Copyright Notice

This online help system (the "System") is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This System may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This System is confidential and proprietary information of CA and protected by the copyright laws of the United States and international treaties. This System may not be disclosed by you or used for any purpose other than as may be permitted in a separate agreement between you and CA governing your use of the CA software to which the System relates (the “CA Software”). Such agreement is not modified in any way by the terms of this notice.

Notwithstanding the foregoing, if you are a licensed user of the CA Software you may make one copy of the System for internal use by you and your employees, provided that all CA copyright notices and legends are affixed to the reproduced copy.

The right to make a copy of the System is limited to the period during which the license for the CA Software remains in full force and effect. Should the license terminate for any reason, it shall be your responsibility to certify in writing to CA that all copies and partial copies of the System have been destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS SYSTEM “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS SYSTEM, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The manufacturer of this System is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2014 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Legal information on third-party and public domain software used in this product is documented in the Third-Party Licenses and Terms of Use (http://docs.nimsoft.com/prodhelp/en_US/Library/Legal.html).
Contact CA

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback about Product Documentation

Send comments or questions about CA Technologies product documentation to nimsoft.techpubs@ca.com.

To provide feedback about general CA Technologies product documentation, complete our short customer survey which is available on the support website at http://ca.com/docs.
# Contents

**Chapter 1: e2e_appmon 2.2**

**Chapter 2: e2e_appmon Overview v2.2**

**Chapter 3: e2e_appmon Probe Deployment**
- Prerequisites ................................................................. 13
- Supported Platforms .................................................. 13
- System Requirements .................................................. 13
- Software Requirements ............................................... 14
- Monitoring System Requirements .................................. 14
- Probe Deployment Information ..................................... 14

**Chapter 4: Upgrade and Migration Considerations**

**Chapter 5: e2e_appmon Configuration v2.2**
- Probe Configuration Interface Installation for e2e_appmon .................................................. 17
- Probe Defaults ............................................................... 17

**Chapter 6: e2e_appmon Configuration Interface v2.2**
- The Setup Tab ............................................................... 20
- The Status Tab ................................................................. 22
- The Messages Tab ......................................................... 28
- The Variables Tab .......................................................... 30
- The Scripts Tab ............................................................... 32

**Chapter 7: e2e_appmon API v2.2**
- nimGetEnv ........................................................................... 36
- nimGetEnvEx ...................................................................... 37
- nimStartRun ....................................................................... 38
- nimProcessExist ................................................................ 39
- nimWaitForWebContents .................................................. 39
- nimWaitForWebContentsEx ............................................... 40
- nimWaitForWindow ........................................................... 41
- nimWaitForWindowText ...................................................... 41
Chapter 1: e2e_appmon 2.2

This section contains the following topics:

e2e_appmon Overview v2.2 (see page 9)
Documentation Changes (see page 11)
Chapter 2: e2e_appmon Overview v2.2

The e2e_appmon probe runs the precompiled NimRecorder scripts for monitoring the response time and other performance parameters. Each transaction runs for a certain time limit and generates the QoS messages from the script. The probe also measures the total run time of each transaction. The e2e_appmon probe also monitors the availability of the client applications.

The other related concepts of the e2e_appmon probe are as follows:

- **Alarms**: You can specify threshold values for generating the alarms when these values are breached.
- **QoS**: You can select the QoS option for generating the QoS messages on total run time for profiles.
- **Scripts**: The e2e_appmon probe runs scripts at specified intervals to monitor the availability and the response time of the target applications. Using the NimRecorder (shipped with e2e_appmon_dev probe), you can make your own scripts. Using the e2e_appmon API, you can include checkpoints within your scripts. The scripts must be compiled on the actual machines where they are intended to run before they are invoked.

**Important!** Do not run other applications or tasks on the monitoring computer, as it disrupts the monitoring and measurement process of the probe.

**Note:** The e2e_appmon probe has certain limitations to use Optical Character Recognition (OCR) in the scripts. So, instead of using OCR, you can use the bitmap synchronization (synchronization on an image) to use the text logo.

- **Sample Script**: The probe is deployed with a sample script. Activate the probe and run the sample script. Compile the script on the target computer before executing it. If the script is executed before compiling, it results in an error message. For example, the error message is: **Error at line 348: Impossible to load the module.**

A compiled script is not delivered because the script is compiled based on each operating system setup. Therefore, the compiled script can run only on similarly configured computers.
Prerequisites

- **Probe Editions**: The e2e_appmon probe is available in two different editions, which are as follows:
  - The runtime edition of the probe enables you to run precompiled scripts.
  - The developer edition of the probe, the `e2e_appmon_dev`, lets you create the scripts and include checkpoints within those scripts. The e2e_appmon_dev probe can measure intermediate times of each process, apart from the total runtime of the script. In addition, some help files for developing scripts are also included.

- **The NimRecorder**: You can use NimRecorder, which is included in the probe package, for writing your own scripts. The NimRecorder can be launched from **Start > All Programs > Nimsoft Monitoring > E2E Scripting**, using the following menu options:
  - Compile *
  - Help *
  - Open Script *
  - Run Script
  - Script Editor *
  - Spy *
  - Uninstall NimRecorder

  **Note**: The options marked with an asterisk (*) are available only in the developer edition of the probe or after installing the NimRecorder manually. Refer the **Troubleshooting and FAQs** section for installing the NimRecorder manually.

The e2e_appmon probe is now available with NimRecorder 5.1 for creating and executing scripts for 64-bit platforms. The NimRecorder 5.1 also supports browser-based scripting for Internet Explorer, Mozilla Firefox, and Google Chrome. Refer the NimRecorder 5.1 help for supported platforms and browser versions with other relevant information.

Access the NimRecorder help from the **Start > All Programs > Nimsoft Monitoring > E2E Scripting > Help** location. This help file is also available from the **Help** menu available in the NimRecorder application Window.

Drop an email to **info@wintask.com** for troubleshooting your script related issues and support on NimRecorder 5.1.

More information:

- e2e_appmon API v2.2 (see page 35)
- E2E Application Response Monitoring (e2e_appmon) Troubleshooting v2.2 (see page 53)
## Documentation Changes

This table describes the version history for this document.

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>What’s New?</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>November 2014</td>
<td>■ Updated the <strong>Overview</strong>, <strong>Software Requirements</strong>, and <strong>Limitations</strong> section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated the NimRecorder to version 5.1.</td>
</tr>
<tr>
<td>2.2</td>
<td>February 2014</td>
<td>■ Updated the <strong>Overview</strong> section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated the <strong>Prerequisites</strong> section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated the <strong>Software Requirements</strong> section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated the <strong>Upgrades and Migrations</strong> section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated the <strong>Setup Tab</strong> section field descriptions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated the <strong>Troubleshooting and FAQs</strong> section.</td>
</tr>
<tr>
<td>2.1</td>
<td>June 2013</td>
<td>■ Added the <strong>Troubleshooting and FAQs</strong> section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated the description of the <strong>Scripts</strong> tab fields that are based on the updated functionality.</td>
</tr>
<tr>
<td>2.0</td>
<td>March 2013</td>
<td>■ Added the functionality of how to create a script package, which can be distributed on the target robot for monitoring.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Upgraded the NimRecorder to version 4.0.</td>
</tr>
<tr>
<td>1.9</td>
<td>November 2012</td>
<td>■ Updated the NimRecorder to version 3.8.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Enabled the QoS source override for script generated QoS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Fixed the scheduling functionality.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ The alert metrics default settings have been added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Added the probe defaults.</td>
</tr>
<tr>
<td>1.8</td>
<td>September 2010</td>
<td>NimRecorder updated to version 3.7a.</td>
</tr>
</tbody>
</table>
Prerequisites

Related Documentation

- Documentation for other versions of the e2e_appmon probe
- The Release Notes for the e2e_appmon probe
- Monitor Metrics Reference Information for CA Unified Infrastructure Management Probes
  (http://docs.nimsoft.com/prodhelp/en_US/Probes/ProbeReference/index.htm)
Chapter 3: e2e_appmon Probe Deployment

This section contains the prerequisites, system requirements, and deployment information for the e2e_appmon probe.

This section contains the following topics:
- **Prerequisites** (see page 13)
- **Supported Platforms** (see page 13)
- **System Requirements** (see page 13)
- **Software Requirements** (see page 14)
- **Monitoring System Requirements** (see page 14)
- **Probe Deployment Information** (see page 14)

### Prerequisites

All the users, included in the probe profile, must have administrative access (read and write permissions) to the e2e_appmon probe and the e2e_appmon_dev directory.

The WTBho Class, a NimRecorder add-on, must be enabled on supported Web for executing the Web-based scripts.

### Supported Platforms

Refer to the [Compatibility Support Matrix](#) for the latest information about supported platforms. See also the [Support Matrix for Probes](#) for more specific information about the probe.

### System Requirements

The e2e_appmon probe is installed on systems with the following minimum resources:

- Memory: 2-4GB of RAM. The OOB configuration of the probe requires 256 MB of RAM
- CPU: 3-GHz dual-core processor 32 bit or 64 bit
Software Requirements

The e2e_appmon probe requires the following software environment:

- Nimsoft Monitor Server 7.1 to 7.6 or CA Unified Infrastructure Management 8.0 or later
- Robot 7.1 or later
- JRE 1.6 or later

Monitoring System Requirements

The e2e_appmon probe monitors response time and availability of the client applications.

Probe Deployment Information

There are three ways to distribute archive packages. You can distribute the package within the web-based Admin Console (for supported probes), from within Infrastructure Manager, or use the standalone Distribution application. See Probe Deployment for more information on deploying probes.
Recompile your existing scripts using the new NimRecorder while upgrading the e2e_appmon probe older than 1.62 version. The new NimRecorder is deployed with probe package for recompiling the existing scripts. The scripts become compatible with the new NimRecorder once recompiled. The NimRecorder is available at the Start > All Programs > Nimsoft Monitoring > E2E Scripting.

Uninstall any previous version of the NimRecorder and ensure that the bin folder under the Nimsoft/e2e_scripting folder does not exist anymore. Delete the bin folder, if exists, and then install the latest version of the probe. Take backup of all existing scripts before uninstalling the NimRecorder.

In case, you are upgrading the e2e_appmon_dev probe from 2.0x to 2.2x and the probe does not upgrade the NimRecorder to NimRecorder 5.1. Navigate to <Nimsoft Installation drive>/Nimsoft/probes/Application/e2e_appmon/install and double-click the nimrecorder50.msi for installing it manually.

Note: If you are upgrading the e2e_appmon from earlier wintask probe, refer the e2e_appmon probe 2.0 documentation or earlier.
Chapter 5: e2e_appmon Configuration v2.2

The e2e_appmon probe is configured to monitor response time and availability of the client applications. You can create monitoring profiles to run the scripts at the specified time interval. You can define threshold values for generating alarms and can configure QoS for generating the performance data.

Double-click the e2e_appmon probe in the Infrastructure Manager, the probe GUI with various tabs and fields for configuring the probe appears.

This section contains the following topics:
- Probe Configuration Interface Installation for e2e_appmon (see page 17)
- Probe Defaults (see page 17)
- e2e_appmon Configuration Interface v2.2 (see page 19)

Probe Configuration Interface Installation for e2e_appmon

The probe configuration interface is automatically downloaded and installed by the Infrastructure Manager when the probe is deployed on a robot.

Probe Defaults

When deploying a probe for the first time on a robot, some configurations are deployed automatically. These probe defaults can be Alarms, QoS, and Profiles, which saves time to configure the default settings. These probe defaults are available on a fresh installation only. The fresh installation implies that instance of that probe is not already available on that robot in activated or deactivated state.

The default profile of the probe generates the QOS_E2E_EXECUTION QoS. The probe user must activate the default profile for generating the default QoS. The default script is supported only on Internet Explorer.
Chapter 6: e2e_appmon Configuration Interface v2.2

The e2e_appmon probe is configured for creating the monitoring profiles. Each monitoring profile can have more than one script to be executed. You can also define thresholds for generating alarms when the script execution time exceeds the limit. The QoS messages are also configured for generating and saving the response time data of the application.

More information:

- The Setup Tab (see page 20)
- The Status Tab (see page 22)
- The Messages Tab (see page 28)
- The Variables Tab (see page 30)
- The Scripts Tab (see page 32)

This section contains the following topics:

- The Setup Tab (see page 20)
- The Status Tab (see page 22)
- The Messages Tab (see page 28)
- The Variables Tab (see page 30)
- The Scripts Tab (see page 32)
The Setup Tab

The Setup tab is used to configure general settings of the probe, which are common for all profiles.

![Setup Tab Image]

**Run as user**

Defines the login user name and password of the target computer where the probe runs the script.

**User check to prevent scripts to be run from the wrong user**

Prevents the script from unauthorized users from being executed on the target system.

**Reset registry settings right after the user is logged on**

Resets the registry settings after successful login of the user. Enabling this option makes the target system less vulnerable to malicious attacks over a network. Also, when the remote desktop connection (RDP) is used, a legal notice is displayed. However, you ensure the presence of the automatic logon settings in the registry.

**Note:** This option aborts the logon process on a slow processing system.

**Log Level**

Sets the level of details that are written in the log file.

**Note:** Select a lower log level during the normal operation and minimize the disk consumption. You can increase the log level while debugging.
Log Size
Defines the size of the log file for the probe for writing the log messages. By default, the size is 100 KB.

Note: When the specified size is reached, the content of the file is cleared.

Run properties
Default run interval (seconds)
Defines how often the script runs.

Default (max) run time (seconds)
Defines how much time the script is allowed to use on one run. This value can be overridden for each of the profiles that are defined under the Status tab.

Command
Defines the path of the directory where the TaskExec.exe file is located. The default location of the TaskExec.exe file is program files (x86)\Nimsoft\e2e_scripting\bin. This TaskExec.exe file is used for executing the compiled scripts on both 32-bit and 64-bit platforms.

.ROB File Directory
Defines the path of the directory where the compiled script files are stored.

Note: The default relative paths of the Command and .ROB File Directory fields does not work. Remove these default paths and configure the absolute path manually.

Suspend
Suspends the execution of scripts. However, it is still possible to retrieve the last executed status and view the screenshots of the script.
The Status Tab

The **Status** tab is used to configure monitoring profiles for the e2e_appmon probe. This tab also displays a list of defined monitoring profiles and allows you to edit and delete the profiles.

The **Status** tab contains the following fields:

- **Name**
  Displays the name of the profile.

- **ROB file**
  Displays the name of the script (a ROB file) run by the probe using different profiles.

- **Last started**
  Displays the last time that the probe was started.

- **Running**
  Displays the running state of the script. **Yes** means running and blank means not running.

- **Time used (last run)**
  Displays the time that is used in last execution of the script.

- **Return code (last run)**
  Displays the status code that is returned after last execution of the script.
Times started
Displays the number of times the script is executed after the e2e_appmon probe is activated last time.

Times killed
Displays the number of times the script is killed after the e2e_appmon probe is activated last time.

Times failed to start
Displays the number of times the script has failed to start after the e2e_appmon probe is activated last time.

Max. Run time
Displays the maximum time for running a profile.

For adding, editing, or deleting the profiles right-click in the window. You can also view the screen dump from the computer hosting the probe.

The Profile properties dialog appears on selecting the Add profile or Edit profile option.
The **Profile properties** dialog contains the following field:

**Name**

Displays the name of the profile.

The **Profile properties** dialog contains the following tabs:

- **Run properties** (see page 24)
- **Scheduling** (see page 26)
- **Advanced** (see page 27)

### The Run Properties Tab

The **Run properties** tab is used to specify the profile script. This tab is also used to configure the runtime environment for the script by handling timeout and error situations.

![Profile properties dialog](image)
The **Run properties** tab contains the following fields:

**Compiled script (.rob file)**
- Specifies a ROB file from the drop-down list. You can select the files from the directory as specified in the **ROB file directory** field of the **Setup** tab.

**Arguments**
- Defines the parameter, if any, required to run the script.

**Max. run time**
- Defines the maximum time for which the script is allowed to run.
  - **Note:** This value overrides the default value that is specified in the **Setup** tab.

**Expected return value = 0**
- Checks the return value from the scripts and determine whether the script is executed successfully or not.

**On timeout**
- **Dump screen on timeout**
  - Saves the snapshot when the script execution time exceeds the limit. By right-clicking the profile under the **Status** tab and selecting the **View screendumps** option, you can view the screen dump for the profile.

  **Kill processes on timeout**
  - Terminates the other associated processes with the script on the timeout.

**On error return**
- **Dump screen on error**
  - Saves a snapshot, when the script does not return an expected result. The snapshot for the profile is displayed by right-clicking the profile under the **Status** tab and selecting the **View screendumps** option. This option gets enabled after selecting the **Expect return value = 0** option.

  **Alarm message**
  - Specifies the alarm message from the drop-down list when the script returns an error.

**Cleanup on timeout or error return**
- Specifies the **ROB** file (a compiled script) from the drop-down list, which runs when the script is timed out or the script returns an error. This script is used to clean up all the actions to run other scripts.

**Send QoS on total run time**
- Generates the QoS data that is related to the **script run time**.
  - **Note:** The script itself is configured to send QoS data using the developer version of the probe.
The Scheduling Tab

The Scheduling tab is used to schedule the run time of the script. This tab also allows you to configure when the script can or cannot run.

The Scheduling tab contains the following fields:

**Run Interval**

Specifies the interval between two consecutive executions of each profile. The interval can either be on every probe interval or be at a specified time interval.

**Only run on**

Provides the following options for restricting the script execution:

- **In time ranges**
  
  Specifies a comma-separated list of time ranges. For example, **10:05-11:30, 12:34-16:00**.
Weekdays

Specifies a comma-separated list of week days or range of week days. For example, mon, thu-sat.

Days of months

Specifies a comma-separated list of month dates. For example, 2-5,14-16,21.

Do not run on

Defines a comma-separated list of dates (in day.month format) when the script does not run. For example, 5.1, 9.1.

The Advanced Tab

The Advanced tab allows you to select the source of the QoS and Alarm messages.

Local machine

Specifies the machine name, where the probe is hosted.

Note: This machine name appears as the source in QoS and alarm messages.
Override with

Defines a custom hostname, which appears as the source in QoS and alarm messages.

The Messages Tab

The Messages tab is used to maintain a pool of alarm messages. These alarm messages are used across the monitoring profiles of the e2e_appmon probe. By default, there are four alarm messages. You can add, edit, and delete messages to this message pool.

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Message Description</th>
<th>Default Level</th>
<th>Subsystem</th>
<th>Default for</th>
<th>Error Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm on start error</td>
<td>Sproville Failed $failed n...</td>
<td>major</td>
<td>1.2.3.7</td>
<td>on_disable</td>
<td>as fail</td>
</tr>
<tr>
<td>Alarm on interval breach</td>
<td>Sproville Did not complete</td>
<td>minor</td>
<td>1.2.3.7</td>
<td>on_kill</td>
<td>as fail</td>
</tr>
<tr>
<td>Alarm on process kill</td>
<td>Sproville Script returned</td>
<td>minor</td>
<td>1.2.3.7</td>
<td>on_unexpected</td>
<td>as fail</td>
</tr>
<tr>
<td>Disable after (specific number of) errors</td>
<td>Sproville. Unable to start</td>
<td>warning</td>
<td>1.2.3.7</td>
<td>on_start_error</td>
<td>as fail</td>
</tr>
<tr>
<td></td>
<td>Sproville. Unable to start</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the drop-down lists of the Alarm message setup grid, you can specify the alarm message to be issued for four different alarm conditions:

- Alarm on start error
- Alarm on interval breach
- Alarm on process kill
- Disable after (specific number of) errors and send message
- Alarm on unexpected returned value
The following options appear on right-clicking the message list:

- **Add message**: This option enables you to define new alarm messages.
- **Message properties**: This option enables you to edit one of the existing alarm messages.
- **Remove messages**: This option enables you to delete alarm messages.

On selecting the **Add message** or **Message properties** option, the **Message properties** dialog appears.
The **Message properties** dialog contains the following fields:

**Name**
Defines a unique name of the message. This name is used to refer to the message from the profiles.

**Default for**
Specifies the alarm situations to be selected as the default alarm message for a specific type of alarm message.

**Text**
Defines the alarm message text. The following variables can be used:

- **$profile**: profile name.
- **$failed**: number of consecutive failures.
- **$sec**: seconds delay on interval breach.
- **$error**: error message on start error.
- **$return**: actual return code from last run.
- **$expected_return**: expected return code after successful run of the script.

**Level**
Specifies the severity level that is assigned to the alarm messages.

**Subsystem**
Defines the **subsystem_ID** of the alarms that the watcher generates. A string or an id is managed by the nas probe.

**The Variables Tab**

The **Variables** tab lets you define variables, which are used in the script. For example, you are using a password in the script and want to encrypt and protect the password being presented in the raw form. In such cases, define the password as a variable and select the **Encrypt** option.

The **Variables** tab contains the following field:

**Quality of Service Variables**

The probe sends QoS messages on script run time. The probe exports the QoS to the script also enabling the script to send QoS.
Three options Add variable, Variable properties, and Remove variable appear on right-clicking the Variables list. On selecting the Add variable or Variable properties option, the Variable properties dialog is displayed. The Variable properties dialog is used to add or modify the variable properties.

The Variable properties dialog contains the following fields:

Variable name
- Defines the variable name, which is unique for each variable.

Crypted
- Encrypts the variable value and prevents the same being displayed in the human readable format.

Variable value
- Defines a value to be assigned to the variable.
The Scripts Tab

The **Script** tab allows you to create independent and deployable script packages. These packages can be deployed on the target robot (similar to other probe packages) for monitoring. You can add more than one scripts and their dependent files to one deployable package.

**Important!** Refer to the Limitations section, which provides information to create and deploy the script packages.

The **Scripts** tab contains the following fields:

**Name**

Defines the name of the script package to be made.

**Version**

Specifies the version number of the script package to be created. By default, its value is 1.0.

**Description**

Defines a short description of the script package.
Scripts

Displays the list of scripts present in the path that is specified in the ROB file directory field of the Setup tab. The list is available only when the probe is activated. More than one script can be added to the package file.

The script execution settings are taken from the associated profile. If more than one profile is associated to a script, double-click the script name and select the profile name.

![Profiles]

The selected profile settings are used to execute the script. Double-click the profile name for editing the profile properties of the script, while selecting the profile.

If no profile is associated with a script, then such script is displayed in red (color) and default settings are used for its execution. You can double-click the script name to define the properties for the script.

Path/Dependent Files

Allows you to browse the dependent files (for example, .bmp files) required to execute the script. You can select more than one file of the same directory in one attempt. This field remembers the last navigation path.

**Note:** The script uses an absolute path of the dependent files.

Add Files

Allows you to add the path of dependent files in the Paths list-box.

Paths List Box

Displays the path list of dependent files to be included, while generating the package. You can right-click any of the dependent files and can select the **Delete** option to remove the selected file from the list.
Publish to Archive

Publishes the script package to the Archive directory of the primary hub.

Limitations

The limitations of creating and deploying the script packages are as follows:

- Script that is recorded on a particular OS can only be deployed on that OS. For example:
  - If recorded on Windows 7, the script does not work on Windows Server 2012 R2 or Windows 8.
  - If recorded on the 64-bit robot, the script can only be deployed on the 64-bit robot.
  - If recorded on OS with IE9, the script can only be deployed on OS with IE9.
  - If recorded on OS, where a dependent file is on the E:\ drive, the target host must also have an E:\ drive present.
  - If recorded with the NimRecorder 3.9, the target host must also have the NimRecorder 3.9.

- The absolute path of dependent files is provided while recording the scripts.

- To create a script package, a profile is created for the scripts that are included in the script package. If there is no profile, a default profile is created with the ROB file name in the configuration file after deploying the package.

  **Note:** As of now, you cannot edit the default settings.

- A profile and a script cannot have the same name on the target robot, otherwise either the profile name, or the script name is overridden. The robot does not display any warning message while deploying the probe.

- The dependent files can only be added from the C:\ drive. This process is in line with how current script path is configured in the probe.

- All the files are copied to the folders according to the robot on which script is recorded, after deploying the package.

- The robots, on which script package is deployed, must be running the e2e_appmon probe. The **Distribution Manager** deploys the script package without any validation.

- You cannot deploy only one script from a package (in case the package contains more than one script) to the robot.
Chapter 7: e2e_appmon API v2.2

The **e2e_appmon developer edition** allows the developer to include checkpoints within the NimRecorder script. This functionality enables the e2e_appmon probe to measure intermediate times of each process in addition to the total runtime of the script.

The package contains the CA Nimsoft API to be used with the e2e_appmon probe. The API allows the NimRecorder to use the functions in the script. The functions enable the programmer to access alarm and quality of service functions from the Nimsoft API in their in-house developed scripts.

This API contains core Nimsoft functions (for sending Alarms and QoS) and other supporting functions to make scripting easier.

While using the e2e_appmon:
- Your script must begin with `Include Nimsoft-functions.src`.
- The `Nimsoft-functions.src` file must be in the same directory as the script.
- The file `nimmacro.dll` must be in the same directory as `Nimsoft-functions.src`.

**SDK_appmon** offers advanced scripting by offering the following functionalities:
- Gathering multiple QoS points to identify the bottlenecks
- Pre and post run cleanup
- Synchronization
- Error handling
- Tuning your script to minimize the resource usage
- Hiding or encrypting the passwords
- Working with Citrix
nimGetEnv

This section contains the following topics:

nimGetEnv (see page 36)
nimGetEnvEx (see page 37)
nimStartRun (see page 38)
nimProcessExist (see page 39)
nimWaitForWebContents (see page 39)
nimWaitForWebContentsEx (see page 40)
nimWaitForWindow (see page 41)
nimWaitForWindowText (see page 41)
nimAlarm (see page 42)
nimAlarmSimple (see page 42)
nimInit (see page 43)
nimEnd (see page 43)
nimSetVariable (see page 44)
nimQoSSendTimer (see page 44)
nimQoSSendNull (see page 45)
nimQoSSendValue (see page 45)
nimQoSSendValueStdev (see page 46)
nimQoSStart (see page 46)
nimQoSStop (see page 47)
nimQoSReset (see page 47)
nimQoSGlobal (see page 48)
nimInitWithMax (see page 48)

nimGetEnv

nimGetEnv$(var$,def$)

Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>var$</td>
<td>The environment variable to fetch.</td>
</tr>
<tr>
<td>String</td>
<td>def$</td>
<td>Default if the variable is not in the environment.</td>
</tr>
</tbody>
</table>
nimGetEnvEx$(var$, def$, ask, pass).

### Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>var$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td></td>
<td>The environment variable to fetch.</td>
</tr>
<tr>
<td>String</td>
<td>def$</td>
<td>Default if the variable is not in the environment.</td>
</tr>
<tr>
<td>Number</td>
<td>ask</td>
<td>If set, the dialog asks the user to input the string.</td>
</tr>
<tr>
<td>Number</td>
<td>pass</td>
<td>If set, the dialog masks out the input string.</td>
</tr>
</tbody>
</table>

**Returns**

String environment value

**Description**

Returns the contents of the environment variable referred to in var$. If the variable is not in the environment, by default it returns the value of def$. 

**Example**

```nim
home$ = nimGetEnv("HOMEDRIVE", "C:" )
```

Dialog used in nimGetEnvEx to allow user to enter a value/string:
```
begin dialog dlgAsk 225,130,200,110
caption "Please enter value"
delpushbutton "&OK", btnOk, 10,50,65,20
editext txtAsk$, 10,10,170,24
enddialog
```

Dialog that is used in nimGetEnvEx to allow the user to enter a password:
```
begin dialog dlgAskSec 225,130,200,110
caption "Please enter value"
delpushbutton "&OK", btnOk, 10,50,65,20
editext txtAsk$, 10,10,170,24, protected
enddialog
```
Returns

String value

Description

Returns the contents of the environment variable referred to in var$. If the variable is not in the environment, it defaults to def$. If ask is set to 1, the dialog prompts you to input the string. If the pass is set to 1, the dialog masks out the input string.

Example

user$= nimGetEnvEx(“Test-user”,“admin”,1,0)

pass$= nimGetEnvEx(“Test-pass”,“admin”,“”,1,1)

nimStartRun

nimStartRun(cmd$)

Parameters

<table>
<thead>
<tr>
<th>String</th>
<th>cmd$</th>
<th>The cmd$ to be executed from the Run entry on the Start menu.</th>
</tr>
</thead>
</table>

Returns

0: OK.

1: Failed to get the Run window.

Description

Allows you to execute the cmd$ from the Run entry on the Start menu. The caller ensures that the command to be executed uses <doublequote> and other special keys to make the command understandable to SendKeys.
nimProcessExist

nimProcessExist(procname$, killproc)

Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>procname$</td>
<td>Name of the process to be located and terminated.</td>
</tr>
<tr>
<td>Number</td>
<td>killproc</td>
<td>Terminates the specified process soft (if 1), and hard (if 2).</td>
</tr>
</tbody>
</table>

Returns

0: process not found.

1: process found.

Description

Locate named process (procname) and terminates the process if (killproc) is (1=soft,2=hard).

Note: The method is deprecated and in wintask 2.6a the function killapp() was added and does the same thing natively.

nimWaitForWebContents

nimWaitForWebContents(pageid$, contents$, load_timeout)

Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>pageid$</td>
<td>The web page to search for the contents specified.</td>
</tr>
<tr>
<td>String</td>
<td>contents$</td>
<td>The contents to search for in the web page specified.</td>
</tr>
<tr>
<td>Number</td>
<td>load_timeout</td>
<td>The number of seconds to wait for match before timeout.</td>
</tr>
</tbody>
</table>
nimWaitForWebContentsEx

Returns

0: timeout, no match.
1: match.

Description

Waits for matching contents of the web page for (load_timeout) seconds.

nimWaitForWebContentsEx

nimWaitForWebContentsEx(pageid$, contents$, content_fail$, load_timeout)

Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>pageid$</td>
<td>The web page to search for the contents specified.</td>
</tr>
<tr>
<td>String</td>
<td>contents$</td>
<td>The contents to search for in the web page specified.</td>
</tr>
<tr>
<td>String</td>
<td>content_fail$</td>
<td>The failure match to be applied as failure match when the content specified fails to match</td>
</tr>
<tr>
<td>Number</td>
<td>load_timeout</td>
<td>The number of seconds to wait for match before timeout.</td>
</tr>
</tbody>
</table>

Returns

0: timeout, no match.
1: match.

Description

Waits for matching contents of the web page for (load_timeout) seconds. Applies a failure match as well when the content fails to match.
**nimWaitForWindow**

nimWaitForWindow(winid$, load_timeout)

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>winid$</td>
</tr>
<tr>
<td>Number</td>
<td>load_timeout</td>
</tr>
</tbody>
</table>

**Returns**

0: timeout, no match

1: match

**Description**

Waits for a window matching (winid$) to appear within (load_timeout) seconds

**nimWaitForWindowText**

nimWaitForWindowText(winid$, textstr$, load_timeout)

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>winid$</td>
</tr>
<tr>
<td>String</td>
<td>textstr$</td>
</tr>
<tr>
<td>Number</td>
<td>load_timeout</td>
</tr>
</tbody>
</table>

**Returns**

0: timeout, no match

1: match

**Description**

Waits for a window matching (winid$) containing the text string (textstr$) to appear within (load_timeout) seconds
nimAlarm

nimAlarm(severity,msg$,supp$,subsys$)

Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>severity</td>
<td>The severity level of the alarm message.</td>
</tr>
<tr>
<td>String</td>
<td>msg$</td>
<td>The alarm message text.</td>
</tr>
<tr>
<td>String</td>
<td>supp$</td>
<td>Suppression ID.</td>
</tr>
<tr>
<td>String</td>
<td>subsys$</td>
<td>The id of the subsystem, identifying the module the alarm is related to.</td>
</tr>
</tbody>
</table>

Returns

0 = OK

Description

Sends alarm message containing severity level, message text, checkpoint id, and subsystem id.

Example

nimAlarm(5,"critical alarm","script-name","E2E-appmon")

nimAlarmSimple

nimAlarmSimple(severity,msg$)

Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>severity</td>
<td>The severity level of the alarm message.</td>
</tr>
<tr>
<td>String</td>
<td>msg$</td>
<td>The alarm message text.</td>
</tr>
</tbody>
</table>
Returns

0 = OK.

Description

Sends alarm message containing severity level and message text. The rest is provided by the global variables suppression_id$ and subsystem$. See also nimSetVariable.

Example

nimAlarm(5, "critical alarm")

nimInit

nimInit()

Parameters

None

Returns

0 = OK

Description

Initializes the Nimsoft SDK components. Must be run if using QoS.

nimEnd

nimEnd()

Parameters

None.

Returns

0 = OK

Description

Unloads the Nimsoft components and releases memory.
nimSetVariable

nimSetVariable(variable$,value$)

Parameters

None

Returns

0 = OK

Description

Sets the global variable named (variable$) to a new value. Allows you to override the suppression-id or subsystem.

Example

nimSetVariable("suppression-id","script")

nimQoSSendTimer

nimQoSSendTimer(target$)

Parameters

<table>
<thead>
<tr>
<th>String</th>
<th>target$</th>
<th>Unique identifier for the QoS checkpoint.</th>
</tr>
</thead>
</table>

Returns

0 = OK.

Description

Send the recorded timer for the specified target.

Example

nimQoSSendTimer("citrix login")
nimQoSSendNull

nimQoSSendNull(target$)

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Target</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>target$</td>
<td>Unique identifier for the QoS checkpoint.</td>
</tr>
</tbody>
</table>

**Returns**

0 = OK

**Description**

Sends a NULL (invalid data) for the specified target.

**Example**

nimQoSSendNull("citrix login")

nimQoSSendValue

nimQoSSendValue(target$, value)

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Target</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>target$</td>
<td>Unique identifier for the QoS checkpoint.</td>
</tr>
<tr>
<td>Number</td>
<td>value</td>
<td>The value to send to the specified target.</td>
</tr>
</tbody>
</table>

**Returns**

0 = OK

**Description**

Sends the recorded value (when not using timers) to the specified target.

**Example**

nimQoSSendValue("xxx", 69)
nimQoSSendValueStdev

nimQoSSendValueStdev(target$, value$, stdev$)

Parameters

<table>
<thead>
<tr>
<th>String</th>
<th>target$</th>
<th>Unique identifier for the QoS checkpoint.</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>value$</td>
<td>Number represented as a string, i.e. &quot;12.45&quot;.</td>
</tr>
<tr>
<td>String</td>
<td>stdev$</td>
<td>Number represented as a string, i.e. &quot;12.45&quot;.</td>
</tr>
</tbody>
</table>

Returns

0 = OK.

Description

Sends the value and standard deviation (when not using timers).

nimQoSStart

nimQoSStart()

Parameters

None

Returns

Nothing

Description

Start the QoS timer
nimQoSStop

nimQoSStop()

Parameters
None

Returns
Nothing

Description
Stop the QoS timer

nimQoSReset

nimQoSReset()

Parameters
None

Returns
Nothing

Description
Reset the QoS timer
**nimQoSGetTimer**

nimQoSGetTimer()

**Parameters**

None

**Returns**

Timer in milliseconds

**Description**

Returns the stored (last) QoS Timer. If the nimQoSStop function has not been called then the time between nimQoSStart was called and the current time is returned.

**nimInitWithMax**

nimInitWithMax(qos_name$, source$, description$, long_unit$, short_unit$, max_value)

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>qos_name$</td>
<td>The qos name of the e2e_appmon probe.</td>
</tr>
<tr>
<td>String</td>
<td>source$</td>
<td>Source from where QOS is coming.</td>
</tr>
<tr>
<td>String</td>
<td>description$</td>
<td>e2e_appmon Script Run Time.</td>
</tr>
<tr>
<td>String</td>
<td>long_unit</td>
<td>Full unit ex. “milliseconds”.</td>
</tr>
<tr>
<td>String</td>
<td>short_unit</td>
<td>Abbr. ex “ms”.</td>
</tr>
<tr>
<td>Int</td>
<td>max_value</td>
<td>QOS max values.</td>
</tr>
</tbody>
</table>
Returns

0 = OK.

Description

This function allows you to set various QoS parameters.

Example

nimInitWithMax(QOS_E2E_EXECUTION, QOS_source1, description$, Milliseconds, ms, 100)
Chapter 8: e2e_appmon QoS Threshold Metrics

Many probes ship with the default QoS threshold values. The default threshold values provide an idea of the type of values to be entered in the fields. These threshold values are not necessarily recommended values. To aid in tuning thresholds and reducing false-positive alarms, this section describes the QoS metrics and provides the default QoS thresholds.

This section contains the following topics:

- e2e_appmon QoS Metrics (see page 51)
- e2e_appmon Alert Metrics Default Settings (see page 51)

**e2e_appmon QoS Metrics**

The following table describes the checkpoint metrics that can be configured using the e2e_appmon probe.

<table>
<thead>
<tr>
<th>Monitor Name</th>
<th>Units</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>QOS_E2E_EXECUTION</td>
<td>Milliseconds</td>
<td>Monitors the script execution time.</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**e2e_appmon Alert Metrics Default Settings**

This section contains the alert metric default settings for the e2e_appmon probe.

<table>
<thead>
<tr>
<th>Alarm Name</th>
<th>Warning Threshold</th>
<th>Warning Severity</th>
<th>Error Threshold</th>
<th>Error Severity</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Major</td>
<td>Profile is disabled by the probe.</td>
<td>1.0</td>
</tr>
<tr>
<td>Killed</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Minor</td>
<td>Profile did not complete on time and had to be killed.</td>
<td>1.0</td>
</tr>
<tr>
<td>ReturnError</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Minor</td>
<td>Script returned an unexpected value.</td>
<td>1.0</td>
</tr>
<tr>
<td>StartError</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Minor</td>
<td>Profile is not able to start and execute the script.</td>
<td>1.0</td>
</tr>
<tr>
<td>TimeBreached</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Warning</td>
<td>Profile does not start on time.</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Chapter 9: E2E Application Response Monitoring (e2e_appmon) Troubleshooting v2.2

This section contains the following topics:

- NimRecorder is not Deployed or Installed (see page 53)
- Scripts not Working Fine after Upgrade (see page 54)
- Scripts not Working due to Synchronization (see page 54)

NimRecorder is not Deployed or Installed

Symptom:

I am using the e2e_appmon probe deployed on my robot. However, the NimRecorder is not getting deployed to record new script and to edit existing scripts. The probe is deploying the script executor only.

Solution:

Instead of deploying the e2e_appmon probe, deploy the e2e_appmon_dev (1.91 or later) probe. The NimRecorder is deployed automatically.

OR

For the e2e_appmon probe (standard edition), install the NimRecorder manually.

Follow these steps:

1. Go to Start > All Programs > Nimsoft Monitoring > E2E Scripting and select the Uninstall NimRecorder.
   The Script Executor is removed.
2. Go to [Nimsoft Installation drive] > Nimsoft > probes > Application > e2e_appmon > Install folder.
3. Double-click the nimrecorder.msi.
   Note: The name of .msi installer file is based on the NimRecorder version. For example, it is nimrecorder50.msi for installing the NimRecorder 5.0.
4. Follow the instructions and install the NimRecorder.
   The help file of the NimRecorder is accessible at Start > All Programs > Nimsoft Monitoring > E2E Scripting > Help after installing the NimRecorder.
Scripts not Working Fine after Upgrade

Symptom
My existing scripts have stopped working after upgrading the probe.

Solution
Recompile your existing scripts with new NimRecorder and make them compatible with new NimRecorder.
Drop an email to info@wintask.com for troubleshooting your script-related issues and support on NimRecorder 5.1.

Scripts not Working due to Synchronization

Symptom
My existing scripts have stopped or lose focus while running the script through NimRecorder because of synchronization of text.

Solution
The UsePage and Web functions are used for synchronization process. The Internet Explorer (IE) uses the window name instance for the bitmap synchronization and it varies with each IE execution. Therefore, the wintask recommendation is to use the Topinstance function of NimRecorder instead of providing a constant instance number.

For example:
Pause 10 secs until
  Bitmap("test.bmp")
  InWindow("IEXPLORE.EXE\Internet Explorer_Server\WinTask_x64 Help - Internet Explorer\1*\2")
  InArea( 101, 172, 26, 41 )
For Topinstance, the constant instance number can be replaced with:
Pause 10 secs until
  Bitmap("test.bmp")
  InWindow("IEXPLORE.EXE\Internet Explorer_Server\WinTask_x64 Help - Internet Explorer\1*\Topinstance()")
  InArea( 101, 172, 26, 41 )

Drop an email to info@wintask.com for troubleshooting your script-related issues and support on NimRecorder 5.1.