as ease-of-implementation, a better user-experience since there is no redirect, and more customizability options. However, this approach is inherently less secure as your app is responsible for maintaining the safety of the user's credentials during the authentication experience. You also forego Azure Identity's out-of-the-box prevention and mitigation of several attack vectors.

Compliance and Data Residency

GDPR Compliance

With the current push towards GDPR compliance, figuring out your GDPR strategy as well as data residency requirements up front is crucial. For a detailed analysis on GDPR requirements, and how to plan for compliance while deploying Azure AD B2C, please refer to the solution guide titled [GDPR Considerations for Customer Facing Applications using Azure AD B2C](#).

Data Residency

Azure AD B2C is available worldwide via the Azure public cloud, in more than 15 instances across the globe. However, the user data resides in one of two geographic locations: EU or US. Consider the following:

- 1) Where you create your Azure AD B2C Tenant: When creating an Azure AD B2C tenant, you have a choice of which region to create it in. If your Azure AD B2C tenant is created in the United States, all copies of your users' data will be stored in the United States. If your tenant is created in Europe, all copies of your users' data will be stored only in Europe.
 - a. For the latest information about where Azure AD B2C's datacenters are available, and which regions have isolated data residency, refer to this article: [Azure Active Directory B2C: Region Availability & Data Residency](#)
 - b. For more information about how Azure AD provides geo-redundancy for its user data, please refer to this article: <https://cloudblogs.microsoft.com/enterprisemobility/2014/09/02/azure-ad-under-the-hood-of-our-geo-redundant-highly-available-distributed-cloud-directory/>
- 2) Use of other supporting data stores: Apart from Azure AD B2C itself, a typical identity deployment tends to also integrate with other CRM systems, blob storage and other supporting systems that support the end-to-end experience.

Implementing an Identity Database Outside Azure AD B2C

In some rare cases, businesses may operate under stringent residency rules that cannot be solved using a distributed identity system such as Azure AD B2C. You can choose to store all your users' attributes, just a small subset needed to identify a user (typically just a username and a password), or even no attributes at all in Azure AD B2C. Some enterprises elect to implement their own user identity database on a platform or service that does comply with local data residency laws and use Azure AD B2C purely as a token broker. While Azure AD B2C does support this model, this approach is a significant undertaking from an implementation standpoint and prevents you from realizing many built-in benefits of the Azure AD B2C platform such as high availability, performance optimizations of using the Azure AD B2C data stores, data security, and the ability to use graph for quick querying of user data.

Localization / Customization

Azure AD B2C has several customization options that enable you to style and brand your app experience. While in some cases, you may choose to leave your user experience completely untouched, a switch-over of the underlying identity platform can also be a good time to consider branding/style refreshes for the apps you are taking to the cloud.

The recommended approach to using Azure AD B2C as your authentication provider is to redirect the actual authentication experience to Azure AD B2C. This involves sending the user to a web-based

authentication page that is served by Azure Identity. These pages are secure and instrumented in many ways to ensure performance and reliability. When redirecting to an Azure AD B2C authentication experience, there are several customizations you can do to ensure every pixel of the experience is consistent with your app's brand and style:

Customization of HTML and CSS of the authentication web experiences: The Azure AD B2C authentication pages can be skinned to look exactly like your own application and brand. Read our online documentation on [Azure AD B2C UI customization](#) to learn how to do this.

Localization of Customer-Facing Text Strings: Azure AD B2C supports 36 different languages out of the box. However, you may also need to update the strings to be more consistent with the tone of your app, or to better suit your specific locale. Azure AD B2C allows you to upload your own language strings that overwrite the default Azure AD B2C localized strings. Refer to our documentation on [language customization](#) to learn more about this topic. For more information, please refer to the following article:

Customization of Authentication URLs: By default, the Azure AD B2C authentication URL points to login.microsoftonline.com. However, some companies prefer that their authentication URLs be from their own domain (aka a vanity domain) such as id.mycompany.com. This feature is currently under development. Additionally, Azure AD B2C supports the [use of b2clogin.com](#) (i.e. yourtenant.b2clogin.com) as an authentication endpoint. For latest information on this and other features coming to Azure AD B2C, refer to the [Azure AD B2C Roadmap](#).

Enablement of JavaScript for fine-grain control of the authentication experience: Currently, the authentication experience is served up on login.microsoftonline.com and doesn't allow the execution of custom JavaScript for security reasons. However, support for JavaScript will be made available in the same timeframe as the functionality around using your own authentication URLs. For latest information on this and other features coming to Azure AD B2C, refer to the [Azure AD B2C Roadmap](#).

Staffing for the Right Roles and Responsibilities

While not a technical factor, an important part of building out your app migration strategy is to ensure you have clearly defined stakeholders to support the migration, and ensure they have the right levels of access and skills to support the initiative. These separations may already exist in your organization. If you have a team dedicated to managing an identity service or platform, they may be the most logical team to take over policy development when working with Azure AD B2C.

Application Development

The application developer will be responsible for the changes to the application to support your business requirements. While changes to support an external authentication provider are not trivial, once they are completed, the developer will no longer need to be responsible for the maintenance of user authentication code. The application developer should work with the policy author to ensure that the identity requirements in the form of claims are supplied to the application. Further, if custom logic is applied when inspecting identity providers, the application developer should ensure that the behavior works properly before adding additional social providers.

Application developers should also be aware that integrating a social identity provider enables them to leverage capabilities related to social media that the legacy identity provider may not have offered. For instance, they may be able to use the social identity provider's token to pull the user's profile picture,

get permission to post on behalf of the user, or get demographic information from the social identity provider's own user graph.

Policy Development

The policy author is responsible for determining the user flow within the authentication process. Policy updates do not necessitate a change to the application. Updates to the policy can add additional social providers, request additional information from the user, or change the validation of user attributes. While the policy author needs to work with the application developer to ensure the right claims are provided to the application, this separation allows for a more agile, iterative development of the user experience.

User Maintenance

Azure AD B2C is designed to allow users to maintain their own identity, including supporting self-service password reset functionality. To the extent that you need to manage users, an administrator within the Azure AD B2C tenant may use the Azure Portal to reset passwords or delete users. However, if you need to provide mid-level admin users such as helpdesk employees a way to do user management without logging into the Azure Portal, you can build a custom admin portal that utilizes the Graph API, or alternatively, buy third-party solutions that offer this capability.

Skills Analysis

There are a variety of engineering skills which will be required when discussing a migration of applications and users to Azure AD B2C. Each of the applications that you currently utilize will need updates – some minor, some major – depending on the current architecture of each. As well, developers with an understanding of claims-based identity are critical in planning the required changes to the applications.

Skill	Purpose
Application codebase language (for each application)	Updating applications to utilize Azure AD B2C as a claims-based identity service
Front-End Development (HTML, CSS)	Development of custom content used as templates in Azure AD B2C
Familiarity with authentication libraries	Updating applications to use authentication libraries such as MSAL and AppAuth .
Web API Development (any language)	Implement web services to support verification and completion of user profile information
Identity Federation	Understand authentication flow between application, federation provider and identity provider
Security Assertion Markup Language (SAML)	Integrate Salesforce, Ping or other SAML identity providers with Azure AD B2C

Testing Your Identity Solution

Testing your Azure AD B2C deployment end-to-end and making sure your rollout strategy factors for any unforeseen circumstances is crucial to a successful launch. Following are some strategies on how to ensure you have the right kind of test coverage for your deployment.

Functional Testing

It is a good idea to set up a pre-production staging tenant that mimics your production experience to fully test your Azure AD B2C deployment before go-live. During development and PoC/pilots of your Azure AD B2C deployment, make sure your tenant is configured to operate in development mode, and enable App Insights for your tenant. App Insights enables you to debug and [troubleshoot custom policies during development](#) and can also be used to [track user behavior by logging authentication events](#) on your tenant.

You may also consider using UI automation tools such as [Selenium](#) to guard against regression of functionality, or to catch service/component outages during the build-out of your solution, and even when your solution is live.

Usability Testing

If you are using the app migration as an opportunity to also update the user interface of your authentication experience, you should consider running a usability testing round with a small user pool to understand if there are any sources of friction for your users as they transition from using the legacy authentication mechanism to the new one.

Policy Testing

The policy defines the user journey during the authentication flow. These should be tested in the same manner an application would be, to ensure that each path through the authentication process works as expected. Further, as the configuration for applications includes the allowed redirect URLs, each policy needs to be tested by the application that will be calling it to ensure the configuration is setup correctly.

Load Testing

Azure AD B2C already scales to billions of authentications per day. Other resources that are called from the authentication flow should be tested to ensure that the anticipated authentication volume can be handled. Hosting these resources in Azure and enabling the proper scaling settings can mitigate some of these concerns. Some resources may reside on-premises and include web services that are called for verification or migration of user data. These resources should be tested to ensure that they can process the required volume of authentications you expect.

Federation Testing

As Azure AD B2C allows the use of external identity providers, the use of these flows need to be tested to ensure correctness. Policies, configuration of external identity providers usually includes setting up a client credentials as well as redirect URLs. These combinations need to be tested, and care needs to be taken to ensure that the configurations are not changed after the testing occurs.

Architect for robustness to failure for scenarios that are adjunct to core auth

Your authentication journey might invoke several different callouts to supporting REST APIs, CRM updates and such. Ensuring your architecture is built in a modular fashion that allows the core authentication and token issuance to succeed while failing other non-essential parts of the journey silently is a good way to prevent your users from being locked out of your app because of a failure in an ancillary component.

For instance, the REST API service can return success or failure codes in a 'Success' or HTTP 200 response that can then be managed by the app instead of resulting in a hard error. Similarly, writes to external systems can be fed into a queue instead of failing the journey if the 'write' operation was not successful.

Roll-Out Planning

Preparing for a smooth roll-out has a lot to do with ensuring your stakeholders are apprised of the upcoming change, and in making sure that you have strategies in place to proactively identify any issues and mitigate them early.

Engage Azure Identity Support

For major deployments with millions of users, proactively reach out to the Azure AD B2C team to ensure your tenant is specifically being monitored during go-live activities for anomalies. This can be done either through escalation paths you have within your Enterprise Agreement such as working with your Microsoft Account Team, or by engaging Product Support Services. You can visit the [Microsoft Support](#) portal to learn more about your support options.

Communication Planning to End Users

Ensure you have carefully thought through the different parts of the identity deployment that end-users will end up coming in direct contact with; and have a strategy for how to proactively let them know about upcoming changes. You may also consider preparing documents and guidance that gets sent to users in case of unforeseen outages as part of the deployment. Some main areas that are worth planning for end-user communication include:

- **Password Reset:** Notification mail that is sent to end-users if you choose to enforce a password reset on them as you migrate to the new identity experience
- **Potential Changes to Single Sign On (SSO) Experience:** Letting users know that SSO across other apps in your ecosystem may not work if you are not doing a full cut-over to the new identity system at launch
- **Retirement or Consolidation of Legacy Apps:** Messaging on your website or apps if the older app is being retired, and how they can move to the new version of your app
- **Disaster Resolution and Remediation Plans:** Guidance for end users in case of a system outage during launch
- **Terms of Use Review:** Informing users about any updates to the Terms of Use that are necessitated by the move to the new identity infrastructure, particularly if you have used the identity migration to ensure compliance with data residency regulations.

Roll-Out Best Practices

Minimizing end-user disruption is key when updating back-end systems such as the authentication provider. Here are some best practices around planning your scale-out:

- Start with one or more of the smaller apps/properties that are less crucial to the business. Alternatively, you might pick an app that is well isolated from the rest of the app-base in terms of its user-base and inter-dependency with other legacy systems
- Begin the roll-out with a series of pilots to user-pools of increasing sizes instead of a Day 1 cutover of your entire user-base

- During the system cut-over, you might need to plan for some amount of down-time. Try to plan the cut-over for a time when you expect low user-load, while still being able to have your technical teams available for troubleshooting.
- Ensure that debug logging is available across all component systems in your architecture, and that no anomalous logs are being generated
- Observe traffic and access patterns to the new system to determine if the traffic has either increased or decreased radically compared to the pre-migration state. Anomalies can point to either lost functionality or to new vulnerabilities being exposed.

Solution support and analytics

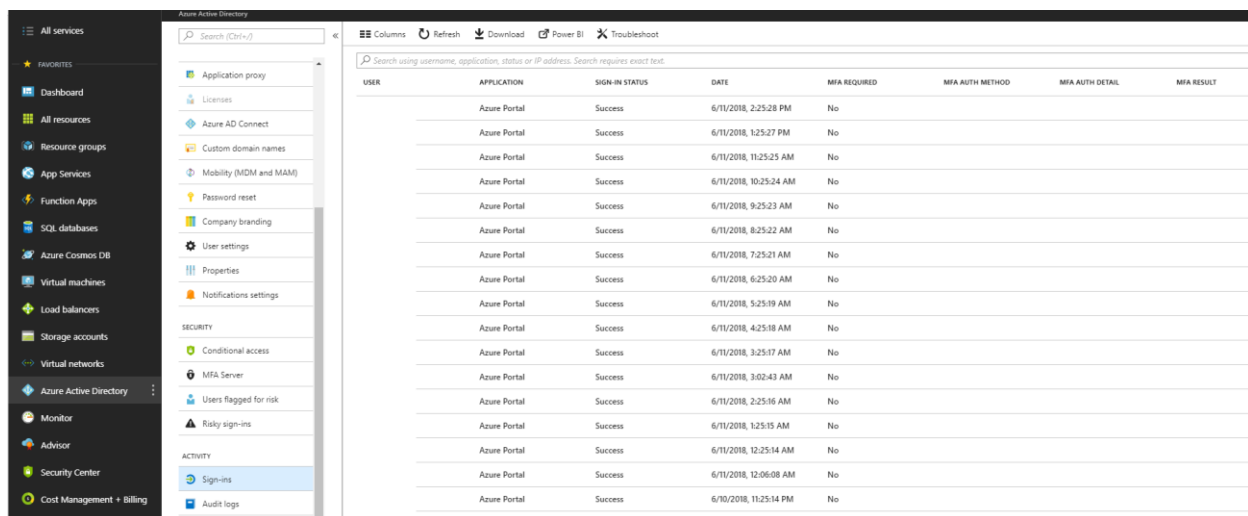
Comprehensive telemetry and analytics are essential to ensure your users can access your apps and services without issue, and can help you identify potential security, scaling, and usability issues proactively. Following are some of the built-in reports and reporting interfaces you can use to assess the health of your overall identity solution.

Reporting and Audit

<https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-faqs>

Sign-In Reports

Sign-in reports provide a record of each sign-in on an Azure AD tenant. You can view these reports in the Azure portal (Azure Active Directory > Activity > Sign-ins)



USER	APPLICATION	SIGN-IN STATUS	DATE	MFA REQUIRED	MFA AUTH METHOD	MFA AUTH DETAIL	MFA RESULT
	Azure Portal	Success	6/11/2018, 2:25:28 PM	No			
	Azure Portal	Success	6/11/2018, 1:25:27 PM	No			
	Azure Portal	Success	6/11/2018, 11:25:25 AM	No			
	Azure Portal	Success	6/11/2018, 10:25:24 AM	No			
	Azure Portal	Success	6/11/2018, 9:25:23 AM	No			
	Azure Portal	Success	6/11/2018, 8:25:22 AM	No			
	Azure Portal	Success	6/11/2018, 7:25:21 AM	No			
	Azure Portal	Success	6/11/2018, 6:25:20 AM	No			
	Azure Portal	Success	6/11/2018, 5:25:19 AM	No			
	Azure Portal	Success	6/11/2018, 4:25:18 AM	No			
	Azure Portal	Success	6/11/2018, 3:25:17 AM	No			
	Azure Portal	Success	6/11/2018, 3:02:43 AM	No			
	Azure Portal	Success	6/11/2018, 2:25:16 AM	No			
	Azure Portal	Success	6/11/2018, 1:25:15 AM	No			
	Azure Portal	Success	6/11/2018, 12:25:14 AM	No			
	Azure Portal	Success	6/11/2018, 12:06:08 AM	No			
	Azure Portal	Success	6/10/2018, 11:25:14 PM	No			

Figure 1 Built-in sign-in reports

Audit Reports

Azure AD generates audit logs containing activity information about resources, issued tokens, and administrator access. Audit reports on both admin and application activity are available in the Azure portal (Azure Active Directory > Activity > Audit logs). For Azure AD B2C, audit reports also contain rich data about authentication, email and MFA activity by end-users. For more information about the data available via audit reports for Azure AD B2C, click here: <https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-reference-audit-logs>

DATE	TARGET(S)	INITIATED BY (ACTOR)	ACTIVITY
6/10/2018, 10:21:09 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 10:19:37 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 10:19:36 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 5:28:02 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 5:27:04 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 5:23:25 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 5:20:58 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 5:18:16 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 5:18:15 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 5:17:16 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 5:15:50 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 5:12:55 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 4:52:15 PM	User : bf2d	9775393	Issue an id_token to the application
6/10/2018, 4:50:15 PM	User : bf2d	9775393	Issue an id_token to the application

Figure 2 Built-in admin audit logs

Usage Reports

Azure AD provides authentication based on user sign-in and Azure Multi-Factor Authentication for end users of your application family across identity providers. When you know the number of users registered in the tenant, the providers they used to register, and the number of authentications by type, you can answer questions like:

- How many users from each type of identity provider (for example, a Microsoft or LinkedIn account) have registered in the last 10 days?
- How many authentications using Multi-Factor Authentication have completed successfully in the last month?
- How many sign-in-based authentications were completed this month? Per day? Per application?
- How can I estimate the expected monthly cost of my Azure AD B2C tenant activity?

Usage reports are only available via the Usage Reporting API and are not available via the Azure portal. They include number of users, number of logins, and volume of MFA.

For more information on how to use Azure AD B2C's reporting API to generate usage reports, click here: [Azure AD B2C Reporting API](#)

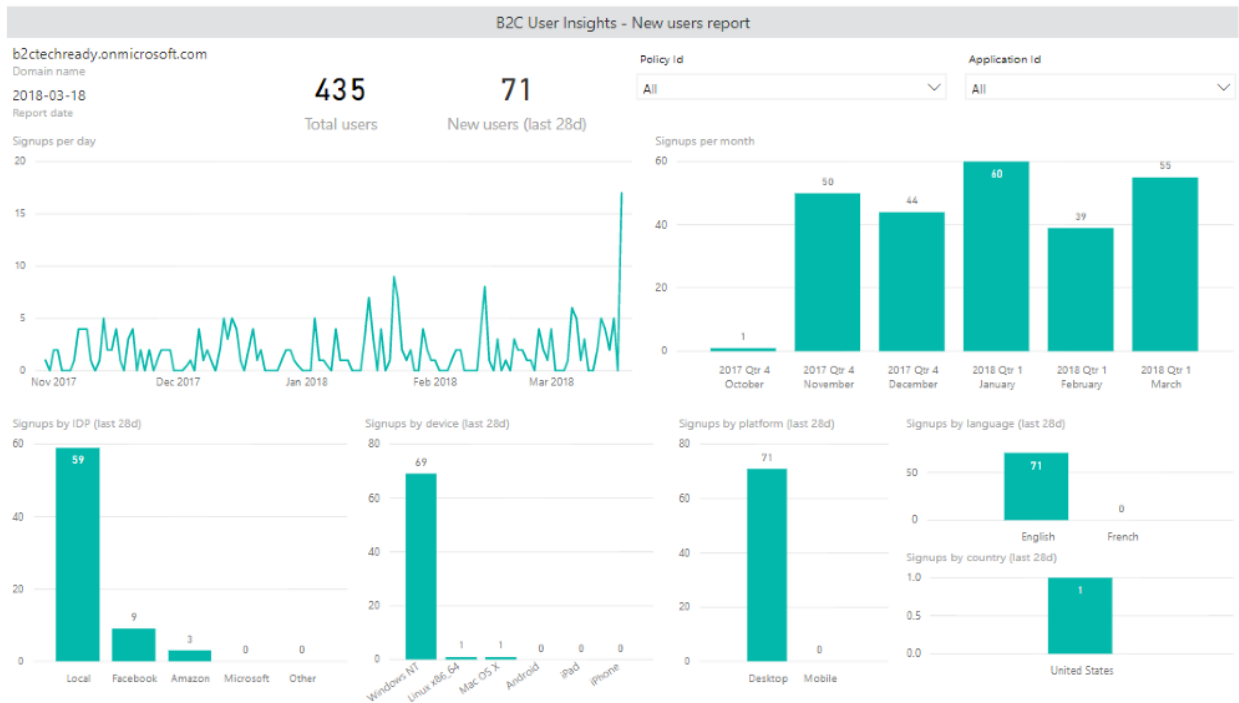


Figure 3 A Power BI dashboard generated using data from B2C's Reporting API

Additional Resources

Azure Support: Depending on your Enterprise Agreement with Microsoft, you can call Microsoft Support and open a ticket for any issue related to your Azure Identity deployment. For more information on how to get in touch with Microsoft Support, please visit our Azure support portal:

<https://azure.microsoft.com/en-us/support>

FastTrack: If you have purchased Enterprise Mobility and Security (EMS) licenses or Azure AD Premium licenses, you may be eligible to receive deployment assistance from the FastTrack program. For more information on how to engage with FastTrack, please refer to our documentation on the [FastTrack Center Eligibility Benefit for Enterprise Mobility and Security](#)

Engage the Product Engineering Team: If you are working on a major customer deployment with millions of users, you can work with your Microsoft account team or your Cloud Solutions Architect to decide if the project's deployment complexity warrants working directly with the Azure Identity Product Engineering team.

EMS Blog: Subscribe to the [EMS Blog](#) to stay up to date with all the latest product announcements, deep dives, and roadmap information provided directly by the Identity engineering team. Further, you can also post comments and get feedback from the engineering group.

Azure Active Directory Public Forums: Azure AD also has several closely monitored channels available to the public. Here are some useful links:

- StackOverflow using the tag '[azure-ad-b2c](#)'
- [UserVoice](#) to submit or vote on new feature requests in Azure AD B2C

- Azure AD B2C Partner Portal: <http://aka.ms/b2cpartnerportal> (available to partners only)
- [MSDN Forum for Azure AD](#)

Code Samples on GitHub: [Azure AD B2C](#)

Training: [Gaining Expertise with Azure AD B2C](#)

© 2018 Microsoft Corporation. All rights reserved. This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

Some examples are for illustration only and are fictitious. No real association is intended or inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes. You may modify this document for your internal, reference purposes

It is understood and agreed to that project plan may provide certain information that is and must be kept confidential. To ensure the protection of such information you should not disclose any part of this plan to anyone unless required to do so by law.