# Secure Upkeeps Using the Forwarder

This tutorial explains how to use theForwarderto add additional security to your Automation upkeeps. To learn how other Chainlink Automation contracts work, click here .

## What is a Forwarder? When is it used?

Starting with Automation 2.0, each registered upkeep under the Chainlink Automation network will have its own uniqueForwardercontract. TheForwarderaddress will only be known after registration, as we deploy a new forwarder for each upkeep. TheForwardercontract is the intermediary between the Automation Registry and your Upkeep contract. TheForwarderwill always be themsg.Senderfor your upkeep.

If yourperformUpkeepfunction is open and callable by anyone without risk of accepting unintentional external data, you don't need to use theForwarder.

## Securing your upkeep

If your upkeep's perform function needs to be permissioned, please consider addingmsg.sender = forwarderat the top of yourperformUpkeepfunction.

To make this work you will need to:

- Createforwarderas a mutable address variable on your contract that onlyyoucan update.forwarderis a unique value that cannot change for your upkeep.
- CreatesetForwarderfunction so you can update theforwarderaddress
- After registration runsetForwarderwith the forwarder address in your UI, or programmatically fetch it usingregistry.getForwarder(upkeepID)using the Registry interface.

## Code example

The code sample below uses the Forwarder:

// SPDX-License-Identifier: MITpragmasolidity^0.8.7;/ * **@dev Example contract which uses the Forwarder * * @notice important to implement {AutomationCompatibleInterface}** *// * 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. */import{AutomationCompatibleInterface}from"@chainlink/contracts/src/v0.8/automation/interfaces/AutomationCompatibleInterface.sol";import{Ownable}from"@openzeppelin/contracts/access/Ownable.so counter counts the number of upkeeps performeduint256publicinterval;// interval specifies the time between upkeepsuint256publiclastTimeStamp;// lastTimeStamp tracks the last upkeep performedaddresspublics_forwarderAddress;constructor(uint256updateInterval){interval=updateInterval;}functioncheckUpkeep(bytescalldata/checkData/)externaloverridereturns(bool,bytesmemory){boolneedsUpkeep=(block.timestamp-lastTimeStamp)>interval;return(needsUpkeep,bytes(""));}functionperformUpkeep(bytescalldata/performData*/)externaloverride{require(msg.sender==s_forwarderAddress,"This address does not have permission to call performUpkeep");lastTimeStamp=block.timestamp;counter=counter+1;}/// @notice Set the address that performUpkeep is called from/// @dev Only callable by the owner/// @param forwarderAddress the address to setfunctionsetForwarderAddress(addressforwarderAddress)externalonlyOwner{s_forwarderAddress=forwarderAddress;}} Open in Remix What is Remix?