Snowbridge Major Milestone Completion and Long Term Success Funding

Executive Summary

Total amount: 505157 DOT and 2812500 USDC *(~5,797,500 USD, based on the current Subscan EMA7 rate of 6.22, with a 5% buffer for slippage)*, broken down into:

- **Audit & Operational costs:** Single payout of 187849 DOT *(based on $1,110,000)*. Paid on execution of proposal
- **Milestone completion bonus:** First 12 out of 24 months vesting payouts, covering the period from July 2024 to July 2025. 13221 DOT per month, totalling 158652 DOT. Scheduled for monthly claim, starting immediately. *(based on $937,500)*
- **3-months post launch bonus:** Single payout of 158656 DOT *(based on $937,500)*. Scheduled for claim 3 months after launch.
- **Successful operation incentives:** First 9 out of 21 months payouts, covering the period from July 2024 to July 2025. 312500 USDC per month totalling 2812500 USDC.

Beneficiary Address: 129Uxho5oKBz6js8dwjoQLzLjYsEs4ZLBtS7KRotQbunVNfP

Work covered:

- Payment for auditing costs *(both retroactive and future 2024 costs)*
- Sponsorship for light client relayer costs for 2024 *(both retroactive and future through ~July 2025)*
- Partial payment for completed engineering and launch milestones *(12 out of 24 months vesting payment)*
- The first 9 out of 21 months of incentive payments, based on the schedule agreed upon in our original proposal, for longer term success in maintenance and development work on the bridge.
- Insurance for users of the bridge and the wider Polkadot community.

Snowbridge, the first fully trustless Polkadot <> Ethereum bridge is live. We've completed all our milestones from our original proposal last year and are set up for the long term success of the bridge.

Demo (on Polkadot): https://app.snowbridge.network/

See the Completed Work and Future sections below for a summary of our completed milestones, last year of work and future plans.

Our last successful proposal was in 2022, where our team and the Polkadot community agreed upon our expected milestone payments and incentives. The agreed total milestone payments and incentives totalled ~$10mil worth of DOT.

This proposal sets up spends for the next 12 months of payments for our 2024 completed engineering milestones, operating costs and team incentives, from July 2024 to July 2025. It also adds additional incentive alignment controls and medium-term insurance guarantees for the bridge.

Further details on the payment and incentive structure, see the **Payout Structure** section below.

Table of Contents

Overview

Snowbridge is live on Polkadot!

Snowbridge assets can work on any parachain that is connected to Asset Hub and has added support for foreign assets. You can demo the bridge on Rococo (here) and Polkadot (here). We already have working and in progress integrations with many parachains, wallets and dapps, including Bifrost, Hydra, Kilt, Colorfulnotion, Subscan, Moonbeam, OriginTrail, Mythical and others. Bridged assets have already been integrated into SubWallet. Talisman and Nova wallet are in progress. We're excited to have the community play with it and unblock new streams of liquidity from Ethereum into and across the Polkadot ecosystem.

- Demo (on Polkadot): https://app.snowbridge.network/
- Technical Docs: https://docs.snowbridge.network/
- Main Code Repo: https://github.com/Snowfork/snowbridge

This proposal reflects on the last year and a half of progress we've made and sets up long term funding incentives for the success of the project.

Context

Snowbridge is the longest running and most complex public goods project in the Polkadot ecosystem outside of Parity itself. We've gone through multiple iterations and evolutions over the last few years as we've been waiting for BEEFY to go live: evolving our Ethereum

Light Client from Proof of Work to Proof of Stake; migrating from our own parachain to BridgeHub; shifting our architecture in tandem with XCM as it has evolved through v4 to maintain an XCM native bridge system; evolving to an agent-based architecture with fully cross-chain accounts.

Purpose

Snowbridge is a critically important upgrade for the Polkadot ecosystem - all Ethereum assets currently in the Polkadot ecosystem flow through bridges that rely on third-party signatures, ie, bridging or governance signatures from private keys, token holders or validators that do not match Polkadot and Ethereum's validator set. All parachains and dApps in the Polkadot ecosystem where these assets flow now have their trust model handicapped by these third parties, rather than taking their full security from the Polkadot Relay Chain. This undermines the value proposition of Polkadot itself.

Snowbridge's model is trustless, ie, it relies only on signatures from first-party Polkadot and Ethereum validators and transparent, permissionless and auditable on-chain logic. It runs on BridgeHub, a system parachain that draws its security entirely from the relay chain and governance directly from the DOT token.

Snowbridge's launch also represents a major opportunity for growth for Polkadot as we unlock a new wave of potential assets and interactions with Ethereum. We hope to see lots more interoperability and experimentation with cross-chain systems over the coming years built on top of Snowbridge.

For a deeper dive into Snowbridge's purpose and value to the Polkadot ecosystem, see the **"Purpose"** section in our original funding proposal (link).

Completed Work

Snowbridge is now live on Polkadot & Ethereum, deployed on BridgeHub and we've completed all our milestones as laid out in our last proposal. Beyond those milestones, there have been a ton of additional scope items that have come up and that we've completed over the last 18 months in addition to the originally planned milestones. This section reflects on all that work.

Milestone M1

Use XCM as base layer messaging protocol

Our messaging protocol has been rebuilt to be fully XCM-native. This milestone was completed for XCMv3, but we've also upgraded to support XCMv4 for launch too. Our bridge is capable of mapping, encoding and decoding between ERC-20 transfers and XCM asset transfers.

Additionally, our Transact functionality maps between arbitrary Ethereum smart contract calls and types and XCM to support communication between EVM contracts and parachain pallets.

The core XCM primitives on the Polkadot side are implemented here: https://github.com/Snowfork/polkadot-sdk/blob/snowbridge/bridges/snowbridge/primitives/core/src/outbound.rs

The core primitives on the Ethereum side are here: https://github.com/Snowfork/snowbridge/blob/3f90b1619d0bdbec8f50b63185b3dcc632321019/contracts/src/SubstrateTypes.sol

Code in our inbound and outbound channels on both sides of the bridge also include significant XCM-related logic.

This was a long, iterative process to get to - we've had to redo design and implementation of our XCM-related systems multiple times as XCM has been changing under our feet - an initial implementation for v2, then v3 but we're in a good stable state for asset transfer now with XCMv4.

For the newer Transact and Polkadot-native assets functionality, there are still some improvements that we hope will become possible once newer XCM functionality related to recovering trapped is shipped to Polkadot itself.

Mint wrapped assets on Statemint *(now renamed to AssetHub)*

Our bridge has been tightly integrated with AssetHub as our primary chain for minting ETH and ERC20 assets from Ethereum. AssetHub has a new foreign assets pallet that is intended to store balances for assets originating from external sources. We adapted our design to this pattern and so AssetHub is the primary reserve for these assets and we support typical XCM messages for asset transfer.

Snowbridge assets look and act like any other asset in the Polkadot ecosystem.

(Of course, it is possible for other parachains to be used as a reserve. For example, some projects may want the reserve for their own assets to be on their own chain rather than AssetHub - our design does support building out a custom implementation for this and we may work on this for some chains in future if there is demand for it)

Polkadot side logic for message routing and AssetHub communication: https://github.com/Snowfork/polkadot-sdk/blob/snowbridge/bridges/snowbridge/primitives/router/src/inbound/mod.rs

Ethereum side logic for asset management: https://github.com/Snowfork/snowbridge/blob/3f90b1619d0bdbec8f50b63185b3dcc632321019/contracts/src/Assets.sol

Governance on Ethereum

We've shipped a cross-chain governance system that allows Polkadot's governance to be used to control the Ethereum side of Snowbridge. This is currently used for modifying bridge parameters, bridge modes, pausing/unpausing and for upgrades to the bridge smart

contracts. This is a novel governance system that is trustless and decentralized, just like the bridge itself.

We also have a system pallet that allows for certain governance messages (like emergency pauses) to be prioritized on a special governance channel to take priority over general bridged messages.

Code for all the above is here.

Additionally, since Polkadot has a very flexible and configurable governance system that is more powerful than much of what is currently used on Ethereum, we see a lot of potential in it being used to control other systems on Ethereum beyond just our bridge. We'd love to see teams use this work to explore cross-chain governance beyond just Snowbridge and for Polkadot to become a governance driver across wider ecosystems.

Upgradable Smart Contracts on Ethereum

Our Ethereum smart contracts are all upgradable through the above cross-chain governance system. Typically, there is a trade-off between upgradability and decentralization and many teams argue against upgradability.

Our novel cross-chain governance system allows for upgradability without compromise on decentralization, as this power remains fully in control of DOT holders and Polkadot's existing governance system.

Our contracts use the ERC-1967 proxy pattern for upgrades. Our cross-chain upgrades have also already been tested and used in production - we had an initial Shell proxy contract that we deployed and then upgraded to our decentralized proxy contract as part of our launch process.

Client Library

Our client library is completed and deployed to NPM for other developers to use. In addition to this, we have also developed a demo UI for Snowbridge which is both ready for use by end-users wanting to bridge assets, as well as by developers that want to use it as a template for building their own UI integrations with Snowbridge.

Velocity Labs is using our client library to launch a planned, dedicated cross-chain webapp for all XCM and Ethereum transfers across the Polkadot Ecosystem.

We're also in conversations with HydraDX, Bifrost, Talisman, Moonbeam and Stellaswap to support them building automated bridging into their dapps using our client library.

Milestone M2

Limiter

After much discussion with Parity and potential users across the Polkadot community, we agreed that using a limiter would not actually be the best fit for our bridge. We support

arbitrary message passing and transacting, and so it would not be possible to or make sense to impose application-specific limits on this functionality.

It would be possible to place a limiter on specific assets being bridged, but this would also not be in the interest of many liquidity providers and users as it could result in delays and constrict flow of liquidity which could worry some of those users.

Instead, we've focused on more robust systems and processes for pausing the bridge in the event of any security risk or hack and powerful cross-chain governance such that DOT holders and Polkadot's governance system can directly control the Ethereum-side state of the bridge.

Security Audits

We've completed multiple rounds of security audits which all went well and cover all code currently deployed on-chain. We've also already made and deployed fixes for all security-critical findings. All of our audits can be found here:

- https://github.com/oak-security/audit-reports/tree/main/Snowbridge

Production Infrastructure

We're live in production now and our infrastructure is set up and ready across Rococo, Kusama and Polkadot.

Snowbridge now runs as part of BridgeHub using shared infrastructure with the bridges team at Parity. Our core parachain infra now runs together on the same parachain as the Polkadot <> Kusama bridge. This means that already today, multiple different parties *(at least 9)* are running bridge hub collator nodes and ensuring availability for Snowbridge's Polkadot-side functionality.

Similarly, our Ethereum contracts are deployed on Mainnet and are fully permissionless, meaning they have the same production-grade availability as Ethereum itself.

The offchain relayer component is not security critical - since our bridge is permissionless, anyone can run relayers. Having said that, our team has set up and is running a production-grade relayer stack with comprehensive logging and debugging tooling, as well as active monitoring of the state of the bridge, message delivery and state of our own relayers. We spent a lot of effort refactoring our relayers so that they can be run and monitored in a much more reliable manner.

We have set up alarms for our team with 24/7 support whenever a critical issue with the bridge is detected, including an unexpected change in latency or unexpected state change. We also have set up well-defined operational security practices internally, as well as in our runbooks and have coordinated with whitelisted caller governance to ensure we have a streamlined process to urgently pause the bridge if needed.

We've also implemented extensive testing across our codebase, including usage of the chopsticks tool for simulating the full Polkadot and BridgeHub stack so that we simulate any new changes and deployments to the bridge before executing them as well as a suite of

smoke tests that we run with every change (https://github.com/Snowfork/snowbridge/tree/main/smoketest).

We've developed a custom suite of infrastructure tools, including Ansible playbooks and scripts to provision and maintain off-chain infrastructure, such as message relayers and archive nodes for Polkadot and Ethereum.

We also developed a custom tool for more automated generation of pre-images for governance actions and upgrades to reduce the risk of any user/manual errors that could occur during these processes and further automate them: https://github.com/Snowfork/snowbridge/tree/main/control

Our runbooks are all here: https://docs.snowbridge.network/runbooks/updating-snowbridge-pallets-bridgehub-and-asset hub-runtimes

Launch on Kusama

The bridge is deployed and ready to go on Kusama too! We have decided to hold off on activating it and have prioritized Polkadot first, specifically because:

- There has been a lot of demand to go live on Polkadot asap and teams that are blocked by us.
- Kusama runs its consensus ~4 times faster than Polkadot, and has a lot more validators. This means that the BEEFY light client relayer costs are ~4-6 times more on Kusama than Polkadot. We're covering Polkadot costs ourselves for the initial launch, but can't afford to do so for Kusama until we secure further funding and set up third-party relayer incentives.

All operational work for the launch is already done, so we have 2 potential paths forward for the Kusama launch:

- Get treasury funding from the Kusama treasury to cover BEEFY light client incentives and activate the bridge
- Use the Polkadot <> Kusama bridge to send Snowbridge assets to Kusama and make Polkadot a dependency of the Ethereum bridge for Kusama.

We'll be doing a separate discussion and proposal with the Kusama community and treasury to finalize the choice of path here and corresponding portion of funding.

Milestone M3

General-Purpose Messaging

Our bridge has been built to support general-purpose messaging from the ground up. Our underlying channel, agent and messaging protocol is agnostic to the contents of messages being sent and all asset-transfer specific functionality has been built on top of this base layer.

Our architecture for general-purpose messaging supports both arbitrary XCM Transact messages as well as Ethereum smart contract calls, allowing both for:

- Ethereum contracts to send XCM messages to a pallet on a Polkadot parachain

- Polkadot pallets and end users to send XCM messages that are converted into Ethereum smart contract calls to any smart contract on Ethereum

We hope to see this power new cross-chain applications that have not been possible before, and we are in discussion with a few teams looking forward to building on top of this.

Our base protocol has already been audited, and the final piece of work to finalize the fee system and activate general-purpose messaging is here:

- https://github.com/Snowfork/polkadot-sdk/pull/114
- https://github.com/Snowfork/polkadot-sdk/pull/116
- https://github.com/Snowfork/snowbridge/pull/1141
- https://github.com/Snowfork/snowbridge/pull/1145

We plan to merge and activate general-purpose messaging soon after our launch of asset transfer, once final audits and security checks on these PRs are completed.

Launch on Polkadot

We're launched and live! Try it out here: https://app.snowbridge.network/

This was a significant effort with support across multiple teams and organizations. Our proposal went out and was executed here.

We've setup all configuration and deployed our contracts to Ethereum and things have been running smoothly since the launch, and our monitoring and operational infrastructure is all live.

Keep up to date with the status of the bridge on our status page here: https://app.snowbridge.network/status

Bridge Statemint *(ie, AssetHub)* assets to Ethereum

We've implemented support for bridging Polkadot-native assets to Ethereum. This has been built to work with native ReserveBackedTransfer XCM messages and fit cleanly into the existing Polkadot paradigm for how assets move across parachains. Transfers to Ethereum work similarly to transfers to any other parachain.

Additionally, beyond just the scoped task, we have added support both for:

- Assets that are native to AssetHub
- Assets that are native to any other parachain on Polkadot

The work for this support is here:

- https://github.com/Snowfork/snowbridge/pull/1155
- https://github.com/Snowfork/polkadot-sdk/pull/128

Similarly to general-purpose messaging, we plan to merge and activate Polkadot-native assets soon after our launch, once final audits and security checks on these PRs are completed.

NFT Support

We've built out support for bridging ERC-721 NFTs between Ethereum and Polkadot via Asset Hub. The XCM primitives for NFTs have developed recently and so Asset Hub now also has good support for them.

With this implementation, ERC-721s will also become XCM-native and any parachain that supports the latest corresponding XCM primitives will be able to use NFTs from Ethereum

The work for this support is here:

- https://github.com/Snowfork/snowbridge/pull/1185
- https://github.com/Snowfork/polkadot-sdk/pull/143

Similarly to general-purpose messaging, we plan to merge and activate NFT Support soon after our launch, once final audits and security checks on these PRs are completed.

Additionally, the above PRs are great templates for ourselves and external developers to use as a reference for how we can extend Snowbridge with new functionality as XCM evolves and new primitives go live. We can easily extend this for new functionality in the future, depending on demand from users, for example support for ERC-1155 or other new NFT-related standards.

Ongoing Maintenance and Support and Additional Work

The Polkadot ecosystem and XCM architecture has changed significantly over the last year and so there was major work done on Snowbridge outside the scope of the tasks scoped in the above milestones.

The majority of the work done over the last year has actually been unrelated to any of the above milestones directly.

We've put a ton of effort into refactoring and evolving our codebase in line with the changes in XCM, Bridge Hub/Asset Hub architecture and BEEFY, as well as taken on many major new items that were needed to ensure success of the project but that were outside the scope of our plans in our original proposal.

This section documents all these additional scope items.

XCM

As described above, we needed to re-implement support for XCM multiple times as we evolved from XCMv2 to v3 to v4 before it stabilized. This involved both work on the Polkadot side, as well as on our XCM-related Solidity code.

There were also changes in the XCM routing from UMP/DMP to HCMP to XCM more generally that we evolved together with.

BEEFY

Beefy changed a few times over the course of the last year. There were some modifications to the header and versioning format, and so we had to modify some of our light client contracts to support these changes.

Additionally, there were many issues and bugs with BEEFY that made it challenging for us to develop and test with. We encountered bugs with BEEFY during local development and on Rococo multiple times where our team was the first to uncover the issue and we were involved in debugging and fixing these issues.

In multiple instances, this required us to shut down and redeploy our Rococo testnet from scratch and redo all configuration and initialization of all our pallets, infrastructure and Ethereum contracts.

Fortunately through this process we also developed thorough deployment and testing infrastructure and an upgrade-based flow for BEEFY upgrades such that once live on Polkadot, we'll be able to make upgrades to new versions of BEEFY with minimal downtime as it evolves further.

Additionally, we worked closely with the Web3 foundation to do further analysis of the risks around potential exploitation of BEEFY, especially in consideration of risks where Ethereum validators behave maliciously. We now have a formal mathematical analysis of BEEFY's security, and have also implemented support for dynamic signature sampling to increase the efficiency and security of the light client.

BridgeHub

At the time of our original proposal, BridgeHub did not yet exist and we were still running Snowbridge on its own parachain. Once the decision to create BridgeHub was finalized, a major piece of work was for us to migrate all of Snowbridge off of its own parachain and onto BridgeHub so that we can maintain shared infrastructure with the bridges team at Parity.

BridgeHub is a very constrained parachain, and this complicated development of the bridge in many ways, compared to our previous parachain:

- OpenGov governance only
- No assets pallet installed, bridges must rely on AssetHub to store wrapped assets
- Relies on DOT for XCM fees, however most third-party parachains are not configured to accept BridgeHub as a trusted reserve for DOT.

AssetHub

At the time of our original proposal, AssetHub was still called Statemint and it worked a little differently. We've needed to refactor and evolve our implementation in multiple steps as AssetHub became finalized and launched as it is today.

Bridge assets are now stored on a newer foreign assets pallet on AssetHub and we adapted our bridge to work with this pattern.

Third-Party Support for Foreign Assets

With AssetHub using a new special pallet for foreign assets, we've needed to work through custom integrations with parachains and wallets to ensure that they're supported. We've made and supported pull requests for other parachain teams (eg: HydraDX) and wallets (eg: SubWallet issue, commit) to get foreign asset support which will unlock support for assets from both the Polkadot <> Kusama bridge and Snowbridge.

We've been pushing for foreign asset support across all parachains in the Polkadot ecosystem and expect to have most parachains upgraded with support soon.

Substrate, Cumulus and Polkadot Upgrades

Beyond these additional items, the ongoing maintenance work of keeping up with new Polkadot, Cumulus and Substrate upgrades has also required significant efforts from our team. We work on the bleeding edge, often using features that are not yet live and so some of that blood comes from our team too - we're often the first to use and test new features and support Parity in debugging and fixing issues that we discover with them.

For example, we discovered a critical issue in the FRAME message-queue pallet that prevented re-entrant processing messages.

We expect this to continue - we already have plans to use new XCMv5 features and expect Snowbridge to keep up to date with the latest changes to Polkadot, especially going forward.

Ethereum Dependency Upgrades

Since our last proposal, Ethereum itself went through some upgrades, notably the Shanghai-Capella and then the Cancun-Deneb hardfork. We implemented support for this (https://github.com/Snowfork/snowbridge/pull/786, https://github.com/paritytech/polkadot-sdk/pull/3029) and additionally, added support for live-upgrades of our Ethereum light client with no downtime.

We did a live test of this upgrade on Rococo and successfully upgraded our Ethereum Light Client in time with a hard fork of Ethereum via an automated upgrade with no downtime or issues.

This puts us in a confident position for Ethereum upgrades going forward on Polkadot. There will be further changes to Ethereum that we'll need to keep up with too, for example the upcoming Prague-Electra upgrade (expected in Q4 2024).

Similarly, the Ethereum client software we depend on *(Lodestar, Ethereum RPC endpoints)* have gone through API changes specifically related to their light client data and so we've needed to multiple pull requests and code changes to update our integrations with those APIs over time.

Solidity also had new version releases over the course of the project, so we updated all our contracts to use the latest Solidity version.

Polkadot SDK Merge

A major piece of work we completed was to merge Snowbridge into the core polkadot git repository.

Pre-merge

This involved multiple PRs where we needed to modify core Polkadot functionality to add features that were missing or make fixes in preparation for our merge - some examples:

- https://github.com/paritytech/polkadot-sdk/pull/2117
- https://github.com/paritytech/polkadot-sdk/pull/2146
- https://github.com/paritytech/polkadot-sdk/pull/2230
- https://github.com/paritytech/polkadot-sdk/pull/2338
- https://github.com/paritytech/polkadot-sdk/pull/3029
- And many others: https://github.com/paritytech/polkadot-sdk/pulls?page=1&q=snowbridge

Pallet merge

The main code merge was a ~2 month process of adapting our pallets to become part of the core Polkadot SDK repo. The main PR for this is here: https://github.com/paritytech/polkadot-sdk/pull/2522

This involved many additional incidental PRs as we got feedback from Parity for fixes we needed on our side, testing and adaptations needed for Snowbridge to become part of the polkadot-sdk continuous integration pipeline, and support to Parity for making fixes needed in the polkadot-sdk.

This also involved modifying the Rococo runtime so that Snowbridge is deployed as part of the core Rococo stack.

Lastly, we had to modify our own internal repositories to suit a development and deployment workflow where we upstream our changes through the polkadot-sdk. This created the need for a more streamlined local development environment for Snowbridge developers working across multiple different repositories, which we put significant effort into creating.

Polkadot SDK CI / Testing

With our code now part of the polkadot-sdk, we also had to create extensive additional testing and coverage for Snowbridge. Our tests now run as part of any upgrade to anything in the polkadot-sdk. This is important, as we need to ensure that changes any developer makes to the polkadot-sdk don't incidentally break Snowbridge. Our test suite includes integration tests that cover underlying bridge functionality and asset transfers that run as part of the polkadot-sdk continuous integration and deployment process.

Polkadot-Fellows merge and release scheduling

Similarly to the polkadot-sdk merge, before going live on Kusama and Polkadot we had a major piece of work to add Snowbridge to the polkadot-fellows runtimes repository so that it can be deployed as a core part of Polkadot and Kusama.

This was a multi-month effort involving collaboration with the Parity and the polkadot-fellows group to merge in. The main PR is here: https://github.com/polkadot-fellows/runtimes/pull/130

Our code is all bundled in crates on crates.io and part of the same process for bundling and deploying code as for any other polkadot-sdk changes and we added documentation and references to all relevant folders needed for this (https://github.com/Snowfork/snowbridge/pull/1101).

During this process, the runtime release schedule was going through operational changes and so we needed significant coordination with multiple teams to time our releases appropriately.

ORML Template

Many parachains today do not yet use the newer assets pallet that the polkadot-sdk and AssetHub provide, but rather use an older ORML library. For legacy support, we needed parachains to add an adapter to their ORML functionality to support the new AssetHub foreign assets.

We developed a template that ORML-based parachains can use for a much smoother integration: https://github.com/Snowfork/parachain-orml-template

Other parachains have a custom asset-manager pallet they use. We're working with those parachains to add support to them too.

Governance Suite

Beyond just supporting cross-chain governance, we felt it would be valuable to build a tool to automate many governance processes. This was done both for productivity gains, but also for security reasons - automated, audited processes around proposals, governance configuration changes and upgrades significantly reduce the likelihood of a broken change slipping through and causing issues for the bridge.

We have seen recent bridge hacks occur due to configuration and upgrade issues during operation of the bridge, as opposed to logic bugs and so we built a tool to automate the generation of pre-images for governance actions and upgrades. The code is here: https://github.com/Snowfork/snowbridge/tree/main/control

Frontend UI & Template

Our plans for the frontend scope was originally to build out a Javascript SDK that other developers could use to integrate support for Snowbridge into their frontend. We completed this, but decided that it would be valuable to also build out a UI of our own. This has been done both so that we can have a public-facing interface for end users at launch, as well as to create a template codebase that other developers can refer to or fork as part of developing their own custom UI, or integrating the SDK into their own applications.

The UI is live and can be used here: https://app.snowbridge.network/
The code for it: https://github.com/Snowfork/snowbridge-app

We are also already working with **Velocity Labs,** who is building their own UI using this template & our SDK.

Message Identification

Making use of the XCM SetTopic instruction, we've ensured that cross-chain events and messages can be linked together. For example, if a user on Ethereum sends a message to Polkadot, a message ID will be carried over the bridge, and inserted into the resulting XCM message.

This is also useful to constrain bad actors from using the bridge as an anonymizing service.

Protocol Versioning

We overhauled all our protocol-level data formats with support for versioning. This is important for ensuring that the bridge can be upgraded live without any downtime or potential data loss.

Gas Price Sync

We've implemented a gas price sync feature, allowing for the most recent Ethereum base gas fee to be synced to Polkadot so that we can price our fees more accurately.

PR here: https://github.com/Snowfork/snowbridge/pull/1188 - this will go live in the coming weeks

Ethereum Light Client Improvements

We made multiple improvements to our Ethereum Light Client to make it more robust and efficient. This includes adding:

- Support for ancestry proofs (https://github.com/Snowfork/snowbridge/pull/765)
- Caching of milagro keys (https://github.com/Snowfork/snowbridge/pull/811)
- Ringbuffer optimizations (https://github.com/Snowfork/snowbridge/pull/815)
- On-demand execution sync (https://github.com/Snowfork/snowbridge/pull/1154) for easier operations

among other improvements.

We also added a suite of fuzz tests for more robust testing security (https://github.com/Snowfork/polkadot-sdk/tree/snowbridge/bridges/snowbridge/pallets/ethereum-client/fuzz)

Documentation Overhaul

We completely overhauled our documentation into a new, much more approachable docs site: https://docs.snowbridge.network/

This covers comprehensive documentation both for researchers and developers looking to understand our technical architecture and implementation, for application and parachain developers looking to build on top of our bridge and for Snowbridge developers and relayers looking for streamlined operations guidance.

Comprehensive Operational Runbooks

We have written out a suite of runbooks that we (and other developers) can use for all common operational processes involved in maintaining and running the bridge long term: https://docs.snowbridge.network/runbooks

These are simplified, step-by-step guides that cover flows for bridge initialization, deployment and execution of upgrades, governance, relayer operations, pausing and emergency management.

These should help ensure that all operational knowledge is well documented, with simple, audited processes that both old and new new Snowbridge developers can run through to reduce the chance of human error during any of these processes.

Integrations

We've been doing a lot of integration support work with all the major parachains and wallets. This has involved digging into the code of each major parachain to ensure that we implement the easiest support path for Snowbridge assets on each, and setting up calls and comms for each.

As mentioned above, we have worked with **HydraDX**, **SubWallet** and **Talisman** to push for foreign asset support. We're working with **Moonbeam** to test our XCM on their custom Moonbase testnet and to get support for Snowbridge assets across all Moonbeam dapps. We have plans with **Stellaswap** to do an integration and potentially collaborate on their liquidating mining program once live on Moonbeam. We've worked with **Mythical** to help them develop a custom integration that will allow them to bootstrap their parachain's native asset from their Ethereum ERC20 token in preparation for airdropping the Polkadot-bridged version of this token to DOT holders. We've worked with **Bifrost** to test an end-to-end successful integration on Rococo with foreign asset support on their chain. We're discussing/testing an integration with **OriginTrail/Neuroweb** on their Rococo testnet. We're in discussions with **Centrifuge** who is looking forward to integrating with custom functionality on top of our arbitrary messaging support as soon as it goes live. We've started discussions with **Interlay** to add support for Snowbridge assets and to bridge their BTC to Ethereum, and **Kilt** to add support for bridging their native token to Ethereum. They're both eager to integrate as soon as Polkadot-Native asset support goes live. We're also working with **OpenZeppelin** to have a Snowbridge integration be part of their upcoming tooling they provide to Polkadot developers.

We're working closely with **Velocity Labs** to help them develop general tooling for the Polkadot ecosystem for cross-chain transfers. They're actively working on launching a new custom web app for cross-chain transfers with Snowbridge <> Polkadot transfers being the primary first flow they'll be supporting, and their XCM SDK will make it even easier for parachains to integrate Snowbridge capabilities.

We are also working with **ColorfulNotion** to build out extended dashboards and monitoring for the bridge. They've set up a Dune dashboard for it here (https://dune.com/substrate/snowbridge) which will be integrated into the main bridge website soon too.

A critical requirement to complete integrations with HydraDX, Moonbeam, Bifrost and OriginTrail/NeuroWeb is the release of polkadot-sdk v1.11.0, which includes an extrinsic to transfer tokens from AssetHub to a 3rd Party Parachain. Now that it has been released, 3rd party parachains can complete their testing and integration.

Present and Near Term Future

Snowbridge is live and ready to use. Our current priorities and focus in the near term future are:

- **wETH Sufficiency:** We are proposing to add wETH as a sufficient asset on Asset Hub so that we can enhance the UX of the bridge for sending ETH over
- **On-demand headers:** We are optimizing our bridge to allow for consensus headers to be submitted on demand, alongside messages so that we can reduce the need to sync headers when bridge activity is low and save on redundant header sync costs.
- **Deeper Integrations:** As described above, we are already working on integrations with multiple teams. We are continuing to do so, as well as deepen those integrations across their suite of products and dapps, especially as we deploy support for Polkadot Native Assets and Transact to enable more powerful integrations.
- **Deeper collab with Velocity Labs:** We are working closely with Velocity Labs to release a bridge and interoperability hub for token transfer across the entire Polkadot Ecosystem. Our collab will also grow developer advocacy, education and marketing to push Snowbridge as a platform for more developers to build on top of. We're also planning to work with them to push DeFi use cases for the bridge over the coming months, specifically related to increasing AMM liquidity and incentives and bringing more high quality ETH liquid staking tokens and liquid re-staking tokens to Polkadot.
- **Fee optimizations**: We have a working initial fee model for more advanced bridging functions, especially for Transact, but there are various architectural improvements we are making to allow for more flexibility for parachains and dapps in the assets and fee structures that will be supported on Snowbridge.
- **Enhanced routing:** The current XCM implementation is limited in how complex we can route messages across multiple parachains while still preserving the correct origin and required fees. We want to explore further support for paying fees in custom assets for wider third-party parachain asset support. We're also looking forward to features beyond XCMv4 and some new functionality related to recovering trapped assets shipping soon so that we can incorporate those into our bridge to improve resiliency against user's mistakenly sending broken XCM messages.
- **Liquidity programs:** We are in discussion with various teams, DeFi dapps/chains and market makers about running a program to bootstrap liquidity for the bridge and, once all our integrations are stable and well-tested, will potentially propose an incentive program to encourage bridging liquidity into the Polkadot ecosystem and across multiple parachains and dapps.
- **Paseo buildout:** The Polkadot community is moving away from Rococo to the new Paseo testnet. We plan to build out BridgeHub and our bridging stack on Paseo
- **Beefy acceleration toggle:** Our bridge allows for a trade off between latency and cost - spending more to accelerate a bridge transaction is possible. We could expose this functionality through our UI to power users who are willing to pay extra to accelerate the bridge.

- **Ongoing maintenance and updates:** Polkadot, Substrate, XCM and Ethereum itself are continually evolving, and it's important for us to keep up to date with all our dependencies and have strong ongoing maintenance of both the onchain systems and offchain infrastructure. Especially now being live, this is a major part of our current focus.

Longer Term Roadmap

Although we've reached a point of maturity with XCM and Bridge Hub for the launch, we still expect there to be much change over the coming year and so do still expect to have a much heavier maintenance and support workload than the typical decentralized project. We do expect to need dedicated team resources to keep up with this maintenance work, even longer term.

Otherwise, beyond our existing focus, there are a wide array of ideas and directions we have for moving Snowbridge forward, making it a better and cheaper product for the Polkadot community and evolving it to support a wider variety of use-cases to enable growth of Polkadot as a whole.

This longer term roadmap will likely be driven primarily by user feedback and direction learned from parachain integrations and community support. We expect it to slowly solidify over the coming months, as we get a lot of more of that feedback, but we have a shortlist of features and ideas that we think will be valuable to work on, specifically:

- **Atomic swaps:** Add an atomic swap layer that runs on top of our bridge, which would allow for near-instantaneous and far cheaper bridge transfers and swaps for smaller amounts and the large majority of users.
- **Further audits:** It's always a good idea to get an additional audit and to set up ongoing security testing of our security protocols and processes. We still need audits for our Polkadot Native Assets and Transact features, as well as some recent improvements we've made to optimize the bridge. Oak Security will support these. We also think it would be valuable to get a second full audit from SR Labs, including further internal training and expanding testing, fuzz testing, red teaming and the running of our bug bounty program.
- **Live ops & Red teaming:** Beyond audits, having strong live-ops, including red-teaming and simulated adversarial attacks in production to further strengthen our security should be valuable to work on. Further testing and improvement of our emergency response systems could be explored too.
- **On-chain gas and weight optimization:** We've implemented various low-hanging fruit optimizations to our contracts and pallet code, but of course there is a long tail of smaller gas optimizations that could be made to further reduce the costs of the bridge.
- **Beefy relayer ticket resumption:** If a transaction for a beefy ticket fails, our existing Beefy relayer keeps it simple and just tries with a new ticket, but it could be optimized to re-use existing tickets and save some costs

- **Frontend relayers:** Snowbridge is permissionless, meaning anyone can be a relayer. Relayer functionality can be ported to run directly on the frontend, allowing even non-technical users to relay messages themselves directly and save on costs.
- **Onchain Epoch Update Incentives:** Bridge epoch updates are currently funded offchain, with no incentives, rather than onchain. The funding and an incentive claim for epoch updates on both sides of the bridge could move on chain to allow more parties to run relayers and claim incentives.
- **BLS-based beefy:** Beefy currently requires verifying each validator signature individually. It could be enhances so that BLS-based signatures are created, allowing for just a single verification pass on Ethereum to save on costs
- **Zero-knowledge bridge exploration:** There has been significant research work on using on-chain ZK-proofs for proving consensus, so that the full client logic does not need to run on-chain. Explore porting our light client logic into a ZKVM so we can take advantage of this.
- **Fallback governance:** The current governance process requires that the core Gateway and BEEFY client function correctly. If there is some unexpected bug that completely breaks either of these contracts, recovery may be challenging. Improving Polkadot to support a fallback governance system which falls back to a recent, past validator set in the event of Beefy breaking with the current validator set could mitigate this theoretical scenario (https://hackmd.io/MlVt_yt9TQC8PvMZY8WCcQ)
- **Frontend widgets & embeds for external dapps:** Enhance our frontend interface so that other dapps can directly embed bridging widgets into their UIs without needing to do a deeper integration with our Javascript SDK
- **Extended NFT Support:** Enhance our NFT apps with functionality for a wider set of NFT standards, for example ERC-1155, royalty standards and token-bound accounts

Funding

Snowbridge is now the longest running and most complex public goods project in the Polkadot ecosystem outside of Parity itself. We're a public-goods focused team competing with projects that have each raised $100m+ in funding at $1b+ valuations. We build a bridge that's not just better for DOT holders due to 100% ownership and aligned team incentives, but that's also a functionally better design based on a trust model without third party signatories that handicap trust and undermine Polkadot's security.

Snowbridge has never raised any funding from outside investors. We've been funded entirely through bootstrapping, early grants and our previous treasury proposal made in November 22. We've successfully stayed a public goods-based project for many years, building long-term, complex tech which is traditionally done through massive raises at high valuations that provide major upside to the team after the project launches.

Our last proposal included incentives designed to sustain a high-performance engineering team to produce great, long term value for the Polkadot ecosystem with clear upside potential that could allow a public goods project to compete with VC funded projects.

It succeeded, paying out 28 months of runway funding covering the period from September 2021 to December 2023; agreeing on milestone targets; and agreeing on aligned longer term incentive funding to ensure future success.

This proposal is being put forward to execute funding for the retroactive milestone payments and kick off the future incentives.

Incentive Funding and Alignment

A key part of the future funding is to ensure that incentives are aligned between Snowbridge and the Polkadot ecosystem. To ensure this, the funding was designed with the following considerations:

- A major portion of payouts are in the form of long-term, vesting exposure to DOT ensuring that the team is incentivized to grow value in the Polkadot ecosystem long term
- The majority of the payout triggers and DOT exposure are weighted to long term success criteria for continued successful operation and integration of the bridge across the Polkadot ecosystem, not just for shipping engineering milestones.
- All the long term payout triggers are conditional on the successful operation of the bridge. If for any reason there is a break, delay, security issue or loss of funds from the bridge, the payout triggers can be paused and should then only resume if the bridge resumes in operation successfully without losses.

Additionally, we think incorporating a community insurance and bug bounty mechanism into these incentives will further cement alignment.

Insurance

The majority of payout triggers in this proposal and our future incentives do not immediately payout to the Snowbridge team - they are scheduled over a 24 month period. We propose that these payouts are also earmarked for a community insurance and bug bounty program that can be used to insure the bridge and its users against some risk of loss in using the bridge.

Snowbridge only gets its upside exposure if the bridge is successful and secure long term, otherwise funds will be used to cover community losses up to the total cap of whatever is remaining in future Snowbridge payouts at the time of loss.

- The insurance will apply in the event of any widespread theft or loss of assets due to a hack that affects a majority of users of the bridge. It will not cover losses due to an individual user, parachain or dApp sending a broken message through the bridge or sending an asset to an inaccessible destination through their own negligence.
- The insurance can only cover assets that have clear price discovery in highly liquid markets - specifically: DOT, ETH, USDC, USDT, DAI, WBTC. We will add more assets to this list based on community discussion. *(It's not possible for us to insure assets that are still illiquid, as they cannot be valued in a straightforward way).*
- The insurance pool will come from our remaining payout incentives that have not yet been triggered at the time of claim. Our long term incentives total to ~$7.5M, so the insurance pool will start with this $7.5M cap and the cap will decline over 24 months until all our payout triggers have occured based on our existing planned payout

structure above. A diagram visualizing this is shown below, at the end of the funding section.
- In the event of any valid insurance claim, the payout schedule to the Snowbridge team should be adjusted and reduced to cover the claim.
- Payouts will be in DOT, based on a market valuation of lost assets at the time of loss.

The intention behind this design is to incentive-align our team with the interest of bridge users, and provide a base layer of protection for the bridge as usage ramps up over time. It is not intended to provide full long term insurance for the entire TVL on the bridge.

Bug Bounty

Similar to the insurance policy above, up to 250,000 USDC of our remaining payout incentives will be made available for a bug bounty. This amount mirrors a similar bug bounty that is active for the Polkadot <> Kusama bridge. The bug bounty will remain open for 24 months, or until the initial 250,000 USDC allocation is used up. The bug bounty program is described in more detail here: https://docs.google.com/document/d/19CDyYdYSFIkHoDJxNMZ6lok8aRkHU8BFx9SMkcf0dlE/edit

Payout Structure

The original funding proposal specified:

- Runway funding up until December 2023
- A milestone and incentive based payout structure for launch + post-launch work.
- Additional expected costs for auditing and for relayer incentives.

For both existing auditing costs and runway costs since January 2024, Snowbridge has reduced payouts and stretched our last proposal's funding to last until end June, as we've been expecting to cover this and future costs via our milestone bonuses and long term operation funding.

Similarly for relayer costs, Snowbridge has been and will continue to cover this ourselves for the next 3 months.

Our original milestones have been completed. This new funding proposal sets up a payment schedule to cover the first 12 out of 24 months of vesting payments for the completed milestones, as well as for 9 out of 21 months of the long term incentive funding triggers. This effectively covers our expected payments from July 2024 to July 2025.

It also specified and covers retroactive audit costs and light client relayer sponsorship since January 2024 and future expected audit costs and light client relayer sponsorship through mid 2025.

For our future runway costs through 2024 and 2025, which includes both maintenance work and our upcoming roadmap, we also expect to cover those via the existing proposed incentive funding.

The treasury should expect an additional future proposal to cover the remainder of our expected milestone and incentive payments beyond July 2025.

The payment structure, repeated from the last proposal *(with audit + relayer costs now specified)*, is as follows:

Payout Trigger	**Amount (based on the Subscan EMA7 DOT exchange rate of 6.22, with a 5% slippage buffer)**
Audit Costs	120156 DOT *(based on ~$230k retroactive costs and ~$480k projected future 2024 costs for future updates, additional audits and red teaming)*
Light Client Relayer sponsorship	67693 DOT (based on projected $400k costs through mid 2025)
P1 - Milestone M1 completed *This proposal sets up payouts for the first 12 months, and the remaining 12 months will be setup in a future proposal*	63462 DOT vesting monthly, linearly over 2 years (based on $375,000 value)
P2 - Milestone M2 completed *This proposal sets up payouts for the first 12 months, and the remaining 12 months will be setup in a future proposal*	95194 DOT vesting monthly, linearly over 2 years (based on $562,500 value)
P3 - Milestone M3 completed *This proposal sets up payouts for the first 12 months, and the remaining 12 months will be setup in a future proposal*	158656 DOT vesting monthly, linearly over 2 years (based on $937,500 value)
P4 - Bridge is live with successful operation and integrations for 3 months, triggering 3 months after launch	158656 DOT (based on $937,500 value)

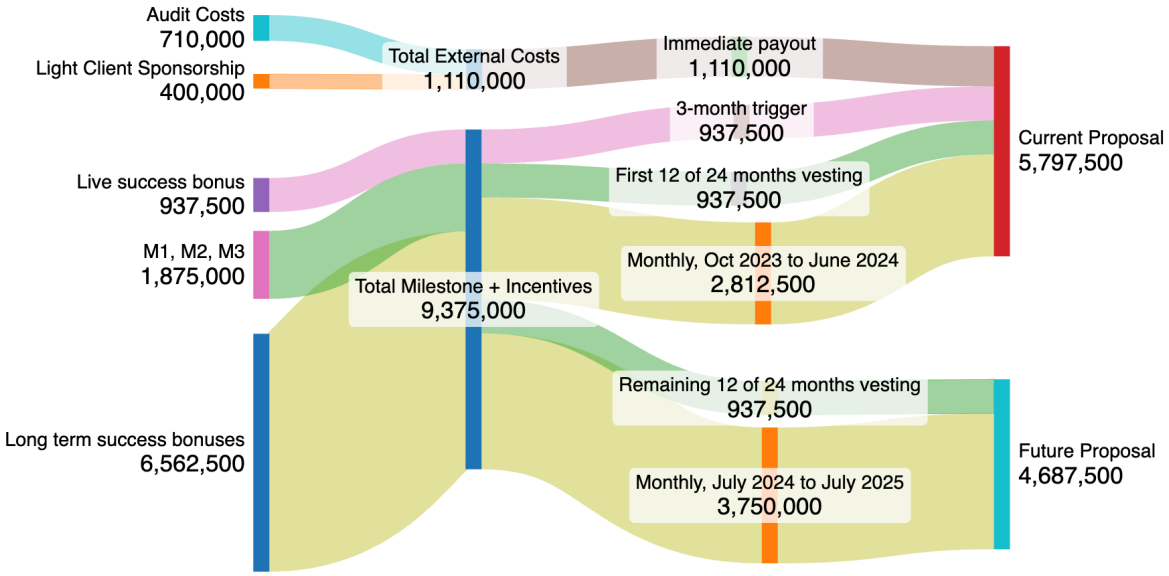
P5 - Continued successful operation and integrations on Polkadot 21 further months after P4 trigger, paid out every 30 days starting 3 months after launch *This proposal sets up payouts for the first 9 months, and the remaining 21 months will be setup in a future proposal*	312,500 USDC per 30 days (based on $312,500)
Total milestone + incentive cost over 24 months	$9,375,000
Total payout in this proposal	**505157 DOT and 2812500 USDC** *(~$5,797,500 USD, which covers the first ~$4,687,500 USD of incentive costs and ~$1,110,000 USD audit + relayer costs)*

Notes:

- *The M1, M2 and M3 payments were originally planned to use a vested_transfer, but they have been changed to use a preset monthly treasury spend schedule with the same linear vesting as a vested_transfer, because the current Big Spender treasury track is not allowed to call the vested_transfer function.*
- *The last proposal also included an additional planned $625k worth of KSM, paid out from the Kusama treasury for continued success of the Kusama <> Ethereum bridge, 3 months after launch. This will be submitted to the Kusama treasury in a later proposal as part of a Kusama launch process.*

Payout structure summary

See below diagram for a summary of the full Polkadot payout structure:



The insurance and bug bounty pool is made up of the combined Live success bonus and Long term success bonus, which totals to $7.5 million at the start.

Technical execution details

The technical execution is a single transaction, with a batch payout that covers all 4 payment buckets mentioned:

- Single treasury.spend of 187849 DOT with no delay. Paid on execution of proposal.
- 12 treasury.spend of 13221 DOT each. Each delayed to be claimable monthly over ~12 months, starting immediately, totalling 158652 DOT
- Single treasury.spend of 158656 DOT. Delayed to be claimable 3 months after launch.
- 9 treasury.spend 312500 USDC each. Each delayed to be claimable monthly over ~12 months, starting 3 months after launch, totalling 2812500 USDC.

Spend Amount	validFrom Block	Days Delayed
187849 DOT	-	0
13221 DOT	-	0
13221 DOT	21,724,000	17
13221 DOT	22,156,000	47
13221 DOT	22,588,000	77

13221 DOT	23,020,000	107
13221 DOT	23,452,000	137
13221 DOT	23,884,000	167
13221 DOT	24,316,000	197
13221 DOT	226.4220833	227
13221 DOT	256.4220833	257
13221 DOT	286.4220833	287
13221 DOT	316.4220833	317
158656 DOT	22,588,000	77
312500 USDC	22,588,000	77
312500 USDC	23,020,000	107
312500 USDC	23,452,000	137
312500 USDC	23,884,000	167
312500 USDC	24,316,000	197
312500 USDC	226.4220833	227
312500 USDC	256.4220833	257
312500 USDC	286.4220833	287
312500 USDC	316.4220833	317

The on-chain execution code for this is as follows:

- Preimage hash: 0xdfa7dc80601d80e5c8f08bbc99b072133243bef4ca07269a19bbbaff15f11d4b
- Preimage bytes:
0x1a025c130504000100a10f01000f00042a097aac06040001010032b451ddfb7b71e4463c38a5bac29b
d9679f4c04d853bae12bb1ee4440c1e1ac00130504000100a10f01000f000037c9f7a205040001010032b
451ddfb7b71e4463c38a5bac29bd9679f4c04d853bae12bb1ee4440c1e1ac0160aa5801130504000100a
10f01000b00f4158a3e78040001010032b451ddfb7b71e4463c38a5bac29bd9679f4c04d853bae12bb1ee
4440c1e1ac00130504000100a10f01000b00f4158a3e78040001010032b451ddfb7b71e4463c38a5bac2
9bd9679f4c04d853bae12bb1ee4440c1e1ac01607b4b01130504000100a10f01000b00f4158a3e780400
01010032b451ddfb7b71e4463c38a5bac29bd9679f4c04d853bae12bb1ee4440c1e1ac01e01252011305
04000100a10f01000b00f4158a3e78040001010032b451ddfb7b71e4463c38a5bac29bd9679f4c04d853b
ae12bb1ee4440c1e1ac0160aa5801130504000100a10f01000b00f4158a3e78040001010032b451ddfb7
b71e4463c38a5bac29bd9679f4c04d853bae12bb1ee4440c1e1ac01e0415f01130504000100a10f01000b
00f4158a3e78040001010032b451ddfb7b71e4463c38a5bac29bd9679f4c04d853bae12bb1ee4440c1e1
ac0160d965011305040001000a10f01000b00f4158a3e78040001010032b451ddfb7b71e4463c38a5bac2
9bd9679f4c04d853bae12bb1ee4440c1e1ac01e0706c01130504000100a10f01000b00f4158a3e7804000
1010032b451ddfb7b71e4463c38a5bac29bd9679f4c04d853bae12bb1ee4440c1e1ac016008730113050
4000100a10f01000b00f4158a3e78040001010032b451ddfb7b71e4463c38a5bac29bd9679f4c04d853ba
e12bb1ee4440c1e1ac01e09f7901130504000100a10f01000b00f4158a3e78040001010032b451ddfb7b7
1e4463c38a5bac29bd9679f4c04d853bae12bb1ee4440c1e1ac01603780011305040001000a10f01000b0
0f4158a3e78040001010032b451ddfb7b71e4463c38a5bac29bd9679f4c04d853bae12bb1ee4440c1e1a
c01e0ce8601130504000100a10f01000b00f4158a3e78040001010032b451ddfb7b71e4463c38a5bac29b
d9679f4c04d853bae12bb1ee4440c1e1ac0160668d01130504000100a10f0002043205e51407009573c2
48040001010032b451ddfb7b71e4463c38a5bac29bd9679f4c04d853bae12bb1ee4440c1e1ac0160aa58
01130504000100a10f0002043205e51407009573c248040001010032b451ddfb7b71e4463c38a5bac29b
d9679f4c04d853bae12bb1ee4440c1e1ac01e0415f01130504000100a10f0002043205e51407009573c24
8040001010032b451ddfb7b71e4463c38a5bac29bd9679f4c04d853bae12bb1ee4440c1e1ac0160d9650
1130504000100a10f0002043205e51407009573c248040001010032b451ddfb7b71e4463c38a5bac29bd

9679f4c04d853bae12bb1ee4440c1e1ac01e0706c01130504000100a10f0002043205e51407009573c24
8040001010032b451ddfb7b71e4463c38a5bac29bd9679f4c04d853bae12bb1ee4440c1e1ac016008730
1130504000100a10f0002043205e51407009573c248040001010032b451ddfb7b71e4463c38a5bac29bd
9679f4c04d853bae12bb1ee4440c1e1ac01e09f7901130504000100a10f0002043205e51407009573c248
040001010032b451ddfb7b71e4463c38a5bac29bd9679f4c04d853bae12bb1ee4440c1e1ac0160378001
130504000100a10f0002043205e51407009573c248040001010032b451ddfb7b71e4463c38a5bac29bd9
679f4c04d853bae12bb1ee4440c1e1ac01e0ce8601130504000100a10f0002043205e51407009573c248
040001010032b451ddfb7b71e4463c38a5bac29bd9679f4c04d853bae12bb1ee4440c1e1ac0160668d01