

Monitoring

MailStore only provides limited notification or monitoring features, but the status of the archiving processes can be monitored using external components.

Using External Monitoring Software

MailStore Nagios/Icinga-Plugin

The [scripting package](#) includes the `check_mailstore.py` plugin. The plugin checks the number of jobs or the number of archived emails in a given period of time. At least MailStore Server 8 is required.

Installation

The directory `mailstoreapi` from the package should be copied below the site-packages directory of your Python installation. The location of the site-packages directory can be found with the following command

```
python -c "from distutils.sysconfig import get_python_lib; print(get_python_lib())"
```

Depending on your distribution, you might have to install the `python-argparse` package.

The plugin connects to the [MailStore Server Administration API](#). Therefore it must be enabled in the [MailStore Server Service Configuration](#).

Usage

A check that monitors the successful execution of profiles could be defined in Nagios/Icinga as follows:

```
define command {  
  
    command_name check_mailstore  
  
    command_line /usr/local/lib/nagios/plugins/check_mailstore.py --host $ARG1$ --password $ARG2$ -s  
since:$ARG3$ --status $ARG4$ -c $ARG5$ -w $ARG6$ --search $ARG7$  
  
}
```

The appropriate service definition might look like this:

```
define service {  
  
    host_name mailstoreserver  
  
    service_description MailStore Succeeded Jobs  
  
    check_command check_mailstore!mailstoreserver!sUp3rs3CcR6ET3!1H!succeeded!8!10!jobs  
  
    use generic-service  
  
}
```

This test checks whether there were more than 10 tasks executed successfully (`--status succeeded`) during the last hour (`-s since:1H`).

Parameters

The plugin supports the following parameters.

```
--help|--h
```

Displays the help page.

```
--host HOST
```

Hostname or IP address of the MailStore Server. The default is *localhost*.

```
--port PORT
```

TCP port on which the MailStore Administration API accepts connections. Default is *8463*.

```
--username USERNAME
```

Username to log on to MailStore Server. This must be a MailStore administrator. By default, *admin* is used.

```
--password PASSWORD
```

The user's password.

```
--start STARTTIME|-s STARTTIME
```

Specifies the start time of the check period. The start time has to be given in the format YYYY-mm-ddTHH:MM:SS (eg 2013-01-01T00:00:00). The *-end* parameter has to be given. As alternative a time period can be given with the format *since:XY*, where X is a number and Y is one of the following letters: Y (year), m (month), d (day), H (hour), M (minute) S (second). Example *-s since: 90M* (last 90 minutes).

```
--end ENDTIME|-e ENDTIME
```

Specifies the end time of the period. The format is YYYY-mm-ddTHH:MM:SS (eg 2013-02-28T23:59:59). When using *since* in *--start*, this parameter is not required.

```
--timezone TIMEZONE
```

MailStore Server stores dates in UTC time. The output of the plugin can be adjusted with this parameter. By default, *localhost* is used. This corresponds to the time zone setting of the operating system of MailStore Server. Using the API command [GetTimeZones](#) the possible values can be shown. In most cases, this parameter is not required.

```
--machinename MACHINENAME|-m MACHINENAME
```

Filters the results by *MACHINENAME*. This is useful when the results of local jobs of different computers are monitored.

```
--profile PROFILE|-p PROFILE
```

Filters the results by archiving profile. The ID or the name of an archiving profile can be given.

```
--status STATUS
```

Filters the results by STATUS. Possible values are *succeeded*, *failed*, *cancelled*, *disconnected*, *threadAbort* and *completedWithErrors*. The status can be negated by prepending a *#*. Default is *succeeded*.

```
--search [jobs|emails]
```

Specifies whether to check on the number of returned jobs or the number of mails archived. Default is *jobs*.

```
--warning WARNING|-w WARNING
```

The warning threshold.

```
--critical CRITICAL|-c CRITICAL
```

The critical threshold.

```
--compare COMPARE
```

Specifies how the values of WARNING and CRITICAL will be compared with the amount of results. Possible values are *lt*, *le*, *eq*, *ge*, *gt* (lesser than, lesser than or equal, equal, greater than or equal, greater than). Default is *le* (lesser than or equal).

```
--DEBUG
```

If given, the matching results will be printed to standard output. This is only useful for debugging purpose.

Other examples

```
check_mailstore.py --host 192.168.0.1 --password sUp3rs3CcR6ET3 -s "since:1d" -c 5 -w 2 --search jobs --status="#succeeded" --compare gt
```

Status is critical if more than (--compare gt) 5 (-c 5) jobs (--search jobs) have NOT ended successfully (--status "#succeeded") within the last day (-s "since:1d"). A warning is issued when more than 2 unsuccessful jobs have been found.

```
check_mailstore.py --host 192.168.0.1 --password sUp3rs3CcR6ET3 -s "since:1d" -c 5 -w 20 --search emails --profile "MailStore Proxy"
```

Status is critical if less than 5 (-c 5) emails (--search emails) were archived within a day by the the profile "MailStore proxy" (--profile "MailStore proxy"). A warning is issued when less than 20 emails were archived.

Monitoring of licenced users

The *check_mailstorelicence*-script from the [scripting-pakage](#), can be used to monitor the existing users in MailStore with Nagios/Icinga. No external arguments can be used, all configuration has to be done inside the file. If you synchronize your users from an external source, and more users than free licences should be created in one step, this monitoring will not holler, because it checks the existing users only and not the users that shall be created.

Command-Definition:

```
define command {  
  
    command_name check_mailstorelicence  
  
    command_line /usr/local/lib/nagios/plugins/check_mailstorelicence.py  
  
}
```

Nagios/Icinga with NSClient++

If you are already using monitoring software, such as Nagios/Icinga, Zabbix or HP OpenView, in your network, we recommend monitoring the results of the Windows task scheduler.

This example requires that in section *[NRPE]* of the file *NSC.ini* the parameter *allow_arguments=1* is set. An alternative, and safer in public environments, is to define an *alias* under section *[External Alias]*.

Under Nagios/Icinga the corresponding service check looks like this:

```
define service {  
  
    use generic-service  
  
    host_name mailstore.mydomain.tld  
  
    service_description Scheduled Tasks  
  
    check_command check_nrpe!CheckTaskSched!filter="exit_code ne 0"  
    "syntax=%title%: %exit_code%" "crit=>0"  
  
}
```

The service check puts out a list of all scheduled tasks in the Windows task scheduler whose exit code is unequal to zero. If there is more than one event, the check status *Critical* is set. The return contains a list of all tasks with exit

codes unequal to zero and their exit codes.

Notifications for Failed Archiving Processes

At this time, MailStore Server's email notification feature only sends an email if the automatic creation of a new standard archive store fails.

This article provides some helpful hints to administrators who would like to receive additional notifications regarding events on their MailStore server.

Notifications for Audit Events

One way for monitoring is the use of the MailStore auditing feature combined with the Windows task planner.

Please keep in mind that this procedure negates the actual purpose of MailStore's auditing feature. Therefore, verify if the trigger parameters are still configured correctly after each update of the MailStore Server.

It is not possible to configure activation triggers in Windows, Windows Vista/7/2008/2008 R2 is needed. They are not available in Windows 2000/XP/2003.

Activating Auditing Features

- Open MailStore Client as administrator.
- Click on *Administrative Tools > Compliance > Auditing*.
- Activate the user activity *ProfileRunArc*.

Now, after archiving profiles have been executed, corresponding entries are made in the event log.

Checking the Windows Event Logs Manually

- Open the *Event Viewer* of your Windows system.
- Click on *Event Viewer (local) > Windows Protocols > Applications*.
- Search for events of source *MailStore Server Auditing*.

If errors occurred while executing the profile, the event level is *Error*; if execution was successful, the level is *Information*.

Creating Notifications

The Windows task scheduler can link tasks to an event. This is used to send an email at the event *Archiving Failed*.

- Open the *Task Scheduler* of your Windows system.
- Create a new folder, e.g. *MailStore Auditing* in the *Task Scheduler Library*.
- Create a task via *Actions > Create Task*. Please note that you will not *Create a Simple Task*.
- Enter a meaningful name.
- Select the option *Run whether user is logged on or not*
- Under *Configure for*, select at least *Windows Vista or Windows Server 2008*. Otherwise the trigger *On Event* is not available.
- Click on the *Triggers* tab.
- Click on *New*.
- Under *Start Task* select the value *On Event*.
- Under *Settings* activate the option *User Defined* and click on *New Event Filter*.
- Under *Event Level* place a checkmark next to *Error*.
- Select *Via Source* and under *Sources* place a checkmark next to *MailStore Server Auditing*.
- Click on *OK* to save the settings.

The criteria for user-defined settings are stored as XML data. Unfortunately, the *Edit Trigger* dialog is unable to convert these XML data back into GUI elements. Subsequent manipulation of the trigger is only possible in XML. If this is not desired, the trigger must be deleted and recreated.

- Change to the *Actions* tab.
- Click on *New....*

E-Mail message via Powershell script

- Create a file with the extension .ps1 with the following content. Adjust the values according to your environment.

```
$EmailFrom = "mailstore@domain.eu"

$EmailTo = "administrator@domain.eu"

$Subject = "MailStore Error"

$Body = "Please check MailStore Server logs"

$SMTPServer = "smtp.domain.eu"

$SMTPPort = 25

$SMTPClient = New-Object Net.Mail.SmtpClient($SMTPServer, $SMTPPort)

$SMTPClient.EnableSsl = $false

$SMTPClient.Credentials = New-Object System.Net.NetworkCredential("username", "password");

$SMTPClient.Send($EmailFrom, $EmailTo, $Subject, $Body)
```

- In the *Action* field select *Start a program*.
- As *Program/script* enter *powershell.exe*. As argument enter the path to the Powershell script
- The execution of scripts has to be enabled via *Set-ExecutionPolicy RemoteSigned*
- If the scripts fails, you may add *-noexit* to the arguments, to keep the powershell window open

Network message

- is no SMTP server available the *msg* program can be used instead, which sends messages over the network.
- In the *Action* field select *Start a program*.
- As *Program/script* enter *msg*. As argument enter */server:ip-address username message*, example:
/server:192.168.2.100 administrator "One archiving job has failed or completed with errors"
- all options of *msg* are documented here <http://technet.microsoft.com/en-us/library/cc755358.aspx>
- the user administrator of machine 192.168.2.100 gets a windows with the notification
- the registry-key *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\AllowRemoteRPC* (REG_DWORD) has to be set to *1* on the receiving machine

🕒Revision #1

★Created 29 December 2021 16:46:39 by Mahesha Damayanthi

✎Updated 29 December 2021 16:55:54 by Mahesha Damayanthi