

This guide explains how to register an upkeep from within your smart contract, also called programmatic upkeep creation. Your contract can then interact with it via the registry to get its balance, fund it, edit it, or cancel it using theupkeepID.

Before beginning this process, complete the following tasks:

1. Ensure the smart contract you want to automate is [Automation Compatible](#) . To learn more about the contracts Chainlink Automation uses, click [here](#) .
2. Ensure you have sufficient LINK in the contract that will be registering the Upkeep. See here to get tested [LINK](#) .
3. Ensure you have the addresses for the LINK token you are using, and the correct registry/registrar. You can find these values on the [Supported Networks](#) page. Note: You can retrieve the LINK token address by calling the function `getLinkAddress` on the registry .
4. Use variables for the registry and registrar addresses that your admin can change as new versions of Chainlink Automation are released.
5. The interface for LINK and Registrar for registration, interface for Registry for subsequent actions
6. Interface is like the API specification of interacting with the contract.

Programmatically registering an upkeep happens in two steps:

1. Call the LINK token to give allowance to the Automation registrar for the amount of LINK you will fund your upkeep with at registration time, e.g. Pizza code to do
2. Call registerUpkeep on the Registrar contract using the RegistrationParams struct. You will receive the upkeepID if successful.

The variable type NameExample valueDescriptionStringNameTest upkeepName of upkeep that will be displayed in the UI.bytesencryptedEmail0xCan leave blank. If registering via UI we will encrypt email and store it here.addressupkeepContractAddressOf your Automation-compatible contractuint32gasLimit500000The maximum gas limit that will be used for your txns. Rather over-estimate gas since you only pay for what you use, while too low gas might mean your upkeep doesn't perform. Trade-off is higher gas means higher minimum funding requirement.addressadminAddressThe address that will have admin rights for this upkeep. Use your wallet address, unless you want to make another wallet the admin.uint8triggerType0 or 10 is Conditional upkeep, 1 is Log trigger.upkeepbytescheckData0xcheckData is a static input that you can specify now which will be sent into your checkUpkeep or checkLog, see interface.bytestriggerConfig0xThe configuration for your upkeep. 0x for conditional upkeeps, or see next section for log triggers.bytesoffchainConfig0xLeave as 0x, placeholder parameter for future.uint96amount1000000000000000000Ensure this is less than or equal to the allowance just given, and needs to be in WEI.

Depending on the trigger you are using, the `triggerConfig` will be different. Browse the triggers below to understand how to set up `triggerConfig`.

### Parameters

For upkeeps with triggers using onchain state only, the following parameters are needed:

[Code sample](#)

```
// SPDX-License-Identifier: MITPragmasolidity0.8.6;port[LinkTokenInterface]from"@chainlink/contracts/src/v0.8/shared/interfaces/LinkTokenInterface.sol";/* THIS IS AN EXAMPLE CONTRACT THAT USES HARDCODED VALUES FOR CLARITY. * THIS IS AN EXAMPLE CONTRACT THAT USES UN-AUDITED CODE. * DO NOT USE THIS CODE IN PRODUCTION. */structRegistrationParams{stringname;bytesencryptedEmail;addressupkeepContract;uint32gasLimit;addressadminAddress;uint8triggerType;bytescheckData;bytestriggerConfig;bytesof string name = "test upkeep"; bytes encryptedEmail = 0x...; address upkeepContract = 0x....; uint32 gasLimit = 500000; address adminAddress = 0x....; uint8 triggerType = 0; bytes checkData = 0x...; bytes triggerConfig = 0x...; bytes ofchainConfig = 0x...; uint96 amount = 1000000000000000000;}interfaceAutomationRegistrarInterface(functionregisterUpkeep(RegistrationParamscallDatarequestDataParams)externalreturns(uint256);contractUpkeepIDconditionalExample(LinkTokenInterfacepublicimmutable i_link;AutomationRegistrarInterfacepublicimmutable i_registrar;constructor(LinkTokenInterface link,AutomationRegistrarInterface registrar){ i_link=link;i_registrar=registrar;}functionregisterAndPredictID(RegistrationParamsmemoryparams)public// LINK must be approved for transfer - this can be done every time or once// with an infinite approval_i.link.approve(address(i_registrar),params.amount);uint256upkeepID=i_registrar.registerUpkeep(params);if(upkeepID==0){// DEV - Use the upkeepID however you see fit}else{revert("auto-approved disabled");}} Open in Remix What is Remix?
```

### Parameters

For upkeeps with triggers using emitted logs, the following parameters are needed:

```

struct LogTriggerConfig {
    address contractAddress; // must have address that will be emitting the login
    uint8 filterSelector; // must have filterSelector, denoting which topics apply to filter
    ex 000, 101, 111... only last 3 bits apply
    bytes32 topic0; // must have signature of the emitted event
    bytes32 topic1; // optional filter on indexed topic 1
    bytes32 topic2; // optional filter on indexed topic 2
    bytes32 topic3; // optional filter on indexed topic 3
    where filterSelector is a bitmask mapping and value is set depending on the selection of filters

    filterSelector
    Topic 1
    Topic 2
    Topic 3
    Empty
    Empty
    Empty
    1
    Filter
    Empty
    Empty
    2
    Empty
    Filter
    Empty
    3
    Filter
    Filter
    Empty
    4
    Empty
    Empty
    Filter
    5
    Filter
    Empty
    Filter
    6
    Empty
    Filter
    Filter
    7
    Filter
    Filter
    Filter
}

```

### Code sample

[illegible]