D3.9: ULOOP Software Suite

Deliverable Number	D3.9
Lead Beneficiary	Caixa Mágica Software
Dissemination Level	Public
Work Package / Task	WP3, Task 3.4
Editors	Alfredo Matos (CMS)
List of Authors	Alfredo Matos and Nuno Martins (CMS)
Project Month & Date	Month 38, 31.10.2013
QAT Reviewer	Paulo Mendes (ULHT)

Executive Summary

This document highlights the final and public release of the ULOOP Software Suite. The final version of the software suite is an evolution of the software suite pre release – D3.7 [1] – correcting functional behaviour, and preparing the software image for public release and use. Similarly to D3.7, it is composed of the software images compiled as the integrated results of the software delivered by the different ULOOP tasks, and complemented by the integration and interoperability blocks developed in Task 3.4, targeting the two main platforms: Android and OpenWrt-Linux.

For clarity this accompanying document details the improvements and changes achieved from the testing, deployment and validation of the Software Suite pre-release, presented in D4.2 [2], a joint Task 4.2 and Task 4.3 effort. Based on the feedback received from the lab testing, the validation in the test sites provided by Task 4.2, and by the pilot deployments in Task 4.3, several aspects were identified and improved upon to enhance the ULOOP Software suite.

The public release of the software image also requires that the installation methods should be updated and improved upon to avoid using private project software repositories and allow easy distribution not only of the software images, but also of the source code. Accordingly we also detail the locations and methodology for the sites where the software is publically available.

This deliverable is comprised of the software source for the integrated ULOOP images, the DoxyGen documentation of the final version of the integration and interoperability blocks, the compiled binaries for the target platforms of the corresponding software, the installation methods and source packages available for installation, composing the public release of the ULOOP Software Suite.

Table of Contents

List of Figures

List of Tables

Acronyms

Acronym	Meaning
AP	Access Point
CI	Continuous Integration
DHCP	Dynamic Host Configuration Protocol
LGPL	Lesser GNU Public License
LUCI	Lua Unified Configuration Interface
SIM	Subscriver Identity Module
SMS	Short Message Service
SVN	Subversion

Acknowledgements

We would like to acknowledge every project member who contributed to the ULOOP Software Suite, especially those involved in the process of improving the changes from D3.7 to D3.9, especially Carlos Ballester Lafuente, from UniGe, and Imanol Fuidio, from FON.

1. Introduction

One of the ULOOP project main goals is to produce a cohesive Software Suite that showcases the different technologies developed in the project. To achieve this, as outlined in the ULOOP's Description of Work, the project defined several stages. The first step was the development of individual task blocks, which when completed were properly integrated into the software image pre-release – ULOOP D3.7 [1].

The ULOOP Software pre-release combined every different piece of software originating from WP3 tasks, specifically Task 3.1, on Trust Management, Task 3.2 on Resource Management, and Task 3.3 on Mobility Aspects, along with the interoperability and integration blocks from Task 3.4, into a combined software suite, with all of the developed components[1] to ensure interoperability of the software between modules, between devices, and across different networks. The software suite combined the main results of the different tasks, targeting the two different target architectures as explained in D3.7: ULOOP OpenWrt images and the ULOOP Android application. These two resulting images work together to provide the ULOOP functionality on Access Points (OpenWrt) and on end-user devices (Android). The integration outcome on the node side is an Android application that can associate to a ULOOP-enabled network and successfully request Internet access considering the ULOOP Node. On the gateway side, the result is a customized OpenWrt firmware that carries that can successfully negotiate Internet access with a client device, and facilitate access to ULOOP resources.

These results are clearly described in D3.7, which also comprises a report that contains the information regarding the components that define each image, the integration steps towards creating the software image, the interoperability software and definitions. More importantly it contains the description of the software images for both the ULOOP Node (Android) and for the ULOOP Gateway (OpenWrt), along with the project's internal build process, which we extend here in Sec. 3 for external release.

[1] One of these developed components is the ULOOP message library, which is discussed in depth in D3.7 and extended here in Sec. 2.5

The ULOOP software suite pre-release was the starting point of the ULOOP Test site and Pilot deployments. In the integration process both the Android and OpenWrt image components were fairly tested in lab environments, to ensure that the software was fully functional. However, the real goal for testing and validation came from Task 4.2 (Test site validation) and Task 4.3 (Pilot validation). With the image release to Task 4.2, intensive testing was carried out on the pre-release, followed by pilot testing and validation.

The testing and validation process involved performing the tests outlined in the ULOOP test suite presented in D4.2 [2] and helped identify several aspects that required improvement on the overall Software Suite. Using the test cases, the ULOOP Software Suite, in its pre-release format, was subjected to in-lab testing on multiple devices and architecture, on the different test sites, and on the ULOOP pilots, which involved real end-users in a deployment environment. The feedback provided by such a demanding testing and validation process created several concrete items that require modifying and updating the ULOOP Software Suite for its final release. In fact, this was the intended process: to have an initial pre-release (D3.7) that could be tested, deployed and validated (D4.2), and then perfect the ULOOP Software Suite towards its public release (D3.9). In terms of overall software components the major blocks, functions and features are similar to those described in D3.7. Therefore, in this deliverable we present the major changes that resulted from the software testing and validation process, along with the steps that are required for a public release of the ULOOP software. We also identify a few open issues that result from the experimental nature of the software, but are interesting to acknowledge and to provide to any follow projects that re-use the ULOOP suite.

The main changes from D3.7 to D3.9 are highlighted in Section 2. Following the identified changes, the required information for public release is outlined in Sections 3, presenting the installation methods for 3rd parties to be able to build and run the software. Section 4 identifies the locations where the Software Suite is made publicly available. This document extends D3.7 in terms of documentation and software, and is accompanied by the corresponding software blocks, similarly to D3.7.

2. Software Suite Updates

The testing and deployment process carried out by Task 4.2 and Tasks 4.3 helped identify several issues that required updating the software suite pre-release. Some were functional that were detected by hands-on validation provided with end-users, while others were a result of testing different deployment scenarios, both on the test sites, and while experimenting with the software pre-release in different ULOOP scenarios.

In this section we outline the most important changes that result from the feedback provided by WP4, and that required significant updates, or provided substantial behaviour modification from the pre-release suite. In this section we only outline the changes, as the rest of the software feature remain similar to D3.7. For more information on the rest of the software please consult D3.7 [1]. It is worth mentioning that none of these changes impact any ULOOP concepts, but rather provide increased usability or streamline functions already present in D3.7.

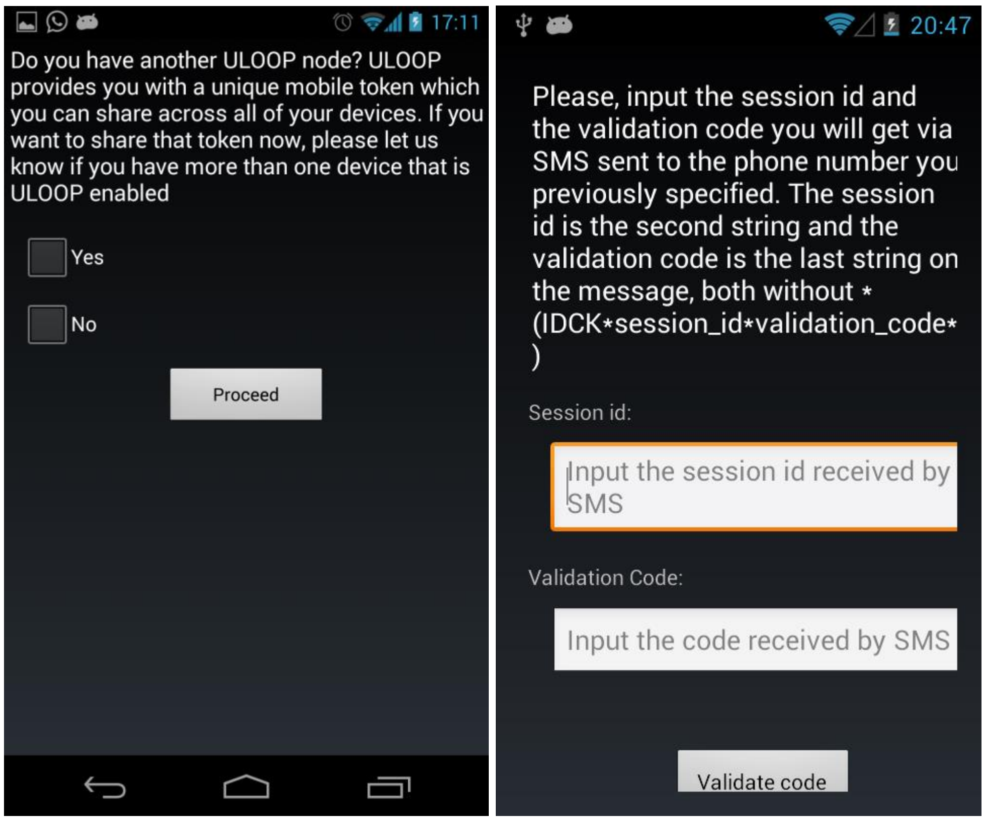
2.1 Crypto-Id Validation with Companion Device

Over the course of the deployment validation, either in the experiments carried out in Task 4.2, or with the pilot deployments belonging to Task 4.3, it was clear that there were several users running ULOOP software on tablet devices.

The main problem arising from this situation is that ULOOP, for legal compliance issues, requires validating the user account through a valid mobile carrier number. To achieve this, ULOOP resorts to Crypto-ID validation process through an SMS service provide by L7, which associates the mobile device number to the Crypto-ID. However, several tablet devices do not have a SIM card or mobile device number, making the validation process impossible to complete.

Enabling users to register using the ULOOP Android application on devices that do not have SIM cards available required the development of an alternate validation process that was no present in the ULOOP pre-release. Until now, a new user needed a valid SIM card on the registering device to receive the confirmation SMS message. Based on the received feedback, WP3 updated the ULOOP software, in this case the ULOOP Android Application to enable users to complete the registration process on a device that does not have a SIM card available by using a companion device. This companion device must have a valid SIM card and corresponding mobile number, where the validation

SMS can be normally received through the carrier network. Using this mechanism, the user will receive an eight character code on the companion device, and then input it in the corresponding place on Android device input form, as outlined in Figure 1.


17:11

Do you have another ULOOP node? ULOOP provides you with a unique mobile token which you can share across all of your devices. If you want to share that token now, please let us know if you have more than one device that is ULOOP enabled

Yes

No

Proceed

20:47

Please, input the session id and the validation code you will get via SMS sent to the phone number you previously specified. The session id is the second string and the validation code is the last string on the message, both without * (IDCK*session_id*validation_code*)

Session id:

Input the session id received by SMS

Validation Code:

Input the code received by SMS

Validate code

Figure 1: ULOOP companion device request and input screens.

From this point on, and following correct reception of the SMS, the registration process is similar to that previously described in D3.7 using the ULOOP Android Application mechanism available in the pre-release software suite.

While this process is not as seamless as previously supported, it allows users on tablet or otherwise missing a SIM card to properly complete the registration process and join ULOOP-enabled networks. This simple usability modification, which does not impact any of the concepts or legal requirements, is a small modification that carries the impact of enabling access to all tablet devices. This process can also be applied to other devices such as Access Points (AP), laptops or any other device, creating a much wider potential audience for the ULOOP Software Suite.

2.2 Access Point Session Termination

Even as an experimental test suite for research purposes the ULOOP Software Suite is intended to reach a considerable amount of users, and even production environments, which was the case of the ULOOP Pilot deployments in Task 4.3. Therefore, a complete usable function set must be provided.

During the deployment phase it was detected that even though the main ULOOP process was being followed, once the network access was granted, there was no actual termination, once the access concluded. In practice, a successful connection enabled the user to gain access to the Internet, bypassing the captive portal limitations, but once available it was never reset. While this behavior is perfectly acceptable for lab testing, it was identified during the testing and experiments that it could not be endured in pilot demonstrations, as users would be able to completely bypass ULOOP mechanism under certain circumstances. These circumstances include leaving and returning to the same AP, would allow access without resource expenditure as outlined in D2.3 [3] and later in D3 [4], if the AP was not manually reset during that period.

To cope with this limitation, the integration task provided a simple beaconing system to determine the availability of the user. Using this straightforward mechanism the AP receives periodic messages asserting that the validated user is still attached to it. Once it is detected that the user is no longer reachable, by failing to provide periodic messages, the corresponding access rules in the ULOOP OpenWrt application will be automatically reset. Pragmatically, the ULOOP OpenWrt daemon will maintain a session open as long as the gateway receives the periodic Message, maintaining user sessions.

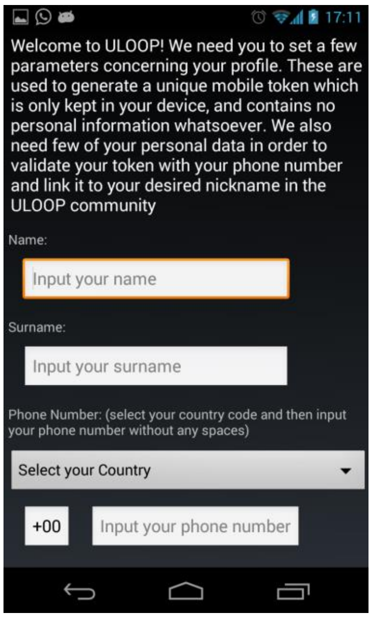
This solution introduces the preliminary support for AP session termination. Even though the initial mechanism implemented is based on periodic messages, it could potentially be extended to support resource starvation mechanisms, time based mechanisms, or any other option for session termination.

2.3 Android UI Update

Testing the ULOOP Software suite with real end-users provided qualitative feedback on the software. This is particularly true for the Android Application, which was the main part available to regular end-users. After collecting and parsing the results and metrics presented in D4.2 [2], obtained from the test site experiments (Tasks 4.2) and from the ULOOP pilots (Task 4.3), it was clear that the application had several aspects that could benefits from usability improvements:

1. There were several errors inputting the validation number on the application.
2. There were several users that validated their Crypto-ID, but then did not proceed to request Internet access, as defined by ULOOP.

On the problem outlined in (1), it was identified that users were trying to complete the registration process, but failing to receive the validation SMS. After checking the SMS service, it was identified that several users were not properly inputting their mobile phone number, and therefore failed to receive the SMS. This was mostly due to the international area code format, where the application requires that neither *00* nor + *signs* are used, and several users, failing to read the instructions, provided their numbers prefixed with *00* or + *sign* or without international prefix altogether*.* The consequence of this issue was the total inability to complete the registration process and consequently to access the network.


17:11

Welcome to ULOOP! We need you to set a few parameters concerning your profile. These are used to generate a unique mobile token which is only kept in your device, and contains no personal information whatsoever. We also need few of your personal data in order to validate your token with your phone number and link it to your desired nickname in the ULOOP community

Name:

Input your name

Surname:

Input your surname

Phone Number: (select your country code and then input your phone number without any spaces)

Select your Country

+00 Input your phone number

Figure 2: ULOOP updated information and input fields for SMS validation.

To correct this issue the input forms were separated in both international prefix and mobile phone number, in order to enable proper validation on the application. The second change was the introduction of pre-filled example values to further clarify what was expected as input. These changes are highlighted in Figure 2.

It is expected that this issue creates a higher success rate on the completion of the registration process, and therefore increase user adoption.

Regarding the second (2) issue identified above, one of the reasons attributed to the failure to request Internet access, it was the fact that all options on the application were equally accessible. Therefore, the UI was adapted to highlight the request Internet button, providing a clear outline of the main function for the application. This is shown in Figure 3.

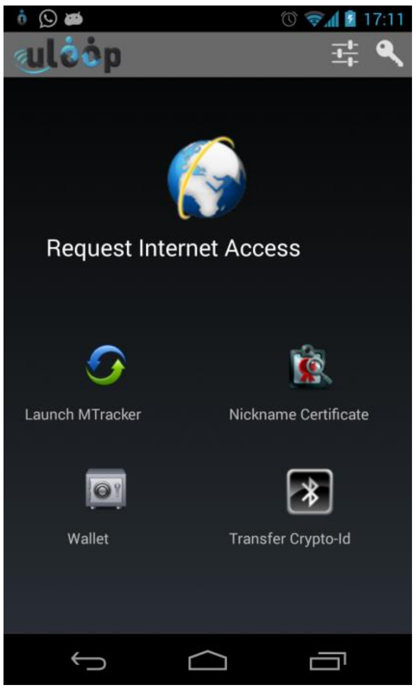


Figure 3: ULOOP validation code (obtained from companion device SMS) input screen.

These two modifications combined, along with the extra information on the expected application behavior are expected to greatly reduce the usability problems identified in the course of the pilot, as well as reduce any friction or frustration from using experimental software.

2.4 Multiple Architecture support

Most of the testing for the D3.7 pre-release images used the same Netgear (WNDR3800) routers, which was an important coordination tool to ensure the integration of the software, without facing any overhead of cross-platform issues. However, when moving to the testing phase in Task 4.2, it was required to run the images on the different images concerning the test sites.

Moving to the different sites required cross compiling the software for the indented architectures, as well as adjusting the configuration to match the different sites. The different architecture encountered on the test sites[2], which required some changes and corrections were the following:

- Level 7 Access Points: PCEngine ALIX 3D
- TUB (BOWL) Access Points: Asus WL-500GP
- UniUrb (UWIC) Access Points: RB433 Mikrotik router board.

The PCEngine Alix 3D compile process was successfully, but several issues regarding the device driver in use for the wireless cards. It was necessary to correctly assign the wireless (ath5k) and then adapt the configuration file to requesting an IP address from Level7 DHCP pool server.

The RB433 Mikrotik router board booted supports almost all ULOOP software installed, but required some modifications, and adaptations. Specifically, since some LUCI modules were not available and not installable, some configuration had to be specified through command line options and manual configuration. This stems from the fact that this router needs a specific OpenWrt revision (rev. 35630), which is not available from the stable release, and impacts the configuration interface.

In conclusion the process of going from 3.7 to D3.9 uncovered several issues that were identified and corrected. Generic issues regarding the default network configuration, which was not suitable for some devices specific to some test sites, created some issues regarding access to network resources. All of these issues were fixed with the support of Task 3.4, enabling the correct deployment in Task 4.2, and

[2] For more information on the Test Sites please consult D4.2 [2].

ensuring the success of the test sites. This also showed the validity of the overall validation process, and of the value of the cross Work package communication channel between Task 3.4 and Task 4.2.

2.5 Libmessage Monitor

One of the most complex situations identified during the testing period was the difficulty of following the ULOOP message flows throughout the different devices and modules. This required following the output of several blocks, and even nodes running the ULOOP software. Given that the ULOOP interfacing is achieved through a common library developed inside Task 3.4, it was possible to develop a mechanism that supports a new visualization mechanism without impacting the software.

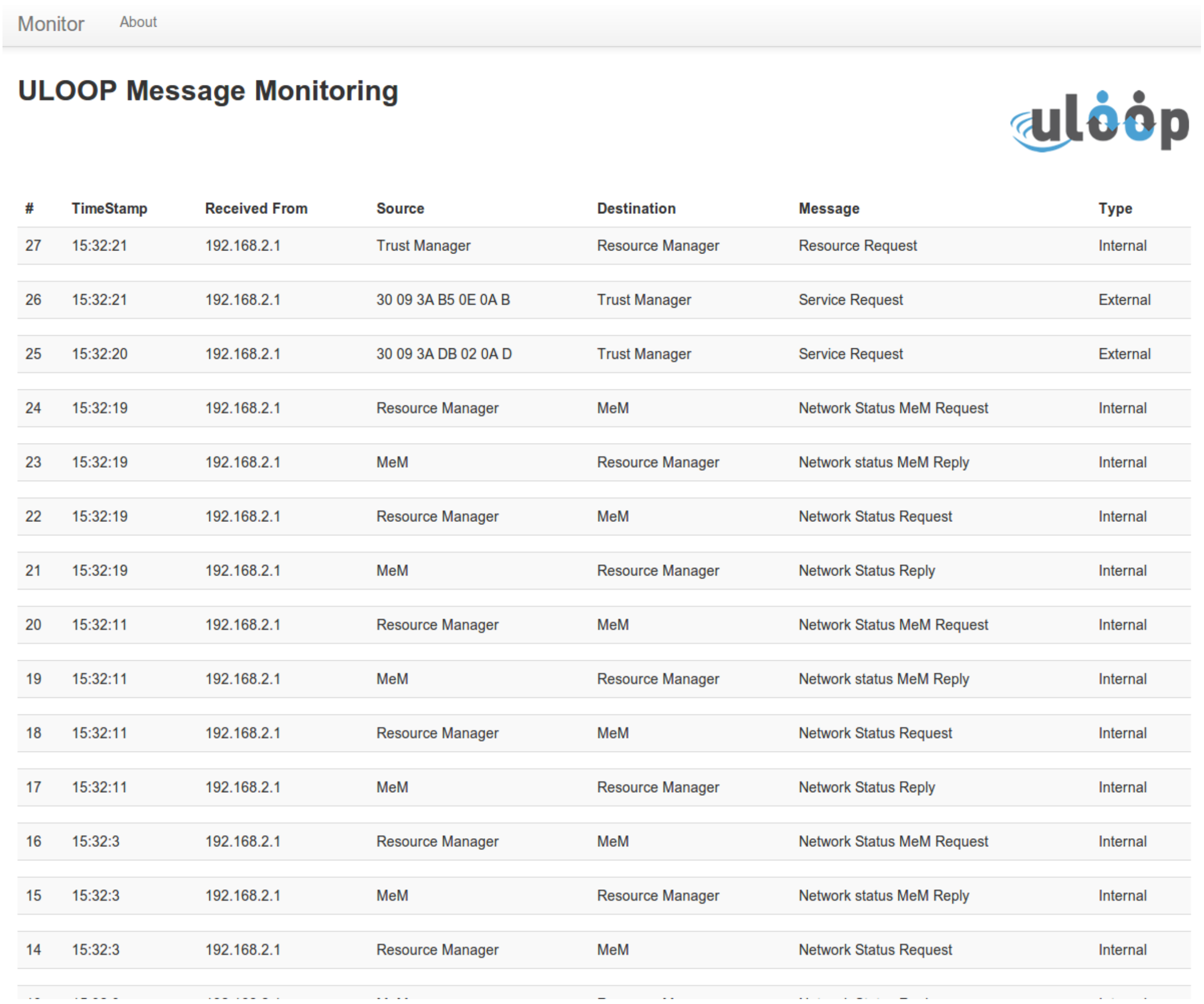
Based on the fact that every software component and node uses the same message library, Task 3.4 developed an extension to the libmessage library that supports a message monitor. The message monitor serves two purposes: to check for proper message format and to create a visual and centralized debug tool that incorporates all of the different components.

The message monitor is node.js[3] server that understands the ULOOP Protocol Buffer format. When receiving a message it validates and outputs the message to an HTML page, where the flows can be verified against ULOOP requirements.

To support this operation feature, a few small changes to the libmessage code (libmessage was thoroughly explained in D3.7) were required. The changes imply that whenever a message is transmitted or received through the common libmessage code, it is copied and transmitted to the monitor server, which displays the received message. An early example of the visualization of the monitor server is provided in Figure 4.

After this refactoring both parts (message and monitor) are clearly separated. On the message library core code, the monitor only touches on specific control points, leveraging the potential of both parts.

[3] Http://nodejs.org



ULOOP Message Monitoring

#	TimeStamp	Received From	Source	Destination	Message	Type
27	15:32:21	192.168.2.1	Trust Manager	Resource Manager	Resource Request	Internal
26	15:32:21	192.168.2.1	30 09 3A B5 0E 0A B	Trust Manager	Service Request	External
25	15:32:20	192.168.2.1	30 09 3A DB 02 0A D	Trust Manager	Service Request	External
24	15:32:19	192.168.2.1	Resource Manager	MeM	Network Status MeM Request	Internal
23	15:32:19	192.168.2.1	MeM	Resource Manager	Network status MeM Reply	Internal
22	15:32:19	192.168.2.1	Resource Manager	MeM	Network Status Request	Internal
21	15:32:19	192.168.2.1	MeM	Resource Manager	Network Status Reply	Internal
20	15:32:11	192.168.2.1	Resource Manager	MeM	Network Status MeM Request	Internal
19	15:32:11	192.168.2.1	MeM	Resource Manager	Network status MeM Reply	Internal
18	15:32:11	192.168.2.1	Resource Manager	MeM	Network Status Request	Internal
17	15:32:11	192.168.2.1	MeM	Resource Manager	Network Status Reply	Internal
16	15:32:3	192.168.2.1	Resource Manager	MeM	Network Status MeM Request	Internal
15	15:32:3	192.168.2.1	MeM	Resource Manager	Network status MeM Reply	Internal
14	15:32:3	192.168.2.1	Resource Manager	MeM	Network Status Request	Internal

Figure 4: ULOOP libmessage monitor server example output screen.

The libmessage monitor allows ULOOP message exchanges to continue seamlessly without any negative impact, and maximizes the visualization, either for debug or demo purposes. If there is no available server, libmessage will silently continue to operate without any negative consequences.

2.6 Continues integration and Issue Correction

Going from a pre-release into the final public Software release required following a continuous integration (CI) methodology, especially as the changes outlined in the previous sections were integrated into the main software suite.

The new software functionalities have been performed under the CI methodology using the validation tests and environments build during the WP4. Once the new functionality was made available, the software was subjected to the validation tests. If any of the tests failed the software was evolved until the new functionality became compatible with the previous running versions and the tests were passed without failures. However, many of the new functionalities were not deployed to the pilot sites due to the time and user constraints (repeating the same test and pilot experiments is not feasible during the lifetime of the project). It is expected that each new feature and correction will help to engage new ULOOP users with the ULOOP communities and facilitate the ULOOP user experience.

As a consequence of the CI and permanent dialog with tasks 4.2 and 4.3, several software issues (e.g. memory leaks, optimizations, etc) were also included in the software. This allowed task 3.4 to increase the overall quality of the software and evolve from a beta release towards a more stable and final software release. An example of such fixed issues would be low priority configurations issues detect during the deployment process. In fact, most network related issues occurred due to improper network configurations, as every device has specific set of interfaces and interface names.

However, while several important issues were corrected, a few minor issues or limitations resulting from the experimental nature of the project were adequately identified, but not entirely resolved. The reasoning behind these decisions were based on the experimental status of the Software Suite, as the results of an R&D project, and to adequately meet the project deadlines, because, as indicated above, developing and integrating software is a continuous effort. Also, there are some aspects that resulted from the applied technology that required adjusting some concepts to pragmatic solutions. Below is the list of the most relevant issues visible on the final ULOOP Software Suite that did not compromise the project's demonstrations, proofs-of-concept, experiments or pilot and test site deployments.

Table 1: Issue List for ULOOP Software Suite final release.

Topic	Detailed Description	Platform Components	Justification
Misconfigured network interfaces	For certain OpenWrt router the interfaces network configuration at boot is not correctly defined	OpenWrt Network configuration	OpenWrt support for detecting specific models under the same profile lacks a configuration mechanism to correctly define the correct network interfaces in the UCI. This occurs on a very limited number of devices, and manual configuration is a

			viable scenario in such cases.
ULOOP Image Build may fail on 64 bit Linux Host architectures	OpenWrt compilation under some 64 bit Linux distributions may fail due to wrong library paths	OpenWrt Build Environment	This is a well-known (reported) non-critical bug in the OpenWrt development environment, and only occurs on specific 64 bit Linux distributions (e.g. OpenSuse). Where some libraries files are not in the expected library path and the project will wait for the OpenWrt fix.
ULOOP tokens with different data types	Tokens in the userspace applications are represented with double data. In the Linux Kernel floating point unit is not used hence the token need to be change to an integer.	OpenWrt userspace and Kernel modules	Due to unavailability of the floating point unit (FPU) in the kernel, the token data type needed to be changed to integer, with multipliers according to precision requirements. This is the most standard way of dealing with FPU operations in the Kernel space.
libmessage (protobuffers) message format errors	In certain conditions the protobuf implementation fails on parsing the received frame, causing the trust association to be restarted	OpenWrt and Android - libmessage	Under certain unverified conditions message parsing may fail, and messages are lost, failing to complete association. This has proven to be a hard to track problem, given the seldom nature. Since retriggering the request completes the association successfully, this is non-critical issue.
ESM limited to a maximum of client two devices	When connecting more than two devices to a router, the ESM will refuse further connections.	OpenWrt - ESM	This is due to the fact that low level 802.11 implementations in the kernel are very challenging, and the current system is sufficient towards a proof-of-concept that demonstrates the required functionality and benefits.
Non-validated Crypto ID's can access the Internet	When a user does not validate, or fails to validate his cryptoID, he can still request Internet Access.	Android Trust Manager	Enforcing validation was not a requirement. To deal with non-validated nodes, a quarantine mechanism would be required, which for experimental purposes

			was not mandatory.
Duplicate User Nick Names	When validating a nick name that already exists, there is an error not displayed by the UI	Android Trust Manager	For test experiments it is unlikely that there will be collision in nicknames. Also, the cryptoID is the main user identifier, and those are cryptographically unique.

3. Installation Methods

During the course of the project it was necessary to create a common development environment[4]. Even though the main part of the environment was already in place through the default OpenWRT build environment, for ULOOP it was also necessary to include code and patches that would modify the final image, either in configuration or available packages.

As clearly outlined in D3.7, the solution was to build a bootstrap script[5] that configures the ULOOP feeds (OpenWrt specific mechanism) and the OpenWRT build environment. Even though the initial goal for the script was to facilitated ULOOP partners to build and develop their work based on the ULOOP software, with the project providing the public image of the source code, it is necessary to provide some updates on the bootstrap script to adapt to external developer's needs.

Upon public release, download and configuration of the OpenWrt build environment with ULOOP software can be done through two simple methods:

- Method #1: Scripted installation (stable or development)
- Method #2: Source package installation

In this section we present a very brief overview of the different available methods to assist future uptake of the ULOOP Software Suite for research and development purposes.

It is worth mentioning that the Android application is installable through the standard Android mechanism, and is easily modifiable through the available project on the source code (using Eclipse).

[4] It is worth mentioning that the Android application is installable through standard Android mechanisms, and is easily modifiable through the available project on the source code (using Eclipse). Therefore, in this section we focus mostly on the OpenWrt aspects, which are the most elaborate. For the Android information regarding the ULOOP Software Suite please refer to D3.7.

[5] More detailed information on the bootstrap script operations and mechanisms is available from D3.7.

3.1 Scripted Installation

The first and initial method is to download and execute a script from the ULOOP package repository. When executed, the script will download every ULOOP component and configure the build environment. This method is not restricted to ULOOP partners, so any developer can use it. An example follows:

```
$ curl -sS http://uloop.eu/software/uloop-software-suite-install.sh | bash
```

The above command will download all the necessary ULOOP package compressed files and the OpenWrt generic build environment. Next it applies all ULOOP specific configurations and patches, which allows building an ULOOP firmware image. This version was designed to be as simple as possible, and requires mostly no user interaction, beyond selecting the correct device architecture and profile on the configuration menu.

The script also allows a specific option, which is to build the development version of the ULOOP Software Suite, directly from an SVN repository. To achieve the ULOOP development version, rather than the stable release, it is necessary to run the script with the 'dev' option as described below:

```
$ curl -sS http://uloop.eu/software/uloop-software-suite-install.sh | bash -s -- --dev
```

This script downloads and configures the OpenWrt build environment with ULOOP code and patches. Before the actual compilation and build takes place, it is necessary to define the router architecture and profile in the Configurations menu (that will be displayed in the console terminal).

From this moment, the compilation toolchain is downloaded, compiled and installed. The toolchain is a set of essential tools to compile the code for the actual router. If no error is shown, the ULOOP firmware image will be at the bin folder of backfire-10.03.1.

3.1.1 Subversion access

There is an alternative way for those wishing to review the SVN repository, in order to understand the evolution of the software and the most recent changes. To do this it is necessary to perform an SVN checkout of the ULOOP repository. This is indented towards experienced development use, as it retrieves the latest modification of the ULOOP source code, and ensures that the development history for every software block can be visualized (which will not happen on other installation methods).

Originally this method was restricted to ULOOP partners, as the SVN repository used in the svn-build script was the private instance only available to ULOOP partners with valid credentials. With the release of the public source code, explained in Section 4, there are public SVN repositories that used by external contributors or interested developers.

As mentioned, to use this method it is necessary to make a copy of ULOOP code by doing a checkout of the ULOOP SVN to a folder on your system, e.g.:

```
$ svn checkout example.com/svn/uloop/v4.0/ "local-uloop-tree"
```

After the checkout is completed, all ULOOP code from version 4.0[6] is on the folder “local-uloop-tree”. Following this checkout the source code is available for inspection. Building directly from this tree is a manual process. However, this can be used to establish a replicate or private repository, and later point the script mentioned in Section 3.1 to this location.

3.2 Source Package Installation

The second method is to download a single compressed file from the ULOOP public repositories mentioned in Section 4, with all ULOOP and OpenWrt build environment pre-configured. This compressed file has all ULOOP patches applied and all ULOOP packages code available. Moreover, all ULOOP specific configurations are as well applied.

The provided source packages include the ULOOP Android code as well as the ULOOP OpenWrt code. The Android part, as mentioned before, is omitted from the instructions, due to the straightforward use of the Eclipse project.

The source package comes with a pre-configure OpenWrt build environment. However, the build environment on the compressed file is generic, which means that it will be necessary to select the correct architecture and profile for the intended router. An example follows:

```
$ cd "extracted-folder-name"/openwrt/

"extracted-folder-name$ make menuconfig
```

[6] Version 4.0 is the internal ULOOP reference to the integrated version of the ULOOP Software Suite.

To start the compilation process execute *make* in the command line interface. The example command is presented below:

```
”extracted-folder-name$ make
```

The source package was developed with the aim of providing a single ULOOP downloadable file, which can be converted into the software images. This means that it is possible to perform modifications of the source code using the final stable release, and continuously build the image without requiring access to public or private subversion repository.

4. Source Code Availability

The initial development of the ULOOP software took place using a project-wide Subversion repository. This allowed the project to use a proper source code revision tool that assisted the development process. However, this repository was limited to project partners, as access required proper credentials. As the ULOOP project moves towards the final and public release of the ULOOP code, it is necessary to make the code available for download at public location sites.

The ULOOP final software suite should be available as publically downloadable software packages (tarballs), as well as through public subversion repositories, mirroring the internal development process and facilitating the access for software developers to build upon the ULOOP contributions.

The project identified several vehicles for the ULOOP software, which is structured accordingly to the type of downloadable content:

- **Source Package Repositories**: the packages contain the source code highlighted in the previous sections, including the Android source code, the OpenWrt source code, and the OpenWrt build environment.

- **Source Code Repositories (SVN)**: this content refers to publically available subversion repositories containing the ULOOP software.

- **Image Repositories (binaries)**: this type of content includes the firmware images as well as the Android application.

The decision for main targets on which to place the different ULOOP downloadable content was based on impact, availably and features. In this case, three locations were identified to receive the ULOOP software suite downloadable content:

- ULOOP Website: This is the main ULOOP website which contains of the ULOOP achievements, milestones and packages. ***URL:*** ***http://uloop.eu/software***

- ResearchGate: ResearchGate is defined as a social network for scientists, and allows the social interaction between research scientists. It is recently gaining a very considerable user base, and is becoming an important dissemination research vehicle for research efforts. ***URL: http://www.researchgate.net/project/ULOOP***

- OpenSourceProjects.EU: Open Source Projects is a recent effort by an FP7 CSA to create a software forge for European-funded projects. It provides subversion repositories, bug tracking tools, and all of the necessary tools to support a development community around an open source project. ***URL: http://opensourceprojects.eu/p/uloop***

These three targets cover most of the ULOOP requirements for publically distributing the final ULOOP Software Suite. Next we determine which content is available on which repositories.

4.1 Source Package Repositories

ULOOP packages are also available from all of the main dissemination vehicles highlighted above. The downloadable packages are provided on the ULOOP website, as part of the project results, on ResearchGate as a project result, and on OpenSourceProjects, as a software project downloadable file.

For the OpenWrt software this is expected to be the most common distributed method, as well as for any developer that desires to quickly inspect the ULOOP Software Suite source code.

4.2 Image and Application Repositories

The most important distribution method for OpenWrt images is the actual source code, as it most likely requires a custom build for the intended device (there are literally dozens of combinations between chipset, make and model for the OpenWrt images). However, ULOOP provides some sample images for one of the main test devices used in the project (Netgear WNDR3800). These images are distributed on the ULOOP website, on ResearchGate and also on OpenSourceProjects.

The most interesting part regarding distribution pertains to the Android image. The ULOOP Android Application is available from the ULOOP official website and OpenSourceProjects as a downloadable APK. Additionally, it is available directly from the Google Play Store as a download like any other Android application. The Google play store provides an important distribution method for Android applications, and provides most trustworthy mechanism for Android applications.

4.3 Source Code Repositories

As the project moves towards the public release, it is possible to provide a public SVN repository. As mentioned before the internal project repository requires valid credentials. However, out of the selected tools for dissemination of the software suite, the OpenSourceProjects platform provides freely and publically available Subversion and GIT repositories. This makes it interesting for the ULOOP project to provide a publically available Subversion repository:

- **Public SVN repository**: http://opensourceprojects.eu/svn/uloop/code/trunk
- **Source Code Browser**: http://opensourceprojects.eu/p/uloop/code

The combined tools provide an interesting way for developers to inspect and retrieve the ULOOP source code, which can be complementary to the build tools defined in Section 3.

5. Conclusions

Evolving from D3.7 involved a coordinated effort with WP4. The effort to continuously integrate the received feedback provided an important roadmap towards the construction of the final ULOOP Software Suite. While the pre-release (D3.7) was already in a usable state, the software test suite, the test site validation, the test site experiments, and the pilot deployments all contributed towards identifying several points where it was possible to improve the software suite, and provide a better results. This conclusion implicitly validates the roadmap of having a pre-release used as internal testing and deployment milestone, generating and incorporating the feedback into a polished and usable stable release, even for an experimental software suite.

The received feedback was of extreme importance, and allowed correcting important aspects such as the companion device validation, or the input field validation changes. These small, but significant changes contribute directly to the quality of the project results, and largely decrease adoption barriers for the ULOOP software suite.

With the software properly defined, based on the evolution from D3.7, we provided several mechanisms and tools first for building, and second for distribution. The building mechanisms provide user-friendly mechanisms for building the entire software suite, especially for OpenWrt, which has the more complex process. Android requires only (re)creating the Android project, which is already provided with the software packages.

Part of the work for this deliverable also requires finding and creating the necessary vehicles for the distribution of the software sources and packages, a standard process for any software release. To achieve this goal, we identified three different targets, the ULOOP site, ResearchGate, and OpenSourceProjects, each with its own value, and providing different vehicles for software download. This deliverable is provided together with the necessary software packages, and source code uploads to the target repositories, completing the ULOOP Software Suite distribution process, and the finalization of the source code, which will be available as an Open Source Software project under the GNU LGPLv3 license.

6. References

[1] ULOOP Deliverable D3.7 – ULOOP Software Suite Pre-release.

[2] ULOOP Deliverable D4.2 – Pilot Deployment and Validation

[3] ULOOP Deliverable D2.3 – ULOOP Overall Specification - http://siti.ulusofona.pt/aigaion/index.php/attachments/single/475

[4] ULOOP Deliverable D3 – ULOOP High Level Architecture Specification. (Private document)