Multi contract execution

Withdedicated msg.sender enabled, you can make use of its multi-call functionbatchExecuteCall to execute multiple functions on different contracts.

```
Copy functionbatchExecuteCall( address[]calldata_targets, bytes[]calldata_datas, uint256[]calldata_values )externalpayableoverrideonlyAuth{ uint256length=_targets.length; require( length==_datas.length&&length==_values.length, "OpsProxy: Length mismatch" );

for(uint256i; i<length; i++) _executeCall(_targets[i],_datas[i],_values[i]); }
```

To do so, you will need to create a task with your dedicatedmsg.sender as the target contract address.

Get your dedicated msg.sender

Copy the address of your dedicatedmsg.sender which can be found in the user drop-down.

?

Create a task calling your dedicated msg.sender

?

Paste the ABI below into the ABI field and selectbatchExecuteCall as the function to be automated.

```
Copy [{"inputs":[{"internalType":"address[]","name":"_targets","type":"address[]"},
{"internalType":"bytes[]","name":"_datas","type":"bytes[]"},
{"internalType":"uint256[]","name":"_values","type":"uint256[]"}],"name":"batchExecuteCall","outputs":
[],"stateMutability":"payable","type":"function"}]
```

?

You can either predefine the arguments or have a resolver return the data.

If you are using a resolver, you must return an ABI encoded payload of thebatchExecuteCall function.

Here is an example resolver that returns a payload that callsincreaseCount on 2 different contracts.

```
Copy interfaceIOpsProxy{ functionbatchExecuteCall( address[]calldatatargets, bytes[]calldatadatas, uint256[]calldatavalues )externalpayable; }

interfaceICounter{ functionincreaseCount(uint256_amount)external; }

contractBatchExecCallResolver{

addresspublicimmutablecounter1; addresspublicimmutablecounter2;

constructor(address_counter1,address_counter2){ counter1=_counter1; counter2=_counter2; }

functionchecker() external view returns(boolcanExec,bytesmemoryexecPayload) { address[]memorytargets=newaddress; targets[0]=counter1; targets[1]=counter2;

bytes[]memorydatas=newbytes; datas[0]=abi.encodeWithSelector(ICounter.increaseCount.selector,[1]); datas[1]=abi.encodeWithSelector(ICounter.increaseCount.selector,[2]);

uint256[]memoryvalues=newaddress; values[0]=0; values[1]=0;

execPayload=abi.encodeWithSelector( IOpsProxy.batchExecuteCall.selector, [targets,datas,values] );

return(true,execPayload); } }
```