sponsoredCall

Permissionless sponsored transactions After reading this page:

- You'll know how to use theSponsoredCall
- SDK method. This uses the1Balance
- payment method, allowing you to sponsor some/all of your user's gas costs.
- You'll see some code which will help you send a relay request within minutes. * Please proceed to ouSecurity Considerations page and read it thoroughly before advancing with your implementation. It is crucial to understand all potential security risks and measures to mitigate them. The nature ofsponsoredCall is permissionless and doesnot enforce any security. Target contracts shouldnot whitelist the calling contract as an authorizedmsg.sender . The contract address is subject to change without further notice.

Overview

sponsoredCall method utilises authentication via a sponsor API key to sponsorgasless transactions for your users securely. The payment method is Gelato1Balance .

SDK method: sponsoredCall

```

Copy constsponsoredCall=async( request:SponsoredCallRequest, sponsorApiKey:string, options?:RelayRequestOptions ):Promise

```

Arguments:

- request
- : this is therequest body
- used to send a request.
- sponsorApiKey
- : an API key used to authenticate your sponsorship.
- options
- :RelayRequestOptions
- is an optional request object.
-

Return Object: RelayResponse

```

Copy typeRelayResponse={ taskId:string; };

```

- taskId
- : your unique relay task ID which can be used fortracking your request
- .
-

Optional Parameters

SeeOptional Parameters .

Using Safe smart contract wallets for sponsoredCall

If you need to dispatch transactions from Safe smart contract wallets using Gelato Relay viasponsoredCall , you can opt to activate Safe-enabled transactions in your Relay Dapp configuration.

?

Usually when submitting Gelato Relay transactions that originate from Safe smart contract wallets, the Safe wallet address is specified in thetarget field of the Relay API request, whereas both the actual target contract address and itscalldata are encoded into theexecTransaction payload. By activating Safe-enabled transactions - accomplished by checking the "Allow sponsored transactions from Safes" box - Gelato Relay will validate your Safe smart contract and decode the target contract address and function selector from theexecTransaction calldata . It will then apply your pre-configured Relay Dapp rules to these values, rather than to the values given in the Relay request.

If your intention is to deploy Safe smart contract wallets prior to their usage, be sure to whitelist themulticall contract address in your Relay Dapp. This contract is typically invoked when you deploy a Safe smart contract wallet before it can be used for the first time.

Sending a Request

Request Body

```

Copy constrequest={ chainId:BigNumberish; target: string; data: BytesLike; };

```

- chainId
- : the chain ID of the chain where thetarget
- smart contract is deployed.
- target
- : the address of the target smart contract.
- data
- : encoded payload data (usually a function selector plus the required arguments) used to call the requiredtarget
- address.
-

Example Code

SincesponsoredCall assumes you have your own security logic built in (i.e. replay and re-entrancy protection), you can go ahead and generate the payload for your function call and populate a request object.

1. Import GelatoRelaySDK into your front-end .js project

```

Copy import{ GelatoRelay,SponsoredCallRequest }from"@gelatonetwork/relay-sdk"; constrelay=newGelatoRelay();

```

1. Deploy a smart contract

This is an example using Gelato'sSimpleCounter.sol which is deployed onGoerli andPolygon .

```

Copy pragmasolidity0.8.17;

contractSimpleCounter{ uint256publiccounter;

eventIncrementCounter(uint256newCounterValue,addressmsgSender);

// increment is the target function to call. // This function increments a counter variable by 1 // IMPORTANT: withsponsoredCall you need to implement // your own smart contract security measures, as this // function can be called by any third party and not only by // Gelato Relay. If not done properly, funds kept in this // smart contract can be stolen. functionincrement()external{ counter++; emitIncrementCounter(counter,msg.sender); } }

```

1. Generate a payload for your target contract

```

Copy // set up target address and function signature abi const counter="0x763D37aB388C5cdd2Fb0849d6275802F959fbF30"; const abi=["function increment()"];

// generate payload using front-end provider such as MetaMask const provider=newethers.BrowserProvider(window.ethereum); const signer=provider.getSigner();

constcontract=newethers.Contract(counterAddress,abi,signer); const { data }=awaitcontract.increment.populateTransaction();
```

```

1. Send the payload to Gelato

```

Copy // Populate a relay request constrequest:SponsoredCallRequest={ chainId:(awaitprovider.getNetwork()).chainId, target:counter, data:data, };

// Without a specific API key, the relay request will fail! // Go to https://relay.gelato.network to get a testnet API key with 1Balance. // Send the relay request using Gelato Relay! constrelayResponse=awaitrelay.sponsoredCall(request,apiKey);

```

Previous Non-ERC-2771 Next callWithSyncFee Last updated2 days ago On this page *Overview * Using Safe smart contract wallets for sponsoredCall * Sending a Request * Request Body * Example Code * 1. Import GelatoRelaySDK into your front-end .js project * 2. Deploy a smart contract * 3. Generate a payload for your target contract * 4. Send the payload to Gelato
```