Bhutan NDI Technical Documentation

1. Introduction

This document describes the Bhutan NDI services and the API that can be consumed by the Client Application. The API is organized around REST, accepts JSON request body, returns JSON-encoded responses and uses standard HTTP response codes. The NDI Client Services described here are intended for the development and testing of the Client Application during the integration process. **The client MUST consult with the Bhutan NDI team to take the integration to production.**

2. Authentication Service

Bhutan NDI API is secured by OAuth 2.0 authentication provided by AWS Cognito. The authentication service exposes an API to generate a JWT access token which needs to be passed as Authorization header. All Bhutan NDI API must be made over HTTPS and if an access token is invalid, expired or not provided, the response will be a status code of 401, i.e., Unauthorized Access.

Swagger URL: https://staging.bhutanndi.com/authentication/swagger

```
curl -X 'POST' \
  'https://staging.bhutanndi.com/authentication/v1/authenticate' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "client_id":"<CLIENT_ID>",
    "client_secret":"<CLIENT_SECRET>",
    "grant_type":"client_credentials"
}'
```

3. Verifier Service

The verifier service exposes API to create proof requests that can be consumed by the Client Application to request for user data in a more secure manner by enhancing user consent and privacy using Bhutan NDI Wallet. The proof request can be customized by the Client Application based on business requirements by using the attributes from the available Schemas. Client Applications need to subscribe to webhook or NATS using *threadId* to consume the proof presentation result shared by the user from Bhutan NDI Wallet.

The attributes can be picked from the available Schema to create a proof request to get data from the issued Verifiable Credentials or use custom attributes along with the required flag to allow self attested data (Self attested attributes must have schema_name in the restrictions)

Please visit the following swagger page for API details:

Swagger URL: https://demo-client.bhutanndi.com/verifier/swagger

Proof Request

The Client Application can use Verifier Service to create a Proof Request as per the business requirement and use QR Code or Deep Link for using Bhutan NDI Wallet. A generic Proof Request flow can be found here.

Example: Creating Proof Request to get ID Number and Full Name from Foundational ID

Request:

```
curl -X 'POST' \
  'https://demo-client.bhutanndi.com/verifier/v1/proof-request' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <ACCESS_TOKEN> ' \
  -H 'Content-Type: application/json' \
  -d '{
  "proofName": "Verify Foundational ID",
  "proofAttributes": [
    {
      "name": "ID Number",
      "restrictions": [
        {
          "schema_name": "https://dev-schema.ngotag.com/schemas/c7952a0a-e9b5-4a4b-a714-1e5d0a1ae076"
        }
      ]
    },
    {
      "name": "Full Name",
      "restrictions": [
        {
          "schema_name": "https://dev-schema.ngotag.com/schemas/c7952a0a-e9b5-4a4b-a714-1e5d0a1ae076"
        }
      ]
    }
  ]
}
'
```

Response:

```
{
  "statusCode": 201,
  "message": "Proof URL created successfully",
  "data": {
    "proofRequestName": "Verify Foundational ID",
    "proofRequestThreadId": "d5356253-9798-4082-a46d-8358a2bb173d",
    "deepLinkURL":
"bhutanndidemo://data?url=https://stage-demo-shortening-url.s3.ap-southeast-1.amazonaws.com/default/55
7182cd-762b-4d3f-af46-79666e77bea4",
    "proofRequestURL":
"https://stage-demo-shortening-url.s3.ap-southeast-1.amazonaws.com/default/557182cd-762b-4d3f-af46-796
66e77bea4"
  }
}
```

- proofRequestURL: Used for generating QR Code
- deepLinkURL: Used for displaying Deep Link
- proofRequestThreadId: Used for subscribing to NATS or Webhook

Get Proof Request

The Client Application can get the Proof Request details which contain the attributes that can be further used in other API calls.

Example: Getting proof request details using threadId

Request:

```
curl -X 'GET' \

'https://demo-client.bhutanndi.com/verifier/v1/proof-request?id=10&threadId=00970f2f-
09c7-4fef-a2aa-6e76e62d6a15&page=1&pageSize=10' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <ACCESS_TOKEN>'
```

Response:

```
{
  "statusCode": 200,
  "message": "Proof request fetch successful",
  "data": {
    "id": 27,
    "threadId": "00970f2f-09c7-4fef-a2aa-6e76e62d6a15",
    "relationshipDid": "ac589f62-4c61-403a-b225-fde8c2a0a178",
    "status": "proofInvitationCreated",
    "connectionDate": "2024-06-18T05:00:09.377Z",
    "requestType": null
  }
}
```

- relationshipDid: Relationship DID of the user. It can be used for sending Proof Request or Credential Offer directly to the user if status is ProofValidated.
- status:
    - ProofInvitationCreated: Proof is not yet shared by user
    - ProofValidated: Proof is shared by user and validated
    - ProofFailed: Proof share failed

Proof Request using Relationship

The Client Application can use this API if it has Relationship DID of a user which can be used for sending the Proof Request directly to the Bhutan NDI Wallet without the need of QR code scan.

Example: Sending Proof Request directly to Bhutan NDI Wallet by using the users Relationship DID

Request:

```
curl -X 'POST' \
  'https://demo-client.bhutanndi.com/verifier/v1/proof-request' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <ACCESS_TOKEN>' \
  -H 'Content-Type: application/json' \
  -d '{
  "proofName": "Verify Foundational ID",
  "proofAttributes": [
    {
      "name": "ID Number",
      "restrictions": [
        {
          "schema_name":
"https://dev-schema.ngotag.com/schemas/c7952a0a-e9b5-4a4b-a714-1e5d0a1ae076"
        }
      ]
    },
    {
      "name": "Full Name",
      "restrictions": [
        {
          "schema_name":
"https://dev-schema.ngotag.com/schemas/c7952a0a-e9b5-4a4b-a714-1e5d0a1ae076"
        }
      ]
    }
  ],
"forRelationship": "243bc389-336b-497d-ae55-db14e6de7125"
}
'
```

Response:

```
{
  "statusCode": 201,
  "message": "Proof URL created successfully",
  "data": {
    "proofRequestName": "Verify Foundational ID",
    "proofRequestThreadId": "8e62c3e8-e0e0-4b65-8eb1-6aa8cb363831"
  }
}
```

- proofRequestThreadId: Used for subscribing to NATS or Webhook

NOTE: Proof shared in the presentation result needs to be accessed as a list.

4. Issuer Service

The issuer service exposes API for creating Relationships, issuing and revoking Verifiable Credentials from the Bhutan NDI Wallet. Any response from the Bhutan NDI Wallet such as accepting the connections or credentials that can be consumed by Client Application needs to consume the response by subscribing to webhook or NATS using threadId as a pattern.

Note: The endorsement of the Schema and Credential Definition for issuing the Verifiable Credential needs to be consulted with the Bhutan NDI team.

All the attributes defined in the Schema must be used while issuing the Verifiable Credential by providing a valid Schema ID and the attributes used are case sensitive.

Please visit the following swagger page for API details.

Swagger URL: https://demo-client.bhutanndi.com/issuer/swagger

Issue Credential

The Client Application can use the Issuer Service to issue Verifiable Credential to Bhutan NDI Wallet. A generic flow for issuing Verifiable Credential can be found here.

Example: Issuing Academic Certificate Verifiable Credential to Bhutan NDI Wallet

Request:

```
curl -X 'POST' \
  'https://demo-client.bhutanndi.com/issuer/v1/issue-credential' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <ACCESS_TOKEN> ' \
  -H 'Content-Type: application/json' \
  -d '{
  "credentialData": {
    "Issuer Name": "Demo",
    "Student ID": "1234",
    "Student Name": "Dorji",
    "Title of Award": "B.Sc. Computer Science",
    "College Name": "Demo College"
  },
"schemaId":"https://dev-schema.ngotag.com/schemas/ff021513-94b1-407d-a0ee-bb829531df4
2",
"holderDID": "did:key:z6Mkkh1pxrNMc4gNv9srXY2pHFxED8cJCuSV3UdgKiybp44Z"
}'
```

Response:

```
{
  "statusCode": 201,
  "message": "Credential offer created successfully",
  "data": {
    "credInviteURL":
"https://stage-demo-shortening-url.s3.ap-southeast-1.amazonaws.com/default/b5f15c2a-2
b2a-42c3-8f91-c032c5eb69d3",
    "deepLinkURL":
"bhutanndidemo://data?url=https://stage-demo-shortening-url.s3.ap-southeast-1.amazona
ws.com/default/b5f15c2a-2b2a-42c3-8f91-c032c5eb69d3",
    "revocationId": "2c9cab83-fb6a-4a7c-ae64-27fbf609e360",
    "relationshipDid": "5c2da139-ceb1-4875-815e-ccfbc7a15b74",
    "issueCredThreadId": "4e9d9ff1-0183-4cd8-92d2-1b77e55a3337"
  },
"schemaId":"https://dev-schema.ngotag.com/schemas/ff021513-94b1-407d-a0ee-bb829531df4
2",
"holderDID": "did:key:z6Mkkh1pxrNMc4gNv9srXY2pHFxED8cJCuSV3UdgKiybp44Z"
}
```

- credInviteURL: Used for generating QR Code
- relationshipDid: Relationship DID of the user if connecting for the first time. The existing Relationship DID will be available via NATS or Webhook for returning users.
- deepLinkURL: Used for displaying Deep Link
- revocationId: Used for credential revocation. Needs to be stored in Client Application
- issueCredThreadId: Used for subscribing to NATS or Webhook

Issue Credential using Relationship

Example: Sending Credential offer directly to user using Relationship DID

Request:

```
curl -X 'POST' \
  'https://demo-client.bhutanndi.com/issuer/v1/issue-credential' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <ACCESS_TOKEN>' \
  -H 'Content-Type: application/json' \
  -d '{
  "credentialData": {
    "Issuer Name": "Demo",
    "Student ID": "1234",
    "Student Name": "Dorji",
    "Title of Award": "B.Sc. Computer Science",
    "College Name": "Demo College"
  },
"schemaId":"https://dev-schema.ngotag.com/schemas/ff021513-94b1-407d-a0ee-bb829531df4
2",
"holderDID": "did:key:z6Mkkh1pxrNMc4gNv9srXY2pHFxED8cJCuSV3UdgKiybp44Z",
"forRelationship": "243bc389-336b-497d-ae55-db14e6de7125"
}'
```

Response:

```
{
  "statusCode": 201,
  "message": "Credential request sent successfully",
  "data": {
    "revocationId": "f4fd6d41-3259-49ef-8209-190892b664ca",
    "relationshipDid": "243bc389-336b-497d-ae55-db14e6de7125",
    "issueCredThreadId": "9e7caed6-2d1c-4e00-b767-a46e4dbcc516"
  }
}
```

- relationshipDid: Relationship DID of the user
- revocationId: Used for credential revocation. Stored in Client Application
- issueCredThreadId: Used for subscribing to NATS or Webhook

Credential Revocation Flow

- To make a Credential Definition revocable, the client has to request Bhutan NDI team
- If Credential Definition is revocable, the API to issue credential returns an additional attribute called revocationId
- Client Application has to store this revocationId against the user record in it's database
- Client Application can use this revocationId to revoke or suspend credential issued to the user
- Allowed Credential Status: ACTIVE, REVOKED, SUSPENDED

Transition Allowed :

- When credential is issued, status is ACTIVE by default
- ACTIVE credentials can be SUSPENDED or REVOKED
- SUSPENDED credentials can return to the ACTIVE or REVOKED stage after condition is achieved.
- REVOKED credential cannot be transitioned to any other state mentioned above.

Impact on Proof Request Flow:

- If status is ACTIVE, holder will be able to share proof
- If status is REVOKED or SUSPENDED, the holder will not be able to share the proof so the holder will have to decline proof request.

Example: Revoking credential issued to Bhutan NDI Wallet

Request:

```
curl -X 'POST' \

'https://demo-client.bhutanndi.com/issuer/v1/revoke_suspend?status=SUSPENDED&revocati
onId=c28a6495-4af8-4332-9b99-9cfa415df238' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <ACCESS_TOKEN >' \
  -d ''
```

Response:

```
{
  "statusCode": 201,
  "message": "Credential status updated",
  "data": {
    "id": 8284,
    "revocationId": "c28a6495-4af8-4332-9b99-9cfa415df238",
    "updatedAt": "2024-06-18T06:18:28.624Z",
    "status": "SUSPENDED",
    "relationshipDid": "f673e379-71a7-4de9-8292-883b132641a1",
    "issuedAt": "2024-06-18T06:20:18.415Z"
  }
}
```

Relationship/Connection

The Client Application can use this API to create Relationships with the user’s Bhutan NDI Wallet which creates the Relationship DID that can be used for sending the proof request or credential offer directly to the user's Bhutan NDI Wallet. The Relationship DID can be stored in the Client Application for future interactions, however the Relationship DID will be invalid if the user deletes this Relationship from the Bhutan NDI Wallet. The Client Application can get the new Relationship DID if the user forms a new Relationship via Connection or Proof Request or Credential Offer.

The Client Application needs to consume the response using the pattern "connectionStatus/${threadId}" on NATS and ${threadId} for subscribing to Webhook.

Example: Creating a Relationship/Connection

Request:

```
curl -X 'POST' \
  'https://demo-client.bhutanndi.com/issuer/v1/connection' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <ACCESS_TOKEN>' \
  -H 'Content-Type: application/json' \
  -d '{}'
```

Response:

```
{
  "statusCode": 201,
  "message": "Connection created successfully",
  "data": {
    "id": 76,
    "threadId": "ee5b10e3-3344-4b40-ba1a-4e6920d67c48",
    "invitationId": "ee5b10e3-3344-4b40-ba1a-4e6920d67c48",
    "relationshipDID": "ee5b10e3-3344-4b40-ba1a-4e6920d67c48",
    "relationshipVK": "",
    "connectionDate": "2024-06-18T10:21:30.580Z",
    "connectionUrl":
"https://stage-demo-shortening-url.s3.ap-southeast-1.amazonaws.com/persist/3b86cb40-b
78b-4377-a517-60a239ed959c",
    "deepLinkURL":
"bhutanndidemo://data?url=https://stage-demo-shortening-url.s3.ap-southeast-1.amazona
ws.com/persist/3b86cb40-b78b-4377-a517-60a239ed959c"
  }
}
```

- threadId: Used for subscribing to NATS or Webhook
- relationshipDID: Relationship DID of the user if connecting for the first time. Existing Relationship DID will be available via NATS or Webhook for returning users.
- connectionUrl: Used for generating QR Code
- deepLinkURL: Used for displaying as Deep Link

5. Webhook Service

Introduction to Webhook Service

Bhutan NDI APIs are inherently asynchronous, meaning that actions get initiated by calling the APIs, and then find out about their completion at a later point in time. Actions can't complete reliably during a single HTTP request because the other side of the request often involves actions by users, examples include accepting invitations or offers and approving actions. Therefore webhook service can be used by client applications to get the asynchronous responses and it includes registration, subscription, updation and deletion of the client webhook endpoint.

The client application will need to expose a webhook endpoint. The webhook endpoint (Eg. **/webhook - HTTP POST**) needs to be served on a public URL so that webhook service can send messages to it.

Please visit the following swagger page for API details.

Swagger URL: https://demo-client.bhutanndi.com/webhook/swagger/

Webhook Authentication

There needs to be some form of authentication on the client webhook endpoint registered with the NDI Webhook Service to increase application security. Currently the only form of authentication would be OAuth2 and steps to authenticate client webhooks can be found in this document.

Client Webhook Registration with Authentication

OAuth2 version 1

Below is the payload to be used during registration of client webhook to use the OAuth 2 v1 flow.
Note: values mentioned in angular brackets should be replaced accordingly.

```
{
    "webhookId": "<webhookId>",
    "webhookURL": "<webhookURL>",
    "authentication": {
        "type": "OAuth2",
        "version": "v1",
        "data": {
            "url": "<access-token-provider-url>",
            "grant_type": "client_credentials",
            "client_id": "<client_id>",
            "client_secret": "<client_secret>"
        }
    }
}
```

OAuth2 version 2

Below is the payload to be used during registration of client webhook to use the OAuth2 v2 flow.

Note: values mentioned in angular brackets should be replaced accordingly.

```
{
    "webhookId": "<webhookId>",
    "webhookURL": "<webhookURL>",
    "authentication": {
        "type": "OAuth2",
        "version": "v2",
        "data": {
            "token": "<fixed-access-token>"
        }
    }
}
```

Example: Registering a demo webhook with authentication type OAuth2 V2.

Request:

```
curl -X 'POST' \
  'https://demo-client.bhutanndi.com/webhook/v1/register' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <ACCESS_TOKEN>' \
  -H 'Content-Type: application/json' \
  -d '{
```

```
  "webhookId": "webhookDemo01",
  "webhookURL": "https://2085-202-144-128-70.ngrok-free.app/webhook",
  "authentication": {
    "type": "OAuth2",
    "version": "v2",
    "data": {
      "token": "<fixed-access-token>"
    }
  }
}'
```

Response:

```
{
  "statusCode": 201,
  "message": "Registration is done successfully",
  "data": {
    "id": 14,
    "webhookId": "webhookDemo01",
    "webhookURL": "https://2085-202-144-128-70.ngrok-free.app/webhook"
  }
}
```

Webhook Service Flow for Client Application

- Client application registers a webhook endpoint and authentication with a unique webhook ID
- Client application subscribes to webhook service using the webhookId and the threadId from other API response
- NDI Webhook Service checks the client webhook endpoint information, if it is:
    - OAuth2 version 1, it checks
        - if it already has an unexpired 'access_token', use it
        - If it doesn't then
            - sends a POST request
                - to the "url"
                - with appropriate form data (which includes "grant_type", "client_id", "client_secret" fields)
            - expects an OK response with json payload with these two fields at minimum
                - access_token (the token to be used later on)
                - expires_in (access token expiry time in seconds)
    - OAuth2 version 2

        - use the fixed access-token provided during register/update endpoint
- Once NDI Webhook Service has the access token it sends the webhook message with a header called 'Authorization'
- Client application receives and authenticates the webhook messages for the subscribed threadId
- Client application sends a 202 Accepted response that will acknowledge the receipt of webhook messages
- Client application unsubscribes to webhook service using the threadId after the process is complete.

Subscribing to Webhook

The Client Application can subscribe to Webhook in order to get the response from the Bhutan NDI Wallet for the status of Relationship, Proof Request and Credential Offer.

Example:

Request:

```
curl -X 'POST' \
  'https://demo-client.bhutanndi.com/webhook/v1/subscribe' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <ACCESS_TOKEN>' \
  -H 'Content-Type: application/json' \
  -d '{
  "webhookId": "webhookDemo01",
  "threadId": "35e21e42-4b40-4d36-aa15-c42ffef8b9b8"
}'
```

Response:

```
{
  "statusCode": 202,
  "message": "Accepted",
  "data": {
    "threadId": "35e21e42-4b40-4d36-aa15-c42ffef8b9b8"
  }
}
```

Unsubscribing to Webhook

The Client Application needs to unsubscribe to webhook once the process is complete.

Example:

Request:

```
curl -X 'POST' \
  'https://demo-client.bhutanndi.com/webhook/v1/unsubscribe' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer <ACCESS_TOKEN>' \
  -H 'Content-Type: application/json' \
  -d '{
  "threadId": "35e21e42-4b40-4d36-aa15-c42ffef8b9b8"
}'
```

Response:

```
{
  "statusCode": 200,
  "message": "Unsubscribed Successfully",
  "data": {
    "threadId": "35e21e42-4b40-4d36-aa15-c42ffef8b9b8"
  }
}
```

Webhook Responses

Relationship Status

Example: New relationship is created for the first time or reconnects after deleting the previous relationship.

```
{
    "type": "relationship-status/accepted",
    "relationshipDid": "4e00676b-472a-402b-9430-3f299b58d6b3",
    "thid": "4e00676b-472a-402b-9430-3f299b58d6b3"
}
```

Example: Relationship already exists for the user

```
{
    "type": "relationship-status/reused",
    "relationshipDid": "4e00676b-472a-402b-9430-3f299b58d6b3",
    "thid": "4e00676b-472a-402b-9430-3f299b58d6b3"
}
```

Present Proof Status

Example: Proof is rejected by the user.

```
{
    "type": "present-proof/rejected",
    "thid": "8033c821-d1aa-4ba4-9abe-0b5227b739ba"
}
```

Example: Proof is shared by the user.

```
{
    "type": "present-proof/presentation-result",
    "verification_result": "ProofValidated",
    "requested_presentation": {
        "unrevealed_attrs": {},
        "predicates": {},
        "identifiers": [
            {
                "schema_id":
"https://dev-schema.ngotag.com/schemas/c7952a0a-e9b5-4a4b-a714-1e5d0a1ae076",
                "cred_def_id": null
            },
            {
                "schema_id":
"https://dev-schema.ngotag.com/schemas/c7952a0a-e9b5-4a4b-a714-1e5d0a1ae076",
                "cred_def_id": null
            }
        ],
        "self_attested_attrs": {},
        "revealed_attrs": {
            "ID Number": [
                {
                    "value": "1234",
                    "identifier_index": 0
                }
            ],
            "Full Name": [
                {
                    "value": "demo name",
                    "identifier_index": 1
                }
            ]
        }
    },
    "relationship_did": "b3ae21f9-eaa9-46bf-bd8f-617a4522fed5",
    "thid": "99ed65da-ff61-48d9-a7fe-f25cc54e3ec0",
    "holder_did": "did:key:z6MkjKxng36Cy4KRtn2Gsn7a3kEHiP9JhGYw72Tu5qzj85jG"
}
```

Credential Offer Status

Example: Credential offer is rejected by the user.

```
{
    "type": "issue-credential/rejected",
    "thid": "b79f3ba5-422e-40d8-81d3-60da13ba0e7f"
}
```

Example: Credential offer is accepted by the user.

```
{
    "type": "issue-credential/accepted",
    "thid": "b79f3ba5-422e-40d8-81d3-60da13ba0e7f"
}
```

Webhook Retries

Webhook service will attempt retries up to 10 times for a webhook message that has already failed. Initially, retries occur with a 2-second interval, doubling with each subsequent attempt, capped at a maximum interval of 30 seconds. Success is indicated by the client's acknowledgement with a status code of 202, signifying acceptance.

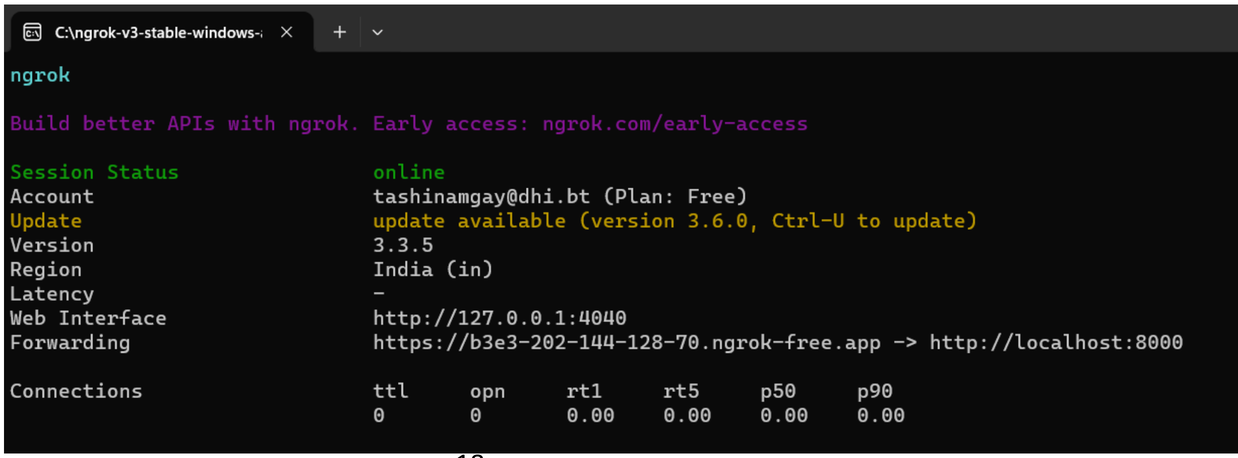
Webhook Service for Development using Ngrok

For development purposes, the client application listening port can be assigned a public URL via means of the Ngrok tool. Ngrok has to be installed to enable the webhook client application to be exposed outside the local development machine.The technical documentation for ngrok can be accessed from https://ngrok.com/docs/what-is-ngrok/.

Steps for using ngrok:

1. Install ngrok from the official ngrok website (https://ngrok.com/).
2. Open the ngrok executable file in your system.
3. The client application should be running in your system (note the port number in which your application is running)
4. To expose the local application outside of the local machine, the following steps must be followed:

    a. Input the command `ngrok http PORT_NUMBER` in the ngrok terminal. (Replace PORT_NUMBER with the actual port in which your local application is running).



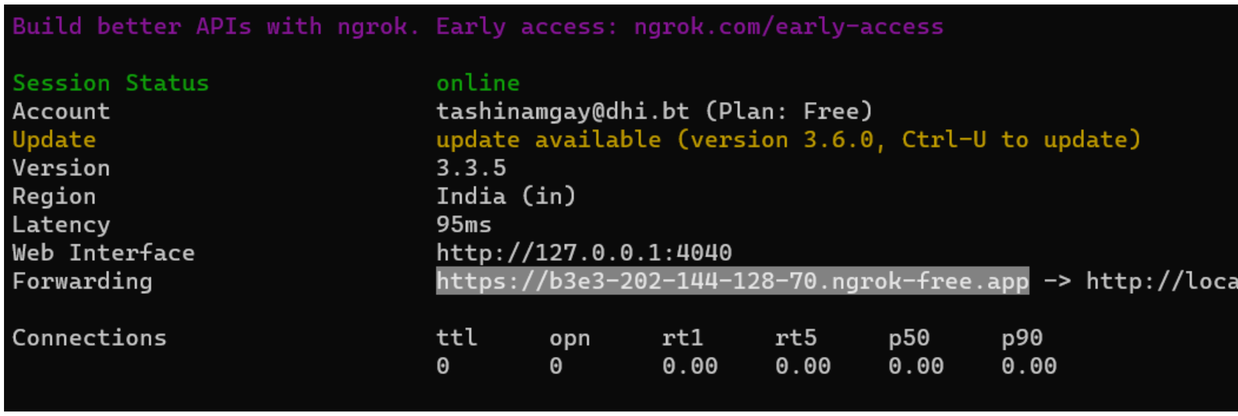
```
ngrok

Build better APIs with ngrok. Early access: ngrok.com/early-access

Session Status                online
Account                       tashinamgay@dhi.bt (Plan: Free)
Update                        update available (version 3.6.0, Ctrl-U to update)
Version                       3.3.5
Region                        India (in)
Latency                       -
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://b3e3-202-144-128-70.ngrok-free.app -> http://localhost:8000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

b. Your application is now exposed outside the local machine. Copy the URL from the forwarding section in the output.



```
Build better APIs with ngrok. Early access: ngrok.com/early-access

Session Status                online
Account                       tashinamgay@dhi.bt (Plan: Free)
Update                        update available (version 3.6.0, Ctrl-U to update)
Version                       3.3.5
Region                        India (in)
Latency                       95ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://b3e3-202-144-128-70.ngrok-free.app -> http://loca

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

c. The client application will need to expose a webhook endpoint (Eg. /webhook - HTTP POST).

d. Now the webhook endpoint (Eg. https://b3e3-202-144-128-70.ngrok-free.app/webhook) can be used for webhook registration.

6. NDI NATS Server

NATS server is used by NDI services as a publish subscribe messaging system based on pattern or subject. Client system needs to subscribe on the NATS server for consuming any responses from Bhutan NDI Wallet.

NATS Connection Configuration

Please find below the configurations for subscribing to NDI NATS server for development:

- **NATS URL:** `https://natsdemoclient.bhutanndi.com`
- **NATS Web Socket URL:** wss://natsdemoclient.bhutanndi.com
- **Seed:** SUAPXY7TJFUFE3IX3OEMSLE3JFZJ3FZZRSRSOGSG2ANDIFN77O2MIBHWUM
- **Pattern**: threadId

NATS Authentication

Client system needs to implement the NATS authentication for subscribing to NATS server:

- Configure seed while connecting to NDI NATS Server. Refer to this document for more details.
- Seed should be treated like a secret
- Sample code for NodeJs:

```
const seed = new TextEncoder().encode(
"SUAPXY7TJFUFE3IX3OEMSLE3JFZJ3FZZRSRSOGSG2ANDIFN77O2MIBHWUM",
);

const nc = await connect({
port: ns.port,
authenticator: nkeyAuthenticator(seed),
});
```

NATS Responses

Present Proof

Example: Proof shared by user

```
{
    "type": "present-proof/presentation-result",
    "verification_result": "ProofValidated",
    "requested_presentation": {
        "unrevealed_attrs": {},
        "predicates": {},
        "identifiers": [
            {
                "schema_id":
"https://dev-schema.ngotag.com/schemas/c7952a0a-e9b5-4a4b-a714-
1e5d0a1ae076",
                "cred_def_id": null
            },
            {
                "schema_id":
"https://dev-schema.ngotag.com/schemas/c7952a0a-e9b5-4a4b-a714-
1e5d0a1ae076",
                "cred_def_id": null
            }
        ],
        "self_attested_attrs": {},
        "revealed_attrs": {
            "ID Number": [
                {
                    "value": "1234",
                    "identifier_index": 0
                }
            ],
            "Full Name": [
                {
                    "value": "demo name",
                    "identifier_index": 1
                }
            ]
        }
    },
    "relationship_did": "b3ae21f9-eaa9-46bf-bd8f-617a4522fed5",
    "thid": "99ed65da-ff61-48d9-a7fe-f25cc54e3ec0",
    "holder_did":
"did:key:z6MkjKxng36Cy4KRtn2Gsn7a3kEHiP9JhGYw72Tu5qzj85jG"
}
```

Reference links :

https://github.com/nats-io/nats.js#services
https://github.com/nats-io/nats.js#publish-and-subscribe

7. QR Code and Deep Link

The QR Code or Deep Link are methods that need to be used by the client system to interact with the Bhutan NDI Wallet of the user while using NDI services.

QR Code is used when the user has two devices where one is used for displaying the QR code and the other one is used for scanning. The URL received from API for creating proof requests, creating connections and issuing credentials need to be encoded into QR Code and displayed to the user.

Deep Link is used when the user uses the same device that has the NDI wallet to access the client system (when the QR Code is not possible to scan). The deep link URL received from API for creating connections, creating proof requests and issuing credentials needs to be displayed as a link to the user.

8. Schema

A Schema refers to a set of attributes or properties that a Verifiable Credential (VC) can contain. The Schema is endorsed on the Blockchain by the NDI Team. The list of Schema can be found here.

9. Bhutan NDI Wallet

Bhutan NDI Wallet is a mobile application that is available for both Android and iOS devices which can be used by the users to interact with the client system integrated with NDI services. The users will be issued with foundational ID and the permanent address credential after successful onboarding to the Wallet. The developers need to use the Demo Bhutan NDI App to test the integration with NDI services.

Platform Compatibility

For now, Bhutan NDI App works best on Android and iOS devices with the following specifications:

- Android – Minimum support is Android 6.0 (API level 23) & maximum support is Android 13 (API level 33) with all architectures.
- iOS – Minimum 13.0 OS version

Note: The Demo Bhutan NDI App is shared with the developers for integration testing purposes. We kindly request you to refrain from sharing it with the public.

The stable version of the Demo Bhutan NDI App can be downloaded from:

- Android phones – (Download from here)
- IOS phones – (Developers need to share Apple ID. Download will be available from TestFlight)

Demo User Data

The following demo user data can be used for onboarding to Demo Bhutan NDI App enabling any developers to use it for the development and testing purposes.

- “FullName”: “Dorji Sonam”,
- “Gender”: “Male”,

- “Bhutanese”: Yes
- "Date of Birth": 19/07/1995
- “ID Type”: National ID Card
- “ID Number”: 1234
- “Dzongkhag”: Trongsa
- “Gewog”: Tangsibjee

10. Terminologies

- **Bhutan NDI Wallet:** Mobile based applications provided by Bhutan NDI to receive, store and share credentials
- **Verifiable Credential:** A Credential/Credentials/Verifiable Credentials(VCs) is/are a set of claim/claims issued to prove his/her identity
- **Deep Link:** A clickable URL link to redirect to a piece of content on the web or a mobile application
- **Client Application:** Any client application such as web application or mobile application that is being integrated with Bhutan NDI
- **DID:** Decentralized Identifiers (DIDs) are unique identifiers that can refer to any subject (person, organization, thing, etc,)