Technical specifications

NYTRO

NFT Upgrade

NyzoSy and Iyomisc

NYZO

Version 2.01

Introduction

Nyzo main strengths

- Diversity – Fairer than PoW and PoS. Time and ipv4 as scarce resources.
- Fast finality – not just 7s block time: once a block is frozen, it's guaranteed to be final.
- Cycle fund with full control by > 50% of the cycle signatures
- Resources efficient

Nytro: Do not tamper with efficient core working, but build on top

- Second layer protocol, only to be processed by applications having an interest.

New NFT features:

- ERC-721 similar token
- User and Admin – limited – data storage
- Ease up Non Fungible Token use for games, art – just to name a few, with a dedicated protocol upgrade.

Base document

- The current document only deals with the NFT part of the NYTRO protocol. **Please refer to the main nytro_protocol.pdf document for basic concepts of the protocol and Tokens operations.**

A second layer protocol

No change to the core layer:

- The verifiers only deal with first layer, regular Nyzo transactions
- They have no knowledge of the layers above
- They will not interpret the second layer data, nor process additional rules.
- They do not need any change to code, consensus or anything
- Verifiers continue to freeze transactions with their associated data, like they already do.

Protocol rules:

- The protocol adds rules to the sender, recipient and data fields of the regular Nyzo transactions.
- The protocol rules define what is a layer-2 valid or invalid transaction.
- The state of the layer-2 objects can be rebuilt anytime from the historical layer-1 transactions, with no extra data.
- Every valid transaction updates the state of the layer-2 objects

Protocol implementations:

- Like with verifiers, there can exist several implementations
- Test vectors are provided to test the implementations with various rules
- In case of conflict, the reference implementation is to be trusted, test vectors added and rules made more clear if needed.

Some insights

Human readable data:

- The protocol was defined so the data can be read by a human. Ascii data, no encoding. One can look through the transactions and read what they do.

Helpers:

- The transactions are however intended to be created through helpers, that make sure you don't send invalid data. We provide online helpers as well as a generic python helper class anyone can use.

Fees:

- Transfer and Ownership transfer operations use the token recipient as Nyzo recipient. This is required due to Nyzo small data field. No other fee than the regular tx fee can then be collected in a direct manner. These operations are then basically for free.
- Token issuance and Mints are sent to the cycle address. Technically, these 3 can require any Nyzo amount to be paid. At start, Mint and Burn will not require a fee.
- Since required fees are at protocol level, they can change if needed. The cycle could vote on them for instance.

Nytro , Tokens and NFT

NYTRO

- NYTRO is the layer 2 protocol, responsible for handling Tokens and NFTs. NYTRO manages both Tokens and NFT operations.
- NYTRO does support the NFT operations since v2.0

TOKENS

- Tokens are fungible tokens, ERC-20 like. They are handled by “T?:” operations

NFTs

- NFT are non fungible tokens, close to ERC-721. They are handled by “N?:” operations and their name is prefixed by a lowercase “n”.

Tokens and NFT Interaction

- Tokens and NFTs can interact together via application specific rules that are not hardcoded in the Nytro Protocol.
- It is possible to burn a token to an NFT “address”.

NFT classes and NFT instances

NFT Class:

- This is the “Collection” the minted NFTs (or instances) belong to
- NFT Class name begins with a lowercase “n”
- An NFT Class name is unique
- An NFT Class can either have an unlimited or max supply
- An NFT Class can have – limited - data bound to it
- The owner of the Class is the only one able to mint NFT related instances.

- Ex: “nDEMO” the DEMO NFT Class

NFT Instance:

- This is the non fungible token itself
- It is identified by the NFT Class name plus a Unique ID for that Class name.
- It can have specific data attached
- Data can be admin (NFT class owner) or user (NFT instance owner) related.

- Ex: “nDEMO:00000001” The unique nDEMO NFT instance, of UID 00000001

Common NFT properties

Token Name:

- Only one NFT Class of a given name can exist. No possible confusion.
- NFT Class name has to follow strict rules, charset and case. No trick with utf8 charsets.

Units:

- Every NFT has an implicit unit of 1, non divisible.
- A NFT class can mint either unlimited or a max amount of NFT instances. This is defined at NFT class issuance and can not change afterward.

Fees:

- Transfer and Ownership transfer of an NFT currently have minimal fees: 0.000001 Nyzos
- NFT Class issuance has a 300 Nyzos fee*****. Objective: Limit spam, limit cybersquatting, reward cycle.
- Every Mint of a NFT instance has minimal 0.000001 Nyzos fee.
  Objective: facilitate NFT distribution with one-step mint and deliver, limit recurrent fees for services.

  ***** Initial amount, could be updated later on.

Anatomy of a NFT transaction 1/2

Regular Nyzo transaction

A Regular Nyzo transaction, from user point of view, consists of:

- A Sender (Nyzo address)
- A Recipient (Nyzo address)
- A Nyzo amount (Nyzos, with micro Nyzo precision)
- A Sender data field of 32 bytes.

Token transaction:

A Token or NFT transaction is a regular Nyzo transaction making use of the Recipient and Sender data fields to convey additional meaning.

- Nyzo verifiers handle that transaction as a regular transaction, only storing the Sender data.
- Nytro Protocol aware clients leverage the extra data to infer tokens or NFT operations and related state changes.

Anatomy of a NFT transaction 2/2

Use of the Recipient address

With token transactions, Recipient is either the recipient of the token operation (ex: token transfer) or the cycle address (ex: issue, mint).

Using the cycle address as Recipient allows to pay fees that can then be used by the cycle fund.

In addition, Nytro can use the recipient field as data for specific operations. In that case, it is to be read as 32 bytes and is not a “real” Nyzo address or wallet.

Sender data format

Ex: “TT:TEST:2”

- Sender data is specified as Ascii text only. It can be read by a human no matter the local encoding.
- “:” char is used as a separator
- First segment denotes the token operation, here : “TT” for Token Transfer. First char is always “T” for Token protocol. This is a compressed namespace.
- Following segments are the parameters of that transfer operation: token name and token amount.

NFT Operations

- Fixed or unlimited max supply, defined at Class issuance

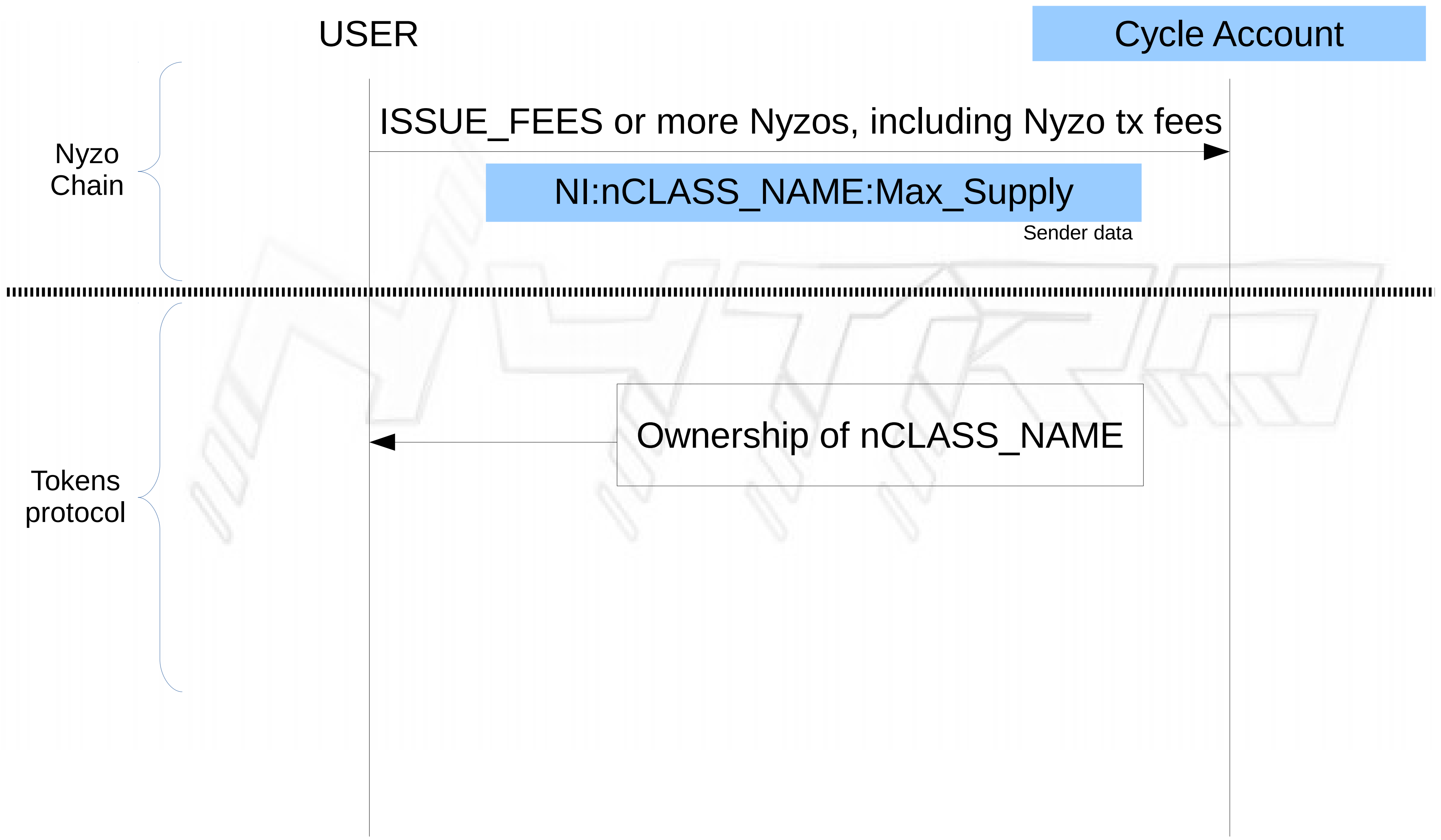
- NI: NFT Class Issuance
  Initial supply is null. Sender becomes the NFT Class owner.
- NT: NFT Transfer
  Current NFT instance owner can send the NFT instance to the given recipient
- NM: NFT Mint
  Current class owner can mint a new NFT instance, adds to supply.
- NB: NFT Burn
  Current NFT instance owner can destroy the NFT, removes from supply.
- NO: NFT Class Ownership change
  Current class owner can transfer ownership (mint and admin data rights) to the recipient.
- NA: NFT Admin data
  Current class owner can attach admin data payload to a NFT Class or instance
- ND: NFT User data
  Current NFT instance owner can attach user data payload to the NFT instance

NFTs OPERATIONS DETAILS

NyzoSy and Iyomisc

Version 2.01

NFT ISSUE – NI:


USER
Cycle Account
Nyzo Chain
ISSUE_FEES or more Nyzos, including Nyzo tx fees
NI:nCLASS_NAME:Max_Supply
Sender data
Tokens protocol
Ownership of nCLASS_NAME

NI: Rules

- nCLASS_NAME – The unique name of the NFT Class

  Mandatory lowercase “n” prefix
  followed by A..Z, 0..9, _
  Min 3 chars, Max 11 chars - including the “n” prefix
  Valid: nTEST, nMY_NFT, n45
  Invalid: Test, nTest, NTEST, nTOOLONGACLASSNAME

- Max Supply – The Max Supply of that NFT class, or -1 for no max

  Integer or -1, with no thousand separator
  Max Supply is strictly greater than 0 or equal to -1
  If Max supply is “-1”, this denotes a class that can be minted at will by the class owner.
  Valid: 1, 10, 1000000, -1
  Invalid: 0, 1 000, 000, “1,000,000”, 3.14

- Whole rules – Apply on the whole NI: transaction

  Max total length is 32 chars, including “NI:” and the “:” separators
  “NI:” transaction has to be sent to the cycle account to be valid
  “NI:” transaction has to send at least NFT_ISSUE_FEES Nyzos to the cycle, with NFT_ISSUE_FEES currently being 300 Nyzos, subject to change after cycle vote.

NI: Validation flow and rules

- Trigger: Data begins with “NI:” - case sensitive.
- Split by “:” delimiter
- NI_R0: consists of exactly 3 segments “segment0:segment1:segment2”
- NI_R1a: segment1 follows CLASS_NAME rules
- NI_R1b: segment1 is not an already issued CLASS_NAME
- NI_R2a: segment2 obeys Max supply format. Regexp: /^[\-0-9]+$/
- Extract “Max Supply” as integer by converting from string
- NI_R2b: Max Supply is > 0 or equals -1
- NI_R3a: recipient is cycle address
- NI_R3b: amount of Nyzos sent – including native Nyzo tx fees - is equal or higher than current NFT_ISSUE_FEES
- Note: The order in which the rules are evaluated is not important since any broken rule renders the token transaction invalid. In practice and for performance reasons low costs rules may be tested first. A canonical test order will be provided for test suites.

NI: Example

I want to create a new Nyzo NFT Class, named “nDEMO”. That name follows the rules and does not exist yet.

I want an unlimited supply.

I send a 300 Nyzos***** (ISSUE_FEES) transaction to the cycle address
“**id__800000000000000000000000000000000000002kmiC2yjF**” with the data properly encoded.

Sender data would then be “**NI:nDEMO:-1**”

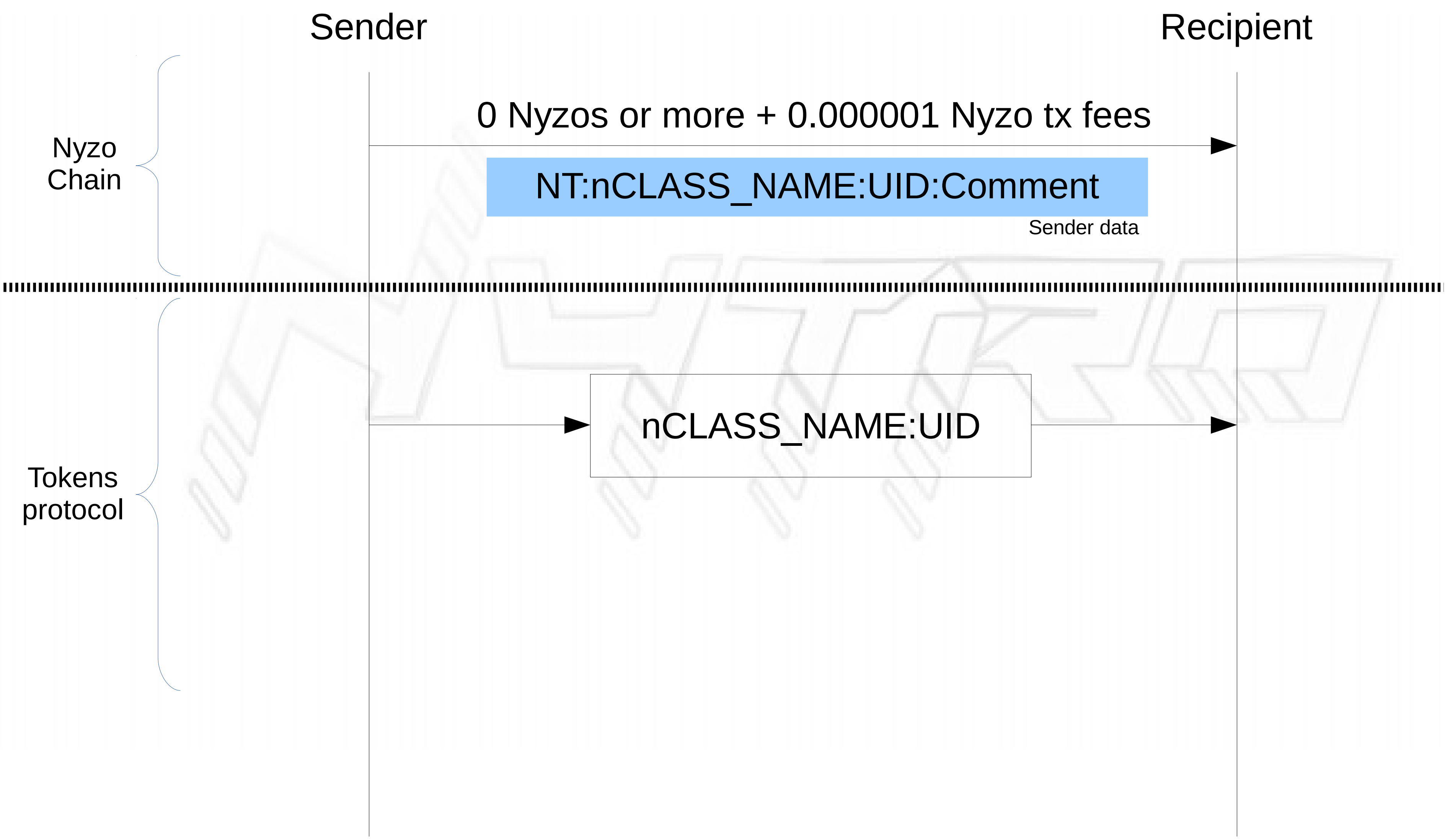
I don't need to remember all the rules and risk errors that would invalidate my operation.
To make sure, I can use an online helper, like the one at https://tokens.nyzo.today/helpers/issue or a wallet that supports tokens operations.

Once my transaction is frozen by the cycle, the token explorer will show my newly created NFT Class and current (zero) supply, with my address as owner.

I can now create actual “nDEMO” NFTs by minting them, via the “NM” operation.

***** Initial amount, could be updated later on.

NFT TRANSFER – NT:



NT: RULES

- CLASS_NAME – The class name of the NFT to send

  This class name has to exist
  CLASS_NAME has to obey the general CLASS_NAME rules about charset, uppercase and size.

- UID

  String.
  a..z, A..Z, 0..9, _ (63 different chars)
  8 chars max, 1 char min
  This UID has to exist for this NFT class name, requires a previous “NM”, Mint operation for that UID.
  The sender needs to own this precise NFT instance (CLASS_NAME:UID)

- Comment

  String, Optional.
  a..z, A..Z, 0..9, _ (63 different chars)

- Whole rules – Apply on the whole NT: transaction

  (Regular Nyzo tx rule)
  Max total length is 32 chars, including “NT:” and the “:” separators
  Can't send to yourself (Nyzo core constraint)

NT: Validation flow and rules

- Trigger: Data begins with “NT:” - case sensitive.
- Split by “:” delimiter
- NT_R0: consists of 3 or 4 segments “segment0:segment1:segment2” or “segment0:segment1:segment2:segment3”
- NT_R1a: segment1 follows CLASS_NAME rules
- NT_R1b: CLASS_NAME NFT class exists
- NT_R2a: segment2 follows UID rules
- NT_R2b: CLASS_NAME:UID NFT instance exists and Sender owns it
- The following rules only apply if segment3 is defined
- NT_R3: segment3 obeys “comment” rules

- Note: The order in which the rules are evaluated is not important since any broken rule renders the token transaction invalid. In practice and for performance reasons low costs rules may be tested first. A canonical test order will be provided for test suites.

NT: Example

I want to send my “nDEMO:00000001” NFT to “id__8cdasPC2QVZ13iG42RWhp47gow9SsIXZgp0Aga59oEITG2X-M7Ur”.

This is a NFT instance I have in my balance.

I send a 1 micronyzo – that is 0.000001 Nyzos, the minimum allowed amount - transaction to the recipient address “**id__8cdasPC2QVZ13iG42RWhp47gow9SsIXZgp0Aga59oEITG2X-M7Ur**” with the data properly encoded.

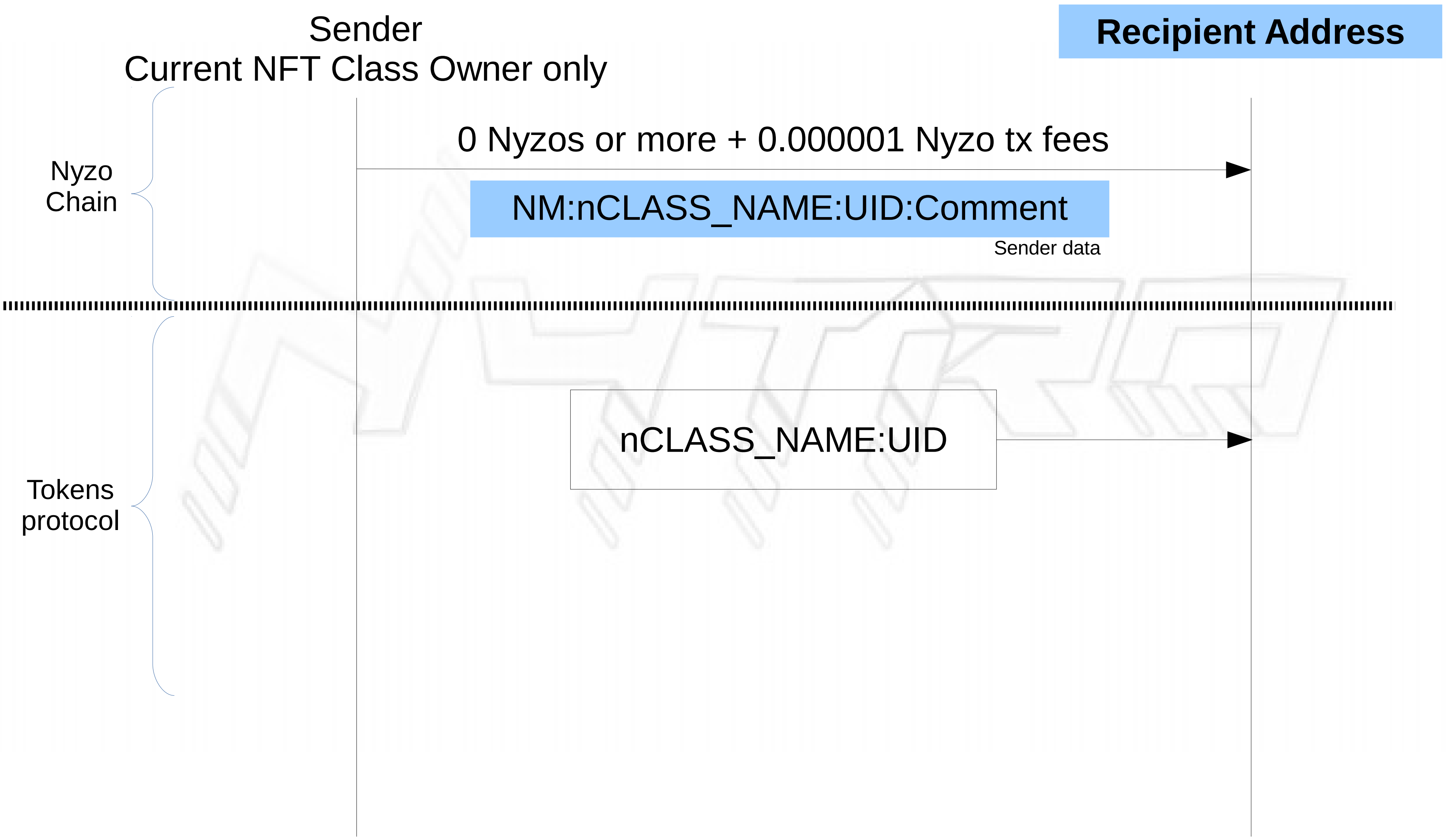
Sender data in that case would be “**NT:nDEMO:00000001**”

I could also add a small comment along, like “Hello”
Sender data in that case would be “**NT:nDEMO:00000001:Hello**”
(Beware, only a subset of ascii is available in comment)

I don't need to remember all the rules and risk errors that would invalidate my operation.
To make sure, I can use an online helper, like the one at https://tokens.nyzo/today/helpers/transfer/ or a wallet that supports tokens operations.

Once my transaction is frozen by the cycle, the token explorer will show the “nDEMO:00000001” NFT instance now belonging to my recipient.

NFT Instance MINT – NM:


Sender
Current NFT Class Owner only
Recipient Address
Nyzo Chain
0 Nyzos or more + 0.000001 Nyzo tx fees
NM:nCLASS_NAME:UID:Comment
Sender data
Tokens protocol
nCLASS_NAME:UID

NM: RULES

- CLASS_NAME – The class name of the NFT to mint

  This class name has to exist
  CLASS_NAME has to obey the general CLASS_NAME rules about charset, uppercase and size.
  The sender needs to own this precise NFT CLASS_NAME

- UID

  UID Obeys the UID rules
  This UID must not exist already for this NFT class name and can't be empty.

- Recipient

  Can be any Nyzo address: will become the owner of the NFT instance.

- Comment

  String, Optional.
  a..z, A..Z, 0..9, _ (63 different chars)

- Can't mint if Max supply is reached

- Whole rules – Apply on the whole NM: transaction

  (Regular Nyzo tx rule)
  Max total length is 32 chars, including “NM:” and the “:” separators

NM: Validation flow and rules

- Trigger: Data begins with “NM:” - case sensitive.
- Split by “:” delimiter
- NM_R0: consists of 3 or 4 segments “segment0:segment1:segment2” or “segment0:segment1:segment2:segment3”
- NM_R1a: segment1 follows CLASS_NAME rules
- NM_R1b: CLASS_NAME NFT class exists and the sender owns it
- NM_R2a: segment2 follows UID rules
- NM_R2b: CLASS_NAME:UID NFT instance does not exist already
- NM_R2c: Max supply is -1 or current instances count < max supply
- The following rules only apply if segment3 is defined
- NM_R3: segment3 obeys “comment” rules

- Note: The order in which the rules are evaluated is not important since any broken rule renders the token transaction invalid. In practice and for performance reasons low costs rules may be tested first. A canonical test order will be provided for test suites.

NM: Example

I'm the owner of the “nDEMO” NFT Class.
I want to mint a new NFT instance and send it to
“id__8cdasPC2QVZ13iG42RWhp47gow9SsIXZgp0Aga59oEITG2X-M7Ur”.

I send a 1 micronyzo – that is 0.000001 Nyzos, the minimum allowed amount - transaction to the recipient address “**id__8cdasPC2QVZ13iG42RWhp47gow9SsIXZgp0Aga59oEITG2X-M7Ur**” with the data properly encoded.

Say I want to mint UID 00000003. This is arbitrary and does not have to be a number. Constraints are to follow the UID rules, and be unique across nDEMO class.

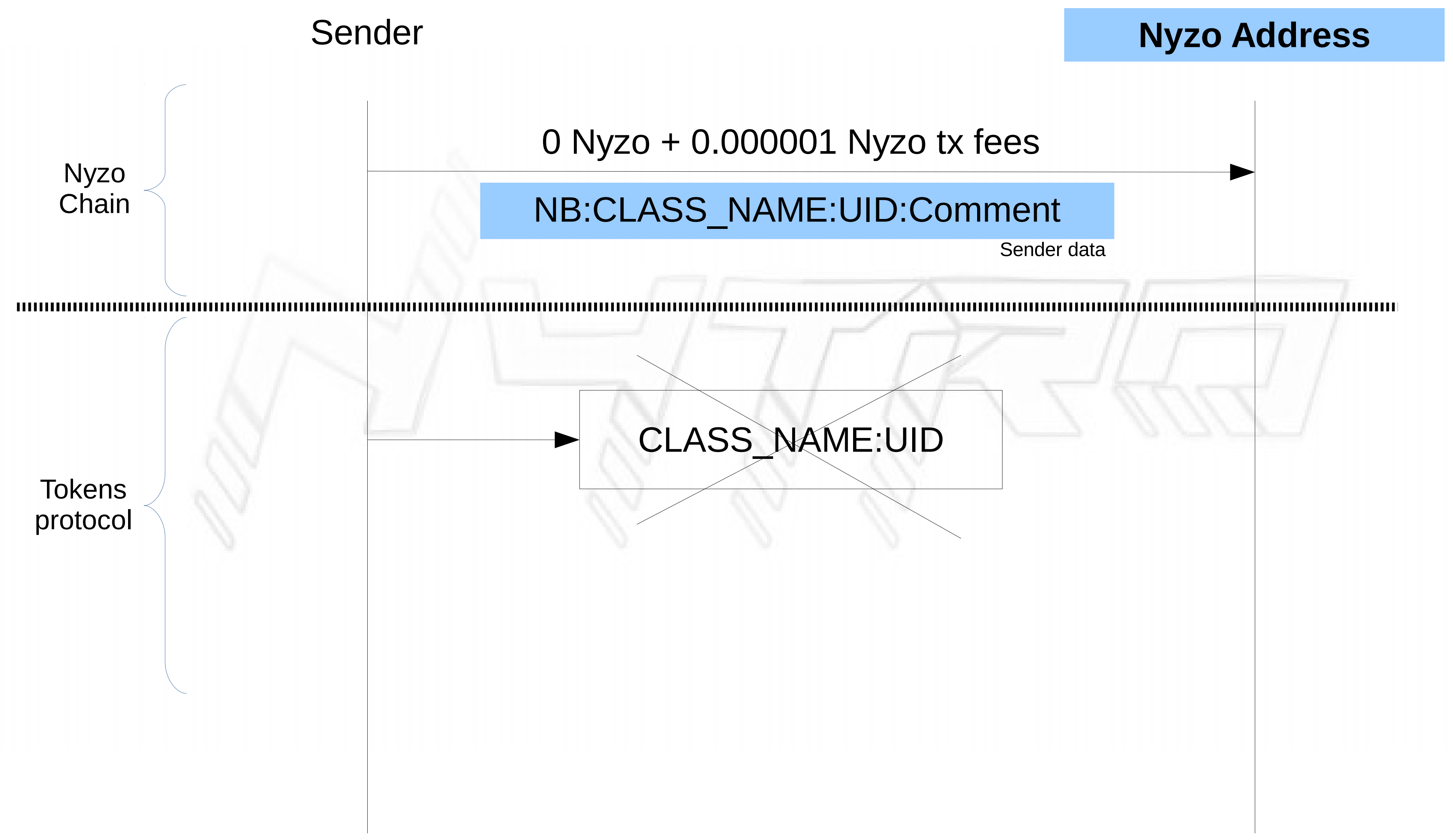
Sender data in that case would be “**NM:nDEMO:00000003**”

I could also add a small comment along, like “Hello”
Sender data in that case would be “**NM:nDEMO:00000003:Hello**”
(Beware, only a subset of ascii is available in comment)

Once my transaction is frozen by the cycle, the token explorer will show the “nDEMO:00000003” NFT instance being created and belonging to the recipient, just with a single transaction and 1 µNyzo.

I can then follow up and attach admin data to that NFT instance if necessary, via “NA” operation.

NFT BURN – NB:



Sender
Nyzo Address
Nyzo Chain
0 Nyzo + 0.000001 Nyzo tx fees
NB:CLASS_NAME:UID:Comment
Sender data
Tokens protocol
CLASS_NAME:UID

NB: RULES

- CLASS_NAME – The class name of the NFT to send

  This class name has to exist
  CLASS_NAME has to obey the general CLASS_NAME rules about charset, uppercase and size.

- UID

  String.
  a..z, A..Z, 0..9, _ (63 different chars)
  8 chars max
  This UID has to exist for this NFT class name.
  The sender needs to own this precise NFT instance (CLASS_NAME:UID)

- Comment

  String, Optional.
  a..z, A..Z, 0..9, _ (63 different chars)

- Whole rules – Apply on the whole NB: transaction

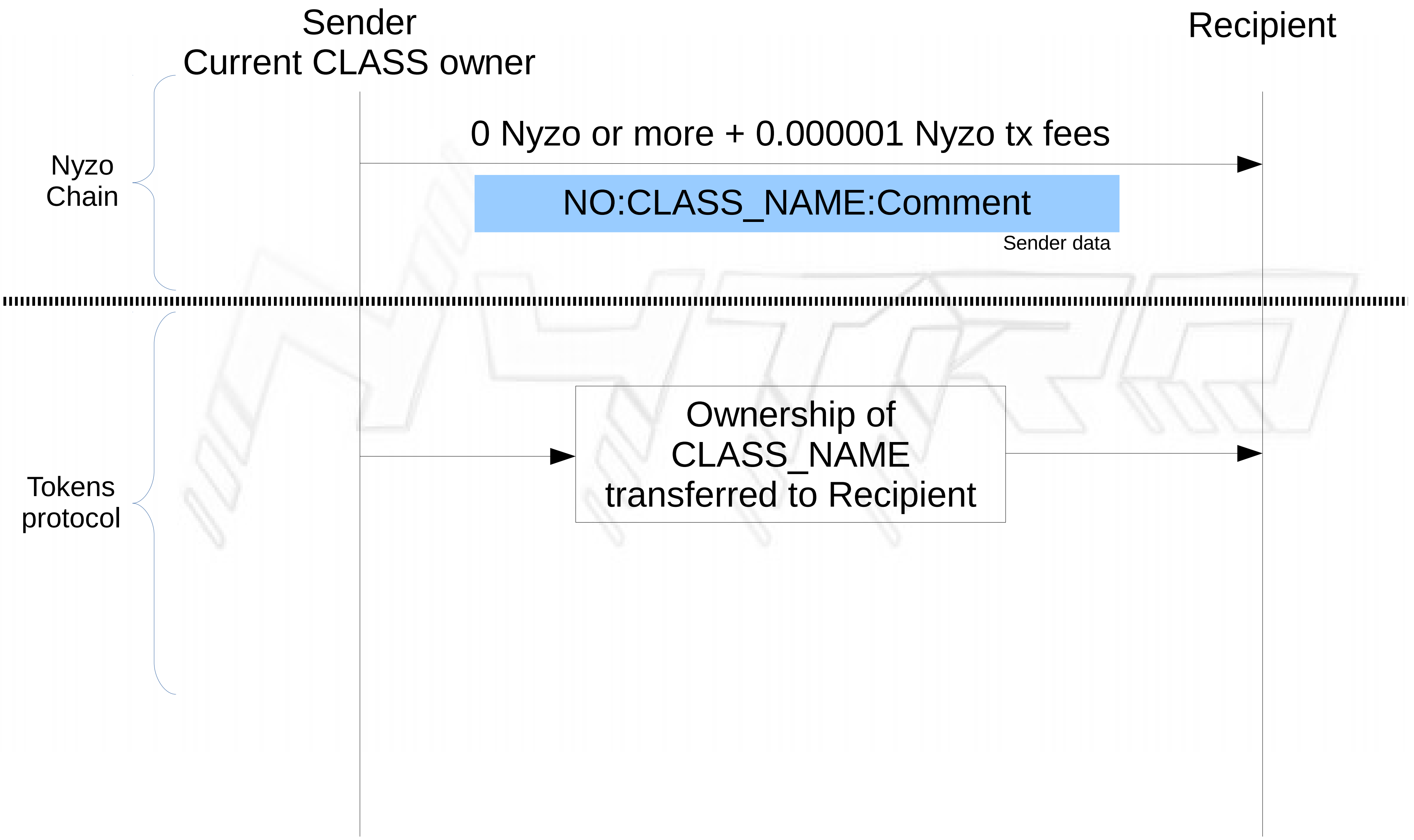
  (Regular Nyzo tx rule)
  Max total length is 32 chars, including “NB:” and the “:” separators
  Can't send to yourself (Nyzo core constraint)

NB: Validation flow and rules

- Trigger: Data begins with “NB:” - case sensitive.
- Split by “:” delimiter
- NB_R0: consists of 3 or 4 segments “segment0:segment1:segment2” or “segment0:segment1:segment2:segment3”
- NB_R1a: segment1 follows CLASS_NAME rules
- NB_R1b: CLASS_NAME NFT class exists
- NB_R2a: segment2 follows UID rules
- NB_R2b: CLASS_NAME:UID NFT instance exists and Sender owns it
- The following rules only apply if segment3 is defined
- NB_R3: segment3 obeys “comment” rules

- Note: The order in which the rules are evaluated is not important since any broken rule renders the token transaction invalid. In practice and for performance reasons low costs rules may be tested first. A canonical test order will be provided for test suites.

NFT Class Ownership Transfer – NO:



NO: RULES

- CLASS_NAME – The name of the NFT Class to transfer ownership of.

  This class name has to exist
  CLASS_NAME has to obey the general CLASS_NAME rules about charset, uppercase and size.
  The sender needs to own this precise NFT CLASS_NAME

- Recipient

  Can be any Nyzo address: will become the owner of the NFT Class.

- Comment

  String, Optional.
  a..z, A..Z, 0..9, _ (63 different chars)

- Whole rules – Apply on the whole NO: transaction

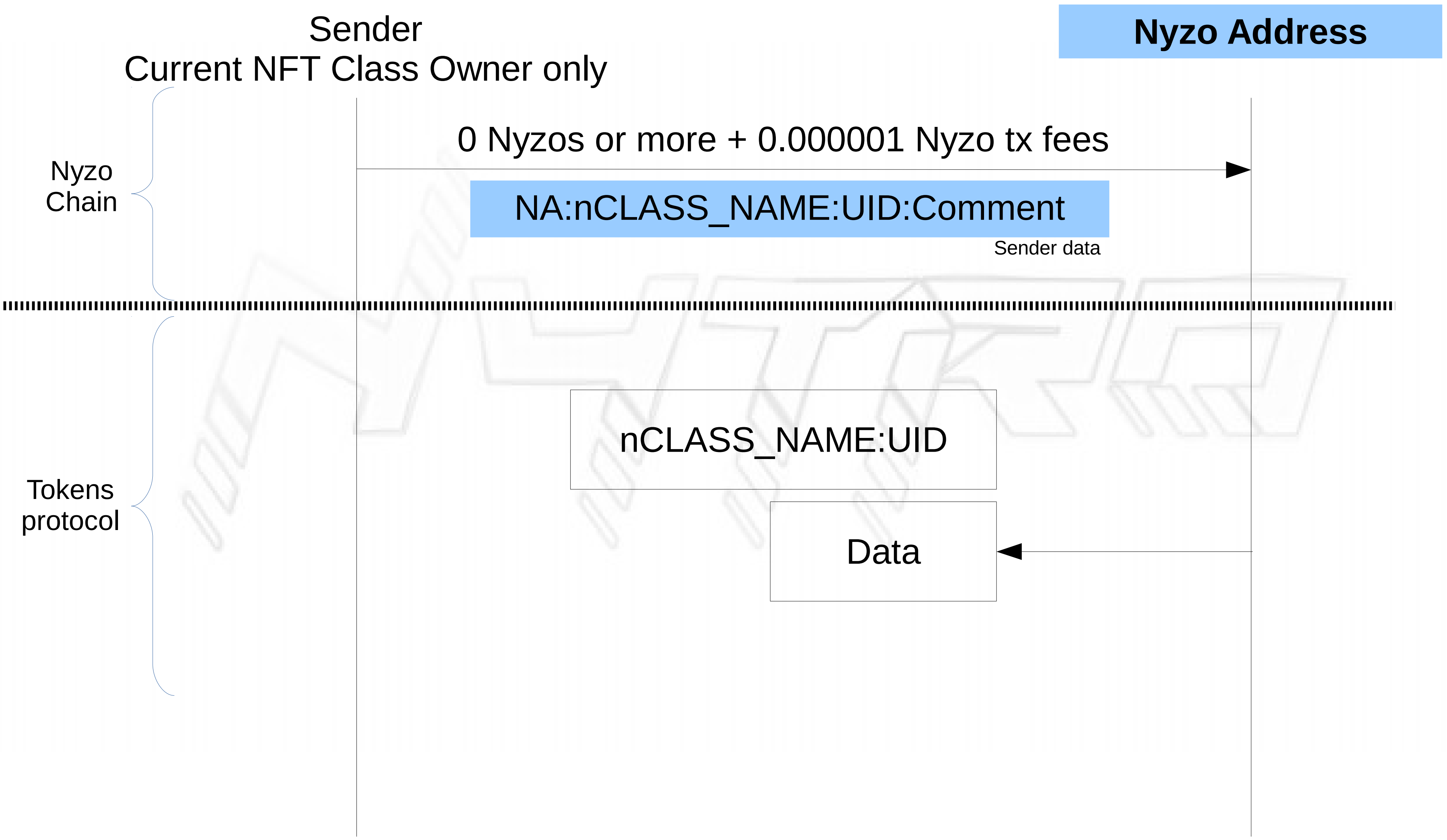
  (Regular Nyzo tx rule)
  Max total length is 32 chars, including “NO:” and the “:” separators

NO: Validation flow and rules

- Trigger: Data begins with “NO:” - case sensitive.
- Split by “:” delimiter
- NO_R0: consists of 2 or 3 segments “segment0:segment1” or “segment0:segment1:segment2”
- NO_R1a: segment1 follows CLASS_NAME rules
- NO_R1b: CLASS_NAME NFT class exists and the sender owns it
- The following rules only apply if segment2 is defined
- NO_R2: segment2 obeys “comment” rules

- Note: The order in which the rules are evaluated is not important since any broken rule renders the token transaction invalid. In practice and for performance reasons low costs rules may be tested first. A canonical test order will be provided for test suites.

NFT Admin Data – NA:


Sender
Current NFT Class Owner only
Nyzo Address
Nyzo Chain
0 Nyzos or more + 0.000001 Nyzo tx fees
NA:nCLASS_NAME:UID:Comment
Sender data
Tokens protocol
nCLASS_NAME:UID
Data

NA: RULES

- CLASS_NAME – The class name of the NFT to add admin data to

  This class name has to exist
  CLASS_NAME has to obey the general CLASS_NAME rules about charset, uppercase and size.
  The sender needs to own this precise NFT CLASS_NAME

- UID

  UID Obeys the UID rules
  This UID can be empty – In that case data is meant for all instances of this class name.
  This UID does not necessarily exist already

- Recipient

  Used to encode the data payload to store. 32 bytes.

- Comment

  String, Optional.
  a..z, A..Z, 0..9, _ (63 different chars)

- Whole rules – Apply on the whole NA: transaction

  (Regular Nyzo tx rule)
  Max total length is 32 chars, including “NA:” and the “:” separators

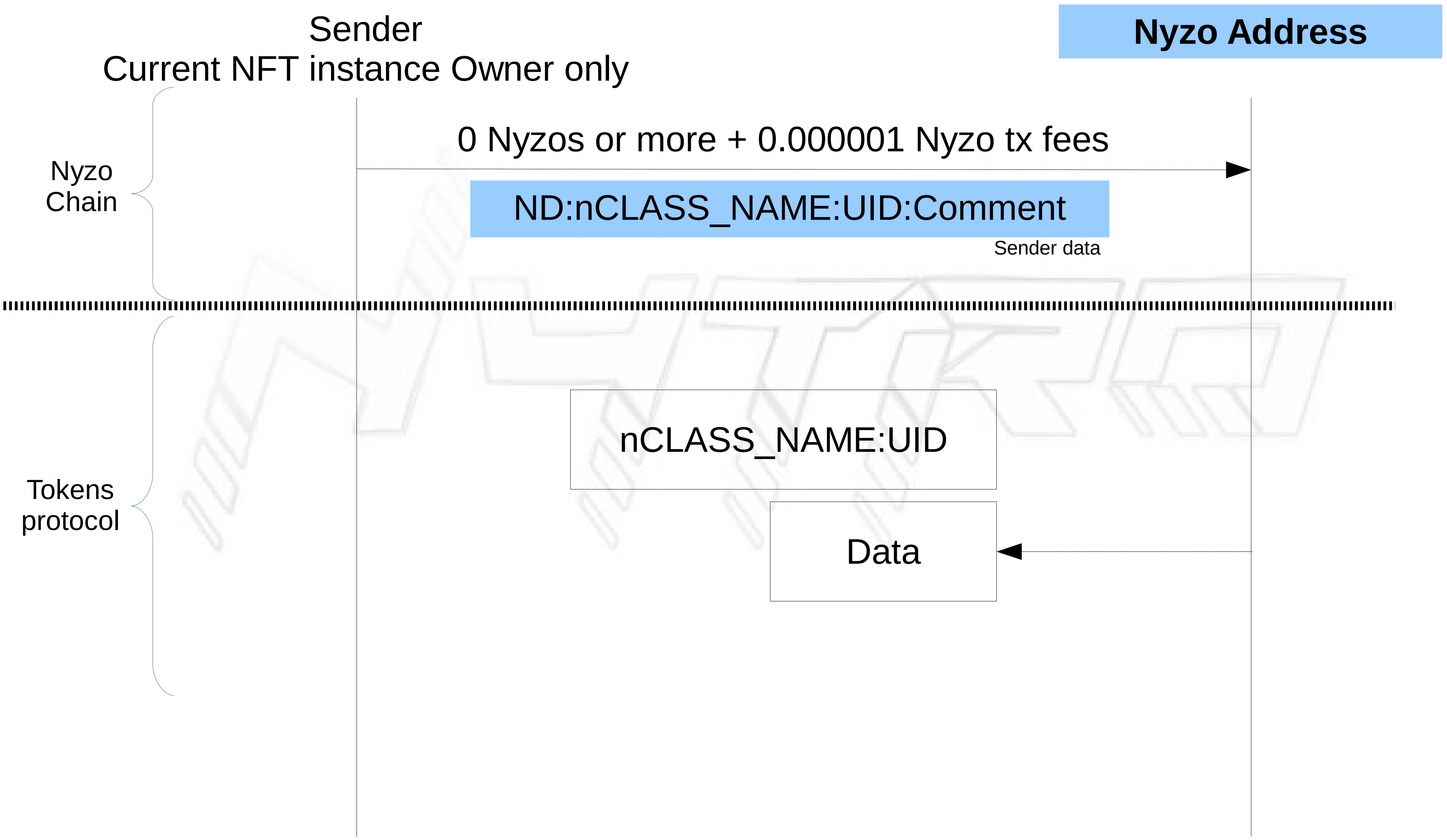
- Encoding and interpretation of the address and comment as data is left to the application layer.

NA: Validation flow and rules

- Trigger: Data begins with “NA:” - case sensitive.
- Split by “:” delimiter
- NA_R0: consists of 3 or 4 segments “segment0:segment1:segment2” or “segment0:segment1:segment2:segment3”
- NA_R1a: segment1 follows CLASS_NAME rules
- NA_R1b: CLASS_NAME NFT class exists and the sender owns it
- NA_R2: segment2 follows UID rules or is empty
- The following rules only apply if segment3 is defined
- NA_R3: segment3 obeys “comment” rules

- Note: The order in which the rules are evaluated is not important since any broken rule renders the token transaction invalid. In practice and for performance reasons low costs rules may be tested first. A canonical test order will be provided for test suites.

NFT User Data – ND:


Sender
Current NFT instance Owner only
Nyzo Address
Nyzo Chain
0 Nyzos or more + 0.000001 Nyzo tx fees
ND:nCLASS_NAME:UID:Comment
Sender data
Tokens protocol
nCLASS_NAME:UID
Data

ND: RULES

- CLASS_NAME – The class name of the NFT to add user data to

  This class name has to exist
  CLASS_NAME has to obey the general CLASS_NAME rules about charset, uppercase and size.

- UID

  UID Obeys the UID rules
  This UID has to exist and be owned by the sender

- Recipient

  Used to encode the data payload to store. 32 bytes.

- Comment

  String, Optional.
  a..z, A..Z, 0..9, _ (63 different chars)

- Whole rules – Apply on the whole ND: transaction

  (Regular Nyzo tx rule)
  Max total length is 32 chars, including “ND:” and the “:” separators

- Encoding and interpretation of the address and comment as data is left to the application layer.

ND: Validation flow and rules

- Trigger: Data begins with “ND:” - case sensitive.
- Split by “:” delimiter
- ND_R0: consists of 3 or 4 segments “segment0:segment1:segment2” or “segment0:segment1:segment2:segment3”
- ND_R1a: segment1 follows CLASS_NAME rules
- ND_R1b: CLASS_NAME NFT class exists
- ND_R2a: segment2 follows UID rules and can't be empty
- ND_R2b: CLASS_NAME:UID NFT Instance exists and is owned by the sender
- The following rules only apply if segment3 is defined
- ND_R3: segment3 obeys “comment” rules

- Note: The order in which the rules are evaluated is not important since any broken rule renders the token transaction invalid. In practice and for performance reasons low costs rules may be tested first. A canonical test order will be provided for test suites.

NFT Data: Important note

Nyzo is **not** fit for large data storage.

Data field for a single transaction is 32 bytes max compared to 186 bytes of a transaction with empty data.

This means storing 32 data bytes really stores 186 + 32 = 218 bytes.

See http://tech.nyzo.co/dataFormats/transaction for reference.

Recipient address is 32 bytes as well, and can be used in some cases, see protocol detail.

Encoding of the payloads into these 32 bytes is left to every app dev.

We will provide some examples as guidelines only.

We urge you not to abuse the system with many transactions for larger payloads. This would be a significant waste of resource, and go against Nyzo aim for efficiency.

If you need a large payload, the logic would be to use ipfs hash for instance.

Since a sha256 ipfs hash is 32 bytes, it can go into a single transaction data, and hold larger data in a ipfs hosted json for instance.

A Nyzo specific distributed data storage service could emerge later on.

DEVELOPERS ANNEXES

NYTRO

NyzoSy and Iyomisc

NYZO

Version 2.01

Online Helpers

We provide Helpers, using the reference implementation and current block, at https://tokens.nyzo.today/helpers/

For instance:

- NFT Class Issuance Helper
  - Tells whether the name you entered follows the NFT rules
  - Checks the NFT Class does not already exist
  - Select the supply type (limited or not)
  - Provides the pre_ nyzostring to sign and forward to issue that NFT class from your wallet
- NFT TRANSFER
  - From your balance page on tokens.nyzo.today
  - Provides a validated pre_ nyzostring to send a specific NFT to a recipient
- Transaction tester
  - Tells whether a token or NFT transaction is valid, and if not: why.
- NFT Explorer
  - A full NFT explorer is also available at https://tokens.nyzo.today/explorer/

NFTs API

NFT explorer features will be available as an API

Tokens Python Class

In addition to the specs and workflows, we also release a reference Python implementation of the protocol.

The “Token” class takes Nyzo transactions as inputs and validates them against the Nytro protocol rules.

Valid transactions are stored in a MySql database, Tokens and NFT states updated.

All you need to reconstruct Tokens state is to feed it the Nyzo history from the first Nytro valid block, 10650000.

The MySql database can then be used to power a Token/NFT explorer or any Token/NFT based service.

This class is the reference implementation and has been used to validate the test vectors.

See Github https://github.com/Open-Nyzo/Project-Nytro

Reserved NFT Classes

The following NFT Class has been issued as unlimited supply NFT:

- nDEMO

This will serve as test NFT for anyone willing to experiment with Nyzo NFT. A faucet will provide them for free.

The following NFT Classes

- nYZO
- nNYZO

Have been issued as unlimited supply NFT as well to avoid confusion and hostile cybersquatting.
They are likely to be used for Nytro NFT marketing and demo projects.

References

- Official Github for Nytro Protocol: https://github.com/Open-Nyzo/Project-Nytro

- Current Nytro aware applications and services:
  - First Token Explorer by Iyomisc https://tokens.nyzo.today
  - Tokens API https://tokens.nyzo.today/api
  - Nyzocli command line client, since 0.0.9 https://github.com/EggPool/NyzoCli

- Current Nytro NFT aware applications and services:
  - WIP