EOS Network Foundation

Part I
2022

EOS API+ Blue Paper

Providing Access for the Next Generation of EOSIO-powered dApps.

Table of Contents

I. Introduction

EOSIO is a third generation blockchain technology platform that has been in development since April of 2017. Despite significant investment towards development of the core codebase, one critical need that has never fully been met is that of external applications seeking to use EOSIO APIs. With limited access to pertinent EOSIO data, developers often need to expend significant effort to meet the data requirements of their applications.

I. The Working Group

The EOS Network Foundation (ENF) started the API+ Working Group in September 2021 as the first step in improving the situation. The teams of EOS Nation, EOS Rio and Greymass were brought together for this task based on their prior knowledge developing, operating and using EOSIO APIs.

The teams were given the direction to create a “Blue Paper" about EOSIO APIs. This document was described as a blend of components that are traditionally found in a white paper, a yellow paper, a request for proposal, and a roadmap. The rest was left up to the team to determine how best to accomplish this with an anticipated publication date in early 2022.

II. The Blue Paper

Based on this mandate, initial research was performed to help understand the services used to offer APIs in EOSIO and the needs of application developers. This research then expanded into other distributed ledger technologies and their approaches in solution design. This Blue Paper is the culmination of the research that was conducted.

To start the paper with, a more in-depth Background presents the history of the group and its work. This is followed by a set of Fundamentals to help define the thinking behind the proposals that are being presented. The current API Landscape is then offered to give a sense of the names of technologies in use along with the option to dive deeper into each solution. The paper then ends with a list of Proposals for consideration and a Conclusion with potential next steps.

III. Post-release

With the release of this Blue Paper, discussion on what comes next will be the topic of debate for the EOS Network Foundation and greater EOSIO ecosystem. The research and proposals presented within need to be viewed with a critical eye and discussed openly. Decisions will have to be made on each proposal with regards to priority, funding, and the parties responsible for it.

We would like to welcome those who wish to share their views to do so.

II. Background

Three independent teams worked together to build this Blue Paper from the ground up. This process provided a unique learning opportunity as there was no set direction provided from which to start. This section will outline how this process was brought together as well as the public engagement process with the different target audiences considered.

I. Group Formation

In September 2021, the EOS Network Foundation (ENF) asked the teams of EOS Nation, EOS Rio and Greymass in a History+ working group. The ENF selected these teams based on their knowledge of EOSIO API operations and experience with history-related solutions. Collectively this group has many years of experience building software, operating infrastructure and deploying history solutions within EOSIO and the greater blockchain ecosystem.

This opportunity empowered this newly formed team to self-organize work towards the creation of the Blue Paper. The initial step for the team was to become familiar with one another and determine how exactly to tackle the work that needed to be done. This was done by sharing information about our team compositions, the API operations we manage, and each of the history solutions we develop and/or operate.

Originally, the mandate from the ENF was to take a deep look at the existing history solutions and recommend a viable path forward for sustainability in these difficult to manage operations. Early on in this process, the teams identified that to solve challenges for history related matters would also require the scope to include a wider range of topics related to APIs. The decision was made to rename the working group to API+ and expand the scope of the research to be performed.

With a paper delivery date of only a few months away, team members started identifying the areas in EOSIO that needed the most attention. Starting with history solutions, this brainstorming led to a discussion about the general philosophy that needed to be considered for every API related proposal. These ideas resulted in the creation of the Fundamentals section of the Blue Paper to highlight what ideals were important.

With the research well underway, the API+ working group was announced in November 2021. This kicked off the public engagement phase allowing interaction between the working group and the broader EOSIO community.

II. Public Engagement

To gather as much information as possible, a wide range of approaches were taken.

The most direct approach was to solicit input from the stakeholders already involved in the EOSIO ecosystem. Dozens of different organizations were approached and responded to our inquiries, providing information about their greatest wants and needs. These conversations led to some of the richest feedback the team would receive.

The team also actively listened to conversations in public discussion channels across all of EOSIO to get a sense of how developers building on EOSIO were utilizing APIs. The team participated in discussions across Telegram, WeChat, Discord, and Twitter. This gave us a sense of their struggles and presented opportunities to improve their development experience.

The information collected came from a wide variety of professionals who depend on EOSIO API services. This includes the developers and operators who are responsible for EOSIO-based applications, exchanges, and other service providers.

III. Writing Process

All of the resulting research data and feedback was collected throughout many smaller documents grouped by topic. Each major topic that was identified was given to members of the working group to further research. The additional findings were brought back and reviewed by the working group and potential solutions were discussed for inclusion in this paper.

The solutions that were deemed viable by the group were then written collaboratively by members of the working group. Cycles of feedback and revision between the group and the writers resulted in the creation of each proposal. All of the proposals were then merged into the Blue Paper draft and further refined to present multiple ideas in one cohesive document.

Now with a single document containing each individual proposal and its justification, a process of deduplication was needed. Many of the proposals repeated the same subjects, tried to explain the same concepts, and shared common requirements to be successful. All of these ideas needed to be expressed outside of the individual proposals, resulting in the creation of the non-proposal content found throughout this document.

III. Fundamentals

Throughout the group's discussions with stakeholders and the subsequent research, many questions were raised about how we should fundamentally address the topics in the Blue Paper. This section outlines both the fundamental Assumptions that were made as well as the Principles we sought to abide by during the creation of the proposals.

I. Assumptions

There is a wide range of needs that EOSIO APIs must be able to fulfill. While it's near-impossible to create a single solution that is capable of meeting all needs, steps can be taken to ensure that most common needs can be met. In order to determine how to cast the widest net possible, a needs assessment was conducted. We make the following assumptions within this Blue Paper based on these efforts.

1. Demographics

The target demographic for the work outlined in this Blue Paper is any party who integrates EOSIO-based blockchains into applications and services. This includes, but is not limited to:

- Application Developers
- Data Analysts
- Exchanges/Custodial Services
- Executives/Leadership Roles
- Operators and Service Providers
- Protocol/Tooling Developers
- Smart Contract Developers

These types of individuals and organizations make up the key stakeholders who will be most impacted by any changes to EOSIO API infrastructure.

2. Common Needs

Throughout discussions with various stakeholders many of the needs identified were common to nearly all parties. It is the working group's assumption that due to the similarities of each participant's feedback, these unmet needs are regularly found throughout the ecosystem.

While the existing API solutions today provide for many common needs, often the APIs are missing important information, provide irrelevant information, or simply don't exist in a manner that is widely accessible. This causes challenges for both developers and operators, as developers are forced to work around these inefficiencies which causes unnecessary load for operators providing the API services.

3. Uncertain Sustainability

Many APIs are operated as public goods and provide data free of charge for anyone who wishes to access them. Without a business model specifically designed to aid in the sustainability of this ecosystem, the operations of key components of the EOSIO API infrastructure are surrounded with uncertainties.

Quite often the operators of these services provide access as part of a loss leader strategy that involves other more profitable products and services. The reliability of these public goods often ends up being dependent on the operator's other service offerings. This situation creates an environmental risk in which the success or failure of a single project can cause issues with accessibility to the blockchain and its data.

To ensure wide availability of these services in the short term, the working group assumes that many operators of API solutions will require subsidization until a point in time when either a solution for self-sustainment is adopted or a sufficiently decentralized alternative exists.

4. Collective Knowledge

With EOSIO-based development representing only a fraction of the research and development happening within the greater blockchain industry, the working group assumes there will be many opportunities to improve adoption and gain knowledge from other ecosystems.

Collectively the EOSIO ecosystem needs to adopt and contribute back to blockchain industry wide initiatives. This can be done through the adoption of standards and usage of technologies that increase the interoperability of the different ecosystems. These efforts will - over time - result in a reduction of friction for businesses, their developers, and the end users of the applications integrated with EOSIO-based blockchains.

II. Principles

There are many ways to design, build and operate APIs. Based on the same assessment of needs within the EOSIO ecosystem and the research performed, the team attempted to establish a list of core principles that will be required as solutions are devised.

1. Developer-First

The first principle is to ensure a strong focus on the needs of developers integrating EOSIO into their applications and services. The overarching goal of meeting these needs is to retain existing developers while also being able to attract new developers into the ecosystem. Any reduction in complexity that allows developers to more easily build on these networks will bring new and interesting uses of this emerging technology.

2. Forward-Looking

While evaluating the needs of developers both inside and out of EOSIO, we are presented with an opportunity to contemplate what future needs there may be. The information collected during this research wherever possible will be used to ensure the capabilities of these systems can be used moving forward in future solutions involving EOSIO APIs.

3. Favor Decentralization

The level of decentralization of each component within EOSIO can potentially fall on many points across a wide spectrum. Certain components will gain or lose the value they can provide depending on how they are approached. During the design of solutions, the benefits and sacrifices of different approaches must be weighed against one another, while ensuring the possibility of decentralization and showing favoritism towards it.

4. Pragmatic

There are a finite amount of both human and financial resources available in the EOSIO ecosystem today. Understanding what is possible is required when presenting a proposal that contains a solution. The solutions presented must consider the reality of what is practical to execute in the short term and what will require long term planning.

5. Deliberate

The adoption of new API solutions will take time. It takes a significant amount of effort for all the applications and services that use APIs to adopt new approaches. The effort required to adopt these changes is also often significant. For these reasons, the development of future API solutions needs to be very deliberate in planning how they are built and rolled out.

6. Accessible

The design of these systems must also include approaches that ensure the services are as widely available as possible. The consumers of EOSIO APIs require that data be accessible in a variety of programming languages, operable from different environments, and based on documented standards using open-source code. No singular point of failure should exist that impacts the ability for consumers to access the data, and anyone should have the ability to provide access to it for themselves.

7. Interoperable

Proposed solutions should find ways to integrate and work with other existing technologies, both inside and outside of EOSIO. The changes made to EOSIO whenever possible should work globally across EOSIO-based blockchains. Opportunities to integrate EOSIO technologies with other external systems should be considered. The EOSIO community should also influence the development direction of these external systems.

IV. API Landscape

A major part of creating the Blue Paper involved researching existing available APIs for EOSIO and solutions available in the broader blockchain space. Understanding where we are today is important to understand where we need to go next when considering the Proposals presented.

To understand the context of the paper requires some explanation about APIs. According to the Wikipedia entry:

> "**an application programming interface (API)** is a connection between computers or between computer programs."

Restated, an API is what allows the applications running on your devices to send and receive information while connected to the internet. In the context of this paper we are generally considering APIs to be services that enable interactions with EOSIO data.

Application Programming Interfaces (APIs) are fundamental to all blockchains. Users need to be able to submit information to be included in the blockchain. They also need to be able to query the blockchain to receive information. Both of these activities require an API between the blockchain and the user application. APIs are part of the core infrastructure that must operate as expected in order to allow even the most basic operations.

There are several different types of APIs used within the EOSIO ecosystem:

> **Native EOSIO APIs** are those that are part of the core code distribution for EOSIO, which can be enabled by any operator simply by setting a configuration option. These traditionally have been developed by Block.one and contributors to the core codebase.
>
> **Community-Developed EOSIO APIs** are those which require additional applications to be configured and run, which leverage the native APIs to make data available in different ways. These traditionally have been developed by community developers and as part of independent projects.
>
> **Non-EOSIO Specific APIs** are those which have been developed to try to standardize access to blockchains while remaining independent of any specific blockchain. EOSIO does not currently support many of these industry standards.

This section highlights the Native and Community-Developed EOSIO APIs. For brevity, non-EOSIO Specific APIs are discussed only where they are relevant to specific proposals.

I. Native EOSIO APIs

EOSIO currently has 4 primary native APIs available for operators to use - **chain**, **history**, **state history**, and **trace**.

1. Chain

The chain API is how clients can read data related to the chain's current state. This includes accounts, tokens, and other contract data. The API endpoints in the chain API were introduced at launch in June of 2018 and have seen minor improvements over time, but no major structural changes. These APIs also allow clients to submit transactions for inclusion in the blockchain.

2. History

The history API was how clients can get information about past transactions that have already been performed on the blockchain. The initial specification for these APIs was introduced at launch in June of 2018, but the native plugins for the core software have since been deprecated and no longer function properly without modification. The way this API was designed never took scaling into consideration and quickly exceeded its own limitations. These APIs were used by many clients to test whether a transaction was completed and the subsequent actions it caused. It was also used to let clients retrieve a past list of transactions for a specific account.

3. State History

The state history API was developed to serve as the basis for a history API replacement. It offers a few new features such as streaming and specialized access to the history of the blockchain state. The state history API was introduced in version 1.8.0 in June of 2019. These APIs allow other applications to consume historical information and use it in the creation of new data sets and APIs.

4. Trace

The trace API was developed in response to pressure from the developer community about a lack of scalability within the previous history solutions released. The trace API offers far less API endpoints and data availability than either history or state history, but does offer a more

flexible way to store and retrieve historical information from the blockchain. The trace API was introduced in version 2.0.4 in March of 2020. The one endpoint the trace API provides gives other applications the ability to consume historical information and use it for various purposes.

II. Community-Developed EOSIO APIs

Since the release of EOSIO many external API services have been developed by the community in response to needs that have been identified by the developer and operator communities. Each of these APIs generates data from one or more of the native APIs above and operates as a separate process further enriching and/or correlating information to create valuable information not readily available from native APIs.

1. API Types

Derived State

A derived state API is one which takes the current state of the blockchains data and expands upon it in a way which the native APIs do not in order to fulfill a specific need. These types of APIs can be used to meet either the needs of a specific contract or of a specific application.

Streaming

The streaming of blockchain data is a feature in high demand by applications and services which require immediate access to relevant data. These APIs are designed to act as both middleware for services to index their own data and for clients that need specific live data.

Transaction History

Despite EOSIO having multiple history related APIs - scalability challenges and a lack of maintenance to applications, APIs, and SDKs have led to the creation of a number of external transaction history solutions.

2. Solutions

Native APIs have always been baked into nodeos and considered official. As new APIs were demanded and Block.one made it clear that it would not be developing those solutions, emerging community efforts self-organized usually inside teams operating as BPs. Those solutions aim at the same goal but have no standardization, making it difficult for developers to migrate among them. The development and operation of those solutions are mostly self-funded and offered for free and depend on other revenue streams from developers and operators, making it difficult for those teams to prioritize the efforts and plan for longer terms. As of February 2022, the following community-developed specialized APIs are most widely adopted:

	Derived State	Streaming	Transaction History
Hyperion	✓	✓	✓
Chronicle		✓	
Light API	✓		
Firehose		✓	
dfuse	✓	✓	✓
Roborovski			✓

Hyperion

Built by EOS Rio and based on the state history APIs in response to the difficulties for the original state history API to deal with the increasing chain size. Hyperion is a full history solution serving the original v1 standard and implementing new endpoints (v2). Hyperion is an active project with multiple updates and new features being constantly pushed. More recently it started allowing for the creation of plug-ins for specific indexing of contracts and is being used to serve AtomicAssets APIs on WAX and EOS, and EVM API responses on Telos.

Chronicle

Chronicle is built by EOS Amsterdam and based on the state history APIs. This streaming solution interprets data from the native APIs and presents it as a stream of JSON objects for use in other applications.

Light API

The Light API is built by EOS Amsterdam and is based on Chronicle, which in turn is based on state history. It's purpose is to provide information about EOSIO blockchain accounts and token balances. It is used by many web applications including Bloks, EOS Authority, and Unicove.

Firehose

The Firehose was built by EOS Canada (now StreamingFast) and uses a modified version of the native EOSIO software. This modified EOSIO has an additional API for realtime streaming of events called "DMLOG". Firehose provides a gRPC API and offers a stream of events for use in indexers and applications. The block stream can be played backwards or forward, understands forks, and resumes from the same point when disconnected. The original EOSIO-based Firehose has been adapted to other blockchains for use in The Graph.

dfuse

dfuse adds richer filtering capabilities to the stream provided by Firehose. Additionally, dfuse does many things including storing the token balance for each account at the last irreversible block, indexing transactions on a variety of attributes, allowing the query of transaction and state, filtering on the needed information. dfuse is only receiving minimal enhancements.

Roborovski

Roborovski is developed by Greymass and based on the trace history APIs. It takes raw data from the trace API and stores it in an optimized format for future use. Indexers are capable of building upon this new format and are used to serve transaction information by ID and account. Roborovski needs an enhanced version of the EOSIO trace API and is not yet open source.

V. Proposals

During the course of the research performed for the API+ working group, dozens of experienced blockchain professionals were interviewed to get a sense of their greatest needs and wants. Some of these individuals identified needs that were caused by a lack of knowledge, in which a simple improvement of educational materials could have aided their work. Oftentimes the needs expressed went beyond a lack of documentation and demanded significant effort to meet.

Developers' efforts can be reduced significantly through strategic improvements to the systems that the EOSIO APIs depend upon. The interconnected nature of these various systems creates a complex web where any change to one system is potentially impactful on another. The required research across these systems will require a holistic approach to effectively draw conclusions.

For these reasons, the first proposal being presented is one to establish a group to coordinate the research and development of these efforts moving forward. The decisions made on how to coordinate the API architecture could potentially affect every other proposal.

Proposal 1: API Research and Standards

Beyond the unmet needs that demand more significant attention, a number of needs that were identified have known opportunities to improve upon. The most common unmet needs served as the basis for all the additional proposals in the Blue Paper.

- Development of known solutions:
  - Proposal 2: Transaction Lifecycle
- Opportunities to support network operations:
  - Proposal 3: Specialized APIs
  - Proposal 4: Central API Service
  - Proposal 5: Distributed API
  - Proposal 6: Blockchain Data Depot
- Potential integrations with existing and emerging industry standards:
  - Proposal 7: Rosetta
  - Proposal 8: The Graph

Each proposal may offer multiple solutions. When calculating total personnel costs the maximum of the pay range is used. When there are several options, each option is standalone and the total cost presented is for that option itself. All costs are listed in US Dollars (USD).

I. Proposal 1: API Research and Standards

Developers and operators working with EOSIO are given a set of basic APIs on which to build applications. Many application needs cannot be met with these basic APIs and they need the more complex tools. Often they may need more than one of these complex solutions. The level of difficulty to build and maintain applications increases with the proliferation of these solutions. This increase in difficulty creates an artificial barrier that often prevents both EOSIO network and application improvements without significant investment of time and effort.

This proposal seeks to reduce this burden through making improvements to the basic APIs, as well as standardizing and improving the components that allow the creation of more complex solutions.

1. Proposal

Establish an API Research and Standards Team (ARS Team) to aid in the building of the next generation of open source software for EOSIO APIs. This is an ambitious effort that will require multiple phases of research and the coordination of development, support, and maintenance across many teams in the EOSIO ecosystem.

This team will be responsible for coordinating this software ecosystem with regards to:

1. What data is made available from the core software (nodeos).
2. How this data is made available for use by external applications.
3. The data formats standardized external APIs should expose.

Additional responsibilities as they are identified can be added to ensure consideration is given for the entire EOSIO ecosystem.

Potential Projects

To give a sense of what types of projects this team would be responsible for helping to coordinate, below are some common needs that were identified from the working groups research.

- **Basic APIs:** Reimagine the basic JSON-RPC APIs to optimize call patterns and improve access to critical data. Implement REST patterns and improve error handling.
- **Native Streaming:** Adding optional native streaming for trace/state changes as it relates to accounts and contracts for clients to subscribe to.
- **Transaction Lifecycle:** Improving how APIs handle the submission of a transaction and its lifecycle until it's been included on-chain.
- **Transaction History:** Standardize how this information is best retrieved from the core software and what the final output for clients needs to consist of.
- **API Extensibility:** Create a framework which applications can use to build contract or application-specific endpoint specifications (e.g. EVM, AtomicHub, etc).
- **API Optimization:** Allowing horizontal scalability and read-only data access, storage requirement lessening, feature and service discoverability, and ease of deployment.

The potential list of projects represents **years** worth of work and will continue to expand in scope as additional needs are identified.

2. Rationale

This work needs to be done out in the open and coordinated between many teams. Historically the development of various EOSIO API components was done in an uncoordinated manner resulting in unstandardized responses, varying requirements, and a lack of a clear path for those seeking to adopt them. These solutions were also often developed in a vacuum and released without consideration towards all of its potential useful applications. The resulting solutions offered niche use cases and a lack of support for software development kits (SDKs).

3. Responsibilities

Research & Evaluation

The team will perform research in order to compile a comprehensive list of ecosystem needs and wants. This can be done by working closely with SDK and application developers. The

evaluation of open source applications and observations from API usage patterns will also highlight areas which may deserve attention.

System Architecture

The information gathered through the research process will be used to determine the best path in prioritizing system architecture. The architecture will determine where each part resides in relation to one another, their dependencies, and how best efforts can be applied towards the creation of new components and comprehensive API solutions.

The EOSIO architecture must be outlined in an easy to understand format for developers who wish to engage. This architecture will consist of things like relationship diagrams, API specifications, and design goals. The format of these documents needs to be easily iterated on, since as stakeholders identify unmet needs, the team must be able to react and adopt components to devise and propose new solutions.

The design of the architecture should also take into account the unique characteristics of each component and their ability to scale. High demand components need to be identified and determine if they can operate independently and scale horizontally.

Specification Proposals

Each defined API component is accompanied by a specification document that describes in detail the expected data both for inputs and outputs. These specifications will offer standards that each individual instance of a component must adopt to ensure interoperability between various components that depend on one another. These proposals will be published and reviewed by the developer community.

Coordinating Development

As needs are identified, additional APIs and software may need to be developed. This team will coordinate development efforts and ensure that the resulting work towards these solutions fits into the overall system architecture. The development itself may be done by any party, whether it be voluntarily, through a bounty, a grant or as part of a larger initiative.

4. Funding Methods

Funding the ARS Team

- **Permanent Team:** The ARS Team could be given an initial budget with expectations that it'll operate autonomously. A multi-sig account would be set up and initial funding would be provided to secure a runway for the team for a specific amount of time. The team should organize itself to anticipate long term operations, which would depend on external entities to evaluate their performance and provide further support.
- **Working Group:** The ARS Team could be given a predetermined amount of funding and finite timeline, similar to the working groups. The team would perform the research required, document the optimal future system architecture, and publish specification proposals. The coordination of development would then be handled by different organizations(s).
- **Existing Organization:** The ARS Team could be a department that operates within a new/existing organization, for example a foundation or company that agrees to take it on. The EOSIO networks could offer grants to help subsidize the costs of the team in that organization.

Development of Software

- **Preallocated Funding:** The ARS Team could be given access to an initial pool of funding to use as it sees fit to fund development work through contract development, bounties, and grants. Additional funding could be allocated to this pool as deemed necessary.
- **Direct Recommendations:** The ENF could take the recommendations of the ARS Team and fund development directly. This could be done at set time intervals (quarterly/yearly) or in an ad-hoc manner, depending on the volume of proposals.
- **Public Recommandations:** The ARS Team could publicly publish its recommendations for development, which would allow stakeholders from various EOSIO blockchains to step up and offer either funding or developer resources.

5. Estimated Costs

The estimated costs outlined here are for the ARS Team alone, any funding of development will go above and beyond and costs will be based on the specific project.

Due to the flexibility of how this team could be assembled based on the options above, the actual costs required per role are broken down into individual elements in this section. The number of actual team members selected will determine the overall price.

Each anticipated role is estimated for a 1-year cost, which can be broken down into smaller time increments based on the information provided.

> **System Architect:** $150,000 to $250,000 per year
>
> Responsible for the creation and adaptation of overall architecture and individual components. Creates technical specifications, understands the interconnected nature of all components, and works to create solutions for external entities. Coordinates with project managers and technical writers.
>
> **Project Manager:** $90,000 to $110,000 per year
>
> Responsible for overseeing the overall project, communicating with various external entities and key stakeholders, and facilitating the collection of research materials. Coordinates with external entities, architects, and technical writers.
>
> **Technical Writer:** $60,000 to $80,000 per year
>
> Responsible for assisting in the creation of technical specifications, standards, system diagrams, and other forms of written communication. Coordinates with both project management and architects.

A minimal team would most likely consist of a single person from each role, though scaling up the number of architects and technical writers would accelerate the rate of production. We should also assume these are full time roles for the people involved that are able to work closely together to maintain this system.

	Minimal Team	Mid-Sized Team	Full Team
System Architectects	1	2	3
Project Managers	1	1	2
Technical Writers	1	2	2
Total Estimated Costs	**$440,000**	**$770,000**	**$1,130,000**

6. Recommendations

The following recommendations are made by the working group:

- The knowledge gained from the SDK development proposal in the Wallet+ paper will provide knowledge that may assist with the architecture of future API systems. For this reason, the team believes that this API Architecture proposal should either happen in parallel with the SDK development or afterwards the SDK project has completed.
- This project requires leadership from people with very specific knowledge of these systems or the ability to learn the landscape very quickly. The quality of the entire resulting API infrastructure will be dependent on the ability of a small core group to ensure a consistent vision. For this reason it's recommended to ensure the right people are selected to participate.
- If the EOSIO+ working group defines an overall structure for coordinating among EOSIO chains, this project should fall under its purview at some level.
- The recommendation is to fund a **Mid-Sized Team** for at least one year. However, it is unlikely that the full budget would be spent during the first year depending on the amount of time to recruit the team members.

II. Proposal 2: Transaction Lifecycle

One of the most crucial times in the EOSIO user experience is whenever a user attempts to submit a transaction. The user must know if they can perform the desired transaction and then when the transaction has been successfully completed. Techniques to accomplish this exist today in many applications and some wallets, but these same technologies should be a core part of all the experience of all EOSIO users.

As such an ubiquitous part of the EOSIO experience, these enhancements should be applied directly to the core API software and made available via a software upgrade (no forking required) for all operators.

1. Proposal

The proposal seeks to hire expert EOSIO developers to create the following four projects to address common user experience issues in the core software.

Transaction Resource Cost Estimation

With the subjective billing system we have at the core of EOSIO today, it is impossible to predict the exact cost of a transaction in either CPU or NET. Estimations before submitting transactions however offer good insight to understanding whether or not a transaction will be successful and roughly what it will consume. A new API endpoint will be created that allows the submission of a transaction that will return with an estimate of its resource usage.

Subjective Billing Improvements

In EOSIO networks today, transactions occasionally get lost in the peer-to-peer network due to “subjective billing” errors and varying rules between peers. The code involved in the propagation of transactions will be looked at and new methods to more intelligently route transactions will be evaluated. Code will then be developed to ensure less transactions are lost without explanation due to the configuration of individual nodes on the network.

Transaction Retry

In order to help improve the user experience of all EOSIO-based applications, a system to intelligently monitor incoming transactions will be integrated into the native API software. This system will track the transactions it receives through its own APIs and ensure they are relayed appropriately on the network.

Transaction Finality Status

Checking on the finality of a transaction is one of the most highly used features for APIs that specialize in history. Clients after submitting a transaction will often in the background query to determine whether or not the transaction was successful. To increase availability of these services and to reduce the complexity of operating one, we would like to see an optional feature integrated into the core system which tracks the most basic information of all recent transactions and their current status.

2. External Considerations

Required Network Improvements

Inconsistencies should be expected on peer-to-peer networks like EOSIO, which many of these projects are specifically designed to help mitigate. There are inconsistencies we were unable to identify and propose solutions for that will need further investigation. For example, resource billing costs are occasionally 10-1000x more expensive than they should be - causing either a transaction to silently fail on the network or users being overbilled for their transactions.

Issues like these affect all these systems' reliability and their accuracy in estimating costs. We would recommend these systems are also investigated in future proposals.

Project Codebase

A few principles are to be followed for managing the the development of these features:

- These new features would be contributed to a separate branch on the Mandel github repo. PRs need to be proposed to be merged with the ‘main’ branch.
- Both unit test and integration test cases need to be added to cover the new functionality. Modifications to existing test cases may be required.
- Code will be developed in such a way as to allow the community to provide feedback as features are at a point where it is relevant to do so. Stated another way, there is to be no “big drop of code” at the end of the project.
- As of this writing, there is no plan for a community-maintained developer portal so documentation will be provided in the form of a markdown document within git. This document will explain how to operate and use all of the new features in the style of “release notes” in previous nodeos releases.

Project Guidance

Either the ARS Team or another well-versed EOSIO developer needs to be available to help guide the development of these features.

3. Cost Estimates

These estimates are for well-implemented features that include proper test cases, readme documentation, developed by an **experienced team** of core EOSIO developers.

	Cost
Development	$380,000
Community Architect/Program Management	$20,000

4. Recommendations

The following recommendations are made by the working group:

- The ENF should contract directly with a company that has expertise to implement these features.
- These features should be included as part of Mandel 3.1 to show momentum on nodeos development.
- Due to the development timing and because the ARS Team has not yet been formed the API+ Working Group should oversee this development.

III. Proposal 3: Specialized APIs

As explained in the background section, a few months after the launch of EOS it became clear that the native history API plugin would not scale to match transaction volume of the block-chain. This was the first example of a function deemed necessary by the developer community that would require significant effort to solve externally of the core software.

Today the teams developing and operating specialized API infrastructure do so at a loss, and are often subsidized through Block Producer Rewards. This funding volatility has shown to stagnate development efforts and create instability in API offerings.

1. Proposal

Several methods are outlined in this proposal to support these specialized APIs. Each option provides a certain level of financial commitment to help ensure availability and stability over the coming year(s).

Software Maintenance

Support the on-going development of the external API software and adapt to the upcoming changes of the core EOSIO software. Maintain security updates, work with SDK developers, update documentation, and offer developer/operator support.

Support Existing Operations

Ensure that specialized API software and services are being maintained and operations continue to meet public API needs in the short term (for at least 1 year).

These APIs will be required by applications and services until a point in time when future architecture is determined and put into production. Depending on the recommendations of the ARS Team, some APIs will continue to be required for a period of time after new solutions are put in production. This is especially true in situations where backwards compatibility cannot be maintained.

Support should be re-evaluated on a yearly basis to determine the health of this portion of the overall EOSIO ecosystem.

Increase Availability

Expand operations to add capacity and/or redundancy, either by supporting existing teams or by offering opportunities to new teams. Utilizing existing teams will centralize responsibility and minimize costs, while bringing in new teams will decentralize responsibility and have higher costs.

Feature Development

Developers of Specialized APIs add new features to their solutions based on their expert knowledge and feedback from operators and users. In the future, new features should follow the standards that will be defined by the ARS Team. The working group recommends **not** funding any new feature development pending this standardization effort.

2. Rationale

User Experience Risk

The reputation of an EOSIO network as a whole partially depends on these specialized APIs being available. In cases where they are unavailable, many applications start receiving innumerable user complaints which are out of their control. Given the goal of attracting more developers to build on EOSIO, it is critical to keep existing applications working smoothly and thus the APIs upon which they depend.

Current Landscape

As of February 2022 the situation within many EOSIO blockchains is less than ideal. To provide an example, specialized services available for public usage on the EOS Network are as follows:

- 2 Hyperion instances (operated by EOS Rio and Sw/eden)
- 1 Roborovski instance (operated by Greymass)
- 1 Firehose/dfuse instance (operated by EOS Nation)
- 1 Light API instance (operated by EOS Amsterdam)

EOS, as a network, currently lacks redundancy, capacity, and compatibility for its specialized APIs. Any one of these services going offline or halting in development is a potential risk for any EOS application that relies on them. Similar situations exist across much of EOSIO when it comes to specialized API solutions. (Note: There exist more general use public API endpoints on EOS than what is listed here.)

3. External Considerations

EOSIO Updates

We assume an increase in significant development on the EOSIO platform is likely in the near-term. This can be seen in projects like Mandel and EVM+ Working Group. Should these sorts of developments generate large amounts of change, the impact of these changes on existing solutions need to be considered. Consequently additional funding above and beyond this basic maintenance may need to be considered for these solutions as these changes are planned.

API Architecture Timing

While improvements to all EOSIO systems are being proposed, we must also consider the amount of time that the planning, development, and deployment of any new system will take. It's likely to take a considerable amount of time before any future solution proposed is production ready and well adopted by a network. With many EOSIO networks already live and operational, existing infrastructure will need to be maintained while future upgrades are planned.

Lack of Qualified Experts

All of these projects have a limited amount of available experts that can be leveraged for development purposes. Should these experts be unavailable in the future, it would cause significant project risk. Should the community wish to add developer redundancy, the expectation should be that the project will need to add more than $150,000 per developer per project per year. If a large amount of development is expected on EOSIO in the near future, this funding should be allocated sooner rather than later as it takes many months for a new developer to be hired and able to usefully contribute to a project.

Network Growth

It is hard to predict the exact operational costs as it highly depends on network usage. More usage leads to more network bandwidth and more server CPU, RAM and storage space. The estimated costs provided could pay for purchase, rental or cloud cost depending on the operators' preference. The assumption used in the proposals is for a network growth rate roughly equivalent to the average growth rate and usage of the network thus far. Additional funding should be considered if there is significant growth.

Supply Chains

As of February 2022, there are on-going supply chain issues impacting computer hardware availability throughout the world. Funding operators well in advance of when they are needed is a prudent move, since at least 6-9 months of runway will be required for any solution requiring dedicated and/or specialized physical hardware.

4. Requirements of Funding Recipients

Transparency

In order to provide transparency, each recipient needs to post a blog article as to project progress at least once per quarter. This article should outline the details of maintenance activities performed. Reporting back to the community is a necessary requirement before any future funding (if any) would be made available to any projects.

Open Source

Any development work being funded for the external APIs should have requirements of it being open source software. For example, the option to support development for Roborovski should be contingent on its release as an open source product.

5. Cost Estimates

Development

In an ever-changing blockchain ecosystem, specialized APIs need ongoing maintenance and development. Development costs include Software Developers, Technical Writers and Project Managers, among other support tasks. The detailed breakdown of estimated costs have been omitted for brevity. These estimates are for 1 year of development.

Software Maintenance: The costs associated with basic maintenance and development of each of these types of APIs. The costs are associated with the anticipated time required by the softwares developer to keep pace with upcoming EOSIO changes, resolve outstanding issues, and other minor improvements.

	Cost
dfuse	$75,000
Firehose	$75,000
Light API	$40,000
Hyperion	$150,000
Roborovski	$75,000

These estimates assume that Mandel 3.1 does not make significant changes to existing nodeos interfaces.

Operations

Operations covers both the infrastructure (servers, network) and operators to keep them running smoothly. The detailed breakdown of estimated personal costs have been omitted for brevity, but hardware infrastructure has been separated for better clarity. These estimates are for 1 year of operations.

Support Existing Operations: The costs of supporting the personnel and existing infra-structure used in the solutions live across EOS today. The numbers are based on the minimal amount that would help ensure operations remain online, not on the total cost of operating those solutions.

Increase Capacity of Existing Operations: The cost associated with increasing the capacity of an existing API operation. These numbers are based on the cost of additional servers and personnel required to operate the solution.

Improve Redundancy of Existing Operations: The cost associated with adding redundancy to existing API operations. These numbers are the combined value of the new hardware and human resources required to geographically distribute the operations to increase resiliency.

Existing Operations	Support		New Capacity *		New Redundancy *	
	Personnel	Infra	Personnel	Infra	Personnel	Infra
dfuse by EOS Nation	$50,000	$10,000	$20,000	$25,000	$20,000	$75,000
Firehose by EOS Nation	$10,000	$10,000	$5,000	$20,000	$5,000	$20,000
Light API by EOS Amsterdam	$10,000	$5,000	$5,000	$10,000	$5,000	$10,000
Hyperion by EOS Rio	$50,000	$10,000	$20,000	$30,000	$20,000	$30,000
Hyperion by Sw/eden	$50,000	$10,000	$20,000	$30,000	$20,000	$30,000
Roborovski by Greymass	$50,000	$10,000	$5,000	$25,000	$5,000	$35,000

* Costs for **New Capacity** and **New Redundancy** are in addition to the **Support** costs.

Add Additional Teams: The costs of recruiting new teams capable of bringing these types of APIs online. These teams should be well versed in existing EOSIO API operations. The costs are increased from using an existing team due to the learning curve and unfamiliar systems used in each solution. An existing team would be tasked to train and guide deployment of the solution.

Additional Teams	Training	Personnel	Infrastructure
dfuse	$10,000	$120,000	$75,000
Firehose	$0	$20,000	$20,000
Light API	$0	$40,000	$10,000
Hyperion	$5,000	$70,000	$30,000
Roborovski	$5,000	$75,000	$35,000

6. Recommendations

The following recommendations are made by the working group:

- Ensure the longevity and success of specialized EOSIO infrastructure providers and software developers, fund both **Software Maintenance** and **Support Existing Operations**. This funding should be reviewed annually.
- Fund **Expanding the Capacity** of existing operations to ensure they are ready for network growth.
- Ask **Additional Teams**, who are experienced with EOSIO to take on hosting Firehose, Roborovski and Light API to increase resiliency for application developers. The result will be a total of 2 teams operating each solution in the Moderate option, and 3 teams in the Strategic option (except for dfuse).

These recommendations are summarized in the **Moderate** investment level:

	Minimal	Moderate	Strategic
Software Maintenance	✓	✓	✓
Support Existing Operations	✓	✓	✓
Expand Capacity of Existing Ops		✓	✓
Expand Redundancy of Existing Ops			✓
Add Additional Teams *			
↳ Firehose/Roborovski/Light API		+1	+2
↳ Hyperion/dfuse			+1
Total Cost	**$690,000**	**$1,110,000**	**$1,900,000**

* in addition to the current teams

IV. Proposal 4: Central API Service

Providing both public and professional API services in EOS today is not a sustainable business. The cost of providing these services exceeds the amount that consumers are willing to pay for access. This is in part due to the difficulties of operating and scaling these services, as well as the ineffective ways applications must access data through the APIs.

Long-term, this situation can be improved with the creation of more efficient EOSIO API solutions and reducing the complexity of operations. In the short-term steps should be taken in order to improve the availability of services for developers who need access today despite these operational challenges.

Non-EOSIO solutions shed light on potential sustainability models for service providers and operators, but are only now facing issues that are inherent to EOSIO chains such as the cost of processing, storing and serving large scale blockchain with hundreds of millions of blocks. And currently two major approaches can be seen in the broad blockchain industry. Centralized

companies offering Blockchain as a Service, such as Infura and Getblocks, and decentralized efforts to create an incentivized ecosystem for providers such as The Graph.

1. Proposal

Fund a dedicated team to provide API services and act as a public resource to onboard new developers and businesses into the ecosystem. Do this through a new funding stream to reduce the dependence of public services on block rewards.

2. Rationale

There are a handful of block producers who provide these types of public services through the use of funding made available in block rewards. The block rewards these organizations receive are primarily intended to facilitate their block production operations, with a variable and limited amount reinvested into additional operations like public API access. This creates situations where most public services are made available on a best-effort basis with no additional support.

3. Proposed Services

Public Goods

It is in this context that we propose that these services will be subsidized, on an ongoing basis, for public goods providing essential services. At a high level this includes:

- **Website/Brand:** The creation of a website to welcome and onboard interested parties.
- **Public Peering:** Reliable connectivity for other operators.
- **Public API Access:** Rate limited access to API services for basic usage.
- **Public Downloads:** The data operators need to get started running their own service.
- **Documentation:** Supporting documentation for operators and developers.
- **Support:** Assistance for operators and developers in regards to services.

Paid Services

Additional paid services could be deployed for guaranteed access to resources, to meet client specific needs, and for more dedicated support to businesses. This could be done either using a for-profit model for the team involved or with this team serving as part of a larger organization.

The free services could serve as a gateway into the professional services.

4. Funding Considerations

The potential methods to create and fund this proposal are many, especially when considering the rate at which the EOSIO ecosystem has changed throughout 2021 and into 2022.

Management Structure

To create an accurate estimate on how to build the team requires answering some fundamental questions in how this proposal is executed.

- **Leadership:** Who leads this initiative?
- **Ownership:** Who owns the brand, website, servers, and other assets?
- **Operations:** Which team is responsible for managing the services?
- **Onboarding:** Which team is responsible for managing the website and documentation?
- **Support:** Which team is responsible to help support developers?

Service Offerings

The operations also have a significant amount of variables that need to be defined, as determined both by the team involved and the requirements of the network's users. The more developer-friendly the service, the higher the cost to provide the service. Here are a few examples of things that should be considered:

- **Regions:** End-user experience is best if the infrastructure is close to them. Which regions will it service? Access in China may require extra attention.
- **Languages:** Which spoken languages does it support?

- **Support:** What is the time zone coverage for customer support?
- **Features:** Which services and functionality does it offer beyond basic APIs?
- **Deployment:** How will these features be deployed, what kinds of servers (owned servers, rented servers, cloud), etc?
- **Capacity:** How many resources are required today, and how many next month?
- **Development Environment:** Should access be provided for both development and production environments?
- **Testnets:** Developers rely on testnets for testing. Which testnets should be supported, if any?
- **Beta Services:** As new APIs are developed should these be provided for developers to test?
- **Open Source:** Should the software developed to assist in operations be made available and supported?

Risk Mitigation

As various options are considered on how to approach the solution, it is important to recognize that the fastest rate of deployment will come from more centralized approaches. Hiring a single team that deploys services using turn-key solutions can happen quickly, but creates reliance on both the single team and the turn-key solutions they select. Conversely, using multiple teams that deploy services using leased servers/data centers will delay the rate at which services can be delivered but help remove any single point of failure.

The answers to all of these questions have drastic effects on the budget required.

5. Possible Delivery Methods

There are multiple ways the solution could be delivered. The delivery method could even change over time if planned for. A few possible scenarios are outlined below and other variations may also provide benefits not outlined in these examples.

	Option 1	Option 2	Option 3
Leadership	Contracted Team(s)	Central Entity	Central Entity
Ownership	Contracted Team(s)	Central Entity	Central Entity
Onboarding	Contracted Team(s)	Contracted Team(s)	Central Entity
Operations	Contracted Team(s)	Contracted Team(s)	Central Entity
Support	Contracted Team(s)	Contracted Team(s)	Central Entity

Option 1: Delegate all responsibility to an contracted team

If this new API operation was delegated to an existing team, the **management structure** questions could all be answered with a single existing team. It could be any of the teams that were part of the API+ working group or another entity in the space capable of completing the task. The **service offerings** would then be defined by the team and transparently reported to the stakeholders who make the funding available at regular intervals.

Accountability would come in the form of future support towards the external team. If the team didn't adequately meet the needs of the community, future funding could serve as leverage to affect change in direction. A potential downside to this option is that the external team would have ownership over the assets (brand, servers, etc) with no course of action should the team halt operations.

Option 2: Leadership by the network, operations managed by external teams

Another option for how the **management structure** questions could be answered using multiple teams contracted by a central entity to perform these tasks on behalf of a network. This would allow one entity to retain ownership of the assets and provide leadership, while outsourcing the work required to external teams.

The **service offerings** could be defined from the leadership and the teams that are contracted to perform the work could make recommendations. This option creates a hierarchy of accountability and ensures the needs of the specific network are met, but depends on the accountability of the central entity serving in the leadership role.

Option 3: Leadership and operations by the network

As an opposite to Option 1, this answers all the **management structure** questions with the one central entity. The central entity would need to have staff available to perform all these functions and would have complete control over both the assets and operations.

6. Estimated Costs

Despite the complexity of determining the best approach, a number of costs can already be estimated using potential approaches based on current conditions. The actual pricing will vary depending on the decisions made in each aspect of the project. On the other hand, some costs are highly dependent on the options selected.

Branding

A strong brand and identity for the service provider can determine how approachable and successful it is. This includes the identity of the provider as well as its design aesthetics and usefulness of its internet presence. The estimated cost for this effort ranges on the low end from $50,000 for basic design work and could go upwards of $200,000 for a well designed identity and associated trademarks.

Organization

A legal entity will be required to purchase hardware and sign contracts with service providers. This cost can be avoided by using an existing organization to provide this service. Otherwise startup costs would be in the $30,000 to $60,000 range.

Ongoing costs for accounting and legal services in the $30,000 to $40,000 per year range need to be considered as well.

Any organization requires leadership roles such as a CEO, CFO and CIO. Depending on the level of complexity of the services provided, these could be part time or full time roles.

Billing Infrastructure

Should paid services be provided a billing infrastructure would be needed. To build this would require a development team consisting of a project manager, front end developer, backend

developer, smart contract developer and writer. The cost of building such infrastructure would be $250,000 - $350,000.

Ongoing yearly maintenance needs to be planned for in the $25,000 - $50,000 per year range as the array of supported services evolves.

Deployment Model

The services being offered and how they are deployed will vary depending on the selected approaches.

Cloud Compute: All hardware hosted by a cloud provider and use a single platform end-to-end. This is the fastest to deploy and most simple to manage. It provides a very flexible environment to help meet the changing needs of the EOSIO ecosystem. All of this comes at a significant cost that will increase alongside usage.

Dedicated Rentals: All hardware hosted by dedicated server rental companies and managed at the server level. While not as fast to deploy and manage as cloud resources, these dedicated servers will require less logistics than owned hardware. The cost of dedicated rentals will be much less than cloud resources, but in the long term still more costly than owned hardware.

Owned Hardware: All hardware purchased and data center space leased to create a dedicated and owned resource. The slowest to deploy, the least flexible, and will require multi-year contracts. The cost of owned hardware will be significant at first and then much less expensive than either cloud or dedicated rentals in the long run.

Hybrid Model: Between these options a hybrid approach can also be adopted to optimize the costs of administration, up-front expenses, and recurring costs.

No matter which option is chosen, additional considerations like deployment automation and monitoring tools need to be created and/or deployed.

Operations / Support

Running services requires staff to manage infrastructure, generate invoices and provide customer support. The costs to provide an excellent level of service can escalate quickly. Discussion is required to understand the level of service that is practical to provide.

Availability: Application developers would ideally like a guarantee of 24/7 service availability. This kind commitment requires significant staffing (either available or on-call) as compared to a Monday-Friday 9-5 support model. The Wikipedia article on High Availability has good background information on this topic.

Languages: The number of spoken languages supported also has a significant impact on costs. The website, documentation, knowledge base, and other assets need to be made available in those languages. Customer service staff also need to be available to answer technical questions in those languages.

7. Recommendations

Based on the research performed by the API+ working group, a number of recommendations can be made on how best to create this service for a network.

Determine the Approach

This proposal with it's large number of possibilities has been difficult to present in a simple manner. In order to progress on this proposal, fundamental decisions will need to be made on how a central solution is approached and how it remains accountable to the network that funds it. **For this reason, no concrete cost estimates are presented in this proposal.** The working group recommends that discussions continue on the desired approach to further refine this proposal and ultimately come up with a funding plan.

Plan an Upfront Investment

Ideally, funding for these services optimally needs to be a multi-year commitment. Typical costs for operations, including data center and hardware leases, are amortized over a 3-year period. Server rentals from hosting providers also benefit from longer term rental periods, typically using 1-year periods. Depending on the approach by the leadership of this project, funding for somewhere between 1 to 3 years will provide optimal use of funding.

Account for Growth

The growth of demand for these services, both free and paid, cannot be immediately accounted for and will need ongoing evaluation by the leadership. Additional resources to provide ample services worldwide will increase any estimated budgets as the network continues to grow. These budgets should assume a reasonable rate of growth and start with more capacity than is required today.

V. Proposal 5: Distributed API

1. Proposal

This proposal is to provide funding to create a White Paper. This White Paper would outline how to build and operate a Decentralized Autonomous Organization (DAO) that incentivizes the operation of APIs of many types on EOS and other EOSIO-based chains. Existing teams would operate their infrastructure, even on a best effort basis, while the DAO would incentivise multiple teams to offer high availability services.

Like in Central API Service, this proposal aims to create highly available API infrastructure. However, the approach is very different. Both proposals could be funded in parallel.

2. Rationale

As shown in Specialized APIs there are already five teams operating Specialized APIs for EOS. Many more teams offer native APIs on EOS and furthermore there are more teams offering APIs and history solutions for other EOSIO-based chains. Those teams have proficient EOSIO operations engineers and are also the developers of some of those solutions. Additionally there is geographic and cultural diversity across the teams.

The existing EOS incentive structure only rewards block producers for producing blocks. Since this incentive structure does not value infrastructure, it is hard to attract any of those teams to provide infrastructure. There needs to be a different way to coordinate efforts among all the teams, and at the same time reward them for the service they already provide. The new structure can leverage the investment already done and operations in place to provide high availability, highly scalable APIs.

3. Estimated Costs

$250,000 to be allocated for up to 5 experienced EOSIO teams to collaboratively write a White Paper. The project would seek other sources of funding after this initial funding such as private token sale for VCs and a potential token offer.

4. Recommendations

The following recommendations are made by the working group:

This proposal does not contain enough information as to how the DAO would work. If the community expresses interest in this proposal, a group of people could gather and create a more detailed proposal. Based on this detailed proposal, funding could be allocated to build the White Paper similar to how the initial working groups were funded.

A strong signal from the community that is a worthwhile project is needed before spending time on a more detailed proposal.

VI. Proposal 6: Blockchain Data Depot

1. Proposal

This is an on-going funding request to support one or more infrastructure providers to make historical data available for download for anyone who wants to use it. This proposal is to formalize this support and have more frequent updates be made available by more providers and/or more locations.

2. Rationale

Historical data available for download is important for application developers who want to get started hosting their own infrastructure. Until now data downloads have been provided by several different block producers in EOS, each with varying capabilities and update frequency. Existing providers are:

- EOS Amsterdam
- EOS Nation
- Greymass
- Sw/eden

A directory of availability of different kinds of data downloads should be provided. bp.json is the current de-facto way to describe API availability, so the bp.json specification should be extended to support these data types. Similarly, the EOS Nation Validator is the de-facto reference for teams to find the location of APIs and other related resources. It should be updated to show the available data for download based on the bp.json specification update.

In the future, the **ARS Team** should also specify data download formats and/or APIs containing metadata related to available data. This could include torrent or IPFS type downloads. Teams providing data download services would need to adjust accordingly.

For any provider who is provided funding download metrics need to be published on a quarterly basis to understand how much usage there is.

3. Estimated Costs

There is a small cost to update the bp.json specification and the validator tools. The cost is for a developer and project manager for a short period of time.

> **Project manager:** $5,000 (one time cost)
>
> Working with the community, update the bp.json specification to reflect the need to specify locations for the various data download types.
>
> **Software developer:** $10,000 (one time cost)
>
> Working with the project manager, update validator reports according to the new specification and test. Respond to user feedback related to these changes.

Support API developers with data downloads from a single team and a single location. These are annual recurring costs for teams that already host EOS infrastructure.

> **Snapshots:** Personnel: $10,000 per year, Infrastructure: $5,000 per year
>
> New snapshots should be made available every 3 hours. Historical snapshots should also be available, but frequency can be reduced to weekly snapshots to save on storage space. Download bandwidth should be at least 1gb/s.
>
> **Blocks:** Personnel: $10,000 per year, Infrastructure: $15,000 per year
>
> Blocks is a very large file. A new file should be made available every week. Several recent copies should be kept. Download bandwidth should be at least 10gb/s.
>
> **State history:** Personnel: $5,000 per year, Infrastructure: $20,000 per year
>
> State history is a very large file. A new file should be made available every week. Several recent copies should be kept. Download bandwidth should be at least 10gb/s.
>
> **Trace history:** Personnel: $5,000 per year, Infrastructure: $15,000 per year
>
> New traces can be split into chunks. These new chunks need to be made available daily. Download bandwidth should be at least 10gb/s.
>
> **Firehose blocks:** Personnel: $5,000 per year, Infrastructure: $15,000 per year
>
> New firehose blocks files are created every 100 blocks. These new block files need to be made available daily. Download bandwidth should be at least 10gb/s.

Possible Options:

- Multiple teams can provide this service, potentially with one being the Central API Service.
- The frequency of updates and amount of copies of data can be increased or decreased (adjust infrastructure costs accordingly).
- The number of data download locations can be increased independently of increasing the number of teams providing the data download (adjust infrastructure costs accordingly).

Dependencies:

- The budget is not meant to cover the entire cost for a team to run EOSIO infrastructure, but just the incremental cost of making data available. It also assumes that an existing EOSIO “expert” team is providing the download service as the budget does not cover any training costs.

- For efficiency, downloads for “state history”, “trace history” and “firehose” should be provided by the same team providing blocks download. Costs need to be increased if not. One provider need not provide all types of data.
- An increased budget should be considered if nodeos development changes file formats causing “old” and “new” files to need to be maintained in parallel.
- An increased budget should be considered if there is a greater than average number of transactions on the network such that the download file sizes increase significantly.

	Minimal	Moderate	Strategic
Update the bp.json specification and validator	✓	✓	✓
Snapshots	2	3	4
Blocks	2	3	4
State history	1	2	3
Trace history	1	2	3
Firehose	1	2	3
Total Cost	**$160,000**	**$265,000**	**$370,000**

4. Recommendations

The following recommendations are made by the working group:

- To have multiple teams at the **Moderate** level provide downloadable data, each from a distinct location for the period of 1 year. Support would need to be re-evaluated annually.
- There should be a page on the main “EOSIO” or “ENF” website (wherever developers will be looking) that lists available download locations. Cost for this option has **not** been calculated because it is not clear how that will be managed as of this writing.

VII. Proposal 7: Rosetta

Creating an EOSIO implementation for Rosetta will make it much easier, or even qualify, EOSIO project tokens to be listed on exchanges. Today these exchanges have to build their own middleware to connect to an EOSIO network, meaning that technical teams must first gain a deeper understanding of the protocol and then develop an in-house solution. An official Rosetta implementation can greatly reduce the cost to integrate EOSIO tokens.

The key deliverable for this work stream is to have a Rosetta implementation for EOSIO that is used by the greatest number of exchanges in order to maximize liquidity for EOSIO based tokens. The logical first step is to understand exchanges' requirements in order to create a comprehensive technical documentation and input on criteria for contractor selection. The suggested structure to support this initial phase is to have an initial group of experienced operators and developers contributing within a few hours to participate in meetings with exchanges and actually write the technical specification.

An initial technical assessment of the solution has raised a few questions to be validated by the community, especially exchanges intending to use it, but consider that the design decisions should not materially affect the development efforts. I.e. implementation as middleware vs embedded on nodeos, indexing and querying historical data, searching transactions and events, and auditing/compliance.

1. Roadmap

The roadmap was structured in a way to make phases completely independent allowing the ENF to have decision points as demands become clearer and there is more information on the technical and compliance requirements for each deliverable.

1. Technical specification (8 weeks)
    a. Hold talks with exchanges and other key stakeholders to understand their demands.
    b. Validate key design decisions and write the technical specification
    c. Discuss and suggest structure going forward.
2. Data API + Construction API basics (16 weeks) - Rosetta is generally divided into Data APIs for retrieving Data from the blockchain, Construction APIs to build and submit transactions and Indexer APIs that implement extra endpoints for events and searches supporting additional integrations. In the first phase the implementation will focus on a basic implementation.

3. Indexer endpoints (8 weeks) - Additional development hours can be allocated to offer indexer capabilities implementing events and search endpoints.
4. Documentation (4 weeks) - Creating the documentation for operators and users to work with EOSIO Rosetta implementation.
5. Developer/exchange support (12 months after delivery)
6. Auditing - May be needed but was not budgeted.

2. Estimated Costs

Phase	Duration	Cost
Tech Spec	8 weeks	$25,000
Data API + Construction API	16 weeks	$75,000
Indexer endpoints	8 weeks	$35,000
Documentation	4 weeks	$7,500
Developer/exchange support	1 year	$22,500
Total Cost		**$165,000**

3. Recommendations

Fund the Tech Spec and commit to funding the development based on the feedback. The need for developer/exchange support should be validated with exchanges and the community.

VIII. Proposal 8: The Graph

1. Proposal

This proposal is to fund a team to investigate building The Graph support for EOSIO based on the Firehose. At the end of this project, EOSIO application developers would be able to create subgraphs for EOSIO-based chains. Several example subgraphs would be created and documentation would be provided showing examples of how to create EOSIO-based subgraphs. Developer support on telegram and/or discord would be available.

Graph Node already supports reading from The Firehose generally. However, the existing EOSIO Firehose is not currently supported. This proposal is to bring EOSIO Firehose up-to-date with other blockchains and add EOSIO support into Graph Node.

2. Rationale

Originally the graph only supported Ethereum-based chains. Since mid-2021, there has been work to expand to support projects like NEAR and Solana based on Firehose technology. It is expected that support for additional protocols will continue to be added to the Graph with the eventuality of it being ubiquitous. EOSIO based chains should not be left behind.

Developers are able to programmatically create a manifest which supplies information about data sources, templates, and some metadata for the subgraph and define a schema to store the data which indicates how to query the subgraph.

Some of the largest DeFi applications on ETH & other EVM compatible chains leverage The Graph to provide rich data to their user dashboards and visuals for in-depth data charts (ex: token price, 24h trading volume, TVL). Some of the largest DEX's such as PancakeSwap, SushiSwap, Synthetix, Uniswap V3, CurveFi signal large stake of GRT tokens to attract quality Indexers. EOSIO projects need to provide compatible interfaces.

DeFi analytic platforms & providers prefer using subgraphs for a few reasons:

- API & tooling to retrieve data is consistent between all subgraphs
- Reliable data source as it is being pulled directly from on-chain blocks
- Subgraphs are immutable and tamperproof

3. External Considerations

- This project depends on maintenance and operational funding for the Firehose.
- It is assumed that a team taking on this project will need to learn one or more technologies in this project and the (EOSIO, The Firehose and The Graph). If a team with expertise in all these projects is available, then project costs and delivery time could be shortened.
- The Graph is ever evolving. It should be expected to fund this project in future years as ongoing work is required to keep up with technology changes going forward.
- If, in the future, The Graph is seen as a core component to be used in many EOSIO-based projects, additional funding should be considered in the future to evolve The Graph in ways that showcase EOSIO functionality.
- The Graph Foundation may provide funding to add additional tech stacks like EOSIO. A formal partnership with the Graph Foundation and this source of funding should be investigated.

4. Estimated Costs

Due to the flexibility of how much investment can be applied to this project, the actual costs required per role are broken down into individual elements in this section. The number of actual team members selected will determine the overall price.

Each anticipated role is estimated for a 1-year cost, which can be broken down into smaller time increments based on the information provided.

> **Sr. Software Developer:** $100,000 to $150,000 per year
>
> Responsible for the research and development of a working solution. Works with many other teams such as Defi Applications and The Graph core team. Coordinates with project managers and technical writers. Provides technical support for operators and subgraph developers via telegram and/or discord. Proficiency is required in both Go (the Firehose) and Rust (Graph Node) programming languages.
>
> **Software Developer:** $70,000 to $100,000 per year
>
> Responsible for the development of example subgraphs that implement. Receives direton from the Project manager and Sr Software Developer. Provides technical support for operators and subgraph developers.

> **Project Manager:** $90,000 to $110,000 per year
>
> Responsible for overseeing the overall project, communicating with various external entities and key stakeholders, and facilitating the collection of research materials. Coordinates with external entities, architects, and technical writers.
>
> **Technical Writer:** $60,000 to $80,000 per year
>
> Responsible for assisting in the creation of operator and developer documentation, and other forms of written communication. Coordinates with both project management and architects.

In addition to the personnel costs, there would be costs to run infrastructure for the project for development and making example subgraphs available for public use. Estimate $20,000 per year.

A few options:

	Experienced Team, Moderate investment	Less-experienced team, Moderate investment	Strategic investment
Infrastructure	✓	✓	✓
Core software development	0.25	0.5	0.5
Subgraph software development	0.25	0.5	1
Project management	0.1	0.2	0.3
Documentation	0.2	0.3	0.5
Total Cost	**$109,500**	**$191,000**	**$268,000**

5. Recommendations

The following recommendations are made by the working group:

- Investigate forming an official partnership with The Graph Foundation.
- There are only a few developers with the combined knowledge of EOSIO, the Firehose and The Graph. Expand the pool of developers who are building on EOSIO by funding a **less-experienced team with Moderate investment** to explore how EOSIO-based blockchains can work with The Graph.

V. Conclusion

In the preceding chapters we have only scratched the surface of the items that can be explored. Hopefully we have provided enough information for the community to come to consensus that we need to keep the fragile parts of the EOSIO ecosystem stable and move forward quickly with modernization efforts.

The working group recommends the following actions be taken:

1. Determine where an **API Research and Standards Team** fits into the EOSIO ecosystem and build a team capable of delivering on a path forward.
2. Sponsor the development of improvements to the **Transaction Lifecycle** within the core software to improve the user experience and aid developers in building applications.
3. Support the public **Specialized APIs** that applications in EOS currently depend upon.
4. Decide on how best to operate a **Central API Service** to help provide access to both users and businesses that operate on EOS.
5. Research the viability of **Distributed API** infrastructure and its economic incentives.
6. Sponsor various teams to offer up the critical **Blockchain Data** that infrastructure operators need to get started quickly.
7. Start an initiative that explores **Rosetta** compatibility for EOSIO to adopt industry wide standards.
8. Develop the software required to integrate **The Graph** protocols with EOSIO to enable the creation of new data sets.

Listed below is the cost breakdown of each proposals' recommendation:

Proposal	Cost
1: API Research and Standards	$770,000
2: Transaction Lifecycle	$400,000
3: Specialized APIs	$1,110,000
4: Central API Service	TBD
5: Distributed API	$250,000
6: Blockchain Data Depot	$265,000
7: Rosetta	$165,000
8: The Graph	$191,000

Each will likely require some discussion, so please give us feedback on what is outlined.

1. What is the right level of investment for each of the initiatives we have outlined?
2. Where should future research be focused?
3. What did we miss?
4. What are the new emerging trends we need to take into consideration?

We propose that the idea of generating research should continue indefinitely with a requirement to create a new paper like this on a periodic basis. This can serve as the first edition. That way there is always a current view of the ecosystem within and outside of EOSIO.

VI. Acknowledgements

The API+ working group is composed of talented, veteran EOSIO developers from these three organizations.

- **EOS Nation:** The developers behind Pomelo, .gems, EOS Name Service, EOSX Block Explorer, and EOS Detective and a top ranked block producer. EOS Nation are also the sole infrastructure providers of dfuse on EOS and also maintain its code repository for EOSIO.
- **EOS Rio:** A genesis EOS block producer and core developer of Hyperion History, a full history solution that provides an optimized data structure and action format for EOSIO
- **Greymass:** A genesis block producer for many EOSIO blockchains working to facilitate the growth of distributed ledger technologies and the infrastructure powering them. Greymass is also the development team behind Anchor Wallet, Fuel, Unicove, EOSIO Signing Request Specification (ESR), as well as many core web and mobile app SDKs.