

URLGenie Management Pack Guide

Verision: 2.0.0.46

Microsoft Corporation

Published: 2020.11.19

Authored by: Tyson Paul | Microsoft Services | Customer Engineer

tyson.paul@microsoft.com

Copyright

This document is provided "as-is". Information and views expressed in this document, including URL and other Internet website references, may change without notice. You bear the risk of using it.

Some examples depicted herein are provided for illustration only and are fictitious.  No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes. You may modify this document for your internal, reference purposes.

© 2020 Microsoft Corporation. All rights reserved.

Microsoft, Windows, Windows Internet Explorer, Windows PowerShell, and Windows Server are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Table of Contents

[Introduction to the URL Genie Management Pack Guide 5](#_Toc528078191)

[Dependencies and Prerequisites 5](#_Toc528078192)

[Management packs this management pack depends on: 5](#_Toc528078193)

[Major Version History 6](#_Toc528078194)

[Supported Configurations 7](#_Toc528078195)

[Getting Started 8](#_Toc528078196)

[Importing the Management Pack 8](#_Toc528078197)

[Setup Overview: 8](#_Toc528078198)

[Security Considerations 8](#_Toc528078199)

[Classes 9](#_Toc528078200)

[Groups 9](#_Toc528078201)

[Customizing this Management Pack 11](#_Toc528078202)

[Create a New Management Pack for Customizations 11](#_Toc528078203)

[Discoveries 12](#_Toc528078204)

[HTTP Request Discoveries 12](#_Toc528078205)

[Optional Discoveries #2 and #3 12](#_Toc528078206)

[Parameters and Instance Properties 13](#_Toc528078207)

[Typical Properties 13](#_Toc528078208)

[Non-Standard Parameters 21](#_Toc528078209)

[Tasks 23](#_Toc528078210)

[URLGenie EnableWatcherNode Task 23](#_Toc528078211)

[URLGenie Generate Config From List 23](#_Toc528078212)

[URLGenie DisableWatcherNode Task 25](#_Toc528078213)

[URLGenie PingServer 25](#_Toc528078214)

[URLGenie Get Certificate Info 25](#_Toc528078215)

[URLGenie TestPath 25](#_Toc528078216)

[Rules 25](#_Toc528078217)

[URLGenie Scripted Request ResponseTime Collection Scripted PerfRule 26](#_Toc528078218)

[Monitors 27](#_Toc528078219)

[URLGenie IE Login Scripted Monitor 27](#_Toc528078220)

[URLGenie Scripted Monitor 29](#_Toc528078221)

[URLGenie CA Untrusted Monitor 30](#_Toc528078222)

[URLGenie Certificate Expired Monitor 30](#_Toc528078223)

[URLGenie Content Monitor 30](#_Toc528078224)

[URLGenie DNS Resolution Failure Monitor 30](#_Toc528078225)

[URLGenie Error Code Monitor 31](#_Toc528078226)

[URLGenie Reachable Monitor 31](#_Toc528078227)

[URLGenie Response Time Monitor 31](#_Toc528078228)

[URLGenie Status Code Monitor 31](#_Toc528078229)

[URLGenie Certificate Expires Soon Monitor 31](#_Toc528078230)

[The Configuration File 32](#_Toc528078231)

[Configuration File Examples 32](#_Toc528078232)

[Example 1: Basic Template 32](#_Toc528078233)

[Example 2: Sample File 34](#_Toc528078234)

[Tutorials 37](#_Toc528078235)

[Activate a Watcher Node 37](#_Toc528078236)

[Create a Configuration File 43](#_Toc528078237)

[1) Use the task: “URLGenie Generate Config From List” 43](#_Toc528078238)

[2) Manually create the file from the blank template example provided in this guide. 46](#_Toc528078239)

[Add a URL Instance to an Existing Config File 47](#_Toc528078240)

[URL Instance Discovery 48](#_Toc528078241)

[Enable Monitoring a URL Instance with a Client Certificate 50](#_Toc528078242)

[How the scripted monitor works. 50](#_Toc528078243)

[Step by Step 50](#_Toc528078244)

[Variable Descriptions 52](#_Toc528078245)

[Monitor a Login Page (Forms Based Login) 56](#_Toc528078246)

[1) Create the URL Instance 57](#_Toc528078247)

[2) Create RunAs Account for the Scripted Internet Explorer Session 59](#_Toc528078248)

[3) Create the Credentials for the Target Web Page Login Form(s) 61](#_Toc528078249)

[4) Configure Security Profiles 63](#_Toc528078250)

[Configure the Watcher Node 65](#_Toc528078251)

[Override the Monitor with Login Parameters 77](#_Toc528078252)

[Use NTLM Authentication 92](#_Toc528078253)

[Load Testing 97](#_Toc528078254)

[Memory Errors 105](#_Toc528078255)

[Troubleshooting 109](#_Toc528078256)

**Revision History**

|  |  |
| --- | --- |
| Release Date | Changes |
| **05/05/2020** | Fixed a few typos. Updated screenshot(s). |
| 10/23/2018 | Updated to reflect MP revision |
| May 2015 | Original release of this guide |

|  |  |
| --- | --- |
| Management Packs | Description |
| **URLGenie.mp** | Main management pack |
| URLGenie.Optional.Enable.Additional.Performance.Collections.mp | Contains overrides to enable additional performance collections for standard URL types. For convenience only. |

# Introduction to the URL Genie Management Pack Guide

The URL Genie Management Pack provides a fast and easy way to implement monitoring for a large numbers of URL instances from only a few up to many thousands! In addition, there are some special features which allow monitoring with client certificates and forms-based authentication.

The URL instances and their respective monitoring criteria get instantiated on any number of “watcher” nodes from one or more XML configuration files. With a single agent task, any managed Windows computer can be activated as a watcher node. The task configures the watcher node with a path to where it should look for configuration files. There can exist any number of configuration files, each with any number of requests defined within. Typically, the configuration files will be centrally located in a single shared network folder. A decent place for the shared configuration folder is on management server or the DataWarehouse server with all watcher nodes configured with the same shared folder path. This is the most simple and scalable configuration.

There are standard monitors which target the standard http and https class types as well as special monitors targeting the special types. Each individual monitor will alert with specific context information. This is significantly different than the Operations Manager standard Web Availability or Synthetic Transaction monitoring which will only alert on the health rollup and contain no specific or useful alert context information.

The standard monitors all support the various types of http authentication: None, Basic, NTLM, Digest, and Negotiate. In addition, there are special monitor types which can be enabled for URLs that require a client certificate or even for web sites that use forms-based authentication, a first for SCOM (to my knowledge)!

## 

## Dependencies and Prerequisites

This management pack requires a minimum version of Operations Manager 2007 R2 or later (to my knowledge). Import will fail in previous versions of Operations Manager 2007.

### Management packs this management pack depends on:

|  |  |
| --- | --- |
| Microsoft.SystemCenter.DataWarehouse.Library | 6.1.7221.0 |
| System.Health.Library | 6.1.7221.0 |
| Microsoft.SystemCenter.InstanceGroup.Library | 6.1.7221.0 |
| System.Performance.Library | 6.1.7221.0 |
| Microsoft.SystemCenter.Library | 6.1.7221.0 |
| System.Library | 7.5.8501.0 |
| Microsoft.Windows.Library | 6.1.7221.0 |
| Microsoft.SystemCenter.WebApplication.Library | 7.0.8560.0 |

# Major Version History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Changes** | **Notes** |
| 2020.11.19 | 2.0.0.46 | * I started noting my dates differently in documentation. I like this format better. * I updated the main Alert view to include all except Closed(255) * I added a link to the Get-CertInfo task to all of the https unit monitors knowledge. | YYYY/MM/DD |
| 05/05/2020 | 2.0.0.44 | * Added DisplayName and Description to MP properties. * MP Guide minor updates. * Improved script 'EnableURLGenieWatcher.ps1' with more logging, better logic and output for user. * Add missing overrides for the additional performance collection rules. * Improved 'ScriptedIELoginContentMatch.ps1'. Added Trim() to some vars that decided to cause problems. |  |
| 10/22/2019 | 2.0.0.39 | Updated alert descriptions for the following alerts: HTTP Request Error: Certificate Authority Error HTTP Request Error: Certificate Expired Error HTTP Request Error: Certificate CN Invalid | One or more alerts had problems displaying variables in the description. |
| 09/27/2019 | 2.0.0.38 | * Added DNSError criteria to Reachable monitor. Previously Reachable monitor would remain green even if DNS lookup failed and site was not retrieved successfully. Now the Reachable monitor type Healthy state requires successful DNS lookup. * Updated "ms" on numerous performance collection rules Counter labels to "sec" to indicate Seconds metric correctly. * Added perf rule to collect StatusCode metric for Special and Standard target types. | Performance signatures for the previous Counter name will exist in Console performance views until the data is groomed out of the OperationsManager database; usually this grooming period is 7 days or less. |
| 12/10/2018 | 2.0.0.36 | Fixed the clientcert group population. |  |
| 10/23/2018 | 2.0.0.30 | I Updated the versioning scheme of this MP.   * Updated ScriptedIELoginContentMatch.ps1. Improved logic to authenticate on sites that have separate username and login pages. Sites like login.live.com use a two page auth process (1:Username, 2:Password). Improved logging. Added URL sequence (ma\* 2 URLs; 1:login page, 2:target page to match content)   \* Change Service Model so "special" URLs don't get targeted with ordinary monitors.   * Remove original HTTPS discovery; all 3 http discoveries can be baked into one discovery script. * Make discovery management group aware for dual homed scenarios. * Add DaysToExpiry monitor * Add TLS support for URLGrabScripted Monitor type * Make sure rules exist for special HttpS. * Make sure tasks (diagnostic and agent) exist for special HttpS. * Make sure monitors exist for special HttpS. * Add knowledge to monitors (example: http://mpwiki.viacode.com/default.aspx?g=posts&t=143294) * Updated Diagnostic task timeouts * Remove 'ClassID' from URLGenie.ReadConfig.Composite.DS * Add TestPath monitor with Discovery RunAs profile. * ID="URLGenie.TestPath.PA" RunAs="URLGenie.Discovery.Profile" * Added CollectResponseBody parameter to applicable workflows. Default: Never (Never, Always, OnContentMatchCriteria) [Microsoft.SystemCenter.WebApplication.Library.MP] * Removed SyncTime for scheduled discoveries * Moved URLGenie Aggregate monitor to standard Availability * Improved GenerateConfigFile task * Fixed RunAs for WatcherNode ConfigFile Path Test Alert Rule * Updated WriteToEventLog parameter for all applicable workflows to boolean (true/false) instead of integer * Added Headers parameter to Scripted monitor type * Removed TestPath alert. Added TestPath monitor to Watcher Node. * Added Monitor: URLGenie Certificate Invalid Monitor * Improved TestPath DS logging * Added aggregate monitor for URL unit monitors * Renamed and improved diagnostic task: TestPort * Enabled Scripted Monitor for CertThumbprint Group * URL Discovery: Improved certthumbprint parsing and validation checking; Added logic to prevent special chars from being hidden in CertThumbprint * Added additional performance collection rules for standard URLs. Disabled by default. * Improved GenerateConfigFile task; added additional parameters * Modified references to support SCOM 2012 SP1 * Add ContentMatchRegExOperator parameter with corresponding function to conduct the correct match sequence/logic * Improve 'Test-Path' task to include list of files in target directory * Expanded ContentMatch monitor type with ability to check ResponseHeaders * Expanded Scripted monitor type to match content from ResponseHeaders * Modified target class for EnableWatcherNode and TestPath tasks. * Added back the MaxLength on the http class properties. These got lost upon converting MP project to a Visual Studio solution. * Fixed this param type in Autooveride probe. (ID="URLGenie.URLProbe.AutoOverride.PA") Should be Integer: * <OverrideableParameter ID="ResponseTimeout" Selector="$Config/ResponseTimeout$" ParameterType="string" /> |  |
| 2017.03.16 | 2017.3.16.1330 | * Added RetryCount into the CD module for the applicable monitor types. Now Retry works correctly as you would expect it to! One giant step for man.. * Fixed the hardcoded headers parameters in numerous places to be dynamic from instance property, like they were originally supposed to be. * Improved the URLGrabScriptedIELogin monitor type significantly. Now includes "logout" override parameter. Added ability to specify submit button "type". * Consolidated the "URLGenie Client Certificate Scripted Monitor" into the "URLGenie Scripted Monitor". Fixed script timeout parameter. Added the ability to monitor URL sequences, submit multiple URLs with multiple criteria. This document has not been fully updated to reflect these changes. I’ll get around to it. For now I simply used strikeout for any references to the ClientCert workflows. The URLGrabScripted type works the same way as the ClientCert so most of the documentation is still applicable. |  |
| 2015.05.08 | 2015.5.8.1706 |  | This is the first version released to community. |

# Supported Configurations

|  |  |
| --- | --- |
| **Component** | **Supported Versions** |
| System Center Operations Manager | 2019, 2016, 2012 R2, 2012 Sp1 |
| Powershell | 3.0 or higher |
| Agent Managed Operating System | 2008, 2008R2, 2012, 2012R2, 2016, 2019, Win7, Win10 |

# Getting Started

## Importing the Management Pack

Other than verifying the management pack dependencies are installed in the management group, there are no other prerequisites before importing.

## Setup Overview:

1) **Import Management Pack**

2) **Figure out where you want to store your configuration files because you will need this information to enable a Watcher node.** You can store the config file(s) locally on each watcher node but the suggested method is to create a shared folder on your management server, data warehouse server, or other file server where your watchers nodes will be able to access them. This way any URL instances that you define in your configuration files can be discovered by any/all watcher nodes depending on how you configure the <watchers> tags. See [Parameters and Instance Properties](#_Parameters_and_Instance) section below.

3) **Activate one or more Watcher nodes.** See [URLGenie EnableWatcherNode](#_URLGenie_EnableWatcherNode_Task) task info below. Make note of the required overrides for this task.

4) **Create one or more configuration files.** See [The Configuration File](#_The_Configuration_File) section of this document. Once you create the configuration files, the discoveries will run on the activated Watcher node(s) and create the URL instances which will become automatically monitored within a few minutes after the group calculation workflows run on your management servers.

See [URLGenie.URL.SPECIAL.Group](#_Groups) for exceptions.

Note: This management pack has been tested thoroughly. However, I cannot guarantee the success of monitoring URL instances that contain complex or unusual characters or strings or overall lengths larger than the URL property [4096]. I STRONGLY suggest that you thoroughly test your URL monitoring in a dev/test environment first and use the Workflow Analyzer and OpsMan event log to watch for failed workflows on your watcher node(s). Use this management pack at your own risk and always test extensively before implementing in a production environment.

# Security Considerations

This URLGenie Management Pack uses the Discovery agent action account to perform URL Instance discoveries. This action account must have the necessary permissions to read the configuration files located in the configured path on the Watcher node(s). See the [URL Instance Discovery](#_URL_Instance_Discovery) tutorial below.

# Classes

The following table describes the available classes:

|  |  |
| --- | --- |
| **Available Groups** | **Description** |
| URLGenie.WatcherNode | Any Windows Computer that has been “activated” with the [EnableWatcherNode task](#_URLGenie_EnableWatcherNode_Task) |
| URLGenie.HttpRequest.Seed | Seed class |
| URLGenie.HttpRequest.Standard | Ordinary URL instance |
| URLGenie.Http**s**Request.Standard | Ordinary URL instance (https) |
| URLGenie.HttpRequest.Special | Special URL instance for scripted monitoring |
| URLGenie.Http**s**Request.Special | Special URL instance for scripted monitoring (https) |

# Groups

The reason that most of these groups exist is to override the monitors that target the members. The reason that certain parameters need to be overridden for these groups is because of limitations on what types of variables can be passed down to the datasource that runs the Microsoft.SystemCenter.WebApplication.UrlProbe module. For example, the URLProbe expects, or rather requires, a verb "type" of GET, POST, or HEAD and will not accept any other variable, type, or text outside of those three specific strings. This makes it impossible to pass the target Verb property down through the stack as a variable into the module. Ideally one could use a target property like this: $Target/Property[Type="URLGenie.HttpRequest"]/Verb$ but this results in a schema error so the verb MUST be hardcoded as one of the three allowed strings, in this case to a default of "GET" (which is the most common verb). Then, the individual instances are overridden by built-in overrides which target the groups that contain those instances.

Groups will auto-populate based on certain properties of the instances.

|  |  |
| --- | --- |
| **Available Groups** | **Description** |
| URLGenie.URL.AuthBasic.Group | This group will auto-populate with any instances which have an authorization type of "Basic". The authenticationscheme parameter will be overridden automatically for the workflows. |
| URLGenie.URL.AuthDigest.Group | This group will auto-populate with any instances which have an authorization type of "Digest". The authenticationscheme parameter will be overridden automatically for the workflows. |
| URLGenie.URL.AuthNegotiate.Group | This group will auto-populate with any instances which have an authorization type of "Negotiate". The authenticationscheme parameter will be overridden automatically for the workflows. |
| URLGenie.URL.AuthNone.Group | This group will auto-populate with any instances which have an authorization type of "None". The authenticationscheme parameter will be overridden automatically for the workflows. |
| URLGenie.URL.AuthNTLM.Group | This group will auto-populate with any instances which have an authorization type of "NTLM". The authenticationscheme parameter will be overridden automatically for the workflows. |
| URLGenie.URL.ClientCert.Group | This group will auto-populate with any instance that has the CertThumbprint property populated. This is useful for URLs which require a client certificate. Normal monitoring with the standard URLProbe datasource module doesn't apply to these types of instances. The members of this group will be monitored automatically by the *URLGenie Scripted Monitor* via a datasource which uses a Powershell script. |
| URLGenie.URL.ProxyAuthBasic.Group | This group will auto-populate with any instances which have a proxyauthenticationscheme type of "Basic". The proxyauthenticationscheme parameter will be overridden automatically for the workflows. |
| URLGenie.URL.ProxyAuthDigest.Group | This group will auto-populate with any instances which have a proxyauthenticationscheme type of "Digest". The proxyauthenticationscheme parameter will be overridden automatically for the workflows. |
| URLGenie.URL.ProxyAuthNegotiate.Group | This group will auto-populate with any instances which have a proxyauthenticationscheme type of "Negotiate". The proxyauthenticationscheme parameter will be overridden automatically for the workflows. |
| URLGenie.URL.ProxyAuthNone.Group | This group will auto-populate with any instances which have a proxyauthenticationscheme type of "None". The proxyauthenticationscheme parameter will be overridden automatically for the workflows. |
| URLGenie.URL.ProxyAuthNTLM.Group | This group will auto-populate with any instances which have a proxyauthenticationscheme type of "NTLM". The proxyauthenticationscheme parameter will be overridden automatically for the workflows. |
| URLGenie.URL.SPECIAL.Group | Any URL which contains a caret ^ will populate into the SPECIAL group. You can leverage this group creatively in the following ways:  If you have an instance with two URLs (separated by the caret ^), you can override/enable the *URLGenie Scripted Monitor* and/or performance rule for this SPECIAL group or just the instance. Read more about the [URLGenie Scripted Monitor](#_URLGenie.URLGrabScripted.Monitor) and how to use it below.  The only good reason why a URL would contain two URLs (which would require the caret ^ to separate them), would be to test a two-page sequence for ContentMatch and ResponseTime. Test the first page, if successful, proceed to test the second. Note: the content match field must contain two strings separated by the caret ^ as well. Read more about the [URLGenie Scripted Monitor](#_URLGenie.URLGrabScripted.Monitor) and how to use it below.  If you have an instance that you want to be effectively "disabled" by default, you could add the caret ^ to the end or beginning of the URL.  Example, if you have an instance that should only be monitored with the URLGenie IE Login Scripted Monitor only and no other monitors, then you could simply add a caret ^ to the URL (beginning or end) in the config file like this:  <url><![CDATA[www.MyLoginPage.com^]]></url>  This will ensure that the instance will be unmonitored unless specifically enabled with an override. |
| URLGenie.URL.VerbGET.Group | This group will auto-populate with any URL instances which have a verb type of "GET". The verb parameter will be overridden automatically for the workflows. |
| URLGenie.URL.VerbHEAD.Group | This group will auto-populate with any URL instances which have a verb type of "HEAD". The Verb parameter will be overridden automatically for the workflows. |
| URLGenie.URL.VerbPOST.Group | This group will auto-populate with any URL instances which have a verb type of "POST". The Verb parameter will be overridden automatically for the workflows. |

# Customizing this Management Pack

## Create a New Management Pack for Customizations

Most vendor management packs are sealed so that you cannot change any of the original settings in the management pack file. However, you can create customizations, such as overrides or new monitoring objects, and save them to a different management pack. By default, Operations Manager 2007 saves all customizations to the default management pack. As a best practice, you should instead create a separate management pack for each sealed management pack you want to customize.

Creating a new management pack for storing overrides has the following advantages:

• It simplifies the process of exporting customizations that were created in your test and pre-production environments to your production environment. For example, instead of exporting a default management pack that contains customizations from multiple management packs, you can export just the management pack that contains customizations of a single management pack.

• It allows you to delete the original management pack without first needing to delete the default management pack. A management pack that contains customizations is dependent on the original management pack. This dependency requires you to delete the management pack with customizations before you can delete the original management pack. If all of your customizations are saved to the default management pack, you must delete the default management pack before you can delete an original management pack.

• It is easier to track and update customizations to individual management packs.

For more information about sealed and unsealed management packs, see Management Pack Formats (<http://go.microsoft.com/fwlink/?LinkId=108355>). For more information about management pack customizations and the default management pack, see About Management Packs in Operations Manager 2007 (<http://go.microsoft.com/fwlink/?LinkId=108356>).

# Discoveries

## HTTP Request Discoveries

The request discoveries are created in such a way so that they can handle very large quantities of instances. They can be configured to each consume X percentage of the total number of instances that exist in the configuration files in a designated folder configured for the watcher node.

By default, HTTP Request Discovery #1 is enabled and configured to consume 100 percent of instances detected. This discovery has been successfully tested with up to 6000 URL instances (with minimal configuration data such as 'body' and 'description' fields). It's unlikely that your watcher node(s) will be able to successfully handle monitoring more URL instances than the discovery workflows can discover.

## Optional Discoveries #2 and #3

NOTE: THESE SHOULD ONLY BE ENABLED AND CONFIGURED FOR LARGE DISCOVERIES. It would be rare and unusual for the average user to need to configure these additional discoveries, but the functionality still exists if needed.

Example:

Let’s say that you have a total of 6000 URLs instances to be monitored with very verbose <description> fields and very lengthy SOAP bodies in the <body> fields. You might have designated 1 very high-end watcher node. You could split the URL discoveries as follows:

URLs 1-6000: <watchers>Servername**1**</watchers>

You would then override the 3 discoveries as follows:

"LowerDiscoveryPercentage" and "UpperDiscoveryPercentage" should coincide with all other "URLGenie.HttpRequest.Properties.DiscoveryX" discoveries. Ranges must overlap as shown below and should encompass 0 to 100.

Example (with all 3 discoveries enabled):

|  |  |  |  |
| --- | --- | --- | --- |
| **Override Target** | **Discovery** | **Values** | **Enabled** |
| Servername**1** | URLGenie.HttpRequest.Properties.Discovery**1** | Lower=0, Upper=33 | True |
| Servername**1** | URLGenie.HttpRequest.Properties.Discovery**2** | Lower=33, Upper=66 | **True** |
| Servername**1** | URLGenie.HttpRequest.Properties.Discovery**3** | Lower=66, Upper=100 | **True** |

## Parameters and Instance Properties

URL instances are created by defining the criteria in the .xml config file.

Most of the available parameters have a default value that gets assigned during discovery only if left blank in the config file (but there are some that do not get a default value: i.e. AuthUser and AuthPass).

### Typical Properties

These are common properties that URL instances can have/use.

AuthenticationScheme [string] <"**None**" | "Basic" | "Digest" | "NTLM" | "Negotiate">

Default: "None"

Description: If authentication method other than None is used, you will have to specify a username (AuthUser) and also configure a Basic account type in the Console, then assign that account to the object instance with the URLGenie Standard Account security profile after the instance has been discovered.

AuthPass [Basic Account Type, configured in Console]

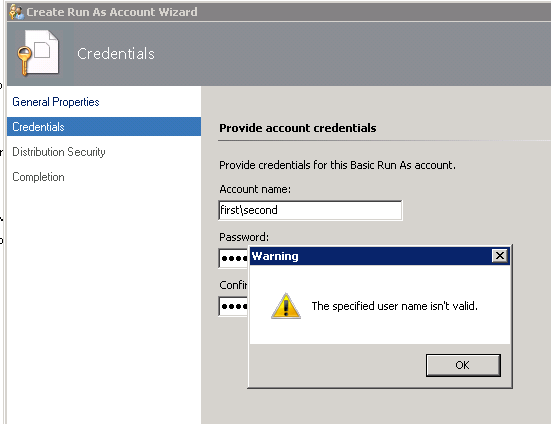
Default: <blank>

Description: THIS IS NOT CONFIGURED IN THE .XML CONFIG FILE. However, this parameter is overrideable for most monitors. If an authentication method other than None is used for the *authenticationscheme* property, you will have to specify a username for *authuser* and also configure a Basic account type in the Console, then assign that account to the object instance with the *URLGenie Standard Account* profile. Only the password from this account is used.

AuthUser [string]

Default: <blank>

Description: Username for use with authentication. The reason that the username is configurable within the .xml config file is to expand the flexibility of this string. Example, I came across a business unit that requested that I test their application with Basic authentication but the username was in this format: 'groupname\username'. You simply cannot create a Basic account in the SCOM console with a username that contains a slash. Example below:



See tutorials below for NTLM authentication example

Body [string]

Default: <blank>

Description: Optional. Usually this is an XML SOAP structure with a verb type of POST.

Example:

------ BEGIN EXAMPLE BELOW ----------

<?xml version="1.0"?>

<soap:Envelope

xmlns:soap="http://www.w3.org/2001/12/soap-envelope"

soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">

<m:GetStockPrice>

<m:StockName>IBM</m:StockName>

</m:GetStockPrice>

</soap:Body>

</soap:Envelope>

--------- END EXAMPLE ABOVE ----------

ContentMatch [string]

Default: "."

Description: Text to match in the response body. This is a simple match and is not case-sensitive. (See *ContentMatchRegExOperator* below.)

Examples:

"cake"

"catfish"

"headed out the door"

"403 - Forbidden: Access is denied"

NOTE: It is possible to have two content match strings but only for use with an instance that has two URLs and only for the [URLGenie Scripted Monitor](#_URLGenie.URLGrabScripted.Monitor). Strings must be separated by the caret ^. See "URL" parameter definition below for more details.

Examples:

website^taste the rainbow

billy joel^piano man

hp^printer

ContentMatchRegExOperator [RegExCompareType]

Default: "DoesNotContainSubstring"

Description: If the criteria evaluates to true when compared against the ContentMatch value, an error condition will result for the [Content](#_URLGenie.AutoOverride.Content.Monit) monitor. This is not configurable as a property in the configuration file but you can override it for most monitor types.

* ContainsSubstring
* MatchesWildcard
* MatchesRegularExpression
* MatchesMOM2005RegularExpression
* MatchesMOM2005BooleanRegularExpression
* **DoesNotContainSubstring**
* DoesNotMatchWildcard
* DoesNotMatchRegularExpression
* DoesNotMatchMOM2005RegularExpression
* DoesNotMatchMOM2005BooleanRegularExpression

Reference: <https://systemcenter.wiki/?GetElement=System.ExpressionEvaluatorSchema&Type=SchemaType&ManagementPack=System.Library&Version=6.1.7221.0>

ContentType [string]

Default: "text/html; charset=UTF-8"

Description: The MIME type of the request body (used with POST and PUT requests)

ErrorTime [double]

Default: 50

Description: Seconds after which the [ResponseTime](#_URLGenie.AutoOverride.ResponseTime.) monitor will raise a Critical alert.  
Note: For Scripted monitor types this represents the Total time to complete all URL tests (for a multi-URL sequence).

Note: These header fields cannot be empty or they will break the URLProbe module. Therefore, if they are left blank in the configuration file, they get populated automatically with basic default data during discovery. The default values are basically inert and serve very little purpose other than being placeholders; they are available should you need to customize them with meaningful or necessary values for your web application.

Header1Name [string]

Default: "Accept-Language"

Description: Optional request header data. See http://en.wikipedia.org/wiki/List\_of\_HTTP\_header\_fields

Header1Value [string]

Default: "en-US"

Description: Optional request header data. See <http://en.wikipedia.org/wiki/List_of_HTTP_header_fields>

Header2Name [string]

Default: "Accept-Charset"

Description: Optional request header data. See <http://en.wikipedia.org/wiki/List_of_HTTP_header_fields>

Header2Value [string]

Default: "utf-8"

Description: Optional request header data. See <http://en.wikipedia.org/wiki/List_of_HTTP_header_fields>

Header3Name [string]

Default: "From"

Description: Optional request header data. See <http://en.wikipedia.org/wiki/List_of_HTTP_header_fields>

Header3Value [string]

Default: "SCOM@yourdomain.com"

Description: Optional request header data. See <http://en.wikipedia.org/wiki/List_of_HTTP_header_fields>

Index [string]

Default: <discovery order>

Description: Technically this property is a string. However, the configuration element should be populated with an integer; a simple whole number. Useful for sorting URLs in an object state view. As of version 2012 R2 this must be a string type to be able to sort by this column in the State view. Therefore, the instance property type is 'string' but this must be a positive integer in this range: [0-99999]

Example:

5

Interval [integer]

Default: 300

Description: Seconds. How often to run the workflow/monitor.

ResponseTimeout [integer]

Default: 60

Description: Seconds after which the workflow will give up and [ResponseTime](#_URLGenie.AutoOverride.ResponseTime.) monitor will raise a Critical alert.   
Note: for the Scripted monitor types the following applies: Specifies how long the request can be pending before it times out. Enter a value in seconds. The default value, 0, specifies an indefinite time-out. This applies for each URL (if using a multi-URL sequence).

RetryCount [integer]

Default: 0

Description: Number of times to retry the monitor before raising an alert. If a specific unhealthy condition has been detected X times in a row, only then will the related monitor change state. The retries will occur at the next schedule *Interval*.

StatusThreshold [double]

Default: 399

Description: If response header status code of the final page (follow redirects is enabled by default) exceeds this number, a critical alert will be raised by the [StatusCode](#_URLGenie.AutoOverride.StatusCode.Mo) monitor.

URL [string, URL]

Default: <blank>

Description: REQUIRED. Must start with http:// or https://. Theoretically you can use an unlimited number URLs in this field only when using the [URLGenie Scripted Monitor](#_URLGenie.URLGrabScripted.Monitor). see below

Example: http://www.myfavoritedomain.com

If you want to configure a URL sequence check you must separate the URLs with the caret ^ character.

Example: https://www.salespage.com/sku?=101**^**https://www.product.com/sku?=104**^**http://myapp.mysite.com

UserAgent [string]

Default: 'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT; Windows NT 6.1; en-US)'

Description: See section 14.43 located here: http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

The User-Agent request-header field contains information about the user agent originating the request. This is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations. User agents SHOULD include this field with requests. The field can contain multiple product tokens (section 3.8) and comments identifying the agent and any subproducts which form a significant part of the user agent. By convention, the product tokens are listed in order of their significance for identifying the application.

Format:

User-Agent = "User-Agent" ":" 1( product | comment )

Examples:

User-Agent: CERN-LineMode/2.15 libwww/2.17b3

User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT; Windows NT 6.1; en-US)

Verb [verb type] <"GET" | "POST" | "HEAD">

Default: GET

Description: For detailed information see here: <http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.2>

WarningTime [double]

Default: <whatever ErrorTime has been set to>

Description: Seconds after which the [ResponseTime](#_URLGenie.AutoOverride.ResponseTime.) monitor will raise a Warning alert. If same as ErrorTime, then only a Critical alert will result.   
Note: For Scripted monitor types this represents the Total time to complete all URL tests (for a multi-URL sequence).

Watchers [string]

Default: <blank>

Description: This will define which watcher nodes will "watch" the particular instance. You can use any part of the FQDN of a server. You can separate multiple strings with a comma.

Examples:

Your watcher node computer names:

1) dc2scomms02.yourdomain.local

2) dc2scomms05.yourdomain.local

3) Myserverms04.mydomain.local

Watcher value/mask Effective watcher node(s)

<watchers>***dc2***</watchers> 1,2

<watchers>***yourdo***</watchers> 1,2

<watchers>***ms0***</watchers> 1,2,3

<watchers>***myserver***</watchers> 3

<watchers>***05,04***</watchers> 2,3

<watchers>***serverms,05***</watchers> 2,3

A screenshot of a cell phone

Description automatically generated

Example of how the watchers value affects where the URL instance gets discovered.

### Non-Standard Parameters

CaseSensitive [int] < true | false >

Default: false

Description: Used with the Scripted URL monitors only. (true = use case sensitive matching.)

CertThumbprint [string]

Default: <blank>

Description: The thumbprint of the client certificate to use in the request. The certificate must be installed on any watcher nodes that run this monitor workflow. The certificate should be installed into the "LocalMachine\My" location. You can view the certificates installed in this location (on the watcher node) by running this command at a Powershell prompt: **gci Cert:\LocalMachine\my**

If you don't see the certificate thumbprint in the output, then you don't have the cert installed in the correct (default) cert store and the workflow will not function correctly for the URL object instance, in which case you will see errors in the OpsMan log. This property is only applicable for the [URLGenie Scripted Monitor](#_URLGenie.URLGrabScripted.Monitor) and [URLGenie.URLGrabScripted.ResponseTimeCollection.PerfRule](#_URLGenie.URLGrabScripted.ResponseTi) workflows.

Thumbprint example: "F945261E79C44E1C5690000291DCBA79C4136712"

CertDir [string]

Default: "LocalMachine\My"

Description: The local certificate store where the "client cert" exists on the Watcher Node. Override optional for the [~~URLGenie.URLGrabScripted.ClientCert.Monitor~~](#_URLGenie.URLGrabScripted.ClientCert) and [~~URLGenie.URLGrabScripted.ClientCert.ResponseTimeCollection.PerfRule~~](#_URLGenie.URLGrabScripted.ClientCert_1) workflows.

CheckErrorData [boolean] < true | false >

Default: false

Description: This setting will also enable the checking of the server response (error data) for the content match string. In some circumstances an error is the expected response and typically in that situation there is no page content to check for a content match. This is only applicable to the ClientCert group members, or those instances with a Certthumbprint value.

If using override for CheckErrorData, be sure to override for the Monitor AND Performance Rule that apply to the Scripted Client Cert objects. This property is only applicable for the [~~URLGenie.URLGrabScripted.ClientCert.Monitor~~](#_URLGenie.URLGrabScripted.ClientCert) and [~~URLGenie.URLGrabScripted.ClientCert.ResponseTimeCollection.PerfRule~~](#_URLGenie.URLGrabScripted.ClientCert_1) workflows.

Proxy [string, URL]

Default: <blank>

Description: Optional. Address of your proxy server.

ProxyAuthenticationScheme [string] <"None" | "Basic" | "Digest" | "NTLM" | "Negotiate">

Default: "None"

Description: If authentication method other than *None* is used, you will have to specify a username (see *ProxyUser*) and also configure a Basic account type in the Console, then assign that account to the object instance with the "URLGenie Proxy Account".

ProxyPass [Basic Account Type, configured in the Console]

Default:

Description: THIS IS **NOT** CONFIGURED IN THE .XML CONFIG FILE. However, this parameter is overrideable for most monitors. If *ProxyAuthenticationScheme* authentication method other than *None* is used, you will have to specify a username (see *ProxyUser*) and also configure a Basic account type in the Console, then assign that account to the object instance with the URLGenie Proxy Account. Only the password from this credential object is used.

ProxyUser [string]

Default: <blank>

Description: Username for use with Proxy authentication. Note: the password must be configured in the Console with a Basic Account type.

TrustAllRemoteSSLCertificates [boolean] < true | false >

Default: true

Description: If the certificate of the URL (for secure addresses: https ) is expired or otherwise untrusted you can choose to ignore the fault and proceed to trust it. Override optional for the *URLGenie Scripted Monitor* and *URLGenie Scripted Request ResponseTime Collection Scripted PerfRule* workflows.

VerbosityLimit [integer]

Default: 1024

Determines how much of the web page content to return in the workflow property bag and also output to the event log. Web pages can be very large, large enough to choke the probe module.

WriteToEventLog [boolean] < true | false >

Default: false

Description: Useful for debugging and troubleshooting. Only present on Scripted monitors/workflows. Will write useful workflow data to OpsMan event log.

# Tasks

## URLGenie EnableWatcherNode Task

Description: Will turn any agent monitored Windows Computer machine into a Watcher Node. In the Monitoring workspace, Windows Computer state view, you must select a Windows Computer object in order for the task to appear in the Tasks pane. This task will create a simple registry entry on the target server. This registry entry is used for Watcher Node discovery.

NOTE: You MUST override this task; specify a path for the ConfigFilesPath parameter. This is the path where the targeted watcher node will pick up the .xml configuration files in order to discover URL instances.

This path is a directory, not a file. It is in this directory where you will store any number of .xml configuration files that contain the properties of the URL instances. (See the [Configuration Files Examples](#_Configuration_File_Examples) below.) This path can be local or a shared network folder. The easiest method is to create a shared folder on your management server, data warehouse server, or other file server where you can store your configuration files, that way any URL instances that you define in your configuration files can be watched by any watcher node on your network. This method of centrally locating the configuration files makes this a very powerful, scalable monitoring solution for a large number of URL instances.

Note: Any configuration files in the designated path must conform to a specific name format or else they will be ignored. See configuration files section for more detail.

Examples:

C:\URLGenie\ConfigFiles

\\YourDWServer.domain.com\SCOM\_DO\_NOT\_TOUCH\URLGenie

## URLGenie Generate Config From List

Will read from files that contain a list of plain URLs and generate basic configuration files for those URL instances.

Task override settings example:

|  |  |  |
| --- | --- | --- |
| **Property** | **Value** | **Description** |
| FileMask | .txt | Task will scan any .txt files present |
| InputPath | C:\MYFOLDER\URLFILES | Folder to search for files that match the FileMask |
| OutputPath | C:\MyConfigFiles | Folder into which the configuration files will be output. Task will output one configuration file for each .txt file found. Output file names will be:  “Requests\_<originalfilename>.xml” |
| IndexBeginAt | 1 | Starting index number for URLs to be created |
| ErrorTime | 30 | (See [Parameters and Instance Properties](#_Parameters_and_Instance) above) |
| WarningTime | 15 | (See [Parameters and Instance Properties](#_Parameters_and_Instance) above) |
| Interval | 300 | (See [Parameters and Instance Properties](#_Parameters_and_Instance) above) |
| GroupID | DevStore | (See [Parameters and Instance Properties](#_Parameters_and_Instance) above) |
| ResponseTimeout | 60 | (See [Parameters and Instance Properties](#_Parameters_and_Instance) above) |
| StatusThreshold | 399 | (See [Parameters and Instance Properties](#_Parameters_and_Instance) above) |
| TimeoutSeconds | 120 | Seconds after which this agent task should quit. |
| Watchers | . | (See [Parameters and Instance Properties](#_Parameters_and_Instance) above) |
| WriteToEventLog | True | This will enable verbose output to the OperationsManager event log for the agent task. |

The above example task settings will execute the task on whichever servers are selected in the Windows Computers state view, to read the contents of any .txt files found in the InputPath path and generate a corresponding output file for each input file to the OutputPath path. Each output file will contain corresponding configuration information for each URL address line item in the input file.

Sample input file: **myurls.txt**

------------ BEGIN SAMPLE INPUT FILE <THIS LINE NOT INCLUDED> ------------

**www.bing.com  
http://www.cypresspos.com/products/  
microsoft.com  
https://chivecharities.org/**

------------ END SAMPLE INPUT FILE <THIS LINE NOT INCLUDED> ------------

Also see [sample configuration files](#_Configuration_File_Examples) below.

## URLGenie DisableWatcherNode Task

Deactivate any active watcher node.

## URLGenie PingServer

Description: Will use Powershell cmdlet 'Test-Connection' against target host.

The task will attempt to ping the URL(s) and any additional hosts that you specify in the Description field of the instance. For the instance you can specify any number of hosts to ping in the description field when you surround the host with certain identifiers as shown in the example below:

"This is an example description for a URL instance that is to be targeted by the URLGrabScripted monitor. In addition to the URL(s) configured for the instance, the following hosts can be pinged with the built-in diagnostic task if this monitor should happen to fail: PINGTEST#MyServer1.Mydomain.com# and also PINGTEST#MyServer2.Mydomain.com# and also this server, PINGTEST#MyOtherServer.Mydomain.com#.

## URLGenie Get Certificate Info

Description: Will use Powershell to retrieve properties of the certificate presented by the host webserver. Only applies to HTTPS URLs.

## URLGenie TestPath

Description: Will test if watcher node has access to specified folder path. Useful when troubleshooting URL instance discovery problems. See the [URL Instance Discovery](#_URL_Instance_Discovery) tutorial below for RunAs account considerations.

# Rules

URLGenie HTTP ResponseTime Collection PerfRule

Default: ENABLED

Description: This tracks the response times of normal/standard URLs (seconds). This is disabled by default for objects that fall into the Scripted and Special groups.

## URLGenie Scripted Request ResponseTime Collection Scripted PerfRule

Default: DISABLED

Description: This tracks the response times of any URLs (except ClientCert group objects) (seconds). This is disabled for ALL objects and must be manually enabled with an override. Also useful for objects that fall into the SPECIAL group: [URLGenie.URL.SPECIAL.Group](#_Groups)

URLGenie Scripted Request StatusCode Collection Scripted PerfRule

Default: DISABLED

Measures the StatusCode returned by the server.

URLGenie HTTP Scripted IE Login ResponseTime Collection Scripted PerfRule

Default: DISABLED

Description: This tracks the response times of URLs that require the ScriptedIELogin monitor type (seconds). This is disabled for ALL objects and must be manually enabled with an override. See the [tutorial](#_Monitor_a_Login) below.

NOTE: See additional configuration requirements in the monitor section below for: "[URLGenie.URLGrabScriptedIELogin.Monitor](#_URLGenie.URLGrabScriptedIELogin.Mon)"

URLGenie HTTP ContentSize Collection PerfRule

Default: DISABLED

Measures the Response Body size (bytes.)

URLGenie HTTP Days to Certificate Expiration PerfRule

Default: DISABLED

Measures the number of days until the certificate expires. This applies to Standard Https URL types only (not Special types).

URLGenie HTTP DNS Resolution Time Collection PerfRule

Default: DISABLED

Measures the DNS Resolution Time (seconds)

URLGenie HTTP Download Time Collection PerfRule

Default: DISABLED

Measure the time required to download the page (seconds)

URLGenie HTTP Time To First Byte Collection PerfRule

Default: DISABLED

Measures the time to receive the first byte (seconds)

URLGenie HTTP Time To Last Byte Collection PerfRule

Default: DISABLED

Measures the time to receive the last byte (seconds)

URLGenie HTTP StatusCode Collection PerfRule

Default: DISABLED

Measures the StatusCode returned by the server.

# Monitors

## URLGenie IE Login Scripted Monitor

Default: Disabled.

This uses a Powershell script that will simulate username/password entry and submission with an Internet Explorer object. The success of this method is subject to many other factors related to configuration of the individual target web server and IE settings on the watcher node. For URL instances that are meant to be monitored by this workflow, add a caret ‘^’ to the beginning of the URL in the config file or if using a 2-URL sequence, separate the URLs with the caret ‘^’.

Example:

<url>

<![CDATA[^http://blog.shoppingwebsite.com/shirts]]>

</url>

This will make it a member of the "[Special](#_Groups)" group and therefore **all rules and monitors will be disabled by default**. Why would you want it to be disabled by default? So that you can configure the required override settings shown below before enabling the one and only monitor that should be applicable to this URL.

Parameters and default values:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Default Value | Description |
| CaseSensitive | boolean | false |  |
| ClickButtonID | string | “EMPTY” | HTML Element ID of the login button. Recommended to simulate the click/submission. |
| ClickButtonName | string | <blank> | Will be used if no ID exists. |
| ClickButtonType | string | <blank> | Helps to identify the correct button if multiple buttons exist. |
| LogOutURL | string | <blank> | Very useful for controlling session state. Helps login simulation work consistently. |
| OptionalElement | string | <blank> | (optional) Some pages might require an additional submission field like "Last Name" (i.e. Delta.com). This provides the ability to input an additional element value if needed. |
| OptionalElementID | string | <blank> | (optional) ID of the optional element. |
| OptionalElementName | string | <blank> | Will be used if no ID exists. |
| PasswordElementID | string | “EMPTY” | ID of the password field element. |
| PasswordElementName | string | <blank> | Will be used if no ID exists. |
| UsernameElementID | string | “EMPTY” | ID of the username element. |
| UsernameElementName | string | <blank> | Will be used if no ID exists. |
|  |  |  |  |

Note: These items above MUST be overridden for each URL instance and are NOT configured within the configuration file(s).

There are some specific configurations required on each watcher node that will run this ScriptedIELogin monitoring workflow.

At this time there is no logic for locating/selecting check boxes. This may be added in the future.

See the [**Tutorials**](#_Monitor_a_Login) section below for specific configuration requirements.

## URLGenie Scripted Monitor

Default: DISABLED.

NOTE: This monitor requires Powershell v3.0 or higher on the watcher node to use the "Invoke-WebRequest" cmdlet.

Description: This monitor is applicable to only the “special” URL types and normal monitoring doesn't apply to these types of instances. Typically, this unit monitor must be manually enabled (via override). This monitor will only be auto-enabled for instances that have a CertThumbprint property populated value. See "CertThumbprint" definition above.

This script will scrape a URL body (and optionally the header and any server error response) for content matching and also check the status code of the response header. You can submit unlimited URLs separated by a caret '^' character.

This scripted monitor type uses a datasource that is designed to parse multiple values from the criteria parameters based on this caret delimiter [^] in order to test a sequence of URLs (1 or more) with individual test criteria.

When multiple URLs are submitted:  
Example, three URLS: **www.abc.com^support.xyz.com^eStore.123.com**

if only one value is submitted for a particular test criteria parameter  
Example: <contentmatch>website,online</contentmatch>then "website" AND "online" will be used as the ContentMatch test keywords for ALL URLs in the sequence and is considered to be a single criteria value even though it consists of two keywords (comma separated).

If more than one value is submitted for a criteria (but less than the total number of URLs)   
Example: <contentmatch>fishing^bowling</contentmatch>   
this is considered two values which is more than 1 but less than the total number of URLs (3) and will result in a monitor failure and script exit.

The reason behind this is so you can submit one parameter to act as a global setting for all URLs OR be explicit with each URL by submitting an equal number of values for the number of URLs.

In the example (for 3 URLs) above you would have to submit either one single value like the following: <contentmatch>contact us,login</contentmatch>

Or…

<contentmatch>login</contentmatch>

Or…  
<contentmatch>contact us</contentmatch>

or you would have to submit 3 separate values separated by the caret:   
<contentmatch>fishing,pole^finance^shirt,pants</contentmatch>  
or…  
<contentmatch>pickle^plum^bread</contentmatch>

You separate multiple string keywords with a comma. Example:   
<contentmatch>fishing,pole</contentmatch>   
will test a page for both "fishing" and "pole" but is considered to be a single parameter for the content match.

Another Example:  
<statusthreshold>399</statusthreshold>

If number of URLs = 4, number of StatusThreshold values = 1, The StatusThreshold test will use that one value (“399”) for all 4 URLs.

Another Example:

<statusthreshold>3200^200^504^403</statusthreshold>

If number of URLs = 4, number of StatusThreshold values = 4, The StatusThreshold test will use the individual values for each individual corresponding URL.

If number of URLs = 4, number of StatusThreshold values = 2 or 3, This will result in failure and script exit.

Specifies how long the request can be pending before it times out. Enter a value in seconds. The default value, 0, specifies an indefinite time-out.

## URLGenie CA Untrusted Monitor

Auto targets any standard HTTPS instances. Will evaluate the Authority for the certificate provided by the endpoint and alert if the CA is found to be untrusted.

## URLGenie Certificate Expired Monitor

Auto targets any standard HTTPS instances. Will evaluate the certificate provided by the endpoint and alert if the cert is found to be expired.

## URLGenie Content Monitor

Auto targets all standard URLs and will alert if no content match is found (default). Other RegExCompareType types can be used with an override. See the “ContentMatchRegExOperator” standard parameter type above.

## URLGenie DNS Resolution Failure Monitor

Auto targets all standard URLs and will alert if unable to resolve DNS for the host.

## URLGenie Error Code Monitor

Auto targets all standard URLs and will alert if the endpoint returns an error code beyond the specified threshold.

## URLGenie Reachable Monitor

Auto targets all standard URLs and will alert if the endpoint does not respond and no connection can be established.

## URLGenie Response Time Monitor

Auto targets all standard URLs and will alert if the specified response time threshold is exceeded.

## URLGenie Status Code Monitor

Auto targets all standard URLs and will alert if the endpoint returns a status code beyond the specified threshold.

## URLGenie Certificate Expires Soon Monitor

Auto targets any standard HTTPS instances. Will evaluate the certificate provided by the endpoint and alert if the cert is found to expire soon.  
  
URLGenie WatcherNode ConfigFile Path Test Monitor  
Will test access to the configuration file of the watcher node.

URLGenie Watcher Request Dependency Monitor  
Default: Disabled. Typically, it doesn't make sense for the watched instances to affect the health of the watcher node/server.

URLGenie Aggregate Health Monitor  
Aggregate for simplification of alerting (if needed). Default: Alerting disabled.

# The Configuration File

The .xml configuration files are what contain the URL instance definitions. The format is very simple as you can see in the examples below.

At the time of this writing you can have any number of files in the designated discovery folder(s). However the total number of objects returned in the discovery is limited to the discovery object limit size of 4MB (depending on the amount of text included in the optional fields: Description, Wiki, Headers, etc.). This is a hardcoded limitation of the size of the discovery object and beyond my control.

NOTE: The config file names MUST be in this format: "Requests.xml" and should use UTF-8 encoding.

File name examples:

Requests.xml

Requests\_dev.xml

RequestsForAccountingDept.xml

Requests\_qa.xml

To disable a discovery file simply delete it or rename it.

Renamed file example:

Requests.xml.INACTIVE

## Configuration File Examples

### Example 1: Basic Template

------------ BEGIN BASIC CONFIG FILE TEMPLATE < **THIS LINE NOT INCLUDED** > ------------

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<URLGenieconfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<requests>

<request>

<authenticationscheme></authenticationscheme>

<authuser></authuser>

<body></body> <!-- can enclose text in CDATA -->

<certthumbprint></certthumbprint>

<contentmatch></contentmatch> <!-- can enclose text in CDATA -->

<description></description> <!-- can enclose text in CDATA -->

<errortime></errortime>

<groupid></groupid>

<header1name></header1name> <!-- can enclose text in CDATA -->

<header1value></header1value> <!-- can enclose text in CDATA -->

<header2name></header2name> <!-- can enclose text in CDATA -->

<header2value></header2value> <!-- can enclose text in CDATA -->

<header3name></header3name> <!-- can enclose text in CDATA -->

<header3value></header3value> <!-- can enclose text in CDATA -->

<headercontenttype></headercontenttype> <!-- can enclose text in CDATA -->

<index></index>

<interval></interval>

<proxy></proxy> <!-- can enclose text in CDATA -->

<proxyauthenticationscheme></proxyauthenticationscheme>

<proxyuser></proxyuser> <!-- can enclose text in CDATA -->

<responsetimeout></responsetimeout>

<retrycount></retrycount>

<statusthreshold></statusthreshold>

<url><![CDATA[HTTP:\\YOUR\_URL\_ADDRESS\_HERE]]></url> <!-- can enclose text in CDATA -->

<useragent></useragent> <!-- can enclose text in CDATA -->

<verb></verb>

<warningtime></warningtime>

<watchers>.</watchers> <!-- This will most likely match all/any of your activated watcher nodes -->

<wiki></wiki> <!-- can enclose text in CDATA -->

</request>

</requests>

------------ END BLANK CONFIG FILE TEMPLATE < **THIS LINE NOT INCLUDED** > ------------

### Example 2: Sample File

------------ BEGIN SAMPLE CONFIG FILE < **THIS LINE NOT INCLUDED** > ------------

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<URLGenieconfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<requests>

<request>

<authenticationscheme>None</authenticationscheme>

<authuser/>

<body>

<![CDATA[<?xml version="1.0"?>

<soap:Envelope

xmlns:soap="http://www.w3.org/2001/12/soap-envelope"

soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">

<m:GetStockPrice>

<m:StockName>IBM</m:StockName>

</m:GetStockPrice>

</soap:Body>

</soap:Envelope>]]>

</body>

<certthumbprint/>

<contentmatch>money</contentmatch>

<description>Test of CNN.com. The SOAP body doesn't actually do anything in this example as CNN.com ignores it. It is only meant to be an example of how to use the 'body' tag with CDATA encapsulation. </description>

<errortime>30</errortime>

<groupid>MyTest</groupid>

<header1name/>

<header1value/>

<header2name/>

<header2value/>

<header3name/>

<header3value/>

<headercontenttype>text/xml</headercontenttype>

<index>10</index>

<interval>60</interval>

<proxy/>

<proxyauthenticationscheme/>

<proxyuser/>

<responsetimeout>60</responsetimeout>

<retrycount>0</retrycount>

<statusthreshold>200</statusthreshold>

<url>

<![CDATA[http://www.cnn.com]]>

</url>

<useragent/>

<verb>GET</verb>

<warningtime>5</warningtime>

<watchers>.</watchers>

<!-- This will most likely match all/any of your activated watcher nodes -->

<wiki>

<![CDATA[http://mywiki.com]]>

</wiki>

</request>

<request>

<authenticationscheme>None</authenticationscheme>

<authuser/>

<body/>

<certthumbprint/>

<contentmatch>National Weather Service Forecast</contentmatch>

<description>A simple test of the National Weather Service Forecast service</description>

<errortime>30</errortime>

<groupid>MyTest</groupid>

<header1name>Accept</header1name>

<header1value>

<![CDATA[text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,/;q=0.8]]>

</header1value>

<header2name>Cache-Control</header2name>

<header2value>max-age=0</header2value>

<header3name>Connection</header3name>

<header3value>keep-alive</header3value>

<headercontenttype/>

<index>2</index>

<interval>120</interval>

<proxy/>

<proxyauthenticationscheme/>

<proxyuser/>

<responsetimeout>60</responsetimeout>

<retrycount>1</retrycount>

<statusthreshold>200</statusthreshold>

<url>

<![CDATA[http://graphical.weather.gov/xml/sample\_products/browser\_interface/ndfdXMLclient.php?lat=38.99&lon=-77.01&product=time-series&begin=2004-01-01T00:00:00&end=2020-04-20T00:00:00&maxt=maxt&mint=mint]]>

</url>

<useragent/>

<verb>GET</verb>

<warningtime>10</warningtime>

<watchers>.</watchers>

<!-- This will most likely match all/any of your activated watcher nodes -->

<wiki>http://mycompanywiki.com</wiki>

</request>

<request>

<authenticationscheme>None</authenticationscheme>

<authuser/>

<body/>

<certthumbprint/>

<contentmatch>Lucky</contentmatch>

<description>Test of Bing.com</description>

<errortime>3</errortime>

<groupid>MyTest</groupid>

<header1name/>

<header1value/>

<header2name/>

<header2value/>

<header3name/>

<header3value/>

<headercontenttype/>

<index>4</index>

<interval>120</interval>

<proxy/>

<proxyauthenticationscheme/>

<proxyuser/>

<responsetimeout>60</responsetimeout>

<retrycount>0</retrycount>

<statusthreshold>200</statusthreshold>

<url>https://www.Bing.com</url>

<useragent/>

<verb/>

<warningtime>2</warningtime>

<watchers>.</watchers>

<!-- This will most likely match all/any of your activated watcher nodes -->

<wiki/>

</request>

<request>

<authenticationscheme>None</authenticationscheme>

<authuser/>

<body/>

<certthumbprint/>

<contentmatch>

<![CDATA[Welcome to MSN^Scores]]>

</contentmatch>

<description> This is a test of two URLs. The scripted monitor AND rule for this instance must be enabled manually with an override. Monitor: URLGenie Scripted Monitor Rule: URLGenie Scripted Request ResponseTime Collection Scripted PerfRule. PINGTEST#www.microsoft.com#</description>

<errortime>15</errortime>

<groupid>MyTest</groupid>

<header1name/>

<header1value/>

<header2name/>

<header2value/>

<header3name/>

<header3value/>

<headercontenttype/>

<index>5</index>

<interval>180</interval>

<proxy/>

<proxyauthenticationscheme/>

<proxyuser/>

<responsetimeout>60</responsetimeout>

<retrycount>1</retrycount>

<statusthreshold>200^399</statusthreshold>

<url>

<![CDATA[http://www.msn.com/^http://msn.foxsports.com/]]>

</url>

<useragent/>

<verb>GET</verb>

<warningtime>5</warningtime>

<watchers>.</watchers>

<!-- This will most likely match all/any of your activated watcher nodes -->

<wiki>http://www.mycompanywiki.com</wiki>

</request>

</requests>

</URLGenieconfig>

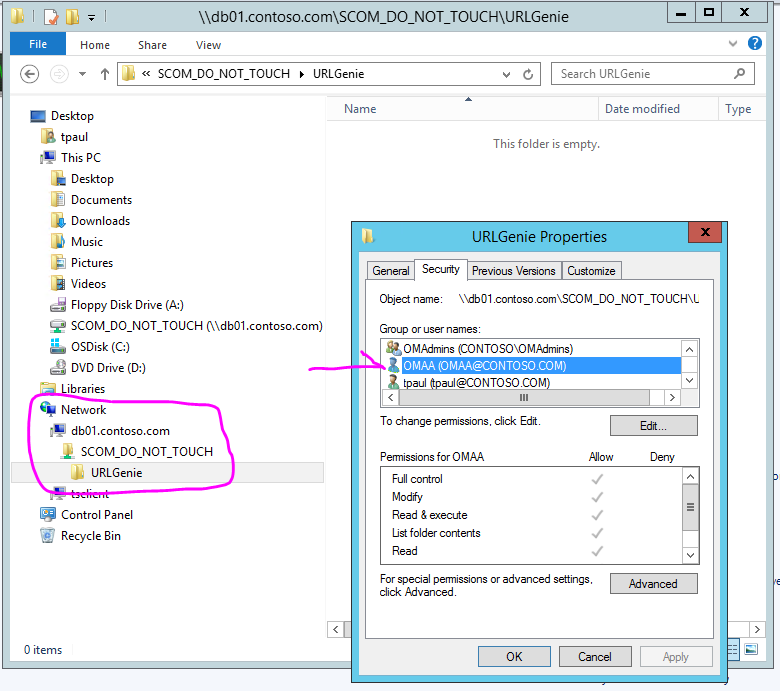
------------ END SAMPLE CONFIG FILE **<THIS LINE NOT INCLUDED>** ------------

# Tutorials

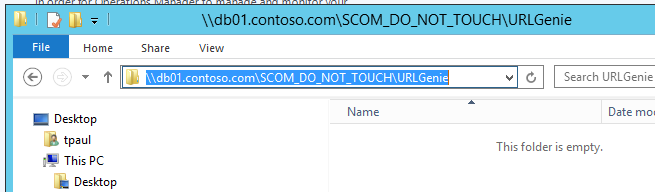
## Activate a Watcher Node

URL instances get discovered from one or more configuration files. You will need a place to store these files. I recommend you create a shared network folder from which all your watcher nodes can access the files.

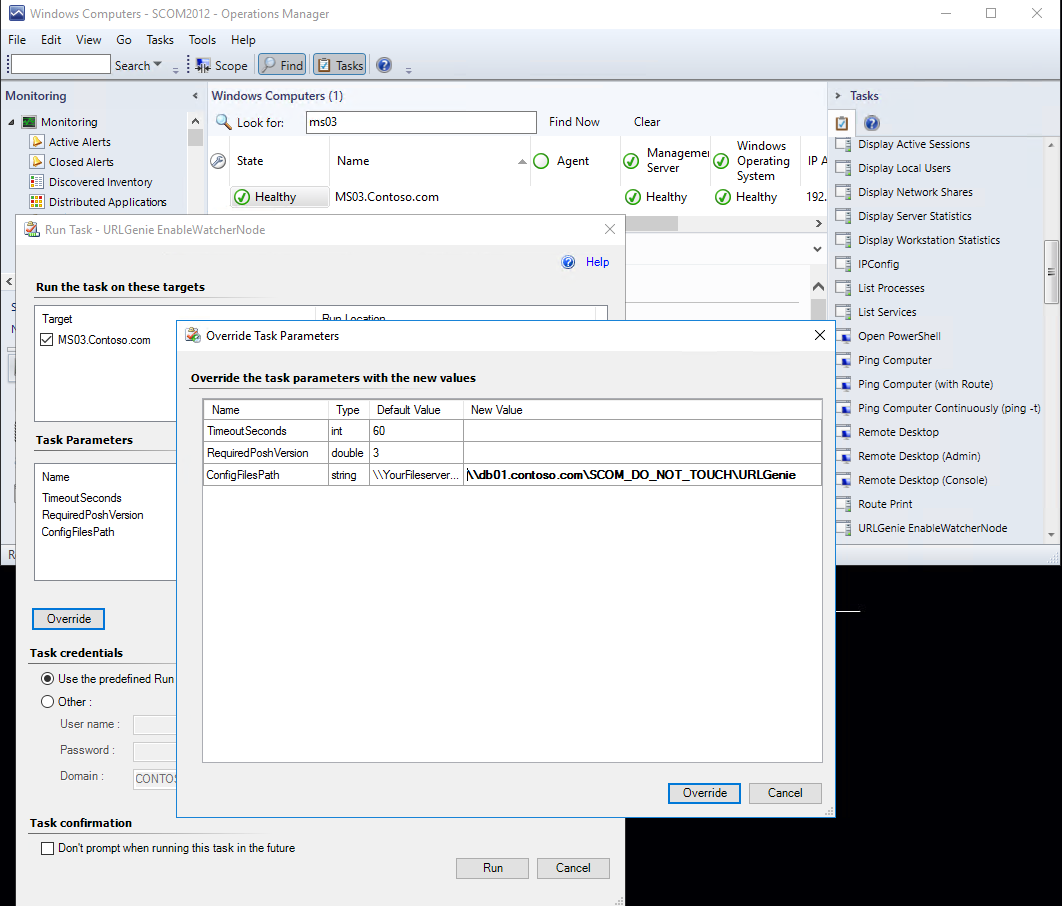
The example below demonstrates a shared folder on the SCOM database server. The OMAA account has share read/write and NTFS full permissions.



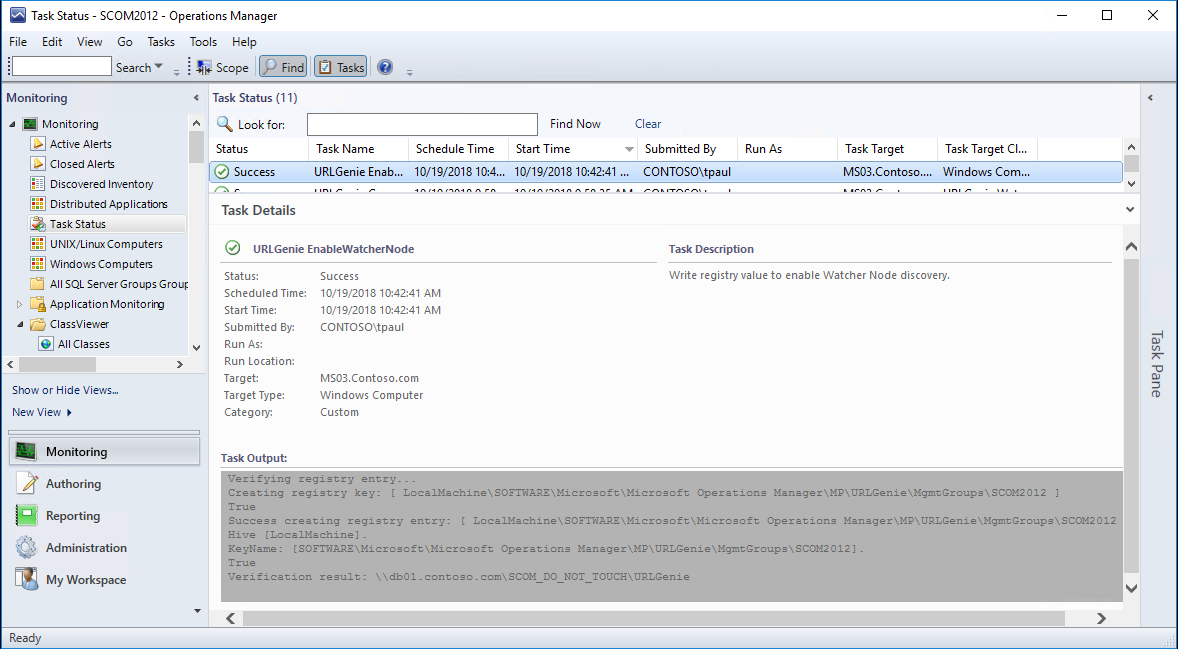
From the management server, navigate to the shared folder using the fully qualified domain name of the file server as shown below. Copy the path to the clipboard.

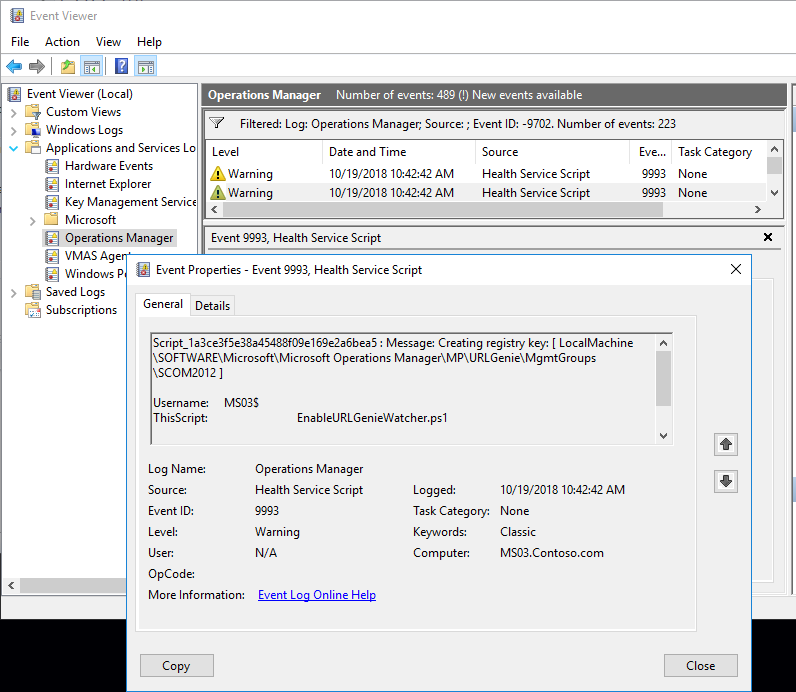


Select a Windows Computer object from any state view, launch the “URLGenie EnableWatcherNode” task and override it with the path to your shared folder as shown below:

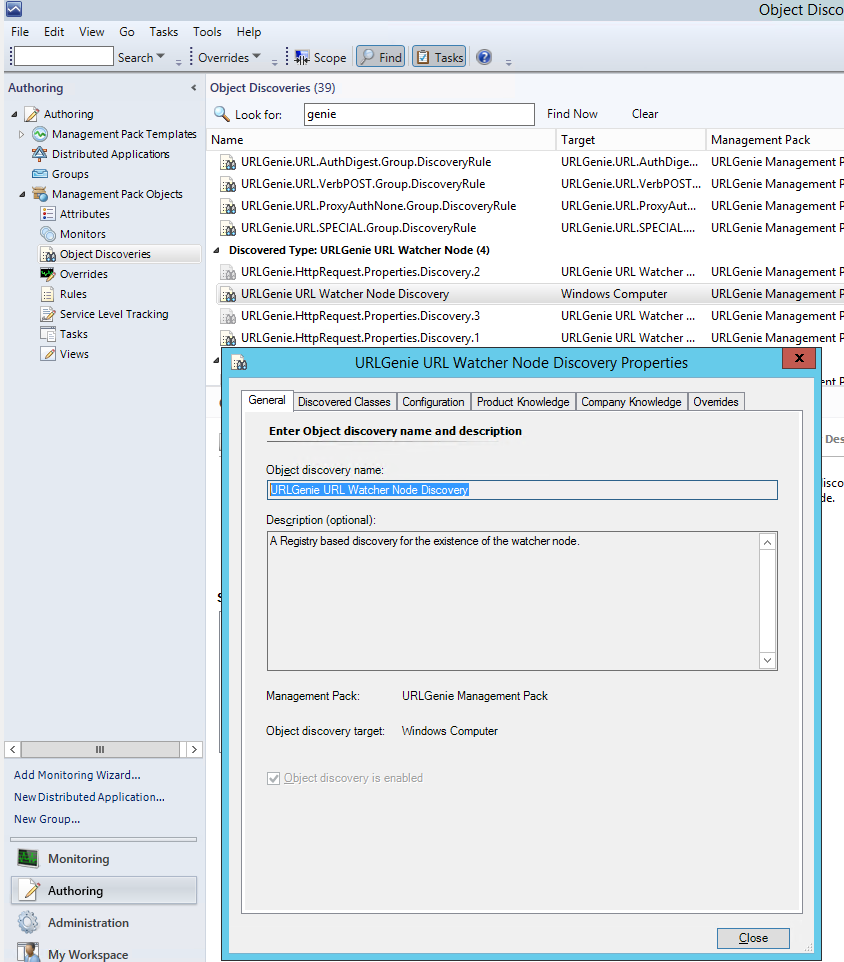


You should see this task status and the following event on the target machine indicating that the registry value was successfully added:

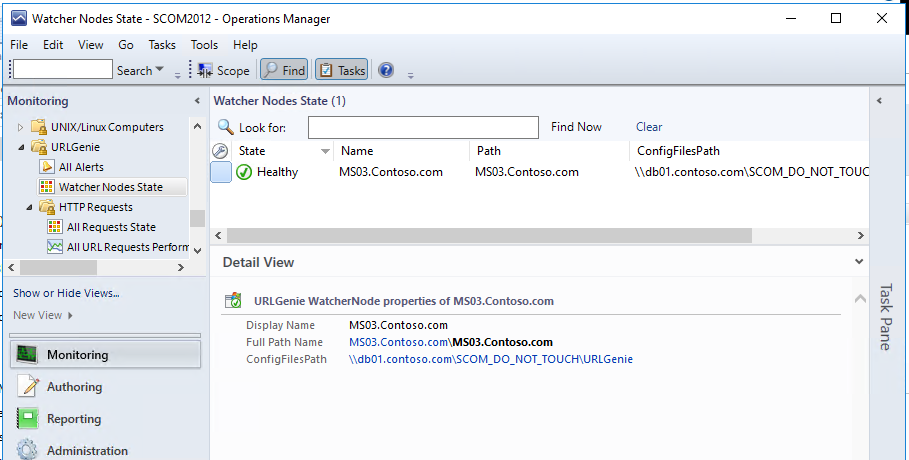




Now the target machine is discoverable by this discovery:



Your watcher nodes should eventually appear here (default discovery interval: 1 hour )



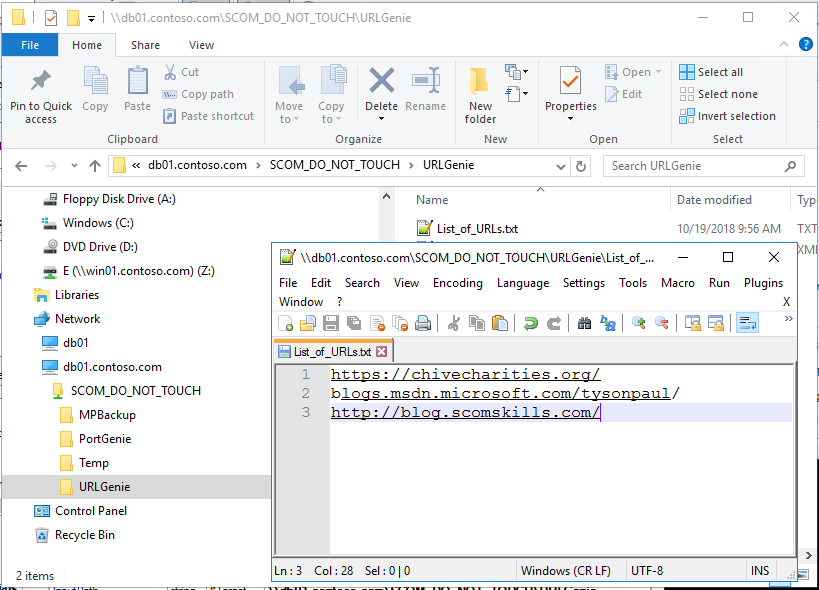
## Create a Configuration File

There are two easy ways to go about this:

1. Use the task: *URLGenie Generate Config From List*
2. Manually create the file from the blank template example provided in this guide.

### Use the task: “URLGenie Generate Config From List”

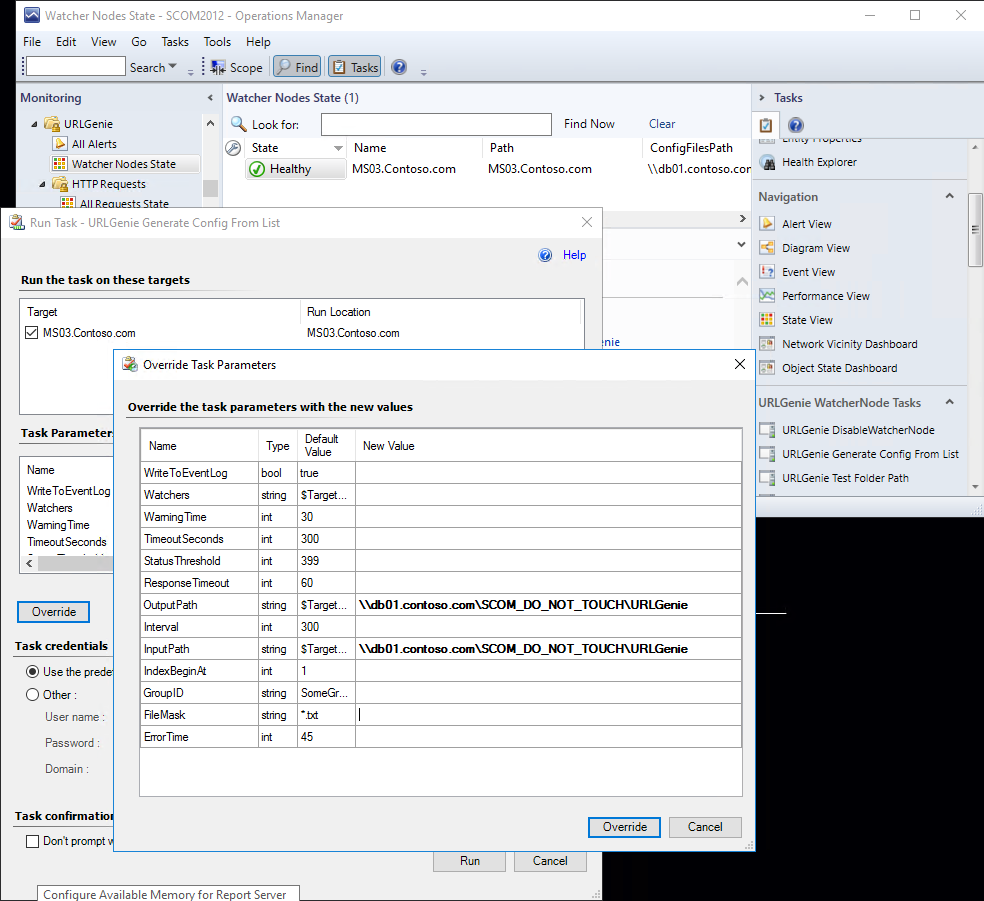
a) Create a simple txt file that contains URLs (see screenshot below):



b) Copy the path of the folder that contains the txt file. Paste this path into the “input file” override field of the task. Use the fully qualified domain name.

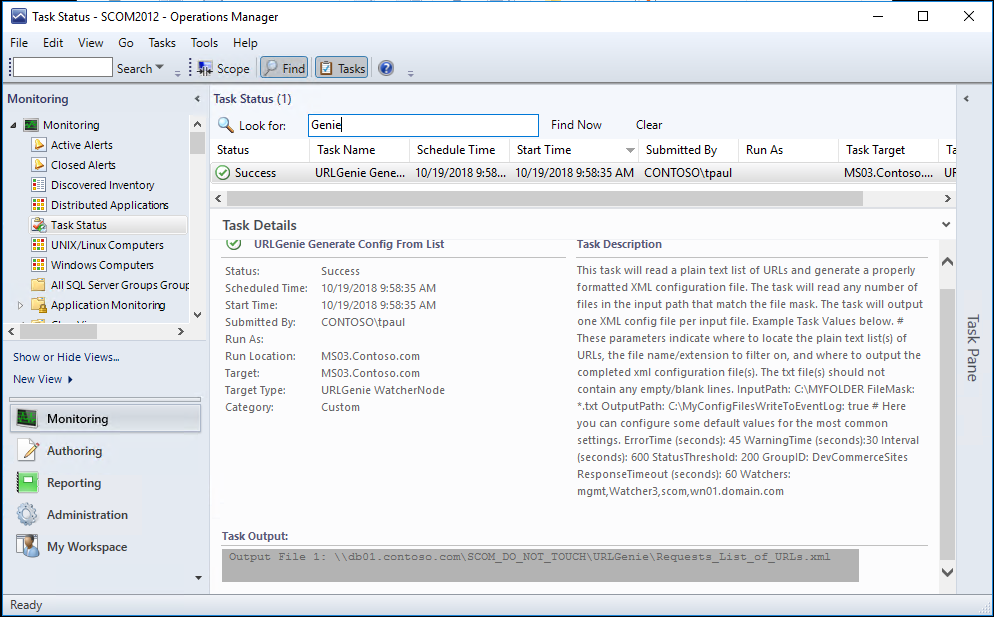
c) Also specify the path where you want the output configuration file to be created.   
**NOTE: you must run this task with a user account that has permissions to access the Input folder and also to create/write the output file to the Output folder.**

In my example, the path is the same as that which has been configured for my watcher node and is where the watcher will look to consume configuration files. You can create your config files anywhere you like but remember that they must end up in the path where your watcher node has been configured to look.

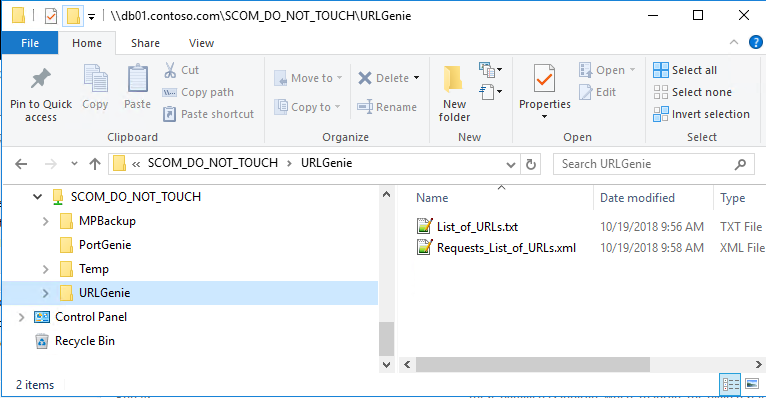


*InputPath* and *OutputPath* are the only two values that are required for this step. The rest are optional and can be modified directly in the Requests file at any time.

Upon successful completion of the task, you will see these task results.



The configuration file will now exist in the Output folder:



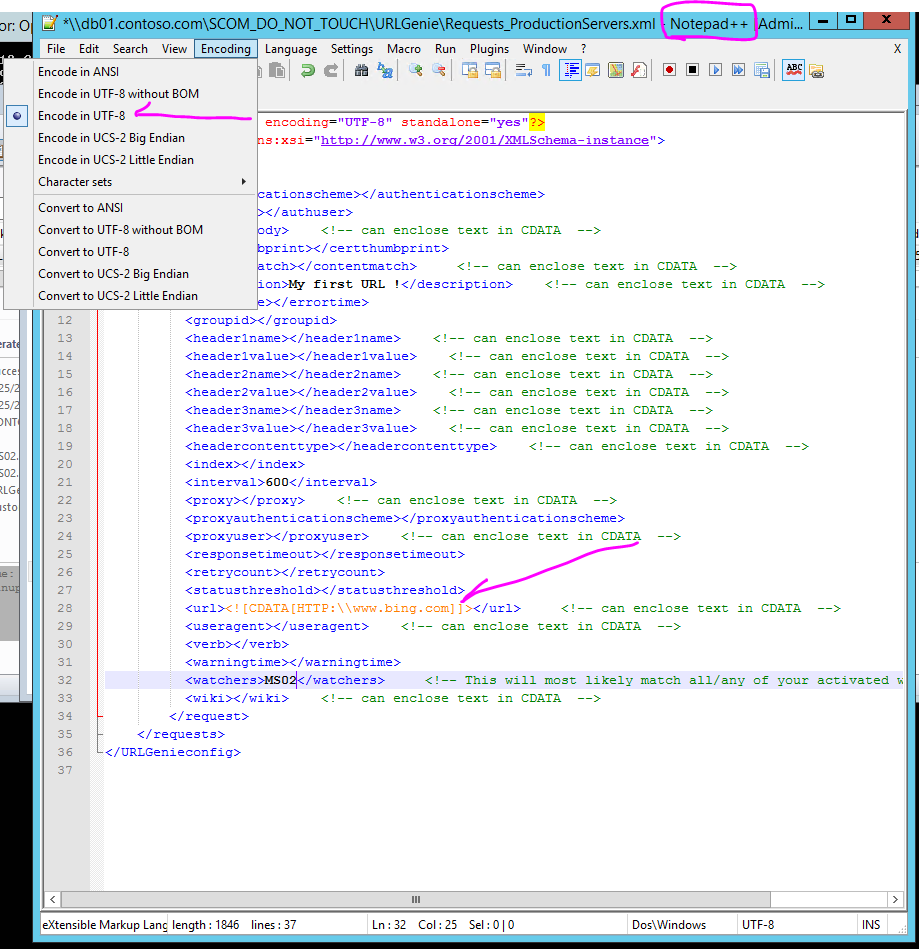
The .txt file is harmless and can remain. Only files which match this mask will be analyzed by the URL discovery: *Requests.xml*

### Manually create the file from the blank template example provided in this guide.

a) Create a file (encoded with UTF-8). The filename MUST begin with “Requests” and end with “.xml”. Example: *Requests\_ProductionServers.xml***Note**: **Notepad++** is a fine editing tool to use for this and it’s free!

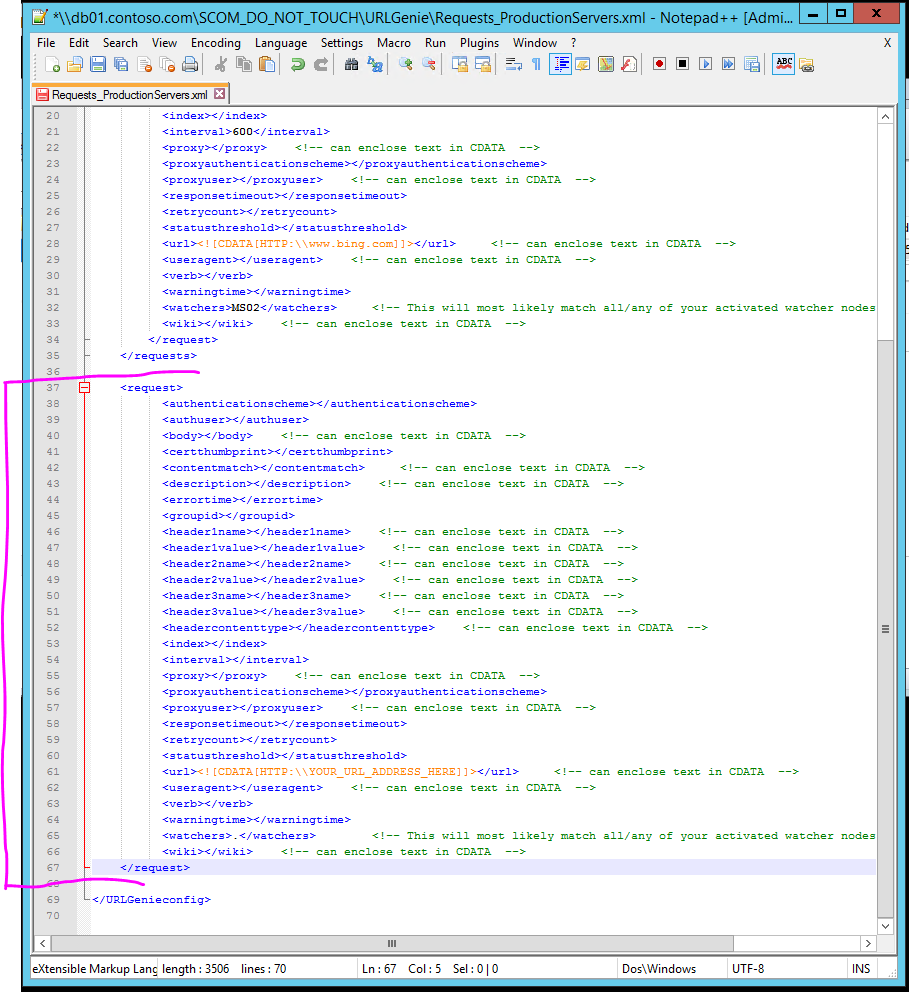
b) Copy the sample template from this MP guide into the file.

c) Customize the URL element and any other elements that you want to change. The only “required” element is the URL. All other URL instance properties will get populated with default values at discovery time if the configuration file elements are not populated.



## Add a URL Instance to an Existing Config File

Simply copy the “<request>…</request>” element and all of the contained elements from the blank template in this MP guide (above) and past the section into an existing configuration file either below or above an existing “<request>…</request>” section. Then customize the new “request” section elements.



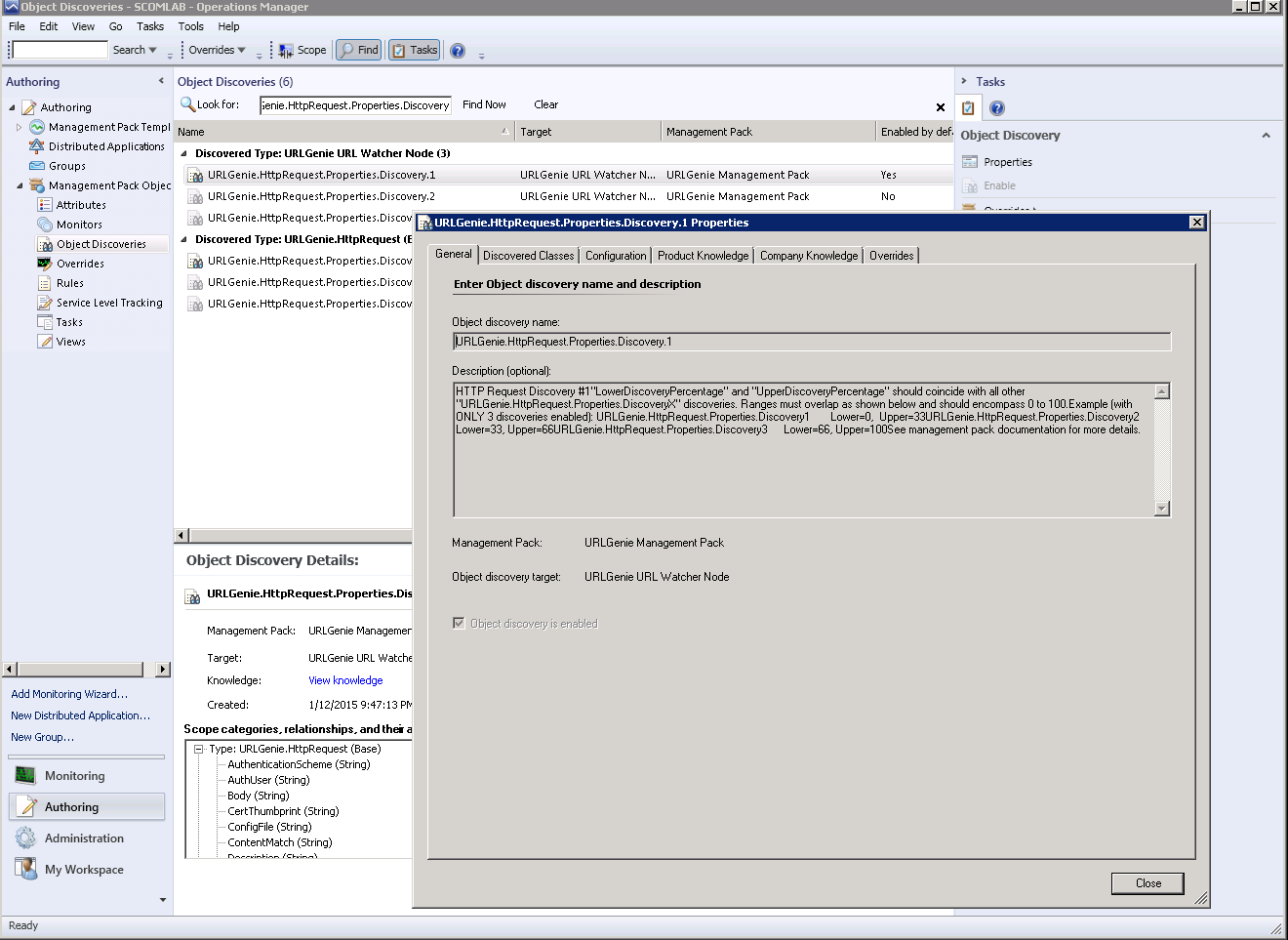
## URL Instance Discovery

Depending on the watcher node that you use to discover URL instances, you may or may not need to configure a RunAs account to perform the discovery. If your watcher node is an Operations Manager Management Server then the URL instance discovery will use the OMAA account by default. If you use an ordinary Windows agent-managed machine, the discovery will use the LocalSystem computer account context by default. In this case you will probably want to specify a RunAs account which has permissions to access the folder where the configuration files are located and add it in this RunAs Profile: URLGenie URL Properties Discovery Account. Be sure to distribute the account to the watcher node agent on the Account->Distribution configuration screen (when using “More Secure” account configuration”.

Once the following items are configured correctly:

1. One or more Watcher nodes have been enabled and discovered (and appear in the Watcher Nodes state view)
2. One or more configuration files exist with one or more URL instances defined within.
3. The designated watcher nodes can access the configuration folder location (using the designated RunAs context)

URLs should get discovered within 10 minutes by this discovery:



Note: There are two additional discoveries that are disabled by default. They can be enabled and configured for situations where you might need to discover massive quantities of URL instances during which the discovery data object size might exceed the size limit. It would take an extraordinary number of instances to need these additional discoveries but the ability exists nonetheless.

## Enable Monitoring a URL Instance with a Client Certificate

### How the scripted monitor works.

This scripted monitor type uses a datasource that is designed to parse multiple values from the criteria parameters

based on this caret delimiter [^] to test a sequence of URLs (1 or more) with individual test criteria.

When multiple URLs are used (Example: www.abc.com**^**support.xyz.com**^**eStore.123.com ):

If only one value is submitted for a particular test criteria parameter (Example: ContentMatch [website,online] ),

then "website" AND "online" will be used as the ContentMatch test keywords for ALL URLs and is considered to be a single criteria value even though it consists of two keywords (comma separated). If more than one value is submitted for a criteria (but less than the total number of URLs)

(Example: ContentMatch [fishing^bowling] ), this is considered two values which is more than 1 but less than the total number of URLs(3) and will result in a monitor failure and script exit. The monitor works this way so you can submit one parameter to act as a global setting for all URLs OR be explicit with each URL by submitting an equal number of values for the number of URLs. In the example (for 3 URLs) above you would have to submit either one single value like the following: [contact us,login] or [login] or [contact us] or you would have to submit 3 separate values separated by the caret: [fishing,pole^finance^shirt,pants] or [pickle^plum^bread].

You separate multiple string keywords with a comma. Example: [fishing,pole] will test a page for both "fishing" and "pole" but is considered to be a single parameter for the content match.

Example:

StatusThreshold=200

If number of URLs = 4, number of StatusThreshold values = 1, The StatusThreshold test will use that one value for all 4 URLs.

Example2:

StatusThreshold=200^200^504^403

If number of URLs = 4, number of StatusThreshold values = 4, The StatusThreshold test will use the individual values for each individual corresponding URL.

If number of URLs = 4, number of StatusThreshold values = 2 or 3, This will result in failure and script exit.

If a URL has the CertThumbprint property populated, this monitor will be automatically enabled by an override that is enabled for the ClientCert group.

### Step by Step

1. Import the MP.
2. If your watcher node is not a SCOM management server, choose a domain user account to run the discovery workflow for the URL instances. Create the Windows account type in SCOM and add it to the RunAs Profile: URLGenie URL Properties Discovery Account for all targeted objects.
3. Create a shared folder on your Datawarehouse server.   
   suggestion: C:\SCOM\_DO\_NOT\_TOUCH\URLGENIE

Share Permissions: Read/write: domain\<youraccount>

1. Put your Requests\_.xml file into the URLGENIE folder. Make sure the file is encoded in UTF8. (you can use Notepad++ for this)

Modify the following elements in the config file to meet your needs: (see MP guide for more information about these properties)

* Certthumbprint
* Contentmatch
* Warningtime
* Errortime
* Interval
* URL (this URL must include a caret ‘^’ to be discovered correctly)

1. Decide which server you want to be your watcher node. I suggest choosing your second (or first) management server. From your watcher node, open Windows File Explorer and navigate to the URLGenie shared folder **using the FQDN of the server**. Highlight and copy that path from the address bar to the clipboard.

Example: \\DB01.CONTOSO.COM\scom\_do\_not\_touch\URLGenie

Find your watcher node in the Windows Computer monitoring view within the SCOM Console, select the watcher node. You will see a task in the Tasks pane titled: “URLGenie EnableWatcherNode”. Run that task, override it with the full network path to the shared URLGenie folder.

1. With the watcher node selected, run the Task: “URLGenie TestPath” . Use the designated discovery Windows domain account credentials and the network path to the share for this task to make sure that your RunAs account can access the URLGenie shared folder. The output of the task should return “Success”. If not, recheck your share path, share permissions, and discovery account and profile configurations.
2. Wait for the watcher node to be discovered. Once the watcher node is discovered, the URL instance should be discovered shortly thereafter. If not, enabled logging on the Discovery workflow: “URLGenie.HttpRequest.Properties.Discovery.1”
3. Once the URL is discovered as a special HTTP**S** type (may take up to 10 mins) monitoring should become active automatically only if a Certthumbprint value has been provided in the config file. If you’ve installed the cert correctly into the correct location on the watcher node, then monitoring should begin working successfully. Below is some additional information about properties related to this workflow.
4. Once your instance turns Healthy, make sure to test it by overriding the ContentMatch to something absurd like: “abracadabra” to test the monitor; force an unhealthy Critical state.

Note: if you override the monitor, you will also have to enable/override the ResponseTime performance rule: “URLGenie Scripted Request ResponseTime Collection Scripted PerfRule” with the same settings.

### Variable Descriptions

CertThumbprint [string] Note: this should be configured within .xml config file

Default: <blank>

Description: The thumbprint of the client certificate to use for the request. The certificate must be installed on any watcher nodes that run this monitor workflow. The certificate should be installed into the "LocalMachine\My" location. You can view the certs installed in this location (on the watcher node) by running this command at a Powershell prompt: **gci Cert:\LocalMachine\my**

If you don't see the certificate thumbprint in the output, then you don't have the cert installed in the correct (default) cert store and the workflow will not function correctly for the URL object instance, in which case you will see errors in the OpsMan log.

Example: "F945261E79C44E1C5690000291DCBA79C4136712"

CertDir [string]

Default: "LocalMachine\My"

Description: The local certificate store where the "client cert" exists on the Watcher Node.

CheckErrorData [boolean] < true | false >

Default: false

Description: This setting will enable the checking of the server response (error data) for the content match string. In some circumstances an error is the expected response and typically in that situation there is no page content to check for a content match. This is only applicable to the ClientCert group members, or those instances with a Certthumbprint value.

If overriding any of these parameters be sure to override for both the applicable monitor AND performance rules.

TLS\_Version [string] < 1.0 | 1.1 | 1.2 | 1.3 > Note: 1.3 is for future use.

Default: 1.2

TrustAllRemoteSSLCertificates [boolean] < true | false >

Default: true

Monitor workflow will ignore problematic server certificates (expired, invalid, etc.)

VerbosityCharLimit [int] < 0..30000>

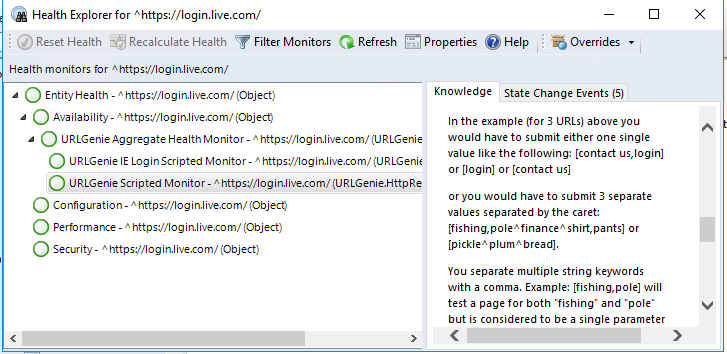
Default: 1024

This will truncate the body/content of the web page returned by the workflow.

NOTE: If your special URL has been configured with a certthumbprint value, monitoring should become active automatically. At first your instance may appear unmonitored. If it does not indicate a health state after a few minutes…

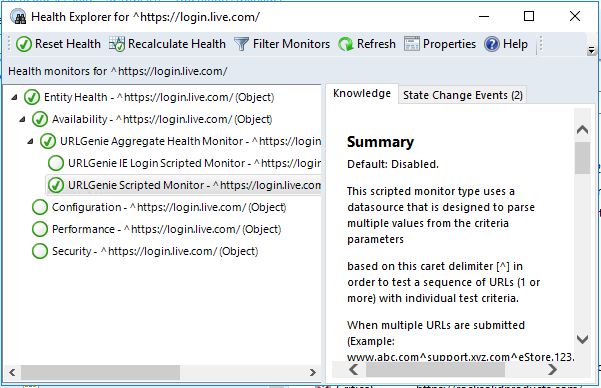
a) You did not populate the “certthumbprint” property in the .xml config file.

b) The group calc rules have not populated the instance into the “URLGenie.URL.ClientCert.Group” group for which all monitoring is disabled by default. In this case wait a few minutes for group calc to catch up. Soon your instance should reflect a health state.

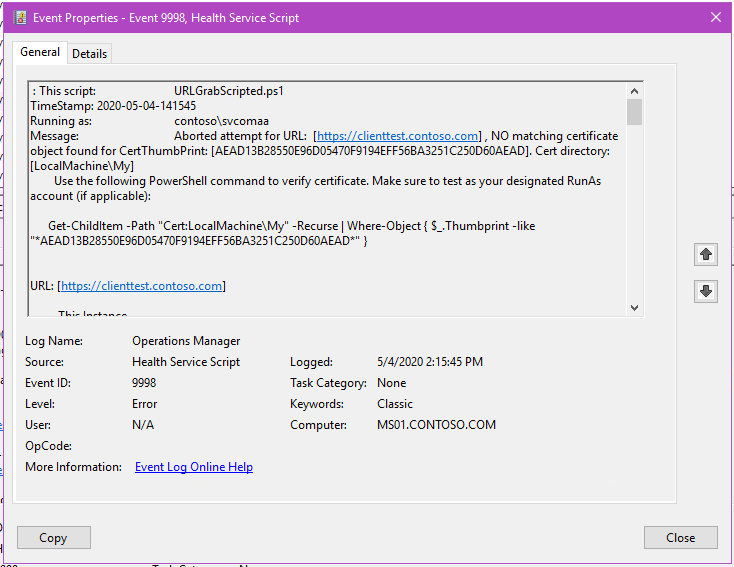


It is a good practice to test the monitor after it is configured. You can set the ContentMatch to something unlikely to appear on the target web page to create a false critical condition.

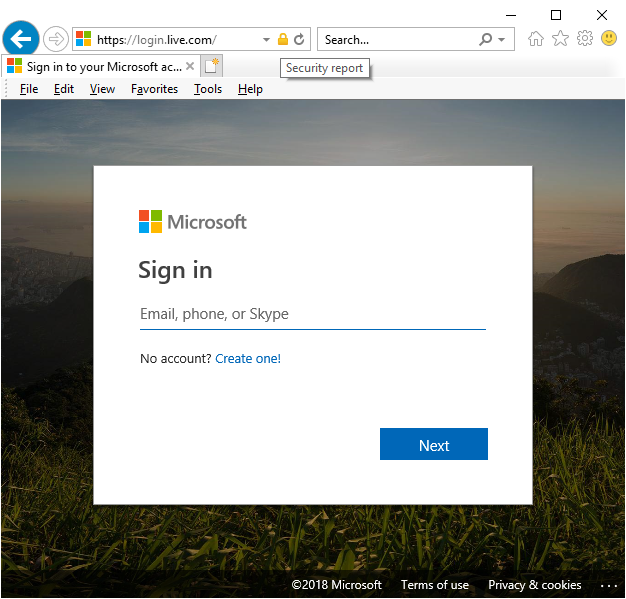
Eventually it should appear healthy.



If the certificate is not found on the server you will see an event logged similar to this:



## Monitor a Login Page (Forms Based Login)

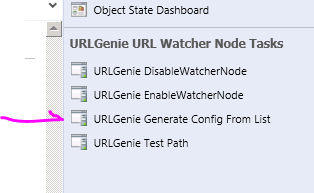


Example: <https://login.live.com>

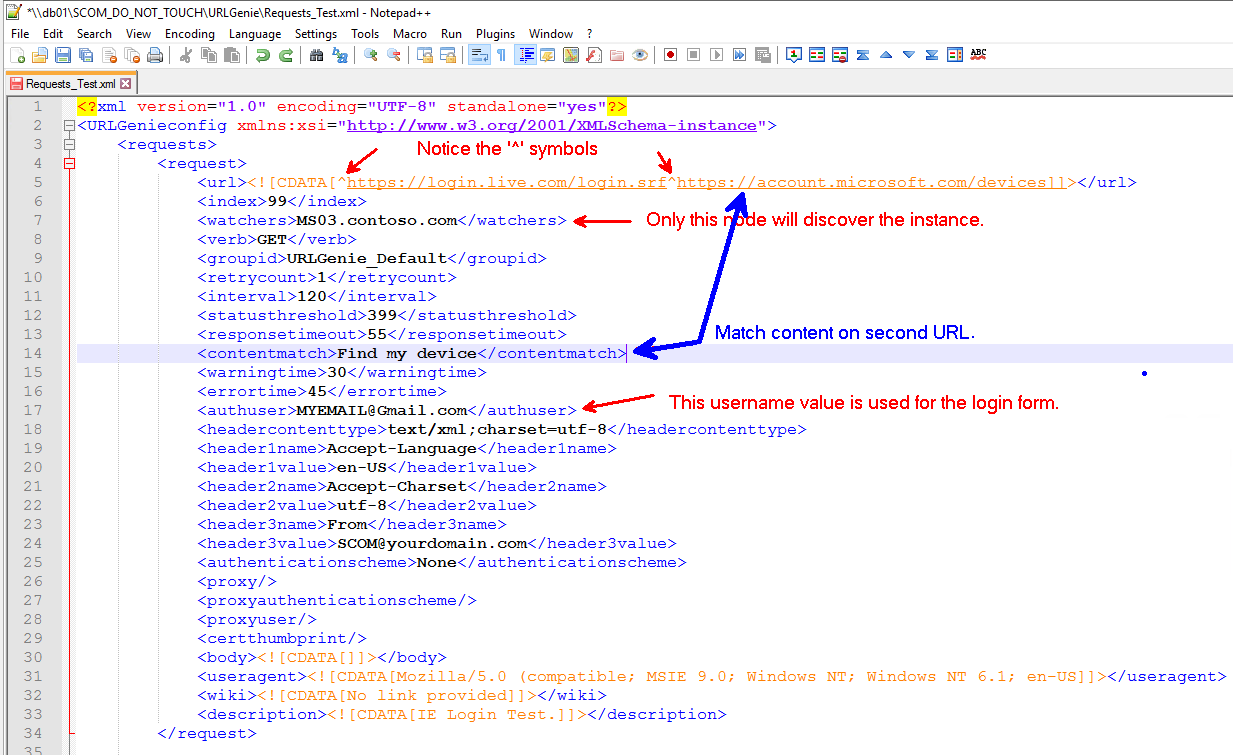
Notice how this site utilized two separate pages for the username and password. No problem!

### 1) Create the URL Instance

Select a watcher node object from the Watcher Node state view. Use the task to generate configuration file from list:



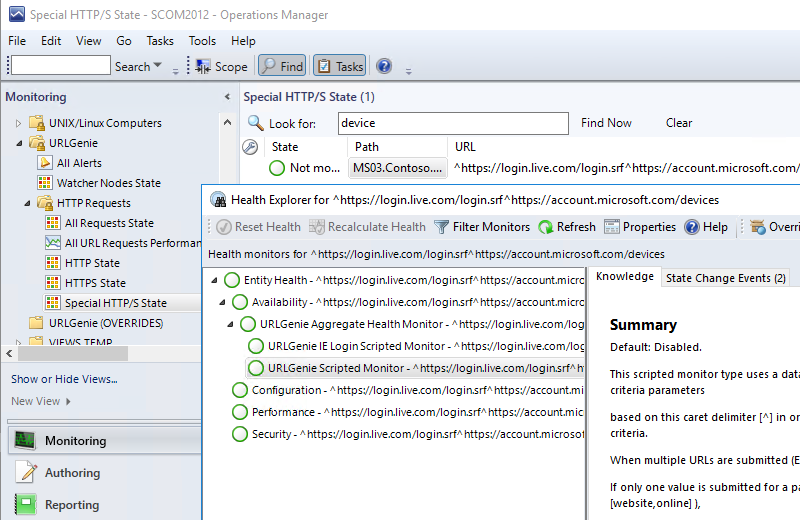
Once the request file exists, I change my watchers value to: *MS03.contoso.com* because I only want that one specific server to discover the URL and run the login test. If I wanted both of my management servers to run the workflows I could leave this as “MS” so both servers (ms02 and ms03) would discover the URL, creating two instances of the same URL. However both servers would need to be configured as described in this tutorial and it’s not a small amount of work to get the settings dialed in.



Notice that I added a caret ^ to the beginning of the URL address. A leading caret is necessary for a single URL when you want it to be discovered as “special” so that normal monitoring does not apply. The example above already has a caret near the middle which separates the two URLs. Therefore, the leading caret is not necessary, but it doesn’t hurt anything and is used here for illustration purposes only.

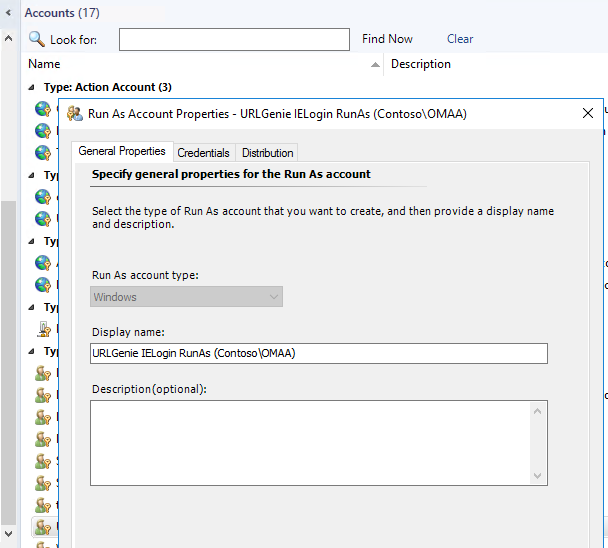
Notice that my web page login username value is input into the configuration file.

When the discovery workflow runs, the caret will be recognized and the URL will be identified (discovered) as an instance of the *URLGenie.HttpRequest.Special* class . ALL monitoring is disabled for this class. This is by design because this URL is special. In this example we must use an override to enable and configure the ScriptedIE Login monitor to test this URL. The necessary overrides are explained further below. Also shown in the image above, the content match value is applicable to the second URL only (if two exist).

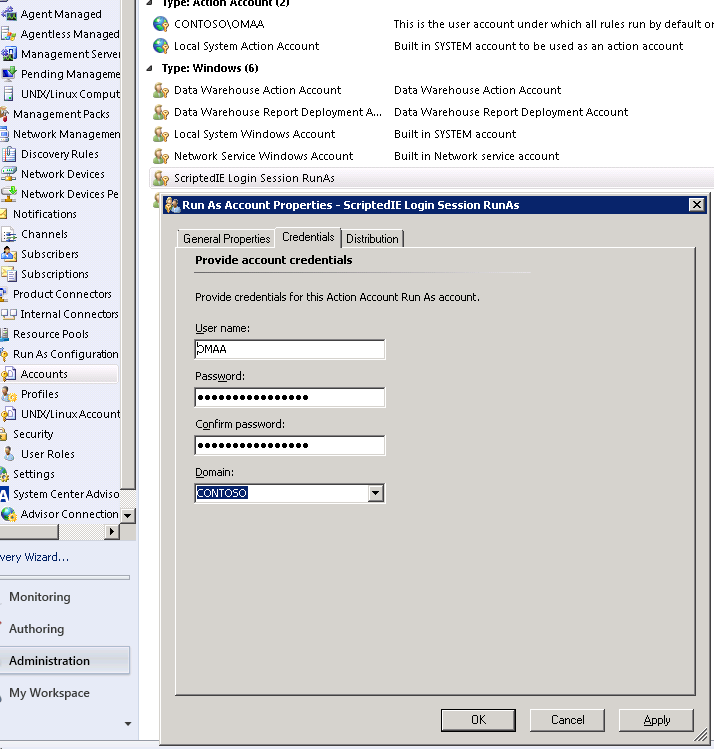


### 2) Create RunAs Account for the Scripted Internet Explorer Session

Create a Windows user account for which to run the ScriptedIE Login workflow. This is the context under which the scripted (Powershell) Internet Explorer session will run. This account must exist in the local Administrators group on the watcher node. I chose to use the OMAA account because this account is already a member of the local Administrators group on my watcher node. I’m using my management server in this example. The management servers have a much higher capacity for URL monitoring than ordinary agent managed machines.

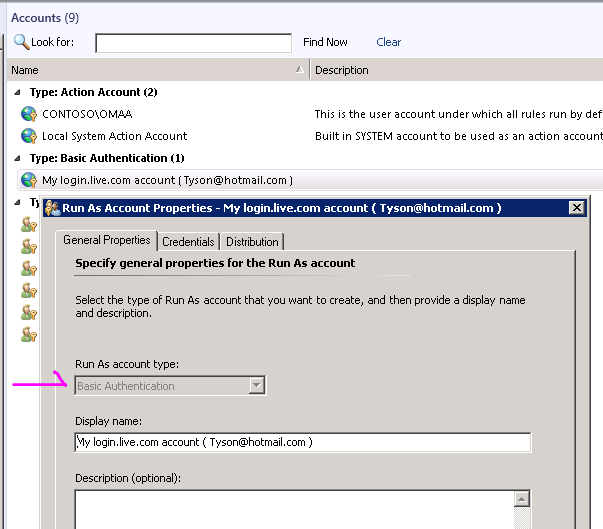


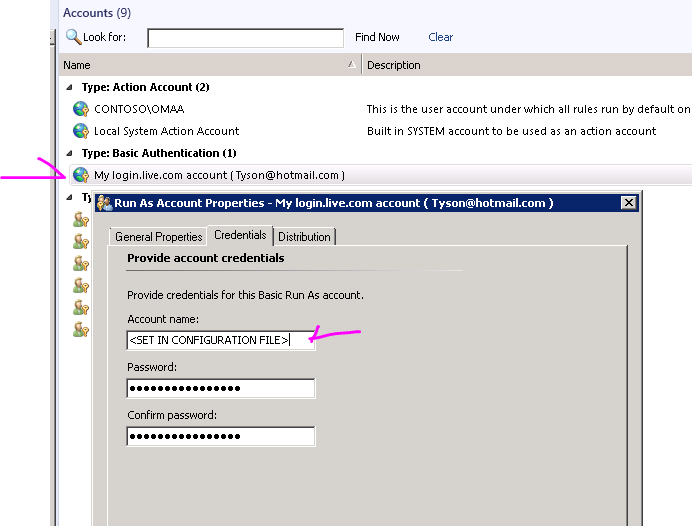
Don’t forget to distribute the RunAs account to your watcher node(s).



### Create the Credentials for the Target Web Page Login Form(s)

Now create the **Basic** account credential with the password that would normally be used in the “Password” field on the target web page login form. The username doesn’t matter here because the username used in the monitoring login test comes from the configuration file “authuser” field.

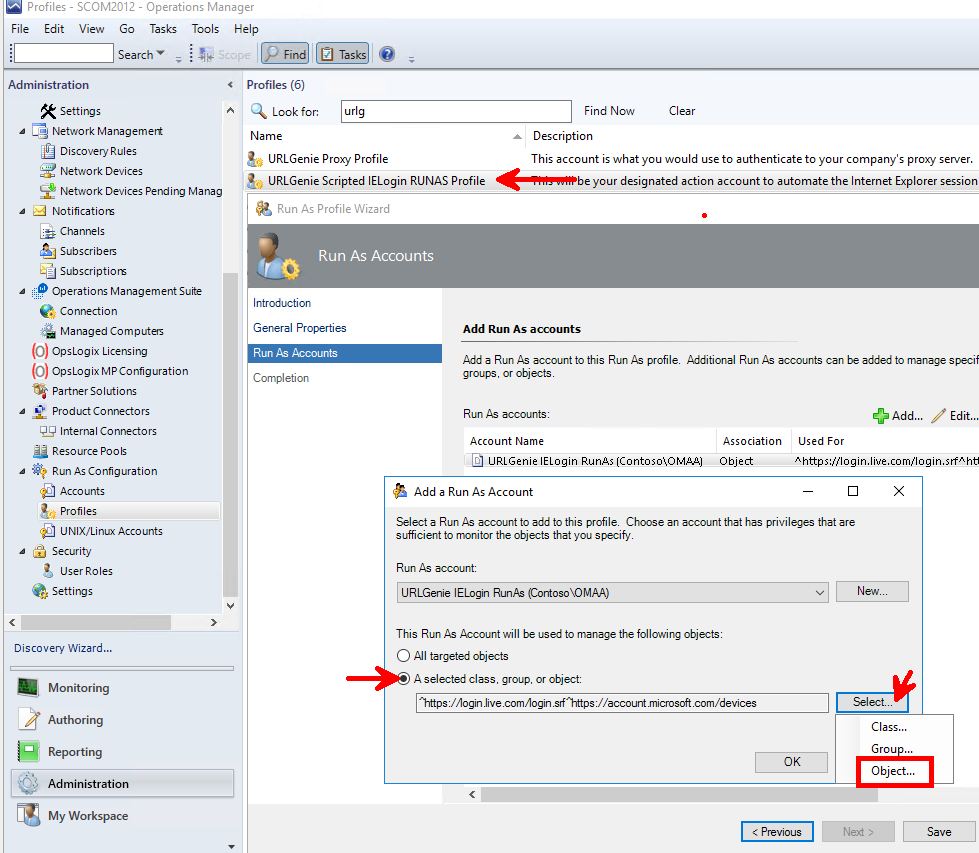




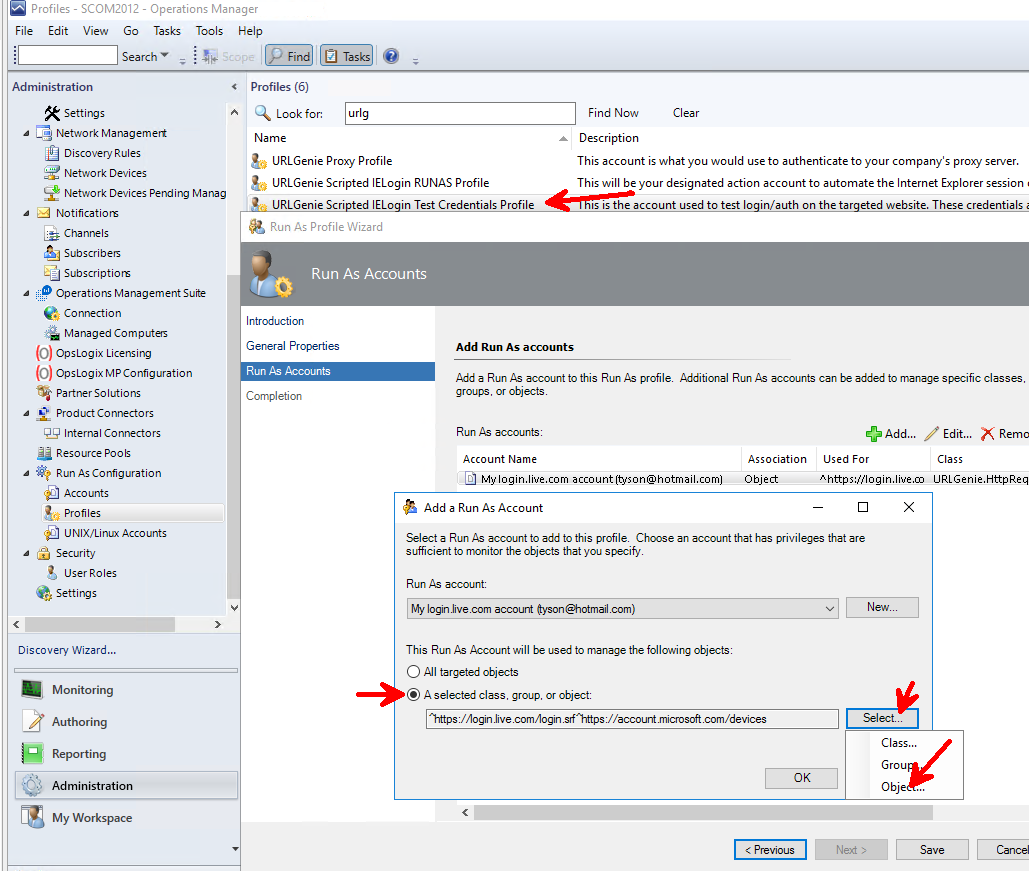
The “Account name:” field doesn’t matter here. This value is not used in the workflow. The username for the login form comes from the configuration file. You only need to make sure the password is correct.

### Configure Security Profiles

Add the ScriptedIE Login RunAs account to the correct security profile: *URLGenie Scripted IELogin RUNAS Profile*. Remember, this is the account that will be executing the workflow (which happens to be a PowerShell module) and it needs to be a local administrator on the watcher node. This is necessary because the workflow uses WinRM to connect to itself (yes, to Localhost) to create the PowerShell session which is capable of launching an internet explorer COM application.



Add the ScriptedIE Login Test account to the correct security profile: *URLGenie Scripted IELogin Test Credentials Profile*. Remember, this is the account which is used for the password field on the login page(s).



### Configure the Watcher Node

#### 1) Log into the Watcher node

Use your designated action account to login to the watcher node where the ScriptedIELogin monitor AND/OR performance rule will run. This will require the action account to have local administrator rights to the machine. In my example I used the OMAA account which is already a member of the Local Administrators group on my watcher node.

(On Windows Vista, and later versions of the Windows operating system, to use the ComputerName parameter of **Invoke-Command** to run a command on the local computer, you must open Windows PowerShell by using the **Run as administrator** option. <https://technet.microsoft.com/en-us/library/hh849719.aspx> )

#### 2) WinRM must be enabled on the watcher node(s)

Open a Powershell command prompt (as Administrator) on the watcher node(s), type the following:

**Winrm quickconfig**

Press (y)es to accept the configuration.

( <https://msdn.microsoft.com/en-us/library/aa384372(v=vs.85).aspx> )

#### 3) Give permission: "Act as part of the operating system"

A. On the watcher node: Type the following command in an elevated command prompt:

**secpol.msc /s**

B. Select "Local Policies" in MMC snap-in

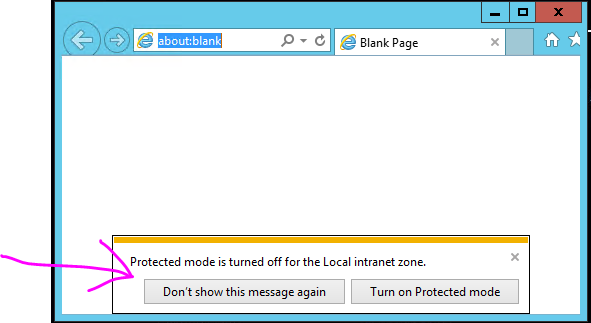
C. Select "User Rights Assignment"

D. Add the SCOM Default Action Account to "Act as part of the operating system"

#### 4) Configure Internet Explorer.

The goal for this portion of the configuration is to be able to open IE and login to the target web page without ANY prompts, popups, or nag screens. Any prompts by the operating system or browser can adversely affect the scripted automation of the login test and cause the monitor to fail. Notice the popup prompt in the screenshot below. You must manually satisfy the popup prompts so that they do not appear again.

**Launch Internet Explorer as the RunAs account** (OMAA in this example) which has been configured in the URLGenie Scripted IELogin RUNAS Profile for the URL instance.



In my lab I enabled Protected Mode.

##### A) Advanced tab

Set the following in the *Internet Explorer -> Internet Options -> Advanced* tab

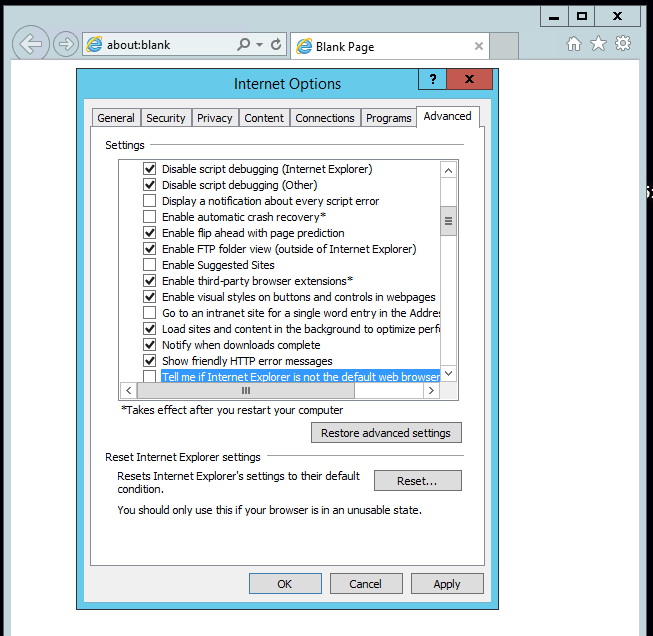
CHECK: Disable script debugging (Internet Explorer)

CHECK: Disable script debugging (Other)

UNCHECK: Display a notification about every script error

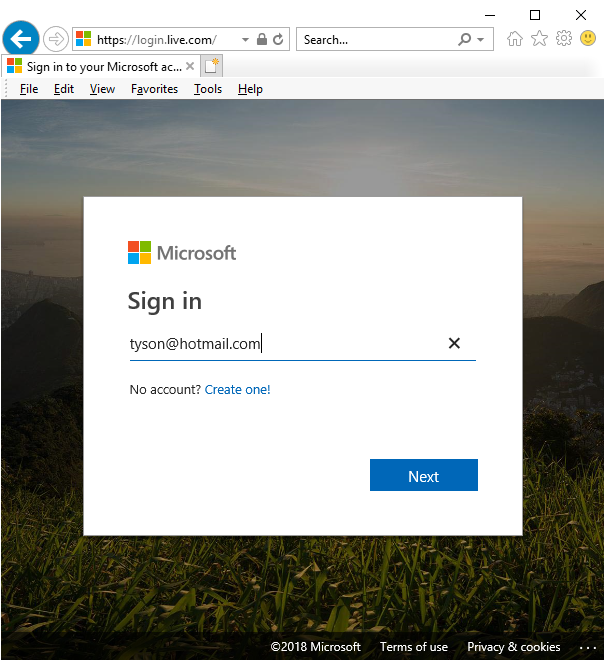
UNCHECK: Enable automatic crash recovery

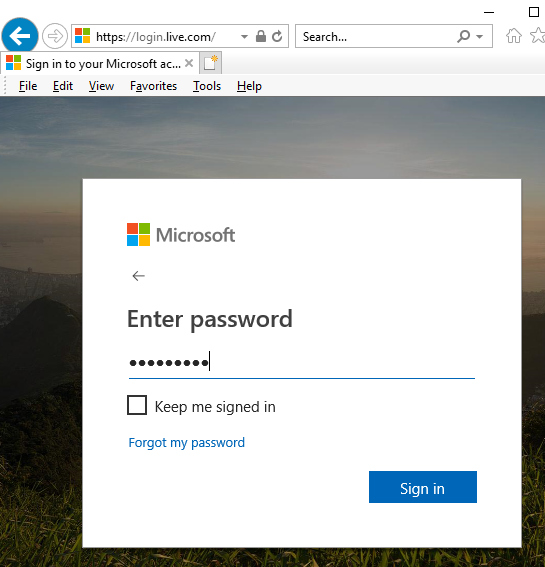
UNCHECK: Tell me if Internet Explorer is not the default web browser



##### B) Test Login, Satisfy Prompts

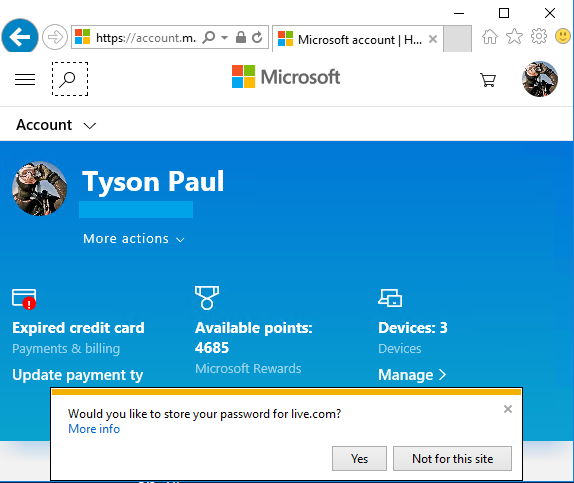
Open Internet Explorer and Navigate to the target URL. Input the username and password and proceed to login. Make note of any popup windows or prompts/notifications. These types of popups can cause the scripted monitor to fail.



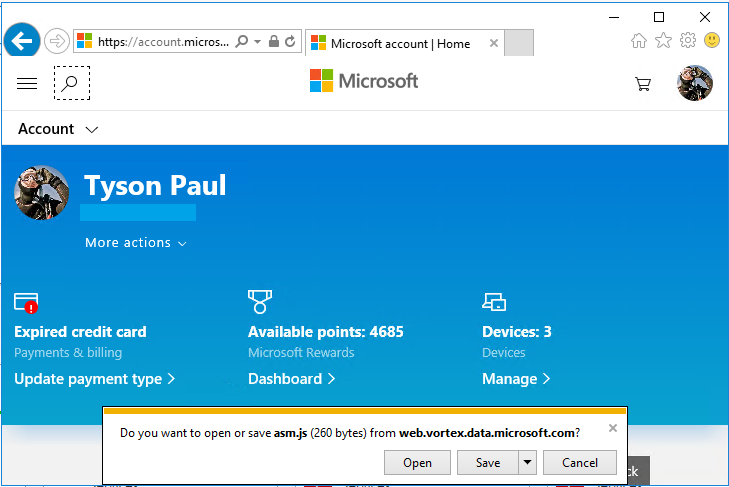


Make sure to satisfy any popups that may occur during your login test.

Examples of prompts that you may see:



No, you don’t want to store your password.



This was a strange one. It only appeared once. I closed the window, logged in again, and never saw the prompt again.

Another example: Upon logging in you may see a popup warning/prompt: "Do you want to view only the webpage content that was delivered securely?" (you don’t). This type of popup may prevent the page from fully loading with the scripted sequence and therefore the monitor may fail to match the content string. You must disable this popup with the following steps:

Go to *Tools->Internet Options->Security* tab

Select the **Trusted Sites** zone

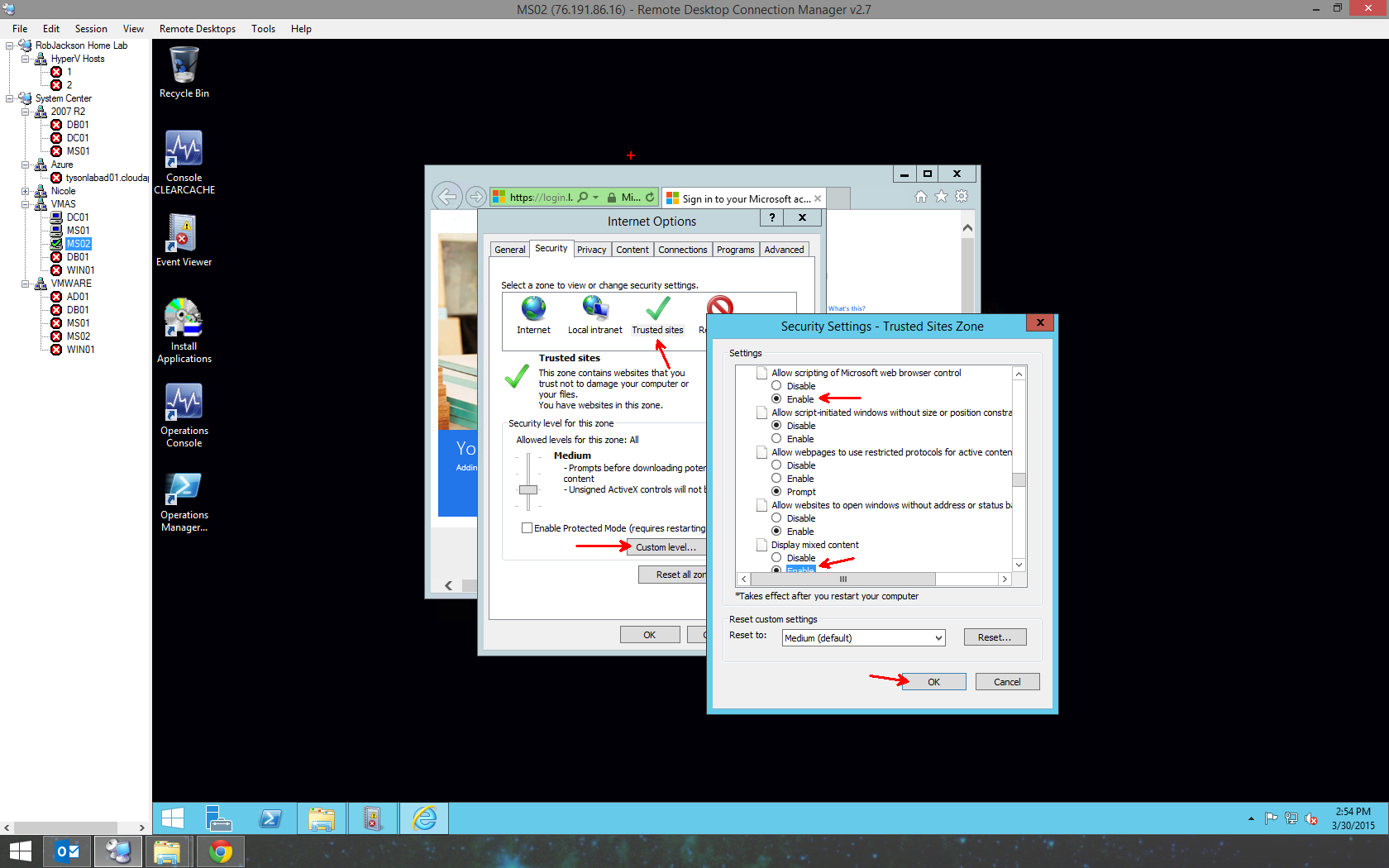
Click the **Custom Level** button

In the **Miscellaneous** section change:

"Allow scripting of Microsoft web browser control" to "Enable"

"Display mixed content" to "Enable"

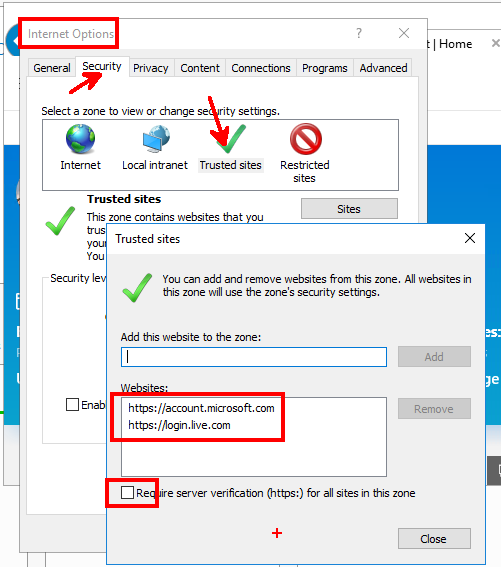
Click “OK” to accept changes.



Click the **Sites** button

Now add the site to the Trusted Sites Zone)

(UNcheck the "require https..." option if necessary)



Notice that I added the initial “login” page and also the next page after successful login because it’s a different domain. Any domains included in your URL (or URL sequence if using two URLs; a login page followed by a target page on which to match the content string ) must be added to the Trusted Sites list.

##### C) REBOOT THE WATCHER NODE

##### D) Complete a Flawless Login Test

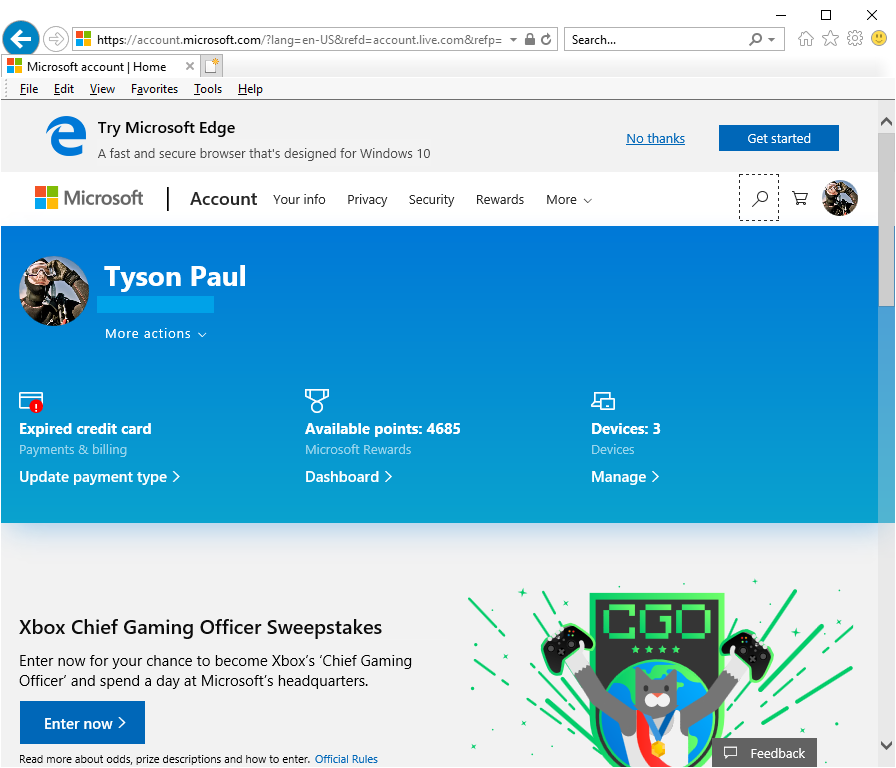
Run through the login steps once more while logged into the machine as the designated RunAs account (Contoso\OMAA in my example). Don’t simply launch IE as the user account because IE will behave slightly differently in this scenario. You want to be logged into the machine to conduct this verification step.

**NOTE:** Make sure you can login to the watcher node (using the designated action RunAs account in the "URLGenie Scripted IELogin RUNAS Account") and launch Internet Explorer and navigate to the login test page without being prompted by ANY popup windows. If you encounter any popup windows, make the necessary configuration changes in IE to prevent these popups. Otherwise the monitor workflow will likely fail and it is very, very difficult to troubleshoot.

Note: in this example the “test” page is the second URL in our sequence:

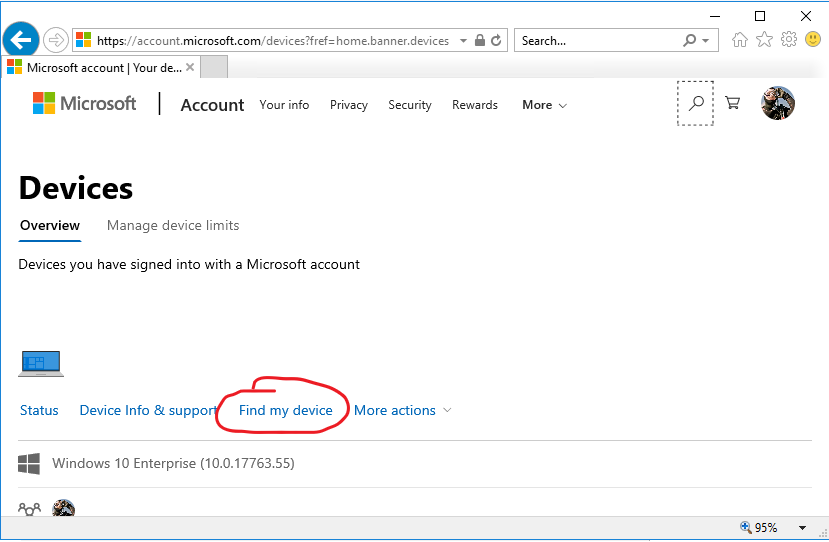
[https://login.live.com/login.srf^**https://account.microsoft.com/devices**](https://login.live.com/login.srf%5ehttps://account.microsoft.com/devices)

Here’s the initial landing page after logging in successfully…



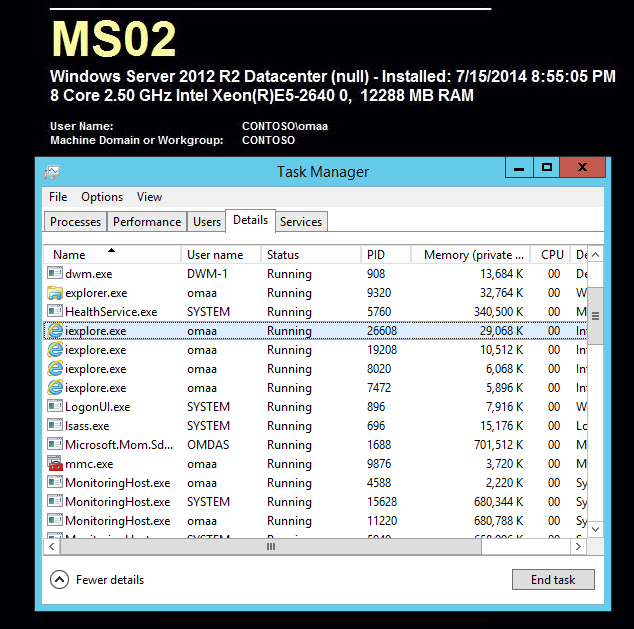
Here’s the actual “test” page, the second URL…

Notice the circled text: “Find my device”. This is our ContentMatch string in the configuration file.

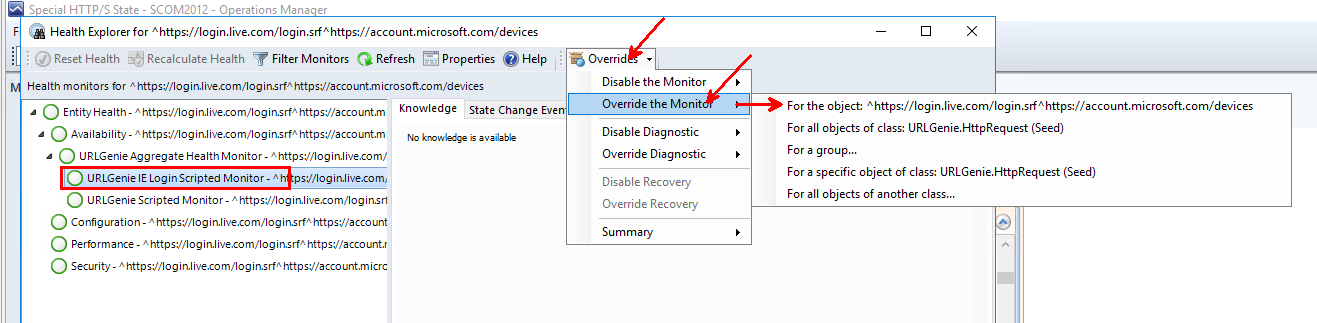


WARNING: This type of monitor is experimental. I've had great success in my lab environment and in a handful of production environments, but this monitor type is not perfect. It simulates user interaction with the web browser. Any interactive prompts resulting from unpredictable configurations will produce unreliable results. After you configure an instance for ScriptedIE Login, be sure to observe the processes on the watcher node and watch out for IExplore.exe processes that grow and don't terminate. If you see the number of IExplore.exe processes continually growing (over a few hours or days) then disable the monitor and recheck the configuration steps outlined above. You probably missed a step and the IE object is getting hung up on an interactive prompt which will be hidden to any logged in user(s). It's not unusual to see 2N+2 IExplore.exe processes in the Task Manager, where N represents the number of instances configured to run on the watcher node. Example: if you have two URL instances configured to use the ScriptedIELogin monitor on a given watcher node then you may see a total of 6 IExplore.exe processes in the Task Manager processes list. This is expected and normal behavior.

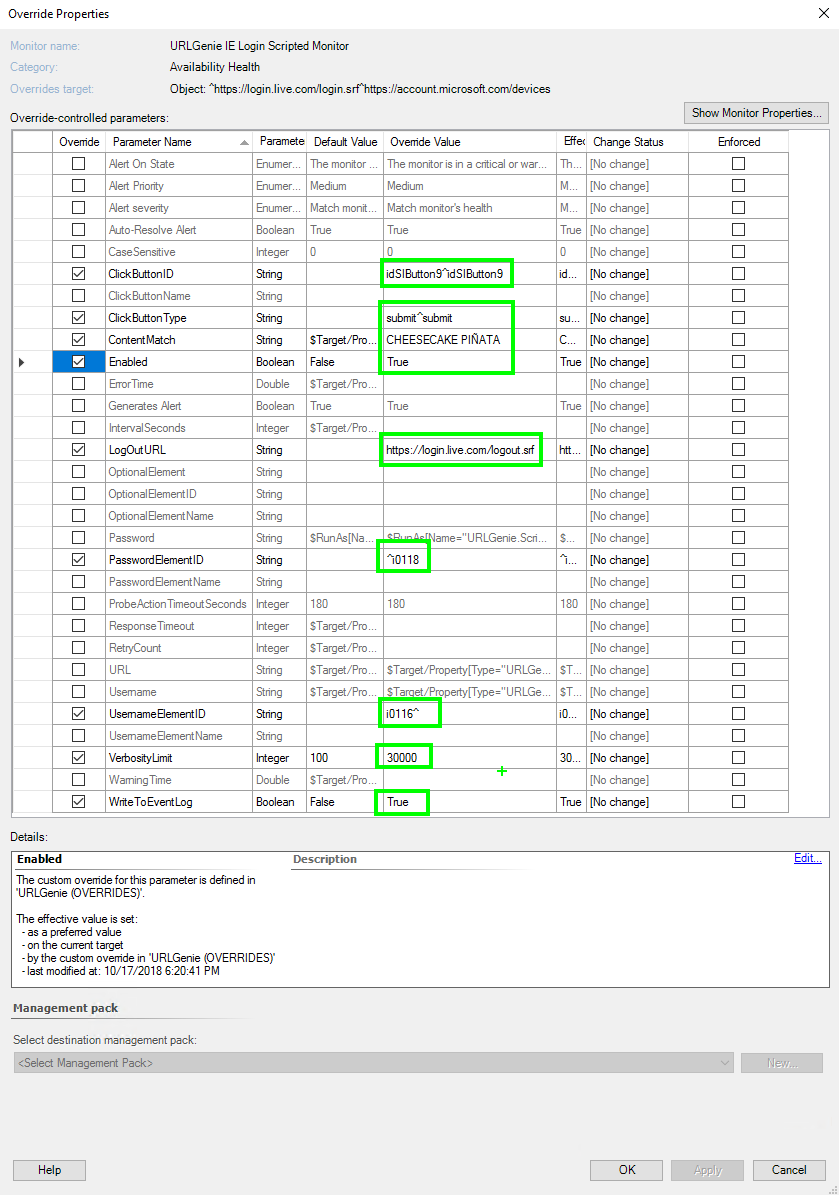
Watching



### Override the Monitor with Login Parameters



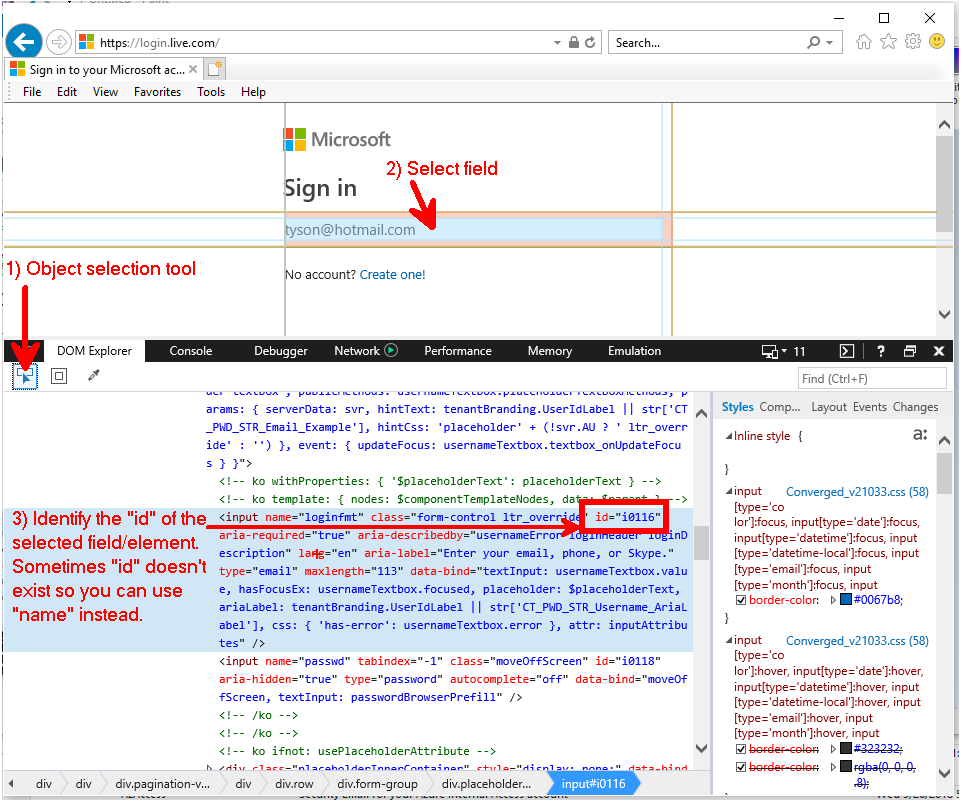
The following values will need to be overridden for the ScriptedIELogin monitor :



In order to determine the correct override values, we need to inspect the login pages.

You will need to find the HTML IDs of the elements designated below. You can find the element IDs by enabling “developer mode” ( Internet Explorer: F12, Chrome: CTRL+SHFT+I ) and selecting the objects to find the section of source code that defines the element properties.

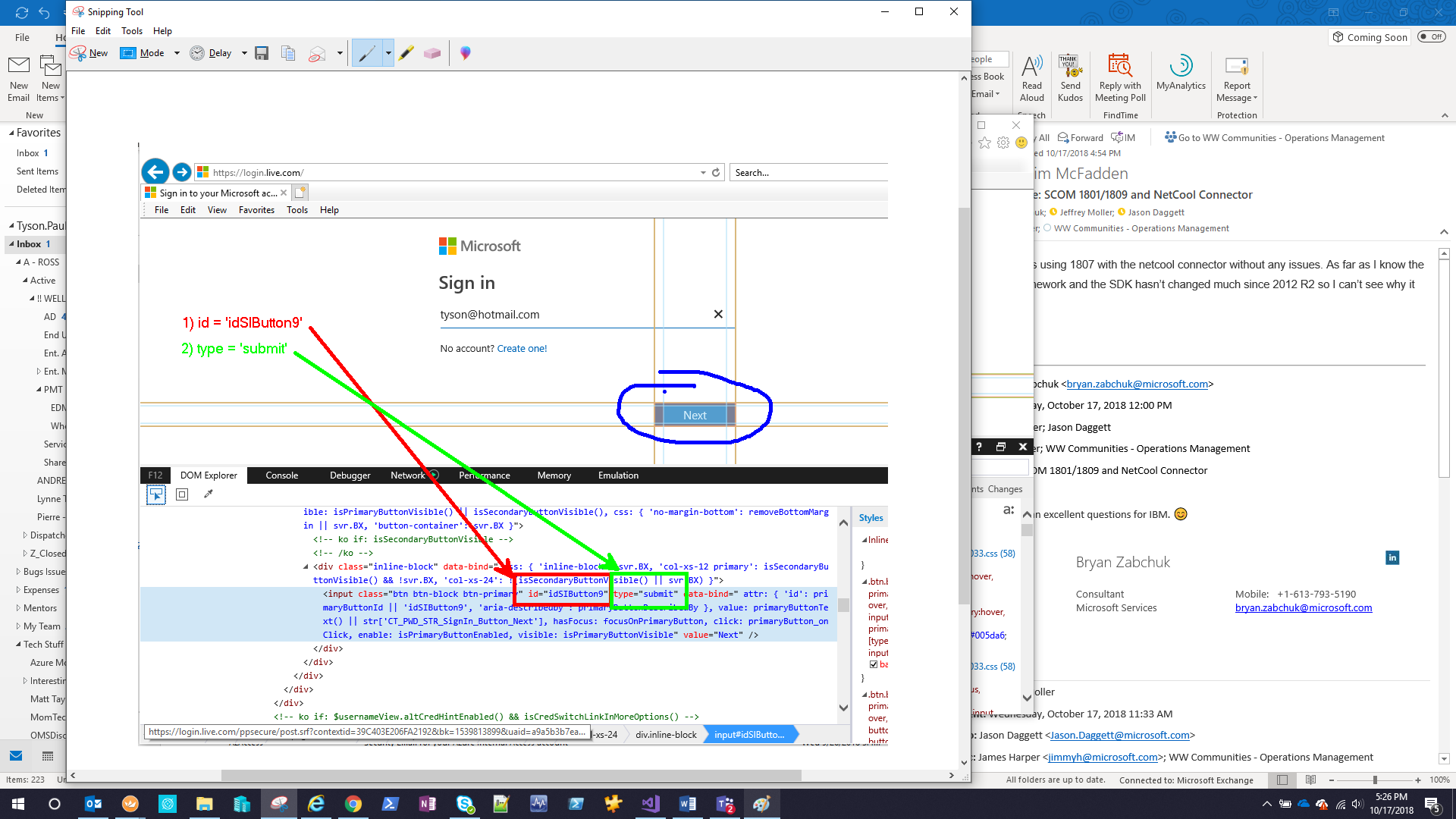
Developer Mode (Internet Explorer)



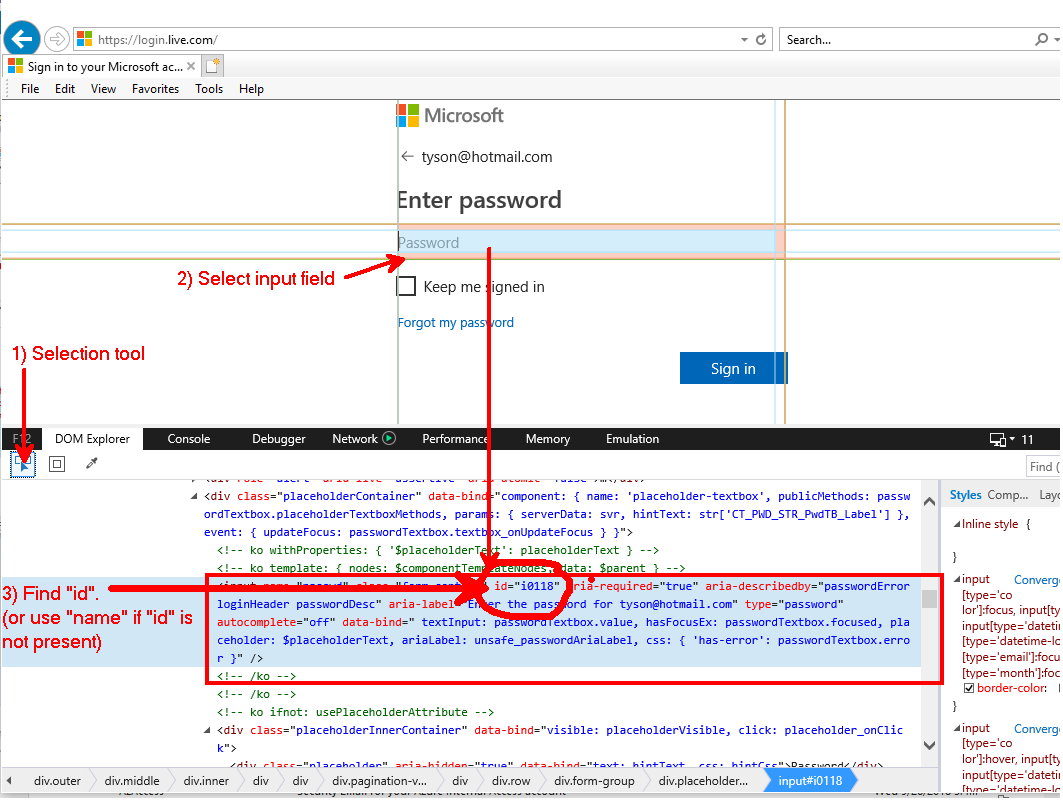
Our Username field ID = **i0116**

NOTE: Because this site uses TWO pages (one for username, one for password) we need to simulate two “submit” actions.

We need to identify the ID and type of the first “Next” submit button as well as the second “Sign in” button.   
starting with the “Next” button …

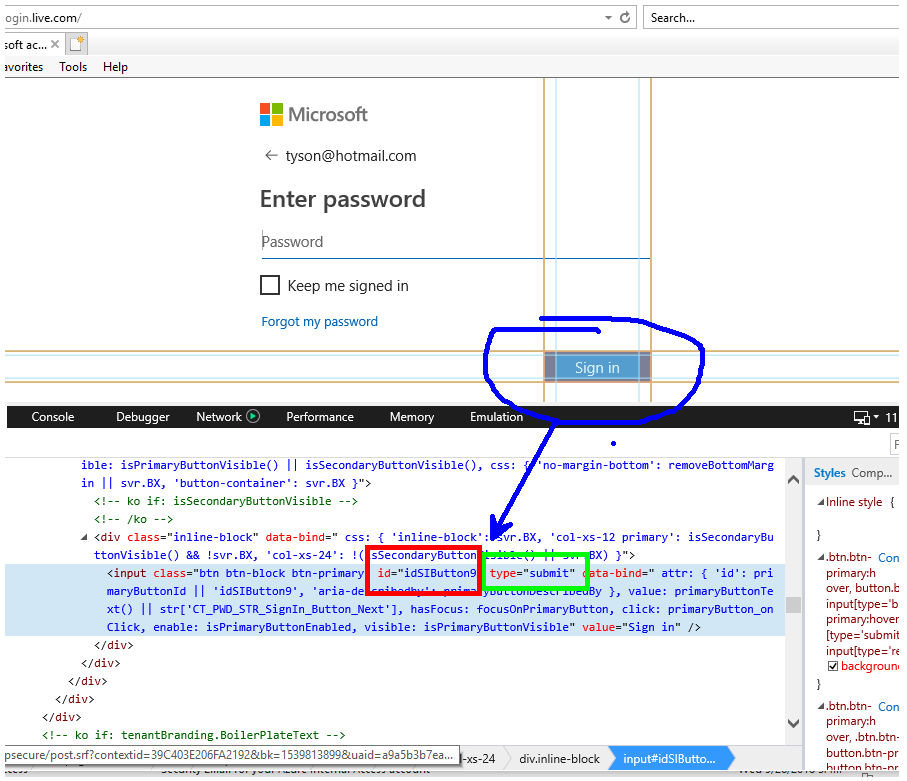


Id = **idSIButton9**  
type = **submit**



In this case the Password field ID is: **i0118**

Do the same for the “Sign in” button. This is the second button that submits the password.

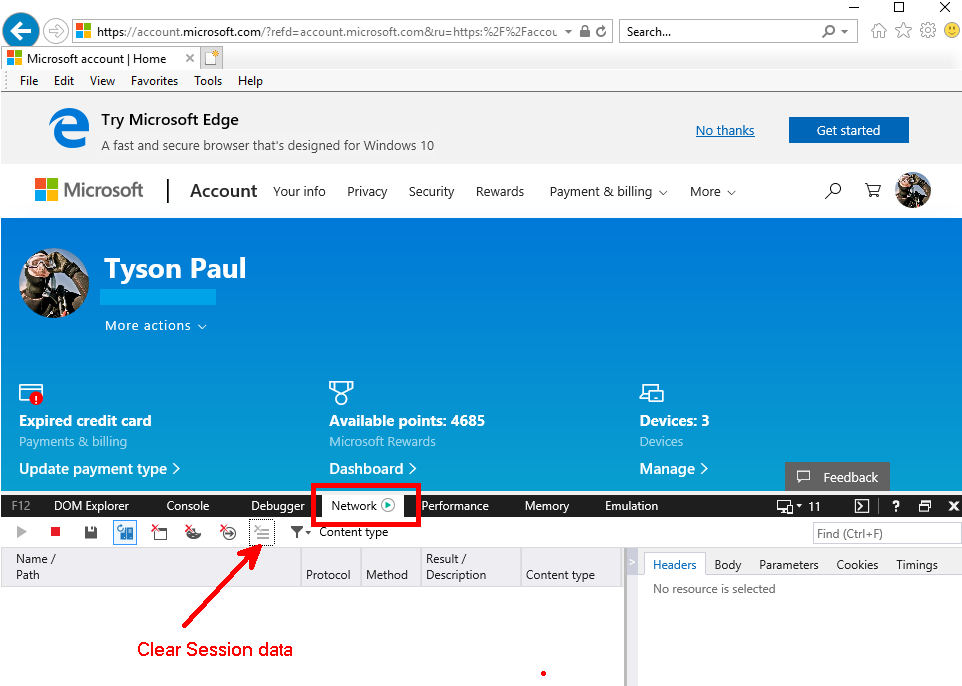


Id = **idSIButton9**  
type = **submit**

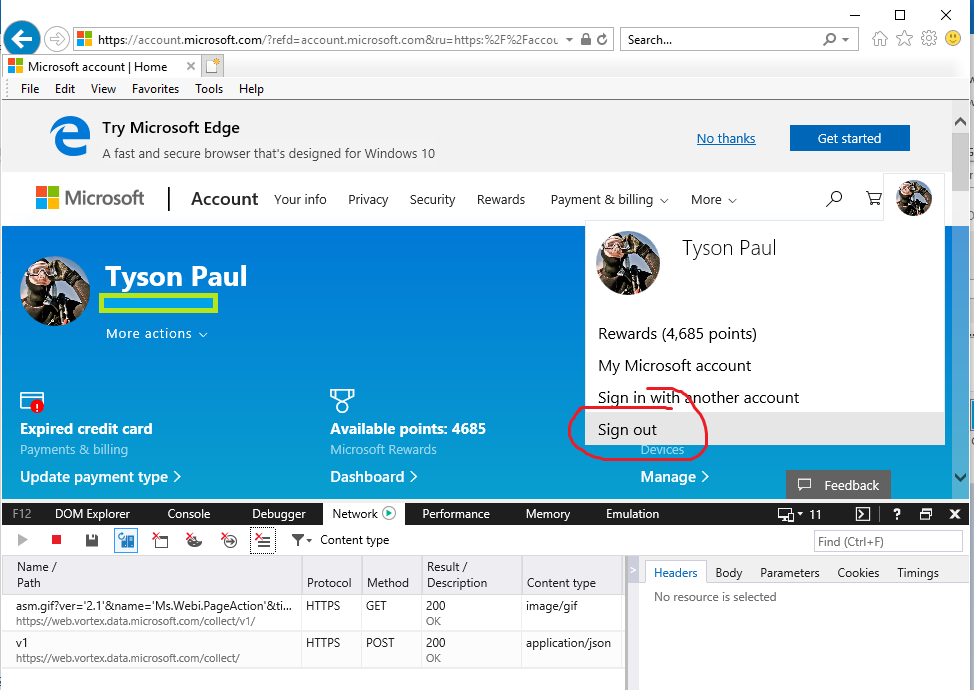
As it turns out, the ID and TYPE are the same as on the previous page. No problem. Keep reading…

Finding the logout URL is a little more tricky. We need to watch the “Network” logging tab when we attempt to log out to see which URL is used for the logout action.

First, let’s clear the existing entries with the “Clear Session” button on the Network tab…

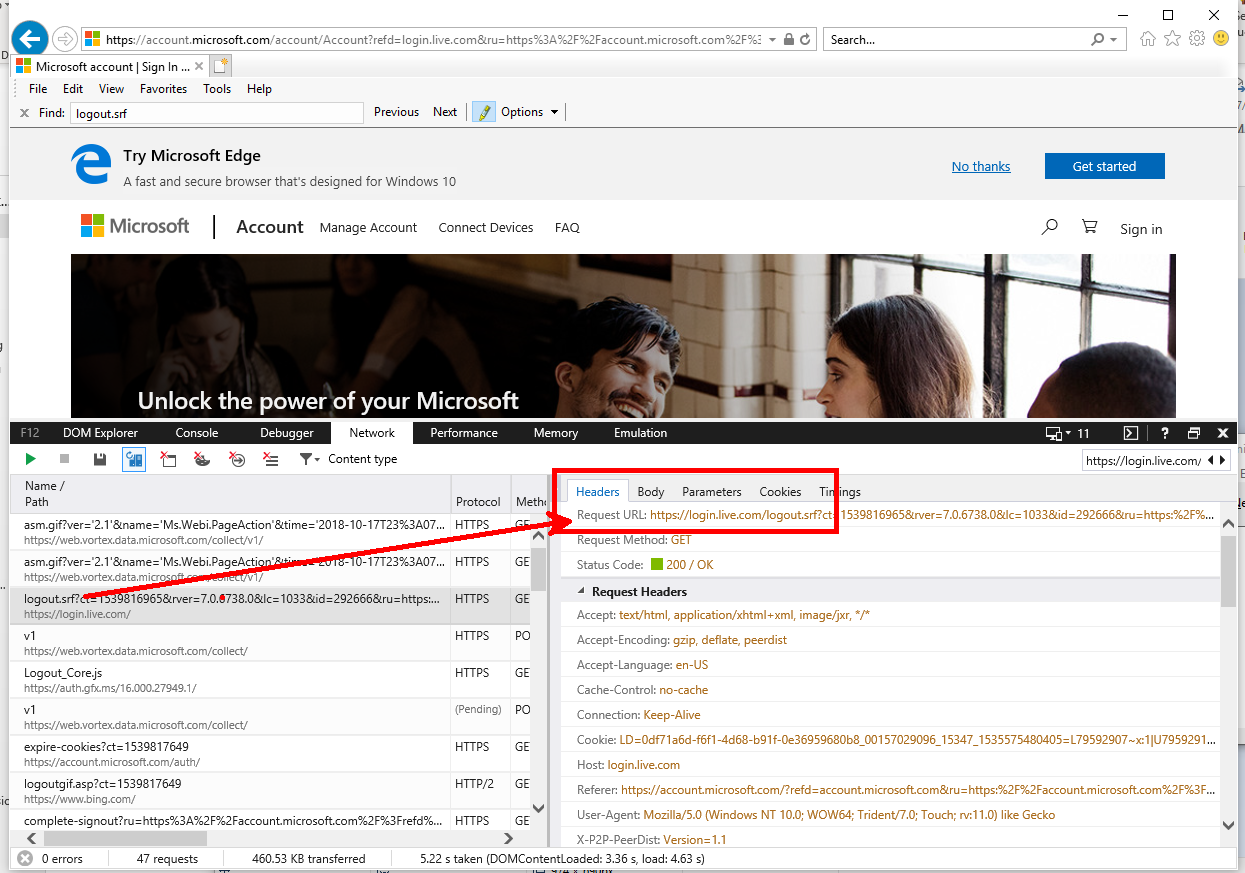


Then we attempt to sign out…



Click the “Sign out” link and watch for activity in the Network tab area.

I look for anything that resembles “logout”. Then log back in and test (copy/paste) that logout URL to see if it will log you out successfully. In this case, it is the correct logout URL. Be sure to thoroughly test your URL.



Logout URL = **https://login.live.com/logout.srf**

Let’s putting it all together. Here are the values that we collected so far:

Username ID: **i0116**  
first submit button ID: **idSIButton9**first submit button type: **submit**

Password ID: **i0118**  
second submit button ID: **idSIButton9**second submit button type: **submit**

Log out URL: **https://login.live.com/logout.srf**

Our override values:

These first few values seem like they should be straightforward however because the datasource is designed to expect up to two login pages (one for username, another one for password) the username and password values have to be separated with the caret, even if there is only one value. The reason is because the scripted datasource will attempt to determine how many login pages to expect based upon how many values you are providing for the various button properties. In this example, we are providing data for two submit buttons (**idSIButton9^idSIButton9**), therefore the datasource will expect two separate screens with forms and buttons. The values get split up based on the caret ‘^’ symbol. Therefore the UserNameElementID value, ‘**i0116^**’, would become interpreted as a two cell array shown in the red box below:

|  |  |  |
| --- | --- | --- |
| Column | 0 | 1 |
| Value | **i0116** | <nothing/blank> |

in programming, arrays begin with 0 (zero).

The first value, $array[0], is ‘i0116’. The second value, $array[1]is empty.

For the PasswordElementID value, ‘^i0118’, when the data is split it becomes the two cell array shown in red below:

|  |  |  |
| --- | --- | --- |
| Column | 0 | 1 |
| Value | <nothing/blank> | **i0118** |

ClickButtonID would end up like this:

|  |  |  |
| --- | --- | --- |
| Column | 0 | 1 |
| Value | **idSIButton9** | **idSIButton9** |

The indices, 0 and 1, correspond to the number and order of the login pages expected. Therefore on the first login page, because the first column/cell (column 0) for UsernameElementID has a value (**i0116**), the script will find the username field with the corresponding ID and populate it with the value. Because the first column/cell (column 0) for PasswordElementID is empty/blank, it will be skipped. The script would locate the submit button matching the value in column 0 (**idSIButton9**) and simulate a click action on the button.

For the second login page, the second column (column 1) will be used for each field; UsernameElementID, PasswordElementID, and ClickButtonID. Therefore the caret plays an important role in helping the script determine when to locate and populate each of these fields.  
this guideline applies to the other alternative element values which can be used when no proper “ID” is assigned to the login page fields

These become the values which are used on each of the two login pages to identify the HTML fields:

|  |  |  |  |
| --- | --- | --- | --- |
| **Login Page** | **First** | **Second** | Notes |
| Username field ID | i0116 | <blank> | No password field exists on the first page |
| Password field ID | <blank> | i0118 | No username field exists on the second page |
| Submit button ID | idSIButton9 | idSIButton9 |  |

These become our override values:

UsernameElementID: **i0116^** (this value is used on the first login page only)  
PasswordElementID: **^i0118** (this value is used on the second login page only)

LogOutURL = **https://login.live.com/logout.srf**

ContentMatch = <something unlikely> (Set this temporarily while configuring the monitor. Once settings are verified, remove this override.)  
Enabled = **True**WriteToEventLog = **True** (Set to True temporarily while configuring the monitor. Once settings are verified, remove this override.)

For ClickButtonID and ClickButtonType , because we have two of each, we will combine them with the caret ‘^’ as shown below. The scripted datasource is smart enough to enumerate these values and use them in order as they are needed.

ClickButtonID = **idSIButton9^idSIButton9** (these values get used on the first and second login pages respectively)ClickButtonType = **submit^submit** (This value isn’t always necessary but it is helpful in cases where the ID is not available or where there might be multiple buttons expected with the same ID but different types. Adding this info only increases your chances of success.)

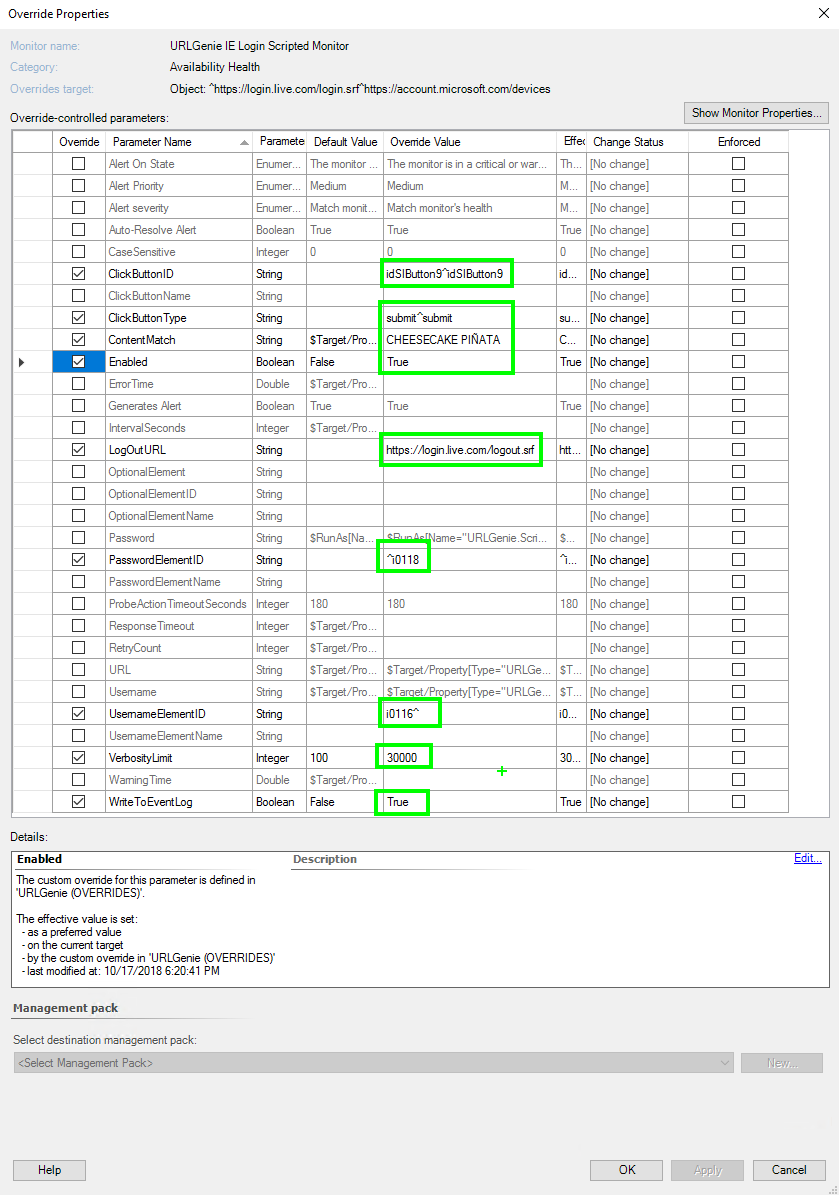
**What about a single login page scenario; one that contains all of the required fields?**

Easy. Just use the single values for the required fields and the script will do the rest.

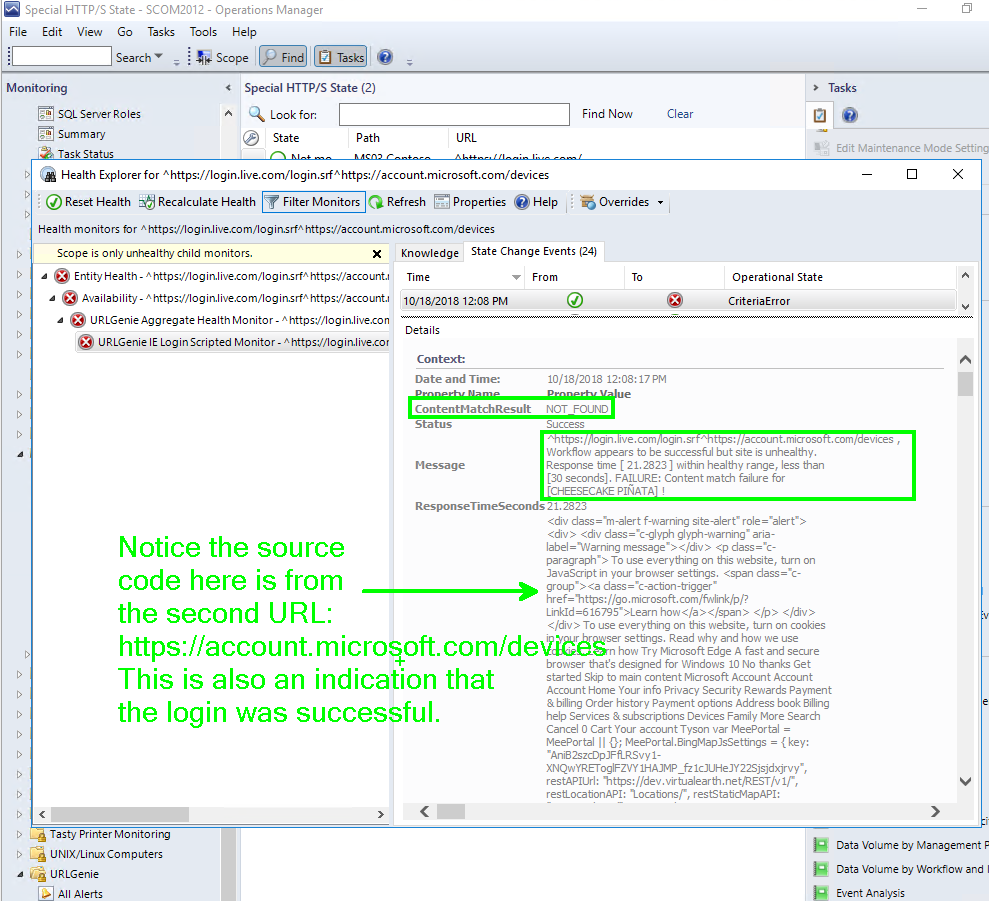
Example:

UsernameElementID = userexampleid  
PasswordElementID = passexampleid  
ClickButtonID = buttonexampleid  
ClickButtonType = submit

Set the override values:

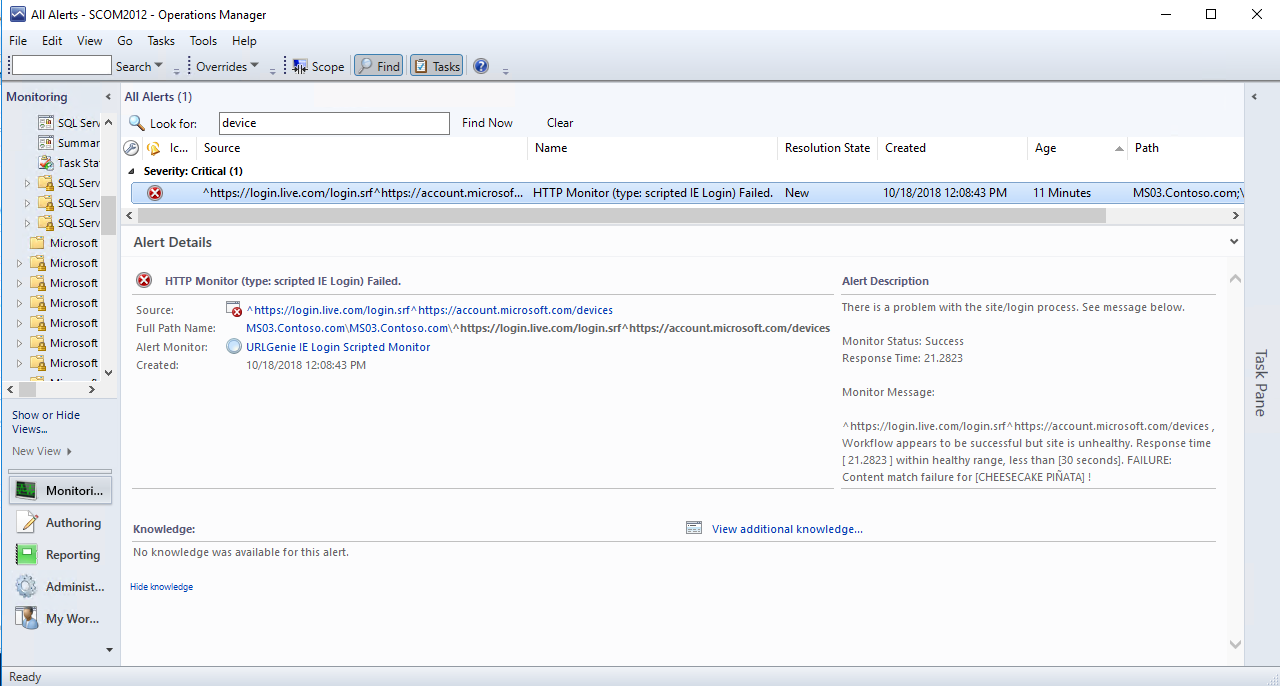


By setting the ContentMatch to something unlikely, I have created a false error condition for the monitor and can now examine the context data to validate that the login is occurring correctly. By looking at the InnerText data retrieved from the web page after logging in, I see text that appears on the expected target page after successfully logging into the site. This is shown in the state change context data area in Health Explorer and also on the Alert Context tab and Description field of the resulting alert.

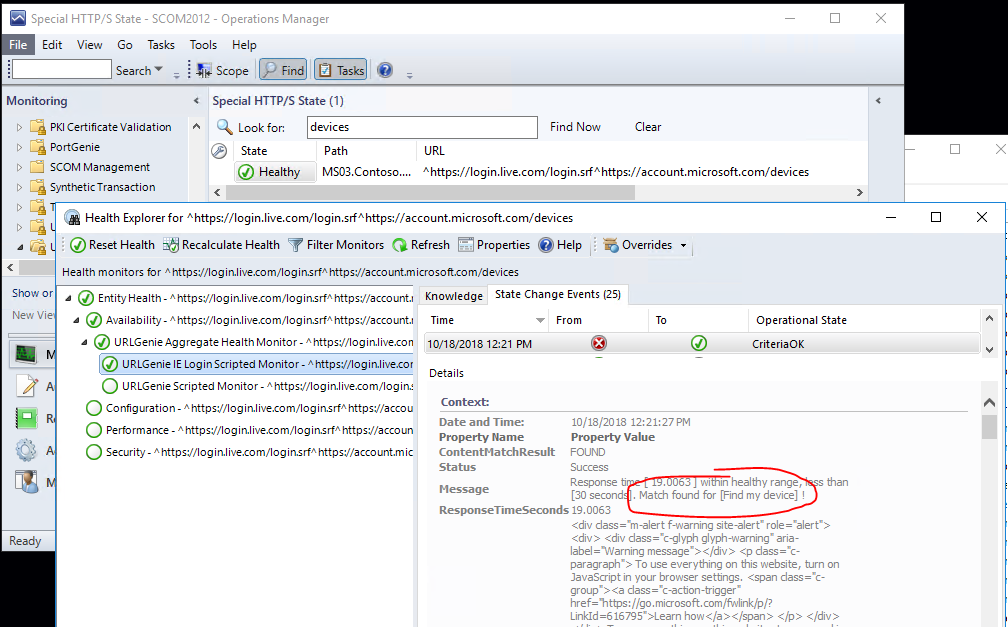


The alert description indicates that the login test was successful but did not find the content match as intended. This is expected behavior due to the ContentMatch override that we put in place for testing purposes. After validating that the content match is working correctly, be sure to correct the override and/or verify your ContentMatch field in the configuration file to text that only appears after logging into the site.

“Find your device” in my example.

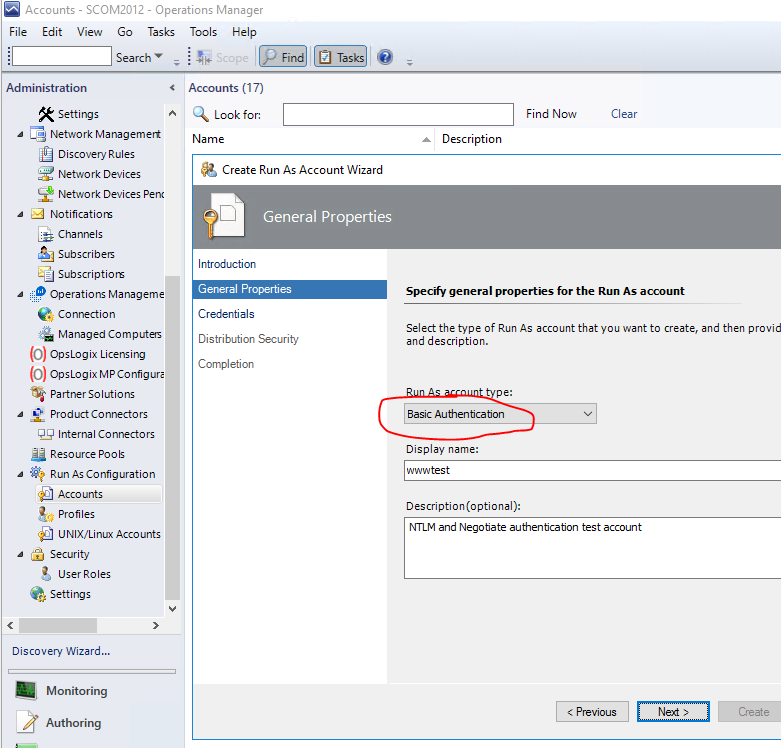


Healthy and Happy!

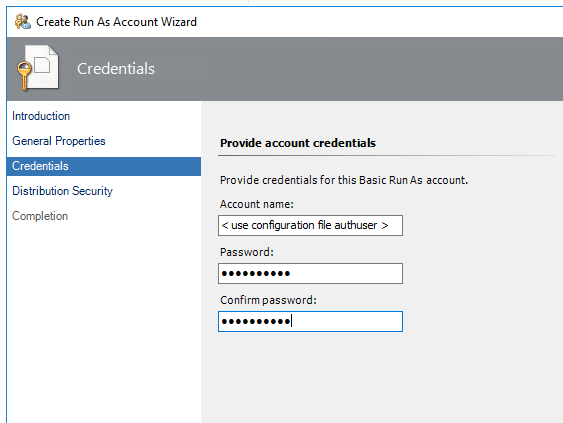


# Use NTLM Authentication

Create a **BASIC** account credential in the Console. (yes, even for NTLM and Negotiate)



Input password.

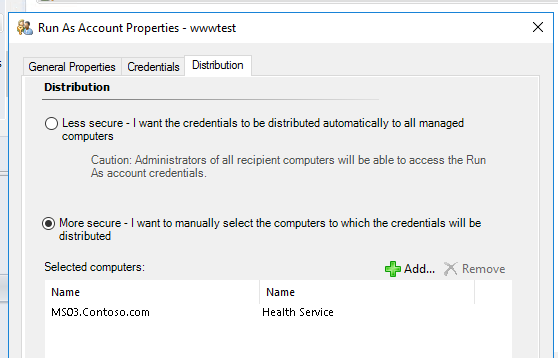


Enter username in config file for the instance. Use <domain>\<username> format:

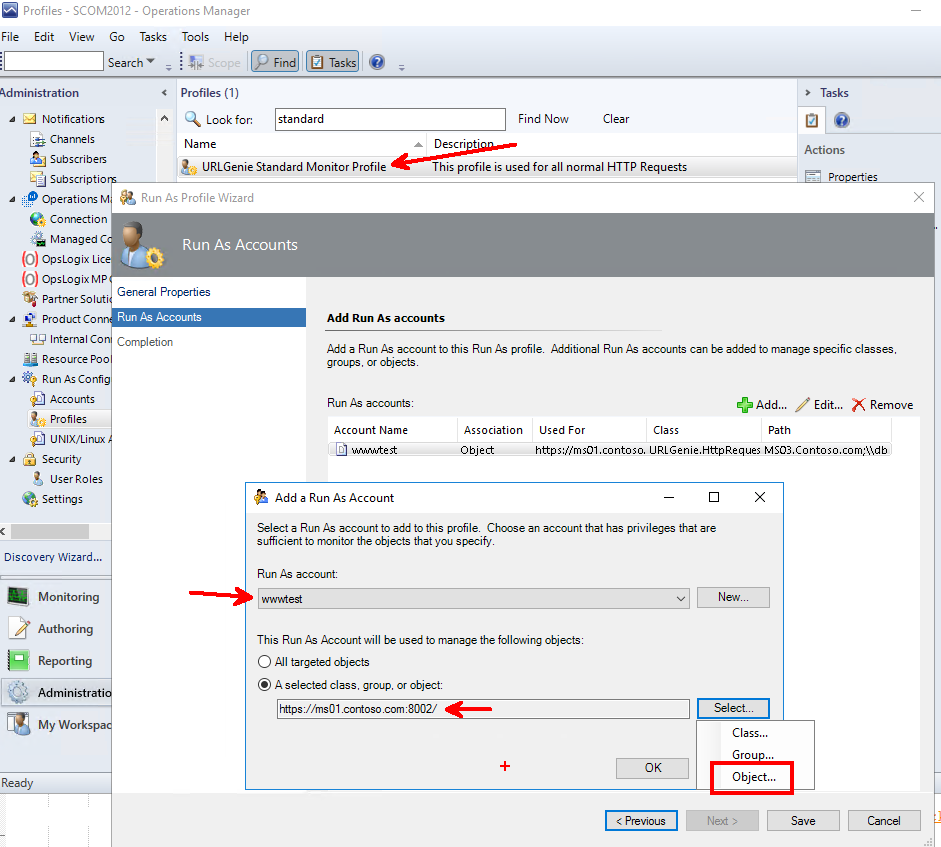


Note the authenticationscheme value. This tutorial is basically the same for **Negotiate** authentication.

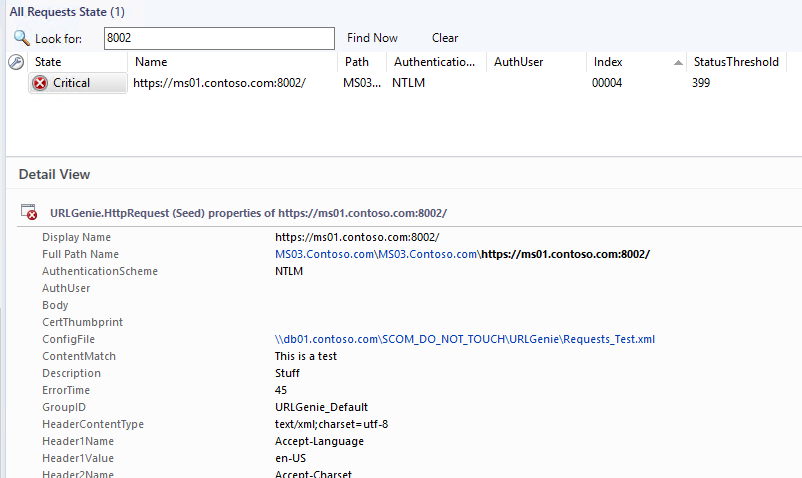
Distribute the account to your watcher node.



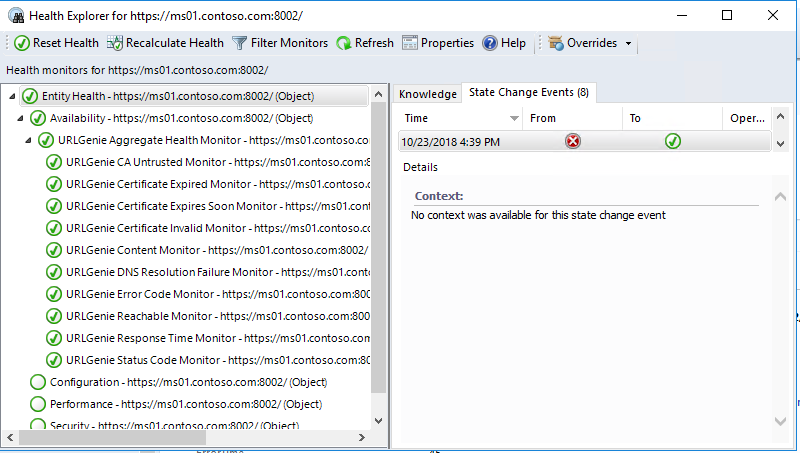
Add the account to the correct security profile: *URLGenie Standard Monitor Profile*

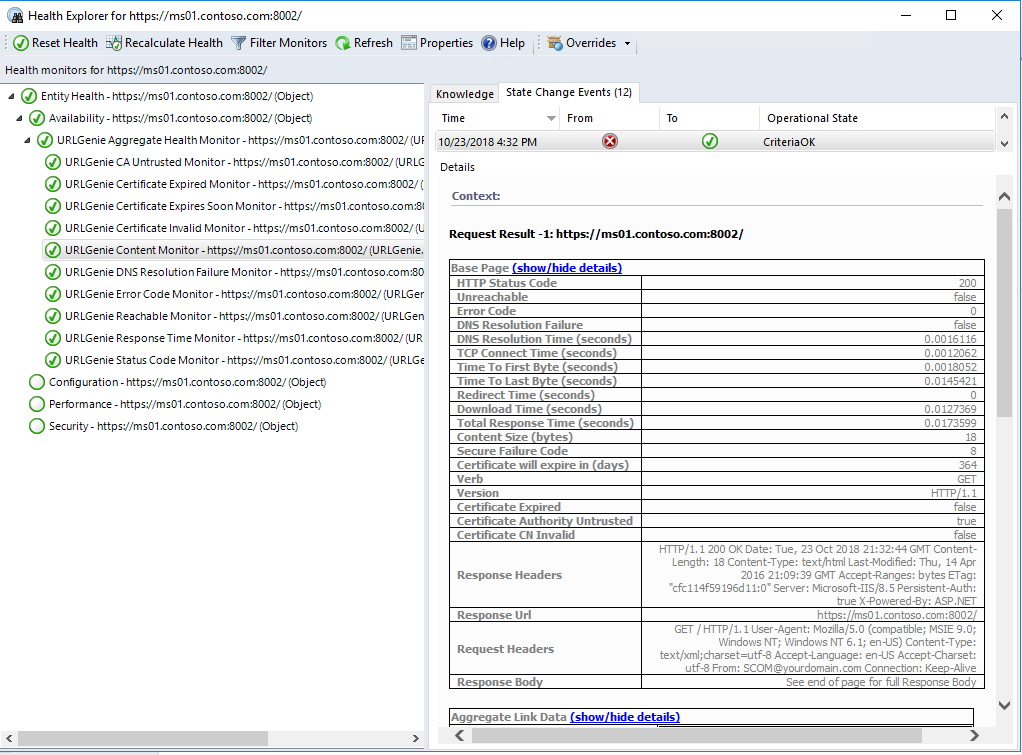


At first my *authuser* value (property) has not been discovered yet…



However, after a few minutes it has become discovered and the instance becomes healthy once it has the correct username to use for authentication.

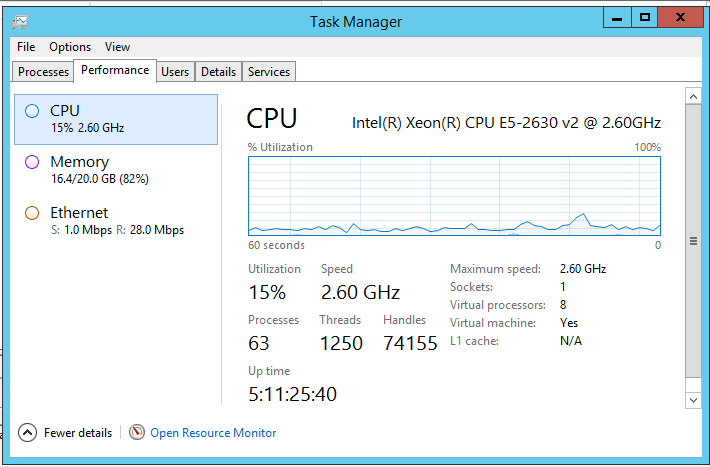


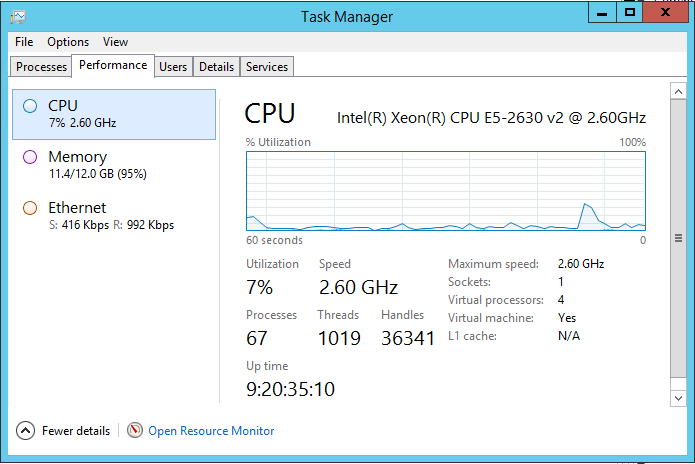


Note: for this test I created a self-signed certificate with IIS Manager on the www server for the HTTPS binding of the test site. I then exported the .pfx cert and imported it into the Local Computer->Trusted Root Certification Authorities->Certificates hive of the watcher node. This step was not required to monitor the target URL successfully however it eliminates the bothersome warning that you get during setup and testing with a web browser.

# Load Testing

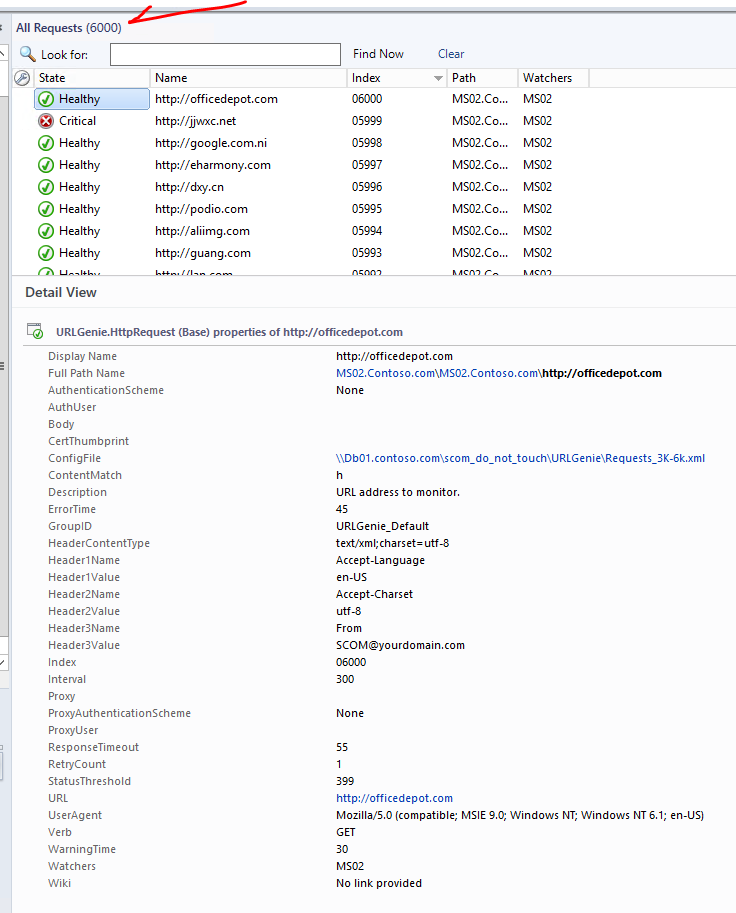
Over a period of about 4 weeks I tested monitoring of various quantities of URLs. What I found is that the number of URLs discovered at any one time is irrelevant. I did not encounter a discovery object size limitation during my testing although it is theoretically possible to exceed the discovery object size limit. With the server specifications shown below I was able to monitor 6000 http URLs reliably from one management server enabled as a watcher node, with the standard monitors (no scripted or https monitor types). My lab consisted of one SQL server hosting the OpsDB and DW, and two management servers. One management server, enabled as the only Watcher Node, had significantly more resources than the other.

  
Management Server / Watcher Node: Windows Server 2012 R2

,  
SQL Server: Windows Server 2012 R2, SQL Server 2014

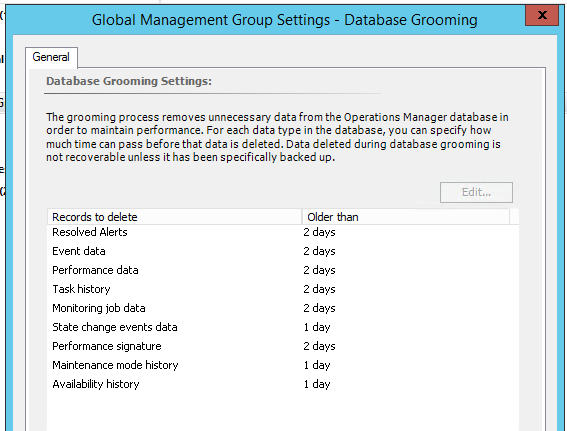
**6000 URLs (http)**

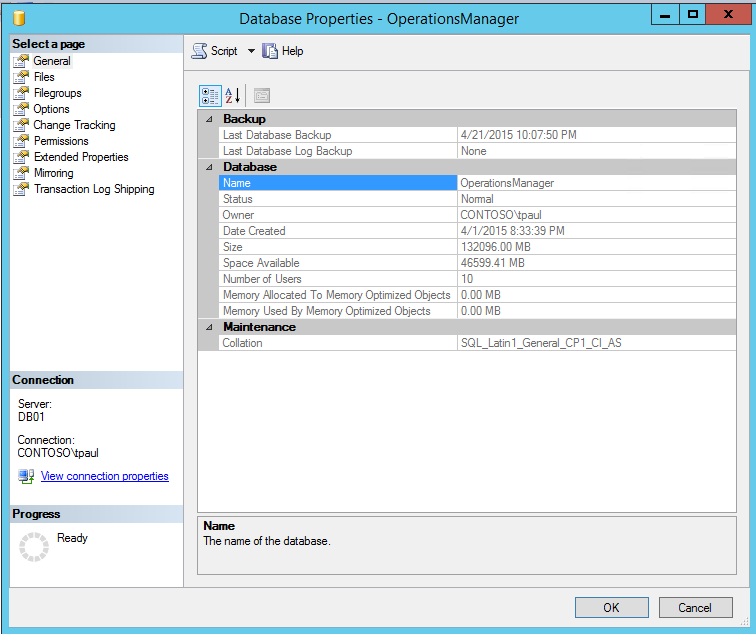
Approximately 6000 URLs seemed to be the max for my lab management group. More than 6000 resulted in low memory problems. See [Memory Errors](#_Memory_Errors) below.

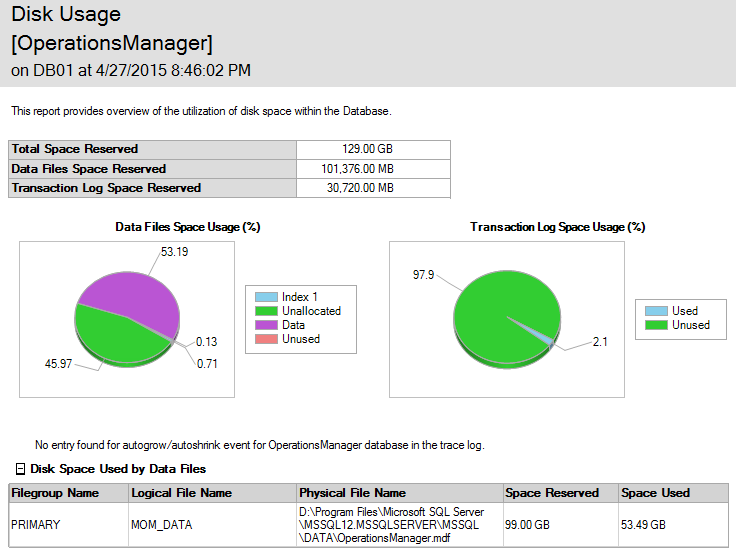


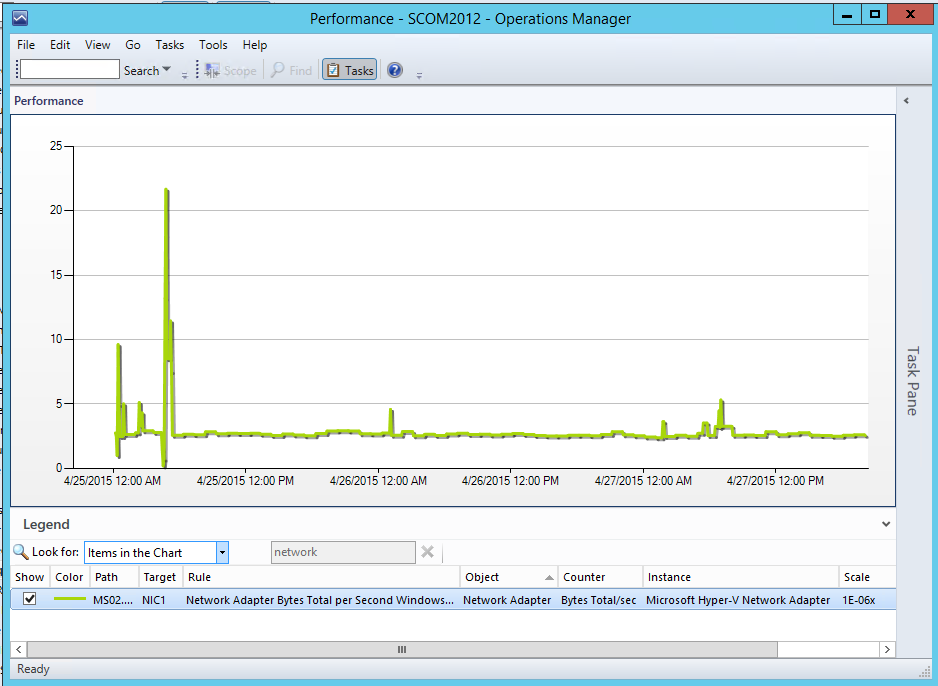
**Databases**

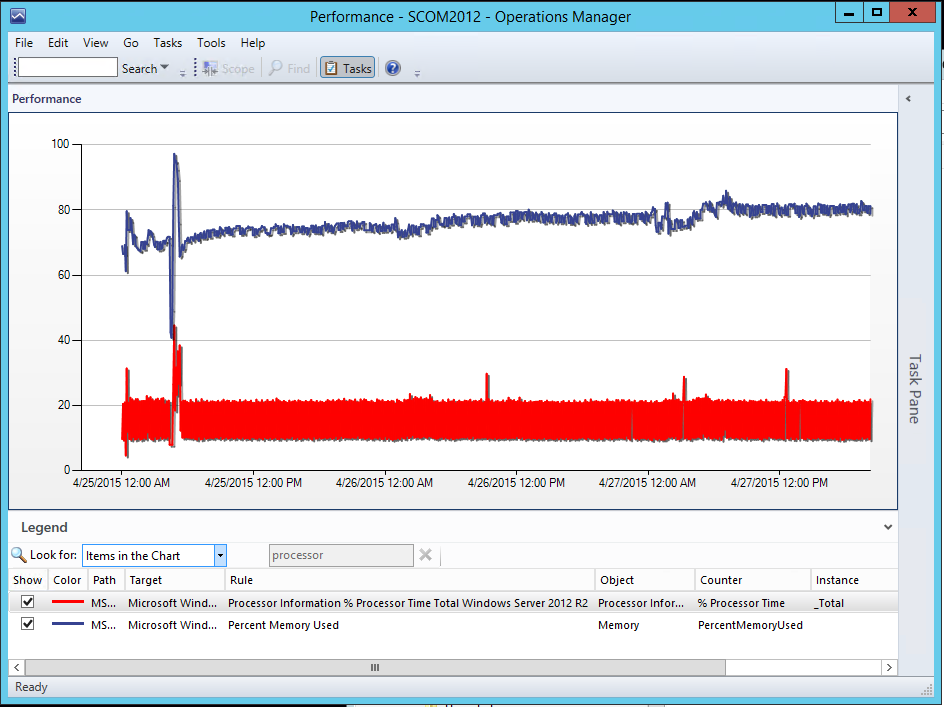
OpsDB Retention:





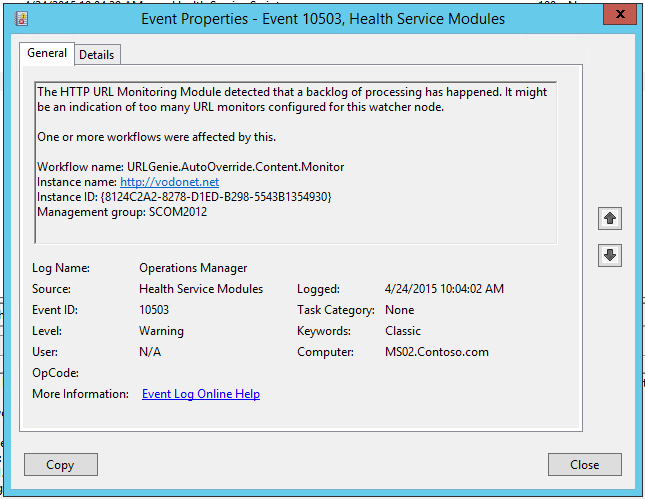


  
2 Days (Network Adapter MB/Second)

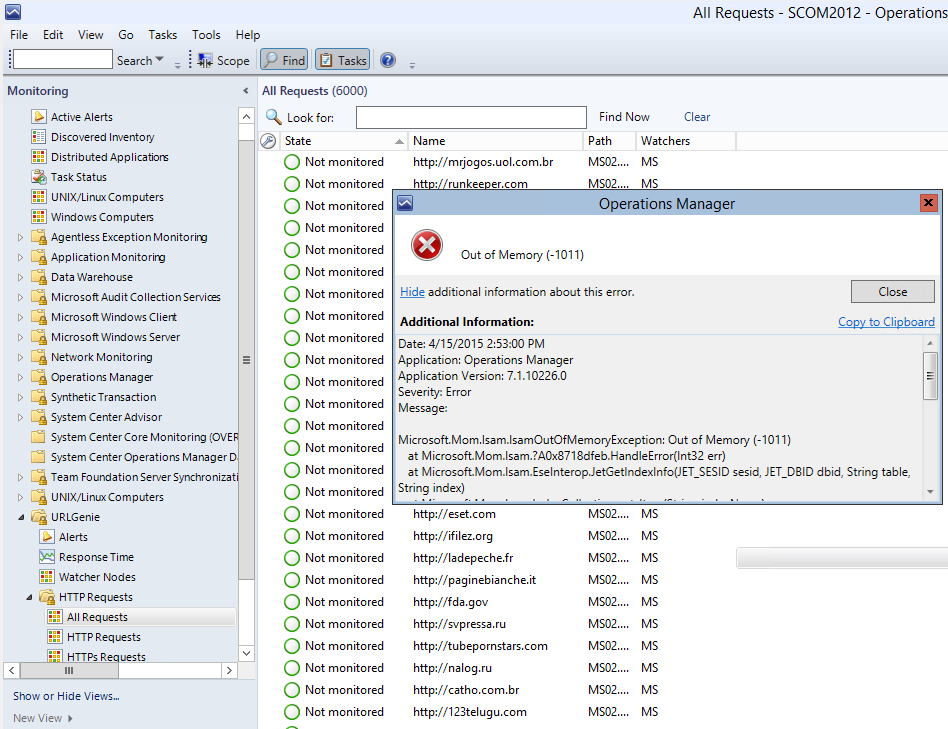
  
2 Days: CPU, Memory Percent Usage

## Memory Errors

I reached 7000 URLs before I started seeing errors like the one below so I dropped my test to 6000 instances. 6000 seemed to be about the max number of URLs that my test server could handle.



(Errors appeared when testing 7000 instances.)



*Date: 4/15/2015 2:53:00 PM*

*Application: Operations Manager*

*Application Version: 7.1.10226.0*

*Severity: Error*

*Message:*

*Microsoft.Mom.Isam.IsamOutOfMemoryException: Out of Memory (-1011)*

*at Microsoft.Mom.Isam.?A0x8718dfeb.HandleError(Int32 err)*

*at Microsoft.Mom.Isam.EseInterop.JetGetIndexInfo(JET\_SESID sesid, JET\_DBID dbid, String table, String index)*

*at Microsoft.Mom.Isam.IndexCollection.get\_Item(String indexName)*

*at Microsoft.Mom.Isam.Cursor.get\_CurrentIndexDefinition()*

*at Microsoft.Mom.Isam.Cursor.MakeKey(Key key, Boolean end)*

*at Microsoft.Mom.Isam.Cursor.GotoKey(Key key)*

*at Microsoft.EnterpriseManagement.Mom.Internal.UI.Cache.EseCursor.Find(Object[] values)*

*at Microsoft.EnterpriseManagement.Mom.Internal.UI.Cache.CacheDirectory.FindIndexTable(Guid managementGroup, String primaryId, String queryType, Guid targetType, Guid target, String criteria, Collection`1 fields)*

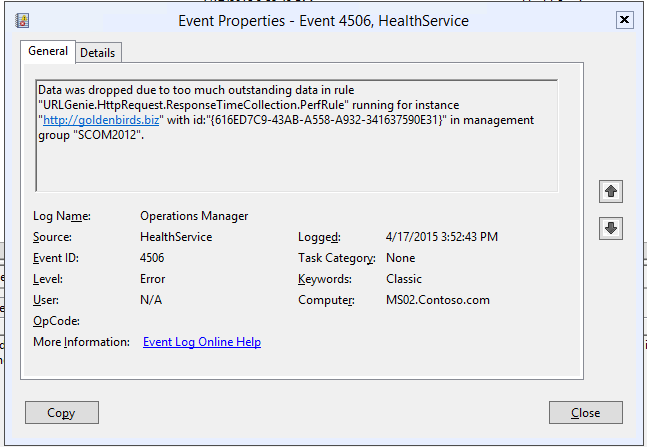
*at Microsoft.EnterpriseManagement.Mom.Internal.UI.Cache.Query`1.GetIndexTable(CacheSession session, Boolean create)*

*at Microsoft.EnterpriseManagement.Mom.Internal.UI.Cache.Query`1.InternalQuery(CacheSession session, UpdateReason reason)*

*at Microsoft.EnterpriseManagement.Mom.Internal.UI.Cache.Query`1.TryDoQuery(UpdateReason reason, CacheSession session)*

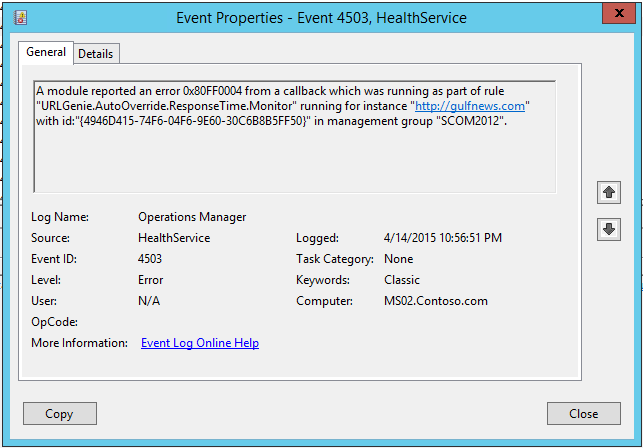
*at Microsoft.EnterpriseManagement.Mom.Internal.UI.Console.ConsoleJobExceptionHandler.ExecuteJob(IComponent component, EventHandler`1 job, Object sender, ConsoleJobEventArgs args)*

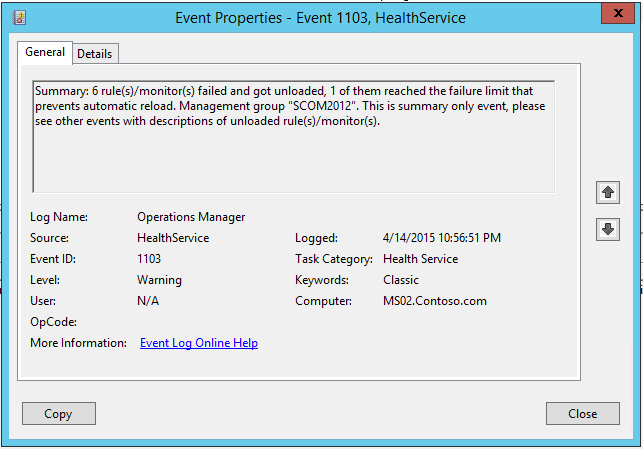
Out of Memory



# Troubleshooting

In my testing of over 6000 URLs I encountered a few that caused monitoring to fail. I simply commented them out. I don’t yet have an answer as to why they caused failures. It’s entirely possible that the body of the page was simply too large. I’ve seen this cause problems before. I have since disabled the “collect response body” feature. It can be enabled with an override.







I simply commented out the problematic URL request elements.