gRPC Overview (Beta)

This content describes an alpha/beta feature or service. These early stage features and services are in active development, so details are likely to change.

This feature or service is currently available in

The Sui Full Node gRPC API provides a fast, type-safe, and efficient interface for interacting with the Sui blockchain. Designed for power users, indexers, explorers, and decentralized apps, this API enables access to Sui data with high performance and low latency.

Refer to Access Sui Data for an overview of options to access Sui network data.

gRPC offers a high-performance, efficient communication protocol that uses Protocol Buffers for fast, compact data serialization. Its strongly typed interfaces reduce runtime errors and simplify client/server development across multiple languages. With built-in support for code generation, you can scaffold clients in Typescript, Go, Rust, and more. This makes it ideal for scalable backend systems like indexers, blockchain explorers, and data-intensive decentralized apps.

In addition to request-response calls, gRPC supports server-side streaming, enabling real-time data delivery without constant polling. This is especially useful in environments where you need to track events and transactions live. gRPC's binary format is significantly faster and lighter than JSON, saving bandwidth and improving latency.

Refer to when to use gRPC vs GraphQL to access Sui data.

Protocol buffers define the gRPC interface. You can find the relevant beta .proto files at sui-rpc-api on Github , which apart from the gRPC messages (request and response payloads) include the following services and types:

These definitions can be used to generate client libraries in various programming languages.

There are some proto files in the folder sui/rpc/v2alpha as well. Those are are in alpha because they are early experimental versions that are subject to change and not recommended for production use.

The TransactionExecutionService currently offers a single RPC method: ExecuteTransaction(ExecuteTransactionRequest) , which is used to execute a transaction request. Whereas the LedgerService includes the core lookup queries for Sui data. Some of the RPCs in that service include:

A FieldMask in Protocol Buffers is a mechanism used to specify a subset of fields within a message that should be read, updated, or returned. Instead of retrieving the entire object, a client can request only the specific fields they need by providing a list of field paths. This improves performance and reduces unnecessary data transfer.

In the Sui gRPC API, FieldMask s are used in requests like GetTransaction to control which parts of the transaction (such as, effects , events) are included in the response. Field paths must match the structure of the response message. This selective querying is especially useful for building efficient applications and tools.

In the Sui gRPC API, identifiers with standard human-readable formats are represented as string s in the proto schema:

Simplest way to experiment with gRPC is by using grpcurl .

Your results might differ from the examples that follow, depending on the breadth and maturity of the gRPC APIs available on Sui Full nodes.

where the port on Sui Foundation managed Full nodes is 443 . It should return something like:

which should return something like:

This is an example to build a Typescript client for Sui gRPC API. If you want to use a different set of tools or modules that you're comfortable with, you can adjust the instructions accordingly.

Install dependencies

Project structure

Download all the sui/rpc/v2beta proto files from Github v2beta in the same folder.

Sample client.ts to get events and effects details of a particular transaction

Run the sample client

This is an example to build a golang client for Sui gRPC API. Feel free to use another set of tools or modules that you're comfortable with.

Install dependencies

First make sure you have go and protoc installed in your environment, and then install:

In your go.mod , add the following (make sure to update the version numbers to the latest versions):

Generate Golang code from proto files

Assuming you have the the proto files from Github v2beta , run:

Sample main.go to get events and effects details of a particular transaction

Run the sample client

If your go.mod is properly set up, and you've already generated the gRPC code (*.pb.go ,* _grpc.pb.go), simply run:

You can replace your_project with a module name or relative import path depending on how your go.mod is defined.

This is an example to build a python client for Sui gRPC API. Feel free to use another set of tools or modules that you're comfortable with.

Install dependencies

First make sure you have python and protoc installed in your environment, and then install:

Generate Python code from proto files

Assuming you have the the proto files from Github v2beta , run:

Sample client.py to get events and effects details of a particular transaction

Run the sample client

Q: In a batch object request (BatchGetObjects), does the field mask specified in individual GetObjectRequest s override the top-level field mask in the BatchGetObjectsRequest ?

Q: In ExecuteTransactionRequest , why is the transaction field marked as optional ?

What is gRPC?

gRPC offers a high-performance, efficient communication protocol that uses Protocol Buffers for fast, compact data serialization. Its strongly typed interfaces reduce runtime errors and simplify client/server development across multiple languages. With built-in support for code generation, you can scaffold clients in Typescript, Go, Rust, and more. This makes it ideal for scalable backend systems like indexers, blockchain explorers, and data-intensive decentralized apps.

In addition to request-response calls, gRPC supports server-side streaming, enabling real-time data delivery without constant polling. This is especially useful in environments where you need to track events and transactions live. gRPC's binary format is significantly faster and lighter than JSON, saving bandwidth and improving latency.

Refer to when to use gRPC vs GraphQL to access Sui data.

Protocol buffers define the gRPC interface. You can find the relevant beta .proto files at sui-rpc-api on Github , which apart from the gRPC messages (request and response payloads) include the following services and types:

These definitions can be used to generate client libraries in various programming languages.

There are some proto files in the folder sui/rpc/v2alpha as well. Those are are in alpha because they are early experimental versions that are subject to change and not recommended for production use.

The TransactionExecutionService currently offers a single RPC method: ExecuteTransaction(ExecuteTransactionRequest) , which is used to execute a transaction request. Whereas the LedgerService includes the core lookup queries for Sui data. Some of the RPCs in that service include:

A FieldMask in Protocol Buffers is a mechanism used to specify a subset of fields within a message that should be read, updated, or returned. Instead of retrieving the entire object, a client can request only the specific fields they need by providing a list of field paths. This improves performance and reduces unnecessary data transfer.

In the Sui gRPC API, FieldMask s are used in requests like GetTransaction to control which parts of the transaction (such as, effects , events) are included in the response. Field paths must match the structure of the response message. This selective querying is especially useful for building efficient applications and tools.

In the Sui gRPC API, identifiers with standard human-readable formats are represented as string s in the proto schema:

Simplest way to experiment with gRPC is by using grpcurl .

Your results might differ from the examples that follow, depending on the breadth and maturity of the gRPC APIs available on Sui Full nodes.

where the port on Sui Foundation managed Full nodes is 443 . It should return something like:

which should return something like:

This is an example to build a Typescript client for Sui gRPC API. If you want to use a different set of tools or modules that you're comfortable with, you can adjust the instructions accordingly.

Install dependencies

Project structure

Download all the sui/rpc/v2beta proto files from Github v2beta in the same folder.

Sample client.ts to get events and effects details of a particular transaction

Run the sample client

This is an example to build a golang client for Sui gRPC API. Feel free to use another set of tools or modules that you're comfortable with.

Install dependencies

First make sure you have go and protoc installed in your environment, and then install:

In your go.mod , add the following (make sure to update the version numbers to the latest versions):

Generate Golang code from proto files

Assuming you have the the proto files from Github v2beta , run:

Sample main.go to get events and effects details of a particular transaction

Run the sample client

If your go.mod is properly set up, and you've already generated the gRPC code (*.pb.go ,* _grpc.pb.go), simply run:

You can replace your_project with a module name or relative import path depending on how your go.mod is defined.

This is an example to build a python client for Sui gRPC API. Feel free to use another set of tools or modules that you're comfortable with.

Install dependencies

First make sure you have python and protoc installed in your environment, and then install:

Generate Python code from proto files

Assuming you have the the proto files from Github v2beta , run:

Sample client.py to get events and effects details of a particular transaction

Run the sample client

Q: In a batch object request (BatchGetObjects), does the field mask specified in individual GetObjectRequest s override the top-level field mask in the BatchGetObjectsRequest ?

Q: In ExecuteTransactionRequest , why is the transaction field marked as optional ?

gRPC on Sui

Protocol buffers define the gRPC interface. You can find the relevant beta .proto files at sui-rpc-api on Github , which apart from the gRPC messages (request and response payloads) include the following services and types:

These definitions can be used to generate client libraries in various programming languages.

There are some proto files in the folder sui/rpc/v2alpha as well. Those are are in alpha because they are early experimental versions that are subject to change and not recommended for production use.

The TransactionExecutionService currently offers a single RPC method: ExecuteTransaction(ExecuteTransactionRequest) , which is used to execute a transaction request. Whereas the LedgerService includes the core lookup queries for Sui data. Some of the RPCs in that service include:

A FieldMask in Protocol Buffers is a mechanism used to specify a subset of fields within a message that should be read, updated, or returned. Instead of retrieving the entire object, a client can request only the specific fields they need by providing a list of field paths. This improves performance and reduces unnecessary data transfer.

In the Sui gRPC API, FieldMask s are used in requests like GetTransaction to control which parts of the transaction (such as, effects , events) are included in the response. Field paths must match the structure of the response message. This selective querying is especially useful for building efficient applications and tools.

In the Sui gRPC API, identifiers with standard human-readable formats are represented as string s in the proto schema:

Simplest way to experiment with gRPC is by using grpcurl .

Your results might differ from the examples that follow, depending on the breadth and maturity of the gRPC APIs available on Sui Full nodes.

where the port on Sui Foundation managed Full nodes is 443 . It should return something like:

which should return something like:

This is an example to build a Typescript client for Sui gRPC API. If you want to use a different set of tools or modules that you're comfortable with, you can adjust the instructions accordingly.

Install dependencies

Project structure

Download all the sui/rpc/v2beta proto files from Github v2beta in the same folder.

Sample client.ts to get events and effects details of a particular transaction

Run the sample client

This is an example to build a golang client for Sui gRPC API. Feel free to use another set of tools or modules that you're comfortable with.

Install dependencies

First make sure you have go and protoc installed in your environment, and then install:

In your go.mod , add the following (make sure to update the version numbers to the latest versions):

Generate Golang code from proto files

Assuming you have the the proto files from Github v2beta , run:

Sample main.go to get events and effects details of a particular transaction

Run the sample client

If your go.mod is properly set up, and you've already generated the gRPC code (*.pb.go ,* _grpc.pb.go), simply run:

You can replace your_project with a module name or relative import path depending on how your go.mod is defined.

This is an example to build a python client for Sui gRPC API. Feel free to use another set of tools or modules that you're comfortable with.

Install dependencies

First make sure you have python and protoc installed in your environment, and then install:

Generate Python code from proto files

Assuming you have the the proto files from Github v2beta , run:

Sample client.py to get events and effects details of a particular transaction

Run the sample client

Q: In a batch object request (BatchGetObjects), does the field mask specified in individual GetObjectRequest s override the top-level field mask in the BatchGetObjectsRequest ?

Q: In ExecuteTransactionRequest , why is the transaction field marked as optional ?

Access using grpcurl

Simplest way to experiment with gRPC is by using grpcurl .

Your results might differ from the examples that follow, depending on the breadth and maturity of the gRPC APIs available on Sui Full nodes.

where the port on Sui Foundation managed Full nodes is 443 . It should return something like:

which should return something like:

This is an example to build a Typescript client for Sui gRPC API. If you want to use a different set of tools or modules that you're comfortable with, you can adjust the instructions accordingly.

Install dependencies

Project structure

Download all the sui/rpc/v2beta proto files from Github v2beta in the same folder.

Sample client.ts to get events and effects details of a particular transaction

Run the sample client

This is an example to build a golang client for Sui gRPC API. Feel free to use another set of tools or modules that you're comfortable with.

Install dependencies

First make sure you have go and protoc installed in your environment, and then install:

In your go.mod , add the following (make sure to update the version numbers to the latest versions):

Generate Golang code from proto files

Assuming you have the the proto files from Github v2beta , run:

Sample main.go to get events and effects details of a particular transaction

Run the sample client

If your go.mod is properly set up, and you've already generated the gRPC code (*.pb.go ,* _grpc.pb.go), simply run:

You can replace your_project with a module name or relative import path depending on how your go.mod is defined.

This is an example to build a python client for Sui gRPC API. Feel free to use another set of tools or modules that you're comfortable with.

Install dependencies

First make sure you have python and protoc installed in your environment, and then install:

Generate Python code from proto files

Assuming you have the the proto files from Github v2beta , run:

Sample client.py to get events and effects details of a particular transaction

Run the sample client

Q: In a batch object request (BatchGetObjects), does the field mask specified in individual GetObjectRequest s override the top-level field mask in the BatchGetObjectsRequest ?

Q: In ExecuteTransactionRequest , why is the transaction field marked as optional ?

Sample clients in different programming languages

This is an example to build a Typescript client for Sui gRPC API. If you want to use a different set of tools or modules that you're comfortable with, you can adjust the instructions accordingly.

Install dependencies

Project structure

Download all the sui/rpc/v2beta proto files from Github v2beta in the same folder.

Sample client.ts to get events and effects details of a particular transaction

Run the sample client

This is an example to build a golang client for Sui gRPC API. Feel free to use another set of tools or modules that you're comfortable with.

Install dependencies

First make sure you have go and protoc installed in your environment, and then install:

In your go.mod , add the following (make sure to update the version numbers to the latest versions):

Generate Golang code from proto files

Assuming you have the the proto files from Github v2beta , run:

Sample main.go to get events and effects details of a particular transaction

Run the sample client

If your go.mod is properly set up, and you've already generated the gRPC code (*.pb.go ,* _grpc.pb.go), simply run:

You can replace your_project with a module name or relative import path depending on how your go.mod is defined.

This is an example to build a python client for Sui gRPC API. Feel free to use another set of tools or modules that you're comfortable with.

Install dependencies

First make sure you have python and protoc installed in your environment, and then install:

Generate Python code from proto files

Assuming you have the the proto files from Github v2beta , run:

Sample client.py to get events and effects details of a particular transaction

Run the sample client

Q: In a batch object request (BatchGetObjects), does the field mask specified in individual GetObjectRequest s override the top-level field mask in the BatchGetObjectsRequest ?

Q: In ExecuteTransactionRequest , why is the transaction field marked as optional ?

Frequently asked questions

Q: In a batch object request (BatchGetObjects), does the field mask specified in individual GetObjectRequest s override the top-level field mask in the BatchGetObjectsRequest ?

Q: In ExecuteTransactionRequest , why is the transaction field marked as optional ?