Register Upkeeps Programmatically

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 you begin

Before beginning this process, complete the following tasks:

1. Ensure the smart contract you want to automate isAutomation Compatible . To learn more about the contracts Chainlink Automation uses, clickhere .
2. Ensure you have sufficient LINK in the contract that will be registering the Upkeep. See here to get testnetLINK .
3. Ensure you have the addresses for the LINK token you are using, and the correct registry/registrar. You can find these values on theSupported Networks page.Note: You can retrieve the LINK token address by calling the functiongetLinkAddresson 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.

Register the upkeep

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. CallregisterUpkeepon the Registrar contract using theRegistrationParamsstruct. You will receive theupkeepIDif successful.

Var typeVar NameExample valueDescriptionStringname"Test upkeep"Name of upkeep that will be displayed in the UI.bytesencryptedEmail0xCan leave blank. If registering via UI we will encrypt email and store it here.addressupkeepContractAddress of 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.

Upkeep registration parameters and examples

Depending on the trigger you are using, thetriggerConfigwill be different. Browse the triggers below to understand how to set uptriggerConfig.

Custom logic upkeeps

Parameters

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

Code sample

// SPDX-License-Identifier: MITpragmasolidity^0.8.6;import{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;bytesoffc** * 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 offchainConfig = 0x; * uint96 amount = 1000000000000000000; */interfaceAutomationRegistrarInterface{functionregisterUpkeep(RegistrationParamscalldatarequestParams)externalreturns(uint256);}contractUpkeepIDConditionalExample{LinkTokenInterfacepublicimm 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 approvali_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-approve disabled");}}} Open in Remix What is Remix?

Log trigger upkeeps

Parameters

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

structLogTriggerConfig{addresscontractAddress;// must have address that will be emitting the loguint8filterSelector;// must have filtserSelector, denoting which topics apply to filter ex 000, 101, 111...only last 3 bits applybytes32topic0;// must have signature of the emitted eventbytes32topic1;// optional filter on indexed topic 1bytes32topic2;// optional filter on indexed topic 2bytes32topic3;// optional filter on indexed topic 3} where filterSelector is a bitmask mapping and value is set depending on the selection of filters

filterSelectorTopic 1Topic 2Topic 30EmptyEmptyEmpty1FilterEmptyEmpty2EmptyFilterEmpty3FilterFilterEmpty4EmptyEmptyFilter5FilterEmptyFilter6EmptyFilterFilter7FilterFilterFilter

Code sample

// SPDX-License-Identifier: MITpragmasolidity^0.8.6;import{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;bytesoffc** * Log trigger details * address contractAddress = 0x...; // e.g. 0x2938ff7cAB3115f768397602EA1A1a0Aa20Ac42f * uint8 filterSelector = 1; // see filterSelector * bytes32 topic0 = 0x...; // e.g. 0x74500d2e71ee75a8a83dcc87f7316a89404a0d0ac0c725e80c956dbf16fb8133 for event called bump * bytes32 topic1 = 0x...; // e.g. bytes32 of address 0x0000000000000000000000000000c26d7ef337e01a5cc5498d3cc2ff0610761ae637 * bytes32 topic2 = 0x; // empty so 0x * bytes32 topic3 = 0x; // empty so 0x * * Upkeep details * string name = "test upkeep"; * bytes encryptedEmail = 0x; * address upkeepContract = 0x...; * uint32 gasLimit = 500000; * address adminAddress = 0x....; * uint8 triggerType = 1; * bytes checkData = 0x; * bytes triggerConfig = abi.encode(address contractAddress, uint8 filterSelector,bytes32 topic0,bytes32 topic1,bytes32 topic2, bytes32 topic3); * bytes offchainConfig = 0x; * uint96 amount = 1000000000000000000; */interfaceAutomationRegistrarInterface{functionregisterUpkeep(RegistrationParamscalldatarequestParams)externalreturns(uint256);}contractUpkeepIDlogTriggerExample{LinkTokenInterfacepublicimmu 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 approvali_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-approve disabled");}}} Open in Remix What is Remix?