Sponsored Transactions

A transaction on Sui takes a payment to execute. The payment, also known as gas, is a list of 0x2::coin::Coin<0x2::sui::Sui> objects. Although gas is a critical piece in Sui tokenomics, it sometimes adds challenges when new Web3 users start to navigate on Sui.

Sponsored transactions are a primitive on the Sui blockchain that enable the execution of a transaction without a user paying the gas. Sponsored transactions can reduce the onboarding friction for users because the feature streamlines the process for end users. Using sponsored transactions, you can execute a transaction without requiring the user to pay it themselves. Instead, you can act as a sponsor of the transaction, offering your own payment gas objects for the transaction.

In a sponsored transaction there are three roles: the user, the gas station, and the sponsor.

It's not uncommon for the gas station and the sponsor to be the same entity. For example, a web3 gaming studio could run its own gas station to sponsor users with real free-to-play gaming experiences at its user acquisition stage. Because it's not always trivial to maintain a gas station for teams of any size, that gaming studio could also leverage third-party gas stations to sponsor transactions they want to promote.

The remainder of this guide assumes the sponsor uses their own gas station.

The following sections describe some common scenarios where sponsored transactions offer an improved user experience.

In this scenario, the sponsor has a specific set of applications they want to sponsor.

In this scenario, the sponsor has few restrictions on the type of transactions the gas payment can be used for.

A sponsored transaction is not restricted to these use cases. Essentially, a sponsored transaction is any transaction jointly made by the user and the sponsor. As long as the stakeholders can agree on the transaction details, then the number of possible ways to provide sponsored transactions is limited only by the imagination. Because at least two stakeholders are involved in a sponsored transaction, however, there are some additional risks that you should take steps to mitigate.

This section is mostly for developers who are interested in building a gas station or integrating with one.

The data structure of a transaction resembles the following:

A few details of note for the preceding code:

So, to construct a correct sponsored transaction, you must first build a TransactionData object. If you are neither the user or the sponsor, you would then pass the transaction to both parties to sign. If you're the sponsor, you would sign the transaction and then pass it and the signature to the other party (in the form of SenderSignedTransaction) for them to sign. In practice, the latter is the more common scenario.

There are three flows of sponsored transaction.

User proposed transaction

(swimlane link)

Sponsor proposed transaction

(swimlane link)

Wildcard gas payment

(swimlane link)

Because at least two stakeholders are involved in a sponsored transaction, you should take steps to mitigate risk.

Client equivocation happens when more than one legit transaction that shares at least one owned object (such as a gas coin object) at a certain version are submitted to the network simultaneously. On Sui, before a transaction is executed, owned objects in this transaction are locked on validators at specific versions. An honest validator only accepts one transaction and rejects others. Depending on the order validators receive these transactions, validators might accept different transactions. In the event of no single transaction getting accepted by at least 2/3rds of validators, the owned object is locked until end of the epoch.

Practically speaking, client equivocation is rare, mostly caused by buggy client software. After all, no one has incentives to lock their

own objects. However, sponsored transactions come with counterparty risks. For example, a malicious user could equivocate the gas station's gas coin object by submitting another transaction that uses one owned object in the gas station signed transaction at the same version. Similarly, a Byzantine gas station could do the same to the user owned objects.

Although this risk might seem trivial, it is helpful to be aware of it. Your gas station should actively monitor user behavior and alert on anything abnormal. Whether you're a user taking advantage of sponsored transactions or a developer integrating with a gas station, consider your reputation to minimize the risk of client equivocation.

Both the user and the sponsor need to sign over the entire TransactionData , including GasData because otherwise a third party (such as a malicious Full node) could snip the partially signed data and cause client equivocation and locking of owned objects.

If you chooses to submit the dual-signed transaction to the sponsor or gas station rather than a Full node, the transaction might be subject to sponsor or gas station censorship. Namely, the sponsor might choose not to submit the transaction to the network, or delay the submission.

You can mitigate this risk by submitting the transaction directly to a Full node.

Roles in sponsored transactions

In a sponsored transaction there are three roles: the user, the gas station, and the sponsor.

It's not uncommon for the gas station and the sponsor to be the same entity. For example, a web3 gaming studio could run its own gas station to sponsor users with real free-to-play gaming experiences at its user acquisition stage. Because it's not always trivial to maintain a gas station for teams of any size, that gaming studio could also leverage third-party gas stations to sponsor transactions they want to promote.

The remainder of this guide assumes the sponsor uses their own gas station.

The following sections describe some common scenarios where sponsored transactions offer an improved user experience.

In this scenario, the sponsor has a specific set of applications they want to sponsor.

In this scenario, the sponsor has few restrictions on the type of transactions the gas payment can be used for.

A sponsored transaction is not restricted to these use cases. Essentially, a sponsored transaction is any transaction jointly made by the user and the sponsor. As long as the stakeholders can agree on the transaction details, then the number of possible ways to provide sponsored transactions is limited only by the imagination. Because at least two stakeholders are involved in a sponsored transaction, however, there are some additional risks that you should take steps to mitigate.

This section is mostly for developers who are interested in building a gas station or integrating with one.

The data structure of a transaction resembles the following:

A few details of note for the preceding code:

So, to construct a correct sponsored transaction, you must first build a TransactionData object. If you are neither the user or the sponsor, you would then pass the transaction to both parties to sign. If you're the sponsor, you would sign the transaction and then pass it and the signature to the other party (in the form of SenderSignedTransaction) for them to sign. In practice, the latter is the more common scenario.

There are three flows of sponsored transaction.

User proposed transaction

(swimlane link)

Sponsor proposed transaction

(swimlane link)

Wildcard gas payment

(swimlane link)

Because at least two stakeholders are involved in a sponsored transaction, you should take steps to mitigate risk.

Client equivocation happens when more than one legit transaction that shares at least one owned object (such as a gas coin object) at a certain version are submitted to the network simultaneously. On Sui, before a transaction is executed, owned objects in this transaction are locked on validators at specific versions. An honest validator only accepts one transaction and rejects others. Depending on the order validators receive these transactions, validators might accept different transactions. In the event of no single transaction getting accepted by at least 2/3rds of validators, the owned object is locked until end of the epoch.

Practically speaking, client equivocation is rare, mostly caused by buggy client software. After all, no one has incentives to lock their own objects. However, sponsored transactions come with counterparty risks. For example, a malicious user could equivocate the gas station's gas coin object by submitting another transaction that uses one owned object in the gas station signed transaction at the same version. Similarly, a Byzantine gas station could do the same to the user owned objects.

Although this risk might seem trivial, it is helpful to be aware of it. Your gas station should actively monitor user behavior and alert on anything abnormal. Whether you're a user taking advantage of sponsored transactions or a developer integrating with a gas station, consider your reputation to minimize the risk of client equivocation.

Both the user and the sponsor need to sign over the entire TransactionData , including GasData because otherwise a third party (such as a malicious Full node) could snip the partially signed data and cause client equivocation and locking of owned objects.

If you chooses to submit the dual-signed transaction to the sponsor or gas station rather than a Full node, the transaction might be subject to sponsor or gas station censorship. Namely, the sponsor might choose not to submit the transaction to the network, or delay the submission.

You can mitigate this risk by submitting the transaction directly to a Full node.

Use cases

The following sections describe some common scenarios where sponsored transactions offer an improved user experience.

In this scenario, the sponsor has a specific set of applications they want to sponsor.

In this scenario, the sponsor has few restrictions on the type of transactions the gas payment can be used for.

A sponsored transaction is not restricted to these use cases. Essentially, a sponsored transaction is any transaction jointly made by the user and the sponsor. As long as the stakeholders can agree on the transaction details, then the number of possible ways to provide sponsored transactions is limited only by the imagination. Because at least two stakeholders are involved in a sponsored transaction, however, there are some additional risks that you should take steps to mitigate.

This section is mostly for developers who are interested in building a gas station or integrating with one.

The data structure of a transaction resembles the following:

A few details of note for the preceding code:

So, to construct a correct sponsored transaction, you must first build a TransactionData object. If you are neither the user or the sponsor, you would then pass the transaction to both parties to sign. If you're the sponsor, you would sign the transaction and then pass it and the signature to the other party (in the form of SenderSignedTransaction) for them to sign. In practice, the latter is the more common scenario.

There are three flows of sponsored transaction.

User proposed transaction

(swimlane link)

Sponsor proposed transaction

(swimlane link)

Wildcard gas payment

(swimlane link)

Because at least two stakeholders are involved in a sponsored transaction, you should take steps to mitigate risk.

Client equivocation happens when more than one legit transaction that shares at least one owned object (such as a gas coin object)

at a certain version are submitted to the network simultaneously. On Sui, before a transaction is executed, owned objects in this transaction are locked on validators at specific versions. An honest validator only accepts one transaction and rejects others. Depending on the order validators receive these transactions, validators might accept different transactions. In the event of no single transaction getting accepted by at least 2/3rds of validators, the owned object is locked until end of the epoch.

Practically speaking, client equivocation is rare, mostly caused by buggy client software. After all, no one has incentives to lock their own objects. However, sponsored transactions come with counterparty risks. For example, a malicious user could equivocate the gas station's gas coin object by submitting another transaction that uses one owned object in the gas station signed transaction at the same version. Similarly, a Byzantine gas station could do the same to the user owned objects.

Although this risk might seem trivial, it is helpful to be aware of it. Your gas station should actively monitor user behavior and alert on anything abnormal. Whether you're a user taking advantage of sponsored transactions or a developer integrating with a gas station, consider your reputation to minimize the risk of client equivocation.

Both the user and the sponsor need to sign over the entire TransactionData , including GasData because otherwise a third party (such as a malicious Full node) could snip the partially signed data and cause client equivocation and locking of owned objects.

If you chooses to submit the dual-signed transaction to the sponsor or gas station rather than a Full node, the transaction might be subject to sponsor or gas station censorship. Namely, the sponsor might choose not to submit the transaction to the network, or delay the submission.

You can mitigate this risk by submitting the transaction directly to a Full node.

Sponsored transaction flow

This section is mostly for developers who are interested in building a gas station or integrating with one.

The data structure of a transaction resembles the following:

A few details of note for the preceding code:

So, to construct a correct sponsored transaction, you must first build a TransactionData object. If you are neither the user or the sponsor, you would then pass the transaction to both parties to sign. If you're the sponsor, you would sign the transaction and then pass it and the signature to the other party (in the form of SenderSignedTransaction) for them to sign. In practice, the latter is the more common scenario.

There are three flows of sponsored transaction.

User proposed transaction

(swimlane link)

Sponsor proposed transaction

(swimlane link)

Wildcard gas payment

(swimlane link)

Because at least two stakeholders are involved in a sponsored transaction, you should take steps to mitigate risk.

Client equivocation happens when more than one legit transaction that shares at least one owned object (such as a gas coin object) at a certain version are submitted to the network simultaneously. On Sui, before a transaction is executed, owned objects in this transaction are locked on validators at specific versions. An honest validator only accepts one transaction and rejects others. Depending on the order validators receive these transactions, validators might accept different transactions. In the event of no single transaction getting accepted by at least 2/3rds of validators, the owned object is locked until end of the epoch.

Practically speaking, client equivocation is rare, mostly caused by buggy client software. After all, no one has incentives to lock their own objects. However, sponsored transactions come with counterparty risks. For example, a malicious user could equivocate the gas station's gas coin object by submitting another transaction that uses one owned object in the gas station signed transaction at the same version. Similarly, a Byzantine gas station could do the same to the user owned objects.

Although this risk might seem trivial, it is helpful to be aware of it. Your gas station should actively monitor user behavior and alert on

anything abnormal. Whether you're a user taking advantage of sponsored transactions or a developer integrating with a gas station, consider your reputation to minimize the risk of client equivocation.

Both the user and the sponsor need to sign over the entire TransactionData , including GasData because otherwise a third party (such as a malicious Full node) could snip the partially signed data and cause client equivocation and locking of owned objects.

If you chooses to submit the dual-signed transaction to the sponsor or gas station rather than a Full node, the transaction might be subject to sponsor or gas station censorship. Namely, the sponsor might choose not to submit the transaction to the network, or delay the submission.

You can mitigate this risk by submitting the transaction directly to a Full node.

Risk considerations

Because at least two stakeholders are involved in a sponsored transaction, you should take steps to mitigate risk.

Client equivocation happens when more than one legit transaction that shares at least one owned object (such as a gas coin object) at a certain version are submitted to the network simultaneously. On Sui, before a transaction is executed, owned objects in this transaction are locked on validators at specific versions. An honest validator only accepts one transaction and rejects others. Depending on the order validators receive these transactions, validators might accept different transactions. In the event of no single transaction getting accepted by at least 2/3rds of validators, the owned object is locked until end of the epoch.

Practically speaking, client equivocation is rare, mostly caused by buggy client software. After all, no one has incentives to lock their own objects. However, sponsored transactions come with counterparty risks. For example, a malicious user could equivocate the gas station's gas coin object by submitting another transaction that uses one owned object in the gas station signed transaction at the same version. Similarly, a Byzantine gas station could do the same to the user owned objects.

Although this risk might seem trivial, it is helpful to be aware of it. Your gas station should actively monitor user behavior and alert on anything abnormal. Whether you're a user taking advantage of sponsored transactions or a developer integrating with a gas station, consider your reputation to minimize the risk of client equivocation.

Both the user and the sponsor need to sign over the entire TransactionData , including GasData because otherwise a third party (such as a malicious Full node) could snip the partially signed data and cause client equivocation and locking of owned objects.

If you chooses to submit the dual-signed transaction to the sponsor or gas station rather than a Full node, the transaction might be subject to sponsor or gas station censorship. Namely, the sponsor might choose not to submit the transaction to the network, or delay the submission.

You can mitigate this risk by submitting the transaction directly to a Full node.