Publication history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Amendment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 Draft</td>
<td>14 Feb 2005</td>
<td>Brett Carnes</td>
<td>first draft</td>
</tr>
</tbody>
</table>

Template information

Name: Component Workbook Template
Reference: 1083
Version: 1.9 Released
Date: January 2004
# Chapter 1 Introduction

1.1 Purpose of document 2
1.2 Context 2
1.3 Acronyms 2

# Chapter 2 Role and responsibilities

2.1 Role 5
2.2 Responsibilities 5
2.3 Exclusions and limitations 6
2.4 Assumptions 6

# Chapter 3 Key scenarios

3.1 Actors 7
  3.1.1 ALE 7
  3.1.2 LIS 7
  3.1.3 IP Device 7
3.2 Scenario descriptions 7
  3.2.1 Start Up 7
  3.2.2 IP Device enters Network 7
  3.2.3 New MAC Notification 8
  3.2.4 Deleted MAC Notification 8
  3.2.5 Delete MAC Request 9
  3.2.6 Query MAC 10

# Chapter 4 Properties

4.1 Performance 11
4.2 Significant date or event readiness 11

# Chapter 5 Design overview

5.1 Design notes 13
  5.1.1 Guidelines 13
  5.1.2 Issues 13
5.2 ALE overview 13
5.3 ALE Taps 15
5.4 ALE Classes 16
  5.4.1 New MAC Notification 17
  5.4.2 Deleted MAC Notification 17
  5.4.3 Delete MAC Request 18
  5.4.4 Query MAC 19
5.5 LIS Communication 19
  5.5.1 Message Buffering 19

# Chapter 6 Interfaces

6.1 User support 21
  6.1.1 Environment 21
  6.1.2 Operation 21
  6.1.3 Debug 21
  6.1.4 Configuration and datafill 22

# Chapter 7 Implementation

7.1 Ale table implementation 25

# Chapter 8 Testing
8.1 Test approach 27
8.2 Test harnesses design 27
8.3 Test procedures 27
8.4 Test coverage 28
  8.4.1 Timer 28
  8.4.2 AleTable 28
  8.4.3 Tap 28
  8.4.4 MessageGenerator 28
  8.4.5 LisCommunicator 29
  8.4.6 Config 29
Chapter 1  Introduction

1.1 Purpose of document

This document contains the specification and design of the Ale component.

The specification and design of the Ale presented in this document outline all of
the components used in the implementation of the Ale and their dependencies.
These components are illustrated by way of various UML diagrams.

The underlying protocol used by the Ale to communicate with the LIS is also
explained.

1.2 Context

The ALE provides a mechanism to notify the LIS of new, updated and deleted MAC
addresses.

The ALE does this by providing functionality to monitor all outgoing ethernet packets
of various IP devices. It also keeps an updated listing of all valid MAC addresses and
incorporates a timer mechanism to ensure that all MAC address entries are valid.

Communication between the ALE and the LIS is implemented in the underlying ALE-
LIS protocol.

1.3 Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALE</td>
<td>Access Location Element</td>
</tr>
<tr>
<td>LIS</td>
<td>Listing Information Service</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>UML</td>
<td>Universal Modelling Language</td>
</tr>
<tr>
<td>VOIP</td>
<td>Voice Over Internet Protocol</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Control Protocol</td>
</tr>
<tr>
<td>STDERR</td>
<td>Standard Error</td>
</tr>
<tr>
<td>STL</td>
<td>Standard Template Library</td>
</tr>
</tbody>
</table>

Printed: 14/2/05
Reference: XXXX
Version: 1.0 Draft
Date: 05 Feb 2005
Chapter 2 Role and responsibilities

2.1 Role

The ALE monitors ethernet frames of various IP devices to determine whether a new MAC address is found or if a MAC address in the internal data structure is expired.

The ALE provides notification to the LIS of any new, updated or deleted MAC addresses and responds to MAC address queries by the LIS.

2.2 Responsibilities

The ALE component has the following responsibilities:

Example:

- Captures all outgoing IPV4 packets.
- Perform select on initialised taps.
- Maintains a table of all MAC address entries, all of which are referenced by a key which is a MAC address.
- Extracts appropriate source and destination addresses of captured packets. Extracted addresses include:
  - Source MAC address
  - Source IP address
  - Destination IP address
- Incorporates timer mechanism to determine if MAC address entry in internal data structure has expired.
- Responds to MAC queries initiated by the LIS
- Send notification to LIS of new MAC addresses found.
- Send notification to LIS of deleted MAC address entries. These notifications are sent when the ALE timer has deemed a MAC address to be expired.
- Implements a message generator and LIS communicator to facilitate the transfer of messages to the LIS.
- Incorporates SHA-1 hash function to ensure reliable delivery of messages.
2.3 Exclusions and limitations

ALE is not IPv6 compliant.

2.4 Assumptions

It is assumed that all WIFI access point will be plugged directly into the ALE.
Chapter 3  Key scenarios

3.1  Actors

3.1.1  ALE

This component.

3.1.2  LIS

The LIS is notified by the ALE when new MAC addresses are found or if MAC addresses have expired. Also queries ALE to obtain information about specific MAC addresses.

3.1.3  IP Device

The IP device is the device connected to and who’s traffic is monitored by the ALE.

3.2  Scenario descriptions

3.2.1  Start Up

Preconditions

None.

Description

• ALE loads the config file.
• Initialises taps with device names loaded from config file.
• Initialises timers time out value loaded from config file.
• Commences capturing packets from tap.

Alternative Outcomes

• If no config file exists the ALE will subsequently terminate.
• If any of the device names retrieved from the config file are invalid and tap is unable to initialise, the ALE will terminate with appropriate error messages notifying which device name is invalid.

3.2.2  IP Device enters Network.

Preconditions

Scenario 3.2.1 completed successfully.

Description

• IP Device broadcasts DHCP packet upon entry to the network.
• ALE captures outgoing DHCP message and extracts source IP address.
• ALE ignores packet with source address of ::0.

3.2.3 New MAC Notification

Preconditions
Scenario 3.2.1 completed successfully.
Scenario 3.2.2 completed successfully.

Description
• ALE captures packet from network and extracts MAC address, source IP address and destination IP address.
• ALE checks MAC address table to see if MAC address exists.
• ALE Message Generator class assembles MAC notification message.
• AES hash generated by ALE.
• ALE’s LIS communicator component adds the new MAC entry to its buffer.
• ALE’s LIS communicator component transmits notification to the LIS.
• LIS acknowledges MAC notification message.

Alternative Outcomes
• If the ALE does not recieve an ACK from the LIS it will continue to send the MAC notification at 10 second intervals (as per the ACK_RETRANS value loaded from the config file) for a pre-determined number of times (determined by the RETRANS_MAX value loaded from the config file).
• If the LIS communicators buffer is not full the UDP packet will not be sent to the LIS until more messages are placed into the LIS communicators buffer or every 100 cycles of the main program loop. Thus ensuring better efficiency by placing multiple messages into a single UDP packet.

3.2.4 Deleted MAC Notification

Preconditions
Scenario 3.2.1 completed successfully.
Scenario 3.2.2 completed with MAC address stored in ALE’s MAC address table.

Description
• ALE captures packet from network and extracts MAC address, source IP address and destination IP address.
• Timer incremented after 60 seconds.
• Timer class increments timer for each entry in MAC address table.
• Timer class checks to see if each entry has expired. eg: if expiry time >=
20.

- Timer class dynamically stores each expired entry.
- Timer class deletes expired entry from ALE’s Mac address table.
- ALE Message Generator class assembles Deleted MAC notification message for each expired entry.
- AES hash generated by ALE.
- ALE’s LIS communicator component adds the new MAC entry to its buffer.
- ALE’s LIS communicator component transmits notification to the LIS.
- LIS acknowledges Deleted MAC notification message.

**Alternative Outcomes**

- If the ALE does not receive an ACK from the LIS it will continue to send the MAC notification at 10 second intervals (as per the ACK_RETRANS value loaded from the config file) for a pre-determined number of times (determined by the RETRANS_MAX value loaded from the config file).
- If the LIS communicators buffer is not full the UDP packet will not be sent to the LIS until more messages are placed into the LIS communicators buffer or every 100 cycles of the main program loop. Thus ensuring better efficiency by placing multiple messages into a single UDP packet.

### 3.2.5 Delete MAC Request

**Preconditions**

Scenario 3.2.1 completed successfully.

Scenario 3.2.2 completed with MAC address stored in ALE’s MAC address table

LIS has not already received a Deleted MAC notification.

LIS has determined that a MAC address has moved to another port.

**Description**

- ALE receives Delete MAC Request from LIS.
- ALE determines if message sender is authentic using AES hash.
- ALE gets entry from MAC address table.
- ALE compares device name and port number of the MAC address entry to that of the device name and port number found in the Delete MAC Request message.
- ALE deletes its entry for that particular MAC address.
- ALE then sends ACK to the LIS.

**Alternative Outcomes**

- If the ALE determines that the SHA-1 hash is invalid the message is
ignored.

• If the device name and port number found in the Delete MAC Request message is the same as the device name and port number found in the ALE’s MAC address table an ACK is sent to the LIS and no further action is taken by the ALE.

### 3.2.6 Query MAC

**Preconditions**

Scenario 3.2.1 completed successfully.

Scenario 3.2.2 completed with MAC address stored in ALE’s MAC address table

**Description**

• ALE receives Query MAC Request from LIS.
• ALE determines if message sender is authentic using AES hash.
• ALE retrieves the entry for that particular MAC address from it’s MAC address table.
• ALE compares IP address in Query MAC Request message to that of the IP address found for that particular entry in the MAC address table.
• ALE then sends ACK to the LIS.
• ALE’s LIS communicator component adds the new MAC entry to its buffer.
• ALE sends message to LIS containing information about the MAC address queried.

**Alternative Outcomes**

• If the ALE determines that the AES hash is invalid the message is ignored.
• If the MAC and IP addresses do not match the ALE will send an ACK followed by 2 messages. For example: If the LIS Queries the MAC address 01-02-03-04-05-06 with an IP address of 1.2.3.4 and the ALE has an entry of 11-22-33-44-55-66 against 1.2.3.4, it should provide two responses. The first containing 01-02-03-04-05-06 with 0.0.0.0 and the second containing 11-22-33-44-55-66 with 1.2.3.4.
• If the LIS communicator’s buffer is not full the UDP packet will not be sent to the LIS until more messages are placed into the LIS communicator’s buffer. Thus ensuring better efficiency by placing multiple messages into a single UDP packet.
Chapter 4 Properties

4.1 Performance

Due to the ALE being implemented on a single thread each packet must be analysed and relevant data stored in an efficient manner.

4.2 Significant date or event readiness

The ALE does not use dates. The timer component of the ALE is only concerned with a signal sent from the kernel indicating that a pre-determined number of seconds has elapsed.
Chapter 5  Design overview

This chapter provides an overview of the <Component Name> high-level design. All class diagrams in this chapter utilize the UML notation.

5.1 Design notes

5.1.1 Guidelines

The Ale component should:

• utilise standard C libraries where possible
• conform to performance issues stated in section 4.2.
• undergo sufficient unit testing throughout development using cxxunit.

5.1.2 Issues

Some performance issues may arise whilst undertaking final testing. A multi threaded design approach should be considered if this is the case.

5.2 ALE overview

The following diagram gives a high level overview of the ALE architecture. It shows the position of the ALE relative to the various IP devices that could be connected to the network monitored by the ALE.
Figure 1 ALE configuration.

- Ethernet Input port 1
- Ethernet Input port 2
- Ethernet Input port 3
- WiFi AP
- SWITCH
- LIS
- Monitor Port and Cache MAC addresses
- Ethernet Address (6 bytes) Count (1 byte)
- 100 Address @ 7 bytes per address is 700Bytes per buffer
- Ethernet Address (6 bytes) Count (1 byte)
- Port (1 byte) MAC Address (6 bytes)
- New MAC Queue
- Notification Process
- ALE Queue

Reference: XXXX
Version: 1.0 Draft
Date: 05 Feb 2005

026_WLS
ALE Design
Component Workbook

Printed: 14/2/05
5.3 ALE Taps

As shown by the Fig 1 in the previous section the ALE monitors all packets coming from all the access points through a tap. This passive ethernet tap splits the signal before it is fed into the ALE’s monitor port. The diagrams below illustrate the ethernet taps schematic.

Figure 2 RJ-45 wiring

Figure 3 Ethernet Tap Schematic

The access point will plug directly into the port labelled Host on the left hand side of the diagram. The ALE will plug into TapA and the switch will plug into the Host port on the right hand side of the diagram. TapB will not be used.

5.4 ALE Classes

The class diagram for the ALE is shown in Figure 4.
5.4.1 New MAC Notification

Figure 5 shown below is a sequence diagram giving an illustration of the interaction between various ALE objects when a new MAC address is discovered.
The sequence diagram shows the interaction of the ALE objects from the initial packet capture at the tap to the transmission of messages to the LIS.

**Figure 5 New MAC Notification**

5.4.2 Deleted MAC Notification

Figure 6 Deleted MAC Notification shows the interaction between various objects in the ALE when a MAC address has expired.

It is to be noted that the Timer class handles the checking and deletion of entries within the AleTable class directly to improve the overall efficiency of the ALE.
5.4.3 Delete MAC Request

Figure 7 shows the interaction between various classes from the time the main program makes a select call and determines there is data to read to when the data is deleted from the Mac address table.
Figure 7 Delete MAC Request

5.4.4 Query MAC

Figure 8 Query MAC Request

5.5 LIS Communication

The LIS Communicator component of the ALE handles all transmission of messages between the ALE and the LIS. The underlying protocol used is UDP with support for acknowledgment and timeouts.
The ALE will provide a mechanism to retransmit un-acknowledged UDP packets after a short interval provided by the ALE config file until such a time that an ACK has been received or the message has been re-transmitted greater than the maximum allowable number of times.

5.5.1 Message Buffering
To ensure maximum efficiency multiple messages to be transmitted to the LIS are contained within a single UDP packet. The LIS Communicator class will buffer all outgoing messages, thus only sending UDP packets to the LIS when close to the maximum allowable UDP packet size is reached.
Chapter 6 Interfaces

6.1 User support

6.1.1 Environment

All files required for the ALE prototype including source files are located in /home/ale/

6.1.2 Operation

The Makefile located in the ale directory takes care of all compilation and inclusion of debug pre-processor directives (see section 6.13).

It must be noted however, to run the ./sniff executable created after compilation by the Makefile, the linux OS must be in super user mode (root). Failure to do so will result in an error whilst capturing packets off the tap.

6.1.3 Debug

Debugging of the ALE is activated by way of the C++ pre-processor at compile time.

The makefile which resides in the ALE parent directory incorporates a FLAGS variable which is used to turn on the debugging of various ALE classes. Each ALE class can be debugged individually or with other ALE classes with debugging output printed to STDERR.

Below is a table containing all of the possible flags that can be used for debugging and their associated classes.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-DSNIFF_DEBUG</td>
<td>Debug flag for main C++ function in charge of select calls etc.</td>
</tr>
<tr>
<td>-DALE_DEBUG</td>
<td>Debug flag for Ale class in charge of processing packets.</td>
</tr>
<tr>
<td>-DALE_TABLE_DEBUG</td>
<td>Debug flag for AleTable class. This debug flag is useful for debugging the ALE's internal data structure holding MAC address entries.</td>
</tr>
<tr>
<td>-DMAC_DEBUG</td>
<td>Debug flag for MacAddress class used as the key in the AleTable.</td>
</tr>
<tr>
<td>-DTIMER_DEBUG</td>
<td>Debug flag for Timer class.</td>
</tr>
<tr>
<td>-DTAP_DEBUG</td>
<td>Debug flag for Tap class. Useful for debugging packet capture and address extraction.</td>
</tr>
<tr>
<td>-DMESS_GEN_DEBUG</td>
<td>Debug flag for MessageGenerator class. Useful for debugging the formulation of out going messages.</td>
</tr>
<tr>
<td>-DDIGEST_DEBUG</td>
<td>Debug flag for Digest class.</td>
</tr>
<tr>
<td>LIS_COMM_DEBUG</td>
<td>Debug flag for LisCommunicator class. Useful in debugging LIS communication and message buffering.</td>
</tr>
</tbody>
</table>

Printed: 14/2/05
Reference: XXXX
ALE Design
Component Workbook
Version: 1.0 Draft
Date: 05 Feb 2005
6.1.4 Configuration and datafill

All configuration information is contained in the file ALE.cfg and includes information needed for initialising the ALE at startup.

Note: It is crucial that the order in which elements are contained within the config file and the space between the elements remain unchanged. It should only be necessary to change the values on the right hand side of the equals sign.

It is also necessary to ensure that the number of device names in the config file is equal to the number of taps and that the timer interval is a value representing the number of seconds.

A sample version of the config file is shown below:

\[
\begin{align*}
\text{NUM_TAPS} &= 3 \\
\text{DEVICE NAMES} &= \text{eth0 eth1 eth2} \\
\text{DEVICE_PORT_NUMS} &= 1700\ 1701\ 1702 \\
\text{ACK_RETRANS} &= 10 \\
\text{DEVICE_ID} &= 127.0.0.1 \\
\text{KEY} &= \text{ox00\ oxFF\ ox10\ oxFA\ ox00\ ox00\ ox2D\ oxFA\ ox00\ etc...} \\
\text{LIS_ID} &= 127.0.0.1 \\
\text{PORT} &= 7515 \\
\text{RETRANS_MAX} &= 10 \\
\text{VERSION} &= 1 \\
\text{TIMER_INTERVAL} &= 6 \\
\text{EXPIRY} &= 60
\end{align*}
\]

Note that eth0 has a port number of 1700, eth1 has port number 1701 and eth2 has a port number of 1702. Also the key must be 20 characters long and expressed in hexadecimal format.

ACK_RETRANS is the time interval in seconds that all unacknowledged packets will be resent.

DEVICE_ID is the ip address of the Ale.

LIS_ID is the ip address of the lis.
PORT is the port number which the Ale-LIS communication takes place on.

RETRANS_MAX is the maximum number of times data can be retransmitted.

VERSION the version number of the Ale-LIS protocol.

TIMER_INTERVAL is the interval in which all of the timer values in the aleTable are incremented and is directly related to the ACK_RETRANS value. For example: If ACK_RETRANS = 10(sec) and TIMER_INTERVAL = 6, the timer values in the aleTable will be incremented every 60sec as (ACK_RETRANS * TIMER_INTERVAL) = 60.

EXPIRY is the value at which Mac address entries in the aleTable are deemed to be expired.
Chapter 7  Implementation

7.1 Ale table implementation

The table used by the ALE to store each MAC address entry is an implementation of a STL map. Each entry has a key/data pair and is referenced by a MAC address contained in the MacAddress class.

The data in the MAC Address table is embedded within the structure AleMapData as is illustrated below in figure 9.

Fig 9 MAC Address table

<table>
<thead>
<tr>
<th>MAC address Table</th>
<th>KEY</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key/Data pair</td>
<td>MacAddress</td>
<td>AleMapData</td>
</tr>
<tr>
<td></td>
<td>-macAdd : byte</td>
<td>+macAddress : byte</td>
</tr>
<tr>
<td></td>
<td>+setMac()</td>
<td>+ipAddress : byte</td>
</tr>
<tr>
<td></td>
<td>+MacAddress()</td>
<td>+portNumber : int</td>
</tr>
<tr>
<td></td>
<td>+MacAddress()</td>
<td>+expiryTime : int</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-deviceName : char</td>
</tr>
</tbody>
</table>
Chapter 8  Testing

8.1 Test approach

The test harnesses used in testing the ALE are mainly focused on unit testing. CxxTest libraries were used to create test suites for every class used in the implementation of the ALE.

The ALE-LIS protocol testing uses manual test harnesses incorporating an implementation of a test LIS.

8.2 Test harnesses design

Test suites for each of the ALE classes are located in the directories which house the source files for that particular class. The table below shows the names of all test suites their locations and names of the classes they test.

<table>
<thead>
<tr>
<th>Test Suite Name</th>
<th>Location</th>
<th>Classes Tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimerTestSuite.h</td>
<td>ale/timer/</td>
<td>Timer (timer.h, timer.cpp)</td>
</tr>
</tbody>
</table>
| AleTableTestSuite.h          | ale/aletable/ | AleTable (aleable.h, aleTable.cpp)  
|                              |             | MacAddress (MacAddress.h, MacAddress.cpp)         |
| TapTestSuite.h               | ale/tap/    | Tap (tap.h, tap.cpp)                               |
| ConfigTestSuite.h            | ale/        | Config (config.h, config.cpp)                      |
| MessageGeneratorTestSuite.h  | ale/messgen/ | MessageGenerator(messageGenerator.h, messageGenerator.cpp) |
| LisCommTestSuite.h           | ale/liscomm/ | LisCommunicator (lisCommunicator.h, lisCommunicator.cpp) |
| DigestTestSuite              | ale/messgen/ | Digest(digest.h, digest.cpp)                       |

8.3 Test procedures

All test harnesses are compiled by the Makefile in the ALE’s parent directory.

Test harnesses are compiled by the command `make test_target_name`
Below is a list of all test target names. **Note:** theses target names are case-sensitive.

- testTimer
- testConfig
- testTap
- testAleTable
- testMessageGenerator
- testDigest
- testLisComm

Once compiled type the command `.runner` to run the compiled test harness.

### 8.4 Test coverage

#### 8.4.1 Timer

- Incrementing of timers for each entry of an empty Mac Address table
- Incrementation of timers and examines deleted Mac Address entry when a single entry in Mac Address table is expired.
- Test incrementation of timers and examines deleted Mac Address entries when multiple entries in Mac Address table are expired.
- Test incrementation of timers when no entries in Mac Address table are expired.

#### 8.4.2 AleTable

- Creation of a MacAddress key
- Insertion of a new Mac address entries.
- Tests the insertion of Duplicate Mac Address entries to ensure that only unique entries are store in the Mac address table.
- Retrieval of data from the Mac address table.
- Search based on MacAddress key

#### 8.4.3 Tap

- Initialisation of tap
- Retrieval of relavent source and destination addresses

#### 8.4.4 MessageGenerator

- SHA-1 Hash Generation
  - Two sets of identical data
  - two sets of different data
8.4.5  **LisCommunicator**

- Buffer control.
  - Adding message to empty buffer
  - Adding message to full buffer
  - Adding message to half empty buffer
- Sending messages.
- Re-transmissions.
- Acknowledgments.

8.4.6  **Config**

- Loading of config file.
- extraction of config properties.