JWT Authentication – Invalid Client Errors

For individualized technical support, consider our App Orchard developer program.

When developing Backend OAuth 2.0 applications, the issue developers most often run into is a HTTP 400 status code with an invalid_client error, which looks like this:

```
{
  "error": "invalid_client",
  "error_description": null
}
```

This error can be caused by a few different underlying issues. A few of the most common are:

Issues with the App Listing

1. Ensure the backend systems consumer type and OAuth 2.0 boxes are checked on the app listing.
2. Ensure you've uploaded a public key to your environment.
3. If creating a user/patient facing app that uses JWTs, make sure that "Require Refresh Tokens" is checked, so that you can then upload your public key.

Issues with the Request Body

1. Ensure in the body of the POST, "client_assertion" is used and NOT "client-assertion".
2. Ensure you aren't double-encoding the "client_assertion_type" parameter's value of "urn:ietf:params:oauth:client-assertion-type:jwt-bearer".

Issues with the Token Request Formatting

1. Ensure the request URL ends in /token and NOT /authorize.
2. Ensure HTTP POST verb is used, NOT GET.
3. Ensure the Content-Type of "application/x-www-form-urlencoded" is used.
4. Ensure the content is in the body of the request, NOT the URL.

Issues with the JWT

1. Ensure the iat and nbf claims are NOT in the future.
2. Ensure the exp claim is in the future.
3. Ensure the exp is no more than 5 minutes in the future.
4. Ensure the exp is not more than 5 minutes after the iat and nbf claims.
5. Ensure the app's client ID is in both the iss and sub claims.

6. Ensure the correct client ID is used – for the Sandbox, non-prod client ID should be used.
7. Ensure the jti is no longer than 151 characters and contains a unique ID.
8. Ensure the JWT was not expired at the time of use.

Issues with the Signature

1. Ensure the private key used to sign the JWT corresponds to the public key registered with the sandbox or Epic customer.
    a. If you re-upload your public key, you need to wait for the changes to sync. This might take up to 15 minutes to sync with the Sandbox, and 12 hours to sync at an Epic organization's Epic instance.
    b. Note the “Sandbox Public Key” on your app is for use against the Epic sandbox. When you are ready to integrate with an Epic customer, you will be asked to upload non-PRD and PRD public keys for each customer.
    c. If your app is cloud-hosted, it is OK to re-use public keys across customers. This assumes you can protect a single copy of the key and are using a different key for non-PRD and PRD.
2. Ensure the signature algorithm indicated is the algorithm your code cryptographically signs the JWT with.
3. Ensure the signature algorithm used is supported. RS256 and RS384 are supported. RS384 is preferred.