Got it, VK. Here's a practical, in-depth guide you can keep on hand when a JWT "just won't work" anywhere in Azure (Entra ID, App Service/Functions, APIM, B2C/External ID, Graph, custom APIs).

JWT debugging in Azure (end-to-end)

0) What token do you have?

- **ID token** (used for sign-in): usually validated by your *client* app; contains user identity claims. Microsoft Learn +1
- **Access token** (used to call an API): validated by the *API*; carries `aud` (resource), `scp` (scopes) or `roles`. Microsoft Learn +2
  Tip: Paste into **jwt.ms** to *inspect* (never trust it for validation). Microsoft Learn +1

---

1) Fast triage checklist (90% of issues)

1. **Decode (don't trust):** note `alg`, `kid`, `iss`, `aud`, `nbf/exp`, `scp/roles`. Microsoft Learn
2. **Issuer OK?**
   - Entra ID (work/school): `https://login.microsoftonline.com/<tenant>/v2.0`
   - External ID (CIAM): `https://{tenant}.ciamlogin.com/<tenant-id>/v2.0`
   - B2C: `https://{domain}.b2clogin.com/tfp/{tenant}/{policy}/v2.0`
     Use the tenant's **OpenID config** to confirm. Microsoft Learn
3. **JWKS match:** fetch `jwks_uri` from the OIDC metadata, pick the key with matching `kid` and verify signature. Microsoft Learn +1
4. **Audience (`aud`) matches your API?** 401 + `IDX10214` = wrong `aud` (common: using appId instead of App ID URI or mixing `api://{client-id}` vs GUID). Fix API settings or middleware. Microsoft Learn +2
5. **Authorization claim present?** Your API must check `scp` (delegated) OR `roles` (app role or user role). Missing/incorrect ⇒ consent or app role assignment. Microsoft Learn +2
6. **Time claims:** `nbf/exp` + clock skew. If tokens seem "randomly" invalid, look at CAE notes below. Microsoft Learn +1

---

2) Proper validation (language-agnostic steps)

1. Read token header → `alg`, `kid`.
2. Read `iss` from payload → fetch **OIDC config** at `…/.well-known/openid-configuration`.
3. From metadata, fetch `jwks_uri` and select the **JWK** whose `kid` matches.
4. Verify signature with that key and `alg`.
5. Enforce claims: `iss`, `aud`, `nbf/exp`, `tid` (if tenant-restricted), `scp/roles`. Microsoft Learn +2

Code snippets

Node (jose)

```
import { createRemoteJWKSet, jwtVerify } from 'jose'; const issuer =
'https://login.microsoftonline.com/<tenant-id>/v2.0'; const openid = await
fetch(`${issuer}/.well-known/openid-configuration`).then(r => r.json()); const JWKS =
createRemoteJWKSet(new URL(openid.jwks_uri)); const { payload } = await
jwtVerify(token, JWKS, { issuer, audience: 'api://<your-api-app-id-or-uri>' }); //
check payload.scp or payload.roles
```

(Uses OIDC metadata → JWKS and enforces `iss/aud`.) Microsoft Learn

Python (PyJWT)

```
import jwt, requests from jwt import PyJWKClient issuer =
"https://login.microsoftonline.com/<tenant-id>/v2.0" openid = requests.get(f"
{issuer}/.well-known/openid-configuration").json() jwks_client =
PyJWKClient(openid["jwks_uri"]) signing_key =
jwks_client.get_signing_key_from_jwt(token) claims = jwt.decode( token,
signing_key.key, algorithms=["RS256","PS256"], audience="api://<your-api-app-id-or-
uri>", issuer=issuer )
```

(Exact JWKS selection by `kid`.) Microsoft Learn

.NET (Microsoft.IdentityModel.Tokens)

```
var issuer = "https://login.microsoftonline.com/<tenant-id>/v2.0"; var configMgr =
new
Microsoft.IdentityModel.Protocols.ConfigurationManager<Microsoft.IdentityModel.Protocols
( $"{issuer}/.well-known/openid-configuration", new
Microsoft.IdentityModel.Protocols.OpenIdConnect.OpenIdConnectConfigurationRetriever());
var config = await configMgr.GetConfigurationAsync(); var tokenHandler = new
System.IdentityModel.Tokens.Jwt.JwtSecurityTokenHandler(); var validationParameters =
new Microsoft.IdentityModel.Tokens.TokenValidationParameters { ValidIssuer = issuer,
ValidAudience = "api://<your-api-app-id-or-uri>", IssuerSigningKeys =
config.SigningKeys, ValidateLifetime = true, ValidateIssuerSigningKey = true };
tokenHandler.ValidateToken(token, validationParameters, out var _);
```

(Loads keys from OIDC metadata and enforces issuer/audience.) Microsoft Learn

---

3) Getting a token on demand (to reproduce issues)

- **Azure CLI (user context):**

```
az login az account get-access-token --resource https://management.azure.com/ #
or for your custom API: az account get-access-token --resource api://<your-api-
client-id>
```

(CLI returns a short-lived token; `--resource` decides the `aud`.) Microsoft Learn +1

- **PowerShell:**

```
(Get-AzAccessToken -ResourceUrl "https://graph.microsoft.com").Token
```

Microsoft Learn

- **Raw OAuth (client credentials):** POST to `/oauth2/v2.0/token` with `scope=<api-app-id>/.default`. Microsoft Learn

---

4) Service specifics

App Service / Azure Functions (EasyAuth)

- Check **Authentication** blade: issuer URL must match your token's `iss`; **Allowed token audiences** must include your API's App ID URI (e.g., `api://...` or resource URL). Mis-matches cause 401. Microsoft Learn +1

Azure API Management (APIM)

- Use the `validate-jwt` policy; set `issuer`, `audiences`, and `openid-config` / `jwks` (or cert). Example:

```
<validate-jwt header-name="Authorization" require-scheme="Bearer"> <openid-config
url="https://login.microsoftonline.com/<tenant-id>/v2.0/.well-known/openid-
configuration" /> <audiences> <audience>api://<your-api-app-id-or-uri></audience>
</audiences> </validate-jwt>
```

(You can also authorize based on claim checks.) Microsoft Learn

OBO (On-Behalf-Of) chains

If your API calls a downstream API, ensure you exchange the inbound user token for a downstream access token (OBO) and validate the *new* token when calling the second API. Microsoft Learn +1

CAE (Continuous Access Evaluation)

Tokens might be long-lived and revoked mid-lifetime; CAE/claims challenges rely on the `xms_cc` **cp1** capability and Conditional Access. This can look like "random 401s" if the client can't handle challenges. Inspect tokens/logs for CAE. JSON Web Toke... +2

---

5) Logs you should check (and how)

Entra sign-in logs (incl. non-interactive):

- Correlate failures and token type, error, CAE state. Example KQL:

```
SigninLogs | where TimeGenerated > ago(24h) | project TimeGenerated,
UserPrincipalName, AppDisplayName, ResourceDisplayName, TokenIssuerType,
AuthenticationDetails, Status.errorCode, Status.failureReason | summarize count() by
Status.errorCode, tostring(Status.failureReason) | order by count_ desc
```

(Use `SigninLogs` / `AADNonInteractiveUserSignInLogs` ; fields include a UniqueTokenIdentifier to correlate.) Microsoft Learn +1

MSAL client logs: temporarily enable **Verbose** logging (redact secrets!) to capture acquisition/refresh/claims challenges. Microsoft Learn +1

---

6) Common breakages → exact fixes

Symptom	Root cause	Fix
`IDX10214: Audience validation failed`	`aud` doesn't match what your API expects (`api://…` vs GUID; wrong App ID URI)	Align middleware/API config with the token's `aud` , or issue tokens for the correct resource. Microsoft Learn +1
`Signature validation failed`	Using wrong keyset/tenant; not rotating keys; caching stale JWKS	Always resolve `jwks_uri` from **issuer's** OIDC metadata and use `kid` match; implement key rollover. Microsoft Learn +1
401 with no `scp/roles`	Missing consent or app role assignment	Request proper scopes (`scp`) or assign app/user roles and re-consent. Microsoft Learn +1
Works in local, fails in App Service	EasyAuth `Allowed token audiences` /issuer mismatch	Set the correct **issuer** and **audience** in Authentication blade. Microsoft Learn
Random mid-session 401s	CAE revocation/claims challenge; client not CAE-capable	Add `xms_cc` capability & handle claims challenges; confirm CAE in logs. JSON Web Toke… +1
B2C tokens rejected by API	Using Entra (work) issuer/metadata against B2C token (or vice-versa)	Use the correct B2C policy issuer + metadata endpoint. Microsoft Learn

---

7) Optional claims & "why is my email missing?"

Tokens are intentionally minimal. Add optional claims (e.g., `email`, `upn`, `groups`) in **Token configuration** or app manifest; not all users have an `email` value. Don't hard-depend on non-guaranteed claims. Microsoft Learn +2

---

8) Pocket commands & tools

- **Inspect only:** `jwt.ms` (client-side decoding), `jwt.io` (careful: don't paste real prod tokens). Microsoft Learn +1
- **Grab a token (user):** `az account get-access-token --resource <audience>`; or PowerShell `Get-AzAccessToken -ResourceUrl ...`. Microsoft Learn +1
- **See OIDC metadata:** `https://login.microsoftonline.com/<tenant>/v2.0/.well-known/openid-configuration` (contains `jwks_uri`). Microsoft Learn
- **APIM gate:** `<validate-jwt>` with `openid-config` or `jwks`. Microsoft Learn

---

9) Minimal decision tree (keep near your console)

1. Is the **issuer** correct for the token type (Entra/B2C/CIAM)? If not → fix metadata/tenant. Microsoft Learn
2. Does **audience** match your API? If not → adjust API config or request token for the right resource. Microsoft Learn
3. Can you **verify signature** against issuer's JWKS (matching `kid`)? If not → update JWKS retrieval/rollover. Microsoft Learn
4. Do you have required **authorization claims** (`scp` or `roles`)? If not → consent/assign roles. Microsoft Learn +1
5. Are **time/CAE** factors revoking the session? If uncertain → check sign-in logs & CAE fields. Microsoft Learn +1