D3.1.2 LMS and eBooks widgets for prototype lab courses – update

Deliverable Editor: Alexander Mikroyannidis (OU)
Other Contributors: Aitor Gomez-Goiri, Christos Tranoris, Daan Pareit, Jono Vanhie-Van Gerwen, Johann M. Marquez Barja, Mohammed Yasin Rahman, Olivier Fourmaux
Reviewers: Valina Katsaounou, John Domingue

<table>
<thead>
<tr>
<th>Document Identifier:</th>
<th>FORGE-2015-P-D3.1.2</th>
<th>Due Date:</th>
<th>31/07/2015</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>1.0</td>
<td>Submission date:</td>
<td>31/07/2015</td>
</tr>
<tr>
<td>State:</td>
<td>Final</td>
<td>Distribution level:</td>
<td>Public</td>
</tr>
</tbody>
</table>

This document is part of a research project funded by FP7 Framework Programme. Grant agreement number 610889.
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Amended by</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>28/05/2015</td>
<td>Alexander Mikroyannidis</td>
<td>Outline and assignment of partner responsibilities</td>
</tr>
<tr>
<td>0.2</td>
<td>11/06/2015</td>
<td>Alexander Mikroyannidis, Aitor Gomez-Goiri, Christos Tranoris, Daan Pareit, Jono Vanhie-Van Gerwen</td>
<td>First round of contributions</td>
</tr>
<tr>
<td>0.3</td>
<td>29/06/2015</td>
<td>Alexander Mikroyannidis, Johann M. Marquez Barja, Mohammed Yasin Rahman, Olivier Fourmaux</td>
<td>Second round of contributions</td>
</tr>
<tr>
<td>0.4</td>
<td>30/06/2015</td>
<td>Alexander Mikroyannidis</td>
<td>Draft for internal review</td>
</tr>
<tr>
<td>0.5</td>
<td>08/07/2015</td>
<td>Valina Katsaounou</td>
<td>Internal review #1</td>
</tr>
<tr>
<td>0.6</td>
<td>18/07/2015</td>
<td>John Domingue</td>
<td>Internal review #2</td>
</tr>
<tr>
<td>0.7</td>
<td>27/07/2015</td>
<td>Christos Tranoris</td>
<td>Scientific Director’s review</td>
</tr>
<tr>
<td>0.8</td>
<td>28/07/2015</td>
<td>Alexander Mikroyannidis, Aitor Gomez-Goiri, Johann M. Marquez, Olivier Fourmaux</td>
<td>Finalised document addressing the comments from the internal review</td>
</tr>
<tr>
<td>0.9</td>
<td>28/07/2015</td>
<td>Aneta Tumilowicz</td>
<td>Final QA</td>
</tr>
<tr>
<td>1.0</td>
<td>31/07/2015</td>
<td>Aneta Tumilowicz</td>
<td>Version for submission</td>
</tr>
</tbody>
</table>
## Table of Contents

Change Log ............................................................................................................................................... 2  
Table of Contents ........................................................................................................................................ 3  
List of Figures ........................................................................................................................................... 4  
Executive Summary .................................................................................................................................... 6  

1 Overview of technologies ...................................................................................................................... 7  
   1.1 Updated widget infrastructure .............................................................................................................. 8  
   1.2 Learning Analytics ................................................................................................................................. 9  

2 Updated widgets ..................................................................................................................................... 11  
   2.1 The PT Anywhere widget .................................................................................................................... 11  
      2.1.1 Widget architecture ....................................................................................................................... 11  
      2.1.2 New features .................................................................................................................................... 12  
      2.1.3 Further development plans ......................................................................................................... 23  
   2.2 The ssh2web FORGEBBox widget ....................................................................................................... 24  
      2.2.1 New features ..................................................................................................................................... 24  
      2.2.2 Further development plans ............................................................................................................ 26  
   2.3 The WLAN widget bundle ................................................................................................................ 26  
      2.3.1 New features ..................................................................................................................................... 26  
      2.3.2 Further development plans ............................................................................................................ 33  

3 New widgets .......................................................................................................................................... 34  
   3.1 The Blacksmith Widget Framework and Anvil Engine ....................................................................... 34  
      3.1.1 Learning context .............................................................................................................................. 34  
      3.1.2 Targeted FIRE testbed .................................................................................................................... 34  
      3.1.3 Implementation ............................................................................................................................... 35  
      3.1.4 Further development plans ............................................................................................................ 41  
   3.2 The HTTP Traffic widget bundle ...................................................................................................... 42  
      3.2.1 Learning context .............................................................................................................................. 42  
      3.2.2 Targeted FIRE testbed .................................................................................................................... 44  
      3.2.3 Implementation ............................................................................................................................... 44  
      3.2.4 Further development plans ............................................................................................................ 44  

4 Conclusions and next steps .................................................................................................................. 46  

APPENDICES ........................................................................................................................................... 47  
Appendix A – Acronyms list ..................................................................................................................... 47
List of Figures

Figure 1: The learner experience provided by LTI 2.0 enabled FORGE widgets. 9
Figure 2: PT Anywhere’s architectural diagram. 11
Figure 3: Current look and feel of Packet Tracer widget’s initial page. 12
Figure 4: Landing page used for initiating a session 13
Figure 5: Error message showing that no instances are available. 13
Figure 6: Information about the resources currently being used by the widget. 14
Figure 7: Devices are connected in a two-step process. In the first one (a), we draw a line between two devices and then we select two free ports to complete the connection (b). 14
Figure 8: Pinging a device using the command line of a PC emulated by PT Anywhere. 15
Figure 9: Creation of a new switch drag and dropping it into the network map. 15
Figure 10: Editing a device’s properties through a jQuery UI based dialog. 16
Figure 11: Conceptual diagram of a sample installation with three PT instances. 18
Figure 12: PT Anywhere’s conceptual blocks. 19
Figure 13: Screenshot of a Learning Locker installation that stores PT Anywhere interactions. 22
Figure 14: Webpage used to create the chart that summarises the user interactions recorded so far in the different sessions initiated. 23
Figure 15: The updated architecture of the ssh2web FORGEBox widget 24
Figure 16: xAPI statements recorded by the ssh2web FORGEBox widget 25
Figure 17: The non-interactive version of the WLAN widget bundle. 27
Figure 18: Message about the availability of the widget. 27
Figure 19: Displaying the next available slots. 28
Figure 20: Initiating the provisioning of resources. 28
Figure 21: Provisioning of resources is in progress. 28
Figure 22: Provisioning of resources is complete. 29
Figure 23: The WLAN widget bundle reservation form. 29
Figure 24: Feedback on reserving multiple instances, calculating the maximum available number. 30
Figure 25: Showing the unavailable nodes when requesting specific nodes. 30
Figure 26: Unobtrusive notification of availability when requesting specific nodes. 30
Figure 27: The updated architecture of the WLAN widget bundle. 31
Figure 28: The Learning Locker dashboard displaying data recorded by the WLAN widget bundle. 32
Figure 29: A WLAN widget featuring touch responsive sliders 33
Figure 30: USRP Ceiling grid at CTVR/TCD. 35
Figure 31: TCD Course deployment architecture. 36
Figure 32: Widget radio controllers. 37
Figure 33: The constellation diagram representing the signal modulated by the OFDM transmitter following up a digital modulation scheme, such as BPSK or 64-QAM.------------------------ 38

Figure 34: The preamble used for synchronization, as it is generated at the transmitter. ------ 38

Figure 35: The entire frame generated by the OFDM transmitter. ----------------------------- 39

Figure 36: The complex plot representing the subcarriers that constitute the entire packet. ---- 39

Figure 37: The preamble at the receiver after the effects of noise and attenuation. --------- 40

Figure 38: The frame detection window, showing the percentage of the packet that was successfully received.--------------------------------------------------------------------------------------------- 40

Figure 39: The preamble at the receiver after being equalized. ----------------------------- 41

Figure 40: The rotation location where the radar beam is pointing to, plus the location of Secondary Users of the radar band. --------------------------------------------------------------- 41

Figure 41: A provisioning message for TCD testbed and courses. -------------------------- 42

Figure 42: The map widget. --------------------------------------------------------------- 43

Figure 43: The HTTP traffic widget. ------------------------------------------------------ 43

Figure 44: The Wireshark widget prototype.----------------------------------------------- 44
Executive Summary

This deliverable presents the updated set of educational widgets that have been developed by WP3 for the prototype lab courses of the project. The FORGE widgets make use of the FIRE testbed APIs that have been made available via the work carried out in WP2. The FORGE widgets are developed based on the learning contexts and requirements derived from the FORGE courses, as these are defined in WP5. Additionally, WP4 provides the methodologies and technologies used to develop and deploy the FORGE widgets via the FORGE infrastructure.

The FORGE widgets offer FIRE testbeds as educational facilities to learners and educators via simple and easy-to-use web interfaces. As a result of this approach, the FORGE widgets can be accessed on the Web via a Learning Management System (LMS), via FORGEBBox, FORGESTore, or any other web page, as well as via the FORGE interactive eBook, which is available in the iBooks format for the iPad and in the ePUB format for all other tablets.

The FORGE widgets can be used in the following learning scenarios:

- In-classroom with the assistance of tutors
- Individually for remote learning
- As self-assessment exercises following traditional lectures

A major new development for the FORGE widgets is the deployment of Learning Analytics. More specifically, the FORGE widgets record user interactions and express them with the use of a standard API, thus ensuring interoperability with Learning Analytics tools and Learning Record Stores for storing, analysing and visualising the collected data.

The remainder of this deliverable is structured as follows. First, an updated overview of the technologies associated with the development of the FORGE widgets is provided. In particular, we present the latest FORGE widget infrastructure, as well as an overview of the adopted Learning Analytics approach. Subsequently, we present in detail the updated and new FORGE widgets. The updated widgets are described with a focus on their new features and further development plans. For the new widgets, we introduce their learning contexts, the FIRE testbeds that they target, their implementation details, as well as the plans for their further development. Finally, the deliverable is concluded and the next steps of this work are discussed.
## 1 Overview of technologies

The following table summarises the technologies associated with the development and deployment of the FORGE widgets:

<table>
<thead>
<tr>
<th>Technology</th>
<th>Description</th>
<th>Usage in FORGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript</td>
<td>A programming language allowing client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed.</td>
<td>Used in the development of FORGE widgets for the Web and eBooks.</td>
</tr>
<tr>
<td>Hypertext Transfer Protocol (HTTP)</td>
<td>An application protocol for distributed, collaborative, hypermedia information systems.</td>
<td>Used as the primary interaction protocol between FORGE widgets and FIRE adapters.</td>
</tr>
<tr>
<td>Extensible Markup Language (XML)</td>
<td>A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.</td>
<td>Used in the communication (requests and responses) between FORGE widgets and FIRE adapters.</td>
</tr>
<tr>
<td>JavaScript Object Notation (JSON)</td>
<td>An open standard format that uses human-readable text to transmit data objects consisting of attribute-value pairs.</td>
<td>Used in the communication (requests and responses) between FORGE widgets and FIRE adapters.</td>
</tr>
<tr>
<td>Resource Specification (RSpec)</td>
<td>A behaviour-driven development framework for the Ruby programming language</td>
<td>Used by FIRE adapters for reservation and/or experimentation descriptions.</td>
</tr>
<tr>
<td>Control and Management Framework (OMF)</td>
<td>A framework for controlling, instrumenting and managing FIRE testbeds.</td>
<td>Used by FIRE adapters for reservation and/or experimentation descriptions.</td>
</tr>
<tr>
<td>iBooks</td>
<td>A proprietary format for eBooks specifically for the iPad and Mac OS.</td>
<td>Used for the deployment and delivery of FORGE widgets and learning materials.</td>
</tr>
<tr>
<td>ePUB</td>
<td>An open eBook standard developed by the International Digital Publishing Forum (IDPF).</td>
<td>Used for the deployment and delivery of FORGE widgets and learning materials.</td>
</tr>
<tr>
<td>Learning</td>
<td>An online software application offering</td>
<td>Used for the deployment and</td>
</tr>
</tbody>
</table>
Management System (LMS) | facilities for student registration, enrolment into courses, delivery of learning materials to students, student assessment and progress monitoring. | delivery of FORGE widgets and learning materials.
---|---|---
Learning Tools Interoperability (LTI) | A specification that standardizes the APIs between LMSs and external tools, enabling external tools to function as if they were native tools inside the LMS. | Used for integrating and using FORGE widgets within a variety of LMSs.
Tin Can API / Experience API (xAPI) | A specification for learning technology that makes it possible to collect data about the wide range of experiences a person has (online and offline). | Used for recording learners' interactions with the FORGE widgets.
Learning Record Store (LRS) | A data store serving as a repository for learning records. | Used for storing the recorded learners' interactions with the FORGE widgets.
Learning Locker | A conformant, open source, Learning Record Store; a type of data repository designed to store learning activity statements generated by xAPI (Tin Can) compliant learning activities. | Used for storing and visualising the recorded learners' interactions with the FORGE widgets.

1.1 Updated widget infrastructure

As described in detail in D4.1.1 (FORGE architecture: Introducing FIRE to the learning community), widgets provide access to remote facilities via web containers and web services. The FORGE architecture defines a widget layer offered by the FORGEBox infrastructure. FORGEBox tries to provide a seamless experience while learners are performing a course, reading content and interacting with facilities. Thus, to enhance reusability, to exploit the best integration with LMSs/VLEs and to enable Learning studies, we have extended the FORGE architecture as explained in the next paragraphs.

In general, it is advised for widget providers to adopt the LTI 2.0 standard, if possible. This enables the smooth integration of widgets with existing LMSs but also a seamless interaction for the end users (learners/Instructors). Thus, widgets will be primarily implemented as HTML/JavaScript bundles. Widgets will automatically be downloaded on the student's device and can then request and retrieve content via specific testbed adapters in a remote FORGEBox instantiation. The primary interaction protocol between widgets and adapters is envisioned to be simple Hypertext Transfer Protocol (HTTP) requests / responses with Extensible Markup Language (XML) or a JavaScript Object Notation (JSON) payload. As adapters are being addressed via these HTTP requests, they typically require some web server scripting. These will typically interact subsequently via command scripts with the specific API of the FIRE resources, possibly using some predefined reservation and/or experimentation descriptions such as Resource Specification (RSpec) and OMF. The adapters are thus responsible for translating widget requests to FIRE APIs and vice versa.
Figure 1 depicts the whole seamless learner experience provided by LTI 2.0 enabled widgets. All user activity on course content and interactive widgets is reported in a Learning Record Store (LRS) via the standard xAPI (TinCan API), more details of which are given in the following section about Learning Analytics.

For developers of widgets and their backend service we propose a widget reference architecture, which presents some basic ingredients that enable a widget with the capabilities of remote lab resources interaction, LTI integration and analytics reporting. This architecture and implementation details, as well as LRS deployment and usage are described in detail in D4.1.2 (FORGE architecture: Introducing FIRE to the learning community – update).

1.2 Learning Analytics

Learning Analytics is the “measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs”.¹ The field of Learning Analytics is essentially a “bricolage field, incorporating methods and techniques from a broad range of feeder fields: social network analysis (SNA), machine learning, statistics, intelligent tutors, learning sciences, and others”.² Learning Analytics applies techniques from information science, sociology, psychology, statistics, machine learning, and data mining to analyse data collected during education administration and services, teaching, and learning. Learning Analytics creates applications that directly influence educational practice.³

---

¹ 1st International Conference on Learning Analytics and Knowledge – LAK 2011 https://tekri.athabascau.ca/analytics/

2015 © Copyright lies with the respective authors and their institutions.
With Learning Analytics in FORGE it is possible to obtain valuable information about how learners interact with the FORGE courseware, in addition to their own judgments provided via questionnaires. Learning Analytics in FORGE is based on tracking learner activities, which consist of interactions between a subject (learner), an object (FORGE learning activity) and is bounded with a verb (action performed).

The FORGE widgets have been extended to support the tracking of learning activities by recording the interactions of learners. In particular, the FORGE widgets record user interactions and express them with the use of the Tin Can API, also known as the Experience API (xAPI). The xAPI provides a framework for making statements like “someone does an action to/with something”, for example “John has initiated an experiment”. It offers an extensible vocabulary of:

- **Verbs**, which describe the action performed during the learning experience, e.g. “answered”, “asked”, “interacted”.
- **Activities**: they describe something with which an Actor interacted, e.g. “course”, “question”, “simulation”.

These statements are then stored and visualised via the Learning Locker. Learning Locker is the reference open source Learning Record Store (LRS). It offers a data repository designed to store learning activity statements generated by xAPI compliant learning activities. Among its features is the ability to visualise data via dashboards, its scalability, the ability to produce custom reports, its RESTful API, as well as the ability to export data as CSV and JSON.

More details on the deployment of Learning Analytics on individual widgets are provided in the widget descriptions in the following sections. These details concern the types of learner activities recorded by each widget, depending on the specific functionalities offered by each widget.

---

5. [http://adlnet.gov/expapi/activities](http://adlnet.gov/expapi/activities)
6. [http://learninglocker.net](http://learninglocker.net)
2 Updated widgets

2.1 The PT Anywhere widget
Packet Tracer (PT) is a network simulator that helps learners to experiment with the network behaviour in a safe environment. It is a graphical application running on top of Windows and Linux systems. PT also has an Android version available.7

The PT Anywhere8 widget, rather than replicate PT’s functionality in a web application, focuses on offering its basic functionality from a minimalistic GUI that can be adapted to what different exercises in a course might require. For that, it uses an HTTP API made as part of the FORGE project to manipulate the PT instances that support it.

2.1.1 Widget architecture
Figure 2 shows PT Anywhere’s architecture. Its core is a web application implemented using the Jersey framework. Jersey is the reference implementation of the Java specification for RESTful Web Services. This makes our web application deployable in most of the existing Java application servers.

---

8 http://pt-anywhere.kmi.open.ac.uk
9 https://jersey.java.net/
One thing to note is that although the widget is provided as part of the same web application that contains the rest of the components, it does not depend on any them. That is, it interacts with the rest of the application solely through its APIs. Therefore, it can be easily served in a completely separate web application and/or machine. In fact, this architecture allows having many widgets implemented using different web frameworks and deployed in different machines.

The web application has the following responsibilities:

- **Handle sessions.** To this end, it uses a Redis database that assigns actual PT instances running in other machines. The information stored in the database is minimal: the machine assignment, an expiration time and the available PT instances. These instances are defined in a property file, which is read during the web application deployment phase.

- **Record user interaction.** As will be explained below, the widget stores user interaction details in the Learning Record Store (LRS) through the Java Tin Can API\(^\text{10}\).

- **Communicate with real PT instances.** The web application uses a java library, which communicates with PT using the IPC (Inter Process Communication) protocol.

### 2.1.2 New features

The widget is written in HTML and JavaScript, and consists solely of the user interface code. It uses jQuery\(^\text{11}\), jQuery UI\(^\text{12}\) (for the menus, dialogs and general graphical components) and vis.js\(^\text{13}\) (for the network map creation) libraries.

![Figure 3: Current look and feel of Packet Tracer widget's initial page.](https://github.com/RusticiSoftware/TinCanJava)

---

\(^{10}\) [https://github.com/RusticiSoftware/TinCanJava](https://github.com/RusticiSoftware/TinCanJava)

\(^{11}\) [https://jquery.com](https://jquery.com)

\(^{12}\) [https://jqueryui.com](https://jqueryui.com)

\(^{13}\) [http://visjs.org](http://visjs.org)
During the last year, the following features have been added to the widget:

1. **Session management.** As the following section will explain in more detail, the API now supports different sessions that use different Packet Tracer instances. The widget has been adapted to this feature through two web pages. The first one is a landing page where the session can be initiated (see Figure 4). This session creation implies the reservation of one of the PT instances running on the background over a given period (5 minutes). If no PT instances are available, the widget shows an error message (see Figure 5).

![Figure 4: Landing page used for initiating a session](image)

![Figure 5: Error message showing that no instances are available.](image)

The second web page supports the administration of sessions. As shown in Figure 6, the page lists all the current sessions allowing the user to visit or delete them (i.e. de-allocate their assigned PT instances). After this list, we can see another list containing all the PT instances used by the widget.
2. **Link devices.** The widget allows the user to connect different devices by drawing a line from one endpoint to the other (see Figure 7).

![Diagram of device connection process](image)

**Figure 7:** Devices are connected in a two-step process. In the first one (a), we draw a line between two devices and then we select two free ports to complete the connection (b).

3. **Console.** One of the most interesting features that Packet Tracer offers are the command lines used to configure devices. These command lines emulate those that can be found in physical switches, routers or normal computers. In the widget, we can access the command line (see Figure 8) by double clicking on any device.
Furthermore, during the last year a lot of focus has been put in making the user interaction less menu dependent. For example, a drag and drop device creation is offered (see Figure 9) and the links are created by drawing a line between two devices. Furthermore, the existing graphical components have been redesigned and standardised using jQuery UI (see Figure 10). The PT Anywhere icons match those used by the PT software.

![Figure 8: Pinging a device using the command line of a PC emulated by PT Anywhere.](image)

![Figure 9: Creation of a new switch drag and dropping it into the network map.](image)
The HTTP API hides the complexity of handling PT instances to the Widget. The Widget translates each action of the user to calls to HTTP resources. In particular, the API offers the following resources in a web application available under the “/api” path:

- `/sessions`: to create or list sessions
- `/sessions/{sessionId}`: to delete existing sessions
- `/sessions/{s}/links/{linkId}`: returns the two ports that a link connects in the network map.
- `/sessions/{s}/devices`: to create or list devices
- `/sessions/{s}/devices/{deviceId}`: to manage a particular device
- `/sessions/{s}/devices/{d}/ports`: lists all ports of a specific device
- `/sessions/{s}/devices/{d}/ports/{portId}`: shows the details of a specific port
- `/sessions/{s}/devices/{d}/ports/{p}/link`: connects one port with another one
- `/sessions/{s}/network`: gathers data from devices in a topology and their links to load the network map at the beginning of a session.

Note that the widget and the API must be as far as possible since the widget might change with time. Furthermore, third parties might be interested in creating their own widgets with different GUI controls. To this end, the widget is not expected to build any URL by itself (i.e., should be driven solely by the content returned by the API or be hypermedia-driven).

To enable this, each resource returns responses in JSON format that contain URLs of related resources. For example, each device offers the URLs to the resources that describe its network interfaces. Complementarily, we also provide these links in the responses’ HTTP headers. Providing URLs to the linked resource, clients (e.g., the widget) can avoid hardcoded and
constructing their URLs based on fixed patterns. This has the following positive side effects: the API can evolve without breaking any client; it becomes self-descriptive (a user can learn how to use it simply by following URLs) and is closer to be hypermedia-driven.

**Websockets API**

The command line interface shown in Figure 8 demands bidirectional asynchronous communication between the widget and the web application. For this, we have used WebSockets\(^{14}\).

In the “/endpoint” path we can find a websocket endpoint for each of the command lines available (one per device in each current session). Using these endpoints we can asynchronously transmit any output to be printed in the command line as well as transmit any command to Packet Tracer.

- /sessions/{s}/devices/{d}/console

**Automatic provisioning**

To ease the replication of the environment, a core strategy for the development of this widget has been to create provisioning and installation scripts for each of the components created:

- PT installation\(^{15}\): this project is made up of two Vagrant and Ansible scripts. Ansible\(^{16}\) is an automation tool which allows to define the steps needed to provision machines. Our Ansible-based script installs the web application and the PT instances (see Figure 11) in Red Hat Enterprise Linux 7 or CentOS 7 systems.

Vagrant\(^{17}\) is a piece of software that facilitates the creation and configuration of virtual development environments. Our Vagrant script defines the requirements for virtual machines where the web application or the PT instances can be deployed. Then, it uses the Ansible script described before to provision them. Although we do not use Vagrant to manage virtual machines in FORGE’s production servers, it helps to flexibly and easily create equivalent development environments.

- Learning Locker installation\(^{18}\): The Packet Tracer widget logs all the events coming from the user interaction in an LRS handled by a Learning Locker installation. To ease this installation, we created several Vagrant and Ansible scripts. These scripts have been publicly available to the Learning Locker community\(^{19}\).

- Maven\(^{20}\): The web application project is “mavenized” to enable its building in practically any operating system with Java and Maven installed. Furthermore, it has a task that can be used to automatically deploy the application in a Tomcat server after building it. Most of the web application’s dependencies are handled by Maven (i.e., they are automatically downloaded from

---


\(^{15}\) [https://github.com/gomezgoiri/pt-installation](https://github.com/gomezgoiri/pt-installation)

\(^{16}\) [http://www.ansible.com](http://www.ansible.com)

\(^{17}\) [https://www.vagrantup.com](https://www.vagrantup.com)

\(^{18}\) [https://github.com/gomezgoiri/learninglocker-centos7](https://github.com/gomezgoiri/learninglocker-centos7)

\(^{19}\) [http://docs.learninglocker.net/installation/](http://docs.learninglocker.net/installation/)

\(^{20}\) [https://maven.apache.org](https://maven.apache.org)
Maven repositories). The only two exceptions are the Packet Tracer Java library (which has not been made available due to intellectual property reasons) and the TinCan API java library (which is expected to be available in public Maven repositories soon).

Using these scripts we can create and configure different installations, which follow the architecture shown in Figure 11.

![Figure 11: Conceptual diagram of a sample installation with three PT instances.](image)

**Learning Analytics**

The Packet Tracer widget is a JavaScript application, which consumes data using an HTTP API. This API provides access to several Packet Tracer instances running in the backend (e.g., handles sessions) and translates the actions to calls to PT's internal protocol. Therefore, it provides a good level of abstraction to capture user interactions.

Consequently, we record anonymous user interactions for the calls that the API receives. The HTTP API is provided as part of a Java web application and therefore we use the Java Tin Can API\(^\text{21}\) to record these interactions. Figure 12 shows where the LRS conceptually resides in the PT Anywhere widget's architecture.

\(^{21}\) [https://github.com/RusticiSoftware/TinCanJava](https://github.com/RusticiSoftware/TinCanJava)
So far, we record the following interactions from users by reusing an existing vocabulary:\footnote{https://registry.tincanapi.com}:

- **Session initiation.** We use the verb "http://adlnet.gov/expapi/verbs/initialized".

```json
{
   "version": "1.0.0",
   "actor": {
      "objectType": "Agent",
      "account": {
         "homePage": "http://forge.kmi.open.ac.uk/pt/widget",
         "name": "anonymous_t62CD6SATquEbfKZiWPP7w--"
      }
   },
   "verb": {
      "id": "http://adlnet.gov/expapi/verbs/initialized"
   },
   "object": {
      "objectType": "Activity",
      "id": "http://ict-forge.eu/widget/packerTracer"
   },
   "context": {
      "registration": "b7ad820f-a480-4eab-846d-f2998963ccef"
   },
   "authority": {
      "objectType": "Agent",
      "name": "New Client",
      "mbox": "mailto:hello@learninglocker.net"
   },
   "stored": "2015-06-12T09:42:27.625200+00:00",
   "timestamp": "2015-06-12T09:42:27.625200+00:00",
   "id": "af75c7c8-6353-44f4-95f1-34e98dc4b08e"
}
```

**Listing 1:** Sample record describing an anonymous user initiating a session.
● **Device creation, update and removal.** We use the verbs "create", "delete" and "update" from "http://activitystrea.ms/schema/1.0/".

```json
{
   "version": "1.0.0",
   "actor": {
      "objectType": "Agent",
      "account": {
         "homePage": "http://forge.kmi.open.ac.uk/pt/widget",
         "name": "anonymous_t62CD6SATquEbfKZiWPP7w--"
      }
   },
   "verb": {
      "id": "http://activitystrea.ms/schema/1.0/delete"
   },
   "object": {
      "objectType": "Activity",
      "id": "http://localhost:8080/webPacketTracer/api/sessions/t62CD6SATquEbfKZiWPP7w--/devices/eM282_4tTUWi2AoZfkmqiVQ--/",
      "definition": {
         "name": {
            "en-GB": "PC5"
         },
      }
   },
   "context": {
      "registration": "b7ad820f-a480-4eab-846d-f2998963c6ef"
   },
   "authority": {
      "objectType": "Agent",
      "name": "New Client",
      "mbox": "mailto:hello@learninglocker.net"
   },
   "stored": "2015-06-12T09:42:36.145600+00:00",
   "timestamp": "2015-06-12T09:42:36.145600+00:00",
   "id": "1c01c321-3f3e-4e06-8480-728d5a6943d"
}
```

Listing 2: Sample record describing an anonymous user deleting a PC in the PT Anywhere widget.

● **Link creation and removal** (i.e. connecting and disconnecting two devices). The link creation and removal is expressed as a user creating a link which has its two endpoints defined as contextual information. Another alternative could have been to use non-existing connect/disconnect verbs to express that a user connects a device to another one (the latter should have been added as contextual information). However, we chose the first alternative because it reuses already existing verbs.
As a first step to help us analyse how students use the widget, we have created a chart that visualizes the different steps carried out by users in their sessions (see Figure 14). This chart is fed by the statements stored in the Learning Locker LRS and allows specifying the different levels to visualise (i.e., the number of session steps or actions to be displayed).

Figure 14 shows the first four steps of the different sessions. The different states for each step (i.e., level) are the device creation (ADD), removal (DEL), update (UPD), connection (CONN) or disconnection (DISCONN). Additionally a NOOP state is used to represent the lack of action in

---

Listing 3: Sample record describing an anonymous user deleting a PC in the PT Anywhere widget.
sessions with fewer actions recorded than levels shown. The main limitation of this chart is that the final state of our widget (i.e. if a student passed or failed the exercise) cannot be discerned yet. This is a future feature that will be included as soon as the widget is divided in widget instances with specific goals.

Figure 13: Screenshot of a Learning Locker installation that stores PT Anywhere interactions.
2.1.3 Further development plans

Future steps of the PT Anywhere widget development include:

**Identifying users.** Right now, all users are anonymous. This complicates the identification of trends and patterns for the Learning Analytics module.

**Capture further user interactions.** For instance, when the user modifies a device’s networking interface settings (e.g., its IP address).

**Handling PT instances.** In the future, the application should be able to create and destroy instances on demand instead of reusing fixed running PT instances. Furthermore, a machine should be able to run multiple instances and we should perform some stress tests over the application to clearly identify the computing requirements of the PT instances. This is vital since the widget will be potentially used by more than a million of students.

**Customizing the GUI.** Each exercise in a course will be focused in a really specific aspect. Consequently, disabling some graphical components might be desirable in some cases in order to avoid distractions. For example, the device creation menu can be hidden in exercises where the network topology should not be changed. In other cases, the GUI will be extended to capture functionalities not yet covered or to adapt the user interface based on the feedback provided by real students.

**Use of Learning Analytics.** We need to analyse the information captured by the widget in order to identify students with problems and proactively help them before they fail the next associated exercise or the course.
Interoperability with LMS. The widget will implement the LTI 2.0 API to facilitate its integration with existing LMSs.

2.2 The ssh2web FORGEBox widget

Bringing ssh and terminal access to the web browser was a high requirement early in the project as arose from the initial courses’ requirements. The ssh2web widget as presented in D3.1.1 (LMS and eBooks widgets for prototype lab courses) offers web-based SSH technology. Web-based SSH makes it possible to access SSH servers through web clients that are based on JavaScript/Ajax or JavaScript/WebSockets technologies.

2.2.1 New features

Learning Analytics

In the ssh2web widget we implemented and integrated xAPI capabilities in order to enable the widget for Learning Analytics. Figure 15 displays our new architecture.

![Figure 15: The updated architecture of the ssh2web FORGEBox widget](image)

As the figure shows, the ssh2web widget is now able to report user activity to an LRS. Currently we are using the LRS integrated in FORGEBox. Our implementation exploits the JavaScript implementation offered by the TinCAN API\(^{23}\).

To make the widget reusable and available to be used by multiple courses in parallel, developers can integrate the widget into their page by passing the following parameters to the calling url:

● **xendpoint:** the endpoint of the xAPI LRS, for example: 
  http://www.forgebox.eu/lrs/learninglocker/public/data/xAPI/

● **xapiauth:** the authorization keys in Base64 format as provided by the LRS

● **actoremail:** the email of the user being tracked

● **actorname:** the name of the user being tracked

ssh2web uses two verbs:

● **Experienced** (id: "http://adlnet.gov/expapi/verbs/experienced") with type: "http://adlnet.gov/expapi/activities/interaction". This is used when a user visits a page.

● **Interacted** (id: "http://adlnet.gov/expapi/verbs/interacted") with type: "http://adlnet.gov/expapi/activities/interaction". This is used whenever the user interacts with the widget and the remote resource (i.e. type and send commands). Figure 16 displays an example of xAPI statements from the widget stored in the FORGEBox LRS.

For reference, this is an example TinCan statement for Experience. This function is reused from several views of the widget:

```json
actor: {
  name: actorname,
  mbox: "mailto:"+actoremail
},

verb: {
  display: {"en-US": "experienced"}
},

object: {
  id: "http://www.forgebox.eu:8080/ssh2web/"+pagename,
  definition: {
    type: "http://adlnet.gov/expapi/activities/interaction",
    name: { "en-US": "ssh2web widget "+pagename }
  }
}
```
2.2.2 Further development plans
Further developments include the integration of the LTI 2.0 API in order to make the widget capable of tight integration with existing LMSs supporting the LTI standard. Additionally, general improvements will be made to both the teacher/admin side as well as the learner experience.

2.3 The WLAN widget bundle
This widget bundle was developed as a one-stop solution for controlling predefined experiments and visualizing the results in an intuitive manner. It has been further expanded during the project to include better UI elements, support Learning Analytics and allow the dynamic provisioning of FIRE resources. The WLAN widget bundle has also been reused in a proof of concept LTE course without requiring modifications to the source code.

2.3.1 New features
Dynamic provisioning and reservation
To solve the problem of scarce resources we have deployed an architecture that allows an always-on course with graceful degradation to a non-interactive version of the course when there are no wireless nodes available or reserved to perform the interactive experiments.

To achieve this goal we have set up an Apache load balancer\(^\text{24}\) on the landing page of our course\(^\text{25}\). Apache was selected as a load balancer because of existing experience, allowing us to perform these complex operations using software that was already in use as a web server. This load balancer uses the concept of a hot-standby to serve the non-interactive content when no interactive instances are available as a fallback mechanism. This fallback content is hosted locally on our portal web server.

Having this fallback content also allows us to seamlessly provide our interactive widgets to the Web or eBooks and create an extension that allows the dynamic provisioning of interactive resources on a FIRE facility based on live demand. An example of the widget bundle when served with the non-interactive version is given in Figure 17.

\(^{24}\) [http://httpd.apache.org/docs/2.2/mod/mod_proxy_balancer.html](http://httpd.apache.org/docs/2.2/mod/mod_proxy_balancer.html)

In this non-interactive version of the WLAN widget bundle the start button of the OMF Experiment Control widget is disabled and a notification is presented to the user to inform them on the availability of the experimentation nodes.

This provisioning widget checks the state of the wireless nodes required to perform the interactive exercises since these are the scarce resources. Depending on the current and future w-iLab.t reservations the user is presented with the choice to reserve and provision three wireless nodes or is notified that the entire testbed is fully booked.

Either way, the next time slots when the interactive exercises will be available are also available through the widget.
If a user requests an interactive version of the course by clicking the button “Start provisioning” the widget will keep informing the user on the provisioning process, but there is no obligation to keep the session open. Once initiated, the course portal will make sure the provisioning process is completed and afterwards the load balancer will redirect any new sessions to the interactive course.

Once initiated, the provisioning process goes through three different states:

1. The wireless nodes are reserved on w-iLab.t and the portal schedules the start of the provisioning process. The user is informed that the provisioning will start soon. This delay is necessary since it can take up to one minute for the reservations to be processed by w-iLab.t

2. The provisioning process is initiated and in total 4 machines are provisioned by two separate processes:
   - 3 specific wireless nodes on w-iLab.t in a fixed topology, to serve as wireless experimentation nodes
   - 1 Experiment Controller on the Virtual Wall, taking care of serving the interactive content of the course, controlling the wireless nodes and collecting the experiment results.

This process is controlled by the jFed tool, which is an ideal candidate since it features a command line interface that allows us to automate the provisioning process and parse the output of the tool. The logs of the provisioning process are parsed by the widget and used to inform the user of the provisioning progress.

---

26 http://jfed.iminds.be/
3. Once the provisioning process finishes, the user is informed that the interactive content is ready. All new sessions to the portal are automatically redirected by the load balancer to the newly provisioned Experiment Controller. Clicking the link in the notification, reloading the page or reopening the widget takes the user to the interactive course.

![Figure 22: Provisioning of resources is complete.](image)

This mechanism of dynamic provisioning is presented to all users visiting our course, but we have also extended this mechanism for teachers and course assistants. For authorized users a separate reservation page is available, allowing more control over the reservation and provisioning process (see Figure 23).

![Figure 23: The WLAN widget bundle reservation form.](image)
This page, as shown in the figure, allows the free selection of a timeslot at any time in the future with available resources. Up to 13 different instances of the interactive course can be scheduled simultaneously, that are automatically configured to avoid interference on the wireless spectrum. These instances can be manually defined by entering the node IDs of the required machines, or automatically assigned by an algorithm that checks a list of known well performing topologies for availability.

Finally, the page features a calendar of scheduled experiments on the w-iLab.t testbed, but always validates the availability of the reservation automatically. The user is presented with the results of this validation through a notification if there are problems with availability, or by changing the button colour to green and enabling the “Reserve” button when all requested nodes are available.

The following figures show the different states of the reservation buttons and possible notifications.

![Figure 24](image1.png)

**Figure 24: Feedback on reserving multiple instances, calculating the maximum available number.**

![Figure 25](image2.png)

**Figure 25: Showing the unavailable nodes when requesting specific nodes.**

![Figure 26](image3.png)

**Figure 26: Unobtrusive notification of availability when requesting specific nodes.**
Learning Analytics

In preparation of the first large scale labs using this widget we have updated the architecture of the widget to include the xAPI for learning analytics. In the updated architecture diagram shown in Figure 27 the new components and interactions are highlighted in green.

![Updated Architecture Diagram]

We have chosen to use an in-house developed PHP implementation of the relevant subset of the xAPI, based on the previous experiences of iMinds using this API. This minimal implementation, tincanlib.php, allows easy extensions and manipulation of the xAPI json object. By moving the xAPI communication to the backend we can securely store the credentials of the LRS and tie in the learning analytics in the backend PHP code without changing the frontend logic and JavaScript.

Currently, three different types of xAPI statements are used:

1. **Initialized** ("http://adlnet.gov/expapi/verbs/initialized"): Formally indicates the beginning of analytics tracking, triggered by a student “viewing” a page or widget. Since this is triggered without any interactions with the widget it is handled by a new PHP script logview.php, which is called by experimentcontrol.js at load time. It contains the (anonymised) user ID and the exercise/widget that was initialized.

2. **Interacted** ("http://adlnet.gov/expapi/verbs/interacted"): Triggered by startexp.php when an experiment is started by the user, containing the user ID, the exercise and possible parameters chosen by the learner. These parameters are stored in serialized json form using the result object, as defined by the xAPI.

3. **Completed** ("http://adlnet.gov/expapi/verbs/completed"): The final verb, signalling completion of an exercise by the learner. It is triggered by stopexp.php so it is closely tied to the actual experimentation. Thanks to this coupling we can also include the duration that a learner took to perform the experiment, formatted using the ISO 8601 duration syntax following the xAPI specifications.
These statements are collected in the Learning Locker LRS. Learning Locker features a simple but effective dashboard (see Figure 28), giving a quick overview of the activities over time, the most active users and activities, which are a direct mapping of the exercises in the iMinds course.

![Learning Locker Dashboard](image)

**Figure 28: The Learning Locker dashboard displaying data recorded by the WLAN widget bundle.**

**Touch responsive sliders**

The primary control objects in most instances of this widget are easy to use sliders, which allow a learner to modify numerical parameters of the experiment (see Figure 29). Since they play a very important role in the user experience we have reworked the sliders to be responsive to touch events on mobile devices, independent of how the widget is viewed (on the Web or inside an eBook). Therefore, we have switched from a custom implementation to the much more powerful jQuery UI slider\(^\text{27}\) in combination with a simple but effective touch extension\(^\text{28}\) and restyled it to fit our look and feel.

By switching to jQuery UI the configuration of the sliders has been simplified and responsiveness of the layout has improved. Additionally there is a direct coupling between the text input fields and the sliders, updating the linked component live when the other is altered.

---

\(^{27}\) https://jqueryui.com/slider/

\(^{28}\) http://touchpunch.furf.com/
2.3.2 Further development plans

Next steps include the parameterization of the widget through query parameters for passing learning analytics metadata. These parameters will be the same or a subset of the ssh2web widget (xendpoint, xapiauth, actoremail, actorname).

The LTI 2.0 API will also be evaluated to see if support can be added to the widget bundle, with the primary focus on user authentication.
3 New widgets

3.1 The Blacksmith Widget Framework and Anvil Engine

3.1.1 Learning context
The Blacksmith framework was developed with the goal of providing lightweight real-time web-based plotting for users running experiments using software defined radios (SDR), in particular for experimenting with Iris\(^\text{29}\), which is the flagship SDR framework used at Trinity College Dublin. Blacksmith is used to enhance the data visualization and facilitate the learning process of the students using the CTVR/TCD Smart and Reconfigurable Radio Testbed. It can be used in any FORGE course to provide the visualization needed for the experiments, as long as it communicates with Iris.

Currently, the Blacksmith framework is providing the experiment result visualization for two advanced courses that TCD has deployed within the FORGE context:

- OFDM wireless communications and signalling
- Dynamic Spectrum Access systems

Both courses are advanced regarding the specialized radio equipment that is required, and also with regards to the previous knowledge required and the learning outcomes within the physical properties of the wireless signals and spectrum access.

The framework provides real-time plots, through a web interface, of the physical properties of the wireless signals. We have already applied to our traditional in-lab laboratory sessions as well as remote laboratories performed in different international locations.

3.1.2 Targeted FIRE testbed
Our testbed possesses 16 Ettus USRP N210s (software defined radio hardware) fixed in the ceiling, forming a grid. Each USRP contains a SBX daughterboard, which provides full-duplex communication capabilities from 400 to 4400 MHz. The large number of SDRs and their wide spectrum range allows many different experiments running simultaneously in different bands. SDR's re-configurability grants flexible and dynamic algorithms to be able to sense and react to the wireless medium accordingly. Figure 30 shows the USRP configuration in the grid at the CTVR’s lab ceiling.

\(^{29}\) [https://github.com/softwareradiosystems](https://github.com/softwareradiosystems)
3.1.3 Implementation

The Blacksmith framework dynamically manages different Anvil Engines, which are in charge of managing the resources and processes for plotting, one different Anvil per experiment type. Each Anvil creates the respective Sparks for its own experiment type. Sparks are the plot handlers, they use the Bokeh interactive visualization library for generating real time plots in modern web browsers that support HTML5. The data input for each Anvil Engine is handled by the Hammer module, which is a TCP socket demultiplexer that receives data for several plots in a single port and redirects the data to its respective plot. The networking features used in the Blacksmith Framework are provided by Twisted, an event-driven networking library. The input data must be formatted to allow the demultiplexing and proper treatment at the Sparks, this task is carried out using Protobuffers, a mechanism for serializing structured data.

Dependencies:

- Bokeh30
- Twisted31
- Protobuffers32

30 http://bokeh.pydata.org
31 https://twistedmatrix.com
32 https://developers.google.com/protocol-buffers
Figure 31 presents the Blacksmith and Anvil integration within TCD’s course deployment architecture. As depicted, Blacksmith listens through a port to the incoming connection carrying signal data from the radio enabled experimentation units. Within Blacksmith, an Anvil instance per experimenter is launched to isolate and process the data for each user and then plot the dynamic results, through widgets, in each experimenter’s session and browser.

Within the TCD course’s framework, we have deployed two types of widgets, called Controllers and Displayers. Controllers are HTML5 widgets that are capable to modify the behaviour of our radio systems, such as configuring the Frequency, Bandwidth, Power Transmission, Cycle Prefix, Sensing Period, among others. Figure 32 shows snapshots of these controllers.
Regarding the widget Displayers, at TCD, we have deployed more than 10 types of widget radio displayers. Studying radio signalling is complex; it requires understanding the behaviour of the wireless signals in time and frequency domain, the wireless impairments, the particularities of the coding scheme and PHY properties. Without going into extensive detail, we describe some of the displayer widgets as shown in the following figures.

Figure 32: Widget radio controllers.
Figure 33: The constellation diagram representing the signal modulated by the OFDM transmitter following up a digital modulation scheme, such as BPSK or 64-QAM.

Figure 34: The preamble used for synchronization, as it is generated at the transmitter.
Figure 35: The entire frame generated by the OFDM transmitter.

Figure 36: The complex plot representing the subcarriers that constitute the entire packet.
Figure 37: The preamble at the receiver after the effects of noise and attenuation.

Figure 38: The frame detection window, showing the percentage of the packet that was successfully received.
3.1.4 Further development plans

As it stands now, Blacksmith works and copes with all the current requirements. Nonetheless, some other features would allow extra flexibility, such as:

- **Support for XML or JSON**: Expanding the range of input types we can process, only an extra type, not both probably.
- **Use of the Curse CLI**: Allow real-time debugging and resource management.

![Image of radar rotation](image-url)
• **Creation of a Data Comprehension Module**: a middle module between the Hammer demultiplexer and the Spark plot handler for making data operations before the plots

• **Provisioning**: we plan to enhance our current provisioning system by adding a reservation mechanism in order to develop automatic provisioning, since currently these processes are semi-automated. Figure 41 shows a snapshot of the provisioning message within TCD courses.

• **LTI 2.0 functionality**: we will study the feasibility of implementing LTI 2.0 functionalities to the Blacksmith framework, perhaps through the Anvil engine.

![Figure 41: A provisioning message for TCD testbed and courses.](image)

### 3.2 The HTTP Traffic widget bundle

The HTTP Traffic widget bundle consists of three widgets that provide a graphical web interface to visualise network traces in different geographical locations.

#### 3.2.1 Learning context

UPMC networking labs are mainly based on network traffic analysis. The HTTP Traffic widget bundle was developed to allow long distance measurement. It enables students to experiment by themselves with the capture of real life network traffic and its analysis. This experiment was previously carried out on pre-recorded files and was unsatisfactory for the students.

This widget bundle has been used in the UPMC prototype course entitled “TCP Congestion Control” and focuses on the impact of geographic distance on HTTP traffic. It allows the students to dissect the packets generated on a FIRE testbed via an interface similar to the well-known software Wireshark that allows remote analysis. It also allows the download of the capture file for a local analysis with common libpcap tools, such as tcpdump, Wireshark, etc.

---

33 [https://www.wireshark.org](https://www.wireshark.org)
Figure 42: The map widget.

The map widget (see Figure 42) shows the generation of HTTP traffic in different geographical locations. The student can choose any of the 3 locations - local, continental and intercontinental nodes – in order to execute his experiment (see Figure 43).

![Map Widget](image)

Figure 43: The HTTP traffic widget.

The map and HTTP traffic widgets are complemented by the Wireshark widget (see Figure 44). This widget provides a graphical interface where the student can visualise the different protocols involved in network transmission. It provides three main panels, as illustrated in Figure 44. The first (top) panel lists the packet with some details. The second (middle) panel provides the list of protocols involved in this packet and details several fields. The third (bottom) panel is used to show a hexadecimal version of the packet.
3.2.2 Targeted FIRE testbed

This widget is used in a course designed for the PlanetLab Europe testbed. PlanetLab Europe is the European portion of the publicly available PlanetLab testbed, a global facility for the deployment of new network services. In the future, this widget could also be used in various courses involving other FIRE testbeds.

3.2.3 Implementation

This widget is developed in JavaScript, using some additional libraries such as jQuery and Datatables. The idea is to maintain it as pure JavaScript, in order to allow compatibility with most modern browsers.

The packet description file is based on libpcap, which is a library used for network traces capture. This library is used by several software applications and provides a common way to save network traces into file.

3.2.4 Further development plans

The initial plan was to get full support of the HTTP protocol so that the student could use it in the UPMC courses. The widget now handles IPv4 and IPv6 connectivity, as well as the TCP stream.

---

34 http://www.datatables.net
35 http://www.tcpdump.org
The widget has been successfully integrated into FORGEBox and the FORGE eBook (ePUB and iBook formats). It is also accessible via a simple web browser.

The current goal is to add Learning Analytics on top of the widget using xAPI. This will allow us to better understand the behaviour of students that are using the widget. For this purpose, the course that will be taught in the next semester will be delivered via FORGEBox, using the FORGEBox infrastructure to record Learning Analytics data. Part of the course may also be delivered using the iBook format on iPads.

In the future, the widget can be extended to support other protocols too. This will allow the widget to be used in other courses involving different network protocols. This widget could also be used in various courses involving other FIRE testbeds.
4 Conclusions and next steps

This deliverable presents both the updated and new educational widgets that have been developed by the FORGE partners in the context of WP3. This deliverable focuses on the technological aspects associated with the development and deployment of the FORGE widgets. In particular, we present the new features of existing FORGE widgets, as well as the newly developed FORGE widgets by introducing their learning contexts, the FIRE testbeds that they target, their implementation details, and the plans for their further development.

The FORGE widgets expose FIRE educational facilities to learners and educators via a web interface, which can be accessed on the Web (e.g. via a Learning Management System) or alternatively via an interactive eBook. They can be used both in formal and informal education settings, i.e. inside and outside of a traditional classroom, plus as self-study materials without the help of a tutor. One significant new development for the FORGE widgets is the deployment of Learning Analytics, achieved by recording, storing and visualising the interactions of learners with the FORGE widgets.

Regarding the next steps for this work, we are working on the deployment of Learning Analytics across all FORGE widgets, supported by the FORGE infrastructure (FORGEBox and FORGEStore). More details about this work on an architectural level will be provided in D4.1.2 (FORGE architecture: Introducing FIRE to the learning community – update).

Additionally, we are continuously refining the specifications of the FORGE widgets based on the updated educational requirements of the prototype courses being developed by WP5. Based on the refined specifications, we will continue updating the implemented widgets and develop additional widgets in order to address new educational requirements.

Finally, we have initiated a series of Open Calls by inviting external organisations to join the project as associate partners and develop learning artefacts using the FORGE infrastructure. This initiative opens up the FORGE infrastructure outside of the project boundaries and will enable external contributions in the form of new learning scenarios, new widgets, new courses and learning materials, as well as evaluations of these widgets and courses with students. A number of external partners, including the GÉANT Association (Netherlands), Universidade de Brasília (Brazil), Universitat Politècnica de València (Spain) and Inria Paris-Rocquencourt (France) have already responded to the first round of the FORGE Open Call and have joined the project as associate partners. The results of this initiative will be described in D6.8 (Opening the FORGE platform and engaging the community) due M36.

36 http://ict-forge.eu/opencall/
APPENDICES

Appendix A – Acronyms list

API – Application Programming Interface
CSS – Cascading Style Sheets
FIRE – Future Internet Research and Experimentation
GUI – Graphical User Interface
HTML – HyperText Markup Language
HTTP – Hypertext Transfer Protocol
IP – Internet Protocol
JSON – JavaScript Object Notation
LMS – Learning Management System
LRS – Learning Record Store
LTI – Learning Tools Interoperability
OEDL – OMF Experiment Description Language
OMF – cOntrol and Management Framework
OML – OMF Measurement Library
RSpec – Resource Specification
SCORM – Sharable Content Object Reference Model
TCP – Transmission Control Protocol
UI – User Interface
WLAN – Wireless Local-Area Network
xAPI – Experience API
XML – Extensible Markup Language