Access Sui Data

You can access Sui network data like Transactions , Checkpoints , Objects , Events , and more through the available interfaces. You can use this data in your application workflows, to analyze network behavior across applications or protocols of interest, or to perform audits on parts or the whole of the network.

This document outlines the interfaces that are currently available to access the Sui network data, along with an overview of how that's gradually evolving. Refer to the following definitions for release stages mentioned in this document:

Currently, you can use any of the following mechanisms to access Sui network data:

You can also use one of the future-oriented interfaces that are available in alpha or beta, but those are not recommended for production use.

You can currently get real-time or historical data from a Sui Full node. Retention period for historical data depends on the pruning strategy that node operators implement, though currently the default configuration for all Full nodes is to implicitly fall back on a Sui Foundation-managed key-value store for historical transaction data .

WebSocket-based JSON RPCs suix_subscribeEvent and suix_subscribeTransaction were deprecated in July 2024. Do not rely on those RPCs in your applications. Refer to Future data access interfaces to learn about a future alternative.

If you need more control over the types, granularity, and retention period of the data that you need in your application, or if you have specific query patterns that could be best served from a relational database, then you can set up your own custom indexer or reach out to a Data indexer operator that might already have set one up.

If you set up your own indexer, you are responsible for its ongoing maintenance and the related infrastructure and operational costs. You can reduce your costs by implementing a pruning strategy for the relational database by taking into account the retention needs of your application.

Primary interfaces to access Sui access data in the future include:

You could still utilize an RPC provider (filter by RPC) or a Data indexer operator with these options, assuming some or all of those providers choose to operate the Indexer 2.0 and the GraphQL RPC Server.

As mentioned previously, gRPC API will replace the JSON-RPC on Full nodes, such that JSON-RPC will be deprecated when gRPC API is generally available. Apart from the message and request format changes between the two, the gRPC API comes with a couple of key functional differences:

See When to use gRPC vs GraphQL with Indexer 2.0 for a comparison with GraphQL RPC.

The gRPC API is in beta, which is a somewhat stable release that is subject to change based on user feedback. You can use it for testing and production readiness in non-production environments.

High-level timeline

The target times indicated below are tentative and subject to updates based on project progress and your feedback.

As mentioned, Indexer 2.0 will include a performant and scalable implementation of the indexer framework. The underlying framework uses the Full node RPCs to ingest the data, initially using the current generally available JSON-RPC, and later using the gRPC API.

Indexer 2.0 will be declarative such that you can seamlessly configure it to load different kinds of Sui network data into Postgres relational tables in parallel. This change is being implemented to improve the performance of the data ingestion into the Postgres database. In addition, you can configure pruning for different tables in the Postgres database, allowing you to tune it for the desired combination of performance and cost characteristics.

Indexer 2.0 is currently in alpha and not recommended for production use. A sneak preview of how you could set up Indexer 2.0 today is available on GitHub .

The GraphQL RPC Server will provide a performant GraphQL RPC layer while reading data from the Indexer 2.0's Postgres database. GraphQL RPC will be an alternative to gRPC API. If you are already using JSON-RPC in your application today, you would have an option to migrate to GraphQL RPC by either operating the combined stack of Indexer 2.0, Postgres database, and GraphQL RPC server on your own, or by utilizing it as a service from an RPC provider or data indexer operator.

GraphQL RPC Server will be a lightweight server component that will allow you to combine data from multiple Postgres database tables using GraphQL's expressive querying system, which is appealing to frontend developers.

See When to use gRPC vs GraphQL with Indexer 2.0 for a comparison with the gRPC API.

GraphQL RPC Server is currently in alpha and not recommended for production use. Check out this getting started document for GraphQL RPCs that refers to the Mysten Labs-managed stack of GraphQL RPC Server along with the underlying Postgres database and Indexer 2.0.

Based on valuable feedback from the community, the GraphQL RPC release stage has been updated to alpha. Refer to the high-level timeline for beta and GA releases in this document.

High-level timeline

The target times indicated in this table are tentative and subject to updates based on project progress and your feedback.

You can use the high-level criteria mentioned in the following table to determine whether gRPC API or GraphQL RPC with Indexer 2.0 would better serve your use case. It's not an exhaustive list and it's expected that either of the options could work suitably for some of the use cases.

This table only mentions the default retention period for both options. The expectation is that it's reasonable for a Full node operator, RPC provider, or data indexer operator to configure that to a few times higher without significantly impacting the performance. Also by default, GraphQL RPC Server can directly connect to a archival key-value store for historical data beyond the retention period configured for the underlying Postgres database. Whereas in comparison, gRPC API will not have such direct connectivity to an archival key-value store.

Relevant guidelines will be provided before each option's respective GA release. Those will include recommendations for how to access historical data beyond the configured retention period for your interface of choice.

Refer to the following articles outlining general differences between gRPC and GraphQL. Please validate the accuracy and authenticity of the differences using your own experiments.

Current data access interfaces

Currently, you can use any of the following mechanisms to access Sui network data:

You can also use one of the future-oriented interfaces that are available in alpha or beta, but those are not recommended for production use.

You can currently get real-time or historical data from a Sui Full node. Retention period for historical data depends on the pruning strategy that node operators implement, though currently the default configuration for all Full nodes is to implicitly fall back on a Sui Foundation-managed key-value store for historical transaction data .

WebSocket-based JSON RPCs suix_subscribeEvent and suix_subscribeTransaction were deprecated in July 2024. Do not rely on those RPCs in your applications. Refer to Future data access interfaces to learn about a future alternative.

If you need more control over the types, granularity, and retention period of the data that you need in your application, or if you have specific query patterns that could be best served from a relational database, then you can set up your own custom indexer or reach out to a Data indexer operator that might already have set one up.

If you set up your own indexer, you are responsible for its ongoing maintenance and the related infrastructure and operational costs. You can reduce your costs by implementing a pruning strategy for the relational database by taking into account the retention needs of your application.

Primary interfaces to access Sui access data in the future include:

You could still utilize an RPC provider (filter by RPC) or a Data indexer operator with these options, assuming some or all of those providers choose to operate the Indexer 2.0 and the GraphQL RPC Server.

As mentioned previously, gRPC API will replace the JSON-RPC on Full nodes, such that JSON-RPC will be deprecated when gRPC API is generally available. Apart from the message and request format changes between the two, the gRPC API comes with a couple of key functional differences:

See When to use gRPC vs GraphQL with Indexer 2.0 for a comparison with GraphQL RPC.

The gRPC API is in beta, which is a somewhat stable release that is subject to change based on user feedback. You can use it for testing and production readiness in non-production environments.

High-level timeline

The target times indicated below are tentative and subject to updates based on project progress and your feedback.

As mentioned, Indexer 2.0 will include a performant and scalable implementation of the indexer framework. The underlying framework uses the Full node RPCs to ingest the data, initially using the current generally available JSON-RPC, and later using the gRPC API.

Indexer 2.0 will be declarative such that you can seamlessly configure it to load different kinds of Sui network data into Postgres relational tables in parallel. This change is being implemented to improve the performance of the data ingestion into the Postgres database. In addition, you can configure pruning for different tables in the Postgres database, allowing you to tune it for the desired combination of performance and cost characteristics.

Indexer 2.0 is currently in alpha and not recommended for production use. A sneak preview of how you could set up Indexer 2.0 today is available on GitHub .

The GraphQL RPC Server will provide a performant GraphQL RPC layer while reading data from the Indexer 2.0's Postgres database. GraphQL RPC will be an alternative to gRPC API. If you are already using JSON-RPC in your application today, you would have an option to migrate to GraphQL RPC by either operating the combined stack of Indexer 2.0, Postgres database, and GraphQL RPC server on your own, or by utilizing it as a service from an RPC provider or data indexer operator.

GraphQL RPC Server will be a lightweight server component that will allow you to combine data from multiple Postgres database tables using GraphQL's expressive querying system, which is appealing to frontend developers.

See When to use gRPC vs GraphQL with Indexer 2.0 for a comparison with the gRPC API.

GraphQL RPC Server is currently in alpha and not recommended for production use. Check out this getting started document for GraphQL RPCs that refers to the Mysten Labs-managed stack of GraphQL RPC Server along with the underlying Postgres database and Indexer 2.0.

Based on valuable feedback from the community, the GraphQL RPC release stage has been updated to alpha. Refer to the high-level timeline for beta and GA releases in this document.

High-level timeline

The target times indicated in this table are tentative and subject to updates based on project progress and your feedback.

You can use the high-level criteria mentioned in the following table to determine whether gRPC API or GraphQL RPC with Indexer 2.0 would better serve your use case. It's not an exhaustive list and it's expected that either of the options could work suitably for some of the use cases.

This table only mentions the default retention period for both options. The expectation is that it's reasonable for a Full node operator, RPC provider, or data indexer operator to configure that to a few times higher without significantly impacting the performance. Also by default, GraphQL RPC Server can directly connect to a archival key-value store for historical data beyond the retention period configured for the underlying Postgres database. Whereas in comparison, gRPC API will not have such direct connectivity to an archival key-value store.

Relevant guidelines will be provided before each option's respective GA release. Those will include recommendations for how to access historical data beyond the configured retention period for your interface of choice.

Refer to the following articles outlining general differences between gRPC and GraphQL. Please validate the accuracy and authenticity of the differences using your own experiments.

Future data access interfaces

Primary interfaces to access Sui access data in the future include:

You could still utilize an RPC provider (filter by RPC) or a Data indexer operator with these options, assuming some or all of those providers choose to operate the Indexer 2.0 and the GraphQL RPC Server.

As mentioned previously, gRPC API will replace the JSON-RPC on Full nodes, such that JSON-RPC will be deprecated when gRPC API is generally available. Apart from the message and request format changes between the two, the gRPC API comes with a couple of key functional differences:

See When to use gRPC vs GraphQL with Indexer 2.0 for a comparison with GraphQL RPC.

The gRPC API is in beta, which is a somewhat stable release that is subject to change based on user feedback. You can use it for testing and production readiness in non-production environments.

High-level timeline

The target times indicated below are tentative and subject to updates based on project progress and your feedback.

As mentioned, Indexer 2.0 will include a performant and scalable implementation of the indexer framework. The underlying framework uses the Full node RPCs to ingest the data, initially using the current generally available JSON-RPC, and later using the gRPC API.

Indexer 2.0 will be declarative such that you can seamlessly configure it to load different kinds of Sui network data into Postgres relational tables in parallel. This change is being implemented to improve the performance of the data ingestion into the Postgres database. In addition, you can configure pruning for different tables in the Postgres database, allowing you to tune it for the desired combination of performance and cost characteristics.

Indexer 2.0 is currently in alpha and not recommended for production use. A sneak preview of how you could set up Indexer 2.0 today is available on GitHub .

The GraphQL RPC Server will provide a performant GraphQL RPC layer while reading data from the Indexer 2.0's Postgres database. GraphQL RPC will be an alternative to gRPC API. If you are already using JSON-RPC in your application today, you would have an option to migrate to GraphQL RPC by either operating the combined stack of Indexer 2.0, Postgres database, and GraphQL RPC server on your own, or by utilizing it as a service from an RPC provider or data indexer operator.

GraphQL RPC Server will be a lightweight server component that will allow you to combine data from multiple Postgres database tables using GraphQL's expressive querying system, which is appealing to frontend developers.

See When to use gRPC vs GraphQL with Indexer 2.0 for a comparison with the gRPC API.

GraphQL RPC Server is currently in alpha and not recommended for production use. Check out this getting started document for GraphQL RPCs that refers to the Mysten Labs-managed stack of GraphQL RPC Server along with the underlying Postgres database and Indexer 2.0.

Based on valuable feedback from the community, the GraphQL RPC release stage has been updated to alpha. Refer to the high-level timeline for beta and GA releases in this document.

High-level timeline

The target times indicated in this table are tentative and subject to updates based on project progress and your feedback.

You can use the high-level criteria mentioned in the following table to determine whether gRPC API or GraphQL RPC with Indexer 2.0 would better serve your use case. It's not an exhaustive list and it's expected that either of the options could work suitably for some of the use cases.

This table only mentions the default retention period for both options. The expectation is that it's reasonable for a Full node operator, RPC provider, or data indexer operator to configure that to a few times higher without significantly impacting the performance. Also by default, GraphQL RPC Server can directly connect to a archival key-value store for historical data beyond the retention period configured for the underlying Postgres database. Whereas in comparison, gRPC API will not have such direct connectivity to an archival key-value store.

Relevant guidelines will be provided before each option's respective GA release. Those will include recommendations for how to access historical data beyond the configured retention period for your interface of choice.

Refer to the following articles outlining general differences between gRPC and GraphQL. Please validate the accuracy and authenticity of the differences using your own experiments.