Smart Contract

Using Automate programatically Tasks created via this route cannot be named Smart Contracts can also create and cancel tasks on Gelato Automate.

You can find a list of example smart contracts here .

AutomateTaskCreator functions

Here are the functions exposed by AutomateTaskCreator which you can use when setting up your smart contract.

_createTask()

Interacts and creates a task on the Gelato Automate smart contract.

```

Copy function _createTask( address execAddress, bytes memory execDataOrSelector, ModuleData memory moduleData, address feeToken )internal returns(bytes32 taskId);

```

- execAddress
  - Address of the contract which Gelato will call.
- execDataOrSelector
  - Signature of function which Gelato will call / execution data (If Resolver Module is not used. More about modules below)
- moduleData
  - Modules that are enabled for the task. (More about ModuleData below)
- feeToken
  - Useaddress(0)
- if using Gelato balance. Use 0xeeeeee... for ETH or native tokens.

ModuleData

```

Copy struct ModuleData{ Module[] modules; bytes[] args; }

```

Modules are conditions / specifications about your task. These are the current available Modules.

```

Copy enum Module{ RESOLVER, TIME, PROXY, SINGLE_EXEC }

```

- RESOLVER
  - Define dynamic conditions and execution data. Smart Contract Resolvers
- TIME
  - Repeated execution at a specific time and interval.
- PROXY
  - Your function will be called by a dedicated msg.sender
- . Dedicated msg.sender
- SINGLE_EXEC
  - Task is cancelled after one execution.

Each Module would require additional arguments which is an encoded data.

You can use these helper functions to get the arguments for each Module.

```

Copy function_resolverModuleArg(address_resolverAddress,bytesmemory_resolverData)

function_timeModuleArg(uint256_startTime,uint256_interval)

function_proxyModuleArg()

function_singleExecModuleArg()

```

CraftingModuleData will look like this if we want to create a task which utiliseRESOLVER ,PROXY &SINGLE_EXEC Module.

```

Copy ModuleDatamemorymoduleData=ModuleData({ modules:newModule, args:newbytes });

moduleData.modules[0]=Module.RESOLVER; moduleData.modules[1]=Module.PROXY;
moduleData.modules[2]=Module.SINGLE_EXEC

moduleData.args[0]=_resolverModuleArg( address(this), abi.encodeCall(this.checker,()) );
moduleData.args[1]=_proxyModuleArg(); moduleData.args[2]=_singleExecModuleArg();

```

Module[] must follow the orderRESOLVER ,TIME ,PROXY ,SINGLE_EXEC

_cancelTask()

Cancels a task owned by the smart contract.

```

Copy function_cancelTask(bytes32_taskId)internal

```

onlyDedicatedMsgSender

Function modifier to restrictmsg.sender to only task executions created bytaskCreator (defined in constructor)Dedicated msg.sender

```

Copy modifieronlyDedicatedMsgSender() { require(msg.sender==dedicatedMsgSender,"Only dedicated msg.sender"); _; }

```

PROXY module must be enabled. Or else the msg.sender will be the Automate smart contract inContract Addresses

_depositFunds()

```

Copy function_depositFunds(uint256_amount,address_token)internal

```

Deposit funds into the contract's Gelato balance.

withdrawFunds()

```

Copy functionwithdrawFunds(uint256_amount,address_token)external

```

Withdraw funds from the contract's Gelato balance. OnlyfundsOwner defined in the constructor can call this function.

Previous Gelato Automate SDK Next Guides Last updated3 months ago On this page *AutomateTaskCreator functions * _createTask() * ModuleData * _cancelTask() * onlyDedicatedMsgSender * _depositFunds() * withdrawFunds()