



## Widening Access to Virtual Educational Scenarios

562463-EPP-1-2015-1-UK-EPPKA2-KA

### Deliverable 2.3 Developed VP system APIs and platform integration

Deliverable number D2.3

Delivery date December, 2018

Status Final

Authors Martin Adler (Instruct), Panagiotis Antoniou (AUTH),  
Sheetal Kavia (SGUL), Martin Komenda (MU), Andrzej  
Kononowicz (KI), Daniel Schwarz (MU), Dimistris Spachos  
(AUTH), Natalia Stathakarou (KI), David Topps (OL),  
Corey Wirun (OL), Luke Woodham (SGUL)



Funded by the  
Erasmus+ Programme  
of the European Union

## TABLE OF CONTENTS

<b>1. INTRODUCTION.....</b>	<b>3</b>
<b>2. LTI.....</b>	<b>3</b>
2.1. CASUS compatibility checks for LTI.....	4
2.2. OpenLabyrinth notes on LTI.....	6
<b>3. XAPI.....</b>	<b>8</b>
3.1. State of the xAPI support in VS systems.....	8
3.1.1. Introduction.....	8
3.1.2. CASUS.....	8
3.1.3. OpenLabyrinth.....	10
3.1.4. xAPI support.....	11
3.2. Testing interoperability of generated xAPI statements.....	14
3.2.1. Aim.....	14
3.2.2. Methods.....	14
3.2.3. Results.....	17
3.2.4. Discussion.....	19
3.2.5. Conclusions.....	19
3.3. Integrating xAPI in the mobile player.....	21
3.4. Educators access to session data of learner in delivery platform.....	22
<b>4. MICROSERVICES.....</b>	<b>22</b>
4.1. Display score of a played VP, into Moodle, for a specific student.....	22
4.2. Moving OpenLabyrinth case to the mobile player.....	23
4.3. Integration of Vispath with OpenLabyrinth.....	24
<b>5. BEST PRACTICES, REFINEMENTS AND CHANGES BASED ON EVALUATION AND MONITORING.....</b>	<b>26</b>
5.1. Basic description of results of evaluation and monitoring.....	26
5.2. Best practices, refinements and changes.....	27
<b>6. CONCLUSIONS.....</b>	<b>27</b>
<b>7. REFERENCES.....</b>	<b>28</b>

## 1. INTRODUCTION

The WAVES project aims to facilitate the proliferation of Scenario Based Learning (SBL) through three outcomes: Community Engagement, Technical Facilitation and Knowledge Support. This is achieved through technical improvements in SBL systems for ease of use, tapping into the academic and business community, and facilitating quality and easily accessible educational content.

A core part of the project's technical outcomes was a revisiting of the state of the art in SBL platform integration tools and APIs in order to identify areas of improvement and to prepare the grounds for establishing guidelines where such improvements would not be provided by the project activities themselves. Given that, the goal of this deliverable is to report on developed VP system APIs and platform integration tools of the WAVES project and to also report the conclusions and guidelines that have emerged from evaluating these solutions in conjunction with existing ones.

Currently, Moodle, CASUS and OpenLabyrinth are the platforms to be addressed. The Technical Reference Group (TRG) of the project had access to these platforms, hence all technical work and consideration has been conducted on them. However, the scope of the work in this document is not limited to these specific systems, but expands into the xAPI [1] standards themselves, as well as the learning record store (LRS) technologies field. Although we used some specific LRSs for our testing, we generalize the approach in this document, so the tools, instructions and guidelines can be easily adopted by any LRS. In some sections, results from WP5 were used to indicate the connections between our evaluation tests and given instructions and best practices.

The target audience for the document is not limited to technologists and includes: learning management system administrators, educators, learning analytics experts and others. The key goal for the document is to provide a general description of the developed system APIs, the results and guidance for platform integration in addressed platforms, tools and technologies and finally, how all these can be expanded in order to fit the specific needs for each institution, educator expert, learner and user.

This document is organized in seven sections with the first being this introduction and the seventh being the bibliography. Sections 2, 3 and 4 present the work conducted from the TRG of the WAVES consortium regarding LTI xAPI and micro services improvements. They are the core technical activities conducted during the project's activities. As such, both technical implementations and suggestions for platform vendors are described. Section 5 summarizes the outcomes from the evaluation endeavour, specifically from the technical evaluation segment. Finally, Section 6 presents the overall conclusions regarding these activities.

## 2. LTI

Learning Tools Interoperability (LTI) [5] seamlessly connects systems to enhance the online learning experience. Accessing Virtual Scenarios through targeted delivery platforms such as

## D2.3 Developed VP system APIs and platform integration

MOOCs (FutureLearn, Open edX) or learning management systems (Moodle, Canvas) is the basic use case behind the development of the LTI specification. LTI allows the seamless connection of web-based, externally hosted applications and content or tools to platforms that present them to the end users. Under this perspective we analyse some basic aspects of LTI integration, mainly for the CASUS and OpenLabyrinth platforms.

More information about the LTI standard can be found in Deliverable 2.4 “Exemplar implementations – OL3 and CASUS’ of the WAVES project.

### 2.1. CASUS compatibility checks for LTI

In Nov. 2018 the CASUS team performed conformance tests against some LTI testing tools found online. We decided not to go through the official and paid certification process which can be found on <https://www.imsglobal.org/lti-certification-suite>, this could be done on request. It was agreed that this was unnecessary for the purpose of this work, as general compliance could be established without this process.

Therefore the following unofficial test suites were found and checked:

- <https://foliotek.github.io/LTI-Tester/>
  - check passed ok. Intuitive test tool, custom parameters can be passed

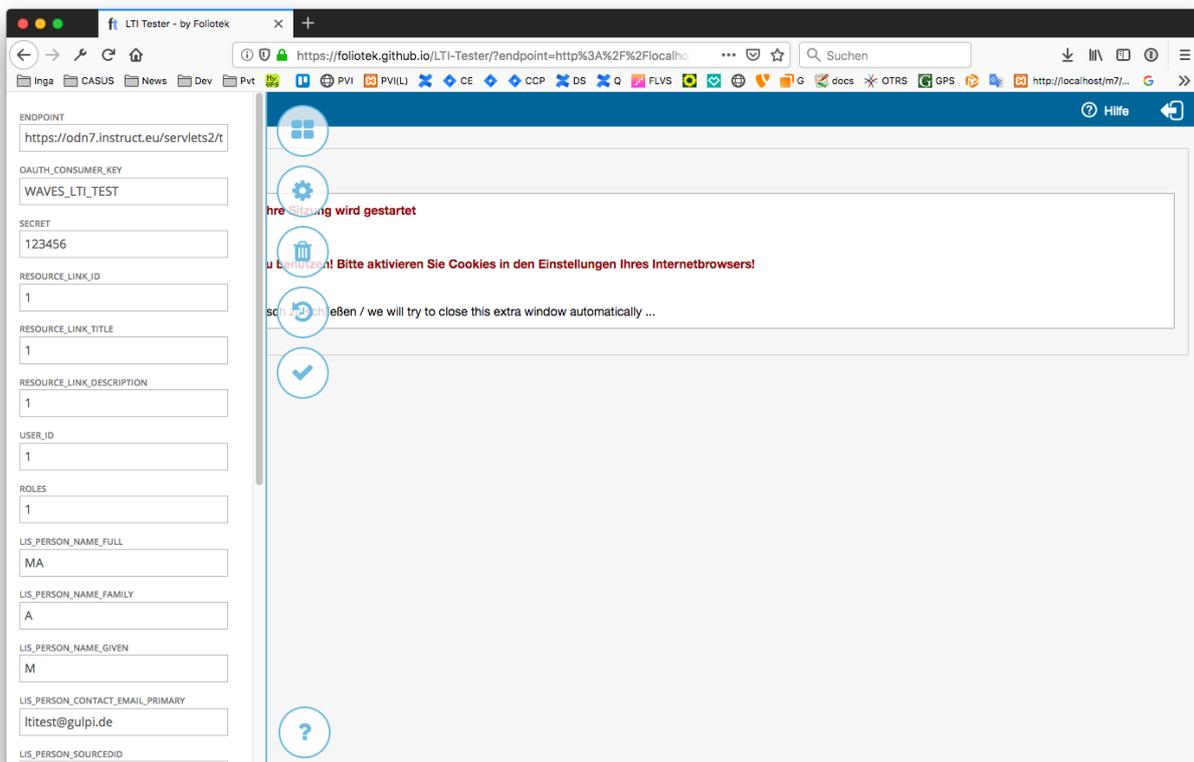


Figure 1: [foliotek.github.io/LTI-Tester](https://foliotek.github.io/LTI-Tester/) User Interface for checking LTI requests

- <https://www.tsugi.org/lti-test/>

## D2.3 Developed VP system APIs and platform integration

- check passed ok. Found that tool was less intuitive, no clear place for passing custom parameters, used query in target URL

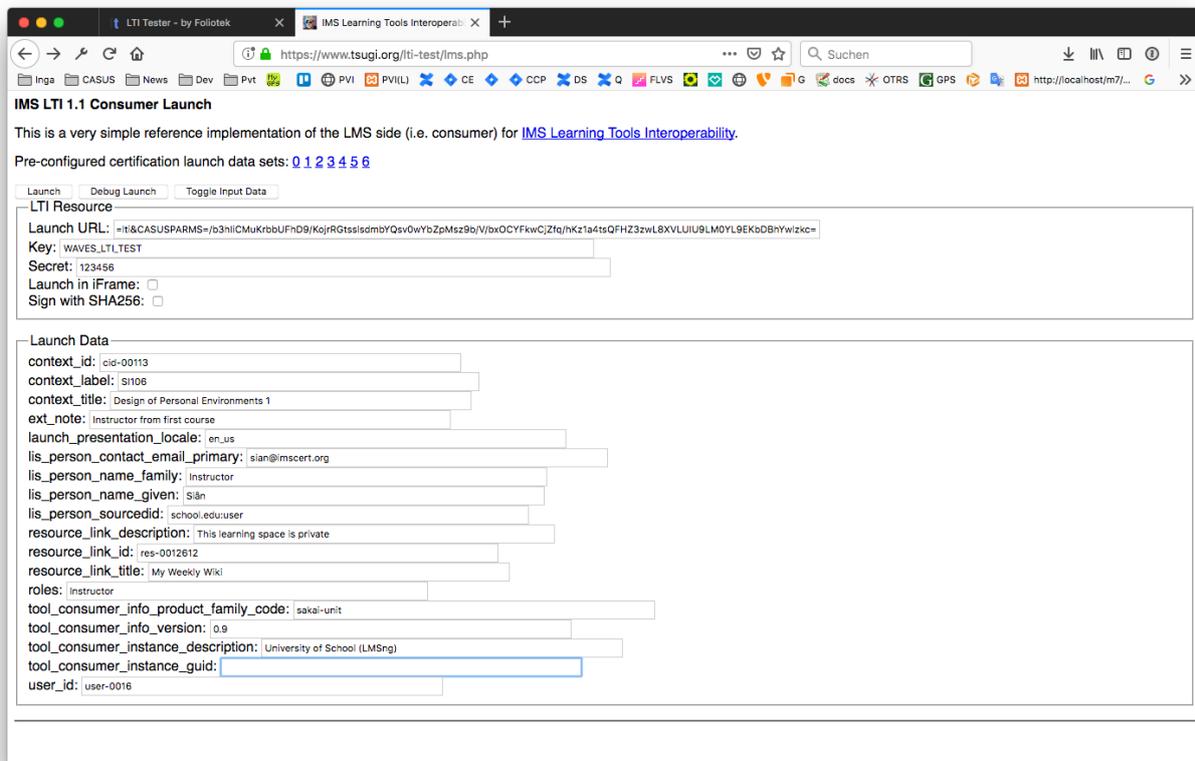


Figure 2: [www.tsugi.org](https://www.tsugi.org) User Interface for checking LTI requests

- <https://lti.tools/test/tc.php>
  - check passed ok. Quite similar to <https://foliotek.github.io/LTI-Tester/>, extra option for passing custom parameters

## D2.3 Developed VP system APIs and platform integration

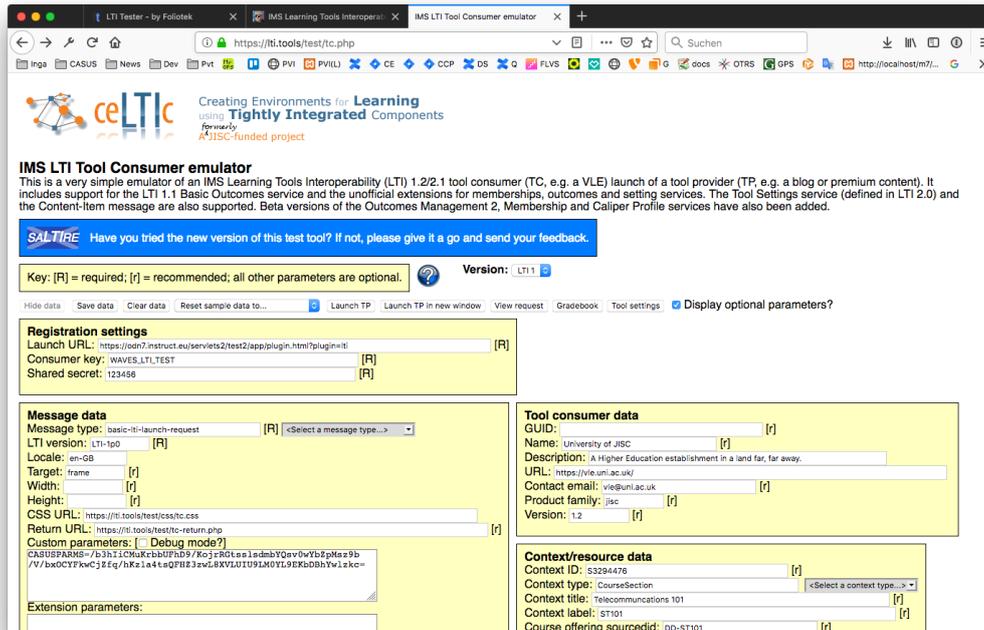


Figure 3: lit.tools User interface for checking LTI requests

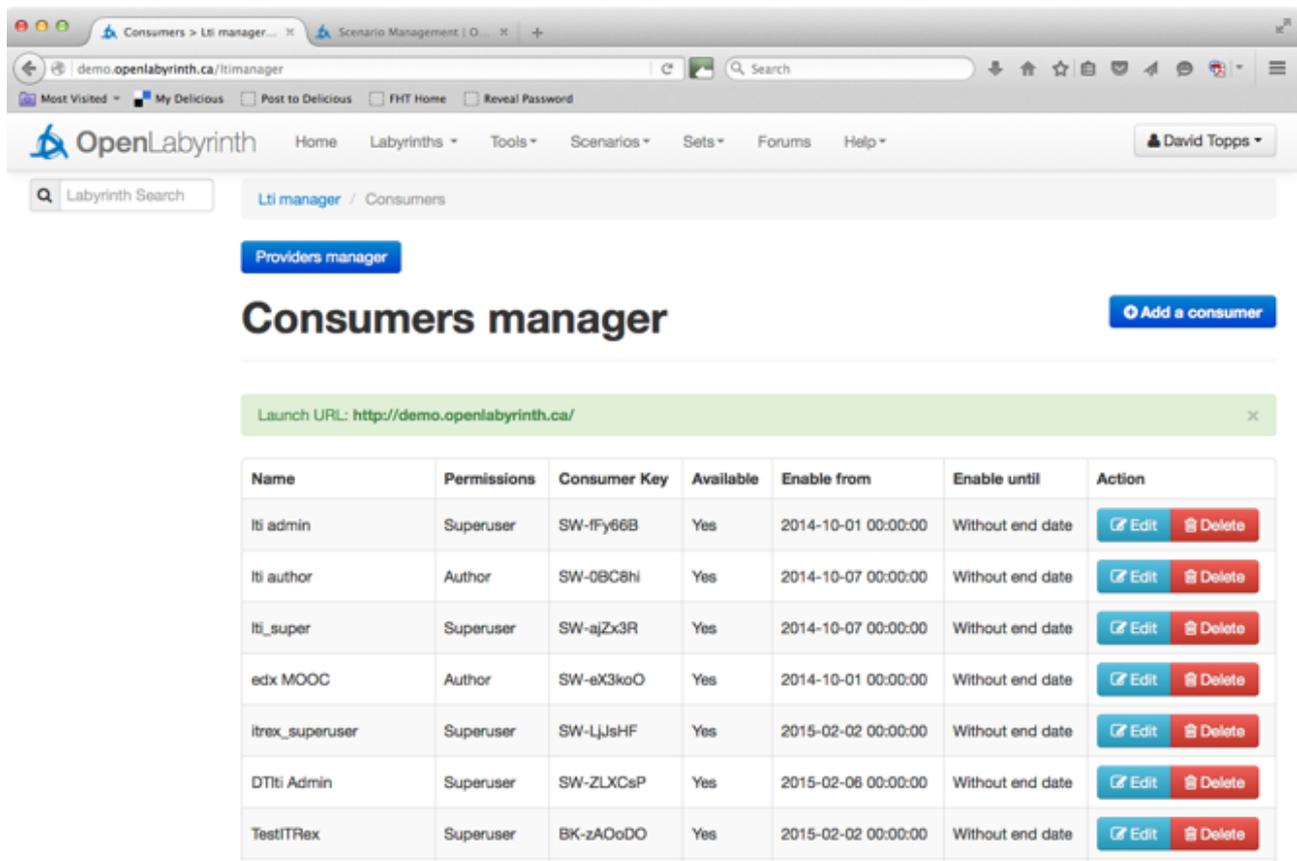
By passing 3 independent LTI testing suites, as well as prior implementations in production use with a couple of universities having been live for one year, the CASUS team is confident that we implemented the LTI standard in CASUS properly. The additional official paid certification process need only be done when clients need more security, and costs charged accordingly.

## 2.2. OpenLabyrinth notes on LTI

OpenLabyrinth provides extensive support for LTI. We described integrations of the OpenLabyrinth application with Open edX and FutureLearn (with detailed, step-by-step instruction for configuration of the LTI tools) in the Deliverable 2.4 “Exemplar implementations – OL3 and CASUS”.

References of integration with other systems and experiences exist also in the official OpenLabyrinth website, inside the discussion forums and blog posts, such as the “Setting Up Desire2learn LTI Authentication To OpenLabyrinth” [3]:

## D2.3 Developed VP system APIs and platform integration



The screenshot displays the 'Consumers manager' interface in OpenLabyrinth. At the top, there is a navigation bar with 'Providers manager' and 'Add a consumer' buttons. Below this is the 'Consumers manager' title and a search bar. A green bar indicates the 'Launch URL: http://demo.openlabyrinth.ca/'. The main content is a table with the following data:

Name	Permissions	Consumer Key	Available	Enable from	Enable until	Action
lti_admin	Superuser	SW-fY66B	Yes	2014-10-01 00:00:00	Without end date	<a href="#">Edit</a> <a href="#">Delete</a>
lti_author	Author	SW-0BC8hi	Yes	2014-10-07 00:00:00	Without end date	<a href="#">Edit</a> <a href="#">Delete</a>
lti_super	Superuser	SW-ajZx3R	Yes	2014-10-07 00:00:00	Without end date	<a href="#">Edit</a> <a href="#">Delete</a>
edx MOOC	Author	SW-eX3koO	Yes	2014-10-01 00:00:00	Without end date	<a href="#">Edit</a> <a href="#">Delete</a>
ltrex_superuser	Superuser	SW-LJJsHF	Yes	2015-02-02 00:00:00	Without end date	<a href="#">Edit</a> <a href="#">Delete</a>
DTlti Admin	Superuser	SW-ZLXCSP	Yes	2015-02-06 00:00:00	Without end date	<a href="#">Edit</a> <a href="#">Delete</a>
TestITRex	Superuser	BK-zAOoDO	Yes	2015-02-02 00:00:00	Without end date	<a href="#">Edit</a> <a href="#">Delete</a>

*Figure 4: Creating an LTI Consumer in OpenLabyrinth*

Under WP 5 we performed some user testing regarding the LTI integration with institutional applications (mainly learning management systems). The interpretation of the detailed results indicates that there are some crucial key points to achieve a successful integration:

1. Compatibility between applications at a technical level, e.g. it's not possible to connect through LTI two web apps that served under different protocols, (https vs https).
2. Compatibility between system versions: Some institutes may use older versions of applications (e.g. the AUTH School of Medicine uses Moodle 1.9). User testing there for LTI compatibility was not completed, because no LTI consumer was available at time of testing.
3. Documentation and help: It is crucial that learning technologists and other key staff receive suitable guidance and documentation in order to complete LTI integration. Taking that under consideration we have prepared the guidelines in the WP 3.

### 3. XAPI

#### 3.1. State of the xAPI support in VS systems

##### 3.1.1. Introduction

XAPI or Tin Can API was founded in 2010 as an extension to SCORM, in a project named “Tin Can”. In 2015 the project name was changed to Experience API (XAPI) and allowed storing of usage information coming from content management or learning management systems in one central place, called the learner record store (LRS). LRS can be completely separate from other systems or could be integrated into existing platforms. The WAVES technical reference group agreed that this technology showed most promise for future widespread adoption, as it addresses a lot of issues of collecting usage data. Even though with the introduction of the EU-GDPR in spring 2018 universities in Europe tend to be more cautious in introducing such central storage systems, the xAPI architecture allows for anonymous storage for non-personal dashboards. Widespread adoption is anticipated to take a couple more years, requiring new projects on digital transformation at institutions for implementation. For business collection of usage data is mostly already done, but xAPI might change architecture and storage, so xAPI may be also interesting for integration projects, even if the data is already stored and analyzed.

In spring 2018 OpenLabyrinth 3 already had support for xAPI built in. Instruct implemented basic xAPI support by spring 2018, tests were performed and as a next step the created statements of both systems were compared. We will present the status of implementations of xAPI in OpenLabyrinth and CASUS.

##### 3.1.2. CASUS

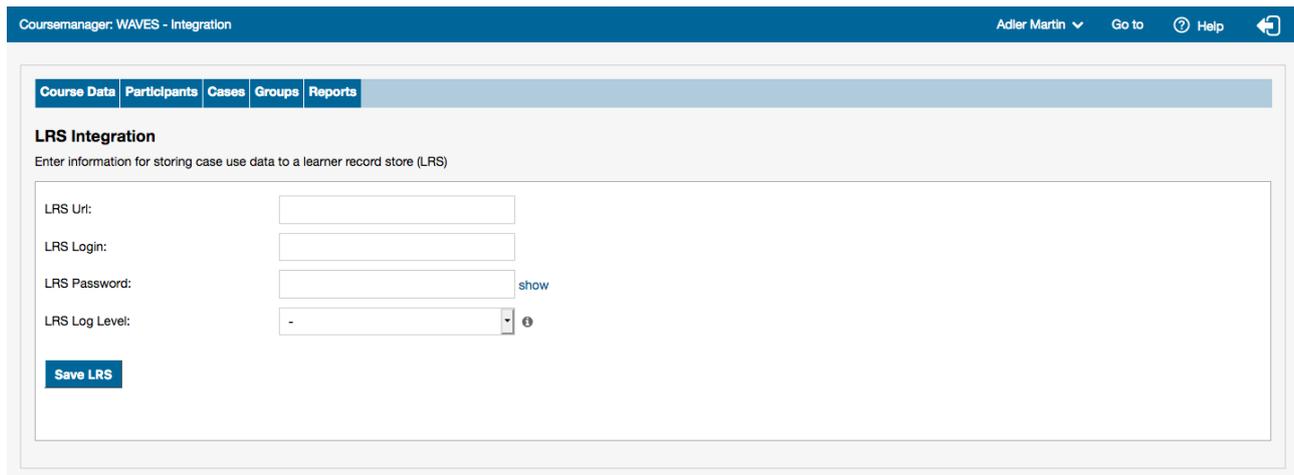
In CASUS [6] support for xAPI was realized using a Google library for xAPI in 2017 and refined 2018. It can be activated either globally for a client for all clients courses, or individually on a course level. To activate the feature only the URL of the LRS plus credentials are needed. Additionally the level of transmitted statements can be selected. At this moment 3 levels are defined:

Case level: only basic statements are transmitted, like open, restart, finish.

Card level: additionally to case level, for each visited card statements are created

Question answer level: additionally to card level statements submitting answers on questions are submitted including contextual data of the question and items.

## D2.3 Developed VP system APIs and platform integration



Coursemanager: WAVES - Integration Adler Martin ▾ Go to Help ↻

**Course Data** | Participants | Cases | Groups | Reports

**LRS Integration**  
Enter information for storing case use data to a learner record store (LRS)

LRS Url:

LRS Login:

LRS Password:  show

LRS Log Level:  ⓘ

Figure 5: CASUS Coursemanager Settings for xAPI LRS

Url of the group:

LRS Url:

LRS Login:

LRS Password:  show

LRS Log Level:  ⓘ

Figure 6: CASUS Client/Group Settings for xAPI LRS on Administration Level

During the WAVES project the implementation was tested against a commercial LRS and the basic reference LSR implementation of adlnet.org. Some minor glitches were removed after scanning the comparisons to Open Labyrinth in summer 2018. All mainly used question and answer types are supported. Adding new statements for a finer granularity is not a difficult task, and requires minimal coding experience though.

With integration of the separate clinical reasoning toolbar a set of fine granular statements are supported. The connection to the URL and credentials defined in CASUS on client or course level was not yet done and will be a future task moving the integration of CASUS, CAMPUS and the clinical reasoning toolbar forward.

Activation of xAPI is possible either on client or course level, the interface inserting credentials and an URL of the LRS is quite straightforward, additionally the logging level can be selected as described above.

## D2.3 Developed VP system APIs and platform integration

### 3.1.3. OpenLabyrinth

OpenLabyrinth supports an integrated xAPI activity tracking, since version 3.4 [7]. Activity metrics, using xAPI, are standards-based, and provide an important step towards objectively tracking learning experiences and outcomes. Although, lots of activity data has been tracked within OpenLabyrinth previously – this xAPI support offers a standardised way for tracking activities across multiple platforms and simulation systems.

All of these activities are captured in a Learning Records Store (LRS). OpenLabyrinth was tested [7] with the following LRS database engines:

- GrassBlade LRS – cheap and simple to maintain
- SCORM Cloud – very standards-compliant
- Wax LRS – stricter statement handling; more analytics
- Watershed LRS – seriously powerful with great analytics
- Learning Locker – open-source, based on MongoDB
- SCORM Engine – simplest way to bridge xAPI to SCORM

Users in most cases are able to pull up an OpenLabyrinth Session Report which will show most of tracked stuff - as far as users have the right privileges on OpenLabyrinth (superuser).

Real time reports are also supported. Using real-time xAPI reports can sometimes be very helpful but there are limitations to this. Users should be aware that sending many xAPI statements from a complex case, being played by many concurrent users, will really slow down the response from the OpenLabyrinth server. When creating real-time xAPI reports, OLab will only transmit the following xAPI statements for the following:

- When a case is Initialized or started
- When a Must Visit or Must Avoid node is touched
- When the value of the Main Counter for the case is altered
- When the case is Completed

During the WAVES project the implementation was tested against a commercial LRS (Grassblade), but generally the comparison and state of support results of xAPI support of OpenLabyrinth can be applied to the rest of the LRSs.

Integrating OpenLabyrinth with Grassblade LRS (xAPI)

There is a video with instructions here: <https://www.youtube.com/watch?v=JzpBfMT5Tqw&t=4>

The detailed data analysis and results, as well as some important notes and comments collected during the user testing related with the use of the xAPI standard., can be found on deliverable 5.3 “Data analysis” of the evaluation work-package.

### 3.1.4. xAPI support

The following tables present the verbs that currently are supported by the CASUS and OpenLabyrinth systems.

#### CASUS

Table 1: CASUS xAPI verbs support

Label	Scope Note	Level	ID
<b>launched</b>	VP/VS is opened	case level	<a href="http://adnet.gov/expapi/verbs/launched">http://adnet.gov/expapi/verbs/launched</a>
<b>resumed</b>	VP/VS is re-opened/ continued	case level	<a href="http://adnet.gov/expapi/verbs/resumed">http://adnet.gov/expapi/verbs/resumed</a>
<b>suspended</b>	VP/VS is stopped (learner might return later)	case level	<a href="http://adnet.gov/expapi/verbs/suspended">http://adnet.gov/expapi/verbs/suspended</a>
<b>terminated</b>	VP/VS has been ended	case level	<a href="http://adnet.gov/expapi/verbs/terminated">http://adnet.gov/expapi/verbs/terminated</a>
<b>accessed</b>	Card has been switched	Card level	<a href="http://xapi.vocab.pub/describe/?url=https%3A%2F%2Fw3id.org%2Fexpapi%2Fdod-isd%2Fverbs%2Faccessed&amp;sid=30963">http://xapi.vocab.pub/describe/?url=https%3A%2F%2Fw3id.org%2Fexpapi%2Fdod-isd%2Fverbs%2Faccessed&amp;sid=30963</a>
<b>answered</b>	Question of any type answered in the VP/VS	card level	<a href="http://adnet.gov/expapi/verbs/answered">http://adnet.gov/expapi/verbs/answered</a>
<b>identified</b>	Components of the clinical reasoning process - findings & differentials have been identified	card level	Extension
<b>diagnosed</b>	a final diagnosis has been made	case & card level	Extension
<b>investigated</b>	a test has been ordered/entered to further confirm or rule out a differential diagnosis	card level	Extension

### D2.3 Developed VP system APIs and platform integration

<b>treated</b>	a treatment has been ordered or performed	card level	Extension
<b>reported</b>	A summary statement has been composed	case & card level	Extension
<b>erred</b>	Errors that have been made/experienced during the clinical reasoning process	case & card level	Extension

## Open Labyrinth

Table 2: Open Labyrinth xAPI verbs support

Label	Scope Note	Level	ID
<b>resumed</b>	VP/VS is re-opened/ continued	Case level	<a href="http://adlnet.gov/expapi/verbs/resumed">http://adlnet.gov/expapi/verbs/resumed</a>
<b>suspended</b>	VP/VS is stopped (learner might return later)	Case level	<a href="http://adlnet.gov/expapi/verbs/suspended">http://adlnet.gov/expapi/verbs/suspended</a>
<b>updated</b>	Indicates the actor prompted a change in a data value or information associated with the object. The Virtual Patient player engine has changed a counter value. This may be triggered by arrival at a particular node, or by a rule within the case design created by the virtual patient author, or by a timer expiration point. Although the counter value may be regarded as a score, note that the ADL verb 'scored' is overall score for the case or exam ( <a href="http://adlnet.gov/expapi/verbs/scored">http://adlnet.gov/expapi/verbs/scored</a> ), which is not the same thing.	Case level Node Level	<a href="http://w3id.org/xapi/medbiq/verbs/updated">http://w3id.org/xapi/medbiq/verbs/updated</a>

### D2.3 Developed VP system APIs and platform integration

<b>UpdatedMainCounter</b>			Return xAPI Updated statement
<b>initialized</b>	Indicates the activity provider has determined that the actor successfully started an activity.	Case level ?	<a href="http://adlnet.gov/expapi/verbs/initialized">http://adlnet.gov/expapi/verbs/initialized</a>
<b>completed</b>	Indicates the actor finished or concluded the activity normally	Case level Node level	<a href="http://adlnet.gov/expapi/verbs/completed">http://adlnet.gov/expapi/verbs/completed</a>
<b>arrived</b>	Indicates the actor arrived in a VS node	Node level	<a href="http://w3id.org/xapi/medbiq/verbs/arrived">http://w3id.org/xapi/medbiq/verbs/arrived</a>
<b>launched</b>	VP/VS is opened	Case level	<a href="http://adlnet.gov/expapi/verbs/launched">http://adlnet.gov/expapi/verbs/launched</a>
<b>responded</b>	Indicates an actor reacted or replied to an object	Node level	<a href="http://adlnet.gov/expapi/verbs/responded">http://adlnet.gov/expapi/verbs/responded</a>
<b>answered</b>	Question of any type answered in the VP/VS. The text of the answer will often be included in the response inside result	Node level	<a href="http://adlnet.gov/expapi/verbs/answered">http://adlnet.gov/expapi/verbs/answered</a>
<b>scored</b>	ADL verb 'scored' is overall score for the case or exam	Case level	<a href="http://adlnet.gov/expapi/verbs/scored">http://adlnet.gov/expapi/verbs/scored</a>
<b>consumed</b>	(deprecated?)	Case level Node level	<a href="http://activitystream.ms/schema/1.0/consume">http://activitystream.ms/schema/1.0/consume</a>
<b>progressed</b>	(deprecated?)	Case level Node level	<a href="http://id.tincanapi.com/extension/ending-point">http://id.tincanapi.com/extension/ending-point</a>
<b>voided</b>	A special reserved verb used by a LRS or application to mark a statement as invalid. See the xAPI specification for details on Voided statements	Case level Node level	<a href="http://adlnet.gov/expapi/verbs/voided">http://adlnet.gov/expapi/verbs/voided</a>

From the available verbs of the specification only a subset is used in each system to cover the most important actions of a user within a session, some disagreements exist on the meaning of a couple of words. We will cover this more in details in the comparison chapter 3.2.

## 3.2. Testing interoperability of generated xAPI statements

### 3.2.1. Aim

The goal of this procedure was to test for interoperability of xAPI statements generated after implementation of the xAPI specification in both virtual scenario systems supported in the WAVES project: OpenLabyrinth 3 (OL3) and CASUS. This aimed to contribute in reaching a common agreement on the xAPI profile of virtual scenarios which could give the developers guidance on how to implement the xAPI specification in their virtual scenario systems.

### 3.2.2. Methods

A test virtual scenario has been implemented and transferred with exactly the same content and structure in two virtual scenario systems: OpenLabyrinth version 3 and CASUS. The topic of the scenario was the mechanisms behind hiccups. The test virtual scenario contained mock-up educational content and comprised of 9 cards arranged in a linear path and 26 interactive elements (Table 3).

Table 3. Structure of the test hiccup scenario

N#	Card name	Interactive elements	List of possible activities [parameter]
1	Introduction	<ol style="list-style-type: none"> <li>[Ext. Link] WAVES project <a href="http://wavesnetwork.eu">http://wavesnetwork.eu</a></li> <li>[Ext. Link] The other Waves Project <a href="https://sites.google.com/a/waves-project.eu/the-waves-project">https://sites.google.com/a/waves-project.eu/the-waves-project</a></li> <li>[Int. Link] Start</li> </ol>	<ul style="list-style-type: none"> <li>- Follow external link 1.1</li> <li>- Follow external link 1.2</li> <li>- Progress scenario with link 1.3</li> </ul>
2	START	<ol style="list-style-type: none"> <li>[Emb. Video] <a href="https://youtu.be/1Q-byXeROZ0">https://youtu.be/1Q-byXeROZ0</a></li> <li>[Ext. Link] Most frequently asked health question on Google <a href="http://www.iflscience.com/health-and-medicine/google-trends-reveals-americas-most-searched-health-questions-in-2017/">http://www.iflscience.com/health-and-medicine/google-trends-reveals-americas-most-searched-health-questions-in-2017/</a></li> <li>[Freetext Answer] Answer question "What do you want to know about hiccups?"</li> <li>[Int. Link] About this condition</li> </ol>	<ul style="list-style-type: none"> <li>- Play, stop, rewind video 2.1 [time]</li> <li>- Follow external link 2.2</li> <li>- Submit answer to question 2.3 [free text]</li> <li>- Progress scenario with link 2.4</li> </ul>
3	About this condition	<ol style="list-style-type: none"> <li>[Emb. Video] <a href="https://youtu.be/UTLz-wHH2Pc">https://youtu.be/UTLz-wHH2Pc</a></li> <li>[Emb. Video] <a href="https://youtu.be/Bd5lr1A97I">https://youtu.be/Bd5lr1A97I</a></li> <li>[Underline Answer] In the following statements, choose the words which make the statement correct.</li> <li>[Int. Link] Quick fixes</li> </ol>	<ul style="list-style-type: none"> <li>- Play, stop, rewind video 3.1 [time]</li> <li>- Play, stop, rewind video 3.2 [time]</li> <li>- Submit answer to question 3.3 [list of highlighted words]</li> <li>- Progress scenario with link 3.4</li> </ul>

### D2.3 Developed VP system APIs and platform integration

4	Quick fixes	<ol style="list-style-type: none"> <li>1. [MRQ Answer] Which of these methods have you tried, now or in the past?</li> <li>2. [Int. Link] More difficult cases</li> </ol>	<ul style="list-style-type: none"> <li>- Submit answer to question 4.1 [list of selected/unselected items]</li> <li>- Progress scenario with link 4.2</li> </ul>
5	More difficult cases	<ol style="list-style-type: none"> <li>1. [Ext. Link] longest case of hiccups is 68 years <a href="http://news.bbc.co.uk/2/shared/spl/hi/pop_ups/05/health_guinness_medical_record_breakers/html/2.stm">http://news.bbc.co.uk/2/shared/spl/hi/pop_ups/05/health_guinness_medical_record_breakers/html/2.stm</a></li> <li>2. [Match Answer] See if you can figure out which med is used for which condition.</li> <li>3. [Int. Link] Weird fixes</li> </ol>	<ul style="list-style-type: none"> <li>Follow external link 5.1</li> <li>Submit answer to question 5.2 [list of paired answers]</li> <li>Progress scenario with link 5.3</li> </ul>
6	Weird fixes	<ol style="list-style-type: none"> <li>1-6. [Ext. Link] Acupuncture; Sexual intercourse; Digital rectal massage; Nasal catheter; Breathing pacemakers; Nebulized lidocaine</li> <li>7. [MRQ Answer] Which of these approaches have you ever heard of before, as a possible fix?</li> <li>8. [Int. Link] Information foraging behaviours</li> </ol>	<ul style="list-style-type: none"> <li>Follow external link 6.1-6.6</li> <li>Submit answer to question 6.7 [list of selected/unselected items]</li> <li>Progress scenario with link 6.8</li> </ul>
7	Information foraging behaviours	<ol style="list-style-type: none"> <li>1. [Table answer] Rate, according to how often you use this strategy, or how effective you find it</li> <li>2. [Int. Link] References</li> </ol>	<ul style="list-style-type: none"> <li>Submit answer to question 7.1 [List of selected options]</li> <li>Progress scenario with link 7.2</li> </ul>
8	References	<ol style="list-style-type: none"> <li>1. [Ext. Link] Dwairy on foraging behaviours <a href="https://bmcfampract.biomedcentral.com/articles/10.1186/1471-2296-12-90">https://bmcfampract.biomedcentral.com/articles/10.1186/1471-2296-12-90</a></li> <li>2. [Ext. Link] Google Scholar vs PubMed <a href="https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1471-1842.2012.00992.x">https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1471-1842.2012.00992.x</a></li> <li>3. [Table answer] Rate, according to how often you use this strategy, or how effective you find it</li> <li>4. [Int. Link] FINISH</li> </ol>	<ul style="list-style-type: none"> <li>Follow external link 8.1, 8.2</li> <li>Submit answer to question 8.3 [List of selected options]</li> <li>Progress scenario with link 8.4</li> </ul>
9	FINISH	<ol style="list-style-type: none"> <li>1. [Freetext Answer] What did you think? Please feel free to send us suggestions on how to improve this scenario.</li> <li>2. [Int. Link] End Session and View Feedback</li> </ol>	<ul style="list-style-type: none"> <li>Submit answer to question 9.1 [free text]</li> <li>Progress scenario with link 9.2</li> </ul>

A protocol was agreed to test the xAPI statements generated after going through the scenario. Table 4 presents a list of all 21 activities set to be performed as part of the test procedures. The “Parameters” column specifies the details of the actives - e.g. what time to start or stop the video playback or what answers to selected in the questions from the scenario. The protocol starts when user opens the virtual scenario for the first time.

D2.3 Developed VP system APIs and platform integration

Table 4: Protocol for a test run of the hiccup virtual scenario

#	Activities	Parameters
1.	Follow external link 1.1	
2.	Progress scenario with link 1.3	
3.	Play video 2.1	0:00
4.	Stop video 2.1	0:03
5.	Rewind video 2.1	1:00
6.	Submit answer to question 2.3	"Start of test procedure"
7.	Progress scenario with link 2.4	
8.	Submit answer to question 3.3	"hunger"; "laughter"
9.	Progress scenario with link 3.4	
10.	Submit answer to question 4.1	"Breath holding"=true; "Drinking from other side of glass"=false; "Burping or belching"=true; "Surprise me"=false
11.	Progress scenario with link 4.2	
12.	Submit answer to question 5.2 [list of paired answers]	{{"baclofen", "epilepsy"}, {"chlorpromazine", "peptic ulcers"}; {"gabapentin", "muscle spasms"}; {"metoclopramide", "schizophrenia"}; {"omeprazole", "vomiting"}}
13.	Progress scenario with link 5.3	
14.	Submit answer to question 6.7	"Acupuncture"=false; "Sexual intercourse"=false; "Digital rectal massage"=false; "Nasal catheter"=false; "Breathing pacemakers"=false; "Nebulized lidocaine"=false
15.	Progress scenario with link 6.8	
16.	Submit answer to question 7.1	{{"Ask a friend or colleague directly", "Strongly disagree"}; {"General Google search", "Disagree"}; {"Use a content specific portal", "Neutral"}; {"Ask Twitter or other social media", "Agree"}; {"Ask on a forum or discussion board (AskJeeves, Quora etc)", "Strongly agree"}}
17.	Progress scenario with link 7.2	
18.	Submit answer to question 8.3	{{"Dwairy", "Strongly disagree"}; {"Scholar vs PubMed", "Disagree"}}

## D2.3 Developed VP system APIs and platform integration

19.	Progress scenario with link 8.4	
20.	Submit answer to question 9.1	“End of test procedure”
21.	Progress scenario with link 9.2	

Both systems had their xAPI activity streams registered in the same learning record store, GrassBlade LRS [8], to minimise the influence of the implementation of the LRS on the differences in the outcomes of the implementation.

### 3.2.3. Results

The procedure was successfully performed in both virtual scenario systems. The LRS recorded 19 xAPI statements for OpenLabyrinth version 3 and 32 for CASUS. The Table 5 summarises the recorded statements synchronised with the cards and steps from the protocol.

*Table 5. Protocol for a test run of the hiccup virtual scenario*

Card_id	H_act_no	H_act_dscr	OL_act_n	OL_xapi_verb	C_act_n	C_xapi_verb
case init	1	open scenario			1	resumed
1	2	open link, go next	1	initialized	3	interacted x2, answered
2	5	media start, stop, rewind, answer [freetext], go next	0	-	3	interacted x2, answered
3	2	answer [underline], go next	10	updated, arrived, 2 x attempted, 4 x interacted, 2 x answered	3	interacted x2, answered
4	2	answer [MRQ], go next	1	updated	3	interacted x2, answered
5	2	answer [match], go next	1	updated	3	interacted x2, answered
6	2	answer [MRQ], go next	1	updated	3	interacted x2, answered

### D2.3 Developed VP system APIs and platform integration

7	2	answer [MRQ], go next	2	updated, arrived	3	interacted x2, answered
8	2	answer [MRQ], go next	1	updated	3	interacted x2, answered
9	2	answer [freetext], go next	2	updated, completed	6	interacted x5, answered
case finish	n/a	close scenario			1	terminated

H\_act\_no: number of activities performed by the human in the card from the hiccup case given in the Card\_id column; H\_act\_dscr: natural language description of activities performed by the human in the given card; OL\_act\_n: number of xAPI statements received from OL3 for this step. OL\_xapi\_verb: names of xAPI verb and their quantity received from OL3 in this step; C\_act\_n: number of xAPI statements received from CASUS for this step. C\_xapi\_verb: names of xAPI verb and their quantity received from CASUS in this step.

The following observations regarding xAPI implementation were made for both systems:

- The basic structure of statements in both systems is the same: **id, actor, verb, object, result, context, stored, timestamp, authority** and follows the xAPI specification.
- Both implementations ignore the following of external links by the student
- Both implementations ignore media control
- In OpenLabyrinth v.3 **timestamp** is the time when the action was really made and in CASUS is the same time as in **stored** (time when action was stored – but it's only a few millisecond difference).
- The main difference between the **object** of CASUS and OpenLabyrinth v.3 is that all xAPI objects in OpenLabyrinth v.3 are activities and you can find out what it is when you read the definition of this **object**. On the other hand, in CASUS, objects have three types – *lesson*, *page* and *question* and the *definition* is less complex.
- The **context** is stored in CASUS only in statements with the **verb answered** and it includes only one extension and its answer type. In OpenLabyrinth v.3, all stored records contain **context**, and for all records this **context** almost the same with some context activities and extensions used by the **object**.

The following observations were xAPI implementation was made for OpenLabyrinth v.3:

- All statements are on scenario level. Assignments of statements to cards (nodes) happens by extensions and/or requires human logic
- Operates mainly on extensions to the specification which might be difficult to interpret in a different context.
- Some statements are issued or skipped (e.g. "arrived" statements) while going through the scenario. Might depend on flags set for individual nodes.
- Forwards xAPI statements from H5P objects embedded in VS packages
- OpenLabyrinth v.3 recognises questions only then when implemented as H5P widgets
- In general OpenLabyrinth v.3 does not record answers to questions in the scenario. The only exception is with the **verb answered** is in the activity #8 (Submit answer to question 3.3)
- Events with verbs *attempted*, *interacted* and *answered* were stored duplicated probably to

D2.3 Developed VP system APIs and platform integration  
the way xAPI is implemented by the H5P widgets.

The following observation regarding xAPI implementation were made for CASUS:

- Statements at 4 levels of granularity: scenario, card/node, question, question item
- Probably a systematic bug of issuing twice the same "interacted" statement
- "question\_item" level of reporting is very detailed and might be argued it is implementation specific
- A minor issue in the test procedure of CASUS was that the case was opened already once before the test procedure had started and because of that the "resume" statement was issued instead of "launched"

### 3.2.4. Discussion

In response to the observations made in the test procedure some of the noticed issues in xAPI implementation in the virtual scenario systems were corrected; for instance the CASUS system no longer duplicates xAPI statement with "interacted" verbs.

We have noticed that even though the statements are xAPI compatible and we agree in general on the meaning of the the verbs, alternative verbs are used to implement the same tasks. For instance OpenLabyrinth v.3 describes the start/finish of the case with the verbs "initialized" and "completed", while CASUS uses the verbs "launched" and "terminated". Changing to the next card is signalled from OpenLabyrinth v.3 by the verb "updated", whereas CASUS uses for that "interacted".

Some of the difference may be explained by the differing strategy of constructing virtual scenarios in both systems: OpenLabyrinth v.3 being a branched and CASUS a semi-linear system.

### 3.2.5. Conclusions

Based on these observations we would like to propose a three level model of conformance with the xAPI virtual scenario profile to be considered in setting the xAPI recommendations by MedBiquitous [11]. The levels are arranged to follow an increasing degree of sophistication.

Table 6: Conformance levels and descriptions

Conformance level	Description
Level 1: "Open/close VS"	The virtual scenario is a black box: the important action is when the user enters and leaves the virtual scenario. Compliant virtual scenario systems reach an agreement on what verbs are used for these activities.
Level 2: "Changing cards/nodes"	The virtual scenario issues xAPI statements when the user changes from one node/card/state to the next one (slideshow mode). Compliant virtual scenario systems reach an agreement on what verbs are used for these activities.
Level 3: "Individual"	The virtual scenario reports on levels of individual activities

### D2.3 Developed VP system APIs and platform integration

interactions”	within cards/nodes (e.g. answering questions, controlling media, etc). This could be further divided into sublevels: 3q - agreement in terms of questions, 3m - media control, 3x - support of external standards e.g. H5P. Compliant virtual scenario systems reach an agreement on what verbs are used for these activities.
---------------	--

For conformance Level 1 the WAVES TRG suggests using the following verbs:

*Table 7: Recommended verbs for conformance Level 1<sup>1</sup>*

Action	Required	Verb suggested
Opening a VS	required	initialized or launched
Reopening a VS	required	resumed
Closing a VS completely worked thru	required	completed or terminated  Potentially accompanied by: • scored
Closing a VS temporarily for later finish	required	suspend

This defines the actual VS xAPI profile, and can be extended in future meetings together with other system providers. This is also in sync with the MedBiquitous profile for virtual patients. However, even documentation from the Tin Can API website and Medbiquitous profile still interpret verbs inconsistently at present so future discussions and decisions may change the current proposal:

- Example “launched”: While TinCan API defines this verb with “[...] There is no expectation if this is done by user or system[...]” the virtual patient platform states that “[...] This is not an action made by the user.”
- Example “answered”: The TinCan API definition say “Indicates the actor responded to a Question”, while the MedBiq profile recommends “responded” and even says concerning answered: “User has responded to a question. Response to an item or question within a node, without moving off that node. (This is not quite the same as ‘answered’ which implies that the answer is correct.)”.

This underlines that xAPI is a relatively new standard. Definitions and interpretations should be discussed more in detail with stakeholders especially after the future of MedBiquitous is clearer in 2019. Further developments on using the data from xAPI will be used for dashboards, and

<sup>1</sup> <https://registry.tincanapi.com/#home/verbs>

## D2.3 Developed VP system APIs and platform integration

common understandings will be necessary to agree on meanings of the verbs, along with mappings of different verbs to single common verbs when necessary.

### 3.3. Integrating xAPI in the mobile player

xAPI is the standard used for learning analytics. In order to provide a more complete solution regarding the collection of the xAPI statements, we implemented basic support for generating and sending statements using the xAPI standards from the WAVES Mobile player.

The WAVES Mobile Player is a progressive web app (PWA) that displays a virtual case in a mobile screen, and is designed not only to fit in small screens, but also to maximise the user experience. This is a very basic demonstration and guidelines on how the mobile player can write statements to an LRS. In examples bellow, we use the ADLNET.GOV LRS [4].

In order to use the (limited) statement support we have to register at:

<https://lrs.adlnet.gov>

Then, open the config.js file under `'/src/app/'` and replace the details:

```
Config.endpoint = "https://lrs.adlnet.gov/xapi/";
Config.user = "username";
Config.password = "password";
Config.actor = { "mbox": "name@example.com", "name": "my name" };
```

Traditionally every xAPI client has to ask the LRS for a specific set of statements to report on. Webhooks are a way to register client applications that are interested in certain sets of statements. Every webhook that is registered contains the client endpoint as well as filters that dictate what kind of statements the client wants and the LRS will send.

In order to use the functionality users must create the according webhooks. More info can be found at [https://github.com/adlnet/ADL\\_LRS/wiki/Webhooks](https://github.com/adlnet/ADL_LRS/wiki/Webhooks).

There are 4 functions that can be used to generate statements inside the WAVES Mobile Player app (file `/src/app/play/play.component.ts`):

`ngOnInit()`: Can be used to generate **'launched'** case level statements (when a VC is opened) (<http://adlnet.gov/expapi/verbs/launched> )

`ngOnDestroy()`: Can be used to generate a case level **'completed'** statement (when a VC is completed) (<http://adlnet.gov/expapi/verbs/completed> )

`ngOnNodeEnter()`: Can be used to generate a node level **'arrived'** statement ( Indicates the actor arrived in a VS node) (<http://adlnet.gov/expapi/verbs/completed> )

`ngOnNodeLeave()`: Can be used to generate a node level **'completed'** statement (when a VC is completed) (<http://adlnet.gov/expapi/verbs/completed> )

### D2.3 Developed VP system APIs and platform integration

Statements can be viewed in the 'Statements section of ADLNET.GOV. This section displays human-readable sentences for your stored statements. Administrators can expand them to view the raw JSON data.

Although, the support for generating statements in a Learning Record Store (LRS) is very limited in the WAVES Mobile Player app, it can be expanded in future to support even more actions. The above instructions are just a showcase of how traditional LRS systems and xAPI can be fitted into the mobile player.

### **3.4. Educators access to session data of learner in delivery platform**

The development of xAPI as a standard for use in education is still in its infancy. The community understanding of how the standard can be used to benefit learners and teachers alike is still evolving, and as a consequence so are the tools to facilitate this.

Learning Record Stores (LRS) represent the primary destination for all xAPI statements generated by the learning systems, and by necessity these are continuing to evolve along with the standards. There are a number of different learning record store systems available, both commercial and open source, but a unifying characteristic of all of them is that they are designed primarily for technologists and those with an underlying knowledge of the standard. The systems require extensive configuration, and have a steep learning curve to their use. They do not provide many easy to use solutions for analysing the data from an end-user perspective.

It is beyond the scope of the WAVES project to explore or try to improve end-user functionality and usability for LRS systems. This project merely seeks to implement the standard within the context of Virtual Scenarios in order to ensure maximum compliance and compatibility with current and future tools that become available. Nevertheless, it is our hope that by supporting the xAPI standard and LRS systems, it will encourage development of these systems in a way that increases their utility to end-users i.e. teachers and educationalists; the development of friendly dashboards and simple reporting or analysis functions will allow educators to acquire meaningful insights from their analytics without requiring them to fully understand the underlying technologies.

## **4. MICROSERVICES**

### **4.1. Display score of a played VP, into Moodle, for a specific student**

The display of basic usage data can be realized with LTI when participating platforms support the LTI outcomes management. This was tested successfully with Moodle as the LTI host, with CASUS as the LTI consumer. Testing and production deployment was realized in 2017 by the CASUS team for a project outside of WAVES and has been in production since then without any known issues.

The functionality is performed through the inclusion of an additional parameter: the LTI launch "lis\_outcome\_service\_url". The URL contains the basic URL + any needed parameter to spool

D2.3 Developed VP system APIs and platform integration

back basic usage data like score and time back to Moodle. The URL usually contains parameters to identify course, link and user of the LTI launch.

More information can be found on <https://www.imsglobal.org/specs/ltiomv1p0/specification>

The score is spooled back in a XML structure, the grade itself by a number from 0.0 - 1.0

Outcomes can be spooled back asynchronously to this URL, so grades can also be updated in low traffic times by the LTI consumer. In CASUS we write any results to be spooled back to an extra table containing the back URL, then we can regularly start a service writing all new results back. If an error occurs, e.g due to maintenance, we can flag this and repeat this until all grade are successfully spooled back.

Moodle can handle external grades as internal grades, so can even be used for creating learning logic, or just for simple reports to the Moodle educator.

Not all LTI providers already support this extension, so we recommend reviewing proposed platforms to establish support for this feature set. If available it can be used as a simple and easy means to activate alternatives to xAPI and LRS when only basic usage data is needed. The LTI provider is used in this scenario as the collection system of all grades. It cannot be an entire replacement of xAPI. Due to the generic specification of xAPI and LRS, xAPI can be much more powerful and collect more kinds of data than just the simple outcome management of LTI, and can operate independently of any one learning management system.

## 4.2. Moving OpenLabyrinth case to the mobile player

In the scope of the WAVES project we implemented a mobile player to enhance the user experience in the usability of delivering virtual cases to mobile devices and, generally, on small screens. To achieve maximum performance we used the latest, state-of-the-art technologies to provide a rich experience to the end user. The WAVES Mobile player is implemented as a progressive web app [10] and, thus, runs completely on the client side (the users' device). This approach allows the player to be compatible not only with current versions of OpenLabyrinth but also in future versions.

The technical reference group in WAVES decided to provide a more complete experience of the mobile player, although that this is something beyond the predefined requirements of the workpackage. We developed a microservice that acts as a connector between the OpenLabyrinth virtual cases and the mobile player. The microservice exports the list of available virtual patients in JSON format, which is the format that the mobile player can be read. Once the user selects a case to play, the service delivers in a similar format the whole case, which can be rendered on the WAVES mobile player.

The developed file resides under the folder services:

```
config.php  
mlist.php  
mplay.php
```

## D2.3 Developed VP system APIs and platform integration

In config.php we define the connection details (local Open Labyrinth installation) and the base URL of the application.

Calling /services/mlist.php displays a list of the public available virtual cases, e.g.:

```
{
  "cases": [
    {
      "base": "http://olab.wavesnetwork.eu/",
      "id": "1",
      "name": "Activity streams and informal learning"
    },
    {
      "base": "http://olab.wavesnetwork.eu/",
      "id": "2",
      "name": "Hiccup Virtual Patient"
    },
    {
      "base": "http://olab.wavesnetwork.eu/",
      "id": "3",
      "name": "Common health topics"
    }
  ]
}
```

Finally, calling /services/mplay.php?id=[id] with the desired virtual case id returns the data in a JSON format that can be consumed by the waves mobile player.

There are restrictions, including a lack of support for avatars or counters, but in general this can be used as a base to further develop the service.

The files and short instructions on how to install and use the microservice can be found under the GitHub WAVES public repository at:

<https://github.com/wavesnetwork/mplayer-connector>

### 4.3. Integration of Vispath with OpenLabyrinth

The pathway visualizer tool extends OpenLabyrinth by a pathway visualization tool which is executed at the end of a scenario, for a single user session, when the final node of the scenario is visited. The graph contains a node for each screen card and the links among them showing the possible transitions between the states in the scenario. The tool highlights all visited nodes from a single session of the user (session\_traces) in a different color.

For the purpose of implementing this tool as a microservice, OpenLabyrinth was expanded using the JavaScript library D3.js that allows manipulating documents based on data. The tool generates

### D2.3 Developed VP system APIs and platform integration

a dynamic HTML5 graph canvas based on the virtual scenario content in OpenLabyrinth. The scenario data are acquired from the database in the JSON format. This has the additional benefit of being loosely coupled with the current OpenLabyrinth database scheme which is going through major changes while transforming to the fourth generation of the platform. This makes this component also potentially useful for other branched virtual scenario systems which could integrate it as a micro-service.

The pathway visualizer tool is executed at the end of a scenario, for a single user session, when the final node of the scenario is visited. The graph contains a node for each screen card and the links among them showing the possible transitions between the states in the scenario. The tool highlights all visited nodes from a single session of the user (session\_traces) in a different color. In the visualization in the figure below the blue nodes indicate the visited screen cards, and the white nodes not selected options.

The tool is available in GitHub and can be enabled by overwriting two files:

<https://github.com/wavesnetwork/vispath>.

Moreover, there is a working demo in the following link:

<http://vp.behmed.org/renderLabyrinth/index/1>

The virtual patient in this example is used as part of the course Behavioral Medicine in Karolinska Institutet.

**Conclusion**

powered by OpenLabyrinth

Well done! You have now reached the end of the case. As anticipated, Mrs Violet still has some problems with leakage via the urethra, especially during the night. However, she does not regret her choice of orthotopic neobladder. She still meets her friends for dinner and to play cards. She quit smoking immediately when she found out that smoking increases one's risk for bladder cancer.

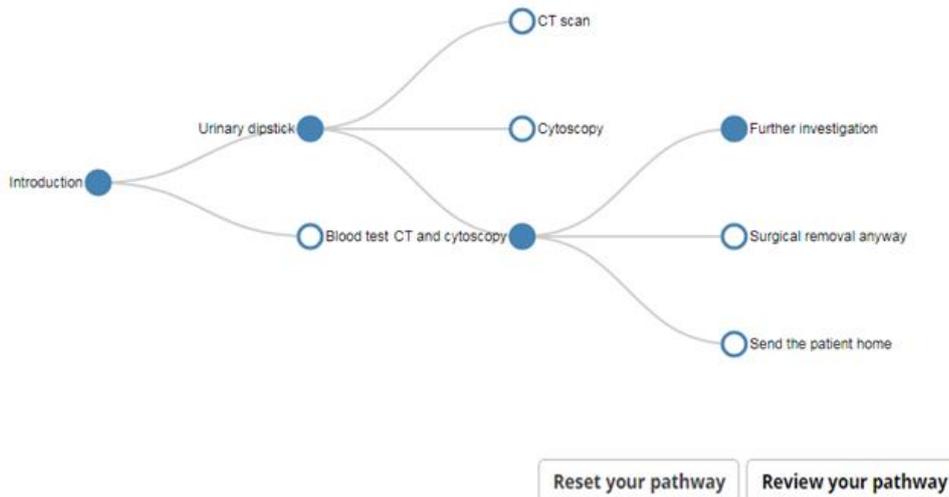


Figure 8: Layout of the learner pathway visualisation component

## 5. BEST PRACTICES, REFINEMENTS AND CHANGES BASED ON EVALUATION AND MONITORING

### 5.1. Basic description of results of evaluation and monitoring

From D5.3 the results of the technical testing are summarized in the following table.

	UT1a			UT1b			UT2		
	Authoring			LTI link			xAPI/VSS-LRS link		
	Success	Time	Difficulty	Success	Time	Difficulty	Success	Time	Difficulty
LMU	Yes	30 min /2	2	Yes	30 min /2	2	Yes	20 min	2
SGUL	Yes	8 min 20s	-	No	5 min	-	Yes	12 min 15s	?
KI	Yes	12 min	1	No	24 min	2	Yes	11 min	3
AUTH	Yes	60 min	3	No	-				
MU	Yes	7 min 35s	1	No	35 min	5			
MU2	Yes	75 min /2	3	No	75 min /2	3			

Table 5: WAVES Technical Testing summary

### D2.3 Developed VP system APIs and platform integration

As can be ascertained from these technical testing results, authoring, the key activity for SBL content creation was smoothly executed from all participants within varying time frames. On the other hand, LTI linking was able to be conducted only in one case with the main causes of failure being a) confusion in the video tutorial and b) non-consistent workflows amongst LMS systems. Finally, regarding xAPI/VSS-LRS links, the tests were successful with relatively consistent and brief time frames.

## 5.2. Best practices, refinements and changes

In this section, we summarize some of the outcomes and best practices concluded from our work in WP2:

- xAPI is a key standard in the field. A minimum baseline of compatibility between systems regarding the xAPI is essential for VS systems. The WAVES group produced the 'Levels of Conformance' as a quantity metric for the compatibility (e.g Table 7: Recommended verbs for conformance Level 1)
- The compatibility of the systems and protocols is an important aspect regarding interoperability and integration between the VS systems.
- It's important to follow the standards when it comes to implementation of LTI and xAPI technologies, in order to achieve maximum compatibility with current and future versions of the VS systems. Progress is better to take place step-by-step, following the evolution of the standards involved (which most of the time is a slow procedure).
- People (the learning technologists, administrators and authors) have a crucial role in successfully implementing integrations between the VS platforms and other applications. Supporting these people with proper documentation, guidelines, best practices and examples helps to prevent failures in implementation.
- While it's not difficult to add or extend new functionality to current VS systems, developers should respect the scope of the followed standards. The WAVES project sets the basic principles on how to enhance usability and develop extensions, while keeping the limits within current standards.

## 6. CONCLUSIONS

This deliverable aims to present the developed APIs and platform integration between VS systems that took place during the WAVES project. We structured our work under 3 main pillars: Learning Tools Interoperability (LTI), Experience API (xAPI) and microservices. We checked the compatibility of the two main VS systems (CASUS and OpenLabyrinth) against LTI. We researched the state of xAPI support in the CASUS and OpenLabyrinth and we moved forward by developing conformance level guidelines. We extended the developed tools from previous work packages, such as the mobile player for virtual patients, in order to provide support for xAPI. We examined the functionality of our tools against Learning Record Stores, aiming at the maximum general compatibility between commercial and open source LRSs. Under the microservices pillar, we described several tools to enhance the information availability across the systems, as well as

D2.3 Developed VP system APIs and platform integration

functions to export VS cases from OpenLabyrinth to the mobile player. Finally, we described in brief the main results from the evaluation and monitoring activities of the WAVES project, and listed some of our findings in a take-away key list of best practices and guidelines.

## 7. REFERENCES

[1] xAPI.com. (2018). What is xAPI aka the Experience API or Tin Can API. [online] Available at: <https://xapi.com/overview/> [Accessed 4 Dec. 2018].

[2] "Learning Tools Interoperability | IMS Global Learning Consortium". Imsglobal.Org, 2018, <http://www.imsglobal.org/activity/learning-tools-interoperability>. Accessed 14 Dec 2018.

[3] "Setting Up Desire2learn LTI Authentication To Openlabyrinth | Openlabyrinth". Openlabyrinth.Ca, 2018, <http://openlabyrinth.ca/setting-up-d2l-to-olab-lti-authentication/>. Accessed 14 Dec 2018.

[4] Adlnet.gov. (2018). About - ADL Initiative. [online] Available at: <https://www.adlnet.gov/about> [Accessed 4 Dec. 2018].

[5] LTI outcome management, <https://www.imsglobal.org/specs/ltiomv1p0/specification>

[6] CASUS: [www.casus.net](http://www.casus.net)

[7] Openlabyrinth.ca. (2018). Using Experience API (xAPI) on OpenLabyrinth | OpenLabyrinth. [online] Available at: <https://openlabyrinth.ca/using-experience-api-xapi-on-openlabyrinth/> [Accessed 4 Dec. 2018].

[8] GrassBlade LRS <https://www.nextsoftwaresolutions.com/grassblade-lrs-experience-api/> [Accessed 11 Dec. 2018].

[9] xAPI verbs registry: <https://registry.tincanapi.com/#home/verbs>

[10] "Progressive Web Apps | Web | Google Developers". Google Developers, 2018, <https://developers.google.com/web/progressive-web-apps/>. Accessed 13 Dec 2018.

[11] MedBiquitous xAPI profile for virtual patients: <https://github.com/medbiq/medbiq/blob/master/xapi/xapi-virtualpatient-profile.md>