Private Typescript Functions

Deployed Typescript Functions are public

When you deploy a Typescript Function the code is stored and pinned on IPFS making it accessible to everyone. If you would prefer to conceal your code, one approach is to store your code in a private Github Gist. Subsequently, this code can be retrieved and executed through a Web3 Function.

Note: this approach introduces a dependency on Github's availability. We aim to directly support private Web3 Function deployments in the future.

Private Typescript Function example

This Typescript Function fetchesonRun.js (Github gist containing concealed code) with its gist id and executes it during runtime. Check out the example on GitHubhere .

The code inonRun.js must be in JavaScript ```

private-w3f/index.ts Copy import{ Web3Function, Web3FunctionContext, Web3FunctionResult, }from"@gelatonetwork/web3-functions-sdk"; import{ Octokit }from"octokit";

// import dependencies used in onRun.js import{ ethers }from"ethers"; importkyfrom"ky";

Web3Function.onRun(async(context:Web3FunctionContext)=>{ const{secrets}=context;

constgistId=(awaitsecrets.get("GIST_ID"))asstring;

constoctokit=newOctokit();

letonRunScript:string|undefined;

// fetch onRun.js from private github gist try{ constgistDetails=awaitoctokit.rest.gists.get({ gist_id:gistId, });

constfiles=gistDetails.data.files;

if(!files)thrownewError(No files in gist);

for(constfileofObject.values(files)) { if(file?.filename==="onRun.js"&&file.content) { onRunScript=file.content; break; } }

if(!onRunScript)thrownewError(No onRun.js); }catch(err) { return{ canExec:false, message:Error fetching gist:{err.message}, }; }

// run onRun.js try{ /* * context are passed into onRun.js. * onRun.js will have access to all userArgs, secrets & storage*/
constonRunFunction=newFunction("context","ky","ethers",onRunScript);
constonRunResult:Web3FunctionResult=awaitonRunFunction(context, ky, ethers);

if(onRunResult) { returnonRunResult; }else{ return{ canExec:false,message:No result returned}; } }catch(err) { console.log(err); return{ canExec:false, message:Error running gist:{err.message}, }; } });

```

WritingonRun.js

Check out an example of a GitHub gist withonRun.jshere .

1.onRun.js file structure

onRun.js should return a promise.

```

onRun.js Copy return(async()=>{ // ... your code here })();

```

1. Using dependencies

Dependencies that are used inonRun.js should be imported into the Web3 Functionindex.ts file, not inonRun.js .

```

private-w3f/index.ts Copy // import dependencies used in onRun.js import{ ethers }from"ethers"; importkyfrom"ky";

```

1. Accessing Web3 Function Context

Web3 Function context which includes,secrets ,userArgs ,multiChainProvider can be accessed normally inonRun.js .

```

onRun.js Copy return(async()=>{ const{secrets,userArgs,multiChainProvider}=context })();

```

1. Return Web3 Function result

Results returned inonRun.js will be bubbled up and returned in the private Web3 Function.

```

onRun.js Copy return{ canExec:true, callData:[{ to:oracleAddress, data:oracle.interface.encodeFunctionData("updatePrice", [price]), },], }j

```

Creating private Typescript Function task

Secrets (strict)

- GIST_ID
- (Github gist id to fetchonRun.js
- from)
-

Make sure to store your GitHub gist id as a secret.

Arguments (not strict)

Since GitHub gists are editable, you can have a userArgs to be a JSON string so that arguments can be editable without re-deploying a web3 function with a different schema.

```

private-w3f/schema.json Copy { "web3FunctionVersion":"2.0.0", "runtime":"js-1.0", "memory":128, "timeout":30, "userArgs":{ "args":"string" } }

```

Exampleargs when creating your task:

```

Copy { "args":"{\"currency\":\"ethereum\",\"oracle\":\"0x71B9B0F6C999CBbB0FeF9c92B80D54e4973214da\"}"

```

Previous Event Trigger Next Callbacks Last updated4 months ago On this page *Deployed Typescript Functions are public * Private Typescript Function example * Writing onRun.js * 1. onRun.js file structure * 2. Using dependencies * 3. Accessing Web3 Function Context * 4. Return Web3 Function result * Creating private Typescript Function task * Secrets (strict) * Arguments (not strict)
```