Technical Documentation for Minstordinals

Galois Field, Noé de Larminat

December 28, 2023

Contents

1 Introduction

1.1 Purpose of the Technical Documentation

The purpose of this technical documentation is to provide comprehensive information about the Minstordinals project. It is intended for developers, users, and anyone interested in understanding and implementing Minstordinals. This documentation covers the technical aspects, installation, configuration, usage, and more, enabling readers to navigate and utilize Minstordinals effectively.

1.2 Prerequisites

Before diving into the technical details and implementation of Minstordinals, it's essential to be aware of the prerequisites necessary for working with this project. Here are the prerequisites:

1.2.1 Hardware Requirements

As Minstordinals is still under development we are purposing here the maximal requirement for us to generate fully this service.

- Adequate storage space for the application and data. Bitcoin full node and `ord` indexed (around 650Gb in total). Moreover, storage for `minstordinals` indexing ;
- A reliable internet connection for interacting with blockchain networks. Calls to archival nodes, bitcoin rpc APIs and ordinals indexers ;
- Enough computer power for the Verifier (BOS workflow verification process).

1.2.2 Software Requirements

To install and run Minstordinals, you'll need the following software:

- Operating System: Minstordinals is platform-agnostic and can be used on various operating systems, including Windows, macOS, and Linux.
- Development Environment: If you plan to make modifications to the code or develop additional features, you should have a code editor or integrated development environment (IDE) installed.
- Dependencies: Minstordinals relies on specific near and taproot/ordinals software libraries and tools, which would be covered in the "Installation and Setup" section.

1.2.3 Skills and Knowledge

While Minstordinals aims to be accessible to a wide audience, having the following skills and knowledge will be advantageous:

- Basic understanding of blockchain technology, NFTs (Non-Fungible Tokens), and Bitcoin Ordinals. Understanding of `minsta` framework ;
- Familiarity with programming languages, mainly `javascript, typescript` and `Rust` to modify the code ;
- Knowledge of how to use command-line tools for software installation and management. Mainly `ord` and `near-cli-rs`.

1.3 Terminology

To facilitate understanding, let's clarify some of the key technical terms and jargon used throughout this documentation:

- **Bitcoin Ordinals/inscription**: Encoding data directly into the Bitcoin's transaction outputs through Ordinal enveloppe ;
- **JSON-based protocol**: A standardized format for representing protocol deployed on Bitcoin ordinals transactions.
- **Cross-Minting process**: The process of creating or issuing NFTs, with a unique token on near and corresponding inscription on Near Protocol ;
- **Verification**: Confirming the authenticity of a `minstordinals` Bitcoin inscription.

Throughout this documentation, these terms will be used consistently to explain and describe the various aspects of Minstordinals.

2 Installation and Setup

Still under consideration. Each tech are not decided.

2.1 System Requirements

2.2 Installation Process

We should use a client server model. Where the client should be built with `npm` and server in Rust compiled through `cargo`.

2.3 Configuration

It will depend about services used to rely each data. We can built our own nodes to rely on ourselves or use tiers-API/RPC to access required data.

3 Workflow

Workflow is mainly divided into 2 parts : a minting process and a verifying process. Last workflow should only be the user front-end workflow not designed for now.

Those workflows are now illustrating the purpose without implementing them. The implementation process will be discussed in Protocol Details **??**.

3.1 Mintbase workflow

The Mintbase workflow covers mainly the minting process in backend from user taking a picture on Minstordinals to Bitcoin inscription.

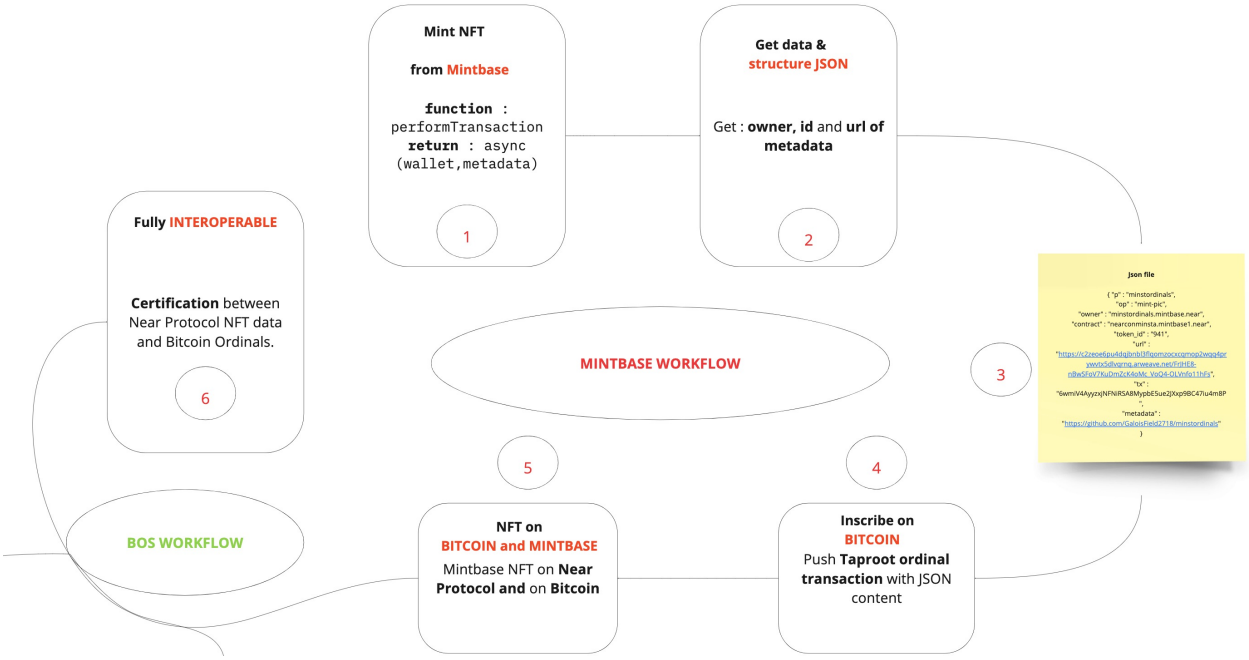

Figure 1: Mintbase workflow

3.2 BOS Verifier workflow

This workflow aims to describe the effective way to verify veracity of data pushed on Bitcoin and produce a proof available on Bitcoin. For a Bitcoin indexer it can consider only verified inscription.

Provide this two steps allow a more open free mint character rather than compact this both transaction into only one.

Making use of BOS tools allows the verifier to access Near on-chain data, use Near notifications tools and can allow smooth integration into others Near app like near.social.

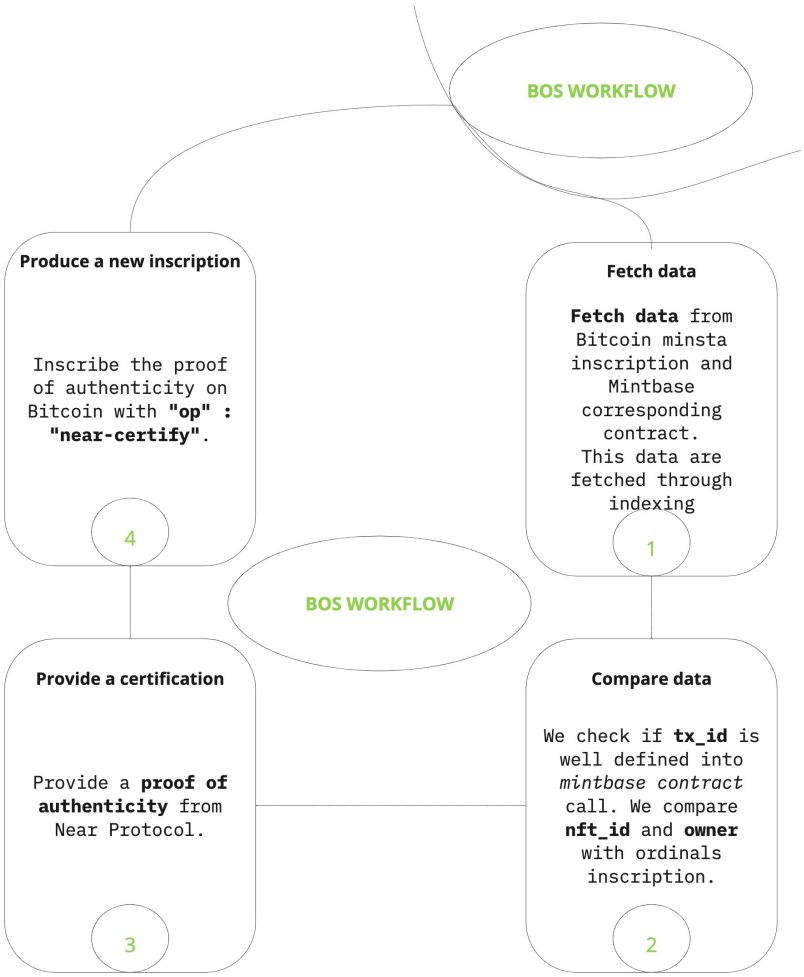

Figure 2: BOS Verifier workflow

3.3 Global overview

This schema is the overview of the backend process to mint and verify all of these cross chain operations.

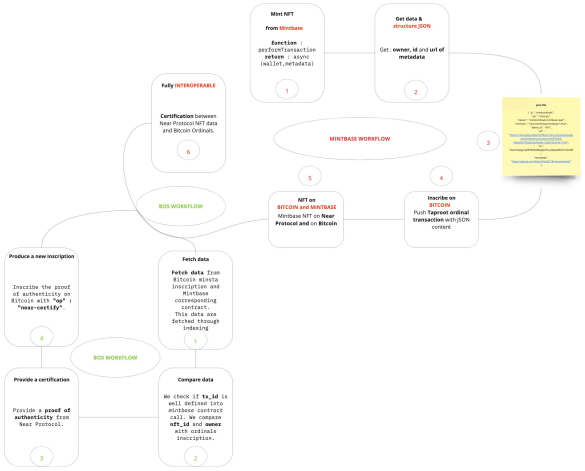

Figure 3: Global workflow

4 Protocol details

This part aims to provide the JSON based files for our Minstordinals protocol.

We describe keywords and important data to catch in the first part. In a second part we look into operations offered by the protocol and structure. Finally we discuss data usage for indexing purpose.

4.1 File Description

4.2 Operations

4.3 Discussing data

4.4 Programming Languages

4.5 Libraries and Tools Used

List the libraries and tools used in the development of Minstordinals.

5 Further Implementation

Discuss any challenges and their solutions. Outline the integration with BOS and how it enhances the project.

6 Usage and API

Describe how users can interact with Minstordinals, including how to mint NFTs, verify certificates, and use the Minstordinals API.

7 Examples

Provide code examples and usage scenarios to help users understand how to work with Minstordinals.

8 Technical Details

Explain the technical aspects, including data structures, algorithms, and security measures used in Minstordinals.

9 Troubleshooting

Offer guidance on identifying and resolving common issues and debugging tips.

10 Appendix

Include additional resources, references, and a glossary to aid users in understanding Minstordinals better.