Oracle Security Module (OSM) - Detailed Documentation

- Contract Name:
- OSM
- Type/Category:
- Oracles - Price Feed Module
- Associated MCD System Diagram
- Contract Source
-

1. Introduction

Summary

The OSM (named via acronym from "Oracle Security Module") ensures that new price values propagated from the Oracles are not taken up by the system until a specified delay has passed. Values are read from a designatedDSValue contract (or any contract that has theread() andpeek() interfaces) via thepoke() method; theread() andpeek() methods will give the current value of the price feed, and other contracts must be whitelisted in order to call these. An OSM contract can only read from a single price feed, so in practice one OSM contract must be deployed per collateral type.

?

1. Contract Details - Glossary (OSM)

Storage Layout

- stopped
- : flag (uint256
-) that disables price feed updates if non-zero
- src
- :address
- of DSValue that the OSM will read from
- ONE_HOUR
- : 3600 seconds (uint16(3600)
-)
- hop
- : time delay betweenpoke
- calls (uint16
-); defaults toONE_HOUR
- zzz
- : time of last update (rounded down to nearest multiple ofhop
-)
- cur
- :Feed
- struct that holds the current price value
- nxt
- :Feed
- struct that holds the next price value
- bud
- : mapping fromaddress
- touint256
- ; whitelists feed readers
-

Public Methods

Administrative Methods

These functions can only be called by authorized addresses (i.e. addressesusr such thatwards[usr] == 1).

- rely
- /deny
- : add or remove authorized users (via modifications to thewards
- mapping)
- stop()
- /start()
- : toggle whether price feed can be updated (by changing the value ofstopped
-)

- change(address)
- : change data source for prices (by settingsrc
-)
- step(uint16)
- : change interval between price updates (by settinghop
-)
- void()
- : similar tostop
- , except it also setscur
- andnxt
- to aFeed
- struct with zero values
- kiss(address)
- /diss(address)
- : add/remove authorized feed consumers (via modifications to thebuds
- mapping)
-

Feed Reading Methods

These can only be called by whitelisted addresses (i.e. addressesusr such thatbuds[usr] == 1):

- peek()
- : returns the current feed value and a boolean indicating whether it is valid
- peep()
- : returns the next feed value (i.e. the one that will become the current value upon the nextpoke()
- call), and a boolean indicating whether it is valid
- read()
- : returns the current feed value; reverts if it was not set by some valid mechanism
-

Feed Updating Methods

- poke()
- : updates the current feed value and reads the next one
-

Feed struct: a struct with twouint128 members,val andhas . Used to store price feed data.

1. Key Mechanisms & Concepts

The central mechanism of the OSM is to periodically feed a delayed price into the MCD system for a particular collateral type. For this to work properly, an external actor must regularly call thepoke() method to update the current price and read the next price. The contract tracks the time of the last call topoke() in thezzz variable (rounded down to the nearest multiple ofhop ; seeFailure Modes for more discussion of this), and will not allowpoke() to be called again untilblock.timestamp is at leastzzz+hop . Values are read from a designated DSValue contract (its address is stored insrc). The purpose of this delayed updating mechanism is to ensure that there is time to detect and react to an Oracle attack (e.g. setting a collateral's price to zero). Responses to this include callingstop() orvoid() , or triggering Emergency Shutdown.

Other contracts, if whitelisted, may inspect thecur value via thepeek() andread() methods (peek() returns an additional boolean indicating whether the value has actually been set;read() reverts if the value has not been set). Thenxt value may be inspected viapeep() .

The contract uses a dual-tier authorization scheme: addresses mapped to 1 inwards may start and stop, set thesrc , callvoid() , and add new readers; addresses mapped to 1 inbuds may callpeek() ,peep() , andread() .

1. Gotchas (Potential Sources of User Error)

Confusingpeek() forpeep() (or vice-versa)

The names of these methods differ by only a single character and in current linguistic usage, both "peek" and "peep" have essentially the same meaning. This makes it easy for a developer to confuse the two and call the wrong one. The effects of such an error are naturally context-dependent, but could e.g. completely invalidate the purpose of the OSM if thepeep() is called where insteadpeek() should be used. A mnemonic to help distinguish them: "since 'k' comes before 'p' in the English alphabet, the value returned bypeek() comes before the value returned bypeep() in chronological order". Or: "peek() returns thek urrent value".

1. Failure Modes (Bounds on Operating Conditions & External Risk Factors)

poke() is not called promptly, allowing malicious prices to be swiftly uptaken

For several reasons,poke() is always callable as soon asblock.timestamp / hop increments, regardless of when the lastpoke() call occurred (becausezzz is rounded down to the nearest multiple ofhop). This means the contract does not actually guarantee that a time interval of at leasthop seconds has passed since the lastpoke() call before the next one; rather this is only (approximately) guaranteed if the lastpoke() call occurred shortly after the previous increase ofblock.timestamp / hop . Thus, a malicious price value can be acknowledged by the system in a time potentially much less thanhop .

This was a deliberate design decision. The arguments that favoured it, roughly speaking, are:

- Providing a predictable time at which MKR holders should check for evidence of oracle attacks (in practice,hop
- is 1 hour, so checks must be performed at the top of the hour)
- Allowing all OSMs to be reliably poked at the same time in a single transaction
-

The fact thatpoke is public, and thus callable by anyone, helps mitigate concerns, though it does not eliminate them. For example, network congestion could prevent anyone from successfully callingpoke() for a period of time. If an MKR holder observes thatpoke has not been promptly called,the actions they can take include:

1. Callpoke()
2. themselves and decide if the next value is malicious or not
3. Callstop()
4. orvoid()
5. (the former if onlynxt
6. is malicious; the latter if the malicious value is already incur
7.)
8. Trigger emergency shutdown (if the integrity of the overall system has already been compromised or if it is believed the rogue oracle(s) cannot be fixed in a reasonable length of time)
9.

In the future, the contract's logic may be tweaked to further mitigate this (e.g. byonly allowingpoke() calls in a short time window eachhop period).

Authorization Attacks and Misconfigurations

Various damaging actions can be taken by authorized individuals or contracts, either maliciously or accidentally:

- Revoking access of core contracts to the methods that read values, causing mayhem as prices fail to update
- Completely revoking all access to the contract
- Changingsrc
- to either a malicious contract or to something that lacks apeek()
- interface, causing transactions thatpoke()
- the affected OSM to revert
- Calling disruptive functions likestop
- andvoid
- inappropriately
-

The only solution to these issues is diligence and care regarding thewards of the OSM.

Previous Oracle Module Next Median - Detailed Documentation Last updated4 years ago On this page *1. Introduction * Summary * 2. Contract Details - Glossary (OSM) * Storage Layout * Public Methods * 3. Key Mechanisms & Concepts * 4. Gotchas (Potential Sources of User Error) * Confusing peek() for peep() (or vice-versa) * 5. Failure Modes (Bounds on Operating Conditions & External Risk Factors) * Authorization Attacks and Misconfigurations

Export as PDF