Sponsored Transactions

A Sui sponsored transaction is one where a Sui address (the sponsor's) pays the gas fees for a transaction that another address (the user's) initializes. You can use sponsored transactions to cover the fees for users on your site or app so that they don't get charged for them. This removes a significant obstacle that web 2.0 users encounter when entering web3, as they often have to purchase tokens to perform a transaction on chain. For example, you could sponsor gamers' early transactions to increase conversion rates.

Sponsored transactions also facilitate asset management as you don't need to maintain multiple accounts with SUI tokens to transfer funds.

You can use Sui sponsored transactions to:

The most significant potential risk when using sponsored transactions is equivocation . In some cases under certain conditions, a sponsored transaction can result in all associated owned objects, including gas in a locked state when examined by Sui validators. To avoid double spending, validators lock objects as they validate transactions. An equivocation occurs when an owned object's pair (ObjectID , SequenceNumber) is concurrently used in multiple non-finalized transactions.

To equivocate, either the user or the sponsor signs and submits another transaction that attempts to manipulate an owned object in the original transaction. Because only the object owner can use an owned object, only the user and sponsor can cause this condition.

A user-initiated sponsored transaction involves the following steps:

GasLessTransactionData is basically TransactionData without GasData . It is not a sui-core data structure, but it is only an interface between user and sponsor.

The following example constructs a GasLessTransactionData object.

A sponsor-initiated sponsored transaction involves the following steps:

You can use a sponsor-initiated sponsored transaction as an advertiser, or to incentivize specific user actions without requiring the user to pay for gas fees.

To use a GasData object to sponsor the gas fees for a transaction, create a GasData object that covers the fees determined for the transaction. This is similar to providing a blank check to a user that can be used only to cover gas fees. The user doesn't need to know how much the fee is or approve it.

A sponsor transaction using a GasData object involves the following steps:

On Sui, a gas station is a concept to describe where you set up processes to sponsor user transactions. You can customize a Sui gas station to support the specific user-facing functionality you need. Some example use cases for a Sui gas station include:

Depending on the nature of your gas station, you can apply different authorization rules to avoid being spammed by bad actors. Possible policies include:

For all gas objects that you provide as a sponsor, you should track if users ever try to equivocate and lock objects. If you detect such behavior, block the user or requester accordingly.

The following Rust SDK code examples demonstrate how to implement a Sui gas station that supports each of the sponsored transaction types described previously.

Use the API endpoint to receive GaslessTransaction transactions and return a sole-signed SenderSignedData object.

Use the API endpoint to receive sole-signed SenderSignedData and return the result of the transaction.

Alternatively, use the API endpoint to return a GasData object.

Use the API endpoint to receive dual-signed SenderSignedData and return the result of the transaction.

For user and sponsor-initiated transactions, users can submit the dual-signed transaction via either a sponsor or a Full node.

The following code block describes the TransactionData structure for sponsored transactions and GasObject . You can view the source code in the Sui GitHub repository.

TransactionData Structure

GasData Structure

To learn more about transactions in Sui, see Transactions .

Potential risks using sponsored transactions

The most significant potential risk when using sponsored transactions is equivocation . In some cases under certain conditions, a sponsored transaction can result in all associated owned objects, including gas in a locked state when examined by Sui validators. To avoid double spending, validators lock objects as they validate transactions. An equivocation occurs when an owned object's pair (ObjectID , SequenceNumber) is concurrently used in multiple non-finalized transactions.

To equivocate, either the user or the sponsor signs and submits another transaction that attempts to manipulate an owned object in the original transaction. Because only the object owner can use an owned object, only the user and sponsor can cause this condition.

A user-initiated sponsored transaction involves the following steps:

GasLessTransactionData is basically TransactionData without GasData . It is not a sui-core data structure, but it is only an interface between user and sponsor.

The following example constructs a GasLessTransactionData object.

A sponsor-initiated sponsored transaction involves the following steps:

You can use a sponsor-initiated sponsored transaction as an advertiser, or to incentivize specific user actions without requiring the user to pay for gas fees.

To use a GasData object to sponsor the gas fees for a transaction, create a GasData object that covers the fees determined for the transaction. This is similar to providing a blank check to a user that can be used only to cover gas fees. The user doesn't need to know how much the fee is or approve it.

A sponsor transaction using a GasData object involves the following steps:

On Sui, a gas station is a concept to describe where you set up processes to sponsor user transactions. You can customize a Sui gas station to support the specific user-facing functionality you need. Some example use cases for a Sui gas station include:

Depending on the nature of your gas station, you can apply different authorization rules to avoid being spammed by bad actors. Possible policies include:

For all gas objects that you provide as a sponsor, you should track if users ever try to equivocate and lock objects. If you detect such behavior, block the user or requester accordingly.

The following Rust SDK code examples demonstrate how to implement a Sui gas station that supports each of the sponsored transaction types described previously.

Use the API endpoint to receive GaslessTransaction transactions and return a sole-signed SenderSignedData object.

Use the API endpoint to receive sole-signed SenderSignedData and return the result of the transaction.

Alternatively, use the API endpoint to return a GasData object.

Use the API endpoint to receive dual-signed SenderSignedData and return the result of the transaction.

For user and sponsor-initiated transactions, users can submit the dual-signed transaction via either a sponsor or a Full node.

The following code block describes the TransactionData structure for sponsored transactions and GasObject . You can view the source code in the Sui GitHub repository.

TransactionData Structure

GasData Structure

To learn more about transactions in Sui, see Transactions .

Create a user-initiated sponsored transaction

A user-initiated sponsored transaction involves the following steps:

GasLessTransactionData is basically TransactionData without GasData . It is not a sui-core data structure, but it is only an interface between user and sponsor.

The following example constructs a GasLessTransactionData object.

A sponsor-initiated sponsored transaction involves the following steps:

You can use a sponsor-initiated sponsored transaction as an advertiser, or to incentivize specific user actions without requiring the user to pay for gas fees.

To use a GasData object to sponsor the gas fees for a transaction, create a GasData object that covers the fees determined for the transaction. This is similar to providing a blank check to a user that can be used only to cover gas fees. The user doesn't need to know how much the fee is or approve it.

A sponsor transaction using a GasData object involves the following steps:

On Sui, a gas station is a concept to describe where you set up processes to sponsor user transactions. You can customize a Sui gas station to support the specific user-facing functionality you need. Some example use cases for a Sui gas station include:

Depending on the nature of your gas station, you can apply different authorization rules to avoid being spammed by bad actors. Possible policies include:

For all gas objects that you provide as a sponsor, you should track if users ever try to equivocate and lock objects. If you detect such behavior, block the user or requester accordingly.

The following Rust SDK code examples demonstrate how to implement a Sui gas station that supports each of the sponsored transaction types described previously.

Use the API endpoint to receive GaslessTransaction transactions and return a sole-signed SenderSignedData object.

Use the API endpoint to receive sole-signed SenderSignedData and return the result of the transaction.

Alternatively, use the API endpoint to return a GasData object.

Use the API endpoint to receive dual-signed SenderSignedData and return the result of the transaction.

For user and sponsor-initiated transactions, users can submit the dual-signed transaction via either a sponsor or a Full node.

The following code block describes the TransactionData structure for sponsored transactions and GasObject . You can view the source code in the Sui GitHub repository.

TransactionData Structure

GasData Structure

To learn more about transactions in Sui, see Transactions .

Create a sponsor-initiated sponsored transaction

A sponsor-initiated sponsored transaction involves the following steps:

You can use a sponsor-initiated sponsored transaction as an advertiser, or to incentivize specific user actions without requiring the user to pay for gas fees.

To use a GasData object to sponsor the gas fees for a transaction, create a GasData object that covers the fees determined for the transaction. This is similar to providing a blank check to a user that can be used only to cover gas fees. The user doesn't need to know how much the fee is or approve it.

A sponsor transaction using a GasData object involves the following steps:

On Sui, a gas station is a concept to describe where you set up processes to sponsor user transactions. You can customize a Sui gas station to support the specific user-facing functionality you need. Some example use cases for a Sui gas station include:

Depending on the nature of your gas station, you can apply different authorization rules to avoid being spammed by bad actors. Possible policies include:

For all gas objects that you provide as a sponsor, you should track if users ever try to equivocate and lock objects. If you detect such behavior, block the user or requester accordingly.

The following Rust SDK code examples demonstrate how to implement a Sui gas station that supports each of the sponsored transaction types described previously.

Use the API endpoint to receive GaslessTransaction transactions and return a sole-signed SenderSignedData object.

Use the API endpoint to receive sole-signed SenderSignedData and return the result of the transaction.

Alternatively, use the API endpoint to return a GasData object.

Use the API endpoint to receive dual-signed SenderSignedData and return the result of the transaction.

For user and sponsor-initiated transactions, users can submit the dual-signed transaction via either a sponsor or a Full node.

The following code block describes the TransactionData structure for sponsored transactions and GasObject . You can view the source code in the Sui GitHub repository.

TransactionData Structure

GasData Structure

To learn more about transactions in Sui, see Transactions .

Create sponsored transactions using a GasData object

To use a GasData object to sponsor the gas fees for a transaction, create a GasData object that covers the fees determined for the transaction. This is similar to providing a blank check to a user that can be used only to cover gas fees. The user doesn't need to know how much the fee is or approve it.

A sponsor transaction using a GasData object involves the following steps:

On Sui, a gas station is a concept to describe where you set up processes to sponsor user transactions. You can customize a Sui gas station to support the specific user-facing functionality you need. Some example use cases for a Sui gas station include:

Depending on the nature of your gas station, you can apply different authorization rules to avoid being spammed by bad actors. Possible policies include:

For all gas objects that you provide as a sponsor, you should track if users ever try to equivocate and lock objects. If you detect such behavior, block the user or requester accordingly.

The following Rust SDK code examples demonstrate how to implement a Sui gas station that supports each of the sponsored transaction types described previously.

Use the API endpoint to receive GaslessTransaction transactions and return a sole-signed SenderSignedData object.

Use the API endpoint to receive sole-signed SenderSignedData and return the result of the transaction.

Alternatively, use the API endpoint to return a GasData object.

Use the API endpoint to receive dual-signed SenderSignedData and return the result of the transaction.

For user and sponsor-initiated transactions, users can submit the dual-signed transaction via either a sponsor or a Full node.

The following code block describes the TransactionData structure for sponsored transactions and GasObject . You can view the source code in the Sui GitHub repository.

TransactionData Structure

GasData Structure

To learn more about transactions in Sui, see Transactions .

Create a Sui gas station

On Sui, a gas station is a concept to describe where you set up processes to sponsor user transactions. You can customize a Sui gas station to support the specific user-facing functionality you need. Some example use cases for a Sui gas station include:

Depending on the nature of your gas station, you can apply different authorization rules to avoid being spammed by bad actors. Possible policies include:

For all gas objects that you provide as a sponsor, you should track if users ever try to equivocate and lock objects. If you detect such behavior, block the user or requester accordingly.

The following Rust SDK code examples demonstrate how to implement a Sui gas station that supports each of the sponsored transaction types described previously.

Use the API endpoint to receive GaslessTransaction transactions and return a sole-signed SenderSignedData object.

Use the API endpoint to receive sole-signed SenderSignedData and return the result of the transaction.

Alternatively, use the API endpoint to return a GasData object.

Use the API endpoint to receive dual-signed SenderSignedData and return the result of the transaction.

For user and sponsor-initiated transactions, users can submit the dual-signed transaction via either a sponsor or a Full node.

The following code block describes the TransactionData structure for sponsored transactions and GasObject . You can view the source code in the Sui GitHub repository.

TransactionData Structure

GasData Structure

To learn more about transactions in Sui, see Transactions .

Code examples to create a Sui gas station

The following Rust SDK code examples demonstrate how to implement a Sui gas station that supports each of the sponsored transaction types described previously.

Use the API endpoint to receive GaslessTransaction transactions and return a sole-signed SenderSignedData object.

Use the API endpoint to receive sole-signed SenderSignedData and return the result of the transaction.

Alternatively, use the API endpoint to return a GasData object.

Use the API endpoint to receive dual-signed SenderSignedData and return the result of the transaction.

For user and sponsor-initiated transactions, users can submit the dual-signed transaction via either a sponsor or a Full node.

The following code block describes the TransactionData structure for sponsored transactions and GasObject . You can view the source code in the Sui GitHub repository.

TransactionData Structure

GasData Structure

To learn more about transactions in Sui, see Transactions .

Sponsored transaction data structure

The following code block describes the TransactionData structure for sponsored transactions and GasObject . You can view the source code in the Sui GitHub repository.

TransactionData Structure

GasData Structure

To learn more about transactions in Sui, see Transactions .