okta Identity Engine

Deploying app-level policies

How to use this guide

The guide is meant to help you understand app-level policies in the Okta Identity Engine (OIE) and show you the options for their implementations. Each use case is independent, so you can jump right to any use case.

Who this guide is for

The guide is meant for technical implementers who design, test, and deploy Okta. This guide provides:

- Requirements and use-cases for the new policy framework
- Core concepts and product capabilities
- Prescriptive steps to deploy app-level policies
- Best practices and product recommendations

Document history

Date	Version Description
December 7, 2021	1.0 Initial publication

Why do we need app-level policies?

In the ever-changing world of technology, every organization strives to find the right balance between the highest levels of security, best user experience, and improved agility. Organizations want to enable frictionless access to resources while reducing the risk of security incidents and adopting modern security frameworks while continually factoring in contextual and adaptive policies for accessing apps.

User behavior has also fundamentally changed. It is now common for users to access critical organizational resources on their personal devices. Organizations face the challenge of enabling such access without the risk of data loss.

What can app-level policies do?

App-level policies allow organizations to model security outcomes for application access based on industry-accepted digital identity best practices (outlined by NIST). App-level policies are deployed in conjunction with your org-level policy and allow granular control at the time of access to a specific application.

With app-level policies, organizations can enable contextual conditional access to their apps based on the risk and security posture when the user accesses the application. App-level policies enforce end-user authentication in the context of the requested application. The user's location and profile (also identified by the Okta sign-on policy) are verified against the app's sign-on policy's group membership and authentication criteria.

Each application can have one policy only, and by default, every app in your org has a sign-on policy already in place. Like other policies, app-level policies are built on rules, and if you don't add any yourself, each app authenticates against a default *catch-all rule*.

By default, the catch-all rule allows access using a single factor when you create a new application. Remember that the definitions of factor and authenticators have changed, and a sign-on policy with two-factor authentication now requires two distinct factors. Two authenticators of the same factor (Knowledge, Possession, or Inherence) are not acceptable; you must use a different factor for each authenticator. For example, a policy requiring two factors cannot be satisfied by using the Email and Phone authenticators, since they both belong to the Possession factor type. For instructions on setting up authenticators, refer to Add an authenticator.

Add an app-level Sign-on policy rule to customize the level of application access. For example, you may want to challenge all default Okta users to provide a password, but challenge Okta users in a designated group to provide both password and email factors. To accomplish this, you can create a rule that challenges default users to provide a password, and then create a second rule that challenges all group members for email and password authenticators.

Keep in mind that the catch-all rule governs behavior for devices that do not meet the "If" conditions of other rules you define. If you plan to have an app-level policy in place that denies access to devices that don't meet certain criteria (such as Registered devices), you must configure a rule that allows access to devices that **do** meet the criteria, **while also** configuring the catch-all rule to deny all other devices.

Benefits of app-level policies

Some of the benefits that app-level policies provide are:

1. Passwordless authentication to specific resources.
2. More robust authentication methods even if the user has less robust methods enrolled (for example, SMS OTP, Email OTP, or Voice OTP).
3. Customization of the number of factor types required per app.
4. Customization of the strength of the authenticator(s) required per app, for example, phishing-resistant, bound to devices, etc.
5. Customization of factor lifetime on a per-app basis.

In addition, the Okta administrator determines which authenticators and authentication methods are available. If a policy requires an authenticator the end-user has not enrolled in already, Okta Verify will prompt the end-user to enroll dynamically.

The administrator can do the following using app-level policies.

- Require higher security for critical apps.
- Require phishing-proof or hardware-bound authenticators.
- Allow passwordless authentication.
- Allow strong factors on in-app transactions.
- Leave some combinations of factors undefined.
- Align better with NIST guidelines.
- Let end-users choose which factor is most convenient to them.
- Require fewer factor prompts. If an end-user has already provided a strong factor for a critical app and then sign in to a non-critical app with less stringent authentication requirements, they will not be prompted again.

Strategic considerations when deploying app-level policies

App-level policies in OIE are a great way to implement a holistic approach to Identity Access Management (IAM) focused on results and outcomes. Okta designed app-level policies to allow Okta administrators to operate at a higher level of abstraction by defining the context and user type depending on the application's requirements.

Key concepts and definitions

App-level policies, along with Device Context and Okta FastPass, are a cornerstone of the Okta Identity Engine. When evaluating whether a user can be granted access, OIE inspects the context (user itself, device, network, and risk) that the user brings (first at org level and then at app level) and determines the authentication methods that will be offered based at both the org-level and app-level policies.

Authenticators and authentication methods

In OIE, we are using a new term instead of factor: **authenticator**. An authenticator is “how” the end-user authenticates (“authenticate with …”). An authenticator is a credential owned or controlled by an end-user -- for example, a secret phrase, possession of an email account, or an instance of a downloaded app.
An authenticator can have one or more **authentication methods**. Examples of authenticators and the authentication methods they offer include:

Authenticator	**Authentication Methods**
Phone	SMS or voice
Password	Secret string or phrase
Okta Verify	One time password (OTP), push notification or proof of possession key, FastPass

Factor type

Authenticators have a **factor type** that can be defined by something the user **knows** (called a knowledge factor), by something the user **has** (called a **possession** factor), or something the user is (called an **inherence** factor).

Authenticator characteristics

In addition, the introduction of authenticators allows for the addition of specific security characteristics that the authenticator can offer.
The characteristics currently supported are for the **Possession** factor type are listed here. Each is valuable to achieve specific outcomes.

- Hardware protection. Requires that keys being used to authenticate are stored in secure hardware (TPM, Secure Enclave) on the device. Currently, only Okta Verify meets this constraint. Because hardware protection implies Device bound, that constraint is selected automatically when Hardware protection is selected.
- **Phishing resistant**. Requires users to provide possession factors that cryptographically verify the login server (the origin). Currently, only FIDO2/WebAuthn satisfies this requirement. Because phishing resistance implies Device bound, that constraint is selected automatically when Phishing-resistant is selected.

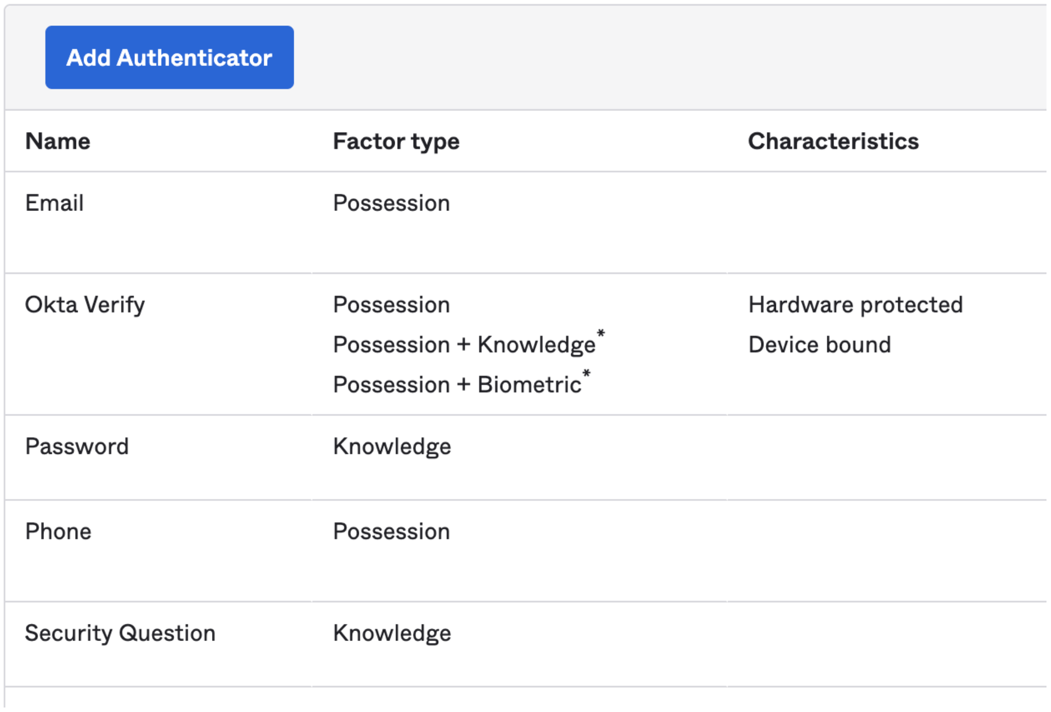
- **Device bound**. Requires the factor's keys to be stored securely on the device and aren't transferable to another device without re-enrolling the factor. Email and SMS are the only possession factors that aren't device bound. This constraint is selected automatically if either of the other constraints is selected.

Authenticator assurance levels (AAL)

Authenticator Assurance Levels are included in the National Institute of Standards and Technology (NIST)'s guidelines for digital identity, defining the level of confidence an app and/or service has that the credential presented for access is truly in the possession of the person whose identity is being asserted to the app/service. For example, is the one-time password (OTP) or biometric information presented to the service actually tied to the username entered for access?

NIST recommends three levels of Authentication Assurance Levels (expressed as AAL1, AAL2, and AAL3) that represent the strength of the authentication process and the binding between an authenticator and the user's identity. AALs define the characteristics of the authentication methods that need to be verified before a specific level of authentication is achieved, such as the number of factors presented, whether the factors are required to be phishing resistant, etc.

Here's an example of a typical set of authenticators, their supported factor types, and their characteristics, taken from the Okta admin console.



Name	Factor type	Characteristics
Email	Possession	
Okta Verify	Possession Possession + Knowledge* Possession + Biometric*	Hardware protected Device bound
Password	Knowledge	
Phone	Possession	
Security Question	Knowledge	

Authentication assurance policies

When creating an app-level policy in OIE, the administrator can specify the combination of authenticators, authentication methods, and factor types that are required, allowing them to define the proper assurance level, including:

- How the user must authenticate, using one factor type or two factor types
- What constraints (specific characteristics required for the authentication method) to be applied to possession factors
- Whether access with Okta FastPass is allowed, and under which constraints (prompt or biometrics)

This constitutes the authentication assurance policy for the app to which the policy is applied.

Context

When a user presents themself to the authentication process, it brings a context with them. This context is a set of information and state about:

- The user itself, its groups if any, and any risk assessment that is attached to them
- The network the user is connected to
- The device the user is using

Refer to the documentation for a complete description of these aspects, as well as to the Device Context Deployment Guide.

Device Context

Device Context is another cornerstone of OIE. It provides maximum visibility on the information and state that the device presents to assess the authentication methods required to grant access to Okta-managed Apps. Device Context is a superset of Device Trust.

With Device Context, the Okta administrator has access to a rich set of data about the user's device:

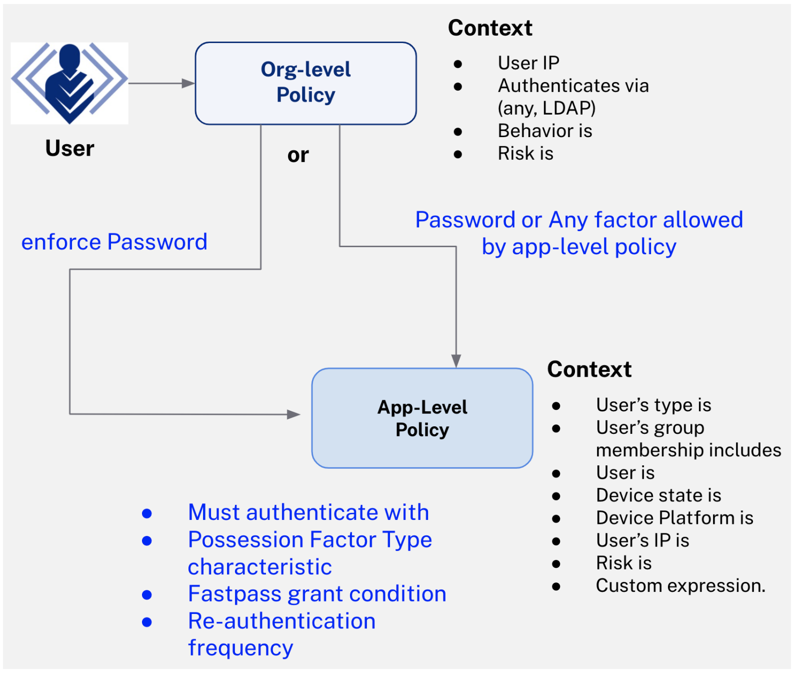
- Context data and state for managed devices, which is also referred to as 'Device Trust'. Device Trust in OIE provides a unified approach for assessing the trust posture of desktop and mobile devices, including:
    - Indication on whether the device is managed by a 3rd party management tool
    - Data or signals collected by 3rd party EDR tools, for example, is a firewall or antivirus present on the device
- Whether the device is registered in Okta Universal Directory

This information can be used to create sophisticated policies.

Org-level policies and app-level policies

It is important to understand that the Okta org-level policy is still the first ‘gate’ for policy evaluation. That policy now includes the ability to delegate to the app-level policy for authentication method selection.

The diagram represents a simple view of the flow between org-level, app-level policies, and the context available to craft the policy.



Understanding how Okta sign-on policies and application sign-on policies work together

In Okta Identity Engine, to satisfy an authentication request for a SAML or OIDC application, constraints specified in the Okta Sign-on Policy (OSOP) that describes the Okta session-wide policy and Application Sign-on Policy (ASOP) that describes the assurance requirements for an app are evaluated together to satisfy assurance requirements. If a valid Okta session already exists, only the application sign-on policy is evaluated.

To understand how these concepts work together, let’s use the following setup for an Okta tenant:

- OSOP is set to “Password/Any Factor/IdP”, requires multi-factor is unchecked and session expiration is set to 2 hours.
- Application 1 ASOP is set to “Any 1 Factor Type / IdP” and re-authentication frequency is set to 2 hours.
- Application 2 ASOP is set to “Any 2 Factors Types” and re-authentication frequency is set to 2 hours.
- Application 3 ASOP is set to “Any 1 Factor Type / IdP” and re-authentication frequency is set to 1 hour.

When the user tries to access Application 1, because an Okta session does not exist yet, OSOP is evaluated and because “requires multi-factor” is unchecked, the user is only prompted for a single factor as per the requirements set on Application 1’s ASOP. The user can use any of the available authenticators that they have enrolled in to satisfy Application 1 assurance requirements (and let’s assume they use their Password for this example). Now let’s assume that the user tries to access Application 2 within the same browser window, and immediately after they access Application 1. Because Application 2 requires two distinct factor types, and because they already supplied their password within the Okta session, they will only have to supply another factor type (for example, Okta Verify Push) to satisfy the application assurance requirements for Application 2.

Now, let’s assume the user has another session (e.g. on a different device/browser) where the user was able to access Application 1 using their password as a factor. If the user was to try to access Application 3 after 1 hour of inactivity, they would be prompted to re-verify with any of their enrolled factors, which includes Password, despite the fact that they had already verified with their password previously in the session because the ASOP for Application 3 sets the re-authentication frequency to a value that is shorter than the validity of the Okta session specified in OSOP.

What do you need to consider before deploying app-level policies?

Here are the key considerations to shape your approach to implementing app-level policies:

1. First, consider your applications: some might be simple and have low security requirements (think about a portal to the company cafeteria). In contrast, others will have much higher security risks (financial transactions, intellectual property). It is good practice to start with a simple model containing three assurance levels: low, medium, high.

2. Consider your users, and put them into categories based on the intrinsic risk they bring to the organization. Employees might have different clearance levels; some 3rd parties might go through security screening, others might not. These categories will

be the foundation to create your user groups in OIE.

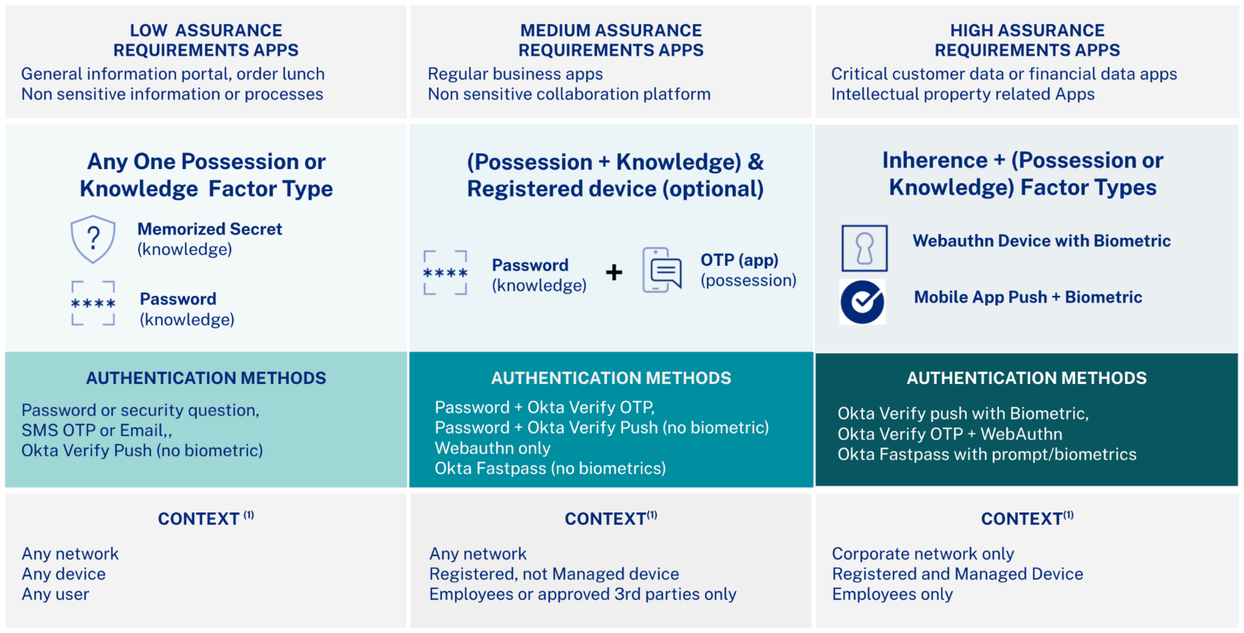
3. Consider the devices your users will bring in to access applications. Personal devices or corporate devices are obvious cases; also consider public devices (or friends of an employee's device).

4. Consider the network requirements: corporate, home network, public network.

5. Consider the authenticators and authentication methods you have available and can enable for your users. This consideration should include an assessment of the user experience you will deliver.

6. Finally, educate your users. Users are the last line of defense. Giving them a basic understanding of the policies and the reason behind them will make them better supporters.

Defining your app-level policy strategy is a multi-dimensional exercise that requires rigor and attention to detail. The diagram below represents an example model for a fictitious organization.



LOW ASSURANCE REQUIREMENTS APPS General information portal, order lunch Non sensitive information or processes	**MEDIUM ASSURANCE REQUIREMENTS APPS** Regular business apps Non sensitive collaboration platform	**HIGH ASSURANCE REQUIREMENTS APPS** Critical customer data or financial data apps Intellectual property related Apps
Any One Possession or Knowledge Factor Type **Memorized Secret** (knowledge) **Password** (knowledge)	**(Possession + Knowledge) & Registered device (optional)** **Password** (knowledge) + **OTP (app)** (possession)	**Inherence + (Possession or Knowledge) Factor Types** **Webauthn Device with Biometric** **Mobile App Push + Biometric**
AUTHENTICATION METHODS Password or security question, SMS OTP or Email,, Okta Verify Push (no biometric)	**AUTHENTICATION METHODS** Password + Okta Verify OTP, Password + Okta Verify Push (no biometric) Webauthn only Okta Fastpass (no biometrics)	**AUTHENTICATION METHODS** Okta Verify push with Biometric, Okta Verify OTP + WebAuthn Okta Fastpass with prompt/biometrics
CONTEXT [1] Any network Any device Any user	**CONTEXT**[1] Any network Registered, not Managed device Employees or approved 3rd parties only	**CONTEXT**[1] Corporate network only Registered and Managed Device Employees only

Example of a strategy to deploy app-level security to protect your applications.

(1) "Context" here describes the complete context that the user presents to authenticate. It includes the device context, user identity, and groups, as well as network context.

Additional considerations

App-level policies are a paradigm shift in IAM for enterprise apps. There are a few additional considerations that must be kept in mind when articulating your strategy and before deployment:

- IAM is the last line of defense before accessing applications or resources. When designing policies, consider the first line of defenses such as your Internet Service Provider (ISP), the routers in your corporate network, and your firewalls. The context presented by a user might already have been filtered by other defenses (IP address ranges, geolocation, for example), yet, in a Zero-trust scenario, you might still want to assess the context again in the policy itself.
- Make sure you truly understand the org-level policy to app-level policy flow. If you choose to delegate to 'any factor allowed by app-level policy', pay extra attention to the app-level policy. For example, if the app-level policy is set to 'Any 1 factor' and email is enrolled as an authenticator, then the user will have access granted with a simple magic link email.

Use cases

This guide describes three typical scenarios that organizations may implement based on their specific IAM and security requirements. These examples are for illustration only and are not recommendations. The use cases provide different scenarios with progressive implementation of the full power of app-level policies.

1. Okta sign-on policy
2. Getting started with app-level policies
3. Full app-level policy deployment

Okta sign-on policy

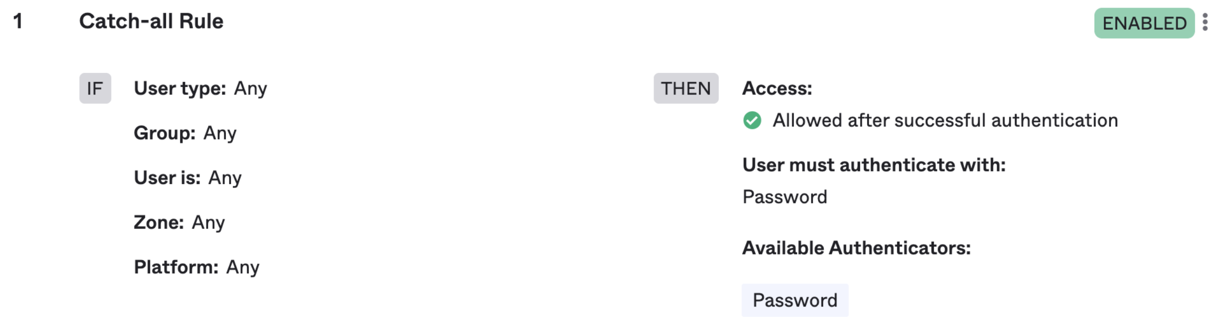
The Okta Identity Engine still provides an Okta sign-on policy that you can use in lieu of app-level policies. In this scenario, the work is at the org level. All apps require a password and a secondary authentication method (from any authenticator made available). All apps have a default **Catch-All** rule.

Note: We included this use case so you can see how to use an Okta sign-on policy in a similar fashion to how you use Okta sign-on policies in Okta Classic, but this use case does not use app-level policies. Everything is done by the Okta sign-on policy.

You can also optionally configure an app-level policy that requires one hardware-protected authentication method for an app with more rigorous security requirements. This app-level policy will limit the authenticators for the second authentication method to Okta Verify.

Setup and Configuration

- You can turn off Okta FastPass if it's already turned on in your org, but it won't be used by an Okta sign-on policy anyway so it's ok to leave it on.
- You can optionally add a user and group to assign the Okta sign-on policy to, but it's not required.
- Create an app integration so that the end-user has something to sign in to. If you're just testing the scenario, you can use a bookmark app to see how it works. In your scenario, the app would be one of your business apps. The app will use all default values and will also use the default Catch-all rule so it applies to all users. It will look similar to this.



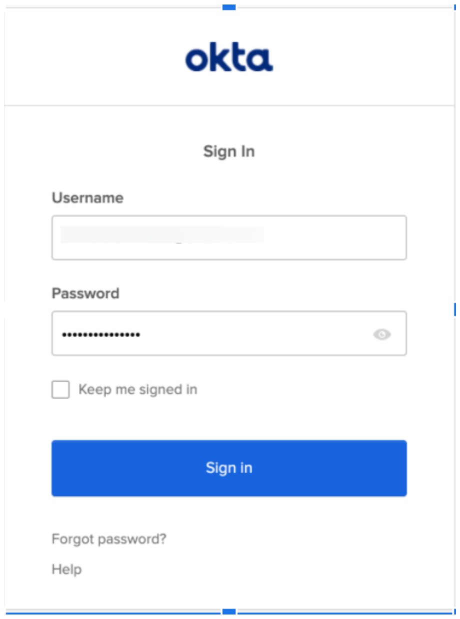
1 **Catch-all Rule** ENABLED

IF
User type: Any
Group: Any
User is: Any
Zone: Any
Platform: Any

THEN
Access:
✓ Allowed after successful authentication
User must authenticate with:
Password
Available Authenticators:
Password

User sign-in experience

When an end-user signs in to the app, you will see the Okta sign-in Widget (SIW) asking for a name and password. This is because the default catch-all rule only allows a password authentication method.


okta

Sign In

Username

Password

•••••••••••••••

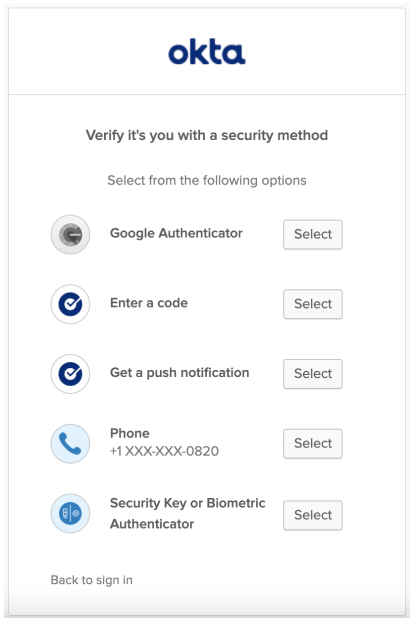
Keep me signed in

Sign in

Forgot password?

Help

After the end-user enters a password, they will see a dialog with choices for additional authentication methods that satisfies the org-level policy. The list contains the authenticators that the end-user has enabled and will look something like this.


okta

Verify it's you with a security method

Select from the following options

Google Authenticator — Select

Enter a code — Select

Get a push notification — Select

Phone +1 XXX-XXX-0820 — Select

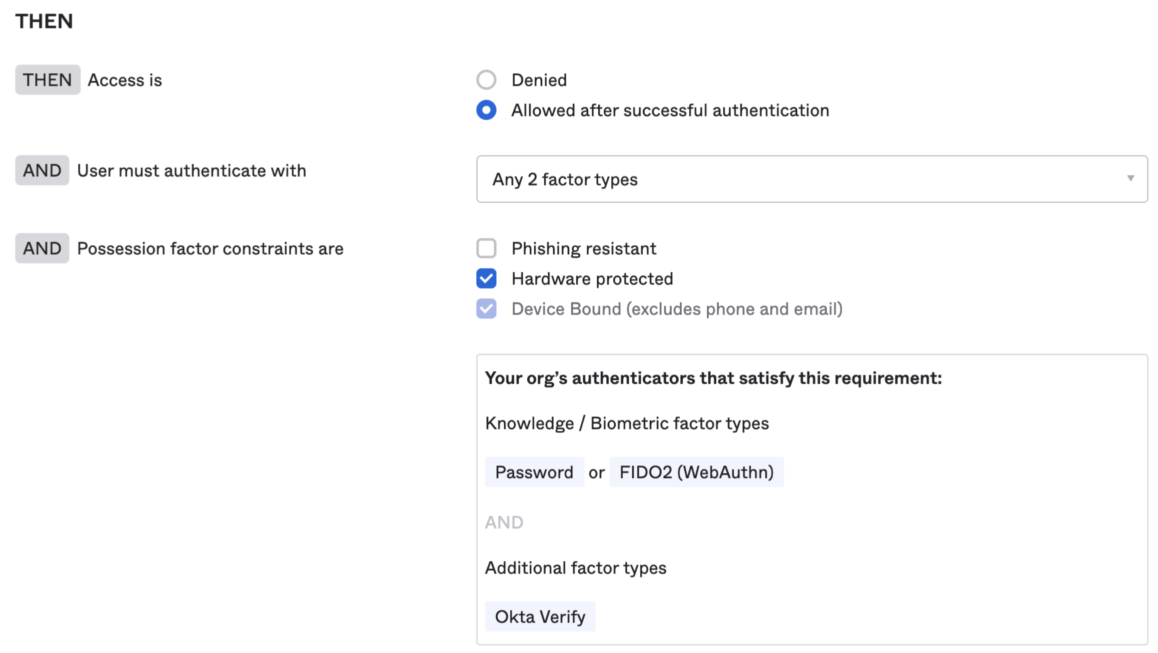
Security Key or Biometric Authenticator — Select

Back to sign in

Optional: add a higher assurance app

For one special higher assurance app, add an app-level policy that requires a hardware protected-authentication method, which limits the authenticators to **Password or**

FIDO2(WebAuthn) + **Okta Verify** in our scenario. Your rule will look something like this.


THEN

THEN Access is
- Denied
- [✓ Allowed after successful authentication]

AND User must authenticate with: Any 2 factor types

AND Possession factor constraints are
- Phishing resistant
- [✓ Hardware protected]
- [✓ Device Bound (excludes phone and email)]

Your org's authenticators that satisfy this requirement:

Knowledge / Biometric factor types

Password or FIDO2 (WebAuthn)

AND

Additional factor types

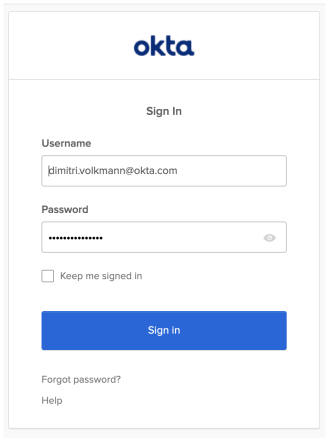
Okta Verify

User sign-in experience

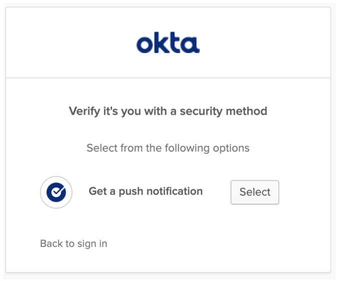
You can see the difference in the user sign-in experience. For example, when a user signs in to a low assurance app, they need two authentication methods to sign in: both password and phone (SMS) are required. However, factors are not equal in terms of the protection they offer. You can require different factors to deploy different assurance levels. For example, if you change the requirements from a password to Okta Verify, it's more secure even though we haven't added a second factor.
Note: Okta Verify is both an inherence and possession factor type when you use biometrics.

You can see this in action by using different authenticators when signing in.

1. Sign in to the bookmark app. You should see the **password** and prompt.



2. Sign in to the higher assurance app you optionally created. Depending on what you have configured for the end-user, you should see a verification prompt for a push notification or other authenticator.


okta
Verify it's you with a security method
Select from the following options
Get a push notification
Select
Back to sign in

3. In a new browser (or incognito tab), sign in to the bookmark app again, using password and Okta Verify.
4. Sign in to the higher assurance app. You will not see an additional prompt because Okta Verify satisfies the app-level policy.

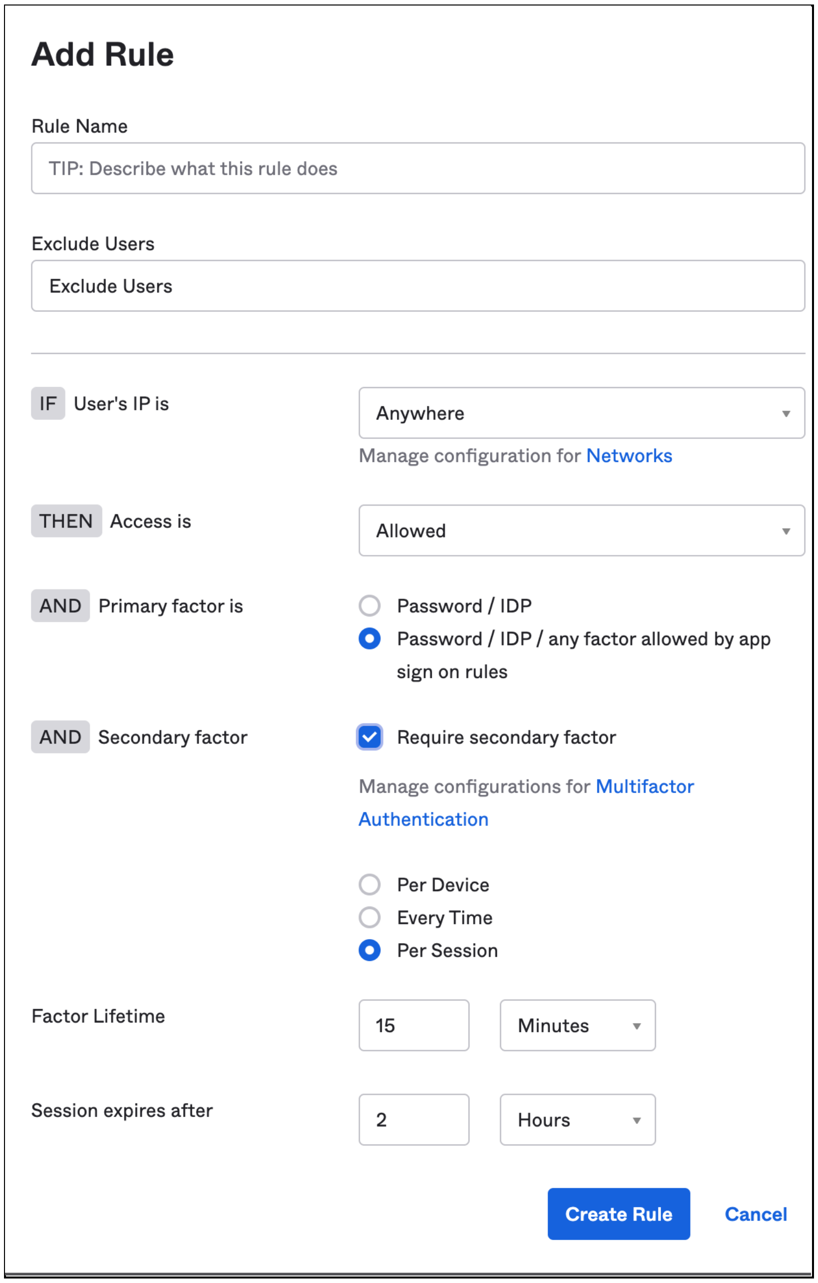
Getting started with app-level policies

In this use case, you'll set up an app policy for a higher level of assurance that comes from the type of factor used. For example, a knowledge factor like a password is not as secure as an inherence factor like a fingerprint. A secondary authentication method at the org level is now required.

Setup and Configuration

Create an Okta sign-on policy

1. In your admin console, go to **Security** > **Okta Sign-on Policy**.
2. Click **Add New Okta Sign-on Policy**.
3. In the **Add Policy** dialog, add a **Policy Name** and optional **Policy Description**.
4. In the **Assign to Groups** field, choose **Everyone**, then click **Create Policy and Add Rule**.
5. In the **Add Rule** dialog, in the **Primary factor is** field, **select Password/ IDP/ any factor allowed by app sign on rules**.
6. In the **Secondary factor** field, check the **Require secondary factor** checkbox and then click **Create Rule**. Your rule should look similar to this.



Add Rule

Rule Name

TIP: Describe what this rule does

Exclude Users

Exclude Users

IF User's IP is: Anywhere

Manage configuration for Networks

THEN Access is: Allowed

AND Primary factor is: ○ Password / IDP [✓ Password / IDP / any factor allowed by app sign on rules]

AND Secondary factor: [✓ Require secondary factor]

Manage configurations for Multifactor Authentication

○ Per Device ○ Every Time [✓ Per Session]

Factor Lifetime: 15 Minutes

Session expires after: 2 Hours

Create groups

Create a group for sign-on and add a person to it. We're using bookmark apps for demonstration purposes, but you'd use whatever app integration you need for each app.

1. In your admin console, go to **Directory** > **Groups** and click **Add Group**.
2. In the **Add Group** dialog, enter a **Name**, for example, **Group for low assurance app**, and optionally a **Group Description** and click **Save**.
3. Go to **Directory** > **People** and click **Add person**.

4. In the **Add Person** dialog, add a **First name**, **Last name**, **Username**, and **Primary email**. Add your own email to the **Secondary email** field.
5. In the **Groups** field, select the group you created in step 2.
6. In the **Password** field, select **Set by admin**, enter a password, and deselect **User must change password on first login**.
7. Click **Save**.
8. Repeat this process for another group (for the high assurance app).

Create bookmark apps

1. In your admin console, go to **Applications** > **Applications** and select **Browse App Catalog**.
2. In the **Search** box, type *bookmark app*, select **Bookmark App** and click **Add**.
3. On the **Add Bookmark App** screen, change the Application label to **Bookmark for low assurance app**.
4. Enter a **URL** and click **Done**. Because this app is for demonstration purposes only, you can choose any url you like. In a real environment, you would use the url of the app you're setting up SSO for.
5. Select the **Assignments** tab.
6. Select **Assign** > **Assign to Groups** and select **Assign** for **Group for low assurance app**. Don't click the group name unless you want to review the group properties.
7. Click **Done**.
8. Repeat this process for another bookmark app that you will use to create the second app-levl sign-on policy (**Bookmark for high assurance app)**.

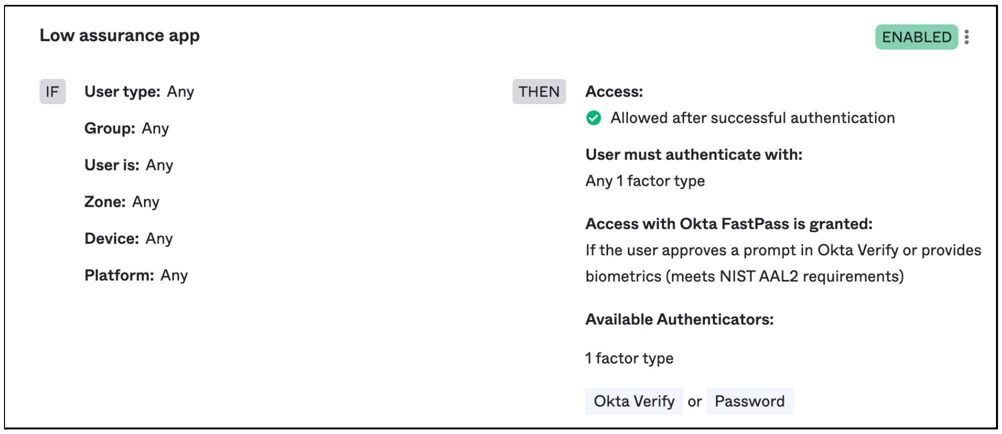
Create app sign-on policies

Create an application sign-on policy that requires one factor type and another that requires two factor types. When creating an app sign-on policy, you should consider who the policy applies to. Which specific users, groups, user types, or specific users should this policy apply to?

App sign-on policy for low assurance app

1. In your admin console, go to **Applications** > **Applications** and select **Bookmark for low assurance app**.
2. Select the **Sign On** tab.
3. You will see a default sign-on policy with a **Catch-all Rule** that requires passwords.
4. Select **Add Rule**.
5. In the **Add Rule** dialog, add a rule name, for example, **Low assurance app rule**.
6. In the **User's user type is** field, select **Any user type**.
7. In the **User's group membership includes a field**, select **At least one of the following groups**.

8. Start typing the name of the group you created in the prerequisites and then select it, for example, **Group for low assurance app**.
9. In the **User is** field, select **Any user**.
10. In the **Device State** field, select **Any**.
11. Use default values for **Device Platform(Any platform)** and the **User's IP**(**Any IP)**. Also, leave the "**The following custom expression is true"** field blank. You can use the Okta Expression Language (EL) to add a custom expression to an app sign-on policy.
12. For the "**User must authenticate with"** field, choose **Any 1 factor type.** The app-level sign-on policy presents the user with a list of their enrolled authenticators (including password), allowing them to pick whichever one they want to use.
13. In the **Access with Okta FastPass is granted** field, choose **If the user approves a prompt in Okta Verify or provides biometrics**.
14. In the **Re-authentication frequency field**, choose **Every sign-in attempt**. That means the authenticator has to be verified every time the user accesses the app. If you choose **Re-authenticate after:**, the authenticator will re-authenticate if it hasn't been used within the given timeframe. The re-authentication frequency timestamp indicates the start of the authorization period and does not change when the authorization is reused for another app. It changes when the authorization period expires and needs to be reauthorized.
15. Click **Save**. You should have a rule that looks similar to this.


Low assurance app

ENABLED

IF

User type: Any

Group: Any

User is: Any

Zone: Any

Device: Any

Platform: Any

THEN

Access:
✓ Allowed after successful authentication

User must authenticate with:
Any 1 factor type

Access with Okta FastPass is granted:
If the user approves a prompt in Okta Verify or provides biometrics (meets NIST AAL2 requirements)

Available Authenticators:

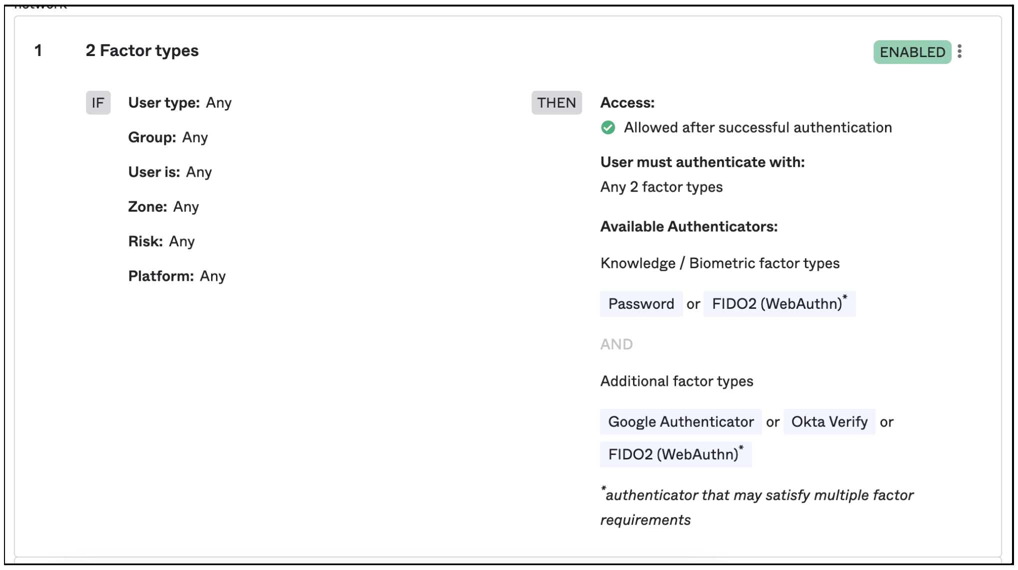
1 factor type

Okta Verify or Password

App sign-on policy for high assurance app

1. In your admin console, go to **Applications** > **Applications** and select **Bookmark for high assurance app**.
2. Select the **Sign On** tab.
3. You will see a default sign-on policy with a **Catch-all Rule** that requires passwords.
4. Select **Add Rule**.
5. In the **Add Rule** dialog, add a rule name, for example, **High assurance app rule**.
6. In the **User's user type is** field, select **Any user type**.

7. In the **User's group membership includes a field**, select **At least one of the following groups**.
8. Start typing the name of the group you created in the prerequisites and then select it, for example, **Group for high assurance app**.
9. In the **User is** field, select **Any user**.
10. In the **Device State** field, select **Any**.
11. Use default values for **Device Platform(Any platform)** and the **User's IP(Any IP)**. Also, leave the "**The following custom expression is true"** field blank. You can use the Okta Expression Language (EL) to add a custom expression to an app sign-on policy.
12. For the "**User must authenticate with"** field, choose **Any 2 factor types.** The app-level sign-on policy presents the user with a list of their enrolled authenticators (including password), allowing them to pick whichever one they want to use.
13. In the **Access with Okta FastPass is granted** field, choose **If the user approves a prompt in Okta Verify or provides biometrics**.
14. In the **Re-authentication frequency field**, choose **Every sign-in attempt**. That means the authenticator has to be verified every time the user accesses the app. If you choose **Re-authenticate after:**, the authenticator will re-authenticate if it hasn't been used within the given timeframe. The re-authentication frequency timestamp indicates the start of the authorization period and does not change when the authorization is reused for another app. It changes when the authorization period expires and needs to be reauthorized.
15. Click **Save**. You should have a rule that looks similar to this.



1 2 Factor types ENABLED

IF User type: Any
Group: Any
User is: Any
Zone: Any
Risk: Any
Platform: Any

THEN Access:
✓ Allowed after successful authentication
User must authenticate with:
Any 2 factor types
Available Authenticators:
Knowledge / Biometric factor types
Password or FIDO2 (WebAuthn)*
AND
Additional factor types
Google Authenticator or Okta Verify or
FIDO2 (WebAuthn)*
*authenticator that may satisfy multiple factor requirements

User sign-in experience

When a user attempts to launch an app governed by an app-level policy, their experience is determined by the assurance level of the policy. Assurance levels are determined by business needs; these are examples only.

Low assurance app:

Launch the application from any device. You should see the prompt for one authentication method.

High assurance app:

Launch the application from any device. You should see the prompts for two authentication methods consecutively.

Catch-all app:

Launch the application from any device. You should only see the prompt for a password.

Switch between low and high assurance:

Launch the low assurance app, then launch the high assurance app. You should see a request for additional authentication.

Launch the high assurance app, then launch the low assurance app. You should be able to open the low assurance app without additional authentication.

Turn on FastPass:
Optionally, turn on FastPass so you can see the end-user experience.

1. Open the admin console for your tenant.
2. Navigate to **Security** > **Authenticators**.
3. In the **Authenticators** section, enable **FastPass using Okta Verify** by selecting **Actions** > **Edit**.
4. On the **Okta Verify** screen, in the **Verification options** section, select **Okta Fast (All platforms)**.
5. In the **Okta FastPass** section, select **Show the "Sign in with Okta FastPass" button**.

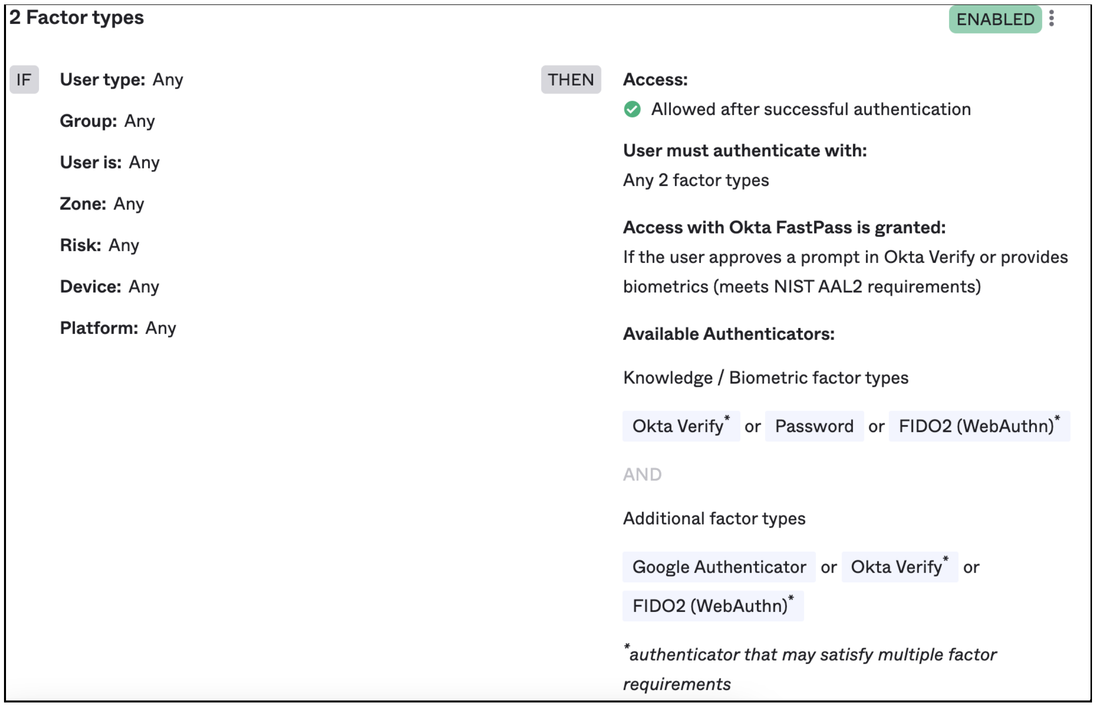
Selecting **Show the "Sign in with Okta FastPass" button** does three things.

1. It walks first-time users through installing Okta Verify and registering a device.
2. It allows an alternative if the end user's configuration doesn't permit silent sign-on. For example, mac users without a device management solution like Jamf Pro or a

safari SSO browser extension will not be able to sign in silently. Enabling the button allows these users a way to sign in.

3. It acts as a backup if Okta Verify doesn't load automatically.

After you've turned on FastPass, you can see it in app-level sign-on policies under **Available Authenticators**.



2 Factor types

ENABLED

IF

User type: Any

Group: Any

User is: Any

Zone: Any

Risk: Any

Device: Any

Platform: Any

THEN

Access:

✓ Allowed after successful authentication

User must authenticate with:
Any 2 factor types

Access with Okta FastPass is granted:
If the user approves a prompt in Okta Verify or provides biometrics (meets NIST AAL2 requirements)

Available Authenticators:

Knowledge / Biometric factor types

Okta Verify* or Password or FIDO2 (WebAuthn)*

AND

Additional factor types

Google Authenticator or Okta Verify* or FIDO2 (WebAuthn)*

**authenticator that may satisfy multiple factor requirements*

End-users can now log in to the higher assurance app with Okta FastPass only, assuming they have biometrics enabled.

Full app-level policy deployment

This is a more advanced app-level policy. In this scenario, the organization has a more sophisticated approach with three different levels of security requirements for their apps: Low, Medium and High. This strategy relies on Device Context (see Device Context Deployment Guide) in addition to the authentication method requirements. The table below summarizes the authentication and Device Context settings that satisfy each security level requirement.

Note: in this use case, Okta FastPass, OIE's new Passwordless solution, is enabled.

	Low assurance apps	Medium assurance apps	High assurance apps
Authentication requirements	Any 1 Factor Type	Any 2 Factor Types	Any 2 Factor Types + Hardware Protected
Device Context	Any	Registered devices	Registered AND Managed devices
Okta FastPass option[1]	Silent auth	Silent auth	Prompt
Re-authentication frequency	Never if session active	Every 2 hours	Every sign-in attempt

[1] For more details about Okta FastPass consult the Passwordless Deployment Guide

Prerequisites

Activate Okta FastPass

1. In the Okta Admin Console, go to **Security > Authenticators**
2. In the Okta Verify row, click **Actions > Edit**
3. Ensure that **Okta FastPass (All platforms)** is checked

Ensure the Okta sign-on policy in use does not require a secondary factor

1. In the Okta Admin Console, go to **Security > Okta Sign-on Policy**
2. In the left pane, select the sign-on policy in use
3. Click the ✏ icon to the right of the appropriate rule
4. Ensure that **Require secondary factor** is unchecked
5. Repeat the steps above for additional sign-on policy and rules as needed

Setup and configuration

1. In the Okta Admin Console, go to **Applications > Applications**
2. Click the app that needs the app-level policy applied
3. Click the **Sign On** tab
4. In the **Sign On Policy** section, click **Add Rule**
5. In the **Rule name** field, enter a name for the rule
   Note: the name should be descriptive (for example, “Salesforce Low Security”). This is helpful when consulting system logs to troubleshoot access issues.

6. Apply settings for the desired assurance level (as detailed in the table above):

Low assurance apps

1. In the **IF** section:
    a. Set **Device state** to **Any**
2. In the **THEN** section:
    a. Set **Access is** to **Allowed after successful authentication**
    b. Set **User must authenticate with** to **Any 1 factor type**
    c. Set **Access with Okta FastPass is granted** to **Without the user approving a prompt in Okta Verify or providing biometrics**
    d. Set **Re-authentication frequency is** to **Never re-authenticate if the session is active**
3. Click **Save**

Medium assurance apps

1. In the **IF** section:
    a. Set **Device state** to **Registered**
2. In the **THEN** section:
    a. Set **Device management is** to **Not managed**
    b. Set **Access is** to **Allowed after successful authentication**
    c. Set **User must authenticate with** to **Any 2 factor types**
    d. Set **Access with Okta FastPass is granted** to **Without the user approving a prompt in Okta Verify or providing biometrics**
    e. Set **Re-authentication frequency is** to **Re-authenticate after** and set the period to **2 hours.**
3. Click **Save**

High assurance apps:

1. In the **IF** section:
    a. Set **Device state** to **Registered**
2. In the **THEN** section:
    a. Set **Device management is to Managed**
    b. Set **Access is** to **Allowed after successful authentication**
    c. Set **User must authenticate with** to **Any 2 factor types**
    d. Set **Possession factor constraints are to Hardware protected**
    e. Set **Access with Okta FastPass is granted** to **If the user approves a prompt in Okta Verify or provides biometrics (meets NIST AAL2 requirements)**
    f. Set **Re-authentication frequency is** to **Every sign-in attempt**
3. Click **Save**

Important note: in this scenario, the medium and high assurance levels are expected to deny access from devices that are not registered or managed, respectively. In order to accomplish this, the Catch-all rule for the app's policy must be configured to deny access. This rule will be applied to devices that do not meet the IF conditions set in the above rules.

To configure the Catch-all rule:

1. In the Okta Admin Console, go to **Applications > Applications**
2. Click the app that needs the app-level policy applied
3. Click the **Sign On** tab
4. In the **Catch-all Rule** field, click the ⋮ icon and select **Edit**
5. In the THEN section, set **Access is** to **Denied**

User sign-in experience

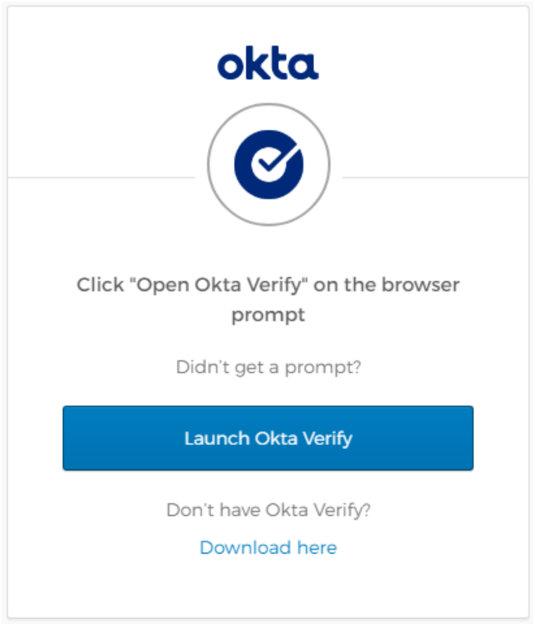
When a user attempts to launch an app governed by an app-level policy, their experience will be determined by which of the above security levels has been applied to the policy. Note that the same user may have different experiences when launching different apps, based upon their policies.

After configuring apps using the settings above, try the following:

Low assurance app
When a user launches the app from any device, only one authentication method will need to be satisfied before access is granted. Entering a password satisfies this requirement.

Medium assurance app

1. When a user launches the app from an **Unregistered** device they will not be able to proceed until Okta Verify is installed and launched on the device, as shown below:



Click "Open Okta Verify" on the browser prompt

Didn't get a prompt?

Don't have Okta Verify?

2. When a user launches the app from a **Registered** device they will need to satisfy two factor types before they can proceed to the app. For example, they may enter a password (Knowledge factor) and respond to an Okta Verify push (Possession factor).
3. If users close and relaunch the app two afters after they first launched the application, they will be prompted to authenticate.

High assurance app

This is similar to the medium assurance app experience, except:

1. When a user launches the app from an **Unregistered,** a **Registered - Not Managed** device, or a device that is not hardware protected, they will not be able to proceed until both requirements are met.
2. Users will need to respond to both factor prompts every time they launch the protected app.