DSProxy

```

Copy constservice=maker.service('proxy');

```

Summary

TheDSProxyService includes all the functionality necessary for interacting with both types of proxy contracts used in Maker products:profile proxies andforwarding proxies .

Forwarding proxies are simple contracts that aggregate function calls in the body of a single method. These are used in theCDP Portal andOasis Direct in order to allow users to execute multiple transactions atomically, which is both safer and more user-friendly than implementing several steps as discrete transactions.

```

Copy // Forwarding proxy

function lockAndDraw(address tub_, bytes32 cup, uint wad) public payable { lock(tub_, cup); draw(tub_, cup, wad); }

```

Forwarding proxies are meant to be as simple as possible, so they lack some features that could be important if they are to be used as interfaces for more complex smart contract logic. This problem can be solved by using profile proxies (i.e. copies ofDSProxy) to execute the functionality defined in the forwarding proxies.

The first time an account is used to interact with any Maker application, the user will be prompted to deploy a profile proxy. This copy of DSProxy can be used in any product, including dai.js, by way of a universalproxy registry . Then, the calldata from any function in the forwarding proxy can be passed to DSProxy'sexecute() method, which runs the provided code in the context of the profile proxy.

```

Copy // Calling the forwarding proxy with dai.js

functionlockAndDraw(tubContractAddress,cdpId,daiAmount,ethAmount) {
constsaiProxy=maker.service('smartContract').getContractByName('SAI_PROXY');

returnsaiProxy.lockAndDraw( tubContractAddress, cdpId, daiAmount, { value:ethAmount, dsProxy:true } ); }

```

This makes it possible for users' token allowances to persist from one Maker application to another, and it allows users torecover any funds mistakenly sent to the proxy's address. Many of the functions inDSProxyService will only be relevant topower users . All that is strictly required to automatically generate a function's calldata and find the correct profile proxy is the inclusion of{ dsProxy: true } in the options object for any transaction — provided the user has already deployed a profile proxy. If that's not certain, it may also be necessary to query the registry to determine if a user already owns a proxy, and tobuild one if they do not.

currentProxy()

- Params:
- None
- Returns:
- promise (resolves to addressor
- null
-)
-

If thecurrentAccount (according theWeb3Service) has already deployed a DSProxy,currentProxy() returns its address. If not, it returnsnull . It will update automatically in the event that the active account is changed. This function should be used to check whether a user has a proxy before attempting to build one.

```

Copy asyncfunctiongetProxy() { returnmaker.service('proxy').currentProxy(); }

```

build()

- Params:
- None
- Returns:
- TransactionObject
-

build will deploy a copy of DSProxy owned by the current account.This transaction will revert if the current account already owns a profile proxy. By default,build() returns after the transaction is mined.

```
Copy asyncfunctionbuildProxy() { constproxyService=maker.service('proxy'); if(!proxyService.currentProxy()) { returnproxyService.build(); } }
```

ensureProxy()

This convenience function will either return an existing proxy or create one.

```
Copy constproxyAddress=awaitmaker.service('proxy').ensureProxy();
```

getProxyAddress()

- Params:
- Address (optional)
- Returns:
- promise (resolves to contract address)
-

getProxyAddress will query the proxy registry for the profile proxy address associated with a given account. If no address is provided as a parameter, the function will return the address of the proxy owned by thecurrentAccount .

```
Copy constproxy=awaitmaker.service('proxy').getProxyAddress('0x...');
```

getOwner()

- Params:
- Address
- Returns:
- promise (resolves to address)
-

getOwner will query the proxy registry for the owner of a provided instance of DSProxy.

```
Copy constowner=awaitmaker.service('proxy').getOwner('0x...');
```

setOwner()

- Params:
- Address of new owner, DSProxy address (optional)
- Returns:
- TransactionObject
-

setOwner can be used to give a profile proxy to a new owner. The address of the recipient account must be specified, but the DSProxy address will default tocurrentProxy if the second parameter is excluded.

```

Copy awaitmaker.service('proxy').setOwner(newOwner,proxyAddress);

```

Previous Transaction manager Next Events Last updated4 years ago On this page *Summary * currentProxy() * build() * ensureProxy() * getProxyAddress() * getOwner() * setOwner()

Export as PDF