MS Exchange 2013

Дефиниране:

1. Managed availability
   1. *Probe engine* – записва поведението на компонентите
   2. *Monitoring probe engine* – взима решение дали даден компонент е Healthy, като сравни това което *Probe* е записъл и това което е предефинирано (т.е. стандартното поведение - *predefined thresholds*). Всяка корекция дефинираща, кое е стандартното поведение и/или промяна в действието, което се изпраща към *Responder* се нарича *Override.*
   3. *Responder engine* – try to recover
2. *Managed availability* се базира на два процеса – ако втория се счупи, целият *Managed availiability* компонент спира да работи
   1. **msexchangehmworker.exe** – the worker process that performs tasks
   2. **msexchangehmhost.exe (Exchange Health Manager Service)** – it manages worker processes
      1. **Set-ServerComponentState –** използва се, за да се спре горния service. Силно НЕ се препоръчва да се спира от task manager
         1. Set-ServerComponentState -Identity **ServerName** -Component Monitoring -Requester Functional -State Inactive
3. *Override* – на всеки компонент на Exchange, който се наблюдава от *Monitoring*, е записано какво е обичайното му поведение. *Monitoring*-a сравнява записаното от *Probe*-a и обичайното поведение. В случай на отклонение се изпраща информация към *Responder*, който изпълнява определени действия за връщане на компонента в “нормалното” му състояния*.* Възможно е да се промени какво е “нормално”, както и какви действия да извърши *Responder*-a. Това се прави в случаи, когато default настройките не вършат работа и се нарича *Override*. Съществуват локални и глобални настройки. Локалните са само за съвъра, на който го прилагаш, а глобалните обхващат множество от сървъри.
   1. Add|Get|Remove-ServerMonitoringOverride – това е за локални промени
   2. Add|Get|Remove-GlobalMonitoringOverride – това е за глобални промени
4. Логове
   1. Event Viewer -> Applications and Services Logs -> Microsoft -> Exchange -> Active Monitoring
   2. Event Viewer -> Applications and Services Logs -> Microsoft -> Exchange -> Managed Availability
5. *Health Mailboxes* – използва се за симулации с акаунти, като например изпращане и получаване на писма.
   1. В ADDC: Скрит контейнер в домейна – Microsoft Exchange System Object > Monitoring mailboxes – Вътре са 10 акаунта, с които се правят тестовете и се препоръчва да НЕ се пипат, нито местят.

cmdlets information:

* ***Get-HealthReport*** - използва се за да видиш Health sets, дали са активни/неактивни, healthy/unhealthy/degradeted(когато под минута не е healthy)
  + Get-HealthReport -Server exchangeSERVER | where {$\_.alertvalue -ne "Healthy" –and $\_.AlertValue –ne "Disabled"}
* ***Get-ServerHealth*** – използва се, когато си видял HealthSet който примерно е Unhealthy и искаш да видиш кой компонент в него е Unhealthy.
  + Get-ServerHealth exchangeSERVER | ?{$\_.HealthSetName -eq "HubTransport"}
* ***Invoke-MonitoringProbe*** – използва се, когато компонента който си избрал да погледнеш има probe. От резултата, който ти върне можеш да разбереш дали изменението още е валиднo, ако ResultTyper: Success*,* ако е Failed продължаваме да борим проблема.
  + Invoke-MonitoringProbe HubTransport\HubAvailabilityProbe -Server exchangeSERVER | Format-List
* ***Set-ServerMonitor*** – използва се, когато искаш да спреш или да пуснеш мониторинг. Примерно когато ще поправяш ръчно нещо.
  + Set-ServerMonitor ~~-~~Server exchangeSERVER -Name Maintenance -Repairing $true
* ***Get-MonitoringItemIdentity*** - cmdlet allows displaying probes, monitors, and responders associated to a particular health set.
* ***Get-ServerComponentState*** -Identity **ServerName** -Component ComponentName[-component is optional] – Използва се за да изкараш списък с компонентите на Exchange и тяхното състояние. Може да променяш състоянието на компонентите ръчно с команда **Set-ServerComponentState** – Identity ServerName -Component ComponentName -Requester Maintenance|HealthAPI|Sidelined|Functional|Deployment -State Active|Inactive|Draining  
  -Identity Името на сървъра, -Component Посочва името на компонента, на който ще се променя състоянието, -Requester Посочва причината, -State Посочва състоянието, като Draining означава да отказва нови връзки, но да приключи настоящите.  
  Използва се когато искаш да правиш промени по компонента и това намалява възможността Managed Availability да промени състоянието на компонента. Ако го изключиш от Services е възможно Managed Availability да види проблем и да се опита да върне компонента в работещо състояние, като за това може да рестартира сървъра или да попадне в bug check.

Troubleshoot:

1. Изваждаме всички **HealthSetNames**, които не са в Healthy състояние и не са Disabled. Използваме, като заменяме сървъра, който ще гледаме.  
     
   **Get-HealthReport** -Server exchangeSERVER | where {$\_.alertvalue -ne "Healthy" –and $\_.AlertValue –ne "Disabled"}
2. За всеки **HealthSetName** показан от горната команда изпълняваме тази и заместваме в HealthSetName. Намираме всички **Monitoring** който са засекли отклонение.  
     
   Get-ServerHealth exchangeSERVER | ?{$\_.HealthSetName -eq "**HealthSetName**" -and $\_.alertvalue -ne "Healthy" -and $\_.AlertValue –ne "Disabled"}  
     
   Можем да вземем **Name** и **TargetResource**, които ще ни трябват после.
3. Изпълняваме командата, с всеки един от върнатите **Monitoring**, от предходната команда. В резултата ни интересува **SampleMask**, който показва **Probe**-а, пълнещ с информация мониторинга.  
     
   (Get-WinEvent -ComputerName exchangeSERVER -LogName Microsoft-Exchange-ActiveMonitoring/MonitorDefinition | % {[XML]$\_.toXml()}).event.userData.eventXml | ? {$\_.Name -eq "\***HubAvailabilityMonitor**\*"}
4. Изпълняваме командата, като заместваме този **HealthSetName**, в който се намира **Probe**-a и самия **Probe**. От предишната команда знаем името на **Probe**-a.  
    **Invoke-MonitoringProbe** **HealthSetName**\**ProbeName** -Server exchangeSERVER | Format-List  
     
   Ако **ResultType** е **Success** то грешката вече не е актуална. Приключваме.
5. Втора команда: След като знаем **SampleMask**, можем да извадим ВСИЧКИ логовете за **Probe**-a с командата, за която са ни необходими **Name** и **ResourceType**(пояснение: ако има само **Name**,а **ResourceType** е празен, то не попада в ресурсна група и се пише '**Name**’, а не '**Name/ResourceType**'.). С първата команда намираме **Name** и **ResourceType**, а с втората изваждаме логовете.   
   ###### Може да се пропусне понеже знаем TargetResource от стъпка 2  
   **Get-MonitoringItemIdentity** -Identity **HealthSetName** -server **ServerName** | select HealthSetName,Name,TargetResource,ItemType | ?{$\_.Name -eq "**Name**"} ###########

$errRecords = (Get-WinEvent -ComputerName **ServerName** -LogName Microsoft-Exchange-ActiveMonitoring/ProbeResult -FilterXPath "\*[UserData[EventXML[ResultName='**Name/ResourceType**'][ResultType='4']]]" | % {[XML]$\_.toXml()}).event.userData.eventXml

1. Тъй като горната команда вади всички логове на **Probe**-a, ние ще филтрираме само грешките със следната команда:  
     
   $errRecords | select -Property \*time,result\*,error\*,\*context  
     
   **Error**, **ExecutionContext**, **FailureContext** са тези, които гледаме.
2. За да разберем какви действия извършва Exchange, при срещане на проблема ни е необходимо да разберем името на **Responder**-a свързан с мониторинга.  
     
   (Get-WinEvent –ComputerName **ServerName** -LogName Microsoft-Exchange-ActiveMonitoring/ResponderDefinition | % {[xml]$\_.toXml()}).event.userData.eventXml | ? {$\_.AlertMask -like “\***MonitorName\***”} | fl Name,AlertMask,EscalationSubject,EscalationMessage,UpdateTime

Срещу полето **Name** е името на **Responder**-a. **EscalationSubject**, **EscalationMessage** какво действие предприема.