That warning in the Microsoft documentation can indeed seem confusing at first glance, but it is specifically cautioning against validating tokens that aren't intended for your API. Here's a breakdown of what it means and why it shouldn't worry you when validating JWTs for APIs that you own:

What the Warning Means:

The warning is saying that you shouldn't attempt to validate tokens issued for Microsoft services (like Graph API or other Microsoft-managed APIs) in your own code. The reason for this is:

1. Special Token Formats: Microsoft services might issue tokens that use a proprietary format (instead of the standard JWT format) or even issue encrypted tokens, which wouldn’t follow the typical JWT validation patterns.

2. Consumer Tokens (Microsoft Accounts): Tokens for Microsoft account users (also known as "consumer users," such as those with @outlook.com or @live.com addresses) might not behave the same way as tokens for Azure AD tenants (typically business or organizational accounts).

Key Point of the Warning:

- Do not try to validate or read tokens that are meant for Microsoft services (like Microsoft Graph API, Office 365, etc.) in your code, since these might not be regular JWTs and may have formats or encryption that prevent them from being processed by your application’s standard JWT validation logic.

When It's OK to Validate Tokens:

If you’ve created an API, and the tokens are intended for your API, you should absolutely validate those tokens. In fact, this is a crucial step in securing your API. Here's why:

1. Audience Claim (aud): Every JWT has an “aud” (audience) claim, which indicates which API the token is intended for. When your API receives a token, it should validate that the “aud” matches your API's identifier.

2. Azure AD Issued Tokens: When Azure AD issues a token for your API (based on client credentials or another flow), that token will be in JWT format, and you can validate it using standard JWT validation logic, including:

- Checking the signature using the public keys from the OIDC metadata.

- Ensuring the “aud” claim matches your API.

- Validating other claims like “exp” (expiration), “iss” (issuer), etc.

To Summarize:

- DO validate tokens issued for your own APIs. If you own the API and Azure AD (Microsoft Entra) issues a JWT for it, you can and should validate that token in your API using the OIDC metadata signing keys.

- DON'T attempt to validate tokens intended for Microsoft-managed services (like Graph API or other Microsoft services) because:

- They might not follow standard JWT formats.

- They might be encrypted.

- They are not intended for your API and shouldn't be processed by your code.

Practical Example:

- If your API has an “aud” value of “api://my-app”, and Azure AD issues a token with that “aud” claim, you should validate it by using the public keys from Azure AD's “.well-known” metadata endpoint.

- However, if you somehow intercepted a token with an “aud” claim for Microsoft Graph (“https://graph.microsoft.com”), the warning is telling you not to attempt to validate that token because it's meant for a Microsoft service, not your API.

Let me know if you need further clarification!