Automatic Test System

Jo Simons (simonjo@telenet.be)

Introduction

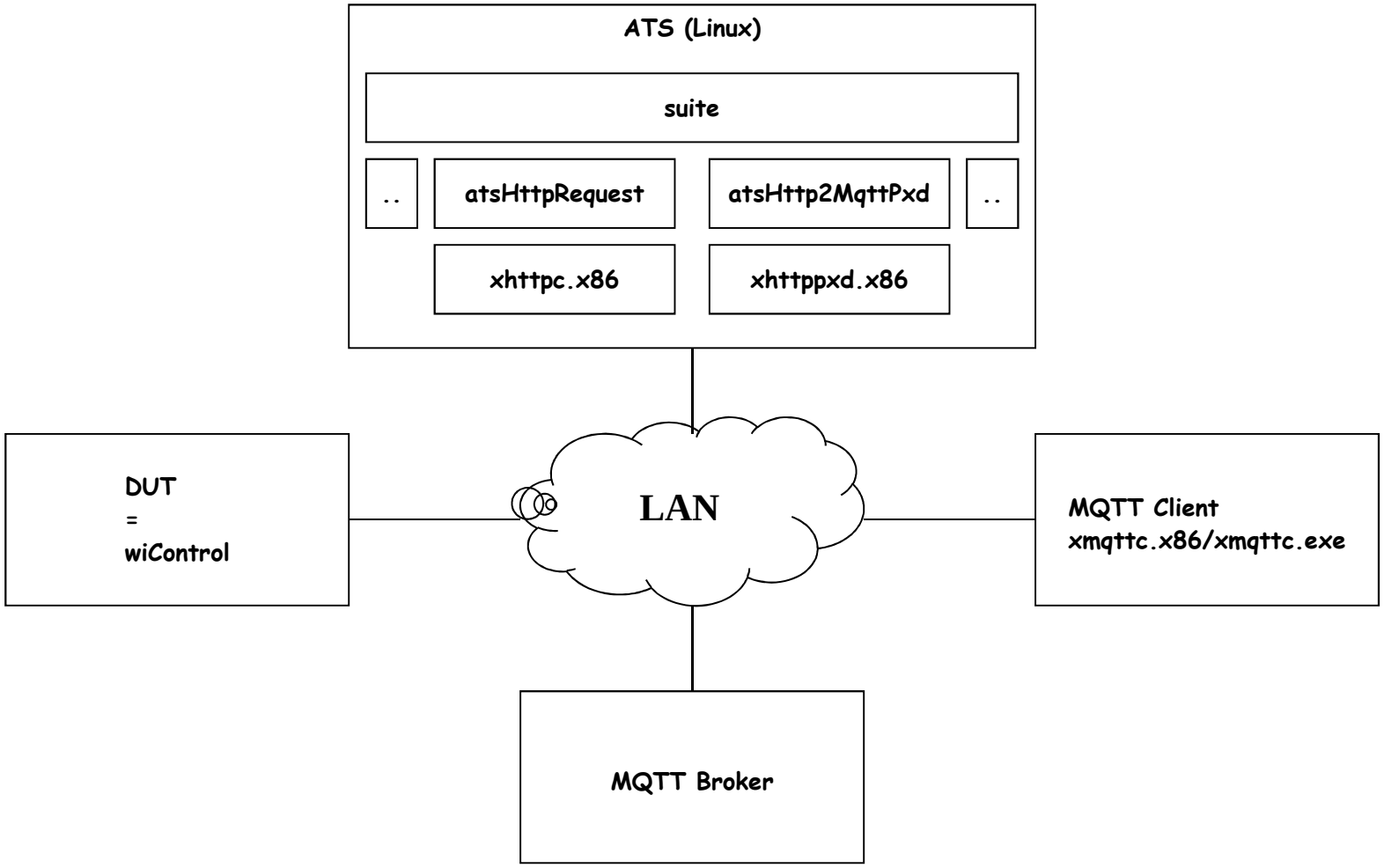
- Linux based test environment, configurable via ini-file
- Fixed directory structure
  - atsLinux
    - /bin core scripts and executables
    - /ini configuration file
    - /log logging directory
    - /suites test suite scripts
  - requires the root directory to be added to the PATH variable
- Core implemented as a set of BASH scripts and some executables
- Ini-file has a flat argument=value style, sections are optional but not used
  - structured argument names -> i.e. webHttpHost_<id>=192.168.0.60
- Test suite scripts
  - use core scripts
  - group functional tests, i.e. for certain module type
  - are datadriven from ini-file, i.e. channel number, ip-address of system to test
  - send detailed logging to stdout, overview logging to stderr
  - each test as a separate function
  - recover function to revert to a known state

Sripts & Executables

- Most important items
  - atsGetEnvFile — returns the name of the ats.ini file
  - atsGetRoot — returns the installation root of ATS
  - atsGetValue — returns a value from ats.ini
  - atsHttpRequest — request an HTTP URL
  - atsHttp2MqttPxd — start/stop HTTP-2-MQTT proxy daemon
  - atsPersistentCounter — manipulates persistent counters
  - atsPrintBanner — generates a banner to stdout
  - atsPrintError — prints error code in decimal & text format
  - atsRunScript — runs a script and captures/filters it's logging
  - atsSleep — pauses execution, encapsulated in stdout logging
  - xhttp.x86 — HTTP client executable
  - xhttppxd.x86 — HTTP proxy daemon

- Usage — scriptname [-option …] [argument …]
  - Where option can be one or more of following
    - -v — return version info
    - -b — return brief description
    - -h — return help pages
    - … — depending on script
  - argument — zero or more arguments depending on script

Test Setup



Suite Script Structure

main

```
nErrCode=0
nTcases=0
nFailed=0

while true do;
  # determine HTTP target and iterations
  nHttpId=$( atsGetValue suiteWebEmul )
  strIter=$( atsGetValue suiteIter_8 )

  for ((i=1; i<=${strIter}; i++)); do
    # test inputs
    nChanIn=0
    printf "\n` date +\"%Y-%b-%d %H:%M:%S\"`
        suiteWiControl_WBS (in${nChanIn})(${i} of ${strIter})\n" >&2
    tcWiControl_IN_02_Ingt0

    # test outputs
    nChanOut=0
    printf "\n` date +\"%Y-%b-%d %H:%M:%S\"`
        suiteWiControl_WBS (out${nChanOut})(${i} of ${strIter})\n" >&2
    tcWiControl_OUT_02_On
    done
  break
  done

# report statistics

exit nErrCode
```

function tcWiControl_OUT_Recover {

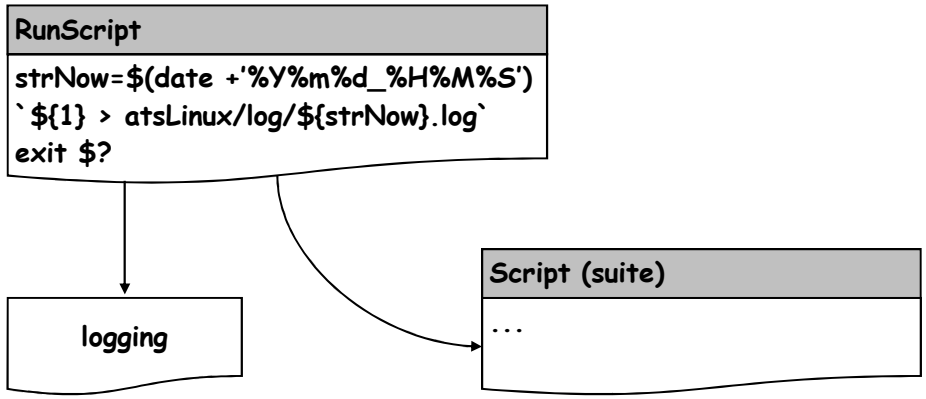
```
while true; do
  atsHttpRequest -q ${nWebId} "ats?ccmd=
    out${nChanOut}.unlock;out${nChanOut}.timeabort;
    out${nChanOut}.off"

  nTcases=$((nTcases+1))
  return
  done

}
```

function tcWiControl_OUT_02_On {

```
while true; do
  _LogTcase tcWiControl_OUT_02_On

  tcWiControl_OUT_Recover

  atsHttpRequest -q ${nWebId} "ats?ccmd=
    out${nChanOut}.on" -eq 1
  if [ $? -ne 0 ]; then break; fi

  _LogSuccess tcWiControl_OUT_02_On
  return
  done

  _LogFailed tcWiControl_OUT_02_On
}
```

atsRunscript

- atsRunscript executes another script while capturing it's logging
- the other script can generate logging to
  - stdout (> &1) for detailed logging
  - stderr (> &2) for overview logging
- atsRunscript will split/filter both streams as follows
  - stdout + stderr are sent to a timestamped logfile in /log/<script-name>.yyyymmdd_hhmmss.log
  - stderr is also sent to the console
- this way you will see the essential info on the console, and details in the logfile



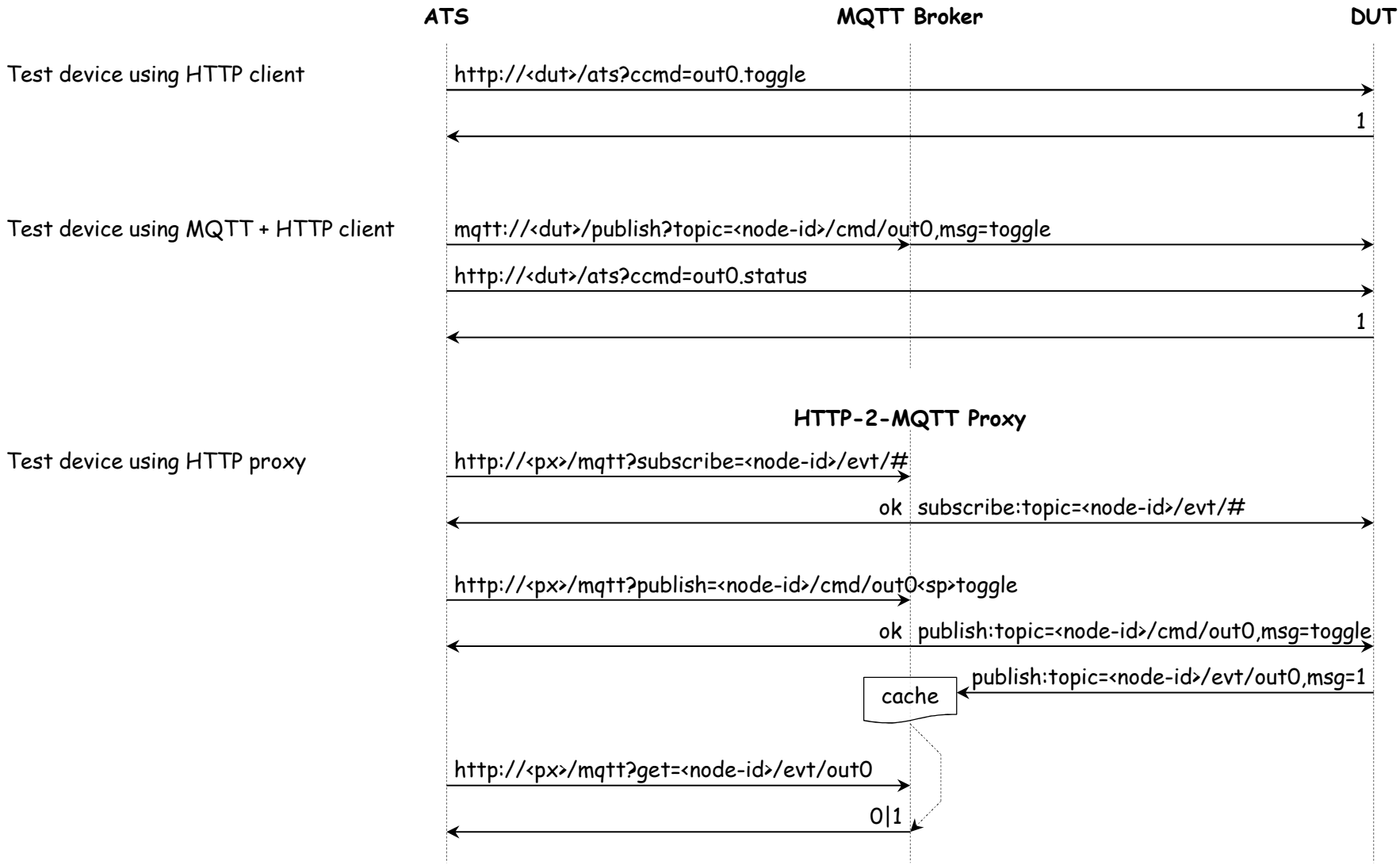
Sample ats.ini

- [web]
- # web emulation configuration data, used by atsWebEmul and atsHttpCommand
- webEmulName_4=Wemos (wiControl)
- webHttpAuth_4=0
- webHttpUser_4=admin
- webHttpPswd_4=admin
- webHttpHost_4=192.168.0.157
- webHttpPort_4=80

- # suite module configuration options
- # suiteName_% : suite name
- # suiteIter_% : suite number of iterations
- # suiteSubs_%_<sub-suite>: parameter for sub-suite, 0=disable, 1=enable
- # suiteWiControl
- suiteName_8=suiteWiControl
- suiteIter_8=1
- suiteSubs_8_WBS=1
- suiteSubs_8_WBR=0

- [mqtt]
- mqttName_0=Broker on Synology ASTR76N0
- mqttHost_0=192.168.0.10
- mqttAuth_0=0
- mqttUser_0=
- mqttPswd_0=
- mqttClient_0=astr76n0-1

Test Scenarios



HTTP Proxy Daemon (1/9)

- Extendible proxy that translates HTTP requests into another protocol, i.e. MQTT
- Allows multiple commands per request to simulate request-response model for protocols that are req-only

- Structured command URL syntax
  - http://<host> [:<port>] / <file> ? <command> *[& [!]<command>]
  - <file> <proto>.html
    - will be sent in response
    - <proto> will be used to select proper handling routines
  - <command> [!] <cmd>=<parm> *[<sp><parm>]
    - [!] suppress command results from terse and verbose response when specified
    - subscribe=<topic> Sends an MQTT subscribe msg to a remote broker
    - unsubscribe=<topic> Sends an MQTT unsubscribe msg to a remote broker
    - publish=<topic><sp><msg> Sends an MQTT publish msg to a remote broker
    - get=<topic> Returns the cached value of last publish msg received for <topic>
    - count=<topic> Returns the cached count of publish msgs received for <topic>
    - clear=<topic> | '*' Clears the internal cache for <topic> or all topics
    - wait=<ms> Pauzes execution for given milliseconds
    - terse Selects terse result to be sent as response (*)
    - verbose Selects verbose result to be sent as reponse (*)
  - (*) commands do not generate terse and verbose result output

HTTP Proxy Daemon (2/9)

- Request handling
  - Specified commands are executed from left to right, during which a terse and verbose result is built up
  - An exclamation mark ‘!’ in front of a command, suppresses it’s results from terse and verbose results
  - The terse and verbose commands override each other, only the last is taken into account

- Response generation
  - If the terse command is specified then the terse result is returned as-is
  - Else if the verbose command is specified then the verbose result is returned as-is
  - Else if <file> is specified then this is returned
  - Else default.html is returned
  - Tags embedded in a response file will be replaced on the fly

- Verbose result
  - Syntax: <cmd> ‘:’ (‘err’ <error> | ‘ok’) <CRLF>

- Terse result
  - Syntax (‘-’ <error> | <result>) *[‘;’ (‘-’ <error> | <result>)]

HTTP Proxy Daemon (3/9)

- Command line syntax
  - xh2mxd.x86 *[<option>]
  - Options are

-?	Display help info
--loglevel <level>	Loglevel, decimal (123) or hexadecimal number (0x123), default=0
--ini <file>	Use ini-file with settings
--locaddr <addr>	Local ip-address on which HTTP server is listening, default=0.0.0.0
--locport <port>	Local ip-port on which HTTP server is listening, default=8080
--locpath <path>	Local directory where served files are located
--remaddr <addr>	Ip-address of remote MQTT broker, default=0.0.0.0
--remport <port>	Ip-port of remote MQTT broker, default=1883
--remuser <user>	Username to use in connection to remote MQTT broker
--rempswd <pswd>	Password to use in connection to remote MQTT broker
--remclient <clientid>	Unique clientid to use for connection to MQTT broker, default=<hostname>
--remalive <seconds>	Keepalive time for connection to MQTT broker, default=60

  - Sample usage: ./xh2mpxd.x86 –remaddr 192.168.0.10 –clientid blabla123

HTTP Proxy Daemon (4/9)

- The ini-file specified with –ini <file> is a plain text file with following syntax
  - [main]
    - loglevel=<number>
  - [httpsd]
    - locaddr=<addr>
    - locport=<port>
    - locpath=<path>
  - [mqttcd]
    - remaddr=<addr>
    - remport=<port>
    - remuser=<string>
    - rempswd=<string>
    - remclient=<string>
    - remalive=<seconds>

HTTP Proxy Daemon (5/9)

- Response files can contain tags that will be replaced on the fly
  - tag syntax ‘<?’ <branch> <node> *[<predef>=<value>] ‘?>’
  - branch ccount | cvalue | cookie | httpsd | include | mqttcd | proxyd | request | response

- Ccount branch Return cache count for given topic
  - syntax ‘<?ccount’ <topic> ‘?>’
  - sample <?ccount “astr76b32/3/0/evt/out0”?>

- Cvalue branch Return cache value for given topic
  - syntax ‘<?cvalue’ <topic> [‘empty=‘ <value>] ‘?>’
  - sample <?cvalue “astr76b32/3/0/evt/out0” empty=“0”?>

- Cookie branch Return request cookie
  - syntax ‘<?cookie’ <cookie-name> [‘empty=‘ <value>] ‘?>’
  - sample <?cookie “myCookie” empty=“nope”?>

- Include branch Include file into response
  - syntax ‘<?include <file> ‘?>’
  - sample <?include “myfile.html”?>

HTTP Proxy Daemon (6/9)

- Httpsd branch
  - connections — Returns current number of HTTP server connections
  - hits — Returns current number of HTTP server hits
  - locaddr — Returns HTTP server local address
  - locport — Returns HTTP server local port
  - locpath — Returns HTTP server local path
  - rxbytes — Returns total number of bytes received
  - txbytes — Returns total number of bytes sent
  - version — Returns HTTP server version

HTTP Proxy Daemon: MQTT support (7/9)

- Incoming MQTT publish messages from a remote MQTT broker are:
  - Stored in an internal value cache: [topic]=msg
  - Counted in an internal count cache: [topic]=<counter>
- Mqttcd branch
  - drops — Returns the number of MQTT broker connection drops
  - remaddr — Returns MQTT broker address
  - remalive — Returns MQTT keepalive time
  - remclient — Returns MQTT clientid
  - remport — Returns MQTT broker port
  - rempswd — Returns MQTT broker password
  - remuser — Returns MQTT broker username
  - rxbytes — Returns total number of bytes received
  - txbytes — Returns total number of bytes sent
  - rxmsgs — Returns total number of messages received
  - txmsgs — Returns total number of messages sent
  - version — Returns MQTT client version

HTTP Proxy Daemon (8/9)

- Proxyd branch nodes
  - alias — Returns proxy alias, ‘xhttppxd’
  - author — Returns proxy author
  - cache — Report current cache content, format: <topic>=<count>,<value><CRLF>
  - created — Returns date of proxy compilation
  - date — Returns current date in format ‘’
  - description — Returns proxy description
  - hostarch — Returns host architecture
  - hostname — Returns hostname, i.e. ‘ASTR76W0’
  - name — Returns proxy name, ‘xHttpPxd’
  - time — Returns current time in format ‘’
  - uptime — Returns uptime in format ‘’
  - version — Returns proxyd version

HTTP Proxy Daemon (9/9)

- Request branch nodes
  - path
  - peeraddr
  - query
  - terse
  - verbose

- Response branch nodes
  - terse
  - verbose