Table of Contents Version 2.92

[1 - Probe\_maint\_on 3](#_Toc23748852)

[1.1 probe\_maint\_on upgrade 4](#_Toc23748853)

[1.2 probe\_maint\_on new installation 6](#_Toc23748854)

[1.3 nimsoft\_generic.dat 11](#_Toc23748855)

[1.4 probe\_maint\_on.cfg 12](#_Toc23748856)

[1.5 start probe\_maint\_on 14](#_Toc23748857)

[1.6 probe\_maint\_on logic flow 15](#_Toc23748858)

[2 - Setup Perl environment 19](#_Toc23748859)

[3 – Maintenance Report 21](#_Toc23748860)

The standard UIM USM/UMP server maintenance environment is sometimes difficult to integrate in daily production environments that need some command-based possibilities.

This customized tool: **probe\_maint\_on** will accept specific "nimalarm" messages to define a new calendar and add a device/server into this calendar.

Example to define a server in maintenance for 3600 seconds:

nimalarm -l 1 "maint\_on fake-server 3600"

Example to remove a server from maintenance: (seconds is set to: 0)

nimalarm -l 1 “maint\_on fake-server 0”

**Updated:**

- 2.2: added option: uim\_https in nimsoft\_generic.dat. The value can be http or https to access the UMP Rest service.

- 2.3: added option: calendar\_seconds in probe\_maint\_on.cfg. This value is by default 60 (was hardcoded at 60 in previous versions). When creating a new calendar, we will take the current time and add this # of seconds to calculate the calendar start time. We need some extra time because when defining the calendar via the REST service the start time cannot be less that the current time at that moment, else the request will fail.

- 2.4: added option: minimum\_seconds in probe\_maint\_on.cfg. All maintenance request with a time in seconds less than this time (default: 60) will be rejected with an error message in the console.

- 2.5: - added option: remove\_msg in probe\_maint\_cfg. If set to y we will auto acknowledge all maint\_on requests in ERROR after generating the reason of the error. If set to n we will still generate the error message (reason why we did not execute the maintenance request), but we will not CLEAR the original maint\_on request.

- in the probe definition the Times was changed, in error, from 600 to 10 minutes. Switched back to 600 minutes. Goal is that the probe restart itself after 10 hours.

- 2.6: - added the option to remove servers from maintenance by specifying the seconds as 0

- 2.7 - if a server has multiple cs\_id, we will define all found cs-ids in maintenance

- extra checks are added on input fields (time numeric, remove possible ‘ round server name)

- the remove “out of maintenance” can be done via a regex match (= multiple servers with a | between server names )

- probe is now installed in /nimsoft/probes/custom/probe\_maint\_on

- 2.8 - we will now check if a server is defined in cm\_computer\_system before creating the calendar

 - remove server must be done with the exact matching name (case insensitive) (the code to do it via regex is there, but not used to avoid errors)

- 2.9 - generate alarm when sql login failure

 - add sql login check at the start of the probe

- 2.9.1 – review doc and update create Perl environment creation

 - extra test on “no seconds” given in nimalarm

 - removed the document section with a customized LUA script to modify the incoming alarm to avoid confusion/complexity

- 2.9.2 – nimsoft\_generic.dat and nimsoft\_generic.pm are modified to add a new parameter: $sql\_driver. This gives you the possibility to add and use a new MSSQL driver on your system to use TLS 1.2

# 1 - Probe\_maint\_on

The probe **probe\_maint\_on** is a customized solution that makes the setting or removing of servers "in maintenance" possible via a simple "nimalarm" command.

The nimalarm is matched by a NAS profile that will repost the message to a custom queue "maint". This new, user created, ATTACH queue "maint" will hold all pending maintenance requests until the probe will/did process them.

For each “in maintenance” request the probe will make a new calendar: "OnDemand\_servername" with as start date/time the current time + 1 minute and as end date/time the current time + the number of seconds given as parameter.

Once the calendar is created we will add the server/device into this calendar.

This also means that after some time you will have a lot of old, expired calendars. A SQL procedure/script/command is provided to delete expired and empty calendars from the database. (or you can use the published tool: nimsoft\_rest\_maintenance.pl with the options -de and -dx)

**Note:** in UIM 9.0.2 the internal cleanup of these tables was added, but is not removing all entries. But due to this partial removal the –dx and –de options (in nimsoft\_rest\_maintenance.pl) are not working 100% anymore.

**Note:** nimsoft\_rest\_maintenance.pl has now a –dy option to delete expired schedules

## 1.1 probe\_maint\_on upgrade

If you are currently running the probe probe\_maint\_on:

* save/backup the files:
	+ probe\_maint\_on.cfg
	+ nimsoft\_generic.dat

from directory: \Nimsoft\probes\system\probe\_maint\_on

or

\Nimsoft\probes\custom\probe\_maint\_on

* import the new probe in your archive
* deactivate the probe: probe\_maint\_on
* drag/drop the new archive version to your UIM robot where the probe was active, by default it will not be activated
* if you plan to use the **not compiled** version:
	+ edit the archive package and select the "probe definitions" tab.
	+ Select probe properties and modify:
	+ the "command" parameter and set this to the correct path to your Perl. (default: C:\strawberry\perl\bin\perl.exe)
	+ the "argument" parameter to: probe\_maint\_on.pl

Example for usage of the **not compiled** Perl source:



**Note**: from version 2.9 the probe will be installed in: /probes/custom/probe\_maint\_on

Example of the default probe with **compiled** Perl source:



* edit in the probe directory: nimsoft\_generic.dat and adapt all parameters based on the backup version (or you can copy the backup version and **add manually** the new "uim\_https=http" line, if not found in your old version)
* edit in the probe directory: probe\_maint\_on.cfg and adapt the parameters with your customized parameters based on the backup version
* you can Activate the new version of the probe. Select "view log" to detect possible error messages during this first start of the new version. We tried to be as explicit as possible in the log file.

**Note**: from version 2.6 we will only deliver a nimsoft\_generic.dax and probe\_maint\_on.cfx; this with as goal not to override your current customized files.

## 1.2 probe\_maint\_on new installation

Probe\_maint\_on runs on the primary hub.

Directory: /nimsoft/probes/custom/probe\_maint\_on.

-pre-requisites:

* (optionally) having Strawberry Perl 5.14.2 installed/copied with the needed packages and put the directory in the path **or** use the .exe version of the probe (= a compiled version of the probe) See appendix on how you can create from scratch a valid Perl environment.
* (optionally) for the maintenance report the MSSQL utility: BCP.EXE on the master hub is a requirement to be able to export all servers in maintenance into a file (MSSQL installation - management tool - basic). (available on request)
* Package: webservices\_rest must be deployed on your UMP server (wasp will restart)

- This probe\_maint\_on\_2.92.zip file must be dragged/copied to the hub archive from where it can be copied to the main hub.

- Create a NAS profile to trap the "maint\_on" messages. Your message string must/can be:

/^maint\_on/

Or, **optional**, you can enter several regex keywords, like:

/^maint\_on|^#SURAT\_I\_0001.\*.-.Demande.de.reboot|^#SURAT\_I\_0000.\*.\*.DEBUT.DE.MAINTENANCE/

In this optional case you must check if the probe has some translation logic for the 2 extra regex values. (see later)

**Note**: by default, the translation logic of the messages is done by the probe Perl source, so that the probe recognizes the server name and the timing in seconds.



**Note1**: in some screen shots we use multiple messages formats that are accepted, these message formats are translated in the probe Perl source as an example.

**Note2**: it is important that you **don't** select the "clear" option in the NAS rule, else you would create a looping rule, because the acknowledged alarm would enter also in the probe.

- Copy your customized version of: nimsoft\_generic.dat (generic UIM parameters and encrypted passwords) to: \Nimsoft\probes\custom\probe\_maint\_on (see next section to create this file if not exist)

- Compiled Perl version (by default in the archive package):

- command: probe\_maint\_on.exe

- argument: (empty)



**or**

- Not Compiled Perl version:

- in the appendix you can find the instructions, if needed, to setup a full Perl environment based on Strawberry Perl with the UIM API’s

- Command: C:\strawberry\perl\bin\perl.exe

- Argument: probe\_maint\_on.pl



**Note:** from version 2.6 we will deliver the probe as a daemon, meaning that it will run permanent. If you want the probe to restart automatically every xx hours, you can change the probe settings into “Timed” and in Execution interval use: 10 hours.

* In the IM GUI select your primary **hub probe** and select the Queues tab and select: New



This new queue will hold all maintenance requests in case the WASP/UMP/REST service is not (yet) active. This queue will also buffer incoming requests until it is processed by the probe.

**Note1**: by default, we start the compiled version of the probe.

**Note2**: if you have a full Strawberry Perl version, with all 3 directories in the path (see last chapter), you can compile, if needed, this Perl by the command (in case you customized the Perl source):

pp -u -C -o c:\unibat\probe\_maint\_on.exe -I lib -l C:/strawberry/c/bin/libxml2-2\_\_.dll -l C:/strawberry/c/bin/libiconv-2\_\_.dll -l C:/strawberry/c/bin/libz\_\_.dll -l C:\strawberry\c\bin\libmysql\_\_.dll -l c:\Strawberry\c\bin\libeay32\_\_.dll probe\_maint\_on.pl

## 1.3 nimsoft\_generic.dat

Directory: /nimsoft/probes/custom/probe\_maint\_on.

nimsoft\_generic.dat is a parameter file that contains all user, password and server definitions so that several custom tools can access UIM internals & SQL DB without having the need of coding extra access parameters on each run.

# -- UMP/Rest server & port

uim\_server=ump\_server\_name

uim\_port=80

uim\_https=http

# --- Nimsoft userid and crypted password (via nimsoft\_crypt.exe)

uim\_user=administrator

**uim\_password**=gWL/M/ij/..

# --- Nimsoft domain, hub and robot to create address to the main hub

uim\_domain=xxx\_domain

uim\_hub=xxx\_hub

uim\_robot=xxx

# --- SQL server, userid, crypted password (via nimsoft\_crypt.exe) and database name

sql\_server=sql\_server\_name

sql\_user=sa

**sql\_password**=gWL/M/ij/..

sql\_db=CA\_UIM

# --- sql type: mysql or mssql

sql\_type=mssql

# - sql\_driver: "SQL Server" (=default) or a manual installed newer driver, example: "ODBC Driver 17 for SQL Server" (\* no quotes around driver name \*)

sql\_driver=SQL Server

# --- end of parameters ---

The 2 passwords are stored in an encrypted form. To generate this encrypted password, use:

nimsoft\_crypt.exe your\_password

As output you will receive the string that you can enter as password in the above file.

## 1.4 probe\_maint\_on.cfg

Directory: /nimsoft/probes/custom/probe\_maint\_on.

The probe must find in his subdirectory a config file: probe\_maint\_on.cfg This file contains some user variables:

<setup>

 loglevel = 1

 debug = n

 logfile = probe\_maint\_on.log

 queue = maint

 prefix = custom

 logsize = 200000

 suppkey = probe\_maint\_on

 calendar\_seconds = 60

 minimum\_seconds = 60

 remove\_msg = y

</setup>

* loglevel: 1. Only possible value for the moment
* debug: n. Can be set to y if the Perl is executed in interactive mode to have more logic info about the execution.
* logfile: name of the logfile in the probe\_maint\_on directory.
* queue: when the probe starts it will try to connect to an existing ATTACH queue with this name
* prefix: what is the format of the queue message:
	+ **custom**: the incoming message can be almost anything; the translation will be handled in the Perl at the comment "custom" (see section 1.4). By default the probe is customized to recognize the prefix: maint\_on
	+ maint\_on: the incoming first word must be this word and can be reformatted by a pre-process LUA script (see section 1.5)
* logsize: size of the logfile
* suppkey: suppression key used by the probe alarms
* calendar\_seconds: number of seconds that will be added to the current time to calculate the calendar start time. Default: 60 (if this value is too small there is a possibility that the calendar start time is less that the current time at the moment the REST define request is executed and the define will fail)
* minimum\_seconds: minimum time that must be entered in the maintenance request, expressed in seconds. Default: 60. **Attention**: if you accept requests that have a time that is too low, the REST service will generate an "internal error"
* remove\_msg: when set to y we will CLEAR invalid maint\_on requests after generating an error message with the reason of the error. When set to n we will still generate an error message, but we will also keep the original maintenance request message. Default: y.

**Note**: you can use the config file like displayed here, this means that the probe will accept a maintenance request via the nimalarm:

nimalarm.exe -l 1 "maint\_on server seconds”

## 1.5 start probe\_maint\_on

After the initial deployment of the probe it's not started by default because you need all the previous customizations steps. If all these steps are done you can start the probe and select the "view log" option"

Sep 20 06:17:19:959 probe\_maint\_on: 0 - 5 - alarm\_level=INFO message=probe\_maint\_on - Wasp Resp services are not ready (yet) msg: read timeout source= suppression=resp

Sep 20 06:21:19:964 probe\_maint\_on: 0 - 5 - alarm\_level=INFO message=probe\_maint\_on - Wasp Resp services are not ready (yet) msg: read timeout source= suppression=resp

Sep 20 06:23:32:737 probe\_maint\_on: 0 - 5 - alarm\_level=CLEAR message=probe\_maint\_on - Was Resp services is responding msg: read timeout source= suppression=resp

Sep 20 06:23:32:737 probe\_maint\_on: RestCheck: REST services are responding (activate in 1 minute)

Sep 20 06:24:32:739 probe\_maint\_on: 0 - 5 - alarm\_level=INFO message=probe\_maint\_on - ready attaching to queue: maint and we can now process maintenance requests source= suppression=

Sep 20 06:33:20:675 probe\_maint\_on: 1 - 0 - Custom decode: server: fake-server-21 seconds: 6000 (maint\_on fake-server-21 6000)

Sep 20 06:33:21:390 probe\_maint\_on: 1 - 1 - Create calendar: OnDemand\_fake-server-21 msg: maint\_on fake-server-21 6000 id: 1304

Sep 20 06:33:21:412 probe\_maint\_on: 1 - 2 - We found 1 devices in cm\_computer\_table

Sep 20 06:33:21:414 probe\_maint\_on: 1 - 3 - We added 1 servers to add in maintenance

Sep 20 06:33:21:804 probe\_maint\_on: 1 - 4 - Received add reply: ok

Sep 20 06:33:21:804 probe\_maint\_on: 1 - 5 - alarm\_level=CLEAR message=maint\_on fake-server-21 6000 source= suppression=

This log will also contain detailed info about every maintenance request.

You can see in this log:

- between 06:17 and 06:23 the Wasp probe is not responding

- at 06:23 the probe\_maint\_on is ready to process alarms

- at 06:33 the probe receives 1 request, and all 6 steps to define a server in maintenance are displayed for debugging purposes

## 1.6 probe\_maint\_on logic flow

This new probe will monitor the attach queue "maint" and will process all (pending) requests.

- when the probe starts, it will verify:

- if sql login works fine, else it will stop with rc: 12 and send a major alarm. The controller probe will restart the probe.

- if the Wasp REST service is responding, else he will wait 60 seconds and try again (this can happen specially after a reboot of the UIM servers)

- once the probe is ready to put servers in maintenance he will send an alarm with severity/level 1:

probe\_maint\_on - ready attaching to queue: maint and we can now process maintenance requests

- if the probe\_maint\_on.cfg parameter prefix is set to **custom**, the default, the probe accepts any message that is entering the queue "maint". It's up to you to customize the probe source (probe\_maint\_on.pl) on the comment "custom" to add additional message formats (other than the standard: maint\_on).

Out of the box the probe will accept the message format:

- **maint\_on** fake-server-15 3600

**Note**: the following section is ONLY needed if you plan to customize the message format other than “maint\_on”

In the custom code you must verify if the message format is correct and populate the variables $m\_server (= server name) and $m\_seconds (= number of seconds that the server will be in maintenance).

**Note**: sample other message formats (that are included for a specific client):

- #SURAT\_I\_0001: we will take the second word and map it to the server name and we will take a default maintenance period of 1800 seconds (the time needed to perform a reboot)

- #SURAT\_I\_0000: we will take the 3th word as server name and use word 17 (=400) to map to the number of seconds to be in maintenance

Custom code in Perl: probe\_maint\_on.pl:

 if ($prefix eq 'custom')

 {

 if ($m\_word eq 'maint\_on')

 {

 $m\_server = $words[1];

 $m\_seconds = $words[2];

 }

 if ($m\_word eq '#SURAT\_I\_0001' && $words[3] eq 'Demande')

 {

 $m\_server = $words[1];

 $m\_seconds = "1800";

 }

 if ($m\_word eq '#SURAT\_I\_0000' && $words[3] eq 'DEBUT' && $words[5] eq 'MAINTENANCE')

 {

 $m\_server = $words[2];

 $m\_seconds = $words[16];

 }

}

**Note**: $words[1] means that we test on the second word of the original alarm

* a first check done is on the number of seconds specified. If the number is 0, we know that this is a remove from maintenance request, else it’s an "add in maintenance" request

- when a new "maint\_on" enters the queue the probe will check if the server exist in the table: cm\_computer\_system. In case the server name does not exist in UIM you will receive:

probe\_maint\_on - Device name: fake-server does not exist maint\_on fake-server 3600)

And the request will be removed from the queue

- next we will define the maintenance calendar and save the calender-id.

- in case there are problems with the calendar definition we will sleep for 60 seconds and try again + we will send an alarm to the console:

probe\_maint\_on - Error adding calendar: fake-server-119 msg: Wasp Rest Services are not responding (anymore)

- as soon as the calendar problem is solved the alarm is cleared

- When we have a return code 0 from the calendar definition we will add the device/server into the calendar.

- in case there are problems with the add device we will sleep for 60 seconds and try again + we will send an alarm to the console:

probe\_maint\_on - Error adding device: fake-server-112 in calendar msg: Wasp Rest Services are not responding anymore

 - in case the time is not totally in digits you will receive:

probe\_maint\_on - time is not in digits (maint\_on 1234 fake-server-14)

- as soon as the add device problem is solved the alarm is cleared

- as soon as the device is added into maintenance we will clear the original "maint\_on" message

- only at that moment we remove the request from the attach queue "maint"

**Note1**: this probe was tested with 600 "maint\_on" requests arriving almost at the same time.

**Note2**: pre 2.6: to avoid problems the probe will restart itself after 10 hours, this way we are sure that all connections are retested and refreshed each time. (this timing can be changed by editing the probe definitions/package).

From version 2.6 the probe is changed from Timed into a daemon. This means that it will not restart every 10 hours anymore.

**Note3**: in case of errors we will restart the probe after the configured parameter value in the config file for: restart\_after .

Logic for a remove from maintenance request:

* List all servers in maintenance by the query:

select distinct cs.name, cs.ip, cs.origin,ms.schedule\_id, ms.SCHEDULE\_NAME, ms.DESCRIPTION, max(CONVERT(VARCHAR(50), ms.START\_TIME,120)) as 'Start Time', max(CONVERT(VARCHAR(50), mw.END\_TIME, 120)) as 'End Time',msm.CS\_ID from CM\_COMPUTER\_SYSTEM cs with(nolock) INNER JOIN MAINTENANCE\_SCHEDULE\_MEMBERS msm with(nolock) ON cs.cs\_id=msm.CS\_ID INNER JOIN MAINTENANCE\_WINDOW mw with(nolock) ON msm.SCHEDULE\_ID=mw.SCHEDULE\_ID INNER JOIN MAINTENANCE\_SCHEDULE ms with(nolock) ON msm.SCHEDULE\_ID=ms.SCHEDULE\_ID WHERE mw.END\_TIME >=GETDATE() group by name,ip,origin,ms.schedule\_id,ms.SCHEDULE\_NAME,ms.DESCRIPTION,msm.CS\_ID

* Check if your server name, given as second word matches the output of the query.
* For every matching servername we will use the REST callback: remove\_computer\_systems\_from\_schedule to remove the cs\_id from the schedule id
* The results of this REST operation are logged in probe\_maint\_on.log

# 2 - Setup Perl environment

* download from: http://strawberryperl.com/releases.html the file:
	+ strawberry-perl-5.14.2.1-64bit.msi (it's a **must** that you download version 5.14.2)
* install the msi in: c:\Strawberry64 (or any other directory)
* verify that the following 3 directories are in the system path:
	+ C:\strawberry64\perl\bin
	+ C:\strawberry64\perl\site\bin
	+ C:\strawberry64\c\bin
* refresh the command prompt or reboot to activate the path
* deploy the probe: SDK\_Perl to the main UIM server (this creates: C:\Program Files (x86)\Nimsoft\perllib)
* copy the directories under: C:\Program Files (x86)\Nimsoft\perllib to C:\strawberry64\perl\lib
* now you are ready to install additional packages via cpan (from command prompt): (you need internet access because these modules are get directly from cpan):
	+ - cpan install Crypt::RC4
		- cpan install XML::Simple
		- cpan install HTTP::Request
		- cpan install DBD::ODBC (here you will receive some messages, but it's normal)
		- cpan install MIME::Base64
		- cpan install Time::Piece
		- cpan install Time::Seconds
		- cpan install LWP::UserAgent
		- (cpan install -f WWW::Mechanize) (generates messages and can take a long time to complete)
		- cpan install PAR::Packer
		- cpan install MIME::Lite
		- cpan install Data::GUID
		- cpan install Sys::HostAddr
		- (cpan install XML::LibXML)
		- cpan –fi Statistics::LineFit
		- cpan install Math::Spline
		- cpan install DBD::CSV
		- (cpan install DBD::Chart)
		- cpan install Net::SNMP
		- pip http://strawberryperl.com/package/kmx/perl-modules-patched/Crypt-OpenSSL-Random-0.04\_patched.tar.gz
		- (cpan -fi Net::SSL::ExpireDate) (generates a lot of messages and can take some time)
		- cpan install Win32::Console
		- cpan install Term::ANSIColor
		- cpan install Win32::Console::ANSI
		- (cpan install Parallel::ForkManager)

**Note**: the lines between () are not needed to run the probe or utility but are packages that are also installed on the original Perl directory received from CA services. (and can be needed if you run other customized reports/tools/probes)

# 3 – Maintenance Report

By default you are running blind regarding the devices that are added into maintenance.

We created a “basic” report that can help you to have an idea what devices are still in maintenance at a certain moment.

Example report:



**Note**: this report shows that we have 25 devices in maintenance and that they are all “new” added devices.

A second example shows the same 25 devices, but in the meantime somebody added 21 other devices:



This basic procedure is a 3 step script:

* step1 will extract, with BCP.EXE all devices in maintenance into a txt file (.bat file that creates the file with bcp.exe)
* Once step1 is executed 2 times step2 will compare the 2 extracted files and produce the above report. (Perl script)
* As last step this report is mailed (optionally) via blat

If you are interested in this script mail me at: luc.christiaens10@telenet.be