E2E_CONV V1.37

Release history

Version	Author	Comments
1.0	Gijsbert Wiesenekker	Initial release.
1.1	Gijsbert Wiesenekker	Added support for ClickMouse
1.2	Gijsbert Wiesenekker	Fixed a couple of bugs. Changed MsgBox to MsgFrame and added support for alarms and total runtime.
1.3	Gijsbert Wiesenekker	Moved e2e_conv.pl functionality into e2e_conv-functions.src, timers can now be nested, added a screen-capture utility and added a LUA script to generate SLA/SLO definitions from e2e_appmon profiles.
1.4	Gijsbert Wiesenekker	Added an option to write the timing of steps to a text file so that the QoS messages can be send with the correct timestamp by the logmon probe.
1.5	Gijsbert Wiesenekker	Added the last line back to e2e_conv-functions.src that was lost upon import into the package. The screen-capture utility returned a corrupt process name on Windows 2003 due to different behaviour of the Windows API. Fixed.
1.6	Gijsbert Wiesenekker	Added a bat file to delete all screendumps older than a month from the Tmp directory.
1.7	Gijsbert Wiesenekker	Synchronize can also be called outside a BeginStep/EndStep pair. The screen-capture can be cancelled by right-clicking.
1.8	Gijsbert Wiesenekker	The screen-capture will show the name of the configuration file as a hint. You can re-use bitmaps by calling Synchronize with the same argument again. A second argument to Synchronize controls if the bitmap has to be found anywhere in the window or at the same location it was captured at. NOTE THAT THIS REQUIRES YOU TO UPDATE YOUR EXISTING SCRIPTS BY ADDING A SECOND ARGUMENT TO THE CALL TO SYNCHRONIZE. If the script aborts while testing it interactively a message box will be shown.
1.9	Gijsbert Wiesenekker	The script will now exit if a MsgBox says that the script will exit. Synchronize will no longer wait Stabilize seconds before taking a screenshot of the region you want to use for validation when the screen-capture utility has run. Clear-alarms are now sent. NOTE THAT THIS REQUIRES YOU TO UPDATE YOUR EXISTING SCRIPTS BY ADDING A ‘Dim ClearStatus(10)’ TO THE BEGINNING OF THE SCRIPT.
1.10	Gijsbert Wiesenekker	The time needed to search for bitmaps to validate steps can take several seconds. The script generates a QoS for this time and it is no longer included in the time needed for a step. NOTE THAT THIS WILL CHANGE YOUR EXISTING

Version	Author	Comments
		BASELINES AND THRESHOLDS AND REQUIRES YOU TO UPDATE YOUR EXISTING SCRIPTS BY ADDING A 'Dim BitmapTimer(10)' TO THE BEGINNING OF THE SCRIPT.
1.11	Gijsbert Wiesenekker	The time needed to search for bitmaps to validate steps can take several seconds. You can specify how you want to handle this time: include it in/exclude it from the time needed for a step and/or send a QoS for it. NOTE THAT EXCLUDING IT FROM THE TIME NEEDED FOR A STEP WILL CHANGE YOUR EXISTING (PRE-VERSION 1.10) BASELINES AND THRESHOLDS.
1.12	Gijsbert Wiesenekker	As KillApp sometimes raises errors on Windows 7 it has been replaced by `nircmd killprocess`. Fixed handling of errors outside BeginStep/EndStep statements.
1.13	Gijsbert Wiesenekker	The e2e_conv version and current working directory are now shown while developing the script. When you deploy a new version of e2e_conv-functions.src you have to recompile all your existing scripts. e2e_comp.txt describes how you can do this using the nexec probe.
1.14	Gijsbert Wiesenekker	A trace is now shown when the script aborts.
1.15	Gijsbert Wiesenekker	One elapsed time was reported in hundreds of a second instead of milli-seconds.
1.17	Gijsbert Wiesenekker	Until now the time needed to search for a bitmap was an estimate. It is now determined exactly. An executable `Workaround.exe` is included as a temporary workaround for CaptureBitmap and Pause until Bitmap issues. The workaround can be enabled by calling Synchronize(..., 2)
1.18	Gijsbert Wiesenekker	Added `(ms)` to threshold alarm message. Threshold alarms use a different suppression key.
1.19	Gijsbert Wiesenekker	Added the 64-bit version of `nircmd` to kill both 32-bit and 64-bit instances of the same process on 64-bit Windows.
1.23	Gijsbert Wiesenekker	The number of steps has been increased to 100 and the number of nested steps to 8. The maximum number of timers is still limited to 8.
1.34		Removed the option to write QoS messages to a text file to be processed by the logmon probe. The problem was that the logmon probe had to check this file at 1 second intervals, so the SLA/SLO reports became useless as these started to report downtime for each 1 second interval. The ClientGrab utility now allows you to select areas near the edges of the screen and guides you through the steps. Changed the sixth argument of `Prepare` from 0, 1,2 and 3 to

Version	Author	Comments
		the easier to understand and remember `"INC"`, `"INCQOS"`, `"EXC"`, `"EXCQOS"`. Changed the second argument to `Synchronize` to the easier to understand and remember `"ANYWHERE"` or `"EXACT"`. Added `MyPause`, `MyUseWindow` and `MyUsePage` subroutines. These routines show what they are waiting for in a `MsgFrame`. Added `BitmapPrecision` as an argument to Synchronize and `MyClickOnBitmap` as it turns out this is a must-have for scripting Citrix in Application Mode. `Synchronize` will automatically do a root-cause analysis if it cannot find the bitmap: first it will try to lower the `BitmapPrecision`. If it still cannot find it will try to find it in the front window (if so the Window title might have changed). All error messages are now very descriptive and are written to the Alarm console *and* are shown in a `MsgBox` when developing the script. If a script fails the line-number at which it fails is also written to the Alarm console.
1.35		When searching for a bitmap the options were to search at the exact location or in the entire screen. When using a BitmapPrecision lower than 100 this caused the script to run VERY slow when searching for it in the entire screen. You now have the option to specify the size of the area to search for the bitmap. Changed the arguments to Prepare from `"INC"`, `"INCQOS"`, `"EXC"`, `"EXCQOS"` to `E2E_CONV_INC`, `E2E_CONV_INC_QOS`, `E2E_CONV_EXC`, `E2E_CONV_EXC_QOS` to make it backward compatible with versions 1.33 and before, and changed the arguments to `Synchronize` and `MyClickOnBitmap` from `"ANYWHERE"` or `"EXACT"` to `E2E_CONV_ANYWHERE` and `E2E_CONV_EXACT` to make them backward compatible with version 1.33 and before and to support the new scaling option. Unfortunately this means you have to update and recompile your existing 1.34 scripts. Added make.exe and a Makefile so that you can easily recompile all .rob files.
1.36		Added nimSetCI. Thresholds are also sent as a QoS so that you can show then in Performance Charts and List Views.
1.37		Not all QoS's were coupled with a CI and nested QoS's were not assigned the correct metric ID. This has been fixed. The metric ID was script-position dependent: if you added a step the metric ID would change. The metric ID is now derived from the name of the target. Fixed some documentation errors.

Version	Author	Comments
TODO		Document "DISAPPEAR" option of Synchronize. Document "WORKAROUND" option of Synchronize. Document the use of TurtoiseSVN for the scripts directory. Perhaps include CCLEANER.

Description
This probe-package provides a Nimrecorder SRC file and a couple of utilities that help turn a recorded Nimrecorder script into a script that can be run by the e2e_appmon probe. The resulting script:

- Creates bitmaps voor validation of steps with a naming convention.
- Generates QoS messages for steps with a naming convention. This allows easy creation of list views, performance charts and SLA/SLO's.
- Generates alarms for failed steps and auto-clears them if the script runs succesfully again.
- Generates a screendump when the script fails and sends QoS NULL messages for all open timers (for all subsequent steps) if it fails.

Installation
Deploy the probe-package to the robots running the e2e_appmon_dev probe. NEVER DEPLOY THE E2E_APPMON PROBE. The idea of the e2e_appmon probe is that you develop the script on one workstation (the one on which you deployed the e2e_appmon_dev probe) and then copy the working script to the other workstations. However, in my experience the script will run on most but not all workstations due to slight differences in (graphics) hardware and software (drivers). You will not be able to troubleshoot the script if you have installed just the e2e_appmon probe on those workstations.

Prerequisites
Install a VNC server on the workstation so that you can watch the script while it is being executed. DO NOT USE MICROSOFT REMOTE DESKTOP, AS YOU WILL NOT GET A CONSOLE SESSION AND IT WILL CHANGE THE RESOLUTION.
TODO: Document the use of (non-)personalized accounts that access the application(s); same account on each workstation means it must be allowed to logon to the workstation and/or to the application multiple times from different locations; password expiration;

Usage
Record a transaction using NimRecorder. The first challenge that you have is to try to execute the script after you have recorded it. 80% of the time it runs right away, but 20% of the time it fails for various reasons because:

- The recorder has missed a mouse- or keyboard click.
- It sends the mouse- or keyboard click to the wrong window.
- Sometimes you need to pause the script a couple of seconds before continuing (After logging on to Citrix you have typically have to wait a couple of seconds before clicking on the application otherwise the Citrix session hangs. My guess is that the ClickHTMLElement is sent 'too soon').
- Sometimes you may already need MyClickOnBitmap (see below).

This why I recommend to limit e2e_appmon scenario's to simple scenario's that 'probe' the main

application tiers, the typical example is ‘Goto website, Logon, Query some information, Logoff’. Our UMP illustrates some of the challenges that you might encounter. The scenario is: start IE, logon to UMP, click on Accounts and Contacts, logout. Record the transaction and make sure you do not press <Enter> after entering the password but click on the Login button and don’t click on Configuration, but hover the mouse over Configuration until the drop-down menu is shown and click on Accounts and Contacts. The resulting script looks like:

```
StartBrowser("IE", "http://172.16.5.125", 3)

UsePage("Home - UMP")
      WriteHTML("INPUT TEXT[NAME= '_58_login']", "administrator")
      WriteHTML("INPUT PASSWORD[NAME= '_58_password']", "P@ssW0rd")
      ClickHTMLElement("A[INNERTEXT= 'Accounts and Contact']")

UsePage("Accounts and Contacts - UMP")
      ClickHTMLElement("LI[OUTERTEXT= 'administrator   (Sign Out)']")
      ClickHTMLElement("A[INNERTEXT= 'Sign Out']")
```

StartBrowser, UsePage, UseWindow, ClickHTMLElement will automatically wait until the browser, page, window or HTMLElement is shown. If you want you can already replace UsePage and UseWindow with MyUsePage and MyUseWindow. These procedures show what they are waiting for in a MsgFrame.

When you run this script again you will find that nothing happens after entering the username/password and that it will time-out in ClickHTMLElement("A[INNERTEXT= 'Accounts and Contact']"). The reason is that the recorder failed to register the click on the Login button (note that newer version of UMP/e2e_appmon do not seem to have this issue any more). This happens VERY often with logon dialog buttons. The workaround is to send an <Enter> after the password:

```
WriteHTML("INPUT PASSWORD[NAME= '_58_password']", "P@ssW0rd<Enter>")
```

If you now run the script again it will logon to UMP, but it will fail in the:

```
ClickHTMLElement("A[INNERTEXT= 'Accounts and Contact']")
```

The reason is that the recorder failed to register the click on the Configuration menu because you did not click on it at all. As a best-practice do not hover/drag, but always click. Knowing that you can now record the script again:

```
StartBrowser("IE", "http://172.16.5.125", 3)

UsePage("Home - UMP")
      WriteHTML("INPUT TEXT[NAME= '_58_login']", "administrator")
      WriteHTML("INPUT PASSWORD[NAME= '_58_password']", "P@ssW0rd<Enter>")

UsePage("Home - UMP")
      ClickHTMLElement("INPUT SUBMIT[VALUE= 'Sign In']")

UsePage("Home - UMP")
      ClickHTMLElement("A[INNERTEXT= 'Configuration']")

UsePage("Configuration - UMP")
```

```
      ClickHTMLElement("A[INNERTEXT= 'Accounts and Contact']")

UsePage("Accounts and Contacts - UMP")
      ClickHTMLElement("A[INNERTEXT='Sign Out']")
```

If possible use keyboard shortcuts instead of mouse-clicks, but these are also not always recorded. If the script has recorded a sequence like:

```
UseWindow()
SendKeys("<Alt <A>> <Alt <B>> <Alt <C>>")
```

It works for most applications better to split this sequence into:

```
UseWindow()
SendKeys("<Alt <A>>")

UseWindow()
SendKeys("<Alt <B>>")

UseWindow()
SendKeys("<Alt <C>>")
```

My guess is that this gives the application more time to handle the keyboard shortcut.

Most applications do not record mouse-clicks at all on certain elements and do not provide a keyboard-shortcut. Take the CSM portal for example. If you click on Tickets it is not registered during recording, BUT THAT DOES NOT MEAN THAT YOU CANNOT CLICK ON IT. The workaround is to use `MyClickOnBitmap` (see below).

The second challenge that you have is to turn this script into a script that can be executed by the `e2e_appmon` probe. Most customers start to look for a button `Make e2e_appmon script` but it is not there: you have to manually add timers and send QoS's and alarms, but you will quickly run into the following issues:

- Timers cannot be nested. This is very counter-intuitive.
- The alarms that you send will not be auto-cleared.
- If you do not choose a naming convention for the QoS's the scripts become very hard to maintain, but even if you choose a naming convention it is not enforced so there is a risk you are not consistent.
- If the script fails in a step no QoS NULLs will be sent for all subsequent steps, but if a script consists of the steps login, query some information and a logoff, and it fails at login, you want QoS NULLs to be sent for the query and logon steps.
- You have to validate if steps have been executed succesfully, and you often have to do this using bitmaps. The standard way is to insert a `Synchronize on bitmap` using the wizard, but if you do not choose a naming convention for the bitmaps the scripts become very hard to maintain. Even if you choose a naming convention if it not enforced. The inserted code displays a MsgBox if it cannot find the bitmap, but this does not work if the script is run by the probe, as you have to click on `OK`.

The supplied e2e_conv-functions.src script addresses most of these issues, but not the missing `Make e2e_appmon script` button: you have to manually modify the freshly recorded script as follows:

Copy/paste the following lines at the beginning of the script:

```
'Copy
Dim AppNames$(100)
Dim StepNames$(100)
Dim StepStatus(100)
Dim ClearStatus(100)
Dim BitmapTimer(100)

include "Nimbus-functions.src"
include "e2e_conv-functions.src"

Prepare("Portal", "", "firefox.exe", "1024 768 32", 20, E2E_CONV_EXC)
'Paste
```

The subroutine Prepare() takes the following arguments:

- The first argument is the name of the source file of the script *without* the .src extension. Unfortunately Nimrecorder does not provide a function to return the name of the current SRC file, so this has to be entered manually. **Prepare uses this name to read the source file to collect all BeginStep/EndStep statements to generate bitmaps, generate alarms and QoS's, so if you enter the name here of another but existing script things will become very confusing!**
- The second argument is the full path to the e2e_scripting\scripts directory. `Prepare` tries to derive it from the environment variable `NIM_ROOT` and the Nimrecorder function `Curdir$` if the argument is equal to the empty string. If that fails you can specify the correct path here.
- The third argument is a comma separated list of applications to kill before the script starts. Make sure to kill also applications started by the main application like Acrobat Reader, Word or the Citrix client.
- The fourth argument is the display resolution (widht, height and color depth) with which to execute the script. This argument will be passed to `nircmd setdisplay` (see the supplied `NirCmd.Chm` file for all options).
- The fith argument is the number of seconds to wait before capturing a bitmap to validate a step (see below).
- The sixth arguments specifies how the time needed to search for bitmaps to validate steps should be handled. You can choose from the following four options:
  - `E2E_CONV_INC`: include it in the time needed for a step (this makes the timers backward compatible with versions 1.9 and before);
  - `E2E_CONV_INC_QOS`: include it in the time needed for a step AND send a QoS for it. The name of the QoS will be the name of the step postfixed with '-Bitmap'
  - `E2E_CONV_EXC`: exclude it from the time needed for a step. NOTE THAT THIS WILL CHANGE YOUR EXISTING (PRE-VERSION 1.10) BASELINES AND THRESHOLDS.
  - `E2E_CONV_EXC_QOS`: exclude it from the time needed for a step AND send a QoS. The name of the QoS will be the name of the step postfixed with '-Bitmap'

Now bracket each step of the script that you want to measure with calls to the subroutines `BeginStep()` and `EndStep()` as follows:

```
'Copy
BeginStep("Login")
'Paste
..
'Copy
EndStep("Login", 0, 0, 0)
'Paste
```

The subroutine `BeginStep()` has the name of the step as the first argument. The subroutine `EndStep()` should have the same name as the first argument (and `e2e_conv-functions.src` will check this). The name is not required as by design an `EndStep()` will always end the previous `BeginStep()`, but it makes it easier to maintain a long script if you can quickly identify the `EndStep`. The second argument is the MINOR threshold in milli-seconds, and the third the MAJOR threshold in milli-seconds. The fourth argument is reserved for future use. No alarms will be generated if these arguments are equal to 0.

Timers can be nested, so you are strongly advised to bracket the complete script with a `"Total"` step:

```
BeginStep("Total")
..
EndStep("Total", 0, 0, 0)
```

This `"Total"` QoS will follow the same naming convention as the other QoS's and a QoS NULL will be sent if the script fails.

Add a call to the subroutine `CleanUp()` at the end of the script.

```
'Copy
Cleanup()
'Paste
```

The script now generates QoS’s and/or alarms, but it does not validate steps yet. If you want to validate a step using bitmaps (other forms of validation like OCR are beyond the scope of this document) all you have to do is to add a call to `Synchronize` BEFORE an `EndStep`:

```
'Copy
Synchronize("", E2E_CONV_ANYWHERE, BitmapPrecision)
'Paste
```

or

```
'Copy
Synchronize("", E2E_CONV_EXACT, BitmapPrecision)
'Paste
```

or

```
'Copy
Synchronize("", Percentage, BitmapPrecision)
'Paste
```

The subroutine `Synchronize` will check if a config file with the name `<scriptname>-<stepname>.cfg` exists. If it exists it will read the Window name, instance number and the coordinates of the region that should be used for bitmap validation from the config file. If it does not exist it will launch the `ClientGrab` utility that allows you to select a Window and a region that should be used for validation. When you click `<F7>` ClientGrab will hide itself and give you five seconds to activate the Window of which you want to select a region. Be careful to click somewhere in the window that does not trigger events and make sure the mouse does not hover over something that causes a highlight and affects the bitmap. After five seconds it will return with a screenshot of that window and you can select a region for validation using `<F8>`. ClientGrab will generate the config file and the script will continue after you close the ClientGrab utility. The subroutine Synchronize will then check if a bitmap file with the name `<scriptname>-<stepname>.bmp` exists. If not it will wait `Stabilize` seconds before taking a screenshot of the region you want to use for validation, store it in a bitmap file `<scriptname>-<stepname>.bmp` and continue. If a bitmap file with the name `<scriptname>-<stepname>.bmp` exists it will try to find the bitmap in the Window. Where it finds it depends on the second argument to Synchronize: if equal to `E2E_CONV_ANYWHERE` it will try to find the bitmap anywhere in the Window, if equal to `E2E_CONV_EXACT` it will try to find the bitmap at the same location it was captured at and if equal to a number greater than or equal to `100` it will enlarge the area to find the bitmap by that percentage, so 100 is the same as `E2E_CONV_EXACT`. The `BitmapPrecision` specifies how the bitmap should match: 100 means an exact match, lower than 100 means a ‘fuzzy’ match and this is often needed in Citrix sessions. However, a value lower than 100 makes searching for the bitmap using `E2E_CONV_ANYWHERE` VERY slow, so you are strongly advised to use either `E2E_CONV_EXACT` or a value slightly larger than 100 for `Percentage` in that case. Note that it is not documented what ‘fuzzy’ means: depending on the bitmap a value of 99 can already cause the ‘wrong’ area to be selected, and the bitmap seems always to be ‘found’ if the value is below 90, so if 100 does not work it requires some trial and error to find a value that ‘works’. I have filed an enhancement request to return the coordinates of the area where it ‘found’ the bitmap, so that you can decide if it is ‘too far off’.

There are two use-cases for these config and bitmap files:

- When you run the same script on another workstation sometimes some or all of the bitmaps are slightly different due to (slight) differences in graphics hardware and drivers. You can just delete some or all `.bmp` files and the script will automatically regenerate them. Note that you have to watch the execution of the script to make sure that the value of `Stabilize` is ‘large enough’ for the content of the windows to stabilize, otherwise wrong bitmaps will be generated.
- If the application has changed and some or all of the bitmaps are different but are still at the same location you can just delete some or all `.bmp` files and the script will automatically regenerate the bitmap files. Note that you have to watch the execution of the script to make sure that the value of `Stabilize` is ‘large enough’ for the content of the windows to stabilize, otherwise wrong bitmaps will be generated.
- If the application has changed and some or all of the bitmaps are different and are not at the same location you can just delete some or all `.cfg` files and the script will automatically launch `ClientGrab` to regenerate the `.cfg` and the `.bmp` files.

The supplied `Portal.src` provides an example of a freshly recorded script edited in the above way. If you want you can replace `UsePage` with `MyUsePage` and `UseWindow` with `MyUseWindow`. These

routines will show that they are waiting for a page or window in a MsgFrame at the top of the display.

`Synchronize` also has another important application. When you record Win32 applications (Microsoft Dynamics is a notorious example) or terminal sessions within browsers you will often find that `Click on the Start button`, `Click on Word` etc. are captured as sequences of pairs of `ClickMouse` statements:

```
ClickMouse(Right,Down,34,753)
ClickMouse(Right,Up,34,753)
```

The problem is that it is not clear from these statements where is clicked, it is error prone (if the button is located at a different position `ClickMouse` will no longer work) and it makes it very hard to maintain the script.

You can replace a pair of `ClickMouse` statements by a call to `Synchronize`, followed by `MyClickOnBitmap` as follows:

```
'Copy
Synchronize("StartButton", E2E_CONV_ANYWHERE | E2E_CONV_EXACT | Percentage,
BitmapPrecision)
MyClickOnBitmap("LS", E2E_CONV_ANYWHERE | E2E_CONV_EXACT | Percentage,
BitmapPrecision)
'Paste
```

The first argument is the name of the substep. `Synchronize` will launch `ClientGrab` and generate a config file with the name `<scriptname>-<stepname>-<substep>.cfg` and a bitmap file with the name `<scriptname>-<stepname>-<substep>.bmp` as before. `MyClickOnBitmap` will search for and click on that bitmap where `"LS"` means a single-click on the left mouse-button, `"LD"` means a double-click on the left mouse-button, `"RS"` means a single-click on the right mouse-button and `"RD"` means a double-click on the right mouse-button. This makes the script more robust against slight changes of the content within the window (if the button is located at a different position `ClickMouse` will no longer work but `MyClickOnBitmap` should still be able to find it) and makes the script much easier to maintain because the first argument to `Synchronize` shows where is clicked.

When you schedule the script in the `e2e_appmon` probe make sure the `e2e_appmon` profile name is the same as the script name (e2e_conv-functions.src will check this). The reason is that the profile name is used to generate the names of the QoS's, and it quickly becomes VERY confusing if you use a different name for the profile.

Miscellaneous remarks
Elements in web-pages often change colour the second time you access them. If these elements are used for bitmap validation and bitmap clicks the colour changes back if the browser is reset and the script will no longer work. You have to manually click through the script to change the colour again. I have filed an enhancement request to have an option to change the bitmaps to black/white.