XCM NFT Proposal

Motivation **2**
Background **2**
XCM Asset Transfer Architecture 3
Challenges and Solutions 4
Collection Registration 4
Addressing of NFTs 4
Fee estimations 4
Existing UI Implementations of XCM 5
Wallets 5
Talisman 5
Subwallet 5
Nova 5
Fearless 6
Bridges 6
API Approaches 6
SDK vs. Client API 7
Milestones and Schedule **8**
Milestone 1 - R&D and POC 8
Milestone 2 - Integration Tools 9
Milestone 3 - New Asset Transactor integrations 10
Milestone 4 - XCM Asset Metadata Operations RFC 10
Schedule 11

Motivation

Interoperability is one of the key value propositions of Polkadot. With XCM and numerous bridges this value became possible. But they are all delivering interoperability only for fungible assets. For Polkadot to fully deliver on its promise, NFT XCM capability is a must have. Moving NFTs between different parachains will unleash the capabilities for the next generation of Web3 applications that are not constrained by any one chain. Current bridges that support NFT transfers have many limitations and a fundamentally new technology is needed to enable trustless and efficient cross chain transfers of NFTs. Polkadot is well positioned to be the basis of this technology, and this proposal will underline how we see that happening.

Background

NFTs as a technology are very different from fungible tokens. Just like in the real world, it's not only the amount that matters, and for them individuality of each NFT (and its description on chain) is what makes them valuable. Building scalable infrastructure for this asset class that transfers them between chains is therefore a separate and much more focused task. Since early 2023 Parity XCM team and NFT team members, Unique Network core team and participants from the ecosystem have contributed to the discussions on what NFT XCM should do and how it should do it. This proposal is the result of their work, as we believe that most of the critical questions have been answered and the project can be publicly evaluated by the Polkadot ecosystem members. We have considered many options for each problem we need to solve, and the resulting proposal is made after they were carefully considered.

Key requirement for NFT XCM is that it should serve all parachains that have NFTs, and due to Polkadot's flexibility there is a number of ways to implement them - there are various Substrate NFT pallets, and 2 smart contract NFTs (Solidity and Ink!). While there may be other ways parachains implement (or want to implement NFTs), these are solid bases for any future needs and are the focus of this proposal. Here is the current inventory of Parachains and the pallet/SC they use, please come forward with info if it's missing here or if correction is needed. If there are any other solid technology options beyond these, we will consider them, so if there's anyone left out - please touch base. Key requirement for the inclusion is that it's a technology that works on a Polkadot parachain, since XCM can not support anything else at this point, and bridges would

be needed for it. Variations of the implementation under proposed base technologies should be addressed in scope of each of them.

Since NFT XCM purpose is to serve all parachains in the way that's optimal for the relay chain, its proposed implementation is public and every stakeholder can participate in it. Proposals for requirements, technical decisions or implementation participation are desirable, all past actions were included in our work so far. With publication of this proposal we invite all interested parties to come forward and express their interest in any of these elements.

From the roadmap and technical perspective our proposal is focused on being practical, and moving from simple to more complex tasks both organizationally and technically. The work under this proposal has a significant impact on standards, and they need to be agreed to in order for the technology to work. Most of organizational work that needs to be done to have a good set of standards should be done in the Milestones 2-4 of the proposal while the simpler portion of the work is done in parallel, which should help to keep the focus of the overall effort on getting things done in a reasonable time and without spending too much resources on it.

XCM Asset Transfer Architecture

The processes of sending fungible or non-fungible assets across parachains are the same. We need the same XCM programs for non-fungible assets as for fungibles. The only difference is the fungibility of the MultiAsset XCM parameter that identifies the asset being transferred in the XCM message: The NFTs have no amount but have an asset instance identifier.
This difference is crucial and must be appropriately handled by the Asset Transactors on each participating chain for NFT XCM transfers to work. An Asset Transactor is an implementation of the XCM TransactAsset trait that provides an interface for the XCM executor to perform asset withdrawals, internal transfers, and deposits.

Before December 20th 2023, the transfer of non-fungible assets through XCM was impossible. We created the first implementations of AssetTransactor capable of NFT transfers for the Quartz and Karura networks and made the first cross-chain NFT transfer. See the "Milestone 1 - R&D and POC" section for details.

However, implementing Asset Transactors is not the only challenge to bring XCM NFT transfer to life since there are questions about foreign NFT collection registration, asset recovery, fee estimation, etc.

Challenges and Solutions

Collection Registration

Similarly to fungible assets, NFT collections need to be somehow registered on the destination chain so that NFT collection on the chain of origination can be matched to an NFT collection on the destination chain. There may be different approaches to collection registration, both permissioned and permissionless relative to chain governance and collection ownership.

Addressing of NFTs

The complex structure of XCM commands, as well as flexibility and liveness of the Polkadot ecosystem in general require that NFT assets are addressed as unique objects within the Global Consensus so that Blockchain of origination, the collection, and NFT ID in the original collection can be unambiguously identified at all times.

Fee estimations

Every XCM program starts with a payment to the destination chain(s) The unused portion of this payment is returned. If the allocated payment is not adequate for execution, the asset is trapped in a special place on the destination chain and needs to be recovered. This creates a problem for the end user who now needs to take additional actions (possibly even resolving the issue through the destination chain governance) in order to gain access to their asset. In order to avoid the situation, fees need to be estimated before sending the XCM transaction, but there is no mechanism in XCM through which such estimation may be done.

We offer an architectural solution to the problem stated above. The fees may be estimated using a destination chain RPC call by the client library at the time it composes the XCM program. So, the execution of the simplest XCM transfer will look like following:

1. Client queries fees from destination chain via the RPC call for the exact XCM NFT transfer command it is going to execute
2. Client composes XCM program using the fee amount reported by the destination chain

3. Client initiates the XCM program execution

Existing UI Implementations of XCM

This review is focused on gathering interested parties together, overviewing their integration solutions in terms of APIs and client libraries being used, and diving into their challenges related to XCM.

As we can see, the fee estimation challenge and having a well supported and stable SDK / API supporting mobile platforms are of the most importance.

Wallets

Talisman

API used: Polkadot JS API
Rationale behind API choice: It is used within the Acala Bridge SDK that was created to facilitate cross-chain transactions via XCM.
Challenges: (1) Reconciling the several XCM "wrapper" pallets that seem to exist: pallet-xcm, xTokens, etc., (2) Staying up to date when HRMP channels open.

Subwallet

API used: Polkadot JS API
Rationale behind API choice: We just use plain Polkadot-js libraries to construct XCM messages. We handle the XCM logic by ourselves. So far, it's the best approach for us since using any external library means we would have to rely on other people to make the update in time.
Challenges: It's very painful to maintain XCM every time there's an update. I've seen other teams running into the same problem. Not being able to estimate gas fee on the destination chain is also a big caveat. Finally, we don't have a main channel that teams can share about XCM updates, so more often than not, we only find out about update when it's already released

Nova

API used: Own implementation for iOS and Android
Rationale behind API choice: No API provided by Parity for mobile

Challenges: The main challenge is XCM fees. The easiest way to add chains is if the fee calculation is proportional to the base fee.

Fearless

API used: Own implementation for iOS and Android
Rationale behind API choice: No API provided by Parity for mobile

Bridges

There are a number of ecosystem bridges designed to transfer assets between parachains. They often share the same function of cross-chain XCM asset transfer, have similar user interface, and similar implementation that uses Polkadot JS API. This great amount of redundant code appeared due to quick ecosystem evolution (and this is normal), yet it was not efficient due to entity duplication. A more efficient use of ecosystem resources would at least share the same integration code, which will become more complex with NFT transfers. Our solution to this issue is to provide the JS library on top of Polkadot JS API that facilitates XCM transactions (including NFT), and the SDK based on that library that is capable of providing the RESTful service making NFT XCM transfers as simple as making a RESTful call.

Below are examples of ecosystem cross-parachain bridges that implement XCM transfers of fungible assets to prove our point:

Karura Bridge: https://apps.karura.network/bridge
Acala Bridge: https://apps.acala.network/bridge
Astar Cross-Chain Transfers: https://portal.astar.network/astar/assets/transfer?token=dot&from=astar-evm&to=polkadot&mode=xcm
Moonbeam Parachain Bridge: https://apps.moonbeam.network/moonbeam/xcm

API Approaches

When sending an XCM command, the interaction between the UI and the chain of origination can be implemented in several ways, which can be grouped into two high-level approaches:

1. Client library, for web applications based on Polkadot JS API or for mobile applications based on the concrete implementation of client
2. The web service (such as sidecar or Unique SDK) that abstracts chain integration behind a fixed RESTful API and stays responsible for timely updates of its own integration with chains, we call this approach “SDK”.

The table below contains comparison between the approaches, both of which have arguable pros and cons and should be selected with consideration of the business, operations, and technical requirements of a concrete application.

SDK vs. Client API

SDK Pros	**Client API Pros**
● Simplicity of RESTful services ● Lower page loading times due to small client JS size ● Faster connection time between client and the node ● Less support efforts due to guaranteed interface ● Shorter go-to-market timeline ● Can be used to protect node RPC/WebSocket endpoints from DOS attacks	● Less code dependencies ● Easier portability for supporting new parachains ● Enables serverless architecture “out of box”
SDK Cons	**Client API Cons**
● Either need to host backend, or sacrifice client complexity, or use a third party hosted backend ● Susceptible to DOS attacks due to lack of decentralization ● Serverless architecture is more complex	● Integration and support complexity ● Breaking changes will cause application downtime ● Node RPC/WebSocket endpoints have to be exposed publicly which will incur high infrastructure costs in case of a DOS attack

Milestones and Schedule

Milestone 1 - R&D and POC

We have started the work on this project with a broad research of all issues related to cross-chain NFT transfers. First and foremost question that was raised is the data model to which all participating parties need to stick: Whether this is a reserve based model when the Reserve parachain is responsible for NFT integrity at all times, a teleportation model when all NFT data is completely transferred to the receiving chain and control is handed over, or some combination of these models in between. The model discussion on the Polkadot forum involved community members and even attracted the attention of Gavin Wood. This was an important discussion because all ecosystem stakeholders needed to eventually agree on the XCM NFT transfer standards. Soon after we realized the importance of the fee estimation issue and started considering different solutions to it. One of the solutions was building the XCM program complexity language that would allow to pass the weight calculation formula between parachains, but this approach got invalidated because of two reasons: (1) complexity, and (2) difficulties related to formula updates in case if some parachain's XCM handling logic changes. Finally, we realized that the asset recovery issue was also important and suggested a Custom Asset Claimer RFC Proposal that was also discussed in this thread on the Polkadot Forum. The findings of our research was summarized at the sub0 conference in September 2023. The presentation and the transcript is available for the review.

To make the process of enabling XCM NFT transfers for the Polkadot ecosystem more iterative and achieve tangible results sooner, we have built the POC of NFT transfer first that only involves ID-based transfer strictly within the reserve based model, leaving NFT metadata intact, since this is the simplest possible form of cross-chain NFT transfer that is already possible with XCM v3. The next step would be to create a toolset that enables other parachains to transfer NFTs, independently of the NFT technology they are using, be it the NFT pallets or EVM smart contracts. Further on we will initiate the discussion, research and development of cross-chain interaction with NFT metadata.

The preparation R&D work described above was a prerequisite for building the POC. This work shows POC value and rationale. The end goal of this milestone is to build a working on-chain solution that allows sending NFT tokens (by ID only) between Acala and Unique networks and produce a practical output of the research we have done for consolidation of our achievements and for enabling further steps. On the implementation level, it extends the XCM AssetTransactors for depositing, withdrawing, and transferring of NFT assets. The parachains involved were chosen to cover the core Substrate use case - XCM transfer between different NFT pallets. Other options will be handled in further milestones.
As we finalize this proposal, the Milestone 1 has been completed with the successful proof that the proposed architecture and approach works. Milestone 1 report is available here.

Milestone 2 - Integration Tools

The goal of this milestone is to provide necessary tooling for creating UIs and for parachain integration. The deliverables include:

- JS library based on Polkadot JS API
    - Composing and sending NFT XCM transactions
    - Requesting fee estimations
- Integration SDK for creating UIs and for wallet integration.
- UI for transferring NFT assets, which is open source and well documented, uses integration SDK, and initially supports transfers between Acala and Unique, but is flexible to include other networks. The UI will support following features (through SDK):
    - Transferring NFTs between networks
    - Capability of NFT Asset recovery. But this is possible only if the "Custom Asset Claimer" RFC is enacted.
    - Fee estimation
- The implementation of fee estimation RPC
- Refinement of the Asset Transactors implemented during the PoC stage: implementing the generalized XNFT pallet that can be integrated into any Substrate chain.
    - Writing an adapter of the generalized XNFT pallet for ORML nft pallet
    - Refinement of the Karura XCM NFT transactor: replacing it with the generalized implementation using the ORML adapter.

    - Refinement of the Quartz XCM NFT transactor: integrating the generalized XNFT pallet.
- Integrating the NFT Asset Transactor to the Aventus network using the generalized XNFT pallet.
- Documentation and education materials
- R&D: Continue the research on how to implement Asset Transactors for EVM chains
- R&D: Research how EVM integration affects the XCM NFT Metadata RFC

Milestone 3 - New Asset Transactor integrations

Integrating and/or adapting the asset transactors to:

- EVM parachains
- Pallets developed by interested parties such as “NFTs” by Parity
    - **Note:** The Uniques v1 pallet is not included because (a) it is deprecated, and (b) we see the signs in the polkadot community (https://github.com/jsidorenko/nft-migrator/) that NFTs may be migrated from it to NFTs pallet.
- R&D: Continue the research on how to implement Asset Transactors for Ink! chains
    - R&D: Research how ink! Integration affects the XCM NFT Metadata RFC

Milestone 4 - XCM Asset Metadata Operations RFC

The main objective of this milestone is to write an XCM Asset Metadata RFC.

The RFC must give the answers for the following:

- What format of metadata should become standard in the XCM? For example, one option is some collection structure of key-value pairs since it is the most general way.
- What metadata operations should become a part of the XCM instruction set?
  The possible options are:
    - Metadata Update – The sender chain sends the metadata write request to the target chain regarding a specified asset. It is up to the target chain how to handle the request.
    - “Request metadata update permission” – The sender chain requests permission to modify the metadata on the target chain. The target chain determines keys to which the update permission could be granted.

- "Allowed to update metadata" Notification – The sender chain notifies the target chain that the target chain has permission to modify specific metadata keys of a particular asset.
- Metadata Query – The sender chain queries the values of the specific metadata keys of a particular asset from the target chain.

Also, during this milestone, we need to find out what API should be added to the most common XCVM implementation, namely, the XCM executor by Parity, so parachains can easily implement handling logic for XCM metadata operations.

Schedule

Milestone 1 - completed on Dec 20, 2023

Milestone 2 - 6-9 months after proposal approval

Milestone 3 - 3-4 months after Milestone 2

Milestone 4 - total 3-4 months, in parallel with Milestones 2 and 3 or immediately after

Estimated completion - end of 2024, with partial deliverables as completed.

Budget and resources

See the file for details:

XCM NFT Proposal Budget

Team: 3 Substrate Developers, 1 UI Developer, 1 Architect, 1 Manager, 1 DevOps, 1 QA, plus external teams for EVM, Ink!, educational content and use cases bounties

Future work: Implementation of NFT Metadata XCM transfers, updates to tooling and UI components. ERC-4626 and similar missing formats.