

# Automation Contracts

Automation Nodes use the following contracts. You can find them in the [Chainlink repository](#) . For details about how to use them, visit the [Creating Compatible Contracts](#) guide. To understand the logic behind these contracts, visit the [Architecture](#) page.

- [AutomationCompatible.sol](#) : Imports the following contracts: \* [AutomationBase.sol](#) : Enables the use of the `cannotExecute` modifier. Import this contract if you need for this modifier. See the [checkUpkeepfunction](#) for details.
- [AutomationCompatibleInterface.sol](#) : The interface to be implemented in order to make your contract compatible. Import this contract for type safety.

## [AutomationRegistry.sol](#)

[AutomationRegistry2\\_1.sol](#) : The registry contract that tracks all registered Upkeeps and the Automation Nodes that can perform them. Note: As Chainlink Automation continues adding new functionalities, a new Automation Registry is deployed and the contract address may change.

## [AutomationRegistrar.sol](#)

[AutomationRegistrar2\\_1.sol](#) : The Registrar contract governs the registration of new Upkeeps on the associated Automation Registry contract. Users who want to register Upkeeps by directly calling the deployed contract have to call the Transfer-and-Call function on the respective ERC-677 LINK contract configured on the Registrar and ensure they pass the correct encoded function call and inputs.

## [UpkeepTranscoder.sol](#)

[UpkeepTranscode4\\_0.sol](#) allows the conversion of upkeep data from previous Automation registry versions 1.2, 1.3, and 2.0 to registry 2.1.

## [AutomationForwarder.sol](#)

[AutomationForwarder.sol](#) is a relay that sits between the registry and the customer's target contract. The purpose of the forwarder is to give customers a consistent address to authorize against that stays consistent between migrations. The Forwarder also exposes the registry address, so that users who want to programmatically interact with the registry can do so. The `forward` function in this contract is called by the registry and forwards the call to the target.

## [CronUpkeepFactory.sol](#)

[CronUpkeepFactory.sol](#) serves as a delegate for all instances of CronUpkeep. Those contracts delegate their `checkUpkeep` calls onto this contract. Utilizing this pattern reduces the size of the CronUpkeep contracts. You can use this contract when creating a time-based upkeep programmatically. You can learn more about creating upkeeps programmatically [here](#) .