Client Integration with NVIDIA Identity Federation System

Integration Guide

Document History

Version	Date	Description of Change
01	Sep 14, 2020	First document edition
02	Sep 21, 2020	Marked Git as internal access only and removed internal system names
03	Sep 21, 2020	Update TTL for Access Token
04	Sep 23, 2020	Additional details added to flows
05	Oct 22, 2020	Fixed Typo

Table of Contents

Chapter 1. Introduction

This document specifies the principles and interfaces through which a client application can interact with the NVIDIA Identity Federation System. Partners and Teams who are building applications that require access to identities can use this document to understand and plan the necessary software integration work.

The NVIDIA Identity Federation System supports 3 flows

- Implicit
- Device Authorization
- Authorization Code

Requirements

To access the system, you require a “client id” provided by NVIDIA on request.

To obtain a client id please contact your Technical Account Manager (TAM) or NVIDIA representative.

Your TAM/NVIDIA representative will ask for the type of flows you want to support in your client and will request the following:

- Client Name
- 1 or many supported redirect HTTP URLs, ensure these redirect URLs use HTTPS. The redirect URL are used to return the user the resulting page with the desired data.
- If you want to use Authorization Code flow you will also be supplied with a “client secret”
- If you need access to a long lived “Client Token”
- Client Type (Enterprise or Gaming) This will associate your client with the appropriate flow with UI element and User consent asked.

Background

To help with the process please familiarizes yourself with OAuth 2.0 and OpenID Connect (OIDC)

- OAuth 2.0 Authorization Framework
- OpenID Connect Implicit Client Implementer's Guide 1.0
- OAuth 2.0 Device Authorization Grant

Chapter 2. OpenID Provider Metadata

This is a well-known URI Discovery Mechanism for the Provider Configuration URI and is defined in OpenID Connect.

Inbound clients that interact with NVIDIA Identity Federation System through OpenID Connect can programmatically obtain the URLs of the endpoints exposed and the corresponding supported options. This information is also known as OpenID-configuration.

It's suggested that clients use this as this will ensure compatibility with clients if there any changes the client will update the URLs.

URL: https://login.nvidia.com/.well-known/openid-configuration

Type: GET

Response:


```
{
  "issuer": "https://login.nvidia.com",
  "authorization_endpoint": "/authorize",
  "token_endpoint": "/token",
  "userinfo_endpoint": "/userinfo",
  "jwks_uri": "/.well-known/jwks.json",
  "scopes_supported": [
    "openid",
    "consent",
    "consent_u",
    "email",
    "profile",
    "phone",
    "address",
    "authz",
    "birthdate"
  ],
  "response_types_supported": [
    "code",
    "id_token",
    "id_token token",
    "code id_token",
    "code token",
    "code id_token token"
  ],
  "response_modes_supported": [
    "query",
    "fragment"
  ],
  "subject_types_supported": [
    "public"
  ],
  "id_token_signing_alg_values_supported": [
    "ES256"
  ]
}
```

Chapter 3. Implicit Flow

The Implicit Flow is intended for applications where the confidentiality of the client secret can't be guaranteed. In this flow, the client does not make a request to the /token endpoint, but instead receives the access token directly from the /authorize endpoint. The client must be capable of interacting with the resource owner's user agent and capable of receiving incoming requests (through redirection) from the NVIDIA identity federation system.

The Implicit Flow follows the following steps:

1. Client prepares an Authentication Request containing the desired request parameters.
2. Client sends the request to the Authorization Server (NVIDIA Identity Federation System).
3. Authorization Server Authenticates the End-User.
4. Authorization Server obtains End-User Consent/Authorization.
5. Authorization Server sends the End-User back to the Client with an Identity Token and, if requested, an Access Token.
6. Client validates the ID token and retrieves the End-User's Subject Identifier.

Example URL:

`https://login.nvidia.com/authorize?response_type=id_token+token&client_id=<replace>&redirect_uri=<replace>&scope=openid&device_id=<replace>&nonce=<replace>`

Type: `GET/POST`

Parameters:

Name	Description	Example Data
response_type	The Response Type informs NVIDIA Identity Federation System of the desired authorization processing flow. Options: id_token, toke, id_token (if the client wants to receive only the id token) id_token token (if the client wants to also receive the access token)	`id_token token`

nonce	The nonce is generated by the application and included in the Identity Token response. This allows applications to correlate the Identity Token response with the initial authentication request.	`02a968b2-a4b6-45f3-94d2-d8d4c0a06930`
client_id	The Client Id is used to identify the client application.	`oxgwb2c34fpv0nHJVEhFZuqRoysIsCbZ9AxmT4Hm6t5`
device_id	The User's Device Id you want to associate with this flow.	`fe6e0058-827b-11ea-bc55-0242ac130003`
redirect_uri	The URL after a successful login that the user will be returned to. (This URL must be registered with the NVIDIA Identity Federation System)	`https://yourclientpage.com/redirect.html`
ui_locales	A space separated list of RFC 5646 language tags, sorted in order of preference	`en-US`
scope	A space separated list of scopes needed. Note that you need to add openid as the first scope in the list as a minimum. The scopes will add data to your Identity Token.	`openid consent`
response_mode	Response mode should be used to change the default response mode for the flow Options: query, fragment Default is fragment	`query`
state	The client may need it to be able to go back where it was. This is typical for web pages when the login is launched in the same tab.	`mystate`
prompt	Instructions for how to handle re-authentication. Space delimited list with one or more of the following values: none, login, consent, select_account (if supported by the IdP else will be ignored)	`select_account`

Response:

A successful response will be a 302 Redirect with the following parameters in the fragments or in the query string of the Redirect URL you provided on the Authorization Request.

Name	Description	Example Data
token_type	When sending tokens in the "Authorization" request header to other servers this must be present. Note that it's always "Bearer".	`Bearer`
expires_in	In seconds how long till the "Access Token" expires	`3600`
access_token	The Access Token is to be used to access information from NVIDIA Identity Federation System.	`YsGsDkMi5I6qYsfBGDHSXVg4G9lTPXX3x8CiU2ASFS12JB0lNhBn1zgcEILHUMZtd8BdjmtCFF0zenMM53fqlg`
id_token	The Identity Token contains information about a user and their authentication status. It can be used by the client both for authentication and as a store of information about the user.	`eyJraWQiOiJkZXYta2V5LWlkIiwiYWxnIjoiRVMyNTYifQ.eyJzdWIiOiJSYU5mYk8wNGllekota0ZxOUh3ZDVzUGZ2ZUdCbllUWHZEalF2aUFkZFFrIiwiaWRwX25hbWUiOiJOVi1BRCIsImlkcF9pZCI6IjF6VThTQ0hZVFIwNHRnN1VlZGt0TFVIaU1OTlRBdkpnWk4wWThuY3dtRzgiLCJpc3MiOiJodHRwczpcL1wvc3RnLmxvZ2luLm52aWRpYS5jb20iLCJleHRlcm5hbF9pZCI6IjdkMzM2OGY5LTZjNmYtNDVjYS04Mjg4LWJmYWQ2Yjk4NGY0YyIsInByZWZlcnJlZF91c2VybmFtZSI6IkNhbSBNb29yZSIsImNvbnNlbnQiOnt9LCJub25jZSI6Im15bm9uY2UiLCJhdWQiOlsiZENfdERajFlaGxhZEp1bzh2MVE3d0hGSmRUaldSZExEdl9qaEtHX2FHZyIsInM6cnRRMkY4UGtwd2tKdWd6RjBrT3JUZXNuLUZEQjNtSXduVWJRamU0QUdENCJdLCJ1cGRhdGVkX2F0IjoxNjAwNDc4MTMwLCJhenAiOiJkQ190RENqMWVobGFkSnVvOHYxUTd3SEZKZFRqV1JkTER2X2poS0dfYUdnIiwiZXhwIjoxNjAwNDkyNTMwLCJpYXQiOjE2MDA0NzgxMzB9.ofohlzl9de9GT4RRdx0KYzjI8Psn6bFW7RPHUVEUgjGp3SdApeWYc4XjDYfo3SU7_mEb62zSm2XcDdYoiByICA`

Chapter 4. Device Authorization Flow

The OAuth 2.0 device authorization grant is designed for clients/applications that lack a browser to perform a user-agent-based authorization or are input constrained to the extent that requiring the user to input text in order to authenticate during the authorization flow is impractical. It enables OAuth clients on such devices (like smart TVs, media consoles, command line clients) to obtain user authorization to access protected resources by using a user agent on a separate device.

URL: https://login.nvidia.com/device/authorize

Type: POST

Headers: Content-Type: application/x-www-form-urlencoded

Body Data:

Name	Description	Example Data
client_id	The Client Id is used to identify the client.	oxgwb2X34fpv0MHXVEhFZuqRoysIsCbZ9AxmT4Hm6t5
display_name	The Name of the device you want to display to the user.	My SHIELD TV
device_id	The User's Device Id you want to associate with this flow.	fe6e0058-827b-11ea-bc55-0242ac130003
scope	A space separated list of scopes needed. Note that you need to add openid as a minimum as we implement OpenID Connect authentication as an extension to the OAuth 2.0 authorization process. The scopes will add data to your identity token.	openid consent

Response:


```
{
"user_code": "37063526",
"device_code":
"eyJraWQiOiJkZXYta2V5LWlkIiwiYWxnIjoiRVMyNTYifQ.eyJhdWQiOiJzdGFyZmxlZXQiLCJpc3MiOiJzdGFyZmxlZXQiLCJleHAiOjE1ODc0ODcyNjAsImlhdCI6MTU4NzQ4MDA2MCwianRpIjoiMWU3ZWRiNzAtODNkZS0xMWVhLTlhZDgtODE3ZjdhYjVhOWJkIn0.Bb3XOjXT-IcTsX72EOTI06lDAknquwtU08bXg2eP0KmhorcmnOuauEPEVk3jGq7AdUAS6eAtsFqD_5J0yn4aZQ",
"verification_uri": "https://static-login.nvidia.com/service/gfn/pin",
"verification_uri_complete": "https://static-login.nvidia.com/service/gfn/pin?user_code=37063526",
"expires_in": 900,
```

```
"interval": 5
}
```

The client should immediately poll the Token Endpoint every "X" in seconds defined in "interval" until it receives a HTTP Status response of 200 with the Access/ID tokens.

While polling the client will receive a HTTP Status response of 400 until the user enters the correct user code on the verification page defined in "verification_uri" and completes the login flow.

If the time of polling elapses past the defined duration in "expires_in" an error should be presented to the user to re-authorize the device.

Chapter 5. Authorization Code Flow

The authorization code flow offers a few benefits over the other grant types. When the user authorizes the application, they are redirected back to the client with a temporary code in the URL. The client exchanges that code for the access token with its client id and secret. Your client must be server-side application because during this exchange, you also pass along your secret, which must always be kept secure.

As the request for the access token is authenticated with the secret, this reduces the risk of an attacker intercepting the authorization code and using it themselves. This also means the access token is never visible to the user, so it is the most secure way to pass the token back to the application, reducing the risk of the token leaking to someone else.

Note: Ensure you have requested that your client is setup to support Authorization Code Flow.

Example URL:

`https://login.nvidia.com/authorize?response_type=code&client_id=<replace>&redirect_uri=<replace>&scope=openid&device_id=<replace>&nonce=<replace>`

Type: `GET/POST`

Headers: `Content-Type: application/x-www-form-urlencoded`

Parameters:

Name	Description	Example Data
response_type	The Response Type informs NVIDIA Identity Federation System of the desired authorization processing flow. Options: code	`code`
nonce	The nonce is generated by the application and included in the ID Token response. This allows applications to correlate the ID Token response with the initial authentication request.	`02a968b2-a4b6-45f3-94d2-d8d4c0a06930`
client_id	The Client Id is used to identify the client application.	`oxgwb2c34fpv0nHJVEhFZuqRoysIsCbZ9AxmT4Hm6t5`

device_id	The User's Device Id you want to associate with this flow.	`fe6e0058-827b-11ea-bc55-0242ac130003`
redirect_uri	The URL after a successful login that the user will be returned to. (This URL must be registered with the NVIDIA Identity Federation System)	`https://redirect.html`
ui_locales	A space separated list of RFC 5646 language tags, sorted in order of preference	`en-US`
scope	A space separated list of scopes needed. Note that you need to add openid as a minimum as we implement OIDC. The scopes will add data to your Identity Token.	`openid consent`
prompt	Instructions for how to handle re-authentication. Space delimited list with one or more of the Options: none, login, consent, select_account	`select_account`
state	The client may need it to be able to go back where it was. This is typical for web pages when the login is launched in the same tab.	`mystate`

Response:

A successful response will be a 302 Redirect with the following parameters in the fragments or query string of the Redirect URL you provided on the Authorization Request.

Name	Description	Example Data
token_type	When sending tokens in the "Authorization" request header to other servers this must be present. Note that it's always "Bearer".	`Bearer`
expires_in	In seconds how long till the "Access Token" expires	`3600`
code	The Authorization code	`YsGsDkMi5I6qYsfBGDHSXVg4G9lTPXX3x8CiU2ASFS12JB0lNhBn1zgcEILHUMZtd8BdjmtCFF0zenMM53fqlg`

Chapter 6. Renewing Tokens

Clients that have obtained an Authorization code, Refresh Token Client Token or a Device Code can retrieve a new Identity Token and a new Access Token by calling the token endpoint.

6.1 Using Authorization Code

URL: https://login.nvidia.com/token

Type: POST

Headers:

```
Content-Type: application/x-www-form-urlencoded
Authorization: Basic Base64 Encode(<clientId:clientSecrect>)
```

Body Data:

Name	Description	Example Data
grant_type	authorization_code	authorization_code
code	The authorization code.	oxgwb2X34fpv0nHJVEhFZuq RoysIDCbZ9AxmT4Hm6t5
redirect_uri	The redirect URI that was used in the authorization code request.	https://myredirect.com/redirect.html

Response:

```
{
  "id_token": "eyJhbGci...vKMzqg",
  "refresh_token": "8xLOxBtZp8",
  "access_token": "SlAV32hkKG",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

6.2 Using a Client Token

URL: https://login.nvidia.com/token

Type: POST

Headers: Content-Type: application/x-www-form-urlencoded

Body Data:

Name	Description	Example Data
grant_type	Grants ("methods") are ways a client can acquire an access token. To use a Client Token it must be in the form of: urn:ietf:params:oauth:grant-type:client_token	urn:ietf:params:oauth:grant-type:client_token
client_id	The Client Id is used to identify the client application.	oxgwb2X34fpv0nHJVEhFZuqRoysIDCbZ9AxmT4Hm6t5
client_token	The long lived Client Token obtained using the short lived Access Token.	eyJraWQiOiJkZXYta2V5LWlkIiwiYWxnIjoiRVMyNTYifQ.eyJhdWQiOiJzdGFyZmxlZXQiLCJpc3MiOiJzdGFyZmxlZXQiLCJleHAiOjE1ODc0NzkwMjYsImlhdCI6MTU4NzQ3MTgyNiwianRpIjoiZjJjYWM5NzAtODNjYS0xMWVhLTlhZDgtODE3ZjdhYjVhOWJkIn0.oRYqvrTY7xyjiRHv3ZPjBLydXNDSUOI8s0EcH05ax_lrF4jhG2MccLKw_Z3fOBTwEwDDw5NeXzCq7FgM0zRadw
sub	The sub claim that identifies the logged in user. Part of id_token.	c-TPu99apqlySzhmTejcunXWQ-1SflLvwvn-c8CPmCg

Response:


```
{
"access_token":
"eyJraWQiOiJkZXYta2V5LWlkIiwiYWxnIjoiRVMyNTYifQ.eyJhdWQiOiJzdGFyZmxlZXQiLCJpc3MiOiJzdGFyZmxlZXQiLCJleHAiOjE1ODc0ODQxNTYsImlhdCI6MTU4NzQ3Njk1NiwianRpIjoiZTQ5NzFjZDAtODNkNi0xMWVhLTg0YzktODU1ZWZiYTc3OGQxIn0.SbZwUYyEXEDNyGebUszfB7mRdxAAIFPt_PtQ2mrnVQRA_gBcFNQhfCurAxg-9f6NwMegykEMtvAWB5uqmTMcIA",
"token_type": "Bearer",
"expires_in": 3600,
"client_token":
"eyJraWQiOiJkZXYta2V5LWlkIiwiYWxnIjoiRVMyNTYifQ.eyJhdWQiOiJzdGFyZmxlZXQiLCJpc3MiOiJzdGFyZmxlZXQiLCJleHAiOjE1ODcxNTU2NzcsImlhdCI6MTU4NzE0ODQ3NywianRpIjoiMTdkYzk3YTAtODBkYS0xMWVhLThkZDEtYTk0NTkxMjNjNTVjIn0.0Dn3joVYAh_INmawmoUkP1CtmVpTwv2olSEFZ7smiziGnGOtt2ZKOE2jcpUiXrHirZyCdLjE8v1x_VnbniEL9g",
"id_token":
"eyJraWQiOiJkZXYta2V5LWlkIiwiYWxnIjoiRVMyNTYifQ.eyJhdWQiOiJ6R3hqUW45WlhycFc4MHdBd1VYakg2bG00TWoyeURYNkkyN0F0TGxPWmxNIiwic3ViIjoiLUg4NF9saVJSZ2QydEg0bUJHZ2ZaWGZLbWN4SDhTbE1VXzdoUHdMSE9WWSIsImlkcF9uYW1lIjoiTlZJRElBLUoiLCJpZHBfaWQiOiJuVjBYWpldlNVMjlFNG51YUZqWVhhdWVyblN2OHZVYkJQeUJOOU1MU1prIiwiaXNzIjoibG9naW4ubnZkaWEuY29tIiwiZXh0ZXJuYWxfaWQiOiIyNTk2NDkwOTMzMzgzOTk2MTAiLCJleHAiOjE1ODc0ODQxNTYsImlhdCI6MTU4NzQ3Njk1Nn0.0H3aopi0rsvKhoOZ6iU1d-Shnzm60cYHlhRi6japG9FdwGjbMgAhf8tEhHUeYHX4d7uEqxIsnVe9HJ47NRkMJA"
}
```


6.3 Using a Device Code

URL: https://login.nvidia.com/token

Type: POST

Headers: Content-Type: application/x-www-form-urlencoded

Body Data:

Name	Description	Example Data
grant_type	Grants (“methods”) are ways a client can acquire an access token. To use a Device Code it must be in the form of: urn:ietf:params:oauth:grant-type:device_code	urn:ietf:params:oauth:grant-type:device_code
client_id	The Client Id is used to identify the client application and is defined by NVIDIA Identity Federation System.	oxgwb2X34fpv0nHJVEhFZuqRoysIsCbZ9AxmT4Hm6t5
device_code	The Device Code obtained during the Device Authorization Process.	eyJraWQiOiJkZXYta2V5LWlkIiwiYWxnIjoiRVMyNTYifQ.eyJhdWQiOiJzdGFyZmxlZXQiLCJpc3MiOiJzdGFyZmxlZXQiLCJleHAiOjE

	1ODc0ODgzODIsImlh dCI6MTU4NzQ4MTE4M iwianRpIjoiYmI0OD FkYzAtODNlM C0xMWVhLTg0YzktOD U1ZWZiYTc3OGQxIn0 .o4jMzuQtvg5sbr8X BoceVrcqsMI6- 4G75nCvo7g- PI6ujyI5mCUOF4BNt vcjEkWLDbFjbFj5HB in-oMqexKDRw

Response:


```
{
"access_token":
"eyJraWQiOiJkZXYta2V5LWlkIiwiYWxnIjoiRVMyNTYifQ.eyJhdWQiOiJzdGFyZmxlZXQiLCJpc3MiOiJzdGFyZmxlZXQiLCJleHAiOjE1ODc0ODQxNTYsImlhdCI6MTU4NzQ3Njk1NiwianRpIjoiZTQ5NzFjZDAtODNkNi0xMWVhLTg0YzktODU1ZWZiYTc3OGQxIn0.SbZwUYyEXEDNyGebUszfB7mRdxAAIFPt_PtQ2mrnVQRA_gBcFNQhfCurAxg-
9f6NwMegykEMtvAWB5uqmTMcIA",
"token_type": "Bearer",
"expires_in": 3600,
"id_token":
"eyJraWQiOiJkZXYta2V5LWlkIiwiYWxnIjoiRVMyNTYifQ.eyJhdWQiOiJ6R3hqUW45WlhycFc4MHdBd1VYakg2bG00TWoyeURYNkkyN0F0TGxPWmxNIiwic3ViIjoiLUg4NF9saVJSZ2QydEg0bUJHZ2ZaWGZLbWN4SDhTbE1VXzdoUHdMSE9WWSIsImlkcF9uYW1lIjoiTlZJRElBLUoiLCJpZHBfaWQiOiJuVjJBYWpldlNVMjlFNG51YUZqWVhhdWVyblN2OHZVYkJQeUJOOU1MU1prIiwiaXNzIjoibG9naW4ubnZkaWEuY29tIiwiZXh0ZXJuYWxfaWQiOiIyNTk2NDkwOTMzMzgzOTk2MTAiLCJleHAiOjE1ODc0ODQxNTYsImlhdCI6MTU4NzQ3Njk1Nn0.0H3aopi0rsvKhoOZ6iU1d-
Shnzm60cYHlhRi6japG9FdwGjbMgAhf8tEhHUeYHX4d7uEqxIsnVe9HJ47NRkMJA"
}
```

6.4 Using a Refresh Token

URL: https://login.nvidia.com/token

Type: POST

Headers: Content-Type: application/x-www-form-urlencoded

Body Data:

Name	Description	Example Data
grant_type	refresh_token	refresh_token
refresh_token	The refresh token	oxgwb2X34fpv0nHJVEhFZu qRoysIDCbZ9AxmT4Hm6t5

Response:

```
{
  "id_token": "eyJhbGci...vKMzqg",
  "refresh_token": "8xLOxBtZp8",
  "access_token": "SlAV32hkKG",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

Chapter 7. Token Deletion

If the client chooses to logout the user of the client application it can invalidate the Token via the SCIM endpoint.

Such tokens include:

- Access Tokens
- Refresh Tokens
- Client Tokens

A. If the client wishes to revoke the provided Access Token. The client will call solely with the Access Token provided for the operation. The client triggers this operation by providing no query parameters to the DELETE request.
B. If the client wishes to revoke all tokens related to the provided Access Token. The client triggers this operation by providing the query parameter level=client to the DELETE request.

To perform the deletion the following scope is needed:

- tk_client - Ability to delete token at client level

URL: `https://login.nvidia.com/assets/v2/Tokens`

Type: `DELETE`

Header: `Authorization: Bearer <Access Token>`

Response: 200 HTTP Status Code

Parameters:

Name	Description	Example Data
level	Defines what level the tokens will be deleted at. Options: client, service	`client`

Response: `200 HTTP Status Code`

Chapter 8. User Info

The user info call allows clients to access additional information about the user that has been provided by the users IdP.

Depending on the scopes requested in the Access Token you get various attributes in the response. e.g. consent, email, birthdate

See the chapter on Scopes for all available scopes.

URL: https://login.nvidia.com/userinfo

Type: GET

Header: Authorization: Bearer <Access Token>

Response:

```
{
"sub": "eM8ReDJKai80Hg9aA9McZ0ZQeLXUaJ9rV12ev0GlP1M",
"idp_name": "NVIDIA-J",
"idp_id": "nV2AajevSU29E4nuaFjYXauernSv8vUbBPyBN9MLSZk",
"external_id": "177190938526875864",
"preferred_username": "cam.moore",
"consent":{
        "trackBehavioralData": "false",
        "trackTechnicalData": "false"
        }
}
```

Chapter 9. Scopes

A client application may request the following scopes during a login flow. Each scope will give the client access to data about the user if available. Not all claims are guaranteed due to what information the IdP can provide.

These scopes provide access to attributes/claims in the "ID Token" and can give the "Access Token" the ability to retrieve attributes /Userinfo.

Claim	Scopes	Description
sub	`openid`	Subject identifier
updated_at	`openid`	Time at which the user information was last updated
idp_name	`openid`	Friendly name of the IdP that authenticated the user.
idp_id	`openid`	Identifier of the IdP that authenticated the user.
external_id	`openid`	Subject identifier assigned by the IdP that authenticated the user
preferred_username	`openid`	Shorthand name by which the user wishes to be referred to at the client
consent	`consent`, `consent_u`	Data of the consent the user has provided
email	`email`	Preferred email address
email_verified	`email`	Email verification status
name	`profile`	User full name
given_name	`profile`	User first name
family_name	`profile`	User last name
middle_name	`profile`	User middle name
nickname	`profile`	User casual name
zoneinfo	`profile`	User time zone
locale	`profile`	User locale represented as a RFC 5646 language tag
birthdate	`birthdate`	User date of birth

phone_number	`phone`	Preferred phone number in E.164 format
phone_number_verified	`phone`	Phone number verification status
address	`address`	User preferred postal address
groups	`authz`	Groups the user is in
entitlements	`authz`	Any entitlements associated with the user
roles	`authz`	Roles the user has
delete	`tk_client`	Ability to delete token at client level
delete	`tk_service`	Ability to delete token at service level

Chapter 10. Token Types

10.1 Identity Token (ID Token)

Identity tokens are JSON Web Tokens (JWT) that conform to section 2 of OpenID Connect Core 1.0.

In your client application follow these instructions to validate the ID token. Additional claims can be requested using Scopes.

The following claims are included in the Identity Token:

Claim	Description
iss	Token Issuer = login.nvdia.com
aud	The client_id used to request the token, followed by the corresponding service identifier. The service identifier is denoted with the s: prefix. For example, ["8362368238bab4d8", "s:c5f5fb387ded6062"]
azp	Authorized party - client_id used to request the token, first item in aud claim
exp	Expiration time as per RFC 7519
iat	Issued at time as per RFC 7519
sub	Subject Identifier – This is unique to each user at NVIDIA
updated_at	Time at which the user information was last updated. Expressed as UNIX timestamp in seconds, exactly like iat and exp
idp_name	Friendly name of the IdP that authenticated the user
idp_id	The Unique Identifier of the IdP that authenticated the user
external_id	Subject identifier assigned by the IdP that authenticated the user
preferred_username	Shorthand name by which the user wishes to be referred to at the client. This is the displayName field in the user profile

10.2 Access Token

An Access Token is a client opaque token as per OAuth / OIDC spec, which allows a client to access NVIDIA Identity Federation System APIs on behalf of a specific user. It is obtained by the client during a login flow.

Access tokens are short lived with a TTL of 1 hour – this can be configured per client on request and approval.

They can be refresh using the tokens to the endpoint /token

On a 401 Response from NVIDIA Identity Federation System the client should discard the Access Token and not re-request with the same token.

10.3 Refresh Token

A Refresh Token is a client opaque token as per OAuth / OIDC spec, which allows a client to renew a specific access token once such an access token has expired. Like the access token, the refresh token is bound to a specific user and (in addition) to a specific client.

Refresh tokens are only assigned to clients if they have a client secret (as that secret is required in the renewal /refresh flow per OAuth spec). The refresh token is obtained by the client during a login flow.

10.4 Client Token

The Client Token is typically used by a “fat client”. A fat client (downloaded to the PC or device with a need for long lived state) that cannot keep a "secret" as state is persisted offline. So, it does not have a refresh token.

The client received an Access Token when the user logged in, but that token is short lived, and the client still needs access the latest User Info or other supported systems after the interaction has been completed.

Clients that have obtained an Access Token can retrieve a Client Token by calling the client token endpoint.

Once obtained a Client Token can be used to get a new "Access Token" and "ID Token"

This operation is only supported for clients that are configured for the use of client tokens.

URL: `https://login.nvidia.com/client_token`

Type: `GET`

Headers: `Authorization: "Bearer <Access Token>"`

Response:


```
{
"expires_in":7776000,
"client_token":"eyJraWQiOiJkZXYta2V5LWlkIiwiYWxnIjoiRVMyNTYifQ.eyJhdWQiOiJzdGFyZmxlZXQiLCJpc3MiOiJzdGFyZmxlZXQiLCJleHAiOjE1ODc0NzkwMjYsImlhdCI6MTU4NzQ3MTgyNiwianRpIjoiZjJjYWM5NzAtODNjYS0xMWVhLTlhZDgtODE3ZjdhYjVhOWJkIn0.oRYqvrTY7xyjiRHv3ZPjBLydXNDSUOI8s0EcH05ax_lrF4jhG2MccLKw_Z3fOBTwEwDDw5NeXzCq7FgM0zRadw"
}
```

Chapter 11. Error Codes

Below is listed is the following errors that can be encountered during the OIDC or Device Authorization. These errors will be returned on the redirect_uri as part of the query string or fragments during the login flow.

If it's a Restful API Request, it will be returned as JSON along with the matching HTTP status code.

Authorization Flow Error:

```
https://request_uri?error_description=user_code+not+found&error=invalid_user_code
```

API Response Error:

```
{
  "error": "invalid_request",
  "error_description": "Authorization header missed"
}
```

Possible Errors:

error	error_description
invalid_user_code	The User Code has been entered is either invalid, expired, not found, or used.
invalid_grant	The refresh token is invalid, expired, revoked, or does not match the redirection URI used in the authorization request, or was issued to another client.
interaction_required	User interaction required.
login_required	Cannot completed login without displaying a user interface for End-User authentication.
account_selection_required	Selected account can't be obtained.
consent_required	The Authorization Server requires User consent.
invalid_request_uri	The request_uri is invalid or contains invalid data.
invalid_request_object	The request parameter contains an invalid Request Object.
request_not_supported	The operation does not support use of the request parameter.
request_uri_not_supported	The operation does not support use of the request_uri parameter.,
registration_not_supported	The operation does not support use of the registration parameter.
authorization_pending	The authorization request is still pending.

slow_down	The authorization request is still pending, and polling should continue, but the interval MUST be increased by at least 5 seconds for this and all subsequent requests.
expired_token	The token has expired.
access_denied	The end-user denied the authorization request.
unauthorized_client	The client is not authorized to request an authorization code using this method.
invalid_request	The request is missing a required parameter, includes an invalid parameter value, or is otherwise malformed.
invalid_scope	The requested scope is invalid, unknown, or malformed.
server_error	The authorization server encountered an unexpected condition which prevented it from fulfilling the request.
unsupported_response_type	The authorization server does not support obtaining an authorization code using this method.
temporarily_unavailable	The authorization server is currently unable to handle the request due to a temporary overloading or maintenance of the server.
invalid_client	No client credentials found.
age_restricted	The authorization server denied the request because the user does not meet the minimum age requirements for the country from which they are connected.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA, the NVIDIA logo, GeForce, and GeForce NOW are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051

http://www.nvidia.com