Research Proposal

# Exploring ActivityPub interoperability for Bluesky

Martin Sonnberger

December 16, 2025

## 1 Introduction

Mastodon and Bluesky are two popular microblogging social media platforms that present themselves as open, decentralized alternatives to traditional platforms such as X (formerly known as Twitter). Bluesky was built as a reference application for its underlying *Authenticated Transfer Protocol*, or short ATProto, which was designed as an alternative to the established *ActivityPub* protocol that forms the foundation of Mastodon and the Fediverse. ActivityPub is built around an inbox/outbox model, similar to that of email. A new user has to choose one of many available servers (instances) when creating a new account, after which they are able to interact with users from all participating instances. ATProto on the other hand features multiple independent components, each of which can be swapped out and self-hosted as long as it follows the protocol's specifications.

While Mastodon has a smaller, but loyal user base of around 10 million users[1], Bluesky experienced rapid growth, especially after the 2024 US presidential election, with a current user count of around 40 million.[2] Since both platforms are established in the wider market, the desire for individuals, companies, and institutions to be represented on both Mastodon and Bluesky has grown. However, maintaining multiple online presences leads to additional work for content creators. To reduce this duplicated work, bridging solutions such as Bridgy Fed[3] have emerged. Bridgy Fed provides a third party bridging service that sits in between Bluesky and Mastodon, translating posts from one network to the other by using synthetic bridged accounts on behalf of users that sign up and opt in to use the service.

[1] https://mastodon-analytics.com/
[2] https://bsky.jazco.dev/stats
[3] https://fed.brid.gy/

In this thesis, we want to take a different approach. Instead of a separate bridging service, we want to explore making Bluesky itself interoperable with ActivityPub and thus all Mastodon instances. We therefore describe an ideal interoperability state that outlines how the user experience for users on both Bluesky and Mastodon would look like if Bluesky had full ActivityPub compatibility. We then package the solution into three work packages with increasing complexity that aim to achieve this ideal state. Finally, we answer the question of how close the ideal interoperability state can be reached with reasonable engineering effort and no protocol or implementation changes in ActivityPub and Mastodon.

# 2 Ideal state

The goal is to achieve seamless "network dual-citizenship", where a modified ATProto Personal Data Server (PDS) functions simultaneously as a native Bluesky PDS and a fully compliant ActivityPub instance. In this state, the protocol boundaries become invisible: the user maintains a single identity that can publish, subscribe, and moderate interactions across both the AT-Proto and ActivityPub social graphs without relying on third-party bridging services. Interactions include liking, replying, and reposting.

Users perceive a unified timeline where posts from both sources are shown, including conversation threads involving participants from both platforms. This interoperability extends beyond data exchange to include governance: user safety controls, such as blocks and content moderation flags, are federally enforced, ensuring that the user retains full agency and protection across both networks.

# 3 Work packages

First, we define the three participating user groups:

1. **Mastodon users:** These are regular users having an account on one of many Mastodon instances, for example `mastodon.social`. They will

be able to interact with federated Bluesky users. The representative of this group is *Mike* and their Mastodon handle is `@mike@mastodon.social`.

2. **Federated Bluesky users:** These are Bluesky users with an account on our modified PDS, hosted on `fedisky.social`. Their accounts will be federated, making their profiles and content visible on Mastodon. They are able to interact with other Mastodon users as well as regular Bluesky users on other PDSs. The representative of this group is *Frank*, and their handles are `@frank.fedisky.social` and `@frank@fedisky.social` for Bluesky and Mastodon respectively.

3. **Regular Bluesky users:** These are Bluesky users with an account on some other, non-federated PDS, such as the default `bsky.social` PDS. They will only be able to interact with other Bluesky users, including federated Bluesky users. The representative of this group is *Bob* with their Bluesky handle being `@bob.bsky.social`

## 3.1 Package 1: Outbound

The first work package is about making federated Bluesky users and content available on Mastodon. It forms itself around the following user story:

> *"As a federated Bluesky user, I can give my handle to a Mastodon user. They can follow me, see my posts, and interact with them. I can see their interactions in my Bluesky notifications."*

This package implements the core infrastructure required to make the federated Bluesky user discoverable and viewable on Mastodon. In order to achieve this, we need to make the following changes to the Bluesky PDS:

1. **Identity & discovery:** WebFinger endpoint that maps the ATProto handle to an ActivityPub Actor URL and document. This allows discovery of the federated Bluesky user on Mastodon (i.e. appearing in search results). An additional `/.well-known/nodeinfo` endpoint provides information about the server and advertises itself as a Fediverse node.

2. **Outbound broadcast:** When a Bluesky post is created on the PDS, it gets converted to an ActivityPub *Note* and delivered to the inboxes of all known Mastodon followers. Replies by federated Bluesky users on the same PDS can also be delivered via ActivityPub; replies from other Bluesky users will not be visible in Mastodon at this point. Additionally, all the federated Bluesky user's posts can be fetched from ActivityPub using a federated Bluesky user's `outbox` endpoint.

3. **Inbound listening:** Accept *Follow*, *Like*, and *Create(Note)* (reply) activities. Follows will be stored, and push notifications will be created for likes and replies. The sequence diagram in figure 1 illustrates what happens when a Mastodon user follows a federated Bluesky user.

## 3.2 Package 2: Inbound

For this next step, we want to bring Mastodon content into Bluesky, by implementing the following user story:

> *"As a federated Bluesky user, I can search for a Mastodon user inside my Bluesky app, follow them, see their posts in my timeline, and interact with them."*

This package deals with data intake and storage. The main challenge is that one cannot write external data into the federated Bluesky user's signed Merkle Search Tree (MST), the data structure that holds all user data. Instead, we need to build a secondary storage system into the PDS that stores all incoming ActivityPub activities. The following steps outline the features of this work package:

1. **Virtual users:** When a federated Bluesky user searches for `@mike@mastodon.social`, the PDS must fetch that Mike's profile and create a temporary "shadow profile" in the local database such that the Bluesky client can render the profile data in the UI. The federated Bluesky user can follow that profile, resulting in a *Follow* activity being sent to their instance.

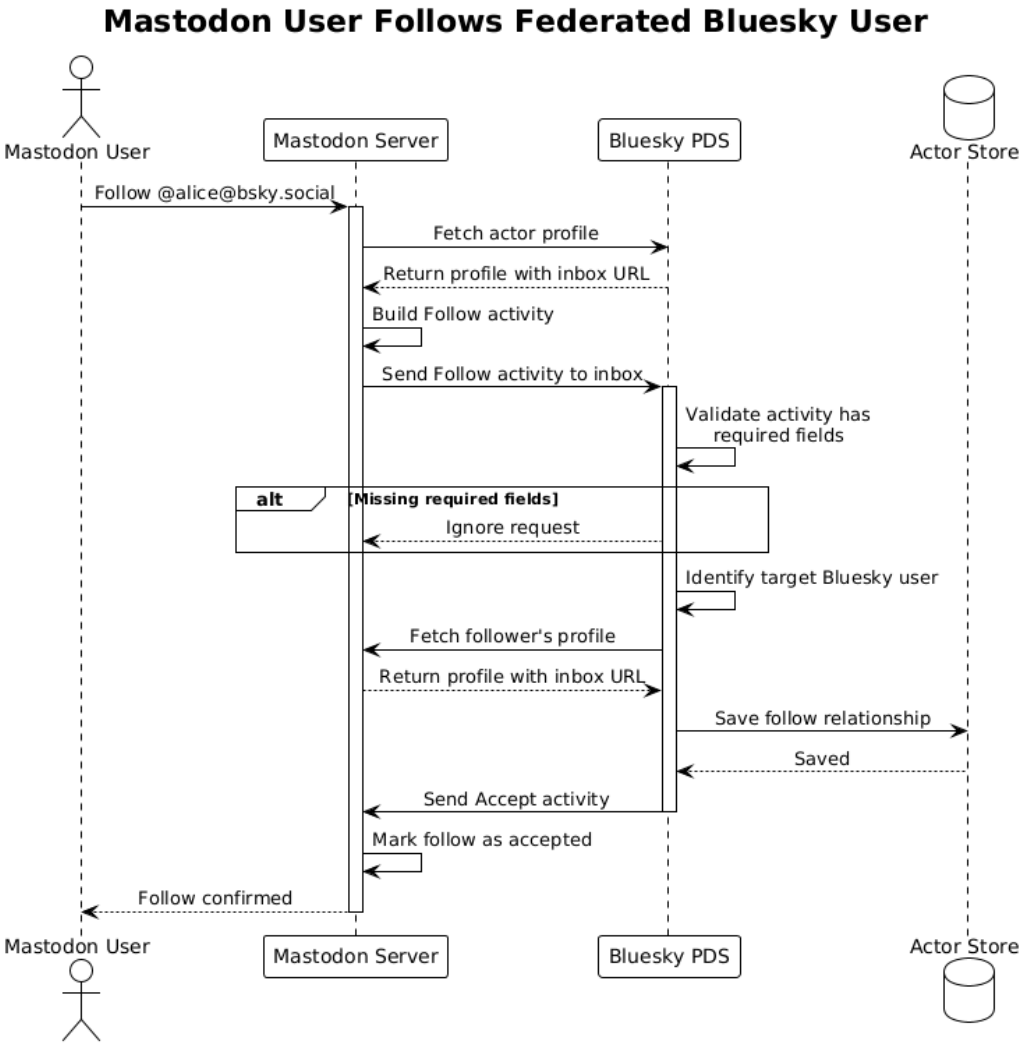


Figure 1: Sequence diagram: Mastodon user follows federated Bluesky user.

2. **Shared inbox:** To scale efficiently, we implement a shared inbox for the whole server. Requests to this inbox include further information about routing activities to specific federated users.

3. **Custom feed generator:** We can implement a custom Bluesky feed that queries the sidecar database for Mastodon posts and merges them with local posts, enabling a hybrid feed with posts from both networks.

## 3.3 Package 3: Governance & Advanced Features

This package refines the user experience and ensures the PDS behaves safely within the wider ecosystem.

1. **Moderation federation:** If a federated Bluesky user blocks an account in Bluesky, the PDS must broadcast a *Block* activity to ActivityPub. On the other hand, if a PDS receives a *Flag* (report) activity from Mastodon, it should ingest it into ATProto's moderation system. Similarly, content moderation (labels in Bluesky, content warnings in Mastodon) should be seamlessly translated between networks.

2. **Account migration:** Implementing the ActivityPub *Move* activity, allowing a federated Bluesky user to forward their Mastodon followers to a different Mastodon instance if they choose to leave our PDS.