Escalation Managers

Using the full policy escalation manager as an example for modular functionality you could implement with custom escalation managers

Table of Contents

Introduction

TheFull Policy Escalation Manager is an example implementation ofEscalation Manager that allows for a high level of control over the arbitration process in theOptimistic Oracle V3 . It can be used to set various policy parameters such as who is able to assert truths, who is able to file disputes, and whether disputes should be arbitrated via the escalation manager or theUMA Data Verification Mechanism (DVM) . By using the Escalation Managers, such as the Full Policy Escalation Manager, users can better tailor the arbitration process to suit their specific needs and security requirements.

An Escalation Manager is a smart contract that is used in the Optimistic Oracle V3 to handle the escalation policy for assertions and can also be used to arbitrate disputes. When one party makes an assertion, it might be challenged by another, known as the disputer. In the event of a dispute, the Escalation Manager is used to decide how the dispute should be handled and how the Optimistic Oracle V3 should respond to it.

The Full Policy Escalation Manager is an Escalation Manager implementation that allows the contract owner to set all policy parameters and store arbitration resolutions for the Escalation Manager.

Some of the key functionality that the Full Policy Escalation Manager allows are:

- Enabling a whitelist of asserting callers, asserters and disputers. This allows the owner to limit who can assert truths, and who can dispute assertions.
- Determining whether disputes should be arbitrated by the Escalation Manager or by the Data Verification Mechanism (DVM).
- Determining whether the resolution of a potential dispute arbitrated by the Oracle should be disregarded (i.e a disputed assertion defaults to false outcome).
- Storing and managing the arbitration resolutions for a given identifier, time and ancillary data.
-

This smart contract is ideal for use cases where a high degree of control over the behaviour of the Escalation Manager is needed, and where the owner wants to have a say in how disputes are handled and acts as a good starting point for projects that want to build their own escalation managers that implement custom logic.

Use Cases

The FullPolicyEscalationManager can be used in a variety of ways depending on the specific needs of the project. Below are a few examples of possible use cases and how to configure the escalation manager for each:

- Blocking assertions by untrusted parties
- : In this case, the project may want to prevent assertions from being made by certain parties that are not trusted. To accomplish this, theblockByAssertingCaller
- and/orblockByAsserter
- parameters can be set totrue
- and the corresponding whitelists can be populated with trusted parties using thesetAssertingCallerWhitelist
- andsetAsserterWhitelist
- functions.
- Validation of disputers
- : In some cases, the project may want to prevent disputes from being filed by untrusted parties. To accomplish this, thevalidateDisputers
- parameter can be set totrue
- and thewhitelistedDisputeCallers
- whitelist can be populated with trusted parties using thesetDisputeCallerWhitelist
- function.
- Allowing arbitration via the escalation manager
- : By default, disputes are arbitrated via the UMA DVM. However, in some cases, the project may want to use the escalation manager to arbitrate disputes instead. To accomplish this, thearbitrateViaEscalationManager
- parameter can be set totrue

- .
- Discarding the oracle resolution
- : In some cases, the project may want to disregard the oracle's resolution when arbitrating disputes. To accomplish this, thediscardOracle
- parameter can be set totrue
- .
- Custom arbitration resolution
- : If the project wants to define a custom resolution for a given identifier, time, and ancillary data, it can be set using thesetArbitrationResolution
- function.
- Mixing different options
- : The FullPolicyEscalationManager can be customized to mix different options, such as allowing arbitration via the escalation manager, but also blocking assertions by untrusted parties, or validating disputers but also allowing custom arbitration resolution.
-

It's important to note that the values ofblockByAssertingCaller ,blockByAsserter ,validateDisputers ,arbitrateViaEscalationManager ,discardOracle are set to false by default.

Events

- EscalationManagerConfigured
- : Emitted when the escalation manager is configured using theconfigureEscalationManager
- function. The event contains the following parameters:
-
    - blockByAssertingCaller
-
    - : A boolean indicating whether assertions are allowed only by whitelisted asserting callers.
-
    - blockByAsserter
-
    - : A boolean indicating whether assertions are allowed only by whitelisted asserters.
-
    - validateDisputers
-
    - : A boolean indicating whether the escalation managershould validate disputers via the whitelistedDisputeCallers mapping.
-
    - arbitrateViaEscalationManager
-
    - : A boolean indicating whether it is determined that the escalation manager should arbitrate disputes.
-
    - discardOracle
-
    - : A boolean indicating whether the escalation manager should disregard the Oracle’s resolution of disputes.
- *
- ArbitrationResolutionSet
- : Emitted when an arbitration resolution is set using thesetArbitrationResolution
- function. The event contains the following parameters:
-
    - identifier
-
    - : The identifier of the assertion.
-
    - time
-
    - : The time of the assertion.
-
    - ancillaryData
-
    - : The ancillary data of the assertion.
-
    - resolution
-
    - : The resolution of the arbitration.
- *
- DisputeCallerWhitelistSet
- : Emitted when a dispute caller is added or removed from the whitelistedDisputeCallers mapping using theaddDisputeCallerToWhitelist

- andremoveDisputeCallerFromWhitelist
- functions. The event contains the following parameters:
-
    - disputeCaller
-
    - : The address of the dispute caller that was added or removed from the whitelist.
-
    - whitelisted
-
    - : A boolean indicating whether the dispute caller was added (true) or removed (false) from the whitelist.
- *
- AssertingCallerWhitelistSet
- : Emitted when an asserting caller is added or removed from the whitelistedAssertingCallers mapping using theaddAssertingCallerToWhitelist
- andremoveAssertingCallerFromWhitelist
- functions. The event contains the following parameters:
-
    - assertingCaller
-
    - : The address of the asserting caller that was added or removed from the whitelist.
-
    - whitelisted
-
    - : A boolean indicating whether the asserting caller was added (true) or removed (false) from the whitelist.
- *
- AsserterWhitelistSet
- : Emitted when an asserter is added or removed from the whitelistedAsserters mapping using theaddAsserterToWhitelist
- andremoveAsserterFromWhitelist
- functions. The event contains the following parameters:
-
    - asserter
-
    - : The address of the asserter to add.
-
    - whitelisted
-
    - : true represents adding and false represents removing the asserter from the whitelist.
- *
-

Functions

getAssertionPolicy

This function returns the Assertion Policy defined by the contract's parameters and functions.

Parameters

- assertionId
- : the ID of the assertion to get the policy for.
-

Returns

- AssertionPolicy memory
- : an object that contains the following properties:
-
    - blockAssertion
-
    - : a boolean indicating whether the assertion is blocked or not.
-
    - arbitrateViaEscalationManager
-
    - : a boolean indicating whether the arbitration should be done via the escalation manager if it is configured.
-
    - discardOracle
-
    - : a boolean indicating whether to ignore the Oracle (DVM or EM) resolution if it is configured.

-
    - validateDisputers
-
    - : a boolean indicating whether to validate disputers if it is configured.
- *
-

Notes

If no configuration is done after deployment, this function returns an all false default policy.

getPrice

ThegetPrice function allows developers to retrieve the resolved price for a given identifier and timestamp. It replicates the interface of the corresponding DVM function to allow the user to use his own dispute arbitration system when arbitrating via the escalation manager in a DVM-compatible manner. It is useful when you want to get the resolved price for a specific request.

Parameters

- identifier
- : A unique identifier that identifies the price requested.
- time
- : A unix timestamp of the price request.
- ancillaryData
- : Arbitrary data appended to a price request to give the voters more info from the caller.
-

Returns

- int256
- : representing the resolved price for the given identifier and timestamp.
-

Notes

If the price is not available, the method reverts. Also, the function requires that the arbitration resolution has been set before calling it.

isDisputeAllowed

This function returns whether a given disputerCaller is authorized to dispute a given assertionId or not.

Parameters

- assertionId
- : The ID of the assertion to check the disputerCaller for.
- disputeCaller
- : The address of the disputeCaller to check.
-

Returns

- bool
- : true if the disputerCaller is authorized to dispute the assertion, false otherwise.
-

Notes

- In order for this function to be used by the Optimistic Oracle V3, validateDisputers must be set to true.
- The whitelistedDisputeCallers is a mapping that contains the addresses of the authorized disputers, which are set through the setDisputeCallerInWhitelist function.
-

configureEscalationManager

This function is used to configure the escalation manager and define the assertion policy for each configuration's rules.

Parameters

- _blockByAssertingCaller

- : A boolean value that indicates if assertions are allowed only by whitelisted asserting callers.
- _blockByAsserter
- : A boolean value that indicates if assertions are allowed only by whitelisted asserters.
- _validateDisputers
- : A boolean value that indicates if the escalation manager should validate disputers via whitelistedDisputeCallers.
- _arbitrateViaEscalationManager
- : A boolean value that indicates if the escalation manager should arbitrate instead of the DVM.
- _discardOracle
- : A boolean value that indicates if the escalation manager should disregard the Oracle's (DVM or EM) resolution.
-

Notes

- This function can only be called by the contract's owner.
- It is important to note that this function only activates the rules that will be executed, each rule must additionally be defined using the other functions.
- The _blockByAsserter must be true if _blockByAssertingCaller is true.
- When this function is called, an eventEscalationManagerConfigured
- is emitted with the parameters of the function call.
-

setArbitrationResolution

This function is used to set the arbitration resolution for a given identifier, time, and ancillary data. This function should be used by the contract owner whenever a dispute arises and it should be arbitrated by the Escalation Manager. The owner of the contract is responsible for determining how to resolve the dispute.

Parameters

- identifier
- : A unique identifier that represents the price requested. This is used to identify the price request in question.
- time
- : A unix timestamp of the price request.
- ancillaryData
- : Arbitrary data that is appended to a price request to give the voters more information from the caller.
- arbitrationResolution
- : A boolean value that indicates if the assertion should be resolved as true (true) or false (false).
-

Notes

- This function is only callable by the owner of the contract.
- This function requires that thearbitrationResolutions[requestId].valueSet
- is equal to false, otherwise the function will revert with the error message "Arbitration already resolved".
- AArbitrationResolutionSet
- event is emitted with the parametersidentifier
- ,time
- ,ancillaryData
- , andarbitrationResolution
- when this function is called.
- ThegetRequestId
- function is used to generate a requestId to uniquely identify the price request. The requestId is generated from the identifier, time, and ancillaryData parameters.
-

setDisputeCallerInWhitelist

ThesetDisputeCallerInWhitelist function allows the owner of theFullPolicyEscalationManager contract to add or remove a given address (disputeCaller) to the whitelist of addresses that are able to file disputes.

Parameters

- disputeCaller
- : The address of the dispute caller to add or remove from the whitelist.
- value
- : A boolean value that represents whether thedisputeCaller
- is being added (true
-) or removed (false
-) from the whitelist.

-

Notes

This function is only used if thevalidateDisputers parameter is set totrue . When a dispute caller is added or removed from the whitelist, theDisputeCallerWhitelistSet event is emitted with thedisputeCaller andvalue as the arguments.

setWhitelistedAssertingCallers

This function allows the owner of the contract to add or remove an address from the whitelist of asserting callers that are authorized to make assertions.

Parameters

- assertingCaller
- : The address of the asserting caller to add or remove from the whitelist.
- value
- : A boolean value that represents whether the asserting caller is being added (true
-) or removed (false
-) from the whitelist.
-

Notes

- This function can only be called by the owner of the contract.
- An eventAssertingCallerWhitelistSet
- is emitted upon the successful execution of this function, which includes theassertingCaller
- and thevalue
- as the arguments.
- The whitelist of asserting callers is used when theblockByAssertingCaller
- configuration is set totrue
- . In this case, only addresses in the whitelist are allowed to make assertions.
-

setWhitelistedAsserters

This function allows an owner to add or remove an address from the whitelist of authorized asserters.

Parameters

- asserter
- : The address of the asserter to add or remove from the whitelist.
- value
- : A boolean value that determines whether the asserter is added or removed from the whitelist.true
- represents adding the asserter to the whitelist andfalse
- represents removing the asserter from the whitelist.
-

Notes

- This function can only be called by the contract owner.
- This function must be used in conjunction withsetWhitelistedAssertingCallers
- in order to have an effect. This means that ifsetWhitelistedAssertingCallers
- is set to true, the assertion will only be made by an address that is in both whitelistswhitelistedAssertingCallers
- andwhitelistedAsserters
- .
- When an asserter is added or removed from the whitelist, an eventAsserterWhitelistSet
- is emitted with theasserter
- andvalue
- as the arguments.
-

Getting started

To get started with the Full Policy Escalation Manager, the first step is to deploy the contract on the EVM network. This can be done using hardhat and a library such asethers.js . Here is an example of how to deploy the contract:

```

Copy consthre=require("hardhat");
```

import{ FullPolicyEscalationManagerEthers__factory }from"@uma/contracts-node";

const{getContractFactory}=hre.ethers;

const{getAddress}=require("@uma/contracts-node");

asyncfunctionmain() { console.log("Running FullPolicyEscalationManager Deployments   ");

constnetworkId=Number(awaithre.getChainId());

constoptimisticOracleV3=getAddress("OptimisticOracleV3",networkId);

constfullPolicyEscalationManagerFactory:FullPolicyEscalationManagerEthers__factory=awaitgetContractFactory("FullPolicyEscalationManager");

constfullPolicyEscalationManager=awaitfullPolicyEscalationManagerFactory.deploy(optimisticOracleV3);

console.log("Deployed FullPolicyEscalationManager: ",fullPolicyEscalationManager.address); }

main() .then(()=>process.exit(0)) .catch((error)=>{ console.error(error); process.exit(1); });

```

Once the contract is deployed, it can be configured to handle disputes in different ways, depending on the use case. TheconfigureEscalationManager function can be used to set the rules for how disputes should be handled. Here are some examples of how to configure the contract for different use cases:

Example 1:

```

Copy //Configure the contract to block assertions by asserting callers that are not whitelisted awaitfullPolicyEscalationManager.configureEscalationManager(true,false,false,false,false);

//Whitelist an asserting caller and asserter awaitfullPolicyEscalationManager.setWhitelistedAssertingCallers("0x123456789abcdef0123456789abcdef0123456789",true);

```

Example 2:

```

Copy //Configure the contract to validate disputers and arbitrate via the escalation manager awaitfullPolicyEscalationManager.configureEscalationManager(false,false,true,true,false);

//Whitelist a dispute caller awaitfullPolicyEscalationManager.setDisputeCallerInWhitelist("0x123456789abcdef0123456789abcdef0123456789",true);

```

Example 3:

```

Copy //Configure the contract to disregard the oracle's resolution awaitfullPolicyEscalationManager.configureEscalationManager(false,false,false,false,true);

```

The Full Policy Escalation Manager can be configured for a variety of use cases depending on the requirements. The examples above show how to deploy the contract, configure it for different use cases and whitelist asserting callers, asserters and disputers. Keep in mind that the contract can only be configured by the owner and that the parameters of configureEscalationManager function must be set according to the desired use case.

Previous Data Asserter Next Sandboxed Oracle Environment Last updated1 month ago On this page *Table of Contents * Introduction * Use Cases * Events * Functions * getAssertionPolicy * getPrice * isDisputeAllowed * configureEscalationManager * setArbitrationResolution * setDisputeCallerInWhitelist * setWhitelistedAssertingCallers * setWhitelistedAsserters * Getting started

Was this helpful? Edit on GitHub
```