Transaction signatures

This guide explains how transactions are signed by the Safe owners using the Protocol Kit.

Before starting, check this guide'ssetup .

Create the transaction

ThecreateTransaction method in the Protocol Kit allows the creation of new Safe transactions and returns an instance of theEthSafeTransaction class.

```
// Create a transaction to send 0.01 ETH const

safeTransactionData :

SafeTransactionDataPartial

= { to :

'0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1' , value :

'10000000000000000' ,

// 0.01 ETH data :

'0x' }

let safeTransaction =

await

protocolKit .createTransaction ({ transactions : [safeTransactionData] })
```

The returnedsafeTransaction object contains the transaction data (safeTransaction.data) and a map of the owner-signature pairs (safeTransaction.signatures). The structure is similar to theEthSafeMessage class but applied for transactions instead of messages.

We uselet to initialize thesafeTransaction variable because we will add the signatures later.

```
class

EthSafeTransaction

implements

SafeTransaction { data :

SafeTransactionData signatures :

Map < string ,

SafeSignature

      =

new

Map () ... // Other properties and methods }
```

Sign the transaction

Once thesafeTransaction object is created, we need to collect the signatures from the signers who will sign it.

Following oursetup , we will sign a Safe transaction fromsafe3_4 , the main Safe account in this guide. To do that, we first need to sign the same transaction with its owners:owner1 ,owner2 ,safe1_1 , andsafe2_3 .

ECDSA signature

This applies toowner1 andowner2 accounts, as both are EOAs.

ThesignTransaction method takes thesafeTransaction together with aSigningMethod and adds the new signature to thesafeTransaction.signatures map. Depending on the type of message, theSigningMethod can take these values:

- SigningMethod.ETH_SIGN
- SigningMethod.ETH_SIGN_TYPED_DATA_V4

```
// Connect the EthAdapter from owner1 protocolKit =

await

protocolKit .connect ({ ethAdapter : ethAdapter1 })

// Sign the safeTransaction with owner1 // After this, the safeTransaction contains the signature from owner1 safeTransaction =

await

protocolKit .signTransaction ( safeTransaction , SigningMethod . ETH_SIGN )

// Connect the EthAdapter from owner2 protocolKit =

await

protocolKit .connect ({ ethAdapter : ethAdapter2 })

// Sign the safeTransaction with owner2 // After this, the safeTransaction contains the signature from owner1 and owner2 safeTransaction =

await

protocolKit .signTransaction ( safeTransaction , SigningMethod . ETH_SIGN_TYPED_DATA_V4 )
```

At this point, thesafeTransaction object should look like this:

```
EthSafeTransaction { signatures :

Map ( 2 ) { '0x90f8bf6a479f320ead074411a4b0e7944ea8c9c1'

=> EthSafeSignature { signer :

'0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1' , data :

'0x969308e2abeda61a0c9c41b3c615012f50dd7456ca76ea39a18e3b975abeb67f275b07810dd59fc928f3f9103e52557c1578c7c5c171ffc983afa5306466b1261f' , isContractSignature :

false } , '0xffcf8fdee72ac11b5c542428b35eef5769c409f0'

=> EthSafeSignature { signer :

'0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0' , data :
```

'0x4d63c79cf9d743782bc31ad58c1a316020b39839ab164caee7ecac9829f685cc44ec0d066a5dfe646b2ffeeb37575df131daf9c96ced41b8c7c4aea8dc5461801c' , isContractSignature :

false } } , data : { ... } }

Thesignatures.data represents a specific signature. TheisContractSignature flag set tofalse indicates that the signature isn't a smart contract signature but an ECDSA signature instead.

An ECDSA signature comprises two 32-byte integers (r ,s) and an extra byte for recovery (v), totaling 65 bytes. In hexadecimal string format, each byte is represented by two characters. Hence, a 65-byte Ethereum signature will be 130 characters long. Including the0x prefix commonly used with signatures, the total character count for such a signature would be 132.

Two more characters are required to represent a byte (8 bits) in hexadecimal. Each hexadecimal character represents four bits. Therefore, two hexadecimal characters (2 x 4 bits) can represent a byte (8 bits).

The final part of the signature, either1f or1c , indicates the signature type.

Safe supports the followingv values:

- 0
- : Contract signature.
- 1
- : Approved hash.
- {27, 28} + 4
- : Ethereum adjusted ECDSA recovery byte for EIP-191 signed message.

Regarding the EIP-191 signed message, thev value is adjusted to the ECDSAv + 4 . If the generated value is28 and adjusted to0x1f , the signature verification will fail as it should be0x20 ('28 + 4 = 32) instead. If v > 30, then the default v(27, 28) was adjusted because of the eth_sign` implementation. This calculation is automatically done by the Safe{Core} SDK.

- Other: Ethereum adjusted ECDSA recovery byte for raw signed hash.

The hexadecimal value1f equals the decimal number31 . If the decimal value is greater than30 , iindicates(opens in a new tab) that the signature is aneth_sign signature.

The hexadecimal value1c equals the decimal number28 , indicating that the signature is a typed data signature.

The initial signature should look like this:

0x969308e2abeda61a0c9c41b3c615012f50dd7456ca76ea39a18e3b975abeb67f275b07810dd59fc928f3f9103e52557c1578c7c5c171ffc983afa5306466b1261f :

Type Description Bytes Value Hex Hex string characters 1 0x Signature Signature bytes 64 969308e2abeda61a0c9c41b3c615012f50dd7456ca76ea39a18e3b975abeb67f275b07810dd59fc928f3f9103e52557c1578c7c5c171ffc983afa5306466b126 Signature Type 1f hex is 31 in decimal 1 1f

Smart contract signatures

When signing with a Safe account, theSigningMethod will take the valueSigningMethod.SAFE_SIGNATURE .

1/1 Safe account

This applies to thesafe1_1 account, another owner ofsafe3_4 .

We need to connect the Protocol Kit tosafe1_1 and theowner3 account (the only owner ofsafe1_1) and sign the transaction.

// Create a new transaction object let transactionSafe1_1 =

await

protocolKit .createTransaction ({ transactions : [safeTransactionData] })

// Connect the adapter from owner3 and the address of safe1_1 protocolKit =

await

protocolKit .connect ({ ethAdapter : ethAdapter3 , safeAddress : safe1_1 })

// Sign the transactionSafe1_1 with owner3 // After this, transactionSafe1_1 contains the signature from owner3 transactionSafe1_1 =

await

protocolKit .signTransaction (transactionSafe1_1 , SigningMethod . SAFE_SIGNATURE , safe3_4 // Parent Safe address)

When signing with a child Safe account, we need to specify the parent Safe address to generate the signature based on the version of the contract.

At this point, thetransactionSafe1_1 object should look like this:

EthSafeTransaction { signatures :

Map (1) { '0x22d491bde2303f2f43325b2108d26f1eaba1e32b'

=> EthSafeSignature { signer :

'0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b' , data :

'0x5edb6ffe67dd935d93d07c634970944ba0b096f767b92018ad635e8b28effeea5a1e512f1ad6f886690e0e30a3fae2c8c61d3f83d24d43276acdb3254b92ea5b1f' , isContractSignature :

false } } , data : { ... } }

Thesignatures.data represents a specific signature. TheisContractSignature flag set tofalse indicates that the signature isn't a smart contract signature but an ECDSA signature instead.

To generate a Safe compatible signature, we use thebuildContractSignature method, which takes an array of signatures and returns another signature that can be used with Safe accounts. After that, we add the signature fromsafe1_1 to our initial transaction.

// Build the contract signature of safe1_1 const

signatureSafe1_1

=

await

buildContractSignature (Array .from (transactionSafe1_1 . signatures .values ()) , safe1_1)

// Add the signatureSafe1_1 to safeTransaction // After this, the safeTransaction contains the signature from owner1, owner2 and safe1_1 safeTransaction .addSignature (signatureSafe1_1)

ThesignatureSafe1_1 object should look like this:

EthSafeSignature { signer :

'0x215033cdE0619D60B7352348F4598316Cc39bC6E' , data :

'0x5edb6ffe67dd935d93d07c634970944ba0b096f767b92018ad635e8b28effeea5a1e512f1ad6f886690e0e30a3fae2c8c61d3f83d24d43276acdb3254b92ea5b1f' , isContractSignature :

true }

TheisContractSignature flag is nowtrue becausesignatureSafe1_1 is an EIP-1271 smart contract signature from thesafe1_1 account.

ThesignatureSafe1_1.data signature should look like this:

0x000000000000000000000000215033cdE0619D60B7352348F4598316Cc39bC6E000000000000000000000000000000000000000000000000000000000000004100000000000000000000000000000

Type Description Bytes Value Hex Hex string characters 1 0x Verifier Padded address of the contract that implements the EIP-1271 interface to verify the signature. The Safe signer address 32

000000000000000000000000215033cdE0619D60B7352348F4598316Cc39bC6E Data position Start position of the signature data (offset relative to the beginning of the signature data). 41 hex is 65 in decimal 32 0000000000000000000000000000000000000000000000000000000000000041 Signature Type 00 for Safe accounts(opens in a new tab) 1 00 Signature Length The length of the signature. 41 hex is 65 in decimal 32 0000000000000000000000000000000000000000000000000000000000000041 Signature Signature bytes that are verified by the signature verifier 65 5edb6ffe67dd935d93d07c634970944ba0b096f767b92018ad635e8b28effeea5a1e512f1ad6f886690e0e30a3fae2c8c61d3f83d24d43276acdb3254b92ea5b1f

2/3 Safe account

This applies to thesafe2_3 account, another owner ofsafe3_4 .

We need to connect the Protocol Kit tosafe2_3 and theowner4 andowner5 accounts (owners ofsafe2_3) and sign the transaction.

// Create a new transaction object let transactionSafe2_3 =

await

protocolKit .createTransaction ({ transactions : [safeTransactionData] })

// Connect the EthAdapter from owner4 and the address of safe2_3 protocolKit =

await

protocolKit .connect ({ ethAdapter : ethAdapter4 , safeAddress : safe2_3 })

// Sign the transactionSafe2_3 with owner4 // After this, the transactionSafe2_3 contains the signature from owner4 transactionSafe2_3 =

await

protocolKit .signTransaction (transactionSafe2_3 , SigningMethod . SAFE_SIGNATURE , safe3_4 // Parent Safe address)

// Connect the adapter for owner5 protocolKit =

await

protocolKit .connect ({ ethAdapter : ethAdapter5 })

// Sign the transactionSafe2_3 with owner5 // After this, the transactionSafe2_3 contains the signature from owner5 transactionSafe2_3 =

await

protocolKit .signTransaction (transactionSafe2_3 , SigningMethod . SAFE_SIGNATURE , safe3_4 // Parent Safe address)

At this point, thetransactionSafe2_3 object should look like this:

EthSafeTransaction { signatures :

Map (2) { '0xe11ba2b4d45eaed5996cd0823791e0c93114882d'

=> EthSafeSignature { signer :

'0xE11BA2b4D45Eaed5996Cd0823791E0C93114882d' , data :

'0xd3e6565e5590641db447277243cf24711dce533cfcaaf3a64415dcb9fa309fbf2de1ae4709c6450752acc0d45e01b67b55379bdf4e3dc32b2d89ad0a60c231d61f' , isContractSignature :

false } , '0xd03ea8624c8c5987235048901fb614fdca89b117'

=> EthSafeSignature { signer :

'0xd03ea8624C8C5987235048901fB614fDcA89b117' , data :

'0x023d1746ed548e90f387a6b8ddba26e6b80a78d5bfbc36e5bfcbfd63e136f8071db6e91c037fa36bde72159138bbb74fc359b35eb515e276a7c0547d5eaa042520' , isContractSignature :

false } } , data : { ... } }

We now have two signatures from the owners,owner4 andowner5 . Following the same process, we can create the contract signature and examine the result.

Thesignatures.data represents a specific signature. TheisContractSignature flag set tofalse indicates that the signature isn't a smart contract signature but an ECDSA signature instead.

To generate a Safe compatible signature, we use thebuildContractSignature method, which takes an array of signatures and returns another signature that can be used with Safe accounts. After that, we add the signature fromsafe1_1 to our initial transaction.

// Build the contract signature of safe2_3 const

signatureSafe2_3

=

await

buildContractSignature (Array .from (transactionSafe2_3 . signatures .values ()) , safe2_3)

// Add the signatureSafe2_3 to safeTransaction // After this, the safeTransaction contains the signature from owner1, owner2, safe1_1 and safe2_3 safeTransaction .addSignature (signatureSafe2_3)

ThesignatureSafe2_3 object should look like this:

0x000000000000000000000000f75D61D6C27a7CC5788E633c1FC130f0F4a62D330000000000000000000000000000000000000000000000000000000000000041000000000000000000000000000000

Type Description Bytes Value Hex Hex string characters 1 0x Verifier Padded address of the contract that implements the EIP-1271 interface to verify the signature. The Safe signer address 32 000000000000000000000000f75D61D6C27a7CC5788E633c1FC130f0F4a62D33 Data position Start position of the signature data (offset relative to the beginning of the signature data). 41 hex is 65 in decimal 32 0000000000000000000000000000000000000000000000000000000000000041 Signature Type 00 for Safe accounts(opens in a new tab) 1 00 Signature Length The length of the signature. 82 hex is 130 in decimal 32 0000000000000000000000000000000000000000000000000000000000000082 Signature Signature bytes that are verified by the signature verifier (130 bytes are represented by 260 characters in an hex string) 130 023d1746ed548e90f387a6b8ddba26e6b80a78d5bfbc36e5bfcbfd63e136f8071db6e91c037fa36bde72159138bbb74fc359b35eb515e276a7c0547d5eaa042520d3e6565e5590641db447277243cf24711dc

The table looks very similar to the previous one, but there are two main differences:

- TheSignature Length
- value has doubled becausesafe2_3
- needs two signatures.
- TheSignature
- value is a concatenation of the two regular signatures.

After following all the steps above, thesafeTransaction now contains all the signatures from the owners of the Safe.

ThesafeTransaction object should look like this:

EthSafeTransaction { signatures :

Map (4) { '0x90f8bf6a479f320ead074411a4b0e7944ea8c9c1'

=> EthSafeSignature { signer :

'0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1' , data :

'0x969308e2abeda61a0c9c41b3c615012f50dd7456ca76ea39a18e3b975abeb67f275b07810dd59fc928f3f9103e52557c1578c7c5c171ffc983afa5306466b1261f' , isContractSignature :

false } , '0xffcf8fdee72ac11b5c542428b35eef5769c409f0'

=> EthSafeSignature { signer :

```
'0xFFcf8FDEE72ac11b5c542428B35EEF5769C409f0' , data :
'0x4d63c79cf9d743782bc31ad58c1a316020b39839ab164caee7ecac9829f685cc44ec0d066a5dfe646b2ffeeb37575df131daf9c96ced41b8c7c4aea8dc5461801c' , isContractSignature :
false } , '0x215033cde0619d60b7352348f4598316cc39bc6e'
=> EthSafeSignature { signer :
'0x215033cdE0619D60B7352348F4598316Cc39bC6E' , data :
'0x5edb6ffe67dd935d93d07c634970944ba0b096f767b92018ad635e8b28effeea5a1e512f1ad6f886690e0e30a3fae2c8c61d3f83d24d43276acdb3254b92ea5b1f' , isContractSignature :
true } , '0xf75d61d6c27a7cc5788e633c1fc130f0f4a62d33'
=> EthSafeSignature { signer :
'0xf75D61D6C27a7CC5788E633c1FC130f0F4a62D33' , data :
'0x023d1746ed548e90f387a6b8ddba26e6b80a78d5bfbc36e5bfcbfd63e136f8071db6e91c037fa36bde72159138bbb74fc359b35eb515e276a7c0547d5eaa042520d3e6565e5590641db447277243cf24711
, isContractSignature :
true } } , data : { to :
'0x90F8bf6A479f320ead074411a4B0e7944Ea8c9C1' , value :
'100000000000000000' , data :
'0x' , operation :
0 , baseGas :
'0' , gasPrice :
'0' , gasToken :
'0x0000000000000000000000000000000000000000' , refundReceiver :
'0x0000000000000000000000000000000000000000' , nonce :
0 , safeTxGas :
'0' } }
```

Propose the transaction

To store the transactions and signatures off-chain, we need to call the Safe Transaction Service API - a centralized and open-source service that anyone can deploy and run.

The Safe Transaction Service is used bySafe{Wallet}(opens in a new tab) to store transactions and signatures by default.

To store a new transaction, we need to call theproposeTransaction from the API Kit, passing the Safe address, an object with the transaction, and a signature from one owner.

```
const
signerAddress
= ( await
ethAdapter1 .getSignerAddress ()) ||
'0x'
// Get the signature from owner1 const
signatureOwner1
=
safeTransaction .getSignature (signerAddress) as
EthSafeSignature
// Get the transaction hash of the safeTransaction const
safeTransactionHash
=
await
protocolKit .getTransactionHash (safeTransaction)
// Instantiate the API Kit // Use the chainId where you have the Safe account deployed const
apiKit
=
new
SafeApiKit ({ chainId })
// Propose the transaction await
apiKit .proposeTransaction ({ safeAddress : safe3_4 , safeTransactionData :
safeTransaction .data , safeTxHash : safeTransactionHash , senderAddress : signerAddress , senderSignature :
buildSignatureBytes ([signatureOwner1]) })
```

The transaction is now publicly available in the Safe Transaction Service with the signature of the owner who submitted it.

Confirm the transaction

To add the signatures from the remaining owners, we need to call theconfirmTransaction , passing thesafeMessageHash and a signature from the owner.

Once a transaction is proposed, it becomes available onSafe{Wallet}(opens in a new tab) . However, to execute the transaction, all the confirmations from the owners are needed.

```
const
signerAddress
= ( await
ethAdapter2 .getSignerAddress ()) ||
'0x'
```

const

signatureOwner2

=

safeTransaction .getSignature (signerAddress) as

EthSafeSignature

// Confirm the transaction from owner2 await

apiKit .confirmTransaction (safeTransactionHash , buildSignatureBytes ([signatureOwner2]))

// Confirm the transaction with the owner safe1_1 await

apiKit .confirmTransaction (safeTransactionHash , buildSignatureBytes ([signatureSafe1_1]))

// Add signature from the owner safe2_3 await

apiKit .confirmTransaction (safeTransactionHash , buildSignatureBytes ([signerSafeSig2_3]))

At this point, the transaction stored in the Safe Transaction Service contains all the required signatures from the owners of the Safe.

ThegetTransaction method returns the transaction with theconfirmations property to check all the added signatures.

// Get the transactions const

signedTransaction

=

await

apiKit .getTransaction (safeTransactionHash)

// Get the confirmations const

confirmations

=

signedTransaction .confirmations

Safe{Wallet}(opens in a new tab) exposes to its users the list of pending transactions.

https://app.safe.global/transactions/queue?safe=:

Execute the transaction

Connect the Safe and the adapter of an owner to the Protocol Kit. Ensure enough funds are available in the owner's account to execute the transaction and cover the gas costs. Once the Protocol Kit is initialized, theexecuteTransaction method receives and executes the transaction with the required signatures.

protocolKit

await

protocolKit .connect ({ ethAdapter : ethAdapter1 , safeAddress : safe3_4 })

// Execute the Safe transaction const

transactionResponse

=

await

protocolKit .executeTransaction (safeTransaction)

At this point, the Safe transaction should be executed on-chain and listed onSafe{Wallet}(opens in a new tab) .

https://app.safe.global/transactions/history?safe=:

ThesafeTransaction.encodedSignature method returns the signatures concatenated and sorted by the address of the signers. It should look like this:

0x000000000000000000000000215033cdE0619D60B7352348F4598316Cc39bC6E000000000000000000000000000000000000000000000000000000000000010400969308e2abeda61a0c9c41b3c61

Type Description Bytes Acc byte Value Hex Hex string characters 1 - 0x 1/1 Safe signer Safe Address 32 32 000000000000000000000000215033cdE0619D60B7352348F4598316Cc39bC6E Data position for 1/1 Safe 104 hex = Signature data for 1/1 Safe start at byte 260 32 64 0000000000000000000000000000000000000000000000000000000000000104 Signature Type Smart contract signature 1 65 00 Owner signature owner1 signature 65 130 969308e2abeda61a0c9c41b3c615012f50dd7456ca76ea39a18e3b975abeb67f275b07810dd59fc928f3f9103e52557c1578c7c5c171ffc983afa5306466b1261f 2/3 Safe signer Safe Address 32 162 000000000000000000000000f75D61D6C27a7CC5788E633c1FC130f0F4a62D33 Data position for 2/3 Verifier 165 hex = Signature data for 2/3 Safe start at byte 357 32 194 0000000000000000000000000000000000000000000000000000000000000165 Signature Type Smart contract signature 1 195 00 Owner signature owner2 signature 65 260 4d63c79cf9d743782bc31ad58c1a316020b39839ab164caee7ecac9829f685cc44ec0d066a5dfe646b2ffeeb37575df131daf9c96ced41b8c7c4aea8dc5461801c 1/1 Safe Signature Length Start of the 1/1 Safe Signature. 41 hex = 65 bytes 32 292 0000000000000000000000000000000000000000000000000000000000000041 Signature owner3 signature 65 357 5edb6ffe67dd935d93d07c634970944ba0b096f767b92018ad635e8b28effeea5a1e512f1ad6f886690e0e30a3fae2c8c61d3f83d24d43276acdb3254b92ea5b1f 2/3 Safe Signature length Start of the 2/3 Safe Signature. 82 hex = 130 bytes 32 389 0000000000000000000000000000000000000000000000000000000000000082 Signature owner4 andowner5 concatenated signatures 130 519 023d1746ed548e90f387a6b8ddba26e6b80a78d5bfbc36e5bfcbfd63e136f8071db6e91c037fa36bde72159138bbb74fc359b35eb515e276a7c0547d5eaa042520d3e6565e5590641db447277243cf24711dc

Signatures Messages

Was this page helpful?

Report issue