# Azure Multi-Factor Authentication (MFA) and Two-Factor Authentication (2FA)

Multi-factor authentication (MFA) is a security approach requiring users to present two or more forms of verification at sign-in. In practice, Azure MFA typically combines **something you know** (a password) with **something you have** (a phone or authenticator app) or **something you are** (biometrics). Two-factor authentication (2FA) is a subset of MFA that uses exactly two factors. For example, a common 2FA scenario is entering a password and then approving a push notification on a mobile app. Adding this extra factor greatly increases security – even if an attacker steals a password, they cannot sign in without the second factor.

### Authentication Methods in Azure MFA

Azure AD supports many MFA methods. These include: Microsoft Authenticator app (push notifications or one-time passcodes), one-time codes via text message (SMS) or voice call, hardware OATH tokens, and FIDO2 security keys (like biometric USB keys). Azure also supports passwordless methods (FIDO2 keys or Authenticator app passkeys) which count as strong factors. For self-service password reset (SSPR), additional methods like **email** or **security questions** may be used (security questions are only used for SSPR). In all cases, Microsoft recommends that users register multiple methods. For example, the Microsoft Entra combined registration feature prompts users to enroll both MFA and SSPR methods together, so if one factor becomes unavailable, they have backups.

### Configuring Azure MFA

To **enable Azure MFA**, administrators typically use Conditional Access policies or the built-in Security Defaults. Microsoft recommends creating a Conditional Access policy (requires Azure AD P1/P2 license) that targets specific users or groups and **requires MFA on sign-in**. For example, an admin can go to the Entra (Azure AD) admin center, navigate to **Security > Conditional Access**, and create a new policy that includes the desired user group and sets **"Grant: Require multi-factor authentication"**. When users in that group sign in to protected apps or the Azure portal, they will be prompted for a second factor.

For tenants without Conditional Access (or on the free tier), enabling **Security Defaults** (found under Azure AD > Properties) will enforce MFA for all users on high-privilege or riskier sign-ins. Alternatively, administrators can enable MFA per-user in the legacy way: in the Azure AD admin center, browse to **Users**, select a user, and choose **Enable MFA**. This per-user enablement forces the user to register an MFA method on their next sign-in. (Note: per-user MFA is not needed if you use Conditional Access or Security Defaults.)

**Managing Azure MFA Settings and Users**

Administrators with the **Authentication Policy Administrator** or **Global Administrator** role can manage MFA for users. In the Azure AD admin portal under **Users**, an admin can view a user's **MFA settings**. Each user account has an MFA **state**: **Disabled** (not enrolled), **Enabled** (user must register at next sign-in), or **Enforced** (user must complete MFA at sign-in). Enabling a user (turning on MFA) moves them to Enabled; once they register a method, they become Enforced. Administrators can switch users between these states or reset their MFA registration if needed. It's recommended that users be allowed to register multiple verification methods, so that if one method isn't available (e.g. lost phone) they can use another.

Administrators can also reset or revoke a user's MFA methods (for example, to force re-registration). Azure AD provides built-in reports and the Microsoft Graph API/PowerShell for viewing which users have registered which methods. In summary, to manage users: sign in to the Entra admin center, go to **Users > All users**, select a user, and view or change **Authentication methods**. After changes, click **Save**. Users will then see prompts to register or re-register on their next login.

**Account Lockout and Smart Lockout**

Azure AD includes protection against brute-force sign-in attempts. Under **Azure AD > Security > Authentication methods > Password protection**, admins can configure **Smart Lockout** settings. You set a **lockout threshold** (the number of failed attempts before lockout) and a **lockout duration**. For example, the default is 10 failed attempts before locking the account for 60 seconds. This helps prevent password-guessing attacks. Once the lockout period elapses, the user may try again.

In older on-premises MFA configurations, there was a separate “Account lockout” setting (in the classic Azure MFA Server) for securing the second-factor attempts. In modern Azure AD, however, Smart Lockout is the primary defense. (In short: configure lockout under the Azure AD Password Protection settings.)

**Extending Azure MFA to On-Premises and Third-Party**

Azure MFA can secure not only cloud sign-ins but also on-premises resources and third-party applications. For VPNs and RADIUS-based sign-ins, Microsoft provides the **NPS extension for Azure MFA**. This is installed on a Windows NPS (RADIUS) server in your network. When a VPN user authenticates (supplying username/password), the NPS extension forwards a second-factor request (via Azure AD) to the user’s registered method (phone call, SMS, or app notification). Only if the user approves the MFA challenge does the NPS server allow the VPN login. In other words, you can require Azure cloud MFA for on-prem VPN or Remote Desktop Gateway access.

For on-premises Active Directory Federation Services (AD FS), Azure MFA can be integrated as an additional authentication provider. Microsoft provides migration guides for moving from the legacy Azure MFA Server (an on-prem component) to cloud-based MFA. In fact, **Azure MFA Server is deprecated**; new deployments should use cloud MFA. Customers using MFA Server should migrate to cloud MFA via the provided utilities.

Similarly, any third-party or on-premises application that uses Azure AD for SSO (via SAML/OIDC or Azure AD Application Proxy) can also require MFA through Conditional Access. In practice, this means once an application trusts Azure AD as the IdP, enforcing MFA for that application is done the same way as for any Azure app.

**Monitoring Azure MFA Activity**

To review MFA usage and troubleshoot, use the **Azure AD Sign-in logs**. Each sign-in event in the logs shows whether an MFA challenge occurred and details of the authentication. In the Azure portal, go to **Azure Active Directory > Users > Sign-in logs**. The log entries include columns for **Authentication Details** and **Conditional Access**, indicating which factors were used and which policy applied.

For a given sign-in, click **Authentication Details** to see the sequence of steps: you’ll see each factor tried, whether it succeeded or failed, and any error reasons. This allows admins to answer questions like: “Was MFA challenged? What method did the user use? Did they fail the MFA?” The logs also summarize how many MFA sign-ins occurred, success rates, and which methods were most common. By filtering the sign-in logs (for example, filtering by "AuthZ Method = MFA"), you can quickly find all events involving MFA.

In addition, **Azure AD Audit Logs** capture MFA-related configuration changes (like enabling MFA) and SSPR events (see below). For MFA-specific reports, the Entra admin center provides built-in pages under “Monitoring” that can visualize MFA registration and usage trends.

## OAuth 2.0 Tokens in Azure AD

Azure AD authentication uses OAuth 2.0 tokens. When a user signs in to an app, Azure AD issues an **access token** (short-lived) and a **refresh token** (longer-lived, up to 90 days). The client uses the refresh token to get new access tokens without prompting the user again. In practice, if a user’s password is changed or reset (or an admin revokes tokens), Azure AD will **revoke** the user’s refresh tokens. This forces the user to reauthenticate (enter credentials again). Since Conditional Access may require MFA on reauthentication, revoking tokens indirectly forces a new MFA challenge if a refresh token is invalidated. For example, if a user resets their password via SSPR, their previous tokens are revoked (per the Azure token revocation rules), so next time they must sign in and complete MFA.

On domain-joined (Azure AD Joined) Windows devices, Azure issues a special **Primary Refresh Token (PRT)** to enable single sign-on across apps. This means a user signs in once on the device (with MFA as required), and most managed applications use the PRT for SSO without re-prompting for credentials. However, Azure AD session and sign-in frequency policies can still force reauthentication. In summary, Azure uses OAuth tokens to maintain sessions, but any trigger (token expiration, password reset, conditional access policy) can force a fresh sign-in and thus another MFA prompt.

## Self-Service Password Reset (SSPR)

Microsoft Entra (Azure AD) **Self-Service Password Reset (SSPR)** allows users to reset or unlock their own accounts without contacting IT support. When SSPR is enabled, a locked-out or forgotten-password user can click the password reset link (passwordreset.microsoftonline.com) and follow on-screen prompts. They prove their identity by using the extra authentication methods they previously registered (similar to MFA). SSPR greatly reduces help-desk workload and downtime, since users regain access on their own.

**Configuring and Deploying SSPR**

To set up SSPR, a Global Admin or Authentication Policy Administrator signs into the Azure portal and goes to **Azure AD > Password reset (under Protection)**. In the **Properties** page, choose whether to enable SSPR for *None*, *Selected* groups, or *All* users. For a phased rollout, it's common to select a pilot group first. For example, select "Selected" and then pick an Azure AD group (e.g. "SSPR-Test-Group") to enable SSPR only for those users. Click **Save** to apply.

Next, configure the **authentication methods**. Still under **Password reset**, go to the **Authentication methods** tab. Set the "Number of methods required to reset" (commonly 2). Then choose which methods the organization will allow. The typical options are **Mobile app notification**, **Mobile app code**, **Email**, **Mobile phone (SMS/voice)**. (Additional options include Office phone or Security questions if desired.) Check the boxes for each allowed method and click **Save**.

Finally, configure **registration options** so that users supply their contact info. You can require users to register by enabling "Ask users to register when they sign in". This prompts users to enter their phone number, email, and app registration info the next time they sign in. Alternatively, admins can pre-populate user contact info. Once enabled and users have registered, SSPR is live for those users.

**SSPR Authentication Methods**

The methods allowed for SSPR are similar to MFA. Azure can send an SMS code or phone call, or push notifications to the Authenticator app, and can send a verification code to an alternate email address. Unlike MFA, SSPR supports **security questions** (if the user is a cloud-only account, though this is deprecated in many regions). In practice, we recommend the stronger methods: at least one mobile device method (app or SMS) and an alternate email. Users must set these up in advance during registration.

When a user requests a password reset, Azure prompts for the required number of verification steps (e.g., a code to phone plus a code to email). Only if they enter the correct codes (or approve the app notification) can they proceed to choose a new password.

For security, Microsoft again recommends combined registration of MFA and SSPR methods, so that users enroll all their methods at once. This way, if a user forgets their password, they already have multiple means (e.g. phone and email) to verify their identity via SSPR.

**Monitoring SSPR Activity**

Azure provides reporting and audit logs for SSPR. In the Azure portal's **Azure AD > Users > Audit Logs**, you can filter by **Service = Self-service Password Management**. This shows all SSPR-related events. Key event types include:

- **Reset password (self-service)** – user successfully reset their own password.
- **Change password (self-service)** – user changed password voluntarily or expired-out-of-turn.
- **Blocked from self-service password reset** – user was throttled (e.g. too many attempts in 24h).
- **User registered for self-service password reset** – user completed their registration info for SSPR.

These logs let admins track usage (e.g. how many resets were done in a week) and identify issues (like repeated failures). Azure also offers built-in SSPR reports (under **Identity > Password Reset**) that visualize how many users have registered, how often resets occur, and method usage. In short, you can monitor SSPR by reviewing the audit entries and built-in reports to ensure it's functioning and to spot any suspicious activity.