

Interact With a Smart Contract

Use Circle APIs to interact with a smart contract you've already deployed [Suggest Edits](#)

In this guide, we will interact with a deployed smart contract using APIs. If you haven't deployed a smart contract, follow the [deploy a smart contract](#) tutorial.

1. Retrieve the Smart Contract's Functions

To fetch the new Contract object, make a GET /contracts/{id} request that includes the contractId object returned from your deployment request. The functions property returned in the successful request contains a list of all of the functions on the contract:

```
Node.js cURL // Import and configure the developer-controlled wallet SDK const { initiateDeveloperControlledWalletsClient } = require('@circle-fin/developer-controlled-wallets'); const circleDeveloperSdk = initiateDeveloperControlledWalletsClient({ apiKey: "", entitySecret: "" });
```

```
// Import and configure the smart contract SDK const { initiateSmartContractPlatformClient } = require('@circle-fin/smart-contract-platform'); const circleContractSdk = initiateSmartContractPlatformClient({ apiKey: "", entitySecret: "" });
```

```
const response = await circleContractSdk.getContract({ id: '0189db84-72b7-7fcc-832b-5bf886b9a0ef' }); curl --request GET \
--url 'https://api.circle.com/v1/w3s/contracts/{id}' --header 'accept: application/json' --header 'authorization: Bearer ' JSON {
  "data": { "contract": { "id": "0189db84-72b7-7fcc-832b-5bf886b9a0ef", "entityId": "dcfa8149-98d8-4ed4-91fb-de7f3627b384",
  "deploymentTransactionId": "4f5bfa38-c598-56a6-932e-8b5bbd3d5fc9", "name": "First Contract Name", "description": "First
  Contract Description", "contractInputType": "BYTECODE", "createDate": "2023-08-09T18:17:17Z", "updateDate": "2023-08-
  09T18:17:17Z", "archived": false, "contractAddress": "0x1e124d7384cd34448ea5907bd0052a79355ab5eb", "blockchain":
  "MATIC-MUMBAI", "status": "COMPLETE", "deployerAddress": "0x1bf9ad0cc2ad298c69a2995aa806ee832788218c",
  "txHash": "0x241c4df6f08f9ed2b569c9f9b1cc48fb6074ffffae57552e716ce059161a743", "abiJson": "[{"name": "inputs",
  "stateMutability": "nonpayable", "type": "constructor", "anonymous": false, "inputs": [{"name": "owner", "type":
  "address", "indexed": true, "internalType": "address", "name": "owner", "type": "address", "indexed": true,
  "internalType": "address", "name": "operator", "type": "address", "indexed": true, "internalType": "address",
  "name": "tokenId", "type": "uint256", "name": "Approval", "type": "event", "anonymous": false, "inputs": [{"name":
  "inputs", "type": "address", "indexed": true, "internalType": "address", "name": "owner", "type": "address",
  "indexed": true, "internalType": "address", "name": "operator", "type": "address", "indexed": false, "internalType":
  "bool", "name": "approved", "type": "bool", "name": "ApprovalForAll", "type": "event", "anonymous": false, "inputs":
  [{"name": "inputs", "type": "address", "indexed": true, "internalType": "address", "name": "previousOwner", "type":
  "address", "indexed": true, "internalType": "address", "name": "newOwner", "type": "address", "name": "OwnershipTransferred", "type":
  "event", "anonymous": false, "inputs": [{"name": "inputs", "type": "address", "indexed": true, "internalType":
  "address", "name": "from", "type": "address", "indexed": true, "internalType": "address", "name": "to", "type":
  "address", "indexed": true, "internalType": "uint256", "name": "tokenId", "type": "uint256", "name": "Transfer", "type":
  "event", "inputs": [{"name": "inputs", "type": "address", "indexed": true, "internalType": "address", "name": "to",
  "type": "address", "indexed": true, "internalType": "uint256", "name": "tokenId", "type": "uint256", "name":
  "approve", "type": "function", "inputs": [{"name": "inputs", "type": "address", "indexed": true, "internalType":
  "address", "name": "balanceOf", "type": "function", "inputs": [{"name": "inputs", "type": "uint256", "name":
  "view", "type": "function", "inputs": [{"name": "inputs", "type": "uint256", "name": "tokenId", "type": "uint256",
  "name": "getApproved", "type": "function", "inputs": [{"name": "inputs", "type": "address", "indexed": true,
  "internalType": "address", "name": "owner", "type": "address", "indexed": true, "internalType": "address",
  "name": "operator", "type": "address", "name": "isApprovedForAll", "type": "function", "inputs": [{"name":
  "inputs", "type": "bool", "name": "view", "type": "function", "inputs": [{"name": "inputs", "type": "string",
  "name": "view", "type": "function", "inputs": [{"name": "inputs", "type": "address", "indexed": true,
  "internalType": "address", "name": "owner", "type": "address", "indexed": true, "internalType": "address",
  "name": "ownerOf", "type": "function", "inputs": [{"name": "inputs", "type": "address", "indexed": true,
  "internalType": "address", "name": "view", "type": "function", "inputs": [{"name": "inputs", "type": "uint256",
  "name": "tokenId", "type": "uint256", "name": "ownerOf", "type": "function", "inputs": [{"name": "inputs",
  "type": "address", "indexed": true, "internalType": "address", "name": "to", "type": "address", "indexed": true,
  "internalType": "uint256", "name": "tokenId", "type": "uint256", "name": "renounceOwnership", "type": "function",
  "inputs": [{"name": "inputs", "type": "address", "indexed": true, "internalType": "address", "name": "to", "type":
  "address", "indexed": true, "internalType": "uint256", "name": "tokenId", "type": "uint256"}]}
```


10. APIs via the
11. abiParameters
12. property.
13. outputs
14. : The output values when the function returns. Some functions do not return anything.
15. stateMutability
16. : Defines how the function interacts with the blockchain.1. view
17.
 1. and
18.
 1. pure
19.
 1. functions do not change the state of the blockchain, so these are
20.
 1. read
21.
 1. functions.
22.
 1. nonpayable
23.
 1. and
24.
 1. payable
25.
 1. functions update the state of the blockchain, so these are
26.
 1. execute
27.
 1. functions.
28.
 1. nonpayable
29.
 1. is the default for a function if no
30.
 1. stateMutability
31.
 1. property exists.

2. Contract Read Function

Once you have selected a read function from the list, you can send an API request to `POST /contracts/{id}/read` . The example below is making a request to the `owner()` function to check the current owner of the contract

- Note:
- The
- `abiFunctionSignature`
- is derived from the function name and the parenthesized list of parameter types. For instance, the
- `abiFunctionSignature`
- for a function named
- `testFunction`
- that takes an
- `address`
- and an
- `uint256`
- would be
- `testFunction(address,uint256)`
- .
- Note:
- Read interactions do not
- require gas on the blockchain, so you are not required to specify a wallet.

```
Node.js cURL // Read Contract isn't on the SDK yet
const response = await circleContractSdk.readContract({
  abiFunctionSignature: 'owner()' });
curl --request POST \
  --url 'https://api.circle.com/v1/w3s/contracts/{id}/read' \
  --header 'accept: application/json' \
  --header 'content-type: application/json' \
  --header 'authorization: Bearer ' \
  --data '{
    "abiFunctionSignature": "owner()"
  }'
Response Body {
  "data": {
    "outputValues": [
      "0x1bf9ad0cc2ad298c69a2995aa806ee832788218c"
    ]
  }
}
```

3. Contract Execution Function

Once you have chosen an execute function you can make a request to POST developer/transactions/contractExecution or user/transactions/contractExecution , for developer-controlled and user-controlled wallets, respectively. The example below uses the developer-controlled endpoint to call the safeMint() function to mint an NFT. The function takes in two parameters: the to address and the assigned tokenId for the NFT. The abiFunctionSignature is safeMint(address, uint256) .

- Note:
- The
- safeMint
- function must be called by the
- owner
- of the contract. We specified this function modifier when generating the contract earlier.
- Note:
- Write functionsdo
- require gas on the blockchain, so a
- walletId
- must be specified in the request.

```
Node.js cURL const response = await circleDeveloperSdk.createContractExecutionTransaction({ walletId: 'ce714f5b-0d8e-4062-9454-61aa1154869b', contractAddress: '0x2f3A40A3db8a7e3D09B0adfEfbCe4f6F81927557', abiFunctionSignature: 'safeMint(address, uint256)', abiParameters: ['0x6E5eAf34c73D1CD0be4e24f923b97CF38e10d1f3', 1], fee: { type: 'level', config: { feeLevel: 'MEDIUM' } } }); curl --request POST \ --url 'https://api.circle.com/v1/w3s/developer/transactions/contractExecution' \ --header 'accept: application/json' \ --header 'content-type: application/json' \ --header 'authorization: Bearer ' \ --data ' { "idempotencyKey": "d6d56c67-9d8b-4e87-a618-1c44fa04a495", "walletId": "ce714f5b-0d8e-4062-9454-61aa1154869b", "contractAddress": "0x2f3A40A3db8a7e3D09B0adfEfbCe4f6F81927557", "abiFunctionSignature": "safeMint(address, uint256)", "abiParameters": [ "0x6E5eAf34c73D1CD0be4e24f923b97CF38e10d1f3", 1 ], "feeLevel": "MEDIUM", "entitySecretCiphertext": "N8sYP20DJvujfnISM4qirGXGN8Pr+FjgMWN7zHc96EiSJnPxoGviniZINaLT..." } Response Body { "data": { "transactionId": "84cb1754-f236-4bfc-8562-70fa6ee168c8" } } Updated about 2 months ago * Table of Contents * * 1. Retrieve the Smart Contract's Functions * * 2. Contract Read Function * * 3. Contract Execution Function
```