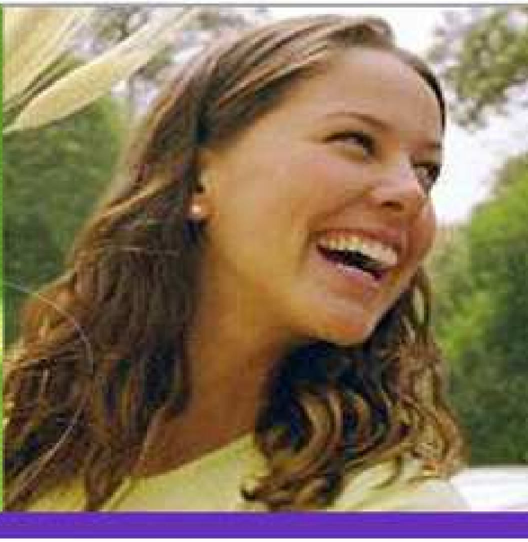
# Automatic Test System



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

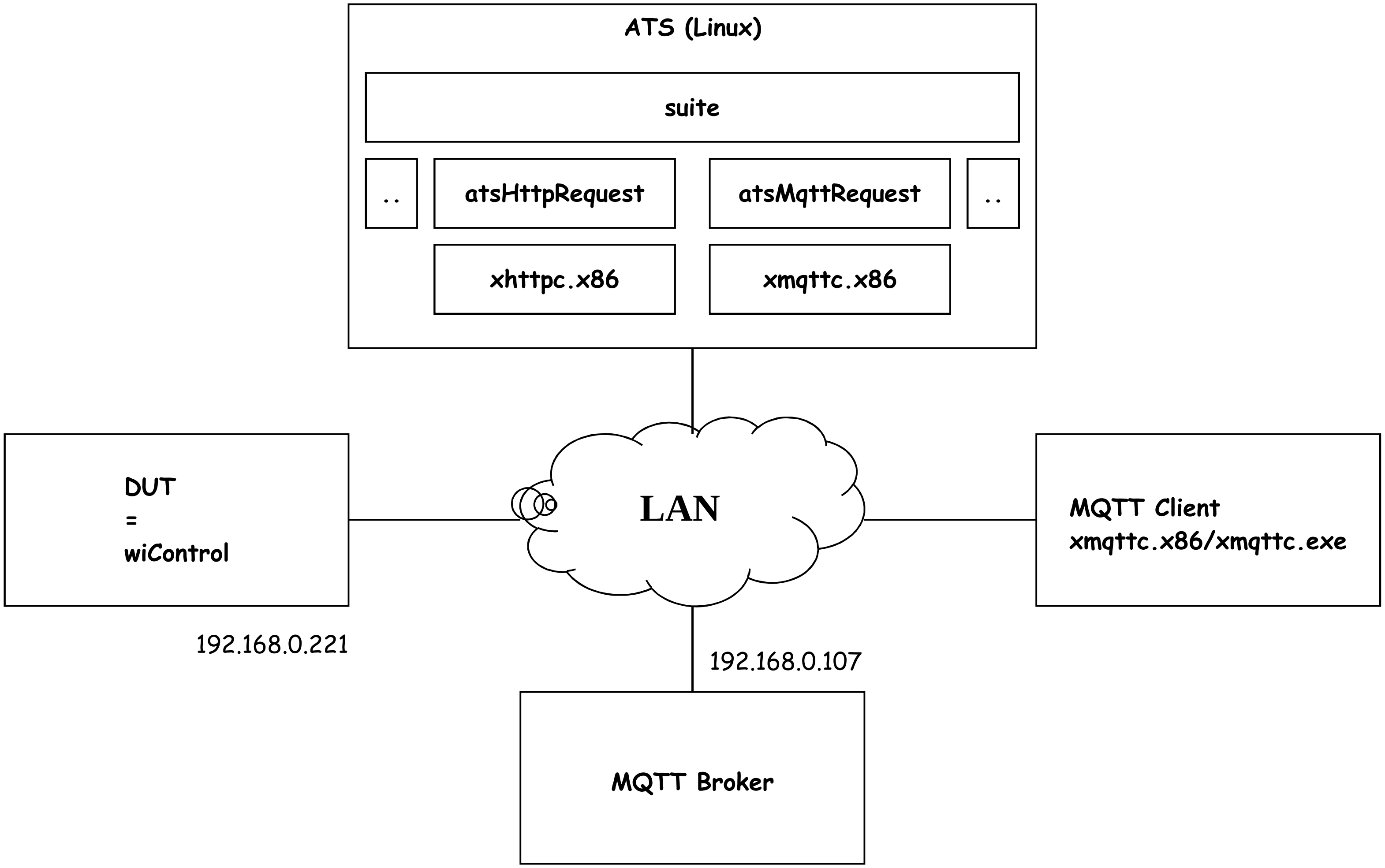
| | |
|---|---|
| atsGetEnvFile | returns the name of the ats.ini file |
| atsGetRoot | returns the installation root of ATS |
| atsGetValue | returns a value from ats.ini |
| atsHttpRequest | request an HTTP URL |
| atsMqttRequest | issue an MQTT publish message to a broker |
| atsPersistentCounter | 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 |
| xmqttc.x86 | MQTT client executable |

- 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


ATS (Linux)
suite
..
atsHttpRequest
atsMqttRequest
..
xhttpc.x86
xmqttc.x86
DUT
=
wiControl
LAN
MQTT Client
xmqttc.x86/xmqttc.exe
192.168.0.221
192.168.0.107
MQTT Broker

# 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
}
```

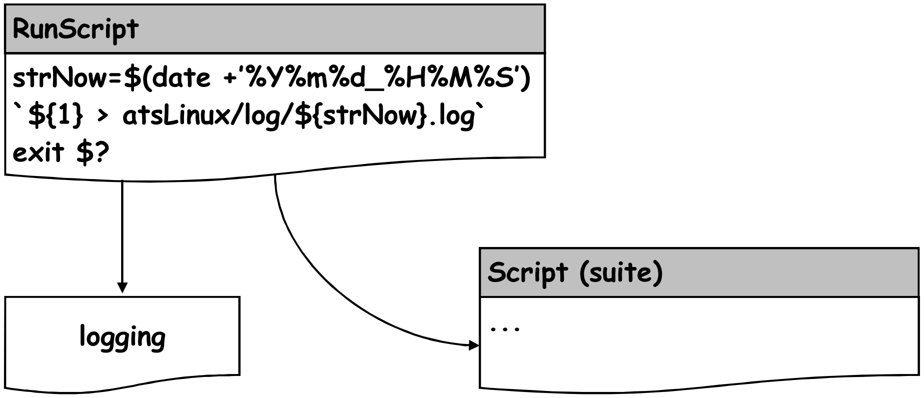
# atsRunscript

- 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



```
strNow=$(date +'%Y%m%d_%H%M%S')
`${1} > atsLinux/log/${strNow}.log`
exit $?
```

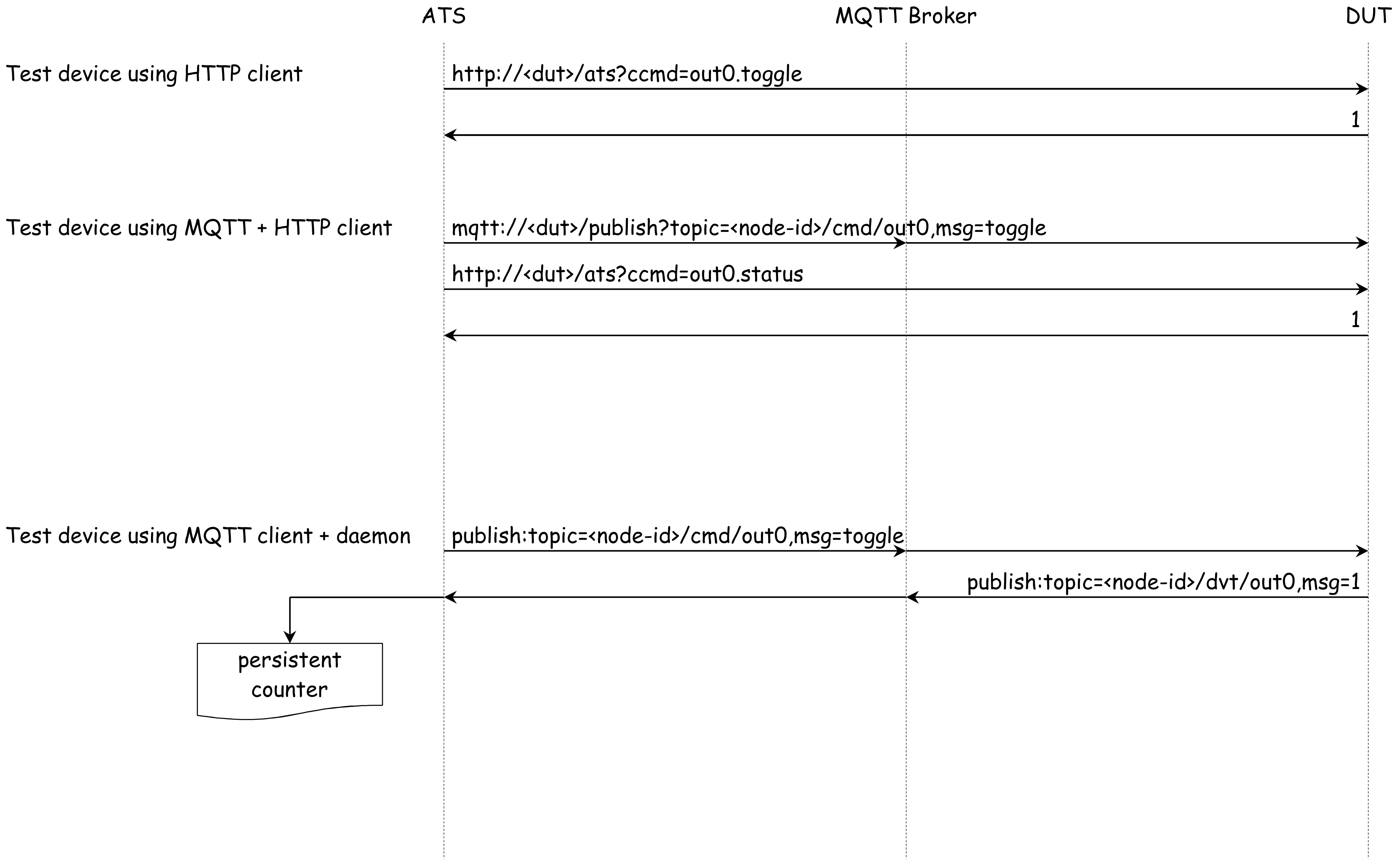
# Sample ats.ini

```
[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



ATS

MQTT Broker

DUT

Test device using HTTP client

http://<dut>/ats?ccmd=out0.toggle

1

Test device using MQTT + HTTP client

mqtt://<dut>/publish?topic=<node-id>/cmd/out0,msg=toggle

http://<dut>/ats?ccmd=out0.status

1

Test device using MQTT client + daemon

publish:topic=<node-id>/cmd/out0,msg=toggle

publish:topic=<node-id>/dvt/out0,msg=1

persistent counter