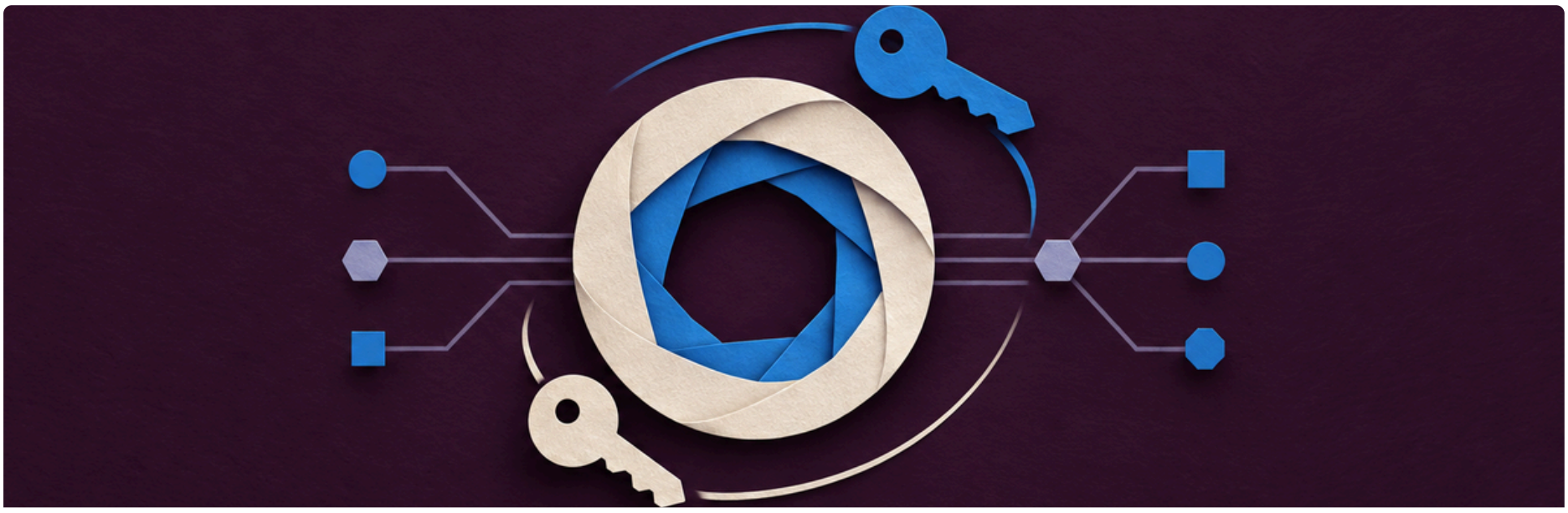
From deployment to optimization, get practical Microsoft 365 and Copilot Studio insights at ESPC Microsoft 365 and AI Conference. Register today!



AZURE INTEGRATION SERVICES BLOG 9 MIN READ

Rotating credentials in Azure API Management



vladvino MICROSOFT
Nov 21, 2026

This guide explains how to find and rotate the secrets and credentials configured in your Azure API Management instances. It includes example Azure CLI operations and links to public documentation. Test every command in a non-production instance first.

Credentials covered

- Subscription keys
- Named values that contain secrets
- Backend credentials (headers, query parameters, client certificates)
- OAuth 2.0 and OpenID Connect client secrets
- Connection strings (for example, Event Hubs loggers)
- Credentials written directly into policies

How to rotate safely

Use this order when both credentials can remain valid:

1. Create the replacement at the issuer, such as the backend service or identity provider.
2. Update API Management with the new secret.
3. Test your APIs.
4. Revoke the old secret.

Find your instances and credentials

Step 1: List your API Management instances

In the Azure portal, search for API Management services. For command-line examples, choose Azure CLI or Azure PowerShell for each operation and fill in the placeholders.

The PowerShell examples require PowerShell 7.4 or later, `Az.Accounts` 5.0.0 or later, and current versions of `Az.ApiManagement`, `Az.ResourceGraph`, `Az.Resources`, and `Az.EventHub`.

Sign in to Azure PowerShell

```
Connect-AzAccount -ErrorAction Stop
$subscriptionId = '{subscriptionId}'
Set-AzContext -Subscription $subscriptionId -ErrorAction Stop | Out-Null
```

For Azure CLI, sign in with `az login`.

List instances in one subscription

Azure CLI

```
az apim list --subscription {subscriptionId} --query "[].{name:name, resourceGroup:resourceGroup}" --output table
```

Azure PowerShell

```
Get-AzApiManagement | Format-Table Name, ResourceGroupName
```

List instances across accessible subscriptions

These examples return one Azure Resource Graph page.

Azure CLI

```
az graph query -q "resources | where type =~ 'microsoft.apimanagement/service' | project name, resourceGroup,
subscriptionId, location"
```

Azure PowerShell

```
$query = "resources | where type =~ 'microsoft.apimanagement/service' | project name, resourceGroup,
subscriptionId, location"
Search-AzGraph -Query $query -UseTenantScope
```

Step 2: List the credentials in each instance

Run the setup for your tool. For `{subscriptionId}`, use the ID of the Azure subscription that contains the API Management instance. For `{sid}`, use the ID of a subscription within that instance.

Azure CLI setup

If you aren't signed in, run `az login`. Select the instance's Azure subscription:

```
az account set --subscription "{subscriptionId}"
```

Fill in `{subscriptionId}`, `{rg}`, and `{apimName}` in this URL. Use the result in place of `{base}` in the CLI examples.

```
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{rg}/providers/Microsoft.ApiManagement/s
ervice/{apimName}
```

If you haven't signed in, run `Connect-AzAccount`. Run this setup in the PowerShell session you'll use for the remaining examples.

```
$subscriptionId = '{subscriptionId}'
$azContext = Set-AzContext -Subscription $subscriptionId -ErrorAction Stop
$rg = '{rg}'
$apimName = '{apimName}'
$apimResourceId =
"/subscriptions/$subscriptionId/resourceGroups/$rg/providers/Microsoft.ApiManagement/service/$apimName"
$apimContext = New-AzApiManagementContext -ResourceId $apimResourceId -DefaultProfile $azContext -ErrorAction Stop
$apiVersion = '2024-05-01'
```

List subscriptions

Subscription keys are under APIs > Subscriptions in the Azure portal.

Azure CLI

```
az rest --method get --url "{base}/subscriptions?api-version=2024-05-01"
```

Azure PowerShell

```
Get-AzApiManagementSubscription -Context $apimContext -DefaultProfile $azContext
```

List named values

Check the Type column under APIs > Named values in the portal, or the `KeyVaultBacked` column in either command's output.

Azure CLI

```
az apim nv list --resource-group {rg} --service-name {apimName} \
  --query '[].{NamedValueId:name, Name:displayName, Secret:secret, KeyVaultBacked:!!(keyVault.secretIdentifier)}' \
  --output table
```

Azure PowerShell

```
Get-AzApiManagementNamedValue -Context $apimContext -DefaultProfile $azContext |
    Format-Table NamedValueId, Name, Secret, @{ Name = 'KeyVaultBacked'; Expression = {
[bool]$_.KeyVault.SecretIdentifier } }
```

List backends

Backend credentials are under APIs > Backends > Authorization credentials.

Azure CLI

```
az rest --method get --url "{base}/backends?api-version=2024-05-01"
```

Azure PowerShell

```
Get-AzApiManagementBackend -Context $apimContext -DefaultProfile $azContext
```

List OAuth and identity providers

OAuth 2.0 and OpenID Connect providers are under Developer portal > OAuth 2.0 + OpenID Connect. Developer portal sign-in providers are under Developer portal > Identities.

OAuth 2.0 authorization servers

Azure CLI

```
az rest --method get --url "{base}/authorizationServers?api-version=2024-05-01"
```

Azure PowerShell

```
Get-AzApiManagementAuthorizationServer -Context $apimContext -DefaultProfile $azContext
```

Azure CLI

```
az rest --method get --url "{base}/openidConnectProviders?api-version=2024-05-01"
```

Azure PowerShell

```
Get-AzApiManagementOpenIdConnectProvider -Context $apimContext -DefaultProfile $azContext
```

Developer portal sign-in providers

Azure CLI

```
az rest --method get --url "{base}/identityProviders?api-version=2024-05-01"
```

Azure PowerShell

```
Get-AzApiManagementIdentityProvider -Context $apimContext -DefaultProfile $azContext
```

List Credential Manager providers

In the portal, go to APIs > Credential manager.

Azure CLI

```
az rest --method get --url "{base}/authorizationProviders?api-version=2024-05-01"
```

Azure PowerShell

```
Invoke-AzRestMethod -Method GET -Path "$apimResourceId/authorizationProviders?api-version=$apiVersion" -
DefaultProfile $azContext
```

List loggers

Check the logger list for Event Hubs and Application Insights connections.

Azure CLI

```
az rest --method get --url "{base}/loggers?api-version=2024-05-01"
```

Azure PowerShell

```
Get-AzApiManagementLogger -Context $apimContext -DefaultProfile $azContext
```

Inspect policies

Check global, API, and operation policies under Design. For product policies, select Products > the product > Policies.

Look for hard-coded passwords, tokens, connection strings, and credentials in headers. Keep the full policy XML from the CLI out of logs. The PowerShell examples print only line numbers for possible matches. Check those lines in the portal's policy editor.

Global policy

Azure CLI

```
az rest --method get --url "{base}/policies/policy?format=rawxml&api-version=2024-05-01"
```

Azure PowerShell

```
$globalPolicy = Get-AzApiManagementPolicy -Context $apimContext -Format rawxml -DefaultProfile $azContext -
ErrorAction Stop
($globalPolicy -split '\r?\n') |
    Select-String -Pattern 'authorization|password|token|key|connection.?string' |
    Select-Object -ExpandProperty LineNumber
Remove-Variable globalPolicy
```

Azure CLI

```
az rest --method get --url "{base}/apis/{apiId}/policies/policy?format=rawxml&api-version=2024-05-01"
```

Azure PowerShell

```
$apiPolicy = Get-AzApiManagementPolicy -Context $apimContext -ApiId '{apiId}' -Format rawxml -DefaultProfile
$azContext -ErrorAction Stop
($apiPolicy -split '\r?\n') |
    Select-String -Pattern 'authorization|password|token|key|connection.?string' |
    Select-Object -ExpandProperty LineNumber
Remove-Variable apiPolicy
```

Step 3: Read the current secret values

Check configuration and deployment records to see what uses each credential. Read a secret only if you need its value and have permission. Keep CLI output out of logs. Don't print the PowerShell variables. Clear them with `Remove-Variable` when you're done.

Subscription keys

Azure CLI

```
az rest --method post --url "{base}/subscriptions/{sid}/listSecrets?api-version=2024-05-01"
```

Azure PowerShell

```
$subscriptionKeys = Get-AzApiManagementSubscriptionKey -Context $apimContext -SubscriptionId '{sid}' -
DefaultProfile $azContext
```

Named value secret

Azure CLI

```
az apim nv show-secret --resource-group {rg} --service-name {apimName} --named-value-id {namedValueId}
```

Azure PowerShell

```
$namedValueSecret = Get-AzApiManagementNamedValueSecretValue -Context $apimContext -NamedValueId '{namedValueId}' -
DefaultProfile $azContext
```

OAuth 2.0 authorization server secret

Azure CLI

```
az rest --method post --url "{base}/authorizationServers/{id}/listSecrets?api-version=2024-05-01"
```

Azure PowerShell

```
$authorizationServerSecret = Get-AzApiManagementAuthorizationServerClientSecret -Context $apimContext -ServerId
'{id}' -DefaultProfile $azContext
```

OpenID Connect provider secret

Azure CLI

```
az rest --method post --url "{base}/openidConnectProviders/{id}/listSecrets?api-version=2024-05-01"
```

Azure PowerShell

```
$oidcProviderSecret = Get-AzApiManagementOpenIdConnectProviderClientSecret -Context $apimContext -
OpenIdConnectProviderId '{id}' -DefaultProfile $azContext
```

Developer portal identity provider secret

Azure PowerShell

```
$identityProviderSecret = Get-AzApiManagementIdentityProviderClientSecret -Context $apimContext -Type
'{identityProviderType}' -DefaultProfile $azContext
```

Legacy management access keys

Azure CLI

```
az rest --method post --url "{base}/tenant/access/listSecrets?api-version=2024-05-01"
```

Azure PowerShell

```
$tenantAccessKeys = Get-AzApiManagementTenantAccessSecret -Context $apimContext -DefaultProfile $azContext
```

Step 4: Rotate each credential type

Run the setup for your tool in Step 2 and keep the same terminal session. Repeat the setup if you switch instances.

Subscription keys

Move callers to the secondary key and verify their requests succeed. Regenerate the primary key, test it, and move callers to it. Verify they've switched before regenerating the secondary key.

Regenerate the primary subscription key

Azure CLI

```
az rest --method post --url "{base}/subscriptions/{sid}/regeneratePrimaryKey?api-version=2024-05-01"
```

Azure PowerShell

```
$sid = '{sid}'
$response = Invoke-AzRestMethod -Method POST -Path "$apimResourceId/subscriptions/$sid/regeneratePrimaryKey?api-
version=$apiVersion" -DefaultProfile $azContext -ErrorAction Stop
if ([int]$response.StatusCode -ne 204) { throw "Primary key regeneration returned HTTP $($response.StatusCode)." }
```

Regenerate the secondary subscription key

Azure CLI

```
az rest --method post --url "{base}/subscriptions/{sid}/regenerateSecondaryKey?api-version=2024-05-01"
```

Azure PowerShell

```
$sid = '{sid}'
$response = Invoke-AzRestMethod -Method POST -Path "$apimResourceId/subscriptions/$sid/regenerateSecondaryKey?api-
version=$apiVersion" -DefaultProfile $azContext -ErrorAction Stop
if ([int]$response.StatusCode -ne 204) { throw "Secondary key regeneration returned HTTP $($response.StatusCode)."
}
```

See Subscriptions in Azure API Management.

Named values

Refresh a Key Vault-backed named value

Azure CLI

```
az rest --method post --url "{base}/namedValues/{namedValueId}/refreshSecret?api-version=2024-05-01"
```

Azure PowerShell

```
Sync-AzApiManagementKeyVaultSecret -ResourceId "$apimResourceId/namedValues/{namedValueId}" -DefaultProfile
$azContext
```

Update a secret stored in API Management

Replace the stored value. Consider moving the secret to Key Vault.

Azure CLI

```
az apim nv update --resource-group {rg} --service-name {apimName} --named-value-id {namedValueId} --value "
{newValue}"
```

Azure PowerShell

```
$newValue = Read-Host 'Replacement named value' -MaskInput
Set-AzApiManagementNamedValue -Context $apimContext -NamedValueId '{namedValueId}' -Value $newValue -DefaultProfile
$azContext -ErrorAction Stop
Remove-Variable newValue
```

See Use named values in Azure API Management policies.

Backend credentials

Stage the replacement at the backend. Update and test API Management before revoking the old credential.

Update a backend header credential

Use these examples only for a backend with a single credential. Use its current ETag.

Azure CLI

```
az rest --method patch --url "{base}/backends/{backendId}?api-version=2024-05-01" --headers 'If-Match={etag}' --
body '{"properties":{"credentials":{"header":{"{headerName}":["{newKey}"]}}}}'
```

Azure PowerShell

```
$backendId = '{backendId}'
$headerName = '{headerName}'
$uri = "https://management.azure.com$apimResourceId/backends/${backendId}?api-version=$apiVersion"
$token = (Get-AzAccessToken -ResourceTypeName Arm -DefaultProfile $azContext -ErrorAction Stop).Token
$null = Invoke-RestMethod -Uri $uri -Method Get -Authentication Bearer -Token $token `
    -ResponseHeadersVariable backendHeaders -ErrorAction Stop
$etag = @($backendHeaders['ETag'])[0]
if (-not $etag) { throw 'The backend response did not include an ETag.' }

$newKey = Read-Host 'Replacement backend key' -MaskInput
$body = @{
    properties = @{
        credentials = @{
            header = @{ $headerName = @($newKey) }
        }
    }
} | ConvertTo-Json -Depth 6

$null = Invoke-RestMethod -Uri $uri -Method Patch -Authentication Bearer -Token $token `
    -Headers @{ 'If-Match' = $etag } -ContentType 'application/json' -Body $body -ErrorAction Stop
```

See Backends in API Management.

In the Microsoft Entra tenant that owns the app, go to App registrations > your application > Certificates & secrets. Use the old secret's Secret ID as `{oldKeyId}`.

Create a replacement client secret

Azure CLI

```
az ad app credential reset --id {appId} --append --display-name "apim-rotation"
```

Azure PowerShell

```
Import-Module Az.Resources
$appId = '{appId}'
$password = New-Object -TypeName
'Microsoft.Azure.PowerShell.Cmdlets.Resources.MSGraph.Models.ApiV10.MicrosoftGraphPasswordCredential'
$password.DisplayName = 'apim-rotation'
$password.EndDateTime = (Get-Date).ToUniversalTime().AddYears(1)
$newCredential = New-AzADAppCredential -ApplicationId $appId -PasswordCredentials $password
```

Test the replacement and remove the old secret

In API Management, open Developer portal > OAuth 2.0 + OpenID Connect and update the secret on the matching server or provider. If you used PowerShell, use `$newCredential.SecretText` without printing it.

Republish the developer portal if it uses this configuration. Get a fresh token in the test console before removing the old secret.

Azure CLI

```
az ad app credential delete --id {appId} --key-id {oldKeyId}
```

Azure PowerShell

```
$appId = '{appId}'
Remove-AzADAppCredential -ApplicationId $appId -KeyId '{oldKeyId}'
```

See Add and manage application credentials in Microsoft Entra ID.

Connection strings

Renew an Event Hubs namespace key

Get the secondary connection string from the namespace's Shared access policies. If the API Management logger uses a named value, update that value. Otherwise, use Logger Update with the current ETag to set `properties.credentials.connectionString` to the secondary connection string. Keep the Event Hub name and other logger settings.

Confirm that new log events arrive and all consumers have switched off the primary key. Then renew it in the Event Hubs subscription.

Azure CLI

```
az eventhubs namespace authorization-rule keys renew --subscription {eventHubsSubscriptionId} --resource-group
{eventHubsResourceGroup} --namespace-name {namespace} --name {ruleName} --key PrimaryKey --output none
```

Azure PowerShell

```
New-AzEventHubKey -SubscriptionId '{eventHubsSubscriptionId}' `
    -ResourceGroupName '{eventHubsResourceGroup}' -NamespaceName '{namespace}' -Name '{ruleName}' `
    -KeyType PrimaryKey -ErrorAction Stop | Out-Null
```

See How to log events to Azure Event Hubs in Azure API Management.

Create the replacement at the source. Reference its Key Vault-backed named value as `{{display-name}}`. Test the policy before revoking the old credential.

See Use named values in Azure API Management policies.

Management API access keys

For legacy Direct Management API keys, verify callers have switched off each key before regenerating it. If they'll use a replacement key, test it before moving them.

Regenerate the primary management access key

Azure CLI

```
az rest --method post --url "{base}/tenant/access/regeneratePrimaryKey?api-version=2024-05-01"
```

Azure PowerShell

```
$response = Invoke-AzRestMethod -Method POST -Path "$apimResourceId/tenant/access/regeneratePrimaryKey?api-
version=$apiVersion" -DefaultProfile $azContext -ErrorAction Stop
if ([int]$response.StatusCode -ne 204) { throw "Primary access key regeneration returned HTTP
$($response.StatusCode)." }
```

Regenerate the secondary management access key

Azure CLI

```
az rest --method post --url "{base}/tenant/access/regenerateSecondaryKey?api-version=2024-05-01"
```

Azure PowerShell

```
$response = Invoke-AzRestMethod -Method POST -Path "$apimResourceId/tenant/access/regenerateSecondaryKey?api-
version=$apiVersion" -DefaultProfile $azContext -ErrorAction Stop
if ([int]$response.StatusCode -ne 204) { throw "Secondary access key regeneration returned HTTP
$($response.StatusCode)." }
```

See Tenant Access - Regenerate Primary Key.

Best practices

- Use managed identities instead of keys and connection strings wherever supported.
- Store secrets in Azure Key Vault and reference them as named values. Use a secret identifier without a version for automatic refresh.
- Rotate credentials regularly and automate rotation where possible.
- Review your configuration for subscriptions, users, policies or backends that you did not create.

Additional documentation

- Subscription - Regenerate Primary Key (REST API): https://learn.microsoft.com/rest/api/apimanagement/subscription/regenerate-primary-key
- Subscription - Regenerate Secondary Key (REST API): https://learn.microsoft.com/rest/api/apimanagement/subscription/regenerate-secondary-key
- Named Value - Refresh Secret (REST API): https://learn.microsoft.com/rest/api/apimanagement/named-value/refresh-secret
- Configure OAuth 2.0 in the developer portal test console: https://learn.microsoft.com/azure/api-management/api-management-howto-oauth2
- About API credentials and credential manager: https://learn.microsoft.com/azure/api-management/credentials-overview

API MANAGEMENT

2 Comment



vladvino MICROSOFT
Joined August 05, 2017

View Profile

Azure Integration Services Blog

Follow this blog board to get notified when there's new activity

Sign in to reply

Enjoying the article? Sign in to share your thoughts.

Sign in

Share this page

What's new

Surface Pro

Surface Laptop

Surface Laptop Studio 2

Copilot for organizations

Copilot for personal use

AI in Windows

Explore Microsoft products

Windows 11 apps

Microsoft Store

Account profile

Download Center

Microsoft Store support

Returns

Order tracking

Certified Refurbished

Microsoft Store Promise

Flexible Payments

Microsoft in education

Devices for education

Microsoft Teams for Education

Microsoft 365 Education

How to buy for your school

Educator training and development

Deals for students and parents

AI for education

Business

Microsoft AI

Microsoft Security

Dynamics 365

Microsoft 365

Microsoft Power Platform

Microsoft Teams

Microsoft 365 Copilot

Small Business

Developer & IT

Azure

Microsoft Developer

Microsoft Learn

Support for AI marketplace apps

Microsoft Tech Community

Microsoft Marketplace

Marketplace Rewards

Visual Studio

Company

Careers

About Microsoft

Company news

Privacy at Microsoft

Investors

Diversity and inclusion

Accessibility

Sustainability

 © Microsoft 2026