# Automation Interfaces

Your Automation-compatible contracts may use the following interfaces. You can find them in the Chainlink repository . To understand how to implement these contracts, visit the Compatible Contracts page .

- If you want a log event to trigger your upkeep, use the ILogAutomation interface.
- If you want to use onchain state (excluding logs) in a custom calculation to trigger your upkeep, use AutomationCompatibleInterface interface.
- If you want to call a function just based on time, consider using a time-based upkeep .
- If you want to use Automation with Data Streams, use StreamsLookupCompatibleInterface interface.

## ILogAutomation

To use log triggers, you will need to implement the ILogAutomation.sol interface in your smart contract. Click on the functions below to understand the parameters:

Function NameDescriptioncheckLogSimulates offchain once a log that matches log specifications is emitted.performUpkeepContains the logic that should be executed onchain whencheckUpkeepreturnstrue.

### checkLog function

checkLogis aviewfunction that will be simulated offchain once a log that matches yourLogTriggerConfighas been emitted. You should parse the log data and the check if something needs to happen onchain. For example, it can trigger the retrieval of Data Streams reports. See the Data Streams Getting Started guide to see an example.

#### Parameters

Variable NameTypeDescriptionPermissible ValueslogstructLogStruct log data of the log triggering the upkeep; Automation will form the triggering log into a struct so the user can query log data incheckLog.struct Log { uint256 index; uint256 timestamp; bytes32 txHash; uint256 blockNumber; bytes32 blockHash; address source; bytes32[] topics; bytes data; }checkDatabytesOptional additional bytes the user wants to provide.checkDatais set at the time of registering the upkeep.

#### Example

// SPDX-License-Identifier: SEE LICENSE IN LICENSEpragmasolidity^0.8.7;structLog{uint256index;uint256timestamp;bytes32txHash;uint256blockNumber;bytes32blockHash;addresssource;bytes32[]topics;bytesdata;}functioncheckLog(Logcalldatal

### performUpkeep function for log triggers

This function contains the code that will be executed onchain to finalize the trade.

#### Parameters

Variable NameTypeDescriptionPermissible ValuesperformDatabytesData encoded in thecheckLogfunction that will be used to execute the function onchainbytes

## AutomationCompatibleInterface

Custom logic upkeeps need to use the AutomationCompatibleInterface.sol interface. Click on one of the functions below to understand its parameters and limits.

Function NameDescriptioncheckUpkeepRuns offchain to determine if theperformUpkeepfunction should be called onchain.performUpkeepContains the logic that should be executed onchain whencheckUpkeepreturns true.

### checkUpkeep function

This view function contains the logic that runs offchain during every block as an eth_call to determine ifperformUpkeepshould be executed onchain. To reduce onchain gas usage, attempt to do your gas intensive calculations offchain incheckUpkeepand pass the result toperformUpkeeponchain. It is a best practice to import the AutomationCompatible.sol contract and use thecannotExecutemodifier to ensure that the method can be used only for simulation purposes.

functioncheckUpkeep(bytescalldatacheckData)externalviewoverridereturns(boolupkeepNeeded,bytesmemoryperformData); Below are the parameters and return values of thecheckUpkeepfunction. Click each value to learn more about its design patterns and best practices:

Parameters:

- checkData: Fixed and specified at upkeep registration and used in everycheckUpkeep. Can be empty (0x).

Return Values:

- upkeepNeeded: Boolean that when True will trigger the onchainperformUpkeepcall.
- performData: Bytes that will be used as input parameter when callingperformUpkeep. If you would like to encode data to decode later, tryabi.encode.

#### checkData

You can pass information into yourcheckUpkeepfunction from your upkeep registration to execute different code paths. For example, to check the balance on a specific address, set thecheckDatato abi encode the address. To learn how to create flexible upkeeps with checkData, please see our flexible upkeeps page.

functioncheckUpkeep(bytescalldatacheckData)publicviewreturns(bool,bytesmemory){addresswallet=abi.decode(checkData,(address));return(wallet.balance<1ether,bytes(""));} Tips on usingcheckData:

- Managing unbounded upkeeps: Limit the problem set of your onchain execution by creating a range bound for your upkeep to check and perform. This allows you to keep within predefined gas limits, which creates a predictable upper bound gas cost on your transactions. Break apart your problem into multiple upkeep registrations to limit the scope of work.

Example: You could create an upkeep for each subset of addresses that you want to service. The ranges could be 0 to 49, 50 to 99, and 100 to 149. * Managing code paths: Pass in data to yourcheckUpkeepto make your contract logic go down different code paths. This can be used in creative ways based on your use case needs.

Example: You could support multiple types of upkeep within a single contract and pass a function selector through thecheckDatafunction.

#### performData

The response fromcheckUpkeepis passed to theperformUpkeepfunction asperformData. This allows you to perform complex and gas intensive calculations as a simulation offchain and only pass the needed data onchain.

You can create a highly flexible offchain computation infrastructure that can perform precise actions onchain by usingcheckDataandperformData. Both of these computations are entirely programmable.

### performUpkeep function for custom logic triggers

WhencheckUpkeepreturnsupkeepNeeded == true, the Automation node broadcasts a transaction to the blockchain to execute yourperformUpkeepfunction onchain withperformDataas an input.

Ensure that yourperformUpkeepisidempotent. YourperformUpkeepfunction should change state such thatcheckUpkeepwill not returntruefor the same subset of work once said work is complete. Otherwise the Upkeep will remain eligible and result in multiple performances by the Chainlink Automation Network on the exactly same subset of work. As a best practice, always check conditions for your upkeep at the start of yourperformUpkeepfunction.

functionperformUpkeep(bytescalldataperformData)externaloverride; Parameters:

- performData: Data which was passed back from thecheckDatasimulation. If it is encoded, it can easily be decoded into other types by callingabi.decode. This data should always be validated against the contract's current state.

#### performData

You can perform complex and broad offchain computation, then execute onchain state changes on a subset that meets your conditions. This can be done by passing the appropriate inputs withinperformDatabased on the results from yourcheckUpkeep. This pattern can greatly reduce your onchain gas usage by narrowing the scope of work intelligently in your own Solidity code.

- Identify a list of addresses that require work: You might have a number of addresses that you are validating for conditions before your contract takes an action. Doing this onchain can be expensive. Filter the list of addresses by validating the necessary conditions within yourcheckUpkeepfunction. Then, pass the addresses that meet the condition through theperformDatafunction. For example, if you have a "top up" contract that ensures several hundred account balances never decrease below a threshold, pass the list of accounts that meet the conditions so that

theperformUpkeepfunction validates and tops up only a small subset of the accounts.
- Identify the subset of states that must be updated: If your contract maintains complicated objects such as arrays and structs, or stores a lot of data, you should read through your storage objects within yourcheckUpkeepand run your proprietary logic to determine if they require updates or maintenance. After that is complete, you can pass the known list of objects that require updates through theperformDatafunction.

## StreamsLookupCompatibleInterface

See theData Streams Getting Started guide to see an example of how to use this interface.

To use Data Streams with Automation, your contract must beAutomation-compatible and include aStreamsLookupCompatibleInterfaceinterface. This interface fetches and processes Data Streams reports.

Function NameDescriptionStreamsLookuprevertTriggers the retrieval of specified reports.checkCallbackSimulates offchain to receive signed reports and conduct final parsing before sending data onchain viaperformUpkeep.

### StreamsLookup revert

Automation network will use this revert to trigger fetching of the specified reports.

#### Parameters

Variable NameTypeDescriptionPermissible ValuesfeedParamKeyStringSpecify the feed identifiers that will be provided"feedIDs"feedsString[]String list of feed identifierse.g.["feedID1","feedID2",..]with a maximum of 5 IDstimeParamKeyStringSpecify query type"timestamp"timeuint256Specify query valueblock.timestampextraDatabytesAny extra data user wants to receive in callback, alongside API bytes[]log data

#### Outputs

The fetched signed reports will be provided to the user incheckCallback . This is an offchain compute that the user can use to structure data as needed before performing the onchain confirmation and validation transactions.

Variable NameTypeDescriptionUsage examplesvaluesbytes[]List of signed reports fetched from APIList of signed reportsvalues[0], values[1]…ordered to match order in feedsextraDatabytesBytes data sent as part of original MercuryLookup revert errore.g. can send log data if you want to decode to use

### checkCallback function

This is aviewfunction that will be simulated offchain to receive the signed reports and conduct any final parsing before sending data onchain viaperformUpkeep. Inputs will match outputs of theoracleLookuprevert.

#### Parameters

Variable NameTypeDescriptionPermissible Valuesvaluesbytes[]List of signed reports fetched from APIList of signed reportsvalues[0], values[1]…ordered to match order in feedsextraDatabytesBytes data sent as part of original MercuryLookup revert errore.g. can send log data if you want to decode to use

#### Outputs

Variable NameTypeDescriptionPermissible ValuesperformDatabytesData encoded in theoracleCallbackfunction that will be used to execute function onchain, e.g. encode signed reports and log data if need bebytesupkeepNeededbooleanWhen set totruethis will trigger theperformUpkeepfunction to be executed onchaintrueoffalse

### checkErrorHandler function

This function is simulated offchain to respond toerror codes before sending data onchain in theperformUpkeepfunction.

#### Parameters

Variable NameTypeDescriptionPermissible ValueserrorCodeuintError code returned by StreamsLookup error handlerExample: "808400" (ErrCodeStreamsBadRequest)extraDatabytesBytes data sent as part of the originalMercuryLookuprevert errorThis field can send log data if you want to decode it and use it.

#### Outputs

Variable NameTypeDescriptionPermissible ValuesperformDatabytesData encoded in theoracleCallbackfunction that will be used to execute function onchain, e.g. encode signed reports and log data if need bebytesupkeepNeededbooleanWhen set totrue, this triggers theperformUpkeepfunction to be executed onchaintrueorfalse