



TECHNICAL SUPPORT COURSE

Microsoft Entra ID: Integrations & Access Troubleshooting

Ten chapters on diagnosing and fixing the failures that stop a client reaching your system through Microsoft Entra ID: how integrations work, credentials and expiry, permissions and consent, the sign-in logs, safe fixes, prevention and a triage playbook.

10 Chapters

Philip Osztromok

Generated with Claude

Contents

Chapter 1	What Entra ID Is — and Where It Sits	2
Chapter 2	How Applications Integrate	13
Chapter 3	What Passes Between Entra and Your System	23
Chapter 4	Credentials and Expiry	36
Chapter 5	Finding the Integration in the Entra Admin Center	45
Chapter 6	Permissions and Consent	55
Chapter 7	Reading the Sign-in Logs	68
Chapter 8	Diagnosing and Fixing an Expired Secret, Certificate or Token	81
Chapter 9	Preventing It Happening Again	92
Chapter 10	Capstone: A Triage Playbook	101

Microsoft Entra ID: Integrations & Access Troubleshooting

Course 1 · Chapter 1 · What Entra ID Is — and Where It Sits

A client rings: nobody on their side can sign in to your system. Their staff can still open Outlook and Teams without trouble, so the problem isn't their internet or their computers. Somewhere between the sign-in button and your system, a connection to **Microsoft Entra ID** has stopped working. This course teaches you to find out where, to read the evidence, and to explain it to the client. This first chapter gives you the map: what Entra ID is, what lives inside it, and where the pieces you'll be troubleshooting sit.

WHAT THIS CHAPTER DOES AND DOESN'T DO

No portal clicking yet, except for one small exercise. The aim is that, by the end, when someone says "the integration in Entra expired", you can say *whose* tenant it's in, *what* the integration is for, and *who* has to fix it. The details of integrations, tokens and logs follow in Chapters 2 to 9.

Identity, Authentication and Authorization

Three words come up in every Entra conversation, and they are not interchangeable. Microsoft's own definitions:

TERM	WHAT IT MEANS	EXAMPLE
Identity	A collection of identifiers or attributes that represents a person, software component, machine or other resource in a system.	A work email address and the account behind it
Authentication (AuthN)	Challenging someone or something for credentials to prove that it is who or what it claims to be.	Password plus an MFA prompt
Authorization (AuthZ)	Deciding whether an authenticated identity may access a particular resource, and at what level.	"This user may open the support system, but not its admin pages"

Authentication comes first and authorization after it. Microsoft's documentation also notes that the information from authentication is carried in an **ID token** and the information from authorization in an **access token**. Don't worry about tokens yet; Chapter 3 follows them step by

step. For now, remember the pair of questions every sign-in answers: *Who are you?* and *Are you allowed in?*

WHY THIS MATTERS FOR SUPPORT

"I can't sign in" can mean two different failures. Either the person couldn't prove who they are (authentication), or they were identified but not permitted (authorization). The fix, and the person who can do it, differ. Later chapters teach you to tell them apart from the logs. See also the Authentication & Session Security course for the general ideas of sessions and credentials.

What Microsoft Entra ID Is

Microsoft Entra ID is a **cloud-based identity and access management service**. Microsoft describes it as providing authentication, policy enforcement and protection for users, devices, apps and resources. In plain terms it is the organisation's central list of who exists and what they may reach, together with the service that checks credentials when they try.

That makes it an **identity provider** (IdP): one trusted place that creates, stores and manages identity information, so that every other system doesn't have to keep its own list of usernames and passwords. Those other systems hand the sign-in to the identity provider and trust its answer. Your product, when it offers "Sign in with Microsoft", is exactly such a system.

A client's user

clicks "Sign in with
Microsoft" on your page

->

The client's tenant

Microsoft Entra ID
checks who they are
(password, MFA, policies)
decides if they may go on

<-

sends a signed answer
(tokens or an assertion)

Your system

-> trusts the answer
lets the user in

Three consequences follow, and they explain most of this course. The **decision to let someone in is made in the client's Entra tenant**, not in your system. Your system's job is to ask correctly and to check that the answer really came from Entra. And when the connection between the two breaks, **every user of that client is affected together**, while other clients, with their own tenants, carry on as normal.

The Name: Azure AD Became Entra ID

You'll meet both names. Microsoft renamed **Azure Active Directory (Azure AD)** to **Microsoft Entra ID**. The change was publicly announced on 11 July 2023, and the new name started to appear across Microsoft experiences from 15 August 2023; plan and SKU display names changed on 1 October 2023. Microsoft says the rename was meant, among other things, to reduce confusion with Windows Server Active Directory.

WHAT MICROSOFT SAYS DID NOT CHANGE

It was a rename, not a new product: existing deployments, configurations and integrations continue to work, and Microsoft states that all existing **login URLs, APIs, PowerShell cmdlets and Microsoft Authentication Libraries (MSAL) stay the same**. Licensing, terms, SLAs and pricing also stay the same. That is why you will keep seeing "Azure AD" or "AAD" in older articles, in scripts, in your own product's documentation and in some error messages. Treat them as the same thing as Entra ID.

OLD NAME	CURRENT NAME
Azure Active Directory / Azure AD / AAD	Microsoft Entra ID
Azure AD tenant	Microsoft Entra tenant
Azure AD Premium P1 / P2	Microsoft Entra ID P1 / P2
Azure AD Conditional Access	Microsoft Entra Conditional Access (second use: Conditional Access)
Azure AD Connect	Microsoft Entra Connect
Azure AD admin center / Azure portal's Azure AD blade	Microsoft Entra admin center (entra.microsoft.com), or Microsoft Entra ID in the Azure portal
Azure AD enterprise application / service principal	Microsoft Entra enterprise application / service principal
Azure AD activity logs / audit log	Microsoft Entra activity logs / audit log

A few names were deliberately **not** changed: Windows Server Active Directory and its related services (AD DS, AD FS), Azure AD B2C, the Azure AD Graph and Azure AD PowerShell modules (both on the path to deprecation, with Microsoft Graph and Microsoft Graph PowerShell as the replacements), and the Microsoft Authentication Library. If a document uses "Azure AD" in one of those names, it is still correct.

EXPECT THE PORTAL TO MOVE

Microsoft changes menu names and layouts often, and the rename itself changed many of them. Every navigation path in this course is checked against Microsoft's documentation as of writing, but the portal may differ by the time you read it. When a menu is missing, search the portal's search box for the name, and trust the *concept* over the exact click path.

The Entra Family, and Where Entra ID Fits

"Microsoft Entra" is the name of a **family** of identity and network-access products. Microsoft Entra ID is the foundational one. The others are good to recognise, because they appear in the portal and in client conversations, but this course concentrates on Entra ID.

PRODUCT	WHAT IT IS FOR (IN MICROSOFT'S WORDS, SIMPLIFIED)	RELEVANCE TO THIS COURSE
Microsoft Entra ID	Cloud identity and access management: authentication, policy enforcement and protection for users, devices, apps and resources	The subject of this course
Microsoft Entra Workload ID	Identity and access management for workload identities: applications, services and containers	Closely related, because an application that signs in without a person is a workload identity
Microsoft Entra External ID	Secure access for guests, partners and customers	Relevant when the client's users are guests rather than employees
Microsoft Entra ID Protection	Detects and reports identity-based risks	Can cause sign-in blocks or extra prompts
Microsoft Entra ID Governance	Automates access requests, assignments and reviews	Mostly background
Microsoft Entra Domain Services	Managed domain services (group policy, LDAP, Kerberos/NTLM) for legacy apps in the cloud	Not used by modern sign-in integrations
Microsoft Entra Verified ID, Private Access, Internet Access, Agent ID	Verifiable credentials; secure access to private and internet resources; identities for AI agents	Not covered

Tenants

Everything in Entra ID lives inside a **tenant**: one organisation's own, separate instance of the service. Each tenant has:

- a **tenant ID**, a unique identifier (a GUID) that names it,
- an **initial domain name** of the form `something.onmicrosoft.com`, created with the tenant, and
- optionally one or more **custom domains**, the organisation's own names, such as `contoso.com`.

If an organisation subscribes to Microsoft 365, Azure or Dynamics CRM Online, it is already using Entra ID: Microsoft's documentation says that every such tenant is automatically a Microsoft Entra tenant. This is why the same set of work accounts opens Outlook, Teams and, once someone has set it up, your system.

ONE CLIENT = ONE TENANT (USUALLY)

Each client organisation normally has its own tenant, with its own users, policies and administrators. That's why "it's broken for Fabrikam but fine for Contoso" is such a useful clue: it points to something in Fabrikam's tenant or in the connection to it, rather than to your system as a whole. A few organisations have more than one tenant, so always confirm *which* tenant ID the failing connection points to.

What lives inside a tenant

OBJECT	WHAT IT IS	WHY A SUPPORT ENGINEER CARES
Users	The identities of people (employees, and guests invited in)	Who is trying to sign in, and are they enabled?
Groups	Collections of users used to grant access	Access is often given to a group, so removing someone from it can lock them out
Devices	Registered, joined or managed devices	Policies can require a compliant or joined device
Applications	The apps that use Entra for sign-in or that call Entra-protected APIs (app registrations and enterprise applications)	This is where "the integration" lives (Chapters 2 and 5)

OBJECT	WHAT IT IS	WHY A SUPPORT ENGINEER CARES
Roles	Built-in Entra roles that delegate administration	Decide what you can see and what the client's admin must do (Chapter 5)
Policies	Rules such as Conditional Access that decide when and how sign-in is allowed	A healthy integration can still be blocked by a policy (Chapter 6)
Logs	Records of sign-ins and of changes made in the tenant	The evidence for every investigation (Chapter 7)

Entra ID and the Other Systems It Touches

Entra ID and on-premises Active Directory

Many organisations have two directories: Windows Server **Active Directory Domain Services (AD DS)** in their own data centre, and Entra ID in the cloud. They are related but not the same thing. Microsoft describes Entra ID as an identity-as-a-service solution for apps across cloud and on-premises, where AD DS was built for managing on-premises infrastructure.

	WINDOWS SERVER ACTIVE DIRECTORY	MICROSOFT ENTRA ID
Where	On-premises servers	Microsoft's cloud
Typical app sign-in methods	LDAP, Windows-integrated authentication (Kerberos, NTLM)	OAuth 2.0, OpenID Connect, SAML and WS-Federation (WS-*)
SaaS apps	Not supported natively; needs a federation system such as AD FS	SaaS apps using OAuth 2.0, SAML or WS-* can use it directly
Devices	Domain join, Group Policy	Microsoft Entra join, Intune management, and checks through Conditional Access

Organisations that have both usually run **Microsoft Entra Connect**, which syncs identities from AD DS to Entra ID so people use one account in both places. For your purposes, remember that *modern sign-in integrations talk to Entra ID, not to AD DS*. If a client says "it's our Active Directory", that is a prompt to find out whether they mean the on-premises directory, Entra ID, or the sync between them.

Entra ID, Microsoft 365 and Azure

Microsoft 365 and Azure use Entra ID for their own sign-in; they are customers of the same service your integration uses. The licence level of the tenant (Entra ID Free, P1 or P2, or Microsoft Entra Suite) decides which features exist. Microsoft 365 E3 includes Microsoft Entra ID P1 and Microsoft 365 E5 includes Microsoft Entra ID P2. The licence matters to this course in one practical way: it limits some features and how long some logs are kept, which Chapter 7 covers precisely.

Entra ID and your system

Your system is neither Microsoft 365 nor Active Directory. It is an **application that relies on the client's Entra ID to sign people in**. In identity vocabulary it is a "relying party", although you'll more often hear "the application" or "the integration". Chapter 2 explains how it is registered with the tenant and what the different integration types are.

Whose Side Is It?

An integration has two ends, and troubleshooting starts by working out which end is at fault and who owns it. In broad terms:

ITEM	NORMALLY LIVES IN	WHO CAN CHANGE IT
The client's users, groups and sign-in policies	The client's tenant	The client's Entra administrators
The integration object for your system (registration or enterprise application)	The client's tenant (details depend on how it was set up; Chapter 2)	The client's administrators, or you if they gave you the right role
The secret, certificate or setting your system uses to talk to Entra	Both ends: it is created in the tenant and then stored in your system	Both sides must agree: a change in one place has to be copied to the other (Chapters 4 and 8)
Your system's own configuration and logs	Your platform	You
Microsoft Entra ID service itself	Microsoft	Microsoft (check the service health notices if everything is failing)

THE MOST COMMON FAILURE, IN ONE SENTENCE

The connection works because a credential in the client's tenant matches what your system holds; credentials have expiry dates; when one expires, or is replaced on one side only, the match fails and every user is refused at once. That is why this course keeps coming back to *expiry* (Chapter 4), *logs* (Chapter 7) and *both sides must change together* (Chapter 8).

Where You Manage Entra ID

You can see and manage Microsoft Entra in three web portals, per Microsoft's documentation:

- the **Microsoft Entra admin center**, <https://entra.microsoft.com>, the dedicated portal;
- the **Azure portal**, <https://portal.azure.com>, where Microsoft Entra ID appears as a service; and
- the **Microsoft 365 admin center**.

Administrators can also automate it through **Microsoft Graph** and the Microsoft Graph PowerShell module. As a support engineer you will mostly be reading the portal, often together with the client's administrator, so the first practical skill is finding which tenant you are looking at.

Finding the tenant ID

These are the steps Microsoft documents. You need to sign in as at least a Global Reader.

```
# Microsoft Entra admin center
```

1. Sign in at <https://entra.microsoft.com> (at least Global Reader)
2. Entra ID > Overview > Properties
3. Scroll to the "Tenant ID" box

```
# Azure portal
```

1. Sign in at <https://portal.azure.com>
2. Microsoft Entra ID > Properties
3. Scroll to the "Tenant ID" box

```
# Azure PowerShell (after Connect-AzAccount)
```

```
Get-AzTenant
```

```
# Azure CLI
```

```
az account tenant list
```

START EVERY TICKET WITH A "TENANT CARD"

Record these on any Entra-related ticket: the client's tenant ID, their initial `.onmicrosoft.com` domain and primary custom domain, the licence level if you know it, and the name of an administrator you can contact. A five-minute habit that saves hours when the person who set the integration up has left the company. Chapter 5 shows where else to look, and Chapter 7's logs ask for the tenant ID too. For the wider habit of keeping this kind of record, see the Documentation & Runbooks course.

Terms to Keep Handy

TERM	MEANING IN ONE LINE
Tenant	An organisation's own instance of Entra ID, with its own users, apps and policies
Tenant ID	The unique GUID that identifies a tenant
Identity provider (IdP)	The service that authenticates users and vouches for them to other systems
Authentication / authorization	Proving who you are / being allowed to do something
Single sign-on (SSO)	Signing in once and using the same identity across many applications
MFA	Multifactor authentication: more than one proof of identity
Conditional Access	Entra policies that decide when sign-in is allowed, blocked or needs extra proof
Microsoft Entra Connect	The tool that syncs identities from on-premises Active Directory to Entra ID
Microsoft Graph	The API through which Entra data and tasks can be accessed programmatically
Azure AD / AAD	The former name of Microsoft Entra ID

ONE WORD OF CAUTION ABOUT WHAT'S COMING

Several later chapters quote Microsoft's lifetimes and limits (how long a client secret can last, how long logs are kept, what a given error code means). Those numbers change, so each is verified against Microsoft's current documentation when the chapter is written, and every chapter points at the Microsoft Learn page it relied on. Always re-check a figure before telling a client it is a rule.

Hands-On Exercises

All three use fictional organisations. Do not paste real tenant IDs, secrets or client names into notes you share.

EXERCISE 1

Fabrikam Ltd's staff use "Sign in with Microsoft" on your product, "Acme Support Desk". Last week it began failing for everyone at Fabrikam, while Contoso, another client, is unaffected. Identify the identity provider, the tenant, the relying party, where authentication and authorization happen, and give three possible causes that fit the evidence.

 [View solution](#)

EXERCISE 2

An old internal runbook mentions ten things: "Azure AD tenant", "Azure Active Directory Premium P1", "AAD Connect", "Azure AD Conditional Access", "Azure AD B2C", "ADAL", "Azure AD PowerShell", "AD FS", "Azure AD Graph" and "DirSync". For each, say whether it was renamed, and if so give the current name; if not, say what it is now.

 [View solution](#)

EXERCISE 3

Write a "tenant card" template for support tickets. If you have read access to any Entra tenant (a work or training tenant), find its tenant ID in the portal, and note its initial domain. If not, fill the card in with the fictional details provided, and explain why each field helps in a sign-in investigation.

 [View solution](#)

CHAPTER 1 QUICK REFERENCE

- **Entra ID** = cloud identity provider: it authenticates, applies policies, and vouches for users to other systems
- **Authentication** proves who you are; **authorization** decides what you may do

- Renamed from **Azure AD** (announced 11 July 2023); URLs, APIs, PowerShell cmdlets and MSAL are unchanged
- Not renamed: Active Directory / AD FS, Azure AD B2C, the Azure AD Graph and Azure AD PowerShell modules, ADAL/MSAL
- Everything lives in a **tenant** (tenant ID + `.onmicrosoft.com` domain + custom domains); one client is normally one tenant
- Your system is an **application that relies on the client's tenant** to sign people in
- The decision is made in the **client's tenant**; a broken connection affects *all* that client's users at once
- Portals: `entra.microsoft.com`, `portal.azure.com`, Microsoft 365 admin center
- Tenant ID: Entra admin center > Entra ID > Overview > Properties (at least Global Reader), or `Get-AzTenant` / `az account tenant list`
- Start every ticket with a **tenant card**: tenant ID, domains, licence level, admin contact

Microsoft Entra ID: Integrations & Access Troubleshooting

Course 1 · Chapter 2 · How Applications Integrate

"The integration expired." The word *integration* hides a lot. It might mean a SAML single-sign-on setup, an app registration with a client secret, a background connection that calls an API, or an automatic user-provisioning job. The four behave differently, break differently, and are fixed by different people. This chapter names the objects that exist in Entra for any integration, then walks through the four common patterns so that you can say which one you're facing.

BY THE END OF THIS CHAPTER YOU CAN

explain the difference between an **app registration** and an **enterprise application**; say whether a given integration is single-tenant or multitenant and what that means for who holds the credentials; and classify an integration as **OpenID Connect sign-in**, **SAML single sign-on**, **API access** or **provisioning**, with the settings and expiry points of each.

Why an Application Needs an Identity

Entra ID can only vouch for people to an application it knows about. Microsoft's documentation puts it this way: to delegate identity and access management to Entra ID, an application must be **registered with a Microsoft Entra tenant**. Registering creates an identity configuration for the application, which lets it integrate with Entra ID. That configuration is what people loosely call "the integration".

Two Objects: the Registration and the Enterprise Application

Two things appear in the portal for the same application, and they are easy to confuse. Microsoft explains them as a blueprint and an instance of it:

	APPLICATION OBJECT (APP REGISTRATION)	SERVICE PRINCIPAL (ENTERPRISE APPLICATION)
What it is	The global definition of the application: the template from which common properties are taken	The local representation of the application inside one tenant

	APPLICATION OBJECT (APP REGISTRATION)	SERVICE PRINCIPAL (ENTERPRISE APPLICATION)
Where it lives	Only in the application's home tenant , the tenant where it was registered	In every tenant where the application is used
How many	Exactly one per application	One per tenant that uses it
What it describes	How tokens are issued for the application, the resources it may need, the actions it can take; you add secrets, certificates and scopes to it	What the application can actually do in that tenant , who can access it, and what resources it can access there
Where you find it (Entra admin center)	Entra ID > App registrations	Entra ID > Enterprise apps , which lists the service principals in a tenant, with permissions, user-consented permissions, who consented, and sign-in information

A registration also gives the application a **globally unique application (client) ID**. That is the value your system quotes when it asks Entra to sign someone in. Microsoft also notes that registering an application through the portal creates both the application object and a service principal in the home tenant automatically. (If you create the registration through Microsoft Graph, creating the service principal is a separate step.)

THREE IDS PEOPLE CONFUSE

Tenant ID names the organisation's tenant (Chapter 1). **Application (client) ID** names the application and is the same everywhere it is used. The objects themselves (the application object and each service principal) also have their own **object IDs**, which differ from each other and from the client ID. When someone pastes "the ID" into a ticket, ask which one it is, and from which page.

What the service principal types mean

The enterprise applications list can contain three kinds of service principal, per Microsoft:

- **Application:** the local instance of a registered application. This is the usual case for an integration.
- **Managed identity:** an identity for an Azure resource to connect to services without storing credentials. It has no associated application object.
- **Legacy:** represents an application created before app registrations existed, or through legacy experiences; it can only be used in the tenant where it was created.

DELETING AND DEACTIVATING: WHAT ACTUALLY HAPPENS

Per Microsoft: deleting an application object also deletes its home-tenant service principal, and **restoring the application object through the app registrations page does not restore the service principal**. If an application is only to be switched off temporarily, it can be *deactivated* instead, which stops new tokens being issued while keeping the objects for investigation or reactivation. So "someone deleted the integration and then put it back" does not necessarily mean the whole thing is back, a point Chapter 7's audit logs help to verify.

Single-Tenant or Multitenant?

When an application is registered, its **sign-in audience** is chosen. This decides who can use it, and it determines where the pieces end up:

AUDIENCE SETTING	TYPE	WHO CAN SIGN IN
Accounts in this directory only	Single tenant	All user and guest accounts in the home directory only. Microsoft's guidance also says to use this option when building an app registration for a third party that tells you to build your own registration for their app.
Accounts in any Microsoft Entra directory	Multitenant	Users and guests with a work or school account from any Entra tenant
Any Entra directory and personal Microsoft accounts	Multitenant	Work or school accounts plus personal Microsoft accounts (such as Outlook.com or Xbox)

A **single-tenant** application has one service principal, in its home tenant. A **multitenant** application also has a service principal created in each tenant where a user or administrator from that tenant has consented to its use. Microsoft's example is a company, Adatum, that builds an "HR app" and registers it in its own tenant; Contoso and Fabrikam each use it, and each ends up with their own service principal, with the permissions their own administrator consented to.

Vendor tenant (home tenant)

App registration (one)

client ID, secrets/certificates --->

redirect URIs, permissions

Client tenants

Fabrikam tenant

Enterprise app (service principal)

consented by Fabrikam's admin

Contoso tenant
----> Enterprise app (service principal)
consented by Contoso's admin

Three typical arrangements, and who holds the keys

How the integration for *your* system was set up decides who can fix an expired credential. These are the common shapes; check with the people who built it which one applies:

ARRANGEMENT	REGISTRATION LIVES IN	CREDENTIAL (SECRET OR CERTIFICATE) LIVES IN	WHO MUST RENEW IT
A. Your product is a multitenant app	Your own tenant	Your tenant's registration, shared by all clients	You. One renewal affects every client at once.
B. The client creates a single-tenant registration for your system	The client's tenant	The client's tenant; then copied into your system	The client's Entra administrator, then you update your side
C. A gallery or SAML enterprise app set up in the client's tenant	The client's tenant (as an enterprise application)	The SAML signing certificate in the client's enterprise application	The client's administrator, then the new certificate goes to you

WHY THIS MATTERS FOR THE SYMPTOMS YOU SEE

If everyone at *one* client is locked out and others are fine, arrangements B and C are the likelier suspects, because the credential belongs to that client's tenant. If *all* clients fail together, suspect arrangement A, or your own side. This is a guide to where to look first, not a rule.

The Four Integration Patterns

Microsoft describes several ways to connect an application to Entra for single sign-on: OpenID Connect and OAuth, SAML, password-based, linked, header-based and Integrated Windows Authentication, with the last three mainly for on-premises apps behind Application Proxy. For a cloud system like yours, four patterns matter, and between them they cover almost every "integration expired" ticket.

PATTERN	WHAT IT IS FOR	WHO CALLS WHOM	A PERSON PRESENT?
1. OpenID Connect / OAuth 2.0 sign-in	Users sign in to your system with their work account	Your system sends users to Entra, then calls Entra's token endpoint with its own credential	Yes
2. SAML single sign-on	Users sign in to your system, using the older, XML-based protocol	Entra sends your system a signed assertion; your system trusts Entra's signing certificate	Yes
3. API access (client credentials)	Your system reads or writes data in the client's tenant, such as through Microsoft Graph, as itself	Your system calls Entra's token endpoint with its own credential, then calls the API	No (background)
4. Provisioning (SCIM)	Entra automatically creates, updates and removes users (and groups) in your system	Entra calls your system , which exposes a SCIM 2.0 endpoint	No (background)

Pattern 1: OpenID Connect and OAuth 2.0 sign-in

Microsoft's guidance is to choose OpenID Connect and OAuth 2.0 if the application supports it. OpenID Connect extends OAuth 2.0 into an authentication protocol and issues an **ID token** to say who signed in. The application's configuration includes:

- its **application (client) ID**;
- the **authority**, a URL of the form `https://login.microsoftonline.com/{tenant}/v2.0`, where `{tenant}` is a tenant ID or domain name (for a single organisation), or `organizations` / `common` / `consumers` for wider audiences;
- one or more **redirect URIs**, the exact addresses in your system where Entra is allowed to send users back;
- the **scopes** it asks for (for sign-in, `openid`); and
- a **credential** (a client secret or a certificate) that your system uses to prove its own identity to Entra when it redeems the sign-in for tokens.

Each app registration has a public **OpenID Connect metadata document**, found under **Entra ID > App registrations > (your app) > Endpoints**, which lists the endpoints and signing keys that a library needs. The credential is the part that expires; the rest only breaks when somebody changes it.

Pattern 2: SAML single sign-on

SAML is an older but widely used protocol. Microsoft says to choose SAML for existing applications that don't use OpenID Connect or OAuth. It is configured on the **enterprise application** (Entra ID > **Enterprise apps** > **All applications** > **(the app)** > **Single sign-on** > **SAML**), by a Cloud Application Administrator, an Application Administrator, or the owner of the service principal. The settings that matter:

- **Identifier (Entity ID)**: a name for your system, typically a URL specific to the application;
- **Reply URL (Assertion Consumer Service URL)**: where Entra posts the signed assertion;
- **Sign on URL**: where users start from your side;
- the **SAML certificate** that signs the assertions, which your system must be given (the portal lets you download it); and
- the values your system needs from Entra: **Login URL**, **Microsoft Entra Identifier** and **Logout URL**.

Two facts from Microsoft's documentation matter for support. SAML SSO can only be configured on **single-tenant or gallery applications**; for a multitenant app, SAML is greyed out. And the signing certificate Entra creates when SAML is enabled is, **by default, valid for three years** (the lifetime can be customised), so the date has to be tracked and its renewal processed. Microsoft recommends naming an owner for it and a closely monitored mailing list for certificate notifications.

Pattern 3: API access with the application's own credentials

Sometimes your system acts by itself, with no user present: reading directory data, calling Microsoft Graph, syncing in the background. This is the OAuth 2.0 **client credentials** flow, which Microsoft describes as letting a confidential client use its own credentials, instead of impersonating a user, to call another service. The settings:

- the **client ID**, the **tenant** it operates against, and a **credential**: a client secret, a certificate, or a federated credential;
- the **application permissions** it has been given, which an *administrator* must grant (there is no user to consent), and which can be used only on that organisation's data; and
- the resource it asks for, with the scope written as the resource identifier followed by

`/.default`.

The reply is an **access token** with an `expires_in` value in seconds. When it expires, the application asks again; Microsoft notes that this flow never issues refresh tokens, because the client's own credentials can be used to get a new access token whenever needed. This is a good illustration of the difference between "the credential expired" and "a token expired", which is the subject of Chapter 4.

Pattern 4: Provisioning with SCIM, where the direction reverses

In provisioning, **Entra is the client and your system is the server**. The Microsoft Entra provisioning service connects to a **SCIM 2.0** endpoint provided by the application and uses it to create, update and remove users, and for some apps groups. The channel is encrypted with HTTPS using TLS 1.2. For this to work Entra needs **credentials to connect to your system's user management API**, which an administrator enters when setting up provisioning, and which can be tested from the portal.

THE CREDENTIAL HERE IS ONE YOUR SYSTEM ISSUED

In patterns 1 to 3, the credential is created in Entra and stored in your system. In provisioning it is the other way round: your system issues a credential, and the client's administrator pastes it into their tenant. If it expires or is rotated on your side, **the failing end is the client's provisioning job**. Microsoft says that if most or all calls to the target system keep failing, for example because of invalid admin credentials, the provisioning job goes into **quarantine**: its cycles gradually slow to once a day, it shows in the provisioning status and by email if notifications are configured, and if it stays in quarantine for over four weeks the job is **disabled**. Provisioning problems appear in the **provisioning logs**, not in the sign-in logs.

Which Pattern Are You Looking At?

Before you open any portal, a few questions usually settle it. Ask the person who reports the problem, or look at your own product's integration screen:

CLUE	POINTS TO
Users click "Sign in with Microsoft" and are sent to a Microsoft page and back	Pattern 1 (OpenID Connect) or 2 (SAML)
Your configuration asks for a client ID and a client secret or certificate	Pattern 1 or 3

CLUE	POINTS TO
Your configuration asks for an Entity ID , ACS/Reply URL or a certificate to upload or a metadata URL	Pattern 2 (SAML)
It's a scheduled or background job; no user involved; fails overnight	Pattern 3 (API access) or 4 (provisioning)
New or leaving staff aren't appearing or being removed in your system	Pattern 4 (provisioning)
The client's administrator presses a Test connection / validate button and it fails	Most often pattern 4 (provisioning), where the portal tests the connection to your endpoint; it can also be a check inside your own product
Everyone at one client fails at the same instant	A shared credential or certificate, or a change in that tenant (Chapter 4)

ABOUT THE "VALIDATE THE CONNECTION" ERROR

Several of the patterns have something that gets "tested" or "validated", and they mean different things. The key question is *where the check runs*: in the Entra portal, in your product, or somewhere else. If you can note the exact wording of the message, the screen it appears on, and who clicks the button, you can usually pin down both the pattern and which end is failing. When you have a real example (redacted of IDs and secrets), Chapters 7 and 8 work through it.

Ownership Summary

	PATTERN 1: OPENID CONNECT	PATTERN 2: SAML	PATTERN 3: API ACCESS	PATTERN 4: SCIM
Configured in Entra on	App registration (plus enterprise app)	Enterprise application	App registration, with admin-granted permissions	Enterprise application, provisioning tab
What can expire	Client secret or certificate	SAML signing certificate	Client secret or certificate	The credential your system issued; or the job quarantined

	PATTERN 1: OPENID CONNECT	PATTERN 2: SAML	PATTERN 3: API ACCESS	PATTERN 4: SCIM
Where to read the evidence	Sign-in logs	Sign-in logs	Sign-in logs (service principal sign-ins)	Provisioning logs
Who fixes the Entra end	Registration owner (you or the client's admin; see the arrangements above)	Client's administrator, or someone with an application role	As for pattern 1	Client's administrator (entering your new credential)

The "where to read the evidence" row is a preview; Chapter 7 covers the logs in detail and checks which tab and which role each needs.

Hands-On Exercises

All three use fictional organisations and invented values. Never use real secrets or tenant details in practice notes.

EXERCISE 1

Classify five fictional integrations by pattern (1 to 4) and by arrangement (A, B or C, or "not applicable"), and say who is likeliest to hold the credential: (a) a nightly job that reads user records from a client's tenant; (b) staff sign in through a "Sign in with Microsoft" button; the client created the registration; (c) a client sends a SAML metadata file and a certificate; (d) new starters appear automatically in your system; (e) your multitenant product's shared sign-in for all customers.

 [View solution](#)

EXERCISE 2

Below is a fictional configuration sheet. Label each value as a tenant ID, client ID, object ID, redirect URI, authority or secret, and decide which of them are safe to share on a ticket and which are not. Then explain why the same client ID can appear in two tenants but the object IDs differ.

 [View solution](#)

EXERCISE 3

Write a short "first questions" checklist (six to eight questions) that you would ask a client who reports "the Entra integration has stopped working", so that by the end you know the pattern, the tenant, the arrangement and the owner. Test it against two fictional reports.

 [View solution](#)

CHAPTER 2 QUICK REFERENCE

- **App registration** (application object): the global definition, one per app, in its home tenant; holds secrets, certificates, redirect URIs, permissions
- **Enterprise application** (service principal): the local instance in each tenant; what the app may do there and who may use it
- Portal: **Entra ID > App registrations** vs **Entra ID > Enterprise apps**
- Deleting an application object deletes its home-tenant service principal; restoring the object does **not** restore the service principal
- **Single tenant** = home tenant only; **multitenant** = a service principal is created in each tenant that consents
- SAML SSO can be configured only for single-tenant or gallery apps
- Four patterns: **OIDC sign-in, SAML SSO, API access (client credentials), SCIM provisioning**
- SAML signing certificate: by default valid for **three years** (customisable); name an owner and a monitored mailbox
- Client credentials flow: the app's own secret/certificate/federated credential; admin-granted application permissions; no refresh tokens
- SCIM provisioning: **Entra calls your system**; quarantine on repeated failures; disabled after four weeks in quarantine
- Always ask: which pattern, which tenant, whose credential, which end is failing

Microsoft Entra ID: Integrations & Access Troubleshooting

Course 1 · Chapter 3 · What Passes Between Entra and Your System

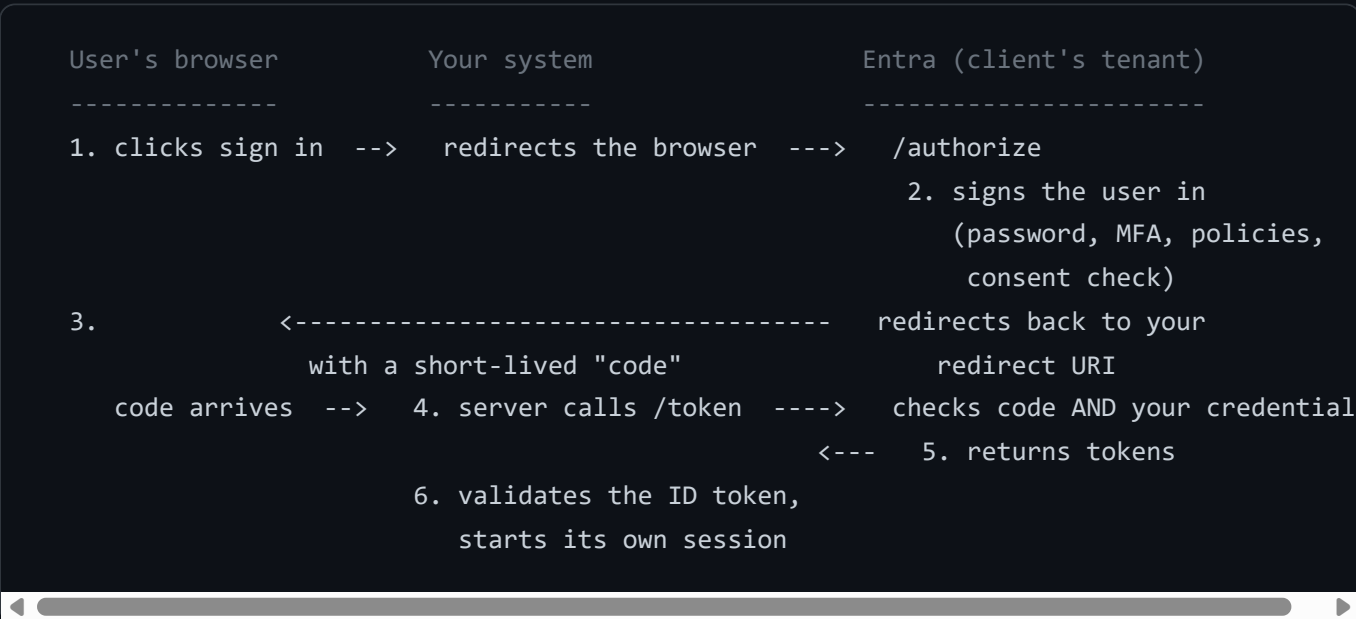
Chapter 2 told you what an integration *is*. This chapter follows what actually *travels* when it works: what your system sends to Entra, what Entra sends back, and what your system must check before it believes the answer. Once you know the steps, a failure stops being "the integration is broken" and becomes "step four failed, and step four is where the credential is used".

TWO KINDS OF TRAFFIC

In a sign-in, some messages go **through the user's browser** (your system redirects the browser to Entra and Entra redirects it back), and some go **directly from your server to Entra** (where your system proves its own identity with its credential). A failure in one has different symptoms and evidence from a failure in the other, which is why the next sections number the steps.

Pattern 1: An OpenID Connect Sign-in, Step by Step

This is the usual "Sign in with Microsoft" flow. Microsoft calls it the **OAuth 2.0 authorization code flow** combined with OpenID Connect. Here it is as one picture, then in detail:



Step 1: your system sends the browser to Entra

Your system redirects the user to the tenant's `/authorize` endpoint, with details in the address. For the authorization code flow, Microsoft documents these parameters:

PARAMETER	WHAT IT SAYS
<code>{tenant}</code> (in the path)	Who may sign in: a tenant ID or domain for one organisation, or <code>organizations</code> / <code>common</code> / <code>consumers</code>
<code>client_id</code>	The Application (client) ID of your registration
<code>response_type</code>	Must include <code>code</code>
<code>redirect_uri</code>	Where Entra should send the user back. It must exactly match one of the redirect URIs registered for the application
<code>scope</code>	The permissions requested; for sign-in, <code>openid</code> (plus <code>profile</code> / <code>email</code> if wanted)
<code>state</code>	A value your system gets back unchanged, used to guard against forged requests
<code>code_challenge</code> (PKCE)	A one-way fingerprint of a secret your system will reveal later; recommended for all application types
<code>nonce</code>	Needed when an ID token is requested directly from <code>/authorize</code> ; it returns inside the ID token to prevent replay

Step 2: Entra signs the user in

This happens entirely inside the client's tenant. Entra shows the sign-in page, checks credentials, applies the tenant's policies (Conditional Access, MFA; Chapter 6), and makes sure the user has consented to the permissions requested in `scope`. Some permissions are admin-restricted, and a user who can't consent to them gets an error saying so. Nothing in your system is involved in this step, and it is where "it works for most users but not one" problems usually live.

Step 3: Entra sends the browser back with a code

On success Entra redirects to your `redirect_uri` carrying a **code** (an authorization code) and your `state`. Microsoft notes that authorization codes are short-lived and typically expire after about one minute. On failure it sends back `error` and `error_description` instead. Your system should check that the returned `state` matches what it sent.

Step 4: your server redeems the code, and proves who it is

This is the one step where your system speaks to Entra **directly, as itself**. It POSTs to the tenant's `/token` endpoint with:

```

POST /<tenant-id>/oauth2/v2.0/token
Host: login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded

client_id=<application-client-id>
&scope=<scopes>
&code=<the code from step 3>
&redirect_uri=<the same redirect URI as step 1>
&grant_type=authorization_code
&code_verifier=<the PKCE secret>
&client_secret=<the application's secret>    <-- or a signed certificate assertion inst

```

For a web application the credential is required. It can be a **client secret**, or a **certificate credential** (the request then carries a `client_assertion`, a signed token, instead of `client_secret`). Microsoft says that for best security it recommends certificates. Public clients such as native apps and single-page apps must not use secrets or certificates here.

THIS IS WHERE AN EXPIRED CREDENTIAL BITES

Everything before step 4 happens in the browser and needs no secret. Step 4 is the first place Entra checks *your system's* credential. If the client secret has expired, been deleted or been replaced without updating your system, the user can sign in perfectly well at step 2 and still be turned away at step 4. Microsoft lists the matching error for this step as `invalid_client`: "*Client authentication failed. The client credentials aren't valid. To fix, the Application Administrator updates the credentials.*" The user sees only "sign-in failed" on your side.

Step 5: Entra returns tokens

The reply is a block of data (JSON) with these parts, per Microsoft:

FIELD	WHAT IT IS	PRESENT WHEN
<code>access_token</code>	The token to present to an API	Always (on success)
<code>token_type</code>	Always <code>Bearer</code>	Always
<code>expires_in</code>	How long the access token is valid, in seconds	Always
<code>scope</code>	The scopes the access token is valid for	Optional

FIELD	WHAT IT IS	PRESENT WHEN
refresh_token	A long-lived token used to get new access tokens	Only if the <code>offline_access</code> scope was requested
id_token	Proof of who signed in	Only if the <code>openid</code> scope was requested

Step 6: your system checks the ID token and starts a session

Microsoft says that web apps must validate ID tokens that reach them through the browser (the hybrid flow), and libraries validate them in every case. In outline, your system checks the token's **signature** against the public keys published in the tenant's OpenID metadata document, and then checks **claims** such as `iss` (who issued it and for which tenant), `aud` (that it was issued for *your* application), `exp` (not expired), and `nonce` (matches what was sent). Libraries do this for you. Entra rotates its signing keys periodically, so a system has to fetch the key set again rather than cache one forever; Microsoft suggests checking for key updates about every 24 hours.

The Tokens, Side by Side

	ID TOKEN	ACCESS TOKEN	REFRESH TOKEN
Purpose	Proof of authentication : who signed in	Permission for authorization : what the caller may do on an API	Gets new access tokens without asking the user again
For whom	Your application (<code>aud</code> is its client ID)	The API (<code>aud</code> is the API); your client treats it as opaque	Your client, presented only to Entra's token endpoint
Format	Signed JWT	Often a JWT, but for Microsoft's own APIs (such as Microsoft Graph) it may not be readable by you	Opaque
Lifetime	Short; the <code>exp</code> claim states it	Short and variable: a random value between 60 and 90 minutes by default, 75 on average; <code>expires_in</code> tells you	Long, with no fixed lifetime for web apps, but it can expire, be revoked or lack privileges; single-page apps get 24 hours
If it expires	The user signs in again (often silently)	Use the refresh token, or ask again	The user (or app) has to sign in again from the start

Remember the ID token and access token division from Chapter 1: *who are you* versus *what may you do*. Chapter 4 returns to these lifetimes, and the important distinction between a *credential* expiring (the client secret) and a *token* expiring.

The claims in an ID token

A **claim** is a statement the identity provider makes about the user or the token. In an ID token the most useful ones for support are:

CLAIM	MEANING (MICROSOFT'S DEFINITIONS, SIMPLIFIED)	WHY A SUPPORT ENGINEER CARES
iss	Issuer: who created the token. Also identifies the tenant for which the user was authenticated	A token from the wrong tenant is rejected
aud	Audience: the intended recipient. In an ID token it is your application's client ID, and a token that doesn't match must be rejected	Mismatch means the wrong app, or a wrong client ID in your settings
tid	The tenant the user signed in to	Confirms which client tenant this is
oid	The immutable object ID of the user's account; the same in every app, different in each tenant	A stable way to identify the user
sub	The subject: immutable, but different for each application	Also stable; specific to your app
iat, nbf, exp	When the authentication happened, the time before which the token isn't valid, and the time on or after which it can't be accepted	Clock problems and expiry show up here
nonce	Echoes the value your app sent	Mismatch means a replayed or mixed-up request
name, preferred_username, email	Human-readable details; need the <code>profile</code> or <code>email</code> scope; the values can change	Display only ; see below
roles, groups	Roles assigned to the user; group membership (with an "overage" fallback)	Used in some authorization decisions

A CLASSIC SUPPORT TRAP: MATCHING USERS BY EMAIL OR USERNAME

Microsoft warns that `preferred_username` and `email` are mutable and that the email isn't guaranteed to be correct; they must not be used for authorization decisions or to identify a user. The stable identifiers are `oid` (or `sub`), with `tid` for the tenant. If your system matched people by email and someone's name or address changed, or a new employee received a former colleague's address, you can see "this user can't sign in" or even "this user sees someone else's account". When one user is affected and everything else looks fine, check what your system uses to match them.

GROUP CLAIMS AND THE OVERAGE LIMIT

To keep tokens small, Entra limits how many groups it lists. Per Microsoft the limits are 150 for SAML tokens and 200 for JWTs: a user in more groups than that gets no group list in the token, only a pointer telling the application to ask Microsoft Graph. If a system grants access by group and a long-standing user suddenly loses access after joining more groups, this is one of the things to rule out.

TOKENS ARE CREDENTIALS: HANDLE THEM LIKE PASSWORDS

Microsoft says developers can decode JWTs on `jwt.ms` for validation and debugging. A real token, though, is a live credential until it expires: anyone holding it can use it. Don't paste real tokens into tickets, chats or online tools, and ask clients not to either. Record the *claim names and values that matter* (`tid`, `aud`, `exp`) rather than the token. Microsoft also says not to validate or read tokens for APIs you don't own, such as Graph.

Reading an Error From Entra

When a token request fails, Entra returns an error in a standard shape. This is Microsoft's own example (the values are samples):

```
{
  "error": "invalid_scope",
  "error_description": "AADSTS70011: The provided value for the input parameter 'scope' :
    Trace ID: 0000aaaa-11bb-cccc-dd22-eeeeee333333
    Correlation ID: aaaa0000-bb11-2222-33cc-444444dddddd
    Timestamp: 2016-01-09 02:02:12Z",
  "error_codes": [ 70011 ],
  "timestamp": "2016-01-09 02:02:12Z",
  "trace_id": "0000aaaa-11bb-cccc-dd22-eeeeee333333",
  "correlation_id": "aaaa0000-bb11-2222-33cc-444444dddddd"
}
```

PART	WHAT TO DO WITH IT
error	A short category (<code>invalid_client</code> , <code>invalid_grant</code> , ...). Tells you the kind of problem
error_description	The human-readable explanation, starting with an <code>AADSTS</code> code. Carries most of the useful information
error_codes	The number from the AADSTS code, on its own (70011 in the example)
timestamp	When it happened (compare with the sign-in logs; Chapter 7)
trace_id, correlation_id	Always copy these onto the ticket. They identify this request, so the client's administrator can find the matching entry in the sign-in logs

The error codes at the token step

ERROR	MICROSOFT'S DESCRIPTION	LIKELY MEANING FOR SUPPORT
invalid_client	Client authentication failed; the client credentials aren't valid	Secret or certificate expired, deleted, wrong or not updated (Chapters 4 and 8)
invalid_grant	The authorization code or PKCE code verifier is invalid or has expired	Code used too slowly or twice; a bug or a clock/session problem. Start sign-in again
unauthorized_client	The client isn't permitted to use this grant type; usually the application isn't registered or isn't added to the user's tenant	Application missing from the client's tenant, or deleted (Chapter 2)
invalid_scope	The scope requested by the app is invalid	A misconfigured scope or resource
consent_required	The request requires consent for a scope the client lacks permission to request	Permissions or consent changed (Chapter 6)
interaction_required	Another step, such as extra authentication, is needed	A policy now needs MFA or similar (Chapter 6)
temporarily_unavailable / server_error	Temporary conditions on the server	Retry; check Microsoft service health if widespread

IF YOU HAVE A REAL "VALIDATE THE CONNECTION" ERROR

Compare it to this structure: is there an `AADSTS` number, a trace ID and a correlation ID? Which `error` category does it sound like? That alone often tells you which step failed. Chapters 7 and 8 use your example, with identifiers redacted.

Pattern 2: What Passes in a SAML Sign-in

SAML follows the same shape with different messages. Your system (the "service provider") sends the browser to Entra with a sign-in request; Entra authenticates the user and sends the browser back to your **Reply URL** (the Assertion Consumer Service) with a **signed SAML token**. There is *no* separate server-to-server step and *no client secret*: the trust rests on the **signing certificate**. Your system has Entra's certificate (Chapter 2) and checks the signature on every assertion.

What the SAML token carries, according to Microsoft:

- the **subject**, also called the name identifier (`nameID`), which identifies the user; by default Entra puts the user's **username (user principal name)** there, and if the sign-in request includes a `NameIDPolicy` with a specific format, Entra honours that format;
- an **attribute statement** of claims; by default these include the user's email address, first name and last name;
- the **issuer** and **audience** (your Identifier / Entity ID), and the signature; and
- group claims, only if configured, subject to the 150-group limit for SAML.

Claims are configured under **Entra ID > Enterprise apps > All applications > (the app) > Single sign-on > Attributes & Claims > Edit**, as at least a Cloud Application Administrator. The classic SAML failure after a certificate renewal: the client renews or rotates the signing certificate in Entra, but your system still holds the old one, so every signature check fails for every user at once. The other classic one: the `nameID` is a username but your system expects an email address (or the other way round), so the user is authenticated yet can't be matched to an account.

Pattern 3: What Passes With API Access (Client Credentials)

This is the same step-4 conversation, without any user or browser. Your system asks the token endpoint for a token for itself:

```

POST /<tenant-id>/oauth2/v2.0/token
Content-Type: application/x-www-form-urlencoded

client_id=<application-client-id>
&scope=https://graph.microsoft.com/.default
&client_secret=<the application's secret>      <-- or client_assertion for a certificate
&grant_type=client_credentials

Reply:  { "token_type": "Bearer", "expires_in": 3599, "access_token": "..."}

```

The scope is the resource followed by `/.default`, meaning "all the application permissions an administrator has granted to this app for that resource". The reply carries **no refresh token** and no ID token; when the access token expires, the application simply asks again using its own credential. So if the credential has expired, *the next request fails and the job stops*. This is why background jobs fail at a particular moment, often overnight, with nobody signed in.

Pattern 4: What Passes in Provisioning (SCIM)

Here **Entra sends requests to your system**. The provisioning service connects to your SCIM 2.0 endpoint, using the credential that was entered in the client's tenant, and uses the SCIM user schema and REST calls to manage users (and groups where supported), over HTTPS with TLS 1.2. Which events cause which calls, according to Microsoft:

WHAT HAPPENS IN THE TENANT	WHAT ENTRA DOES TO YOUR SYSTEM
A user is assigned to the app (or comes into scope) and no match exists	Creates the user, using the attributes in the attribute mapping
A matching user already exists	Updates it; the "matching attribute" (for example <code>userPrincipalName</code> mapped to <code>userName</code>) decides which user is which
A user is unassigned, falls out of scope, is disabled or soft-deleted	By default, disables them: for SCIM apps a disable sets the <code>active</code> property to false
A user is hard-deleted (in Entra, 30 days after soft-delete)	Deletes the user, if the delete action is enabled

Entra runs an **initial cycle** over everything in scope and then repeated **incremental cycles**, remembering the ID your system gives each user. It records what it did, and what your system answered, in the **provisioning logs**. If the connection credential is rejected, most calls fail, the job goes into quarantine (Chapter 2), and your own server logs will show refused calls, not Entra sign-ins. Provisioning has no user, no browser and no tokens to look at: just calls to your system, and your system's answers.

Where Can Each Step Fail?

This table pulls the chapter together. Use it as a first guess; the evidence in Chapter 7 confirms it.

STEP	FAILS WHEN	TYPICAL SYMPTOM
1. Redirect to Entra	The redirect URI is not registered exactly as sent; the application isn't in the tenant	An error page from Microsoft before any sign-in; <code>unauthorized_client</code>
2. User signs in	Wrong credentials, MFA, a Conditional Access block, no consent, user not assigned	One user or group affected; a Microsoft error page after the sign-in
3. Redirect back	The user cancels or consent is refused; the returned <code>state</code> doesn't match what was sent	An <code>error</code> such as <code>access_denied</code> arrives at your redirect URI
4. Redeem the code (credential used)	Secret or certificate expired, removed, or changed on one side only	Everyone at that client fails together with <code>invalid_client</code> . If it fails now and then with <code>invalid_grant</code> , the code was late or reused
5. Tokens returned	Requested scopes or consent no longer valid	<code>invalid_scope</code> , <code>consent_required</code>
6. Your system validates	Wrong audience or issuer, bad signature, expired token, keys not refreshed, user can't be matched	Entra succeeded but your system still refuses; check your own logs
SAML: signature check	Signing certificate renewed in Entra but not updated in your system	Everyone at that client fails; a signature error in your logs

STEP	FAILS WHEN	TYPICAL SYMPTOM
API access: each request	Credential expired; permissions removed	A background job stops; no user is involved
Provisioning: Entra calls you	The credential you issued expired, was rotated, or your endpoint is down	Quarantine; users not created or removed

Hands-On Exercises

All three use fictional values. Never practise with real tokens, secrets or tenant details.

EXERCISE 1

A support engineer wrote down six events from a fictional "Sign in with Microsoft" attempt to Acme Support Desk by a user at Fabrikam, but not in order. Put them in order, name the step each belongs to, say whether it goes through the browser or directly between servers, and say at which event an expired client secret would show up.

1. Acme's server sends a request to the tenant's `/oauth2/v2.0/token` with the code, its client ID and its client secret.
2. The browser arrives at `https://acme.example/auth/callback?code=...&state=...`
3. Acme checks the ID token's signature, `aud`, `iss` and `exp`, then sets its own session cookie.
4. The browser is sent to `login.microsoftonline.com/<fabrikam-tenant>/oauth2/v2.0/authorize?client_id=...&redirect_uri=https://acme.example/auth/callback&scope=openid profile&state=...`
5. Entra replies with JSON containing `access_token`, `id_token` and `expires_in`.
6. The user types a password and approves an MFA prompt on a Microsoft page.

 [View solution](#)

EXERCISE 2

Here is the decoded payload of a fictional ID token, with what your system expects. Identify each claim, decide which claim to use to recognise the user, and find at least three reasons this token should be rejected for a Fabrikam sign-in to Acme Support Desk (there are four).

```
{
  "iss": "https://login.microsoftonline.com/33333333-3333-3333-3333-333333333333/v2",
  "aud": "99999999-9999-9999-9999-999999999999",
  "tid": "33333333-3333-3333-3333-333333333333",
  "oid": "55555555-5555-5555-5555-555555555555",
  "sub": "AbCdEf123",
  "name": "A. Student",
  "preferred_username": "a.student@fabrikam.example",
  "nonce": "xyz789",
  "exp": "2026-10-01 08:00 UTC"      (shown as a date for readability)
}
```

Your system expects:

```
Fabrikam's tenant ID ..... 11111111-1111-1111-1111-111111111111
Acme's client ID ..... 22222222-2222-2222-2222-222222222222
The nonce it sent ..... n-0001
The time now ..... 2026-10-01 10:30 UTC
```

 [View solution](#)

EXERCISE 3

Below are six fictional error responses, shortened to their `error` value and the first words of the description (`AADSTS<n>` stands for a number you would copy from the real message). For each, say which step failed, who must act (client administrator, your team or the user), and what you'd record from the message before doing anything else.

1. `invalid_client`: "AADSTS<n>: ... client credentials ..."
2. `invalid_grant`: "AADSTS<n>: ... authorization code ... expired ..."
3. `unauthorized_client`: "AADSTS<n>: ... application not found in the directory ..."
4. `invalid_scope`: "AADSTS<n>: ... the provided value for 'scope' is not valid ..."
5. `consent_required`: "AADSTS<n>: ... requires consent ..."
6. `interaction_required`: "AADSTS<n>: ... additional authentication required ..."

CHAPTER 3 QUICK REFERENCE

- OIDC sign-in in six steps: redirect to `/authorize`; user signs in; code returned; **server redeems code with its credential**; tokens returned; your system validates the ID token
- `redirect_uri` must **exactly match** a registered redirect URI; the authorization code lives about one minute
- The credential (secret, or certificate assertion) is first checked in **step 4**: an expired one gives `invalid_client` for everyone at once
- **ID token** = who signed in (for your app); **access token** = what the caller may do (for the API, opaque to the client); **refresh token** = renews access tokens
- Access token default lifetime: random 60 to 90 minutes (75 average); use `expires_in`; refresh tokens are long-lived but can expire or be revoked
- Identify users by `oid` (or `sub`) with `tid`, **never by email or preferred_username**
- Error shape: `error`, `error_description` (AADSTS code), `error_codes`, `timestamp`, `trace_id`, `correlation_id`: copy the IDs onto the ticket
- SAML: no secret, trust rests on the **signing certificate**; NameID defaults to the username (UPN); group limit 150 for SAML, 200 for JWT
- Client credentials: no user, no refresh token, scope is `resource/.default`; an expired credential stops the job at the next request
- SCIM: Entra calls your system; disables via `active` = false; failures appear in the **provisioning logs** and as quarantine
- Tokens are credentials: never put real ones on tickets or online decoders

Microsoft Entra ID: Integrations & Access Troubleshooting

Course 1 · Chapter 4 · Credentials and Expiry

"It expired" is the most common explanation for an integration that suddenly stops. But Entra has several different things that can expire, on very different timescales, and they are fixed in very different ways. A token that lasts an hour and a client secret that lasts two years are not the same kind of thing. This chapter sorts them out, with the lifetimes Microsoft documents, so you can say exactly *what* expired and *who* has to act.

THE ONE DISTINCTION TO REMEMBER

Tokens expire all the time, in minutes or hours, and are renewed *automatically* by software.

Credentials (a client secret, a certificate, a SAML signing certificate, a provisioning credential) expire on a *calendar date*, and are renewed only when *a person* creates a replacement and puts it in the right places. When an integration stops working for everyone at once, a credential is the suspect. A token expiring normally just causes a quiet renewal.

What Can Expire

ITEM	USED BY	LIFETIME (AS DOCUMENTED)	RENEWED BY
Authorization code	OpenID Connect sign-in (Chapter 3, step 3)	About 1 minute	Automatic: a new sign-in issues a new one
Access token	Calling an API	A random value between 60 and 90 minutes (75 on average) by default	Automatic: refresh token or a new request
Refresh token	Getting new access tokens	90 days by default; 24 hours for single-page apps and for email one-time-passcode flows	Replaced at every use; if it expires or is revoked, the user must sign in again
Client secret	Your system proving its identity to Entra (patterns 1 and 3)	Up to 24 months (Microsoft recommends under 12)	A person , who creates a new secret and updates your system
Certificate credential	The same, with a certificate instead of a secret	The certificate's own validity (and any tenant policy limit; see below)	A person , or automation

ITEM	USED BY	LIFETIME (AS DOCUMENTED)	RENEWED BY
Federated credential	A workload proving identity through another trusted identity provider	No secret to expire	Not applicable
SAML signing certificate	Entra signing assertions to your system (pattern 2)	By default 3 years ; the date can be chosen when creating one	A person in the client's tenant, then your system must be updated
Provisioning credential	Entra connecting to your SCIM endpoint (pattern 4)	Set by <i>your</i> system, so whatever you chose	You issue it; the client's administrator enters it

The sources are Microsoft Learn pages on access tokens, refresh tokens, authorization code flow, adding app credentials and managing federation certificates. Check them before quoting a number to a client; Microsoft does change defaults.

Client Secrets

A client secret is described by Microsoft as sometimes called an "application password": a string your system sends to prove who it is. It is created in the application's registration, under **App registrations > (your app) > Certificates & secrets > Client secrets > New client secret**, by choosing a description and an expiry (or a custom lifetime).

- **Lifetime:** limited to two years (24 months) or less. A custom lifetime longer than 24 months can't be set. Microsoft recommends under 12 months.
- **The value is shown only once.** Microsoft says the secret *Value* is never displayed again after you leave the creation page. If nobody copied it, it cannot be recovered; a new secret has to be created.
- **Microsoft discourages them in production.** The documentation says client secrets are less secure than certificates or federated credentials, and recommends a certificate before moving to production.

WHAT THIS MEANS WHEN A CLIENT SAYS "WE NEED A NEW SECRET"

Creating a new secret does *not* change anything in your system by itself, and an old secret that is still in your configuration keeps working until its own expiry date. Replacing a secret is two jobs: create the

new one (in the client's tenant, or yours, depending on the arrangement from Chapter 2) **and** enter its value in your system. Do them in that order, keep the old one until the new one is proved working, then remove the old one (Chapter 8).

Certificates on an App Registration

A certificate is the credential Microsoft recommends. Per the documentation, a public certificate is uploaded under **Certificates & secrets > Certificates > Upload certificate**, accepted as `.cer`, `.pem` or `.crt`, and the certificate's **thumbprint** is what the client application uses to refer to it. Your system holds the matching private key and signs a short assertion with it for each token request (Chapter 3). Two points for support:

- The certificate has its own validity dates. When they lapse, token requests fail in the same way an expired secret does.
- Whoever manages the private key manages the risk. For production Microsoft recommends a certificate from a well-known certificate authority, managed through a service like Azure Key Vault. See the HTTPS/TLS Fundamentals course for how certificates work in general.

SAML Signing Certificates

When SAML single sign-on is set up on an enterprise application, Entra generates a self-signed signing certificate that is, by default, **valid for three years**. This is the certificate your system uses to check every assertion (Chapter 3). A few rules from Microsoft's certificate tutorial matter in practice:

- **The expiry date can't be edited after saving.** To get a different date you create a *new* certificate (the portal offers any date between today and three years away), save it, give it to the application, then make it **active**. New certificates start as *Inactive*; **Make certificate active** promotes it and demotes the old one.
- **Entra emails before expiry:** 60, 30 and 7 days before the SAML certificate expires. By default the only recipient is *the administrator who added the application*; up to five addresses can be listed, and a distribution list is suggested if more people need it. The sender is `azure-noreply@microsoft.com`, which is worth allow-listing. If the addresses were set through Graph or PowerShell, Microsoft advises opening the SAML page in the portal to confirm notifications are registered.

- **Renewal without downtime:** create the new certificate with a date that overlaps the old one, give it to the application, then activate it. If the application accepts only one certificate at a time, choose a short maintenance window.
- **Role needed:** Privileged Role Administrator, Cloud Application Administrator or Application Administrator.

THE TWO-SIDED TRAP

Microsoft warns that if the certificate is renewed in Entra but not updated in the application, authentication on the application may fail. It also notes that if both an *expired* certificate and an *inactive valid* certificate exist, Entra automatically uses the valid one, and users might then experience an outage if the application hasn't been updated. Conversely, an application that *doesn't* check certificate expiry will keep accepting an expired certificate that still matches. Two things follow: renewing in the portal alone is never enough, and an integration that "didn't break" on the expiry date isn't necessarily healthy.

METADATA MAKES ROLLOVER EASIER

Microsoft's guidance for software vendors recommends reading the application's **federation metadata URL** regularly (at least every 24 hours), adding any newly published certificate as a secondary one, and switching to it when the customer activates it. If your product can do that, a certificate renewal in the client's tenant stops being a two-person emergency. If it can't, ask whether that is something your team could add: it removes the "both ends must change together" problem for SAML.

Tokens and Refresh Tokens

Access tokens expire after roughly an hour and the software quietly gets another, so a token expiring is not by itself a fault. Refresh tokens are different, because their end *does* surface to people. Microsoft documents their default lifetime as 90 days (24 hours for single-page apps and email one-time-passcode flows), and says they can be **revoked** by the service at any time before then. Your application is expected to handle revocation by sending the user to an interactive sign-in again.

Revocations that Microsoft lists include:

- the user revokes their own refresh tokens, or an administrator revokes all of a user's tokens (everything is revoked);

- an administrator resets the user's password in the Microsoft Entra admin center or the Microsoft 365 admin center (revokes tokens issued to confidential clients as well as most others);
- a password *change by the user* or a self-service reset, which revokes some kinds of token but, per Microsoft's table, **leaves a confidential client's token alive**; and
- a password simply *expiring*, which doesn't revoke tokens.

For support, this explains a particular pattern: an integration that acts for *one user* in the background (for example, reading their calendar) works for weeks and then stops after that person's password is reset by an administrator or their tokens are revoked. That is not an expired credential and renewing a secret will not fix it; the user has to authorise the integration again.

Rules That Shape These Lifetimes

A tenant can restrict how applications are configured, using **app management policies**, which Microsoft describes as controls over how app owners and administrators configure applications and service principals. They are found at **Entra ID > Enterprise apps > Application policies** and configured by a Security Administrator together with an Application or Cloud Application Administrator (or by a Global Administrator). Among the restrictions Microsoft lists:

RESTRICTION (PORTAL NAME)	EFFECT	WHY A SUPPORT ENGINEER CARES
Restrict max password lifetime	Caps how long a client secret may last	The client may be unable to create the lifetime you asked for; secrets may be short, so renewals are frequent
Restrict max certificate lifetime	Caps how long a certificate credential may last	Likewise for certificates (Microsoft's own example uses 180 days)
Block password addition	Prevents any new secret being added to applications	A client may <i>be unable to renew a secret at all</i> and must move to a certificate or federated credential
Block multitenant applications	Prevents apps being created or promoted as multitenant	Can change which of the arrangements in Chapter 2 is even possible

So "the client can't create a new secret" is not necessarily a permissions issue: a tenant policy may be forbidding it. Ask the administrator whether application policies are in force before

assuming anything else.

Finding Out What Expires, and When

Chapter 5 walks through the portal in detail, but the places to look follow directly from this chapter:

YOU WANT TO SEE	LOOK AT
Client secrets and certificates	Entra ID > App registrations > (the app) > Certificates & secrets, which lists each credential with its expiry
SAML signing certificate	Entra ID > Enterprise apps > All applications > (the app) > Single sign-on > SAML Certificates (edit icon): the status, expiry date and thumbprint of each certificate
Credentials expiring soon, tenant-wide	Entra ID > Overview > Recommendations > Renew expiring application credentials

That recommendation, per Microsoft, appears when an application registration has a credential **expiring within the next 30 days**. The roles that can view it include Reports Reader, Security Reader and Global Reader. Microsoft's documented fix is the sequence you will use in Chapter 8: *add a new credential, update the service to use it, check in the sign-in logs that the new credential is the one being used, then remove the old one*. Microsoft's own sample of that recommendation lists a required licence of Microsoft Entra Workload ID, so confirm what the client has before telling them it will be there.

Symptoms: Which Kind of Expiry Is It?

WHAT THE CLIENT REPORTS	MOST LIKELY	FIRST CHECK
"Everyone at one client stopped being able to sign in, from one moment"	An expired client secret or certificate (OIDC), or a changed or expired SAML signing certificate	Expiry dates on the registration or the SAML certificate (Chapter 5); the sign-in logs for the error (Chapter 7)
"A nightly job stopped, nobody is signed in"	An expired secret or certificate used for API access	Credential expiry; a 'credentials invalid' error at the token request

WHAT THE CLIENT REPORTS	MOST LIKELY	FIRST CHECK
"New staff haven't appeared in our system"	The provisioning credential that <i>you</i> issued has expired or changed, or the job is quarantined	The provisioning status and logs; your system's credential
"It worked after we renewed the certificate on Monday, but now everyone fails"	Entra's certificate was renewed but your system wasn't updated (or the reverse)	The SAML certificate status and thumbprint on both sides
"One person's background sync stopped after their password was reset"	Revoked refresh token for that user	Ask the user to authorise the integration again; this is not a credential renewal
"Users sign in fine but our system says 'invalid token'"	Not an expiry: validation (audience, issuer, signing keys) on your side	Chapter 3's step 6

WHAT YOU CAN TELL THE CLIENT

"Your sign-in to us goes through a connection that uses a credential in your Microsoft tenant. Those credentials have an expiry date, and when one expires, everyone is refused at the same time even though their accounts are fine. We need your Entra administrator to check the expiry date and create a replacement, and then we'll update our side to match." It is accurate, it doesn't blame anyone, and it tells them who has to do what.

Hands-On Exercises

All three use fictional data. "Today" in the exercises is **1 October 2026**.

EXERCISE 1

For each of the following, say whether it is a **credential** or a **token**, how long it typically lasts, and who or what renews it: (a) a client secret; (b) an access token; (c) a SAML signing certificate; (d) a refresh token for a normal web app; (e) an authorization code; (f) a certificate credential on an app registration; (g) a provisioning credential issued by your system.

 [View solution](#)

EXERCISE 2

Audit this fictional credential register as at 1 October 2026. Work out what has already expired, what must be renewed within 30 days, which SAML certificate notification emails have already been sent, and which entry cannot be right.

#	ITEM	TYPE	CREATED	EXPIRES
1	Acme sign-in (Fabrikam tenant)	Client secret	2025-01-15	2026-10-20
2	Acme SAML SSO (Contoso tenant)	SAML signing certificate	2023-11-25	2026-11-25
3	Directory sync job (Fabrikam tenant)	Client secret	2024-09-28	2026-09-28
4	User provisioning (Contoso tenant)	SCIM credential we issued	2026-03-01	2027-03-01
5	Reporting API job (Contoso tenant)	Certificate credential	2025-02-14	2027-02-14
6	Test app (Fabrikam tenant)	Client secret	2026-09-30	2028-12-01

 [View solution](#)

EXERCISE 3

Three fictional reports. For each, say whether the likely cause is an expired *credential*, a changed or mismatched certificate, or a revoked token; who must act; and what you'd check first. (a) "Fabrikam's nightly sync failed at 00:01 on 28 September and has failed every night since." (b) "On Monday Contoso's administrator renewed the SAML certificate. Since Monday afternoon nobody at Contoso can sign in." (c) "Our integration reads Dana's calendar. It stopped on Tuesday, the day IT reset Dana's password. Nobody else is affected."

 [View solution](#)

CHAPTER 4 QUICK REFERENCE

- **Tokens** expire in minutes or hours and renew automatically; **credentials** expire on a calendar date and need a person
- Authorization code about 1 minute; access token random 60 to 90 minutes (75 average); refresh token 90 days (24 hours for single-page apps and email OTP flows)
- **Client secret**: 24 months at most (Microsoft recommends under 12); the value is shown **once**; discouraged in production

- **Certificate credential:** recommended over secrets; upload `.cer` / `.pem` / `.crt`; refer to it by thumbprint
- **SAML signing certificate:** default 3 years; the date can't be edited after saving, so create a new certificate; new ones start Inactive until you "Make certificate active"
- SAML expiry emails: **60, 30 and 7 days** before; default recipient is the admin who added the app; up to five addresses
- Renewing in Entra alone is never enough: the application must be updated too; an app that doesn't check expiry may mask the problem
- Refresh tokens can be revoked early (admin password reset, user or admin revoking tokens); that needs a new user sign-in, not a new secret
- App management policies can cap lifetimes or block new secrets entirely: **Entra ID > Enterprise apps > Application policies**
- Where to look: Certificates & secrets (registration), SAML Certificates (enterprise app), Recommendations (credentials expiring within 30 days)
- Replace in order: create new, update your system, prove it works, then remove the old

Microsoft Entra ID: Integrations & Access Troubleshooting

Course 1 · Chapter 5 · Finding the Integration in the Entra Admin Center

You now know what an integration is made of, what travels through it, and what expires. This chapter puts you in front of the portal. Three questions come first, in this order: **Which tenant am I looking at? What am I allowed to see there?** And **where, exactly, is the integration?**

Most of this chapter's value is in avoiding wasted time: opening the wrong tenant, looking at the registration when the enterprise application is the problem, or trying to change something your role doesn't allow.

THE PORTAL CHANGES, THE CONCEPTS DON'T

Every menu path below comes from Microsoft's own documentation as of writing, and Microsoft renames and rearranges pages often (Chapter 1). If a menu isn't where this chapter says, use the portal's search box and the page names as clues, and treat the *what you are looking for* column as the lasting part.

Step 1: Get Into the Right Tenant

Entra pages show whichever tenant (directory) you are currently signed in to. Microsoft's guidance is that if you have access to more than one tenant, you use the **Settings** icon in the top menu of the admin center to switch to the one you want. A support engineer who works with many clients can easily end up in the wrong one, and every screen looks plausible. After switching:

1. Open **Entra ID > Overview > Properties** (Chapter 1) and compare the **tenant ID** with the one on your tenant card.
2. Note the tenant's domain name, so it is clear in any screenshot you take.

WHERE THE OBJECTS LIVE DEPENDS ON THE ARRANGEMENT

From Chapter 2: in arrangement A (your multitenant app) the *app registration* is in **your own tenant** and only the *enterprise application* is in the client's. In arrangements B and C both live in the **client's** tenant. So "which tenant?" is sometimes *two* tenants. Look at the registration in the tenant that owns it, and at the enterprise application in the tenant that is complaining.

Step 2: Know What You Are Allowed to See

A support engineer from a vendor is not an administrator of the client's tenant, and shouldn't be. There are three realistic situations, and Microsoft's documentation describes what each can see.

A. You sign in as a guest (a B2B guest user)

Guests have limited default permissions. Microsoft states that by default a guest can manage their own profile and retrieve some information about other users, groups and apps, but *cannot enumerate all users, groups or other directory objects*. In the table of default permissions, both default and restricted guests can **read properties of registered and enterprise applications** and **list permissions granted to applications**. That is useful: it is a fair amount of what you need to *look*. Two cautions:

- Guests can be added to administrator roles, which then grant full read and write permissions for that role. So "guest" tells you the starting point, not the limit.
- The tenant may restrict guests further, and a setting called **Restrict access to Microsoft Entra administration portal** can stop non-administrators loading a set of frequently visited admin pages. Microsoft says it is not a security measure and most pages are still reachable by a direct link, but it can make the portal look half empty.

B. You are given a role

Microsoft's "least privileged roles by task" page lists, for each job, the smallest role that can do it. The rows that matter here:

TASK	LEAST PRIVILEGED ROLE (MICROSOFT)	OTHER ROLES THAT CAN ALSO DO IT
Read all configuration of an enterprise application	The default user role	(any user, including a guest, subject to the tenant's restrictions)
Read audit and sign-in logs	Reports Reader	Application Administrator, Cloud Application Administrator and others in a longer list
Read provisioning logs	Reports Reader	Enterprise application owner, Application Administrator, Cloud Application Administrator and others

TASK	LEAST PRIVILEGED ROLE (MICROSOFT)	OTHER ROLES THAT CAN ALSO DO IT
Read recommendations (such as "Renew expiring application credentials")	Reports Reader	Security Reader, Global Reader and others
Update single sign-on, provisioning or properties of an enterprise application	Enterprise application owner	Cloud Application Administrator, Application Administrator
Consent to delegated permissions, or to application permissions not for Microsoft Graph	Cloud Application Administrator	Application Administrator
Consent to application permissions for Microsoft Graph	Privileged Role Administrator (on paper)	Microsoft adds that in practice this typically requires Global Administrator

A role that fits most diagnosis is **Reports Reader** for the logs and recommendations, together with the default ability to read application configuration. **Global Reader** is a broader read-only role, and Microsoft's own tenant-ID instructions use it as the minimum for viewing the Properties page. The principle in Microsoft's SSO planning page is to always use the role with the fewest permissions that does the job, and to remove roles that were only needed temporarily. Clients may also give you a role through Privileged Identity Management, which grants a role just-in-time for a limited period.

C. The client's administrator drives, and you watch

In practice this is the most common and the safest. The administrator opens the pages, you read the values over a screen share or from screenshots, and nothing is shared except what you need. Everything in this chapter works that way: you just need to know which page to ask them to open.

WHAT YOU CAN'T SEE IN ANY CASE

The **value** of a client secret. Microsoft says the value is shown only when the secret is created. Even the highest role sees only a secret's description, expiry date and similar details afterwards. If a secret's value has been lost, nobody can retrieve it; a new one must be made.

Step 3: Find the Application

There are two lists, one for each object from Chapter 2. Both are searchable by name; the **Application (client) ID** is the most reliable thing to search for, because names can be duplicated (Microsoft notes you can have several registrations with the same name).

YOU WANT	GO TO	IT LISTS
The registration (global definition, credentials)	Entra ID > App registrations	The application objects in <i>this</i> tenant, which is the app's home tenant
The enterprise application (local instance, sign-in behaviour)	Entra ID > Enterprise apps > All applications	The service principals in <i>this</i> tenant, with their permissions, user-consented permissions, who consented, and sign-in information

You can jump between the two. On the registration's Overview page there is a link, **Managed application in local directory**, to the corresponding service principal in the same tenant. If the client can't find the registration at all, that is informative: in arrangement A it is simply in your tenant, not theirs.

Step 4: Read the App Registration

This page answers "how is the application defined, and what does it hold?" The sections that matter, from Microsoft's documentation:

SECTION	WHAT YOU LEARN	WHAT TO COMPARE OR CHECK
Overview	The Application (client) ID , the key identifier; other identifiers and a link to the managed application. The Essentials area is also where the tenant ID and the supported account types usually appear, though layouts vary	The client ID against what is in your system's settings; the tenant against your tenant card
Authentication	Platforms and redirect URIs ; whether ID tokens are enabled (for implicit and hybrid flows)	Each redirect URI must <i>exactly</i> match what your system sends (Chapter 3, step 1), including trailing slashes
Certificates & secrets	Certificates, client secrets and federated credentials, each with an expiry	Expiry dates (Chapter 4); the certificate thumbprint against the one your system uses

SECTION	WHAT YOU LEARN	WHAT TO COMPARE OR CHECK
Token configuration	Optional claims added to tokens	Whether a claim your system expects is configured
API permissions	The permissions requested, and whether admin consent has been granted ("Granted for <tenant>")	Chapter 6
Endpoints	The OpenID Connect metadata document and the authorize/token endpoints for this app	The authority and tenant your system uses
Manifest	The application object as text, including settings not shown elsewhere	Rarely needed by support

CHANGING THE REGISTRATION CHANGES THE ENTERPRISE APPLICATION TOO, IN ONE PLACE ONLY

Microsoft says that changes to the application object are reflected in its service principal *in the home tenant only*, and that deleting the application object deletes the home-tenant service principal (Chapter 2). Never delete or "tidy" an application during an investigation. Deactivation exists for a reason.

Step 5: Read the Enterprise Application

This is the tenant-local face of the integration and the place where the *quietest* causes hide. Open **Properties** first. Microsoft documents these settings:

PROPERTY	WHAT IT DOES	WHY IT MATTERS
Enabled for users to sign in?	If No , <i>no</i> users can sign in, even if assigned; tokens aren't issued for the application, and service principals can't use application permissions to access it	A switch that stops everything. Someone turning it off "temporarily" looks exactly like an expired credential from outside
Assignment required?	If Yes , only users and applications assigned to the enterprise application can obtain a token; if No , any user (including invited external users) can sign in	Explains "works for some, not for others". Global Administrators can sign in regardless of this setting
Visible to users?	Whether the app appears in My Apps and the Microsoft 365 launcher	Affects launching from My Apps, not sign-in through your own page

PROPERTY	WHAT IT DOES	WHY IT MATTERS
Application ID	The unique identifier of the application in this directory	The client ID; Microsoft Support also asks for it
Object ID	The ID of the service principal ; differs from the application object's ID	Used for assigning users and for Graph and PowerShell operations on this tenant's instance
Homepage URL	The URL launched from My Apps	Can't be edited here; it is edited on the application object
Notes	Free text, up to 1,024 characters	Sometimes holds a useful history; worth reading

Then check the other sections, many of which later chapters expand:

SECTION	WHAT YOU LEARN	CHAPTER
Users and groups	Who is assigned to the app	This chapter
Single sign-on	The SAML configuration: Basic SAML Configuration (Identifier, Reply URL, Sign-on URL), Attributes & Claims, and the SAML certificates with status, expiry and thumbprint	2, 3, 4
Provisioning	The provisioning job, its credential, its status (including quarantine) and attribute mappings	2, 7
Permissions	Which permissions were granted, and by whom	6
Sign-in logs, Audit logs, Provisioning logs	What happened and when	7
Owners	Who can manage this particular application	This chapter

Owners: the person who knows

An **owner** of an enterprise application can manage its tenant-specific configuration (single sign-on, provisioning, user assignments) and add or remove other owners, and, unlike an Application Administrator, only for the applications they own. The same is true of owners of app registrations. The **Owners** page is often the quickest route to the person who set the integration up. Two caveats from Microsoft: if *Restrict access to Microsoft Entra administration portal* is on,

non-administrator owners can't use the admin center to manage their apps, and owners added through Graph or PowerShell (not the portal) can't manage some SAML settings such as attributes and claims or the certificate properties.

The Five-Minute Inspection

A repeatable order, so nothing is missed. Write the answers on the tenant card from Chapter 1.

1. **Right tenant?** Compare the tenant ID. Note the domain.
2. **Find both objects.** The registration (in whichever tenant owns it) and the enterprise application (in the client's). Search by client ID. Note both object IDs.
3. **Enterprise application Properties.** Is "Enabled for users to sign in" Yes? Is "Assignment required" Yes, and if so is the affected user assigned? Anything in Notes?
4. **Credentials.** On the registration, Certificates & secrets: what is there, and what expires when? For SAML, the certificate status and expiry. Today's date against each.
5. **Configuration against your system.** Client ID, redirect URIs (exactly), the SAML Identifier and Reply URL, certificate thumbprints: do the two sides agree?
6. **Permissions and consent.** Are the requested permissions granted? (Chapter 6.)
7. **Provisioning,** if relevant: its status, any quarantine message, when it last ran.
8. **Evidence.** Then the logs (Chapter 7), with the time of the failure and the correlation ID from the error (Chapter 3).

DON'T CHANGE ANYTHING YET

Inspection is read-only. Resist "fixing" something you notice on the way (an unassigned user, a stale redirect URI) until you know it's the cause. A change made too early can hide the evidence, and it can break something else that depended on the old setting. The audit logs (Chapter 7) record who changed what, including the changes made during your investigation.

What to Ask the Client's Administrator For

Give them a short, specific request, say why you need it, and keep it to what you need. A model structure, filled in with the pages from this chapter:

Hello <name>,

Our sign-in to <product> for your users is failing (since <time>, all users). To find the cause we'd like to check, in your Microsoft Entra admin center, how the connection to our system is set up. Could you either share your screen for ten minutes, or send us screenshots of the following? Please hide anything that looks like a secret; we never need one.

1. Entra ID > Overview > Properties: the Tenant ID.
2. Entra ID > Enterprise apps > <app name> > Properties: "Enabled for users to sign in?" and "Assignment required?".
3. Entra ID > App registrations > <app name> > Certificates & secrets: the list of secrets/certificates with their expiry dates (never the values).
4. Entra ID > Enterprise apps > <app name> > Sign-in logs: any entries from <time>, and the Correlation ID of one failure.

We only need read access. If it's easier, a "Reports Reader" role for our engineer for a day would also let us see the logs ourselves. Thank you.

WHAT YOU CAN TELL THE CLIENT

"We don't need to change anything in your tenant to find out what's wrong; we need to look at how the connection is set up, and at one log entry. The safest way is for your administrator to share their screen while we look together, and we will never ask for a password or a secret value." That addresses the fear most administrators have, which is being asked to hand over access.

Hands-On Exercises

All three use fictional organisations and values. Never put real secrets, tokens or tenant details into practice notes.

EXERCISE 1

Using the table of least privileged roles in this chapter, choose the smallest role (or "default user") for each task, and say whether you, as a vendor support engineer, would ask for it or ask the client's administrator to do it instead: (a) read the sign-in logs for a failing app; (b) see when a client secret expires; (c) create a new client secret; (d) make a new SAML signing certificate active; (e) read the provisioning logs; (f) grant admin consent for a Microsoft Graph application permission.

EXERCISE 2

Below is a fictional shared screenshot, written out as text, of Fabrikam's enterprise application and its registration, against what your system has configured. Today is 1 October 2026. Run the five-minute inspection and list every problem or oddity you find, ranked by how likely it is to explain "nobody at Fabrikam can sign in".

```
== Fabrikam tenant: Entra ID > Overview > Properties ==
Tenant ID ..... 11111111-1111-1111-1111-111111111111

== Enterprise apps > Acme Support Desk > Properties ==
Enabled for users to sign in? ... Yes
Assignment required? ..... Yes
Application ID ..... 22222222-2222-2222-2222-222222222222
Notes ..... "Set up by J. Smith (left company)"

== Enterprise apps > Acme Support Desk > Users and groups ==
Assigned: "Support Staff" (group, 14 members)      <-- affected users are in "Sales"

== App registrations > Acme Support Desk > Authentication ==
Redirect URIs ..... https://app.acme.example/auth/callback/

== App registrations > Acme Support Desk > Certificates & secrets ==
Client secret "prod 2025" ..... expires 2026-09-28
Client secret "test" ..... expires 2027-02-01

== Your system's configuration ==
Client ID ..... 22222222-2222-2222-2222-222222222222
Redirect URI sent ..... https://app.acme.example/auth/callback
Client secret in use ..... "prod 2025"
```

EXERCISE 3

Write the message you would send to Contoso's Entra administrator after their users report that SAML sign-in to your product stopped working at 09:15 today. It should explain the problem in a sentence,

ask for exactly the pages you need (using this chapter's names), say what you do and don't need, offer a screen share, and avoid asking for any secret. Then list three things you would check yourself first, before sending it.

 [View solution](#)

CHAPTER 5 QUICK REFERENCE

- First: **right tenant** (Settings icon to switch; check Entra ID > Overview > Properties against the tenant card)
- Where the objects live depends on arrangement: registration in the owner's tenant, enterprise application in each client's
- Guests can by default read properties of registered and enterprise applications and list granted permissions, but can't enumerate users or groups
- **Reports Reader** reads sign-in, audit and provisioning logs and recommendations; app configuration is readable by the default user role; Global Reader is broader read-only
- Graph application-permission consent is, in practice, usually a Global Administrator job
- You can never see a client secret's **value**; only its description and expiry
- **Entra ID > App registrations** = the definition and credentials; **Entra ID > Enterprise apps** = the tenant-local instance and its sign-in behaviour; "Managed application in local directory" links the two
- Check the enterprise application's **Enabled for users to sign in?** and **Assignment required?**: either can stop sign-in with a healthy integration
- Redirect URIs must match exactly (trailing slash included); compare client IDs and certificate thumbprints on both sides
- Owners can manage only their own applications; the Owners page often finds the person who knows
- Inspection is read-only; ask the client's administrator to drive, and never ask for a secret

Microsoft Entra ID: Integrations & Access Troubleshooting

Course 1 · Chapter 6 · Permissions and Consent

Chapters 4 and 5 were about credentials: the integration proving who it is. This chapter is about what happens *after* it proves who it is. An integration with a perfectly valid secret and certificate can still fail, because Entra asks three further questions before it issues a token: **what is this application allowed to do?** (permissions and consent), **who is allowed to use it?** (assignment), and **under what conditions?** (Conditional Access, MFA and guest rules). These are the quiet causes. Nothing has expired, and nothing in your system has changed. Someone in the client's tenant changed a rule.

GATE	THE QUESTION	WHO CONTROLS IT	TYPICAL SYMPTOM
1. Permissions and consent	May this <i>application</i> have this access?	The client's administrator (or, for low-impact permissions, the user)	A consent prompt, "Approval required", or an error saying consent is missing
2. Assignment	May this <i>user</i> use this application?	The client's administrator or the enterprise application's owner	"Works for some people, not others"; an error that the user isn't assigned to a role for the app
3. Conditions	Is <i>this sign-in</i> , from this place and device, acceptable?	The client's security team (Conditional Access, cross-tenant access settings)	"Blocked by Conditional Access"; MFA demands; guests only, or one location only

VERIFY AGAINST YOUR OWN TENANT

Every statement below comes from Microsoft's current documentation, but consent and Conditional Access behaviour depends heavily on the client's own settings, licences and policies. Treat each section as "what to check", and confirm the real setting on the client's tenant (with their administrator) before telling anyone it is the cause.

Part 1: Permissions — What the Application May Do

Back in Chapter 2 you met the two ways an application accesses data. Microsoft's permissions overview calls them **delegated access** (the application acts on behalf of a signed-in user) and

app-only access (the application acts as itself, with no user present). Each uses a different kind of permission, and the difference decides who must approve it.

	DELEGATED PERMISSIONS	APPLICATION PERMISSIONS
Also called	Scopes, OAuth2 permission scopes	App roles, app-only permissions
Access context	On behalf of a signed-in user	No user; the app's own identity (Chapter 3's client-credentials flow)
What the app can reach	Only what <i>both</i> the permission <i>and</i> the signed-in user can reach. Microsoft's example: an app with <code>Files.Read.All</code> delegated can read only the files that user can personally access	Everything the permission covers. An app with the Graph application permission <code>Files.Read.All</code> can read any file in the tenant
Who can consent	Users for their own data (where the tenant allows it); admins for all users	Only an administrator
How it is requested	Statically (a list on the app registration) or dynamically (individual permissions requested at sign-in)	Static only (the list on the app registration)
Recorded as (Microsoft Graph)	An <code>oAuth2PermissionGrant</code>	An <code>appRoleAssignment</code>

For your integration, the practical reading is this. If it signs *users* in and then calls an API for them (an OIDC integration), it uses **delegated** permissions, even if that is just the basic `openid` and `profile` sign-in scopes. If it runs in the background and calls an API for itself (the client-credentials pattern from Chapter 3, whose token request asks for `resource/.default`), it uses **application** permissions, which only an administrator can approve. SAML single sign-on and SCIM provisioning don't use this permission model for their core job; their access is governed by the enterprise application's configuration and the provisioning credential instead.

WHY "IT ASKS FOR TOO MUCH" MATTERS TO THE CLIENT

Microsoft's own warning is that granting tenant-wide admin consent is a sensitive operation, potentially giving the application's publisher access to significant portions of the organisation's data, such as role management, full access to all mailboxes or sites, or full user impersonation. A careful client administrator will ask what each permission is for. Know your product's permission list, and be able to justify each entry in a sentence.

Part 2: Consent — Who Said Yes

A permission listed on a registration is only a *request*. **Consent** is the recorded approval. Without a record of consent covering what the application asks for, Entra won't issue the token.

User consent

When a user signs in, Entra checks whether consent for the required permissions already exists. If there is no record of user or admin consent, the user is shown a consent prompt listing the permissions and the publisher. If they accept, the consent is recorded and they usually don't need to consent again. But user consent is possible *only* where the tenant allows it for that application and those permissions. If it's disabled, or the user isn't allowed to consent to what's requested, **the user is not prompted at all**, which can look to the user like a plain failure with no explanation.

The tenant's user-consent setting is one of three built-in options, according to Microsoft:

- **Disable user consent:** users can't grant permissions to applications. They keep signing in to applications they already consented to, or that an administrator consented to for them, but can't approve anything new.
- **Users can consent to applications from verified publishers or your organisation, but only for permissions you select:** only permissions classified as low impact, and only for verified-publisher or in-tenant applications.
- **Users can consent to all applications:** any permission that doesn't itself require admin consent.

A client may well have changed this setting as part of a security tidy-up. If it moved from the third option to the first, a *newly onboarded* user of your product, who needs consent for the first time, suddenly can't complete sign-in, while existing users carry on.

Admin consent

Some permissions can *only* be consented to by an administrator: application permissions, and many high-privilege delegated ones. Authentication requests prompt for admin consent if consent wasn't granted and one of those permissions is requested. Custom application scopes aren't treated as high-privilege. Administrators can also consent **on behalf of the whole organisation**, after which users aren't prompted. Microsoft's page on granting consent lists who can do it:

ROLE	MAY GRANT TENANT-WIDE CONSENT FOR
Privileged Role Administrator	Apps requesting any permission, for any API
Cloud Application Administrator, Application Administrator (and AI Administrator, per the current page)	Any permission for any API <i>except</i> Microsoft Graph app roles (application permissions)
A custom directory role with the permission to grant permissions to applications	The permissions that role covers

That table matches Chapter 5's note that Graph application permissions need a higher-privileged person than most client administrators expect. Plan for a request to travel upward.

Where to see consent in the portal

WHERE	YOU LEARN
Entra ID > Enterprise apps > All applications > <app> > Permissions (under Security)	The permissions granted in this tenant; the page where an authorised admin selects Grant admin consent . Chapter 5's note applies: the enterprise application exists in this tenant only if someone consented or it was added
Entra ID > App registrations > <app> > API permissions (under Manage)	The permissions the application <i>requests</i> , and whether consent has been granted. Only in the tenant that owns the registration (Chapter 5)

The admin consent URL

When the client's administrator hasn't granted consent, the easiest route is often a link they can open. Microsoft documents the tenant-wide admin consent URL format, which needs only the application's client ID:

```
// Tenant-wide admin consent URL (format documented by Microsoft)
https://login.microsoftonline.com/{organization}/adminconsent?client_id={client-id}

// {client-id}      = the application's client (application) ID
// {organization} = the client's tenant ID or a verified domain name of the tenant
//                (the value "organizations" consents in the home tenant of whoever signed up)
```

The administrator should still review the listed permissions before accepting. Never ask for consent to be granted blindly, and never send a link that you cannot explain. Your product may have its own "connect" page that builds this URL for the customer; check how yours does it before improvising one.

The admin consent workflow

If users aren't allowed to consent, the tenant may have an **admin consent workflow**. A user signing in sees an **Approval required** window, types a justification and selects **Request approval**; a **Request sent** message follows. Designated reviewers are notified, and the user is emailed when the request is approved, denied or blocked. Microsoft notes that if a user sends several requests, only the first is submitted. So "I clicked request approval three days ago and nothing happened" is a real, common support case: the answer is to find out *who the reviewers are* and whether anyone is watching their queue.

FOUR WAYS CONSENT GOES MISSING AFTER IT WORKED

1. **New permissions requested.** Microsoft says a user may be prompted again even after an administrator consents, for example if the application requests a permission the administrator hasn't granted. A product update that adds a scope can break sign-in for a tenant that consented to the old list.
2. **Consent revoked or removed.** The permissions overview lists "previously granted consent is revoked" as a cause of a new consent prompt.
3. **Granting consent again can replace earlier grants.** Microsoft warns that granting tenant-wide admin consent may revoke permissions already granted tenant-wide for that application (permissions users granted for themselves aren't affected). Re-consenting is not always harmless.
4. **The policy changed.** The tenant's user-consent setting was tightened (above).

Chapter 7's audit logs will show consent and permission changes with who made them and when.

Part 3: Assignment — Who May Use the Application

Chapter 5 introduced the **Assignment required?** property. This is its mechanism. Even after tenant-wide admin consent, *who can sign in* is a separate decision. Microsoft is explicit that granting tenant-wide admin consent allows all users to access the application *unless otherwise restricted*, and that the way to restrict it is to require user assignment and then assign users or groups.

- With **Assignment required = Yes**, only assigned users, groups and applications can obtain a token. Unassigned users are refused.
- Assign under **Entra ID > Enterprise apps > <app> > Users and groups > Add user/group**. If the application defines no roles, the role is **Default Access** (its ID is all zeros).
- Roles that can assign: Cloud Application Administrator, Application Administrator, User Administrator, or the owner of the service principal.
- **Group assignment** needs Microsoft Entra ID P1 or P2. Assigning a group gives access to the users in that group; it **does not cascade to nested groups**, and nested group membership is not currently supported.
- Microsoft adds that applications that require user assignment must have their permissions consented to by an administrator, even if the tenant's user-consent policy would otherwise allow users to consent.

THE NESTED-GROUP TRAP

A client says "everyone in Sales is in the group we assigned". But "Sales" was nested inside "All Staff", and "All Staff" is the group assigned. Because assignment doesn't cascade to nested groups, the Sales users are not assigned. The fix is to assign the Sales group directly, or add the users to the assigned group. The symptom is exactly "works for some, not others".

Part 4: Conditional Access and MFA — The Conditions

Conditional Access is Entra's policy engine. Microsoft describes policies at their simplest as if-then statements: *if* a user wants to access a resource, *then* they must complete an action. It gathers **signals** (user, group, IP location, device, application, sign-in risk) and makes a **decision**: block access, or grant access with requirements such as multifactor authentication, a compliant or hybrid-joined device, an approved client app, a password change or acceptance of terms of use. Two facts from Microsoft's overview matter for troubleshooting:

- Policies are enforced **after first-factor authentication** is completed. So a user can type the right password and then be stopped by a policy.
- It needs Microsoft Entra ID P1 licences (Microsoft 365 Business Premium also includes it). Risk-based policies need Entra ID P2. When licences expire, Microsoft says policies aren't automatically disabled or deleted, but can't be updated.

Conditional Access lives at **Entra ID > Conditional Access**, readable by someone with at least the Security Reader role. The **Policies** page lists them, including **report-only** policies (which log what would have happened without enforcing), and has a **What If** tool.

Two ways this hits an integration

	USER SIGN-IN (OIDC / SAML)	SERVICE PRINCIPAL SIGN-IN (CLIENT CREDENTIALS)
Policy targets	Users, groups, apps, locations, devices, risk	Workload identities: individual service principals that the policy lists directly
Possible outcomes	Block, or grant with controls (MFA, compliant device and so on)	Block only; no MFA is possible for a workload identity
Typical trigger	A new policy requires MFA or a compliant device for "all cloud apps"; the user is on a new network	A policy blocks token requests from outside known public IP ranges, or on service-principal risk
Licence note	Entra ID P1	Workload Identities Premium is required to create or change such policies; existing ones keep working but can't be modified
Coverage	Applies to the user's sign-in to the app	Single-tenant service principals registered in the client's tenant. Multitenant and third-party SaaS applications aren't covered , and managed identities aren't either
Where it shows	Sign-in logs > the Conditional Access tab of an event	Sign-in logs > Service principal sign-ins ; failure reason "Access has been blocked due to Conditional Access policies."

WHY YOUR ARRANGEMENT MATTERS AGAIN

From Chapter 2: if your product is a *multitenant* application (arrangement A), Microsoft says Conditional Access for workload identities does not cover it, so a client's background (client-credentials) token requests aren't blocked by that feature. If instead the client created the registration in their own tenant (arrangements B and C), that service principal *can* be targeted, for instance by an IP-range rule. The obvious cause of "server calls blocked from our data centre only" is then a location policy that doesn't list your fixed IP addresses. Confirm which arrangement applies before diagnosing.

MFA and sign-in prompts

A new policy that demands MFA is a common surprise. For interactive sign-in, the user is simply prompted. But if your integration uses a flow that *can't* show a prompt (a non-interactive background refresh, an embedded browser, a legacy protocol), the request fails with a message saying interaction or MFA is required (Chapter 3's `interaction_required` family; the matching AADSTS numbers are below). Microsoft lists "blocking sign-ins for users who try to use legacy authentication protocols" as a commonly applied policy, which can break older connectors silently.

Part 5: Guests and Cross-Tenant Settings

If the people who cannot sign in are **guests** (B2B users from another organisation, or your own support staff signing in to the client's tenant), look at the cross-tenant rules. Microsoft's overview describes **cross-tenant access settings** that control **inbound** access (users from external organisations reaching the client's resources) and **outbound** access (the client's users reaching others). Key facts:

- By default, B2B collaboration with other Entra organisations is enabled, and the default settings apply to all external organisations unless **organisation-specific settings** exist, which take precedence.
- Access settings can be scoped to specific users, groups and applications, and blocking inbound access for all external users means all applications are blocked too. Microsoft warns that changing defaults to block can break existing business-critical access.
- **Trust settings** decide whether the client's Conditional Access policies accept the MFA and device claims from a guest's home tenant. If MFA isn't trusted, a guest who already did MFA at home is asked again. If a policy requires a compliant device and the claim isn't trusted, the guest is blocked.
- Configuring these settings needs at least Security Administrator, and targeting specific users or apps requires a Microsoft Entra ID P1 licence on the tenant being configured.
- Changes are recorded in the **audit logs** under the category `CrossTenantAccessSettings` (Chapter 7).

Also remember the plainer cause: the guest account may simply not exist yet in the client's tenant, which produces its own errors (below).

Reading the Errors

Chapter 7 covers the sign-in logs and error codes in depth. For now, here are the codes most related to this chapter, with the meaning given in Microsoft's current error-code reference (the error name is the part before the dash). Treat the code as a pointer, then confirm in the sign-in log.

CODE	NAME AND MICROSOFT'S MEANING (SHORTENED)	POINTS TO
AADSTS65001	DelegationDoesNotExist: the user or administrator hasn't consented to use the application	Gate 1: consent missing or revoked
AADSTS90094	AdminConsentRequired: administrator consent is required	Gate 1: an admin must consent
AADSTS65004	UserDeclinedConsent: the user declined to consent to access the app	Gate 1: the user said no; retry and accept
AADSTS50105	EntitlementGrantsNotFound: the signed-in user isn't assigned to a role for the signed-in app; assign the user	Gate 2: assignment required, user not assigned
AADSTS53003	BlockedByConditionalAccess: access has been blocked by Conditional Access policies; the policy doesn't allow token issuance	Gate 3: find the policy that applied
AADSTS50076	UserStrongAuthClientAuthNRequired: because of a configuration change such as a Conditional Access policy, the user must use multifactor authentication	Gate 3: MFA now required; retry interactively
AADSTS50079	UserStrongAuthEnrollmentRequired: MFA required, and the user must register security info (or a federated user needs the MFA claim from their identity provider)	Gate 3: MFA registration or federation
AADSTS50020	UserUnauthorized: the account doesn't exist in this tenant and can't access the application; it must be added as an external user	Gate 3/guests: no guest account in the client's tenant
AADSTS50034	UserAccountNotFound: the account must be added to the directory; may also mean the app chose the wrong tenant	Wrong tenant, or user not in it
AADSTS500011	InvalidResourceServicePrincipalNotFound: the resource principal wasn't found in the tenant (not installed or consented, or request sent to the wrong tenant)	The enterprise application isn't there yet, or wrong tenant

CODE	NAME AND MICROSOFT'S MEANING (SHORTENED)	POINTS TO
AADSTS700016	UnauthorizedClient_DoesNotMatchRequest: the application wasn't found in the directory/tenant (not installed, not consented, or wrong identifier or tenant)	Wrong client ID or tenant, or app not present
AADSTS7000112	UnauthorizedClientApplicationDisabled: the application is disabled	Chapter 5's "Enabled for users to sign in?" set to No

Who Is Affected? A Triage Table

Before opening any logs, the *pattern of who fails* already points at a gate. This is the most useful one-minute habit in the chapter.

PATTERN	SUSPECT FIRST	CHECK
Every user, started at one moment	Credential expiry (Chapter 4), disabled app, tightened Conditional Access or revoked consent	Enterprise app Properties; audit logs around that time
Some users, others fine	Assignment (including nested groups), or a Conditional Access policy targeted at a group	Users and groups; the failed users' group membership; the CA tab of a failed event
Only new users	User consent disabled or tightened; the consent workflow with no reviewer	User consent settings; the admin consent workflow
Only guests or external staff	Guest account missing; cross-tenant inbound settings; untrusted MFA claim	The user's type in the client tenant; cross-tenant access settings
Only from one office or country	A location-based Conditional Access policy	The CA tab: which policy, and its named locations
Only background / API calls	Application permission consent missing; a workload-identity policy; secret expiry	Service principal sign-ins log; Permissions page
After a product update	New permission requested that the client hasn't consented to	The registration's API permissions vs the enterprise application's Permissions

WHAT YOU CAN TELL THE CLIENT

"Your connection to our system is configured correctly and its credentials are valid. What's stopping people is a rule in your Entra tenant: either approval hasn't been given for what our application needs, or the user isn't on the list of people allowed to use it, or a security policy is blocking this kind of sign-in. All of these are decisions your administrator controls. We'd like to look at three pages together: the application's Permissions, its Users and groups, and the log entry for one failed sign-in, which will name the exact rule." This keeps the conversation about *settings*, not blame, and it asks for a short screen share rather than any extra access.

DON'T ASK TO DISABLE A SECURITY POLICY

If Conditional Access is the cause, the correct request is to **adjust the policy for the integration** (for example, add the integration's fixed IP address as an allowed named location, or exclude the service principal where the client's security team agrees). Asking a client to switch off a policy altogether will, rightly, be refused, and shouldn't be your suggestion. Let the client's security team decide.

Hands-On Exercises

All three use fictional organisations and values. Never put real secrets, tokens or tenant details into practice notes.

EXERCISE 1

For each situation, say which gate it points to (permissions/consent, assignment or conditions), the most likely single cause, and the first page you would ask the client to open: (a) only newly hired Fabrikam staff can't sign in to your product, while older staff can; (b) after your product's version 4.2 release, every Contoso user is prompted for consent and the first screen says approval is required; (c) 40 users in the "Sales" group fail, but the "Support Staff" group works, and Sales sits nested inside "All Staff" (the assigned group); (d) your server's nightly token request has been failing since Tuesday, with a service-principal sign-in showing "Access has been blocked due to Conditional Access policies."; (e) your own support engineer, signed in as a guest, sees AADSTS50020.

 [View solution](#)

EXERCISE 2

Below is a fictional extract of what the Contoso administrator showed you on a screen share, plus the error from the user's browser. Decide what is wrong and what is *not* wrong, naming each gate. Then say

exactly what you would ask the administrator to do, and who in their organisation might have to approve it.

```
== Enterprise apps > Acme Reports > Properties ==
Enabled for users to sign in? ... Yes
Assignment required? ..... No

== Enterprise apps > Acme Reports > Permissions ==
Microsoft Graph  User.Read (Delegated)           Granted for Contoso
Microsoft Graph  Directory.Read.All (Application)  <-- listed as requested; not

== App registrations > Acme Reports > Certificates & secrets ==
Client secret "prod 2026" ..... expires 2027-06-30

== Error shown to the nightly sync job ==
AADSTS<n> ... (code removed; the description says the application has not been
granted the application permissions it requested)
```

 [View solution](#)

EXERCISE 3

Write the one-paragraph reply you would send to a non-technical Fabrikam IT manager who asks, "Our security team just rolled out a rule requiring multifactor authentication for all cloud apps and now the nightly job that syncs data into your product fails, but users can still log in. Is your product broken?" Explain, in plain language, why a rule about MFA can stop a background job, which kind of identity the job is, what two things might need to happen on the Fabrikam side, and what you do *not* want them to do. Then list the three facts you would check first to decide which of two likely situations you are in.

 [View solution](#)

CHAPTER 6 QUICK REFERENCE

- Three gates after valid credentials: **permissions and consent** (app), **assignment** (user), **conditions** (Conditional Access, MFA, guests)
- **Delegated** permissions (scopes) act for a signed-in user and reach only what both the permission and the user allow; **application** permissions (app roles) act with no user, need an administrator, and reach

everything the permission covers

- User consent works only where the tenant allows it; if not, users aren't prompted and the **admin consent workflow** ("Approval required") may be the route
- Tenant-wide consent roles: Privileged Role Administrator for anything; Cloud Application / Application Administrator for everything *except* Graph application permissions
- Consent can vanish or be needed again: a new permission requested, consent revoked, re-consent replacing earlier grants, or tightened policy
- Where: **Enterprise apps** > <app> > **Permissions** (granted here); **App registrations** > <app> > **API permissions** (requested)
- Admin consent URL: `https://login.microsoftonline.com/{organization}/adminconsent?client_id={client-id}`
- Assignment: **Assignment required = Yes** means only assigned users/groups; group assignment needs Entra ID P1/P2; **nested groups aren't covered**
- Conditional Access is enforced **after** first-factor sign-in; read it at **Entra ID** > **Conditional Access**; check **report-only** and the sign-in log's Conditional Access tab
- Workload-identity policies can only **block**, cover single-tenant service principals only (not multitenant apps), and need Workload Identities Premium to change
- Guests: check the account exists, cross-tenant inbound settings, and whether MFA/device claims are trusted
- Codes: 65001/90094/65004 consent; 50105 assignment; 53003/50076/50079 Conditional Access/MFA; 50020/50034 account or tenant; 500011/700016/7000112 app missing or disabled
- Pattern of *who* fails points at the gate; ask the client to adjust the rule, never to switch off security

Microsoft Entra ID: Integrations & Access Troubleshooting

Course 1 · Chapter 7 · Reading the Sign-in Logs

Every earlier chapter ended with "then check the logs". This chapter is that step. Entra records what it decided, and why, every time an application or user asks for a token. If you can find the right entry, the log usually names the cause directly, and that is far better evidence for a ticket (and for a client) than a guess. The skill is not reading every field. It is knowing **which log**, **which filter**, and **which five or six fields** to copy down.

VERIFIED AGAINST CURRENT MICROSOFT PAGES, BUT PORTALS CHANGE

The log names, filters and retention figures below come from Microsoft Learn pages dated 2025 to 2026. Retention and licensing are the figures most likely to change; check the retention page before you promise a client that "we can look back 90 days".

Four Logs, Not One

Entra keeps several activity logs. Choosing the right one is half the job. They are all under **Entra ID > Monitoring & health** in the admin center, and all entries are system generated and can't be changed or deleted.

LOG	IT ANSWERS	FOR YOUR INTEGRATION
Sign-in logs	Who or what asked for a token, for which app and resource, and was it granted? Four types (next table)	The main evidence for "can't sign in", expired secrets and blocks
Audit logs	What was <i>changed</i> in the tenant, by whom, and when (users, groups, applications, consent)	"Who rotated or removed the secret?" "When was consent revoked?"
Provisioning logs	What the provisioning service did for each user or group sent to a non-Microsoft service	SCIM problems (Chapter 2): create, update, delete, disable
Microsoft Graph activity logs	The Graph API requests an app made	Rarely needed by support; they exist only if the client routes them to storage or analytics

The four kinds of sign-in

The sign-in logs page has four tabs, one per sign-in type. The most common mistake is looking at the first tab (interactive) and concluding "no failure was recorded".

TAB	MEANING (MICROSOFT)	LOOK HERE WHEN
Interactive user sign-ins	A user provides an authentication factor: a password, an MFA response, a biometric, a QR code	A person was at the screen: OIDC or SAML sign-in to your product (Chapter 3, steps 1 and 2)
Non-interactive user sign-ins	A client acts on a user's behalf with no factor required: using a refresh token to get an access token, redeeming an authorization code, background SSO. Entries are grouped, and the IP address of a confidential client's refresh shows the original token issuance, not the actual source	Step 4 (code redemption) or silent refresh failed; "it signs in, then drops out an hour later"
Service principal sign-ins	No user: the app presents its own credential (a secret or certificate), as in the OAuth client-credentials flow. Entries are grouped when the service principal, status, IP address and resource all match	Background or API integrations; an expired secret or certificate on an app-only job (Chapter 4)
Managed identity sign-ins	Azure resources whose secrets Azure manages	Almost never, for a vendor integration

WHERE THE FAILURE LIVES, BY CHAPTER 3'S STEP

The browser redirect and user sign-in (steps 1 to 2) are recorded as **interactive**. Your server redeeming the code for tokens (step 4) is an app acting on behalf of a user, so look in **non-interactive** as well, and check **service principal** sign-ins when no user is involved at all. If nothing is where you expect, check all three before deciding the request never reached Entra.

Getting to the Logs

1. Sign in to the Microsoft Entra admin center as at least a **Reports Reader**, which Microsoft names as the least privileged role for accessing the activity logs.
2. Go to **Entra ID > Monitoring & health > Sign-in logs** (or Audit logs, or Provisioning logs).

3. Better for one application: open **Enterprise apps** > <app> and use its own Sign-in logs, Audit logs and Provisioning logs pages. The filter is already set to that app.

Microsoft lists licensing for each log: audit and sign-in logs are available on the Free edition, while **provisioning logs need Entra ID P1 or P2** (or the Suite). If a client's Provisioning logs page is missing, licensing is a likely reason. Also, any user can see *their own* recent sign-ins at mysignins.microsoft.com, which is a handy, permission-free way for an affected user to confirm a time and error without involving an administrator.

Retention: How Far Back Can You Look?

This matters more than anything else in the chapter, because "it started failing last Tuesday" is only useful if Tuesday's data still exists. From Microsoft's data-retention page:

REPORT	ENTRA ID FREE	ENTRA ID P1	ENTRA ID P2
Audit logs	7 days	30 days	30 days
Sign-ins	7 days	30 days	30 days
Risky sign-ins (security signal)	7 days	30 days	90 days
Microsoft Graph activity logs	Not available	Not retained unless integrated with storage or analytics tools	

- The retention page does not give a separate figure for provisioning logs in this table; **confirm provisioning-log retention in the client's tenant** before relying on it.
- Changes aren't retroactive: when a tenant upgrades from Free to a premium edition, only data still within the 7-day free period is available. Expired data can't be recovered unless it was archived.
- Longer retention comes from **exporting**: archive to an Azure storage account, send to a Log Analytics workspace (Azure Monitor), stream to an event hub or SIEM. Configuring that needs at least Security Administrator. Microsoft 365 E5 or similar licences can use Microsoft Purview Audit (Premium) for audit logs; the Microsoft 365 Unified Audit Log is a separate system.

THE CLOCK STARTS WHEN THE PROBLEM DOES

If a client first tells you about a failure on day 9 and they are on the Free edition, the first day of evidence is already gone. Ask for logs *the same day*, save what you see (screenshots, or the download button for JSON or CSV), and attach it to the ticket. Chapter 9 will make this a standing rule for the support team.

Finding the Entry

Filter the sign-in logs

Use **Add filters**. Microsoft's commonly used filters:

FILTER	USE IT FOR
Date	The window around the failure. Remember that date and time in the log are shown in the time zone of the person using the admin center, not of the user who tried to sign in
Application	Your product's enterprise application name, or search by its Application ID. Names can be duplicated, so prefer the ID
Status	Failure to see failures only (also Success, Interrupted)
User	The UPN of the affected person
Correlation ID	All the requests that are part of <i>one sign-in attempt</i> . Best for pinning one failure from a user's error message (Chapter 3)
Request ID	One sign-in request; Microsoft notes it corresponds to an issued token
Resource	The service the sign-in was for (for example Microsoft Graph)
IP address	Source of the request: useful for location-based policies (Chapter 6)
Conditional Access	Not applied, Success or Failure
Client app	Browser, mobile and desktop, or legacy authentication clients

The best route: use the ID from the error

Chapter 3 told you to record the *trace ID* and *correlation ID* from the error. Now they pay off: paste the **Correlation ID** into the filter and you get exactly the attempt the user saw. Microsoft adds a caution: the correlation ID value is based on parameters passed by a client, so Entra can't

guarantee its accuracy. If a correlation ID finds nothing, search by user, application and time instead.

Reading One Entry

Select a row to open the activity details. Microsoft frames every sign-in as **who** (the user), **how** (the client application) and **what** (the resource being accessed). Work through these tabs, and copy the fields in the table that follows.

TAB	WHAT IT SHOWS	USE IT TO
Basic info	The bulk of the details: identifiers, status, the sign-in error code and failure reason, application, resource, client app, authentication requirement	Find the error and its plain-English meaning
Location / Device info	IP address and approximate location; browser, operating system, and whether the device is compliant, managed or hybrid joined	Spot a location or device condition (note: IP-to-place mapping is best effort, and VPNs and mobile carriers distort it)
Authentication Details	The policies applied (such as Conditional Access or security defaults), the sequence of methods used, and whether each step succeeded	See where MFA happened or failed
Conditional Access	Every policy that could apply, with a result of Success, Failure, Not Applied or Disabled	Find the exact blocking policy (Chapter 6)
Report-only	What policies in report-only mode would have done	Predict the effect of a policy someone is about to turn on

The fields to copy into the ticket

FIELD	WHY YOU WANT IT
Date (with time zone)	Lines up with the client's report and with audit-log changes
Status, Sign-in error code, Failure reason	Microsoft: the failure reason describes the error, and the <i>Additional Details</i> often say how to fix it
Application and Application ID; Resource and Resource ID	Confirms the right integration, and which API was called

FIELD	WHY YOU WANT IT
User, User ID, User type	Member, guest or external (Chapter 6)
Authentication requirement; Client app; Client credential type	Whether MFA was required, whether the client was legacy, and which credential was presented
Correlation ID, Request ID	Microsoft's own guidance when unsure: gather both for further analysis or a support request
Home tenant and resource tenant; Cross-tenant access type	For guest and multi-tenant cases (Chapters 2 and 6)
Conditional Access result and policy names	Names the blocking rule

USE THE ERROR LOOKUP, NOT MEMORY

Microsoft provides an error code lookup at login.microsoftonline.com/error (enter the number to get a description and remediation) and an authentication and authorization error-code reference. Search the number there rather than relying on a table in a course, including the one below.

The AADSTS Codes You Will Meet

Chapter 6 listed the consent, assignment and Conditional Access codes. These are the ones connected with credentials, tokens and the wrong tenant or application, in Microsoft's current wording (error name before the dash, shortened). Use them as a starting point, then confirm with the log's *Failure reason*.

CODE	NAME AND MICROSOFT'S MEANING (SHORTENED)	POINTS TO
AADSTS7000222	InvalidClientSecretExpiredKeysProvided: the provided client secret keys are expired; create new keys, or consider certificate credentials	Expired client secret (Chapters 4 and 8)
AADSTS7000215	Invalid client secret is provided; "developer error", the app is signing in without the correct authentication parameters	Wrong or mistyped secret, or the secret <i>ID</i> used instead of its value (Chapter 4)
AADSTS7000218	The request body must contain <code>client_assertion</code> or <code>client_secret</code>	The app sent no credential at all

CODE	NAME AND MICROSOFT'S MEANING (SHORTENED)	POINTS TO
AADSTS700027	Client assertion failed signature validation	Wrong or mismatched certificate and key
AADSTS50012	AuthenticationFailed: one of a list of reasons, including that the signing certificate isn't valid or its thumbprint isn't authorized, or the assertion has an invalid signature	Certificate credential problems (expired, wrong thumbprint)
AADSTS700030	Invalid certificate: subject name in the certificate isn't authorized	Certificate subject mismatch
AADSTS700082	ExpiredOrRevokedGrantInactiveToken: the refresh token expired due to inactivity	A normal part of the token lifecycle (Chapter 4's refresh-token lifetimes)
AADSTS70008	ExpiredOrRevokedGrant: the refresh token has expired due to inactivity	Same family: ask the user to sign in again
AADSTS50173	The provided grant has expired because it was revoked; a fresh token is needed (for example the user's password was changed or reset)	Refresh-token revocation (Chapter 4)
AADSTS70043	BadTokenDueToSignInFrequency: the refresh token expired or is invalid due to sign-in frequency checks by Conditional Access	A session-lifetime policy (Chapter 6)
AADSTS50011	InvalidReplyTo: the reply address is missing, misconfigured, or doesn't match those configured for the app	Redirect URI mismatch (Chapters 3 and 5)
AADSTS700016	The application wasn't found in the directory or tenant (not installed, not consented, wrong identifier or tenant)	Wrong client ID or tenant (Chapter 6)
AADSTS7000112	UnauthorizedClientApplicationDisabled: the application is disabled	Chapter 5's "Enabled for users to sign in? = No"
AADSTS90002	InvalidTenantName: the tenant name wasn't found; check the tenant ID	Wrong or mistyped tenant

WHAT THIS TABLE DOES NOT SAY: "WHICH CREDENTIAL"

AADSTS7000222 tells you a secret has expired but not whether it is the one your system is using. A tenant often holds several secrets. Match the secret your system uses (Chapter 4's "two-sided" trap) against the dates in Certificates & secrets before you name the cause to the client.

Not every error number means the integration is broken

Microsoft lists some codes that appear in healthy tenants: **50058** (user authenticated but not yet signed in, which appears when a user didn't finish signing in, so the User field may even show a GUID); **90025** (an internal service hit its retry allowance, usually resolved automatically); **500121** (the user didn't complete the MFA prompt); **70046** (session expired or reauthentication check failed). A handful of these in a busy tenant are normal. The question to ask is whether the failures are *constant and all users* or *sudden and specific*.

The Audit Log: What Changed and Who Did It

Where the sign-in log says "what happened", the audit log answers "what was changed". It records changes to applications, groups, users and licences. You can filter by **Service**, **Category**, **Activity**, **Status**, **Target** (the object changed), **Initiated by** and a date range. Open an entry to see the **Correlation ID**, the actor, the target, and, where applicable, the old and new values of changed properties. Category **ApplicationManagement** is where integrations live. Activity names from Microsoft's current reference that matter here:

ACTIVITY NAME	WHAT IT CAN TELL YOU
Update application – Certificates and secrets management	Someone added, removed or changed a secret or certificate on the registration: a prime candidate for "who rotated it?"
Add service principal credentials / Remove service principal credentials	Credential changes on the enterprise application side
Create / Update / Delete Certificate	SAML signing certificate changes (Chapter 4)
Update service principal	Changes such as disabling sign-in ("Enabled for users to sign in?") or other properties
Add app role assignment to service principal / Remove app role assignment from service principal	Assignment and application-permission changes (Chapter 6)
Consent to application / Add delegated permission grant / Remove delegated permission grant / Restore consent	Consent granted, removed or restored (Chapter 6)
Add / Delete / Hard delete / Update application; Remove / Restore service principal	The object itself was added, changed, deleted or restored (Chapter 2's deletion caveats)

ACTIVITY NAME	WHAT IT CAN TELL YOU
Add / Remove owner to application / service principal	Ownership changes (Chapter 5)

The model is a timeline. Find the *first failed sign-in* in the sign-in log, then look at the audit log for the minutes, hours and days *before* it. A secret deleted at 17:02 and the first failure at 17:05 is a story. Cross-tenant access changes appear under the category `CrossTenantAccessSettings` (Chapter 6).

WHOSE AUDIT LOG?

The audit log you can see is the *client's* tenant's. Changes made in your own tenant (arrangement A's app registration) appear only in your logs. Chapter 5's rule applies again: be sure which tenant you are in.

The Provisioning Log (SCIM)

For SCIM provisioning (Chapter 2), the sign-in log is the wrong place. The **provisioning log** (Entra ID > Monitoring & health > Provisioning logs, or the enterprise application's own Provisioning logs page) shows each identity processed: the **Identity**, the **Action** (Create, Update, Delete, Disable, StagedDelete, Other), the source and target systems, and a **Status** of Success, Failure, Skipped or Warning. Opening an entry gives four tabs:

- **Steps:** the sequence for that object: import, match, scope check, evaluate, provision.
- **Troubleshooting & Recommendations:** the error code and reason, often with guidance.
- **Modified Properties:** old and new values.
- **Summary:** an overview with identifiers for the object in both systems.

Extra filters worth knowing: **Job ID**, **Cycle ID** and **Change ID**; Microsoft says the cycle and change IDs can be shared with product support to find the exact event. A "Skipped" status has several causes (for example, an object out of scope) and isn't necessarily a failure.

Exporting Evidence for a Ticket

1. **Select the entry, copy the fields** from the table above into the ticket (text, not just a screenshot, so it can be searched later).
2. **Download** the filtered list from the log page as **JSON or CSV**, which Microsoft describes as the manual, cheapest option for a long-term backup.
3. **Screenshots** must hide anything sensitive. The logs contain user names and IP addresses, so treat them as personal data and share only what the ticket needs.
4. For repeatable or longer-term needs, the client's administrator can send logs to **Log Analytics**, a storage account or an event hub (diagnostic settings, which needs Security Administrator).

EMPTY RESULTS ARE NOT ALWAYS "NOTHING HAPPENED"

Microsoft documents that queries can return no results, or HTTP 429 (Too Many Requests), because of throttling that depends on current system demand. Their workarounds: make the date range smaller and repeat, or stream logs to a Log Analytics workspace and query there. Before telling a client "there are no failures", try a narrower window and check the other sign-in tabs.

A Worked Reading

A fictional entry from a nightly sync job, copied the way you would copy it into a ticket:

```
== Sign-in logs > Service principal sign-ins ==
Date ..... 2026-10-02 02:00:04 (UTC+1)    <-- note the time zone
Status ..... Failure
Application ..... Acme Reports Sync
Application ID ..... 22222222-2222-2222-2222-222222222222
Resource ..... Microsoft Graph
Sign-in error code ..... 7000222
Failure reason ..... Invalid client secret provided. The provided client secret keys
Correlation ID ..... 33333333-3333-3333-3333-333333333333
Conditional Access ..... Not applied

== Audit logs, filtered Target = Acme Reports Sync, 2026-09-25 to 2026-10-02 ==
2026-09-30 17:02  Update application - Certificates and secrets management  Success
                  Initiated by: admin@fabrikam.example    Target: Acme Reports Sync
2026-09-30 17:04  Update service principal                                     Success
```

Reading it: **who** is the application itself (service principal); **how** is a client secret; **what** is Microsoft Graph. The error says the secret presented has expired, and the audit log shows credentials were changed two days earlier. That doesn't yet tell you whether the new secret was *installed on your side*; the evidence supports the Chapter 4 "two-sided" hypothesis, which Chapter 8 will test and fix.

WHAT YOU CAN TELL THE CLIENT

"We've found the entry for last night's failed run in your Microsoft Entra sign-in log. It says the secret the application presented was expired. Your audit log also shows secrets for this application were changed on 30 September. So the likely situation is that a new secret was created but our system is still using the old one. We would like to confirm that by checking which secret is shown as current in your tenant, with your administrator watching. Nothing is wrong with your users' accounts." Quote the log's own words and times; it keeps the conversation factual.

Hands-On Exercises

All three use fictional organisations and values. Never put real secrets, tokens or tenant details into practice notes.

EXERCISE 1

For each situation, say which log and which sign-in tab (if relevant) you would open first, one filter you would apply, and one field you would copy: (a) a user says "sign-in to your product fails with an error page showing a correlation ID"; (b) your nightly import job, which uses a client secret and no user, stopped on Tuesday; (c) a client's users can sign in but are asked to sign in again every hour, via your server's refresh; (d) a client reports that their directory-to-product user sync created some users but not others; (e) the client's administrator asks "who changed the secret on your app, and when?"

 [View solution](#)

EXERCISE 2

A Free-edition client tells you on 14 October that sign-ins failed "sometime in the first week of October, definitely before the 6th". Using the retention figures in this chapter, what is the earliest date you could still see in the sign-in and audit logs, what would change if they were on P1, and what would you ask them to do today? Then write the three-line instruction you would give so the next failure isn't lost.

EXERCISE 3

Interpret these fictional log extracts. For each, state the likely cause, the gate or chapter it belongs to, your confidence, and the one extra piece of evidence that would confirm it. Extract A: Service principal sign-in, Failure, error 7000215, Conditional Access Not applied. Extract B: Interactive sign-in, Failure, error 53003, Conditional Access tab: policy "Block outside UK" = Failure. Extract C: Non-interactive sign-in, Failure, error 700082, user last used the app 100 days ago. Extract D: Interactive sign-in, Failure, error 50011, redirect URI shown in the error ends without a trailing slash.

CHAPTER 7 QUICK REFERENCE

- Logs live under **Entra ID > Monitoring & health**: Sign-in, Audit, Provisioning; **Reports Reader** is the least privileged role; provisioning logs need P1/P2
- Four sign-in tabs: **interactive**, **non-interactive**, **service principal**, managed identity: check all three relevant ones before saying nothing was recorded
- **Retention: Free 7 days; P1 and P2 30 days** for audit and sign-in logs (risky sign-ins 7/30/90); not retroactive; longer needs export (storage, Log Analytics, event hub)
- Filter by **Correlation ID** (one sign-in attempt), Request ID, Application ID, User, Status = Failure, IP, Conditional Access
- Date and time show in the viewer's time zone, not the user's
- Copy: date and zone, status, error code, failure reason, app and resource IDs, user and type, authentication requirement, correlation and request IDs, Conditional Access result
- Look up codes at login.microsoftonline.com/error: 7000222 expired secret; 7000215 invalid secret; 50012 / 700027 / 700030 certificate; 700082 / 70008 / 50173 refresh token; 50011 reply URL
- Audit log: **ApplicationManagement**, "Update application – Certificates and secrets management", credentials, consent, role-assignment and service-principal activities
- Timeline method: first failure in the sign-in log, then the audit log for the period before it
- Provisioning log: Identity, Action, Status; tabs Steps, Troubleshooting & Recommendations, Modified Properties, Summary; cycle and change IDs for support
- Empty or throttled (HTTP 429) results: shorten the date range or use Log Analytics

- Treat log extracts as personal data; share only what the ticket needs

Microsoft Entra ID: Integrations & Access Troubleshooting

Course 1 · Chapter 8 · Diagnosing and Fixing an Expired Secret, Certificate or Token

This is the chapter the course was built for. A client's users can't reach your system, the integration's validation fails, and the suspicion is that something expired. You now have all the pieces: what the credentials are (Chapters 2 and 4), where to find them (Chapter 5), what else could be blocking sign-in (Chapter 6), and where the evidence lives (Chapter 7). Here they are joined into one repeatable procedure: **stop, preserve, decode, confirm, fix in the safe order, prove it, tell the client.**

THE WORKED EXAMPLE IS FICTIONAL

The real error message from your own "validate the connection" check was not available when this chapter was written, so every error in it is a made-up, redacted example built from the patterns Microsoft documents. The method does not depend on the exact text. Exercise 3 asks you to run it on your own real error (with tenant IDs, client IDs, secrets, names and domains removed), which is the best use of this chapter.

Step 0: Stop and Preserve

Before anyone changes anything:

1. **Save the evidence** (Chapter 7): the failed sign-in entry's fields, the correlation ID, the time and time zone, and a download of the filtered log. On the Free edition the sign-in and audit logs go back only seven days.
2. **Note the exact error** that your system or the client saw, as text, including every identifier you can find (redacted when it leaves your team).
3. **Don't "try a new secret" blindly.** A rushed new secret that is pasted wrongly creates a second problem on top of the first and muddies the logs.

Step 1: Decode the Message

Error text from Entra has a predictable shape (Chapter 3): a short *error* name, a longer *error_description* that begins with an `AADSTS` number, then a timestamp, a trace ID and a

correlation ID. Here is a fictional, redacted example of an app-only (client credentials) failure:

```
{
  "error": "invalid_client",
  "error_description": "AADSTS7000222: The provided client secret keys for app
    '22222222-2222-2222-2222-222222222222' are expired. Create new keys for your app,
    or consider using certificate credentials for added security.
    Trace ID: 44444444-4444-4444-4444-444444444444
    Correlation ID: 33333333-3333-3333-3333-333333333333
    Timestamp: 2026-10-02 01:00:04Z",
  "error_codes": [7000222]
}
```

PIECE	WHAT IT TELLS YOU
error: invalid_client	Chapter 3's step 4 failure: the client (your app) could not authenticate. The failure is about the <i>credential</i> , not the user
AADSTS7000222	Microsoft's name for it: InvalidClientSecretExpiredKeysProvided. The secret presented has expired
app '2222...'	The application (client) ID. Check it matches the registration you think it is (Chapter 5)
Trace ID and Correlation ID	Your handles for the sign-in log (Chapter 7)
Timestamp (UTC)	The moment of failure. Convert it to the time zone your log viewer uses

Different credentials fail with different numbers. Match the number to the right kind of expiry before touching anything:

WHAT YOU SEE	LIKELY MEANING	WHICH FIX
AADSTS7000222	Client secret has expired	Fix A or B (below)
AADSTS7000215	Invalid client secret: wrong value, secret <i>ID</i> used, removed secret	Fix A, after checking what your system stores
AADSTS50012, 700027, 700030	Certificate-credential problems (not valid, thumbprint not authorised, signature or subject mismatch)	Fix B

WHAT YOU SEE	LIKELY MEANING	WHICH FIX
SAML sign-in fails after the date the signing certificate expired; your app says the assertion signature is invalid or expired	SAML signing certificate expired, or renewed on one side only	Fix C
Provisioning goes to quarantine or reports authorization failures; Test Connection fails	The provisioning credential (secret token) your SCIM endpoint accepts has changed or expired	Fix D
AADSTS700082 / 70008 / 50173 / 70043	A <i>token</i> (refresh token) expired or was revoked. This is not a broken integration	Ask the user to sign in again; see Chapter 4

Step 2: Confirm the Cause in Three Places

Never act on the error alone. Three independent checks, each from an earlier chapter:

1. **The log (Chapter 7).** The service-principal sign-in entry for the failure: same error code, same application ID, right time.
2. **The portal (Chapter 5).** **Entra ID > App registrations > <app> > Certificates & secrets:** which credentials exist, and their expiry dates, against today's date. For SAML: **Enterprise apps > <app> > Single sign-on > SAML Certificates.**
3. **The audit log (Chapter 7).** Filter the application and look for "Update application – Certificates and secrets management" or certificate activities in the days before the failure. Did anyone rotate, delete or replace a credential?

The three together usually produce one of four stories:

STORY	EVIDENCE	MEANING
1. It simply expired	Log: 7000222. Portal: the secret your system uses shows an expiry before today. No recent audit entries	Nobody renewed it in time. The classic case
2. Renewed on one side only	Portal: a newer secret exists. Audit: credential added or changed shortly before the failure. Log: still 7000222 or 7000215	Chapter 4's two-sided trap: Entra has a new secret; your system still sends the old one
3. Deleted or replaced	Audit: removal of a credential. Log: 7000215. Portal: secret list lacks the one you send	Someone cleaned up, or a pipeline replaced it

STORY	EVIDENCE	MEANING
4. Not the credential at all	Log shows a different code (consent, assignment, Conditional Access, disabled app), or the credentials are all valid	Back to Chapters 5 and 6. Do not rotate a healthy secret

WHOSE TENANT IS IT, AGAIN?

The same symptom is fixed in different places depending on the arrangement from Chapter 2. In **arrangement A** (your multitenant app) the credentials belong to *your* registration in *your* tenant: you rotate them, the client changes nothing, and a client-side "expired secret" is impossible. In **arrangements B and C** the registration is in the **client's** tenant: *their* administrator creates the new credential, and *you* (or they) install the value in your system. The fix below describes both, and is clearest about who does each part.

Step 3: Fix in the Safe Order

The principle is the same for every credential type, and it comes straight from Microsoft's own renewal guidance: **add the new credential alongside the old one, update the application, prove the new one works, and only then remove the old one.** Four verbs: *create, install, prove, remove*. Reversing "prove" and "remove" is the commonest self-inflicted outage.

Fix A: Client secret (create, install, prove, remove)

1. **Create.** The app registration's owner or an Application Administrator opens **Entra ID > App registrations > <app> > Certificates & secrets > Client secrets > New client secret**. They add a description (a good one: the system and the date, such as "AcmeReports prod 2026-10"), choose an expiry, and select **Add**. Microsoft limits the lifetime to 24 months or less and recommends less than 12.
2. **Copy the Value, not the Secret ID.** Microsoft is explicit that the secret **Value** is shown only on this page and never again after leaving it. If it's lost, create another. Pasting the *Secret ID* in place of the value is a well-known cause of 7000215.
3. **Transfer it safely.** The value goes straight into your product's credential field (or your secrets store) by the person who holds it. Never email it, paste it into a ticket or chat, or screenshot it. If a handover is unavoidable, use a secure channel your organisation approves.
4. **Install.** Update the credential in your system (your product's connection settings, or the secrets store it reads), then restart or reload anything that caches it.

5. **Prove.** Trigger the connection (a validation, a job run or a test sign-in) and open the *service-principal sign-in* log: it should now show **Success**. Microsoft's guidance is to use the sign-in logs to validate that the **Key ID** of the credential matches the one just added; check the sign-in details for the credential identifier, and if it doesn't show in the tenant you are using, a clean Success after the change is the practical proof.
6. **Remove.** Only after proving it (and after any other system that shares the registration has been updated), delete the old credential. Wait for a safe window, such as a full business cycle for overnight jobs.

Fix B: Certificate credential

1. **Create/obtain** a new certificate and key pair. Microsoft recommends a certificate over a client secret for production, and a certificate from a well-known certificate authority managed in a vault such as Azure Key Vault, rather than a self-signed one (self-signed is acceptable for testing).
2. **Upload** the public part under **Certificates & secrets > Certificates > Upload certificate** (file types `.cer`, `.pem`, `.crt`), then select **Add**. The private key never goes into Entra.
3. **Record the thumbprint** Entra shows, and compare it with the certificate your system will sign with. A thumbprint mismatch is the commonest certificate error (50012).
4. **Install** the new certificate and private key in your system, **prove** a successful service-principal sign-in, then **remove** the old certificate.

Fix C: SAML signing certificate

A SAML signing certificate is created by Entra (Chapter 4: three-year default; the date can't be edited after saving, so you create another). Because your system must trust whichever certificate Entra signs with, **both sides must change**, and the order matters. From Microsoft's certificate-management tutorial, which needs at least Cloud Application Administrator or Application Administrator:

1. **Create a new certificate:** Enterprise apps > <app> > Single sign-on > SAML Certificates > Edit (pencil) > **New Certificate**. Choose an expiry (default is three years from today, and you can pick any date up to three years), then **Save**. It appears with status **Inactive**.
2. **Download** it in the format your application expects (Base64 .cer, raw, PEM or federation metadata XML, as your product's setup guide says).
3. **Upload it to your system** as a trusted signing certificate. If your system supports *more than one* certificate, add it while the old one still works.

4. **Make certificate active** (ellipsis on the new row). The new certificate becomes **Active** and the old one **Inactive**. From now on Entra signs assertions with the new key.
5. **Prove:** sign in to your product with a test user. Then remove the old certificate from your system once confirmed.

WHAT MICROSOFT SAYS ABOUT EXPIRED CERTIFICATES

- If an *expired* certificate exists and you generate a new one, the new certificate is considered for signing tokens *even though it's not yet active*, and the expired one is no longer used.
- If both an expired and an inactive valid certificate exist on the application, Entra uses the valid certificate automatically, and users might experience an outage if your application hasn't been given the new one.
- If an application doesn't validate the certificate's expiry and the certificate matches on both sides, the app may keep working even though the certificate has expired. That is accidental protection; don't rely on it.
- If the application can handle only **one** certificate at a time, schedule a short downtime window for the swap between "Make certificate active" and uploading to your system.

For a better long-term design, Microsoft's ISV guidance is that your application should read the per-tenant federation metadata URL regularly (at least every 24 hours), accept additional signing certificates, and promote the new one when it's activated. That's the way to stop SAML certificate expiry from needing any human action. Chapter 9 returns to this.

Fix D: Provisioning (SCIM) credential

A SCIM connection has a **Tenant URL** (your SCIM endpoint) and a way of authenticating to it. Microsoft documents that a request carries an OAuth 2.0 bearer token. Options include a **long-lived bearer token** entered in the optional **Secret Token** field (supported for existing and non-gallery apps, not for new apps), an OAuth 2.0 client-credentials grant, or leaving the Secret Token field blank so that Entra sends its own Entra-issued token. Which one applies depends on how your product set it up.

1. **Create/issue** the new credential on *your* side (a new bearer token in your product, or a new client secret for the client-credentials option).
2. **Install** it in the client's tenant: Enterprise apps > <app> > Provisioning, where the client's administrator pastes the new value into **Secret Token** (or the client ID and secret fields).

3. **Prove:** select **Test Connection**. Microsoft says it queries the SCIM endpoint for a user that doesn't exist using a random GUID, and expects **HTTP 200 OK with an empty SCIM ListResponse**; a failure shows error information. Then confirm in the *provisioning log* (Chapter 7) that new cycles succeed.
4. **If the job was in quarantine**, check the provisioning status after the fix and use the provisioning log to confirm that normal operation has resumed (Chapter 2 described quarantine; confirm the current restart behaviour in Microsoft's provisioning pages for the client's setup).

Whichever way your product handles the credentials, "Test Connection" is a useful word to know: it is probably what the client calls "validate the connection". Confirm with the client's administrator which button or page they mean.

Step 4: Prove It Properly

CHECK	PASS LOOKS LIKE
Sign-in log (service principal, or interactive for SAML/OIDC)	A fresh entry after the change: Status = Success, the same application ID, and no error code
Credential identity	The Key ID (or thumbprint) matches the new credential, where the log shows it
Your system	The validation or job passes; real data flows; an affected user can reach the system
Provisioning (if relevant)	Test Connection passes and the latest cycle in the provisioning log has Success entries
Second attempt later	The next scheduled run or the next day's sign-ins also succeed (some caches only fail on the second use)

Only then remove the old credential, and record the fix (below).

If It Still Fails

NEW SYMPTOM	LIKELY REASON	CHECK
7000215 straight after the change	Secret ID pasted instead of value; stray space or line break; the secret was typed in the wrong environment	Re-enter carefully; compare the first characters of the value with the portal's display only if the client's policy allows. Create another secret rather than guessing
Still 7000222 after the change	Your system still uses the old value (cached, a different config file, or a second instance)	Restart/reload the process; check every place the credential is stored; look at which secret the log says is in use
Works for one client, not another	Per-tenant credentials (arrangements B and C) were updated for one tenant only	Check the tenant card (Chapter 1) for each client
A different error appears	The credential was the first problem; consent, assignment or Conditional Access is the next one (Chapter 6)	Read the new code; do not assume the fix failed
SAML sign-in fails after "Make certificate active"	Your system doesn't yet trust the new certificate	Upload it (Fix C, step 3); if urgent and the old certificate is not yet expired, ask the client's administrator whether to make the old one active again. Do not skip this question if the old certificate has expired
No change at all and no new log entry	The request never reaches Entra	Network, DNS or proxy on your side; Chapter 7's "check all tabs, narrower window"

While You Work: What to Tell the Client

WHAT YOU CAN TELL THE CLIENT (AT EACH STAGE)

Diagnosis: "The credential our integration uses to prove its identity to your Microsoft Entra tenant expired on 28 September. That is why your users could not sign in; nothing is wrong with their accounts or with your tenant."

Plan: "We'd like your administrator to create a replacement alongside the old one, so that nothing is removed until we've confirmed the new one works. That takes about ten minutes, plus a short test."

Done: "A new credential is in place and sign-in is working again as of 10:42. We will remove the old

one after a safe period. To make sure it doesn't happen again, we'd like to agree an expiry reminder with you." (Chapter 9)

ASK FOR THE SECRET VALUE ONLY THROUGH YOUR PRODUCT

Never ask a client to send you a secret in an email, ticket or chat message, and never agree to receive one that way. If your product's settings page accepts the value, the client's administrator enters it there. If your support team ever *does* receive a secret by accident, treat it as exposed: ask for a new one to be created and the exposed one deleted.

Record the Fix

A short, searchable ticket note makes the next occurrence (and Chapter 9's register) far easier:

```
Client / tenant ..... Fabrikam (tenant ID 1111...)      -- tenant card
Integration ..... Acme Reports Sync (client ID 2222...)
Arrangement / pattern ... B (client's registration) / client credentials
Symptom ..... Nightly job failed from 2026-10-02 01:00 UTC
Evidence ..... SP sign-in: AADSTS7000222; corr ID 3333...; audit: no changes
Cause ..... Client secret "prod 2025" expired 2026-09-28
Fix ..... New secret "AcmeReports prod 2026-10" (expires 2027-09), insta-
           old secret removed 2026-10-09 after 7 days of clean runs
Proof ..... SP sign-in Success 2026-10-02 10:42; job ran 2026-10-03
Prevention ..... Expiry reminder added for 2027-08-15 (owner: ...)
```

Hands-On Exercises

All three use fictional organisations and values. Never put real secrets, tokens or tenant details into practice notes.

EXERCISE 1

A colleague proposes this plan for an expired client secret: (1) delete the expired secret in Entra to avoid confusion; (2) create a new one; (3) email the Secret ID to you; (4) paste it into the product; (5) if it works, close the ticket. Find every problem with the plan, say which of the chapter's principles each breaks, and rewrite it as a correct plan using the verbs create, install, prove, remove.

 [View solution](#)

EXERCISE 2

Contoso's SAML signing certificate for your product expired on 28 September; their users can't sign in. In the portal there is also a second certificate, status Inactive, expiring in 2029, created in August. Your product's SAML settings can hold only one signing certificate at a time. Write the fix as numbered steps with who does each, decide whether to schedule a window and why, say what you would check before and after, and say what you would recommend for the longer term.

 [View solution](#)

EXERCISE 3

Run the full method on a real error: take the actual error from your own "validate the connection" check (redact tenant IDs, client IDs, secrets, names and domains first) and complete the decode table, the three-place confirmation and the fix-record note from this chapter. If you don't have one to hand, use this fictional one instead: `invalid_client`, `AADSTS7000215`: Invalid client secret provided, occurring two hours after the client's administrator created a new secret. Decide which of the four stories applies and write the fix and the message to the client.

 [View solution](#)

CHAPTER 8 QUICK REFERENCE

- Order: **stop, preserve, decode, confirm, fix, prove, tell**; save logs first (Free keeps 7 days)
- Decode: error name, `AADSTS` number, app ID, trace ID, correlation ID, timestamp; 7000222 = expired secret, 7000215 = invalid secret, 50012/700027/700030 = certificate, 700082/70008/50173/70043 = token (not a broken integration)
- Confirm in three places: sign-in log, Certificates & secrets (or SAML Certificates), audit log; four stories: expired, renewed on one side, deleted, not the credential
- Arrangement A: you rotate in your own tenant; B/C: the client's administrator creates and you install
- Safe order for every credential: **create, install, prove, remove**; never remove first
- Client secret: New client secret; copy the **Value** (shown once), not the Secret ID; max 24 months, under 12 recommended; never send by email

- Certificate credential: upload .cer/.pem/.crt; record the thumbprint; Microsoft recommends certificates over secrets
- SAML certificate: New Certificate (Inactive) → download → upload to your system → **Make certificate active** → test; plan a window if your system takes one certificate
- SCIM: Secret Token / OAuth credentials; **Test Connection** expects HTTP 200 with an empty ListResponse; then check the provisioning log
- Prove with a new Success entry in the right sign-in tab (and the Key ID or thumbprint where shown); then remove the old credential after a safe period
- Record the fix: symptom, evidence, cause, fix, proof, prevention date, owner

Microsoft Entra ID: Integrations & Access Troubleshooting

Course 1 · Chapter 9 · Preventing It Happening Again

A credential that expires is the most predictable incident there is. Unlike a disk failure or a bad release, it comes with a date printed on it. When one catches a team out, the date was not the problem: **nobody owned it, nobody was reminded, or the reminder went to someone who had left.** Prevention is therefore not a clever technology. It is a small, boring system: a list of every expiry, an owner for each, warnings that reach a real person, and a rehearsed renewal. This chapter builds that system out of what Entra offers and what your own product can add.

WHAT THIS CHAPTER IS NOT

Scripting and alerting are covered at overview level only. Microsoft changes the exact reports, preview features and licensing often (several relevant items below are marked preview), so the chapter names what exists and what each is good for, and tells you to verify the current details before you promise a client something specific.

Why These Incidents Recur

ROOT CAUSE	WHAT IT LOOKS LIKE	SYSTEMIC FIX
No one has the list	The expiry date exists only in the portal; no one looked until it failed	A register (below)
Reminders go to the wrong place	Entra's emails go to whoever created the app: someone who left, or an unread mailbox	Shared or distribution-list recipients; verify them
No one owns renewal	The client assumes the vendor will renew; the vendor assumes the client will	An agreed owner on each side, written down
Renewal is risky	Past renewals broke things, so people delay them	A rehearsed create-install-prove-remove runbook (Chapter 8)
Two-sided mismatch	Renewed in Entra but not in the product, or the reverse	One checklist covers both sides; product shows what it is using

ROOT CAUSE	WHAT IT LOOKS LIKE	SYSTEMIC FIX
Short surprise lifetime	A client's policy caps lifetimes lower than anyone expected	Ask about their application management policies up front (Chapter 4)

Who Can Prevent What

Your reach depends on the arrangement (Chapter 2). Prevention you control completely is different from prevention you can only *request*.

ARRANGEMENT	CREDENTIALS OWNED BY	YOU CAN DO DIRECTLY	YOU MUST ASK THE CLIENT FOR
A: your multitenant app	You (registration in your tenant)	Everything: register, owners, reminders, rotation, longer-term credential design	Mainly consent, assignment and Conditional Access, not credentials (Chapter 6)
B: client creates the registration	The client (their tenant)	Your product's side: where the credential is stored, warnings, overlap support	Expiry dates, owners, notification recipients, rotation scheduling
C: SAML or SCIM to your product	Shared: Entra signs (client side), your product trusts or accepts (your side)	Metadata auto-rollover, multiple-certificate support, SCIM token lifecycle	Notification emails, who activates new certificates and when

What Entra Tells You (and Whom)

SIGNAL	WHAT IT COVERS	WHO SEES IT AND CAVEATS (MICROSOFT)
SAML certificate expiry emails	Sent 60, 30 and 7 days before a SAML signing certificate expires; sender azure-noreply@microsoft.com	By default only the administrator who added the application; up to five addresses can be set (use a distribution list for more). If set through Graph or PowerShell, open the SAML certificate page in the admin center to confirm the notification registration; otherwise the emails might not be sent

SIGNAL	WHAT IT COVERS	WHO SEES IT AND CAVEATS (MICROSOFT)
Recommendation: Renew expiring application credentials	Credentials on an <i>application registration</i> that expire within the next 30 days	Entra ID > Overview > Recommendations; readable by Reports Reader, Security Reader or Global Reader. Currently listed as preview; the page lists a required licence of Workload ID. Data is processed daily and can lag, up to 72 hours at times. Email notifications for recommendations (preview) go to the Application Administrator role, and with PIM the recipient must be elevated or no one gets the email
Recommendation: Renew expiring service principal credentials	The same idea for credentials on service principals	Listed in Microsoft's table, also preview; read its page for the detail
Usage & insights: Application credential activity (preview)	For every application credential: its type (certificate or secret), last used date and expiration date , in one list	Entra ID P1 or P2; Reports Reader at least. The nearest thing to a built-in register
Recommendation: Remove unused credentials from applications	Old credentials no longer used	Helps with the "remove" step of Chapter 8; preview

MICROSOFT DOCUMENTS SAML EMAILS; DON'T ASSUME THE SAME FOR SECRETS

The page for managing certificates describes the 60/30/7 email for SAML signing certificates. The documents consulted for client secrets and certificate credentials describe the 30-day recommendation and the credential-activity report, but not an automatic email to a named owner. Treat "Entra will warn us" as true for SAML certificates and *partly* true for everything else, and build your own warning for the rest.

The Expiry Register

The register is a list, one row per credential per client, that your team owns. It is deliberately a plain spreadsheet or ticketing-system view. Its value is that someone looks at it on a schedule.

COLUMN	WHY IT'S THERE
Client and tenant ID	From the tenant card (Chapter 1)
Integration and application (client) ID	Which object; search by ID, not name
Arrangement and pattern	A/B/C and OIDC / SAML / client credentials / SCIM, so you know who acts
Credential type, description and Key ID (or thumbprint)	Distinguishes the several secrets one app may hold; Microsoft's own check is matching the Key ID (Chapter 8). The value is never recorded
Created and expires (date)	The date everything hangs on. Record the date as shown in the portal; note the time zone question below
Owner (client) and owner (us)	A named person and a team mailbox on each side
Notification recipients set	Who Entra emails (SAML), and whether it was verified
Renewal plan and window	Who creates, who installs, when (a quiet period for single-certificate products)
Last verified	The date someone last looked at the live portal value

Where the data comes from

- **Your own tenant:** the Application credential activity report, or Microsoft Graph. The `passwordCredential` object has `displayName`, `startDateTime`, `endDateTime` (always UTC), `keyId` and a `hint` (first three characters). The secret text itself is returned only once, when created; there is no way to retrieve it later.
- **A client's tenant:** you rarely have the access to run reports, so the data arrives by request: the administrator reads dates to you on a screen share (Chapter 5), or exports the Application credential activity list. Add it to the register the same day.
- **Your product:** if it stores the credential, it should also store (and show) the *expiry date* and last-success time, so the register can be checked against what is in use.

```
client,tenant_id,integration,app_id,arrivalment,type,description,key_id,expires,client_of
Fabrikam,1111...,Acme Reports Sync,2222...,B/client-creds,secret,"prod 2026-10",aaaa...,
Contoso,5555...,Acme SSO,6666...,C/SAML,signing cert,"Entra generated",bbbb...,2026-12-0
Northwind,7777...,Acme multi-tenant,8888...,A/OIDC,secret,"vendor 2026",cccc...,2027-03-
```

When to Renew: Lead Times

Entra's own signals suggest the rhythm: 60, 30 and 7 days for SAML emails; a 30-day window for the credential recommendation. A practical schedule for your register, scaled to how hard the client is to coordinate:

DAYS BEFORE EXPIRY	ACTION
90	Register flags it; send the first client reminder (template below); confirm the owner still exists
60	Agree the renewal window; for SAML, confirm notification recipients; check the client's lifetime policy
30	Create the new credential alongside the old one (it does no harm, since the old one keeps working); for SAML create the Inactive certificate
14	Install and prove (Chapter 8); activate SAML certificates; leave the old credential in place
7	Escalate if not done; the last chance for a calm renewal
After expiry + safe period	Remove the old credential; update the register with the new date; close the loop with the client

TIME OF DAY IS NOT VERIFIED

The documents consulted give expiry dates but not the hour or time zone at which a secret stops working. Don't plan to renew "on the last day". Renew at least a week early, so the exact moment never matters.

Choosing Lifetimes and Credential Types

- **Client secrets** have a maximum lifetime of 24 months; Microsoft recommends under 12. Longer isn't available, so "set it and forget it" doesn't exist for secrets.
- **SAML signing certificates** default to three years and can be set to any date up to three years ahead; the date can't be edited after saving.
- **Certificates over secrets.** Microsoft recommends certificates for production. A certificate doesn't remove expiry, but a CA-issued certificate kept in a vault such as Azure Key Vault is

easier to renew in a controlled way than a pasted string.

- **No stored secret at all, where possible.** Microsoft's guidance for migrating away from secrets points to managed identities (for workloads running in Azure) and federated identity credentials (for supported external workloads). These apply to *your* workloads, and only where the platform supports them; they are not something a client can switch on for a product that does not support them.
- **The client's policy may decide for you.** Application management policies can limit the maximum lifetime of passwords and certificates, or block adding secrets (Chapter 4). Ask during onboarding what their limits are, and design for the shortest.
- **Don't confuse short with safe.** Short lifetimes shrink the damage of a leak but multiply the number of renewals. If you shorten lifetimes, the register and the runbook must be solid first.

Making Rotation Boring in Your Product

This is the part you control completely, and it is where most recurring pain is removed. Ask your engineering team for these, in rough priority order:

1. **Accept two credentials at once.** Support an old and a new secret or certificate in parallel, so the sequence create, install, prove, remove never involves downtime or a precisely timed swap.
2. **For SAML, read the federation metadata URL regularly** (Microsoft recommends at least every 24 hours), add newly published signing certificates as secondary, and promote them when Entra activates them. Then the client's rollover needs no action from you.
3. **Show the expiry.** The connection settings page should display which credential is in use, its expiry date if it can be known, and the date of the last successful sign-in.
4. **Warn early and loudly.** Product-side warnings at 60, 30 and 7 days to the client's named contacts, and an alert to your own support team.
5. **Make "validate the connection" informative.** Distinguish "credential expired" (7000222), "credential invalid" (7000215), "consent missing" and "blocked by Conditional Access" with plain-language messages, instead of a generic failure. Chapters 6 to 8 give you the mapping.
6. **Never log or display the secret** once stored; record only a short hint, such as the first three characters (the same approach as Graph's `hint`).

Catching Failures Early

If prevention fails, you want the first failure, not the thousandth. Two layers:

- **In your product:** alert your support team on the first failed token request or provisioning cycle for any client, with the error code attached.
- **In the client's tenant:** an administrator with at least Security Administrator can send the Entra logs to a Log Analytics workspace through **Entra ID > Monitoring & health > Diagnostic settings** and build alerts on failed sign-ins for the integration's application. This is a client-side project with its own cost and decision-making, so offer it; don't assume it. Chapter 7's retention figures (7 days Free, 30 days P1 and P2) are the reason many clients benefit.

Client-Facing Prevention

Onboarding checklist (ask once, record the answers)

- Named owner for the integration, and a **shared mailbox** rather than one person.
- Notification recipients set on the SAML certificate (up to five; use a distribution list).
- Their application management policy limits, if any.
- Which licence they hold (P1/P2), because it decides log retention and which reports exist.
- How a renewal is requested and approved on their side, and the usual lead time.
- Whether they allow us a read role such as Reports Reader for diagnosis.

A renewal reminder you can send at 90 days

Subject: Action needed by <date>: renewal of the <product> connection to your Microsoft

Hello <name>,

The credential that lets <product> connect to your Microsoft Entra tenant is due to expire <date> (about 90 days from now). If it isn't renewed first, your users will not be able to sign in to <product> (or <nightly sync will stop>) from that day.

Renewing takes about ten minutes, and nothing is removed until we have confirmed the new

1. Please tell us who on your side will do this, and a good week to schedule it.

2. We will send you the exact steps and join a short call to test afterwards.

We will remind you again at 30 days. If you would like the reminders sent to a shared mailbox instead of an individual, let us know the address.

Thank you,
<name>

WHAT YOU CAN TELL THE CLIENT

"Credentials like this are designed to expire, for security. It is a routine renewal, not a fault, and we'll remind you well in advance. If you tell us who should receive the reminders, a shared mailbox is best because people change roles. The renewal itself is short and doesn't affect your users when it's done in the right order."

Hands-On Exercises

All three use fictional organisations and values. Never put real secrets, tokens or tenant details into practice notes.

EXERCISE 1

Today is 1 October 2026. You support four integrations: (1) Fabrikam, arrangement B, client secret expiring 20 October 2026; (2) Contoso, SAML signing certificate expiring 5 December 2026, notification emails go to one person who left in August; (3) Northwind, arrangement A, your own multitenant app's secret expiring 1 March 2027; (4) Tailspin, certificate credential expiring 9 October 2026, no owner recorded. Build the register rows, work out for each which step of the lead-time schedule it is already at, rank them by urgency, and write the single next action for each.

 [View solution](#)

EXERCISE 2

Write the 30-day reminder to Fabrikam's IT manager, who replied to the 90-day reminder with "we'll sort it" and has not been in touch since. The message must be short, say plainly what happens if nothing is done, propose a specific renewal slot, name what you need from them and what you will do, and avoid blame and jargon. Then list the three things you do inside your own team that same day.

EXERCISE 3

Post-incident review: Contoso's SAML certificate expired. Entra's expiry emails went to an administrator who left three months earlier. Your product accepts one certificate only. The renewal took four hours of downtime on a weekday. List at least six systemic fixes (not "be more careful"), sort them into what your team controls and what you can only request of the client, and pick the two with the biggest effect, explaining why.

CHAPTER 9 QUICK REFERENCE

- Expiry is predictable; incidents come from no list, no owner, wrong recipient or risky renewal
- Arrangement decides your reach: A you control everything; B/C you control your product's side and request the rest
- SAML cert emails at **60/30/7 days**, default to the app's creator, up to **five** recipients; verify the notification registration if set by script
- Recommendation "Renew expiring application credentials" flags credentials expiring in **30 days** (preview; daily, up to 72h lag; emails to the Application Administrator role, PIM caveat)
- Usage & insights **Application credential activity** (preview) lists type, last used and expiry; needs P1/P2 and at least Reports Reader
- Register columns: client, tenant, app ID, arrangement, type, description, Key ID/thumbprint, expiry, owners, notify set, verified; never the secret
- Schedule: 90 flag, 60 plan, 30 create new, 14 install and prove, 7 escalate; renew at least a week early
- Secrets max 24 months (under 12 recommended); SAML cert up to three years; prefer certificates; check the client's application management policies
- Product: accept two credentials, read metadata at least every 24 h, show expiry, warn early, informative validation, never log secrets
- Early detection: first-failure alert in your product; optional Log Analytics alerts in the client's tenant (Security Administrator)
- Onboarding: named owner and shared mailbox, notification recipients, policy limits, licence, renewal process, read-role permission

Microsoft Entra ID: Integrations & Access Troubleshooting

Course 1 · Chapter 10 · Capstone: A Triage Playbook

Nine chapters of parts; this one is the whole machine. It joins everything into a single path from **"the client says they can't sign in"** to **root cause, fix, client message and prevention**, shows it working on three realistic cases, and ends with the artefacts your team can actually use: a flowchart, an evidence checklist, communication templates, an escalation checklist and a one-page runbook.

THE CASES ARE FICTIONAL, AND SO IS THE PRODUCT

All three cases, organisations, IDs and log lines are invented, built from behaviour documented in the earlier chapters. Your real error and your four open questions (which integration type you use, which arrangement, what access you have, where "validate the connection" runs) were never supplied to the course. Exercise 2 and the closing section show how to fill those gaps in yourself, which is what makes the playbook yours.

The Playbook at a Glance

Read the tree top to bottom. Each branch names the chapter that explains it.

CLIENT CAN'T SIGN IN / INTEGRATION FAILING

- |
- |— 0. PRESERVE: save the error text, time + zone, a log download (Ch 7, 8)
- |
- |— 1. WHO is affected, and since when? (Ch 6 table)
 - |— everyone, started at one moment credentials / disabled app / policy / con
 - |— some users assignment (nested groups?) / group-target
 - |— only new users user consent disabled or tightened
 - |— only guests guest account / cross-tenant / untrusted I
 - |— one place or network location Conditional Access
 - |— background/API only app permission / workload policy / secret
 - |— right after OUR release new permission requested
- |
- |— 2. RIGHT TENANT? RIGHT APP? tenant card, client ID, both objects (Ch 1, 5)
 - |— five-minute inspection: Enabled? Assignment required? credentials? URIs match?
- |
- |— 3. THE LOG: which sign-in tab? filter by correlation ID (Ch 7)

```

|   | 7000222 / 7000215 / 50012 / 700027 / 700030 ... CREDENTIAL      -> Ch 8 fix
|   | SAML assertion rejected / cert date passed ..... SAML CERT    -> Ch 8 fix C
|   | 65001 / 90094 / 65004 ..... CONSENT                          -> Ch 6
|   | 50105 ..... ASSIGNMENT                                         -> Ch 6
|   | 53003 / 50076 / 50079 / 70043 ..... CONDITIONS                -> Ch 6
|   | 50020 / 50034 / 500011 / 700016 / 90002 ..... TENANT / ACCOUNT / APP MISSING
|   | 50011 ..... REDIRECT URI                                       -> Ch 3, 5
|   | 7000112 ..... APP DISABLED                                     -> Ch 5
|   | 700082 / 70008 / 50173 ..... TOKEN, not integration (Ch 4)
|   |  nothing found ..... check other tabs, narrower window
|
| 4. CONFIRM IN THREE PLACES: log, portal, audit log                  (Ch 8)
|   |  four stories: expired / renewed one side / deleted / not the credential
|
| 5. FIX IN SAFE ORDER: create, install, PROVE, remove               (Ch 8)
|   |  adjust the rule for the integration, never switch security off (Ch 6)
|
| 6. TELL THE CLIENT at diagnosis, plan, done                        (Ch 5-9 boxes)
|
| 7. PREVENT: register, owner, recipients, lead times, product      (Ch 9)

```

The Evidence You Record Every Time

From Chapters 3, 7 and 8, one list. Copy it into the ticket before any change is made, and keep it in text so it can be searched later (redact tenant IDs, client IDs and names before it leaves your team):

GROUP	FIELDS
The error	error name; error_description; AADSTS number; trace ID; correlation ID; timestamp (and time zone)
Who and what	tenant ID (tenant card); application (client) ID; resource; user or service principal; user type (member or guest)
How	pattern (OIDC, SAML, client credentials, SCIM); arrangement A, B or C; authentication
Entra state	Enabled for users to sign in?; Assignment required?; credentials with expiry dates and Key IDs; permissions granted; Conditional Access result and policy names
Changes	audit entries for the relevant days: who, what, when

Case 1: The Overnight Job That Stopped

Fictional setting. Fabrikam's nightly sync into your product stopped on 2 October. Their users can still sign in. Arrangement B (the registration is in Fabrikam's tenant), client-credentials pattern.

STEP	FINDING	DECISION
0 Preserve	Job log shows <code>invalid_client</code> , AADSTS7000222, 02:00:04 UTC, with trace and correlation IDs. You download the day's service-principal sign-ins	Evidence safe
1 Who	Only the background job; interactive sign-in is fine	Credential / permission / workload policy suspected. Not assignment or user consent (no user involved)
2 Right app	Tenant ID matches the tenant card; client ID matches. Enterprise app "Enabled for users to sign in? = Yes"	Not Chapter 5's master switch
3 Log	Service principal sign-ins, Failure, 7000222, Conditional Access Not applied	Credential family, and not a policy
4 Confirm	Portal: secret "prod 2025" expired 2026-09-28; no newer secret. Audit: no changes	Story 1: simply expired
5 Fix	Client administrator creates "prod 2026-10" (under 12 months), enters the Value into your product, you prove with a fresh Success, remove the old secret a week later	Create, install, prove, remove
6 Tell	"The credential expired on 28 September; nothing wrong with your users or tenant; replacement takes ten minutes."	Diagnosis, plan, done messages
7 Prevent	Register row with owner and 90-day flag; shared mailbox; 14-day install step	Chapter 9

Case 2: SAML Sign-in Failing for Everyone

Fictional setting. From Monday 5 December every Contoso user gets a SAML error page in your product. Arrangement C. Your product holds one signing certificate.

STEP	FINDING	DECISION
0 Preserve	The product's error says the assertion signature was rejected; user reports begin 08:40	Save text and times
1 Who	Everyone, starting at one moment, a date that looks round	Expiry or a change; test the certificate date first
2 Right app	Enterprise app Single sign-on > SAML Certificates: the Active certificate expired 2026-12-04; an Inactive certificate valid to 2029 exists	Strong lead
3 Log	Interactive sign-ins show Entra issuing tokens; the failure is on your side. (Entra records the sign-in; your product rejects the assertion)	Not a Chapter 6 gate
4 Confirm	Audit log: the Inactive certificate was created in August by the former administrator. Your product still trusts only the expired certificate. Story 2: renewed on one side only	Fix C
5 Fix	Agree a short window; download the 2029 certificate, upload to your product, Make certificate active back to back; test user signs in; remove the old certificate	Single-certificate product means a pair of steps
6 Tell	"Your signing certificate expired on 4 December; a replacement already existed but our system hadn't been given it. We'll swap it in now, which takes about 15 minutes."	Plain statement, no blame
7 Prevent	Support multiple certificates and read the federation metadata URL at least every 24 hours; notification list to a distribution list; register entry	The strongest fix: remove the human step

Case 3: The Integration Is Healthy, but Access Was Removed

Fictional setting. After Northwind's security review on Thursday, a number of staff can no longer open your product. The credentials are fine and nothing expired. Arrangement C (SSO), OIDC for the API.

STEP	FINDING	DECISION
0 Preserve	Users report an "Approval required" screen (new starters) or an error naming no assigned role (existing staff)	Two symptoms, so two causes
1 Who	Some existing users; all new users	Some: assignment. New: user consent

STEP	FINDING	DECISION
2 Right app	Properties: Enabled = Yes; Assignment required = Yes (was No last week). Users and groups: only "Support Staff" assigned	A setting, not a credential
3 Log	Existing users: AADSTS50105. The new-starter screen is the admin consent workflow ("Approval required")	Chapter 6, gates 2 and 1
4 Confirm	Audit log, Thursday afternoon: Update service principal (assignment required changed) and a change to the tenant's user-consent setting	Story 4: not the credential. Do NOT rotate anything
5 Fix	Their administrator assigns the "Sales" and "Operations" groups (directly, since nested groups don't inherit), and either grants admin consent for the app's permissions or names reviewers for the consent workflow	Adjust the rule for the integration, do not switch the controls off
6 Tell	"Your tenant now restricts who may use applications and who can approve them. That is a sound control; we just need the right groups and approver on the list."	Treat the change as legitimate
7 Prevent	Add "assignment and consent settings" to the onboarding checklist; ask the client to tell us before security reviews that change them	Chapter 9

Client Communication Template

A reusable shape for the three stages from Chapter 8, plus the update rhythm in between:

1. ACKNOWLEDGE (within the hour)

"We have your report and are investigating. First, can you confirm the time it started, whether it affects everyone?"

2. DIAGNOSIS (when the log confirms it)

"We found the cause: <one plain sentence, e.g. the credential used by the connection on <date>>. Your users' accounts and your tenant are not at fault. It is fixed by <action>."

3. PLAN

"We'd like <role> to <action> (about <n> minutes). Nothing is removed until we've confirmed one works. We never need a password or secret value."

4. UPDATES (at the interval you promised)

"Status at <time>: <done / waiting on / next step>. Next update at <time>."

5. DONE

"Sign-in has worked again since <time>. We'll remove the old credential on <date> and a reminder for <next expiry> to the shared mailbox you chose."

DON'T: blame a named person; promise a cause before the log confirms it; ask for a secret; suggest turning off a security control; use acronyms the reader hasn't met.

Escalation Checklist

Escalate with the evidence list already filled in, so the next person doesn't start from nothing.

ESCALATE TO	WHEN	SEND
Our engineering team	The failure is in our product: assertion rejected, credential not used, validation message wrong, cached secret, no multiple-certificate support	Error text, version, tenant card, steps, logs from our side, the portal facts (dates, thumbprints)
The client's Entra administrator	A setting or credential in their tenant must change: assignment, consent, policy, new credential	Specific page names (Chapter 5), exactly what to change and why, and the safe order
The client's security team	Conditional Access, cross-tenant or consent policy is blocking the integration	Policy name from the log's Conditional Access tab, the integration's fixed IP or identity, and a narrow request
Microsoft support (via the client, unless we hold support rights in the tenant)	Credentials are valid, consent and assignment are in order, no policy applies, yet Entra still returns an unexpected failure	Tenant ID, application ID, correlation ID and request ID, timestamp with time zone, error code and trace ID, what was changed and when, and a statement of what has been ruled out. Microsoft's guidance for unresolved sign-in problems is to open a support request
Our incident process	Many clients are affected at once, a credential may be exposed, or a security control looks bypassed	Follow the incident workflow from the Incident Response & Ticketing Workflows course; treat any leaked secret as compromised and rotate it

The One-Page Runbook

- | | |
|----------------|--|
| 1. PRESERVE | Error text, time + zone, download Entra logs (Free keeps 7 days; P1/P2 |
| 2. IDENTIFY | Tenant card: client, tenant ID, app (client) ID, arrangement A/B/C, pat |
| 3. WHO FAILS | All / some / new / guests / one place / background / after release -> |
| 4. LOOK | Sign-in log (right tab) by correlation ID. Copy: code, reason, app, res |
| 5. INSPECT | Right tenant? Enterprise app: Enabled? Assignment required? Credentials
URIs/thumbprints match our side? Permissions granted? (Read-only; chang |
| 6. CONFIRM | Log + portal + audit log agree on ONE cause. If not, keep looking (never |
| 7. FIX | Credential: create new -> install -> PROVE -> remove old. Secrets or
SAML cert: New Certificate (Inactive) -> upload to us -> Make certifi
SCIM: new token -> Secret Token -> Test Connection -> check pro
Consent / assignment / policy: client admin changes; adjust, never disal |
| 8. PROVE | New Success in the right sign-in tab; Key ID/thumbprint matches; real d |
| 9. COMMUNICATE | Acknowledge / diagnosis / plan / updates / done (templates). No blame, i |
| 10. RECORD | Symptom, evidence, cause, fix, proof, prevention date, owner. |
| 11. PREVENT | Register row, owner + shared mailbox, 90/60/30/14/7 reminders, notifica |
| ESCALATE | Product fault -> Engineering Setting/credential -> client Entra admin
Unexplained after 1-6 -> Microsoft support with IDs Many clients/expos |
| NEVER | Ask for a secret by email/chat. Remove the old credential before proving
service principal to "tidy". Ask a client to switch off a security polic |

Where This Meets the Rest of the Site

This course is the Entra-specific layer on top of general support skills taught elsewhere on the site: tokens, sessions and OAuth in *Authentication & Session Security*; certificates in *HTTPS/TLS Fundamentals*; reading logs in *Logging & Log Analysis*; diagnosing a failing application in *Web & Application Troubleshooting*; ticket handling in *Incident Response & Ticketing Workflows*; wording for non-technical readers in *Customer/User Communication for Support*; turning a procedure into a team asset in *Documentation & Runbooks*; and the habits that keep secrets safe in *Security Basics for Support Technicians*.

Make It Yours: Three Tasks

1. **Answer the four open questions** for your own systems and write them at the top of the runbook: which integration type(s) you use (SAML, OIDC with a secret or certificate, SCIM, or

another); whether you use arrangement A, B or C (or a mixture); what access you normally have in a client's tenant (none, guest, a read role, or screen share only); and whether the "validate the connection" check is in your product or in Microsoft's portal.

2. **Collect three real, redacted errors** from past tickets and file each under the playbook's log-code branches. Anything that doesn't fit shows a gap worth fixing.
3. **Fill the first ten rows of the expiry register** from your busiest clients, using Chapter 9's columns, and agree the owner for each.

Course Complete: What You Can Now Do

CHAPTER	YOU CAN NOW...
1. What Entra ID Is	Explain tenants, the rename from Azure AD, and whose side owns what; keep a tenant card
2. How Applications Integrate	Tell an app registration from an enterprise application; recognise OIDC, SAML, client credentials and SCIM, and arrangements A, B and C
3. What Passes Between Entra and Your System	Follow a sign-in step by step, read token claims, and know where each step fails
4. Credentials and Expiry	Distinguish credential expiry from token expiry; know lifetimes, the two-sided trap and where to look
5. Finding the Integration	Switch to the right tenant, find both objects, know your role's reach, and run the five-minute inspection
6. Permissions and Consent	Separate permission, assignment and Conditional Access causes; read the consent and policy errors
7. Reading the Sign-in Logs	Pick the right log and tab, filter by correlation ID, know retention, and capture evidence
8. Fixing an Expired Credential	Diagnose in three places, fix each credential type safely and prove it
9. Preventing It Happening Again	Run a register, use Entra's notices, choose lifetimes, and make rotation routine
10. Capstone	Triage any "can't sign in" report with one playbook, communicate it, escalate it and document it

Entra changes often. The concepts in this course (identity provider, two objects, credentials versus tokens, three gates, evidence first, create-install-prove-remove) will outlast the menu names; keep Microsoft Learn open when you need the current spelling of a portal path or a figure, and update your runbook when it changes.

Hands-On Exercises

All three use fictional organisations and values. Never put real secrets, tokens or tenant details into practice notes.

EXERCISE 1

Run the playbook on a new case. Tailspin's partner firm Litware has guests in Tailspin's tenant who use your product via SSO. Since Monday morning, Litware's staff (and only Litware's staff) get blocked. Tailspin's own staff are fine. Two of the Litware guests' sign-in log entries show AADSTS53003 and the Conditional Access tab lists a policy "Require compliant device" = Failure; a third shows AADSTS50020. Work through every branch: who is affected, the likely gates, what you check in the portal, the story, the fix, the message to Tailspin and the prevention step. State which parts you can be confident about and which need more evidence.

 [View solution](#)

EXERCISE 2

Personalise the runbook. Write answers to the four open questions as if for a real or realistic product (or the fictional "Acme Support Desk": multitenant app, OIDC and SAML, a vendor support role with screen share only, "validate the connection" inside the product's admin page). Then edit three steps of the one-page runbook so they match: step 2 (what the tenant card holds), step 5 (what you can inspect yourself and what you must ask for), and step 7 (the fixes you actually perform).

 [View solution](#)

EXERCISE 3

A diagnosis turns out wrong. In Case 1 you told Fabrikam "the secret expired", sent the diagnosis message, and they created a new secret. The job still fails, now with AADSTS7000215 (invalid secret). Write the correction message, state what you got wrong and what the evidence now says, and list what

you will change in your process so a wrong early diagnosis is less likely. Then re-order the evidence you should have gathered before sending the first message.

 [View solution](#)

CHAPTER 10 QUICK REFERENCE

- Path: **preserve, who is affected, right tenant and app, the log, confirm in three places, fix in safe order, tell the client, prevent**
- Code families: 7000222/7000215/50012/700027/700030 credential; 65001/90094/65004 consent; 50105 assignment; 53003/50076/50079/70043 conditions; 50020/50034/500011/700016/90002 tenant, account or app missing; 50011 redirect URI; 7000112 disabled app; 700082/70008/50173 token
- Four stories: expired, renewed one side only, deleted/replaced, not the credential (then do not rotate anything)
- Safe order: **create, install, prove, remove**; SAML: new cert Inactive, upload to us, Make certificate active, test
- Escalate with the evidence list: product fault to engineering; setting or credential to the client's administrator; policy to their security team; unexplained to Microsoft with tenant, app, correlation and request IDs
- Never: ask for a secret by email or chat, remove before proving, delete to tidy, ask to disable security, rotate a healthy secret
- Prevent: register, owners and shared mailbox, 90/60/30/14/7 reminders, verified notifications, product that accepts two credentials and reads federation metadata
- Make it yours: answer the four open questions, file three real errors, fill ten register rows