Co-funded by the Horizon 2020
Framework Programme of the European Union

HORIZON2020 FRAMEWORK PROGRAMME

ICT – 21 -2014

Advanced digital gaming/gamification technologies

ProsocialLearn

Gamification of Prosocial Learning

for Increased Youth Inclusion and Academic Achievement

D5.6

1st Platform operations report

Document Control Page

WP/Task	WP5 / T5.3
Title	D5.6 - 1st Platform operations report
Due date	28/02/2017
Submission date	09/06/2017
Abstract	Document provides a report on platform operation during trials
Author(s)	Francesco D'Andria (ATOS)
Contributor(s)	Simon Crowle (ITINNOV), Jose Miguel Garrido (ATOS), Enrique Quiros (ATOS), Kostas Apostolakis (CERTH)
Reviewer(s)	Christopher Peters (KTH)
Dissemination level	☐ internal ☒ public ☐ confidential

Document Control Page

Version	Date	Modified by	Comments
0.1	09/01/2017	Francesco D'Andria (ATOS)	ToC
0.2	16/01/2017	Enrique Quiros, Jose Miguel Garrido, Francesco D'Andria (ATOS)	Section 2, Annex B
0.3	20/01/2017	Simon Crowle (IT-Innovation)	Annex A
0.4	27/01/2017	Francesco D'Andria (ATOS)	Annex B, C, D
0.5	13/03/2017	Kostas Apostolakis (CERTH)	Section 2.6
0.9	24/04/2017	Francesco D'Andria (ATOS)	Preparation of the final draft version ready to be internally reviewed.
1.0	28/04/2017	Francesco D'Andria (ATOS)	Final version of the document after the internal review

List of Abbreviations

Abbreviation	Description
API	Application Programming Interface
CLI	Common Line Interface
EC	European Commission
GLM	Game Lifecycle Management
LG	Learning Group
LP	Lesson Plan
LRS	Learning Record Store
LTP	Lesson Plan Template
PAM	Prosocial Adaptation Manager
PLO	Prosocial Learning Objective
T	Task
WP	Work Package

Executive summary

WP5 “Prosocial Platform development and Operation” aims at performing the systems integration of all the ProsocialLearn technologies and will operate the ProsocialLearn platform for development of games.

The project integration strategy has been aligned with the major milestones of the project lifecycle and dependencies described through architecture. Incremental versions of the platform have been delivered offering increasing levels of functionality.

On the other hand, the work carried out in the task T5.3 “Prosocial platform operations” aims at deploying and operating the ProsocialLearn Platform for the five Gaming Providers (are delivering prosocial game through our platform) will be used in the longitudinal studies conducted by WP7 to validate the benefits of using the ProsocialLearn platform. The platform will be offered as a service with consideration of the Quality of Service needed by game developers and teaching professionals dependent on it.

This deliverable D5.6 presents a first version of the ProsocialLearn platform operations report. It will be mainly focused on describing the interaction between Gaming Providers and the platform itself. It contains the basic description of a four phase procedure gaming providers have to follow to create prosocial games that can be managed by the ProsocialLearn platform.

The final version of the ProsocialLearn platform operations will be presented in the deliverable D5.7 “2nd Platform operations report” and will build on this deliverable to describe also the interaction of actors (i.e. schools) with our platform.

Index

1 Introduction

A core capability of the ProsocialLearn platform is the digital distribution of prosocial games.

It offers a pro-social methodology to implement digital games aimed at increasing social inclusion and academic performance as well as software facilities and a community to support the creation and delivery of digital games for children (7-10 years old) within educational systems that support learning of prosocial skills.

The platform provides a freely available application programming interface (API), which gaming developers can use to integrate many of ProsocialLearn functions into their games, including emotion and engagement monitoring, in-game achievements, micro-transactions, etc.

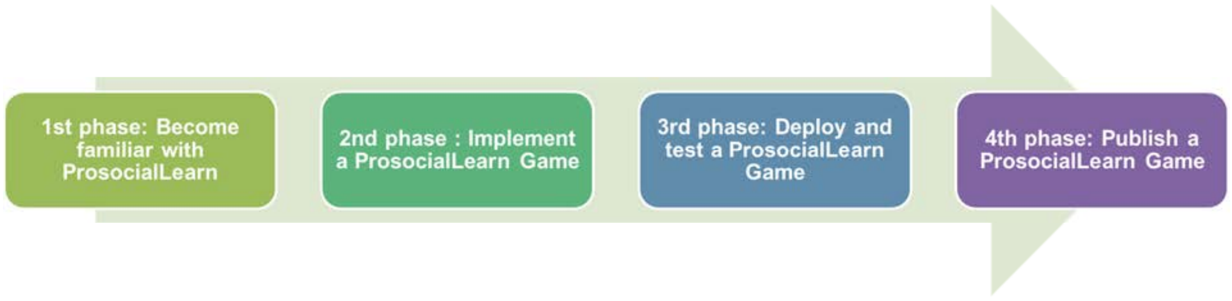
The main objective of this document is to arrange a guide gaming developers can use to learn about how-to design, implement, deploy and test prosocial games rely on the ProsocialLearn approach.

During the lifecycle of the ProsocialLearn project five gaming providers (PlayGen Ltd, REDIKOD AB, ANIWAY OY, HYPESLUGS SRL and MAD ABOUT PANDAS UG) are adopting the ProsocialLearn platform and infrastructure both for integrating the main functionalities the platform provides and deploying their games in a secure and scalable cloud environment.

Intentionally, the report will not face the interaction with schools' actors. This will be faced in the last in the deliverable D5.7 – 2nd Platform operations report. This report will be delivered by the end of the project (M36, December of 2017).

1.1 Purpose and structure of the document

This deliverable contains the basic description of the procedure gaming providers have to follow to create prosocial games can be managed by the ProsocialLearn platform. At this scope 4 learning phases have been defined.


1st phase: Become familiar with ProsocialLearn
2nd phase : Implement a ProsocialLearn Game
3rd phase: Deploy and test a ProsocialLearn Game
4th phase: Publish a ProsocialLearn Game

<u>1st phase: Become familiar with ProsocialLearn</u>

Gaming Developers have to get familiar with the ProsocialLearn platform. The project Gitlab[1] provides information to customers (Gaming Providers and Schools' actors) about ProsocialLearn platform and services. The ProsocialLearn webpage[2] provides details about the ProsocialLearn project. It links to detailed information about:

[1] https://gitlab.atosresearch.eu/ari/prosociallearn/

[2] https://gitlab.atosresearch.eu/ari/prosociallearn/tree/master/documentation/developerguide/psl-architecture

- Platform introduction[3]
- Platform overview[4]
- Platform requirements[5]
- Systems architecture overview[6]
- Business architecture overview[7]**¡Error! No se encuentra el origen de la referencia.**

This information is available in the public deliverables D2.4 "2nd System Requirements and Architecture" [1], D3.2 and D3.3 "1st and 2nd Prosocial affect fusion and player modelling" [2] and [3].

2nd Phase: Implement a prosocial game compatible with ProsocialLearn

During this phase, gaming developers will learn how-to design and implement prosocial games that will be managed by the ProsocialLearn platform (topic developed in Chapter 2).

3rd Phase: Deploy a ProsocialLearn Game

In this 3rd phase, game developers learn to deploy a prosocial game, using the ProsocialLearn deployment facilities, to the ProsocialLearn cloud infrastructure (topic developed in Chapter 3).

4th Phase: Publish and Run ProsocialLearn Game

The 4th phase is a *how-to* concering the use of the ProsocialLearn facilities to publish games in the official ProsocialLearn market store to be acquired by schools' actors (topic developed in Chapter 4).

Therefore, the document is organized as follows:

Chapter 2 briefly summarises Game service interfaces, Lesson plan template in JSON, Game lifecycle management, Lesson management

Chapter 3 introduces the ProsocialLearn infrastructure that involves a testbed and the following technologies: Docker, OpenShift, Jenkins.

Chapter 4 overviews the ProsocialLearn MarketPlace service.

Chapter 5 conclude the document.

1.2 Scope and Audience of the document

The dissemination level of this document is public.

This deliverable provides useful information for all Gaming Providers (not only the five enterprises belong to the ProsocialLearn consortium) aimed at developing prosocial games relying on the ProsocialLearn platform.

[3] https://gitlab.atosresearch.eu/ari/prosociallearn/blob/master/documentation/developerguide/psl-architecture/PlatformIntroduction.md

[4] https://gitlab.atosresearch.eu/ari/prosociallearn/blob/master/documentation/developerguide/psl-architecture/PlatformOverview.md

[5] https://gitlab.atosresearch.eu/ari/prosociallearn/blob/master/documentation/developerguide/psl-architecture/PlatformRequirements.md

[6] https://gitlab.atosresearch.eu/ari/prosociallearn/blob/master/documentation/developerguide/psl-architecture/PlatformInformationSystemsArchitecture.md

[7] https://gitlab.atosresearch.eu/ari/prosociallearn/blob/master/documentation/developerguide/psl-architecture/PlatformBusinessArchitecture.md

2 How-to implement a prosocial game compatible with the ProsocialLearn platform

2.1 Game Architectural Approach

ProsocialLearn is a digital distribution platform for pro-social games. It offers a pro-social methodology to implement digital games.

The platform provides a freely available and easy-to-integrate Application Programming Interface (API), which gaming developers can use to integrate many ProsocialLearn functions into their games, including gaming management, emotion and engagement monitoring through player (face, audio and body) monitoring, gaming adaptation, in-game achievements, micro-transactions, etc.

Games managed by the ProsocialLearn platform have to follow a well-defined architectural style. Below are a short list of the main characteristics a ProsocialLearn game should have:

- The games implement a standard architecture pattern providing a series of interfaces as presented in Figure 1. The ProsocialLearn technical approach allows games developers the liberty to use their favourite technology (Unify 3D, Unreal Engine, Node.js, etc.) to implement games.
- In this context games are going to be governed by the ProsociaLearn platform. The platform manages (create, start, stop/pause and destroy) game matches which are instances of games. Those instances serve one or more players that play together (even cooperating) to the same goal. The interaction between the platform and the game is supported through well-defined APIs.
- Games matches are managed through a well-defined lifecycle defining the multiple game states as presented in Figure 4.
- ProsocialLearn games are typically multiplayer. A multiplayer video game is a video game in which more than one person can play in the same game environment at the same time. Multiplayer games allow players interaction with other individuals in partnership, competition or rivalry, providing them with social communication absent from single-player games. In multiplayer games, players may compete against two (or more) human contestants, work cooperatively with a human partner to achieve a common goal, supervise other players' activity, co-op, and objective-based modes involving controlling points on a map[8].
  The interaction among players during the game play should be transparent to the platform. However, the platform is notified about any relevant actions taken by the game necessary for measuring player performance and progress towards objectives.

2.2 Game Service Interfaces

Game servers must expose a series of interfaces (see Figure 1) to be managed by the ProsociaLearn platform:

- **getLessonPlanTemplate():** Interface invoked by the ProsociaLearn platform during the game registration in the market place to retrieve the Lesson Plan Template (LPT) as a JSON. LPT provides a detailed description of the game with a series of instruction that can be tailored

[8] https://en.wikipedia.org/wiki/Multiplayer_video_game

by teachers for specific Learning Groups. This includes default game configuration of offline and real-time adaptation mechanisms and a Lesson Allocation Plan for distributing members of Learning Groups to Game instances (additional information can be found in the ProsocialLearn deliverable D2.4 [1]). In Annex D, the reader can find the code necessary to implement the interface (Game Server side). Moreover, there is a full example representation (JSON file) of the Lesson Plan Template document gaming providers has to provide together with the game.

- **connectPlayer(MatchID, PlayerID):** Interface invoked by the GameFront_end to retrieve match context and start the match.
- **setGameState(gameState: {start|pause|stop}, gameId):** Interface invoked by the platform to set a specific game state.
- **getFrontEndURL(matchID):** Interface invoked by the platform to know where the game front-end is located.
- **playGame(matchID):** Interface invoked by the game front-end to retrieve the specific context of the match.

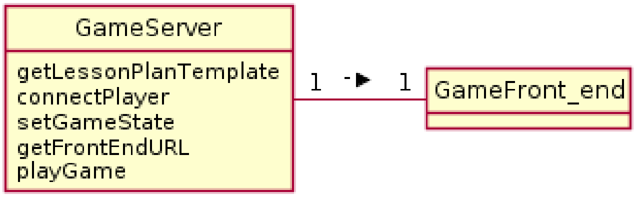


Figure 1 - ProsocialLearn Games conceptual classes

2.3 Interaction between the ProsocialLearn Platform and the Game

The ProsocialLearn platform is concerned with managing the lifecycle of lessons (prosocial game as part of a lesson) delivered to Learning Groups within a Learning Provider organisation (school). The platform is responsible also for ensuring that all platform resources necessary to deliver the learning experience are provisioned and available for the period of learning.

A Teacher describes the Lesson requirements through a Teacher's dashboard by selecting a *Learning Unit* and tailoring a *Lesson Plan Template* (provided by the Game Provider) for a Learning Group. This process includes tailoring group and individual level prosocial learning objectives, selecting an allocation plan for distributing students in the group to game instances, and setting any game specific configuration if needed. The lesson is given a start time and duration which is initially taken from the Lesson Plan metadata. The platform is the only responsible to implementing this process (games are not called during the lesson definition; the Figure 2 shows the sequence diagram about how the platform creates/stores a Lesson). Further information about what a lesson is can be found in [1].

The platform uses such information to provision resources for a Lesson at runtime. On the other hand, the platform provides a fully automated procedure to minimise teacher administration tasks, therefore increasing their productivity.

Game Lifecycle Management – GLM - (component belongs to the platform) provisions and configures game instances for Lessons. Game instance management is achieved using a Game Management API implemented by Game Servers allowing the platform to automate the allocation of Learning Groups members to Game instances.

Lesson Management – LM - provides the GLM with a Lesson plan and the GLM, using a Game Management API, initiates the creation of game instances on a game server and allocation Students to those game instances. The GLM implements the allocation algorithms for associating students to game instances based on requirements in the Lesson Plan (e.g. Random or full customised) and considers the constraints of the game itself (e.g. the number of players for game).

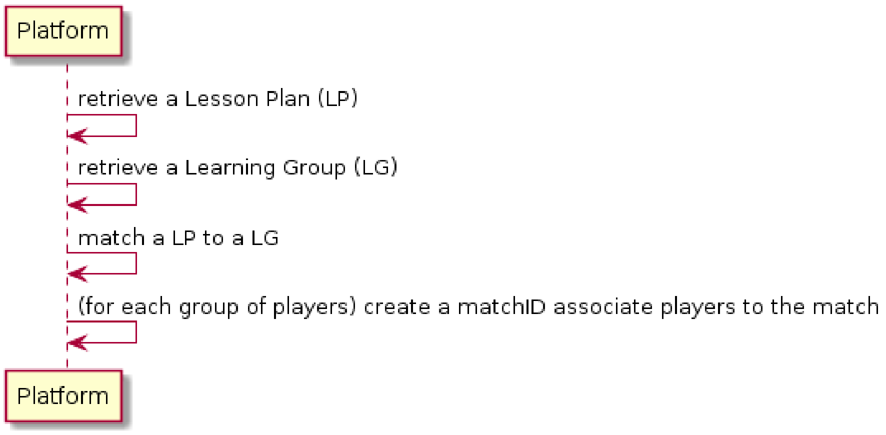

Platform
retrieve a Lesson Plan (LP)
retrieve a Learning Group (LG)
match a LP to a LG
(for each group of players) create a matchID associate players to the match
Platform

Figure 2 - Sequence Diagram 1: The platform creates/stores a Lesson

Each Game Instance may have a platform configuration. This info (that has been already provided by the game during its registration in the market place) is sent back from the platform to the game as soon as the player logs-in to the platform to play the game. Figure 3 shows the sequence diagram about players log-in to the platform and join a multiplayer match (the sequence is not focusing on security interaction).

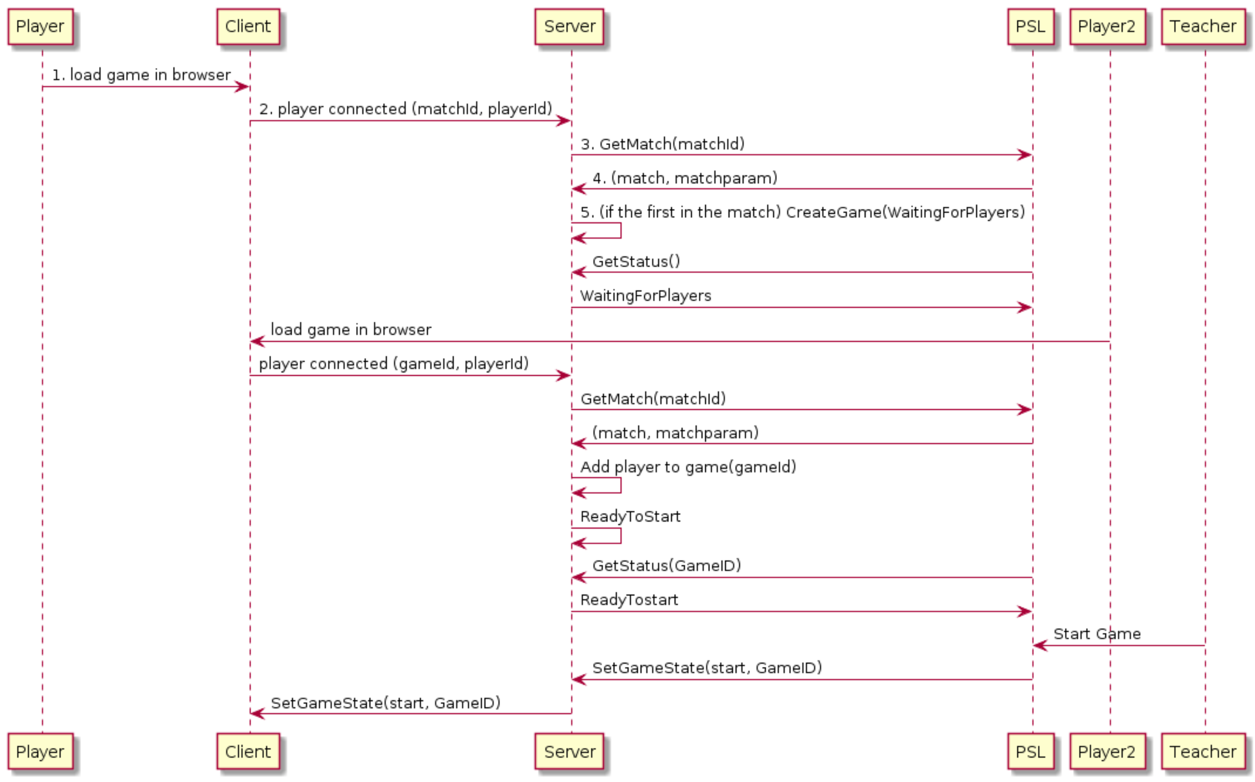


Figure 3 - Player log-in to the platform and join the match (without focusing on security)

2.4 Game Matches Status

Matches pass through states as presented in Figure 4 and summarized below.

- Game match state "initializing": the game server creates the game match when the first players joins the match. The match is not stated yet. Automatically the match passes to the next state "waitingForPlayers".
- Game match state "waitingForPlayers": players are able to join the game match. The match is not stated yet. As soon as the match is complete (all the players joined the match) match pass to the next state "ReadyToStart".
- Game match state "ReadyToStart": the game is ready to start. Once the Teacher click on Start Game match pass to the next state "Running".
- Game match state "Running": the match is running. The match can change state to "Stopped" or "Paused" if the teacher asks for Stopping or Pausing the match. The match can change state to error if an internal error rises.
- Game match state "Stopped": when the round is out or the teacher stopped the game for any reason.
- Game match state "Paused": the match is in pause.
- Game match state "Error": when match is in error.

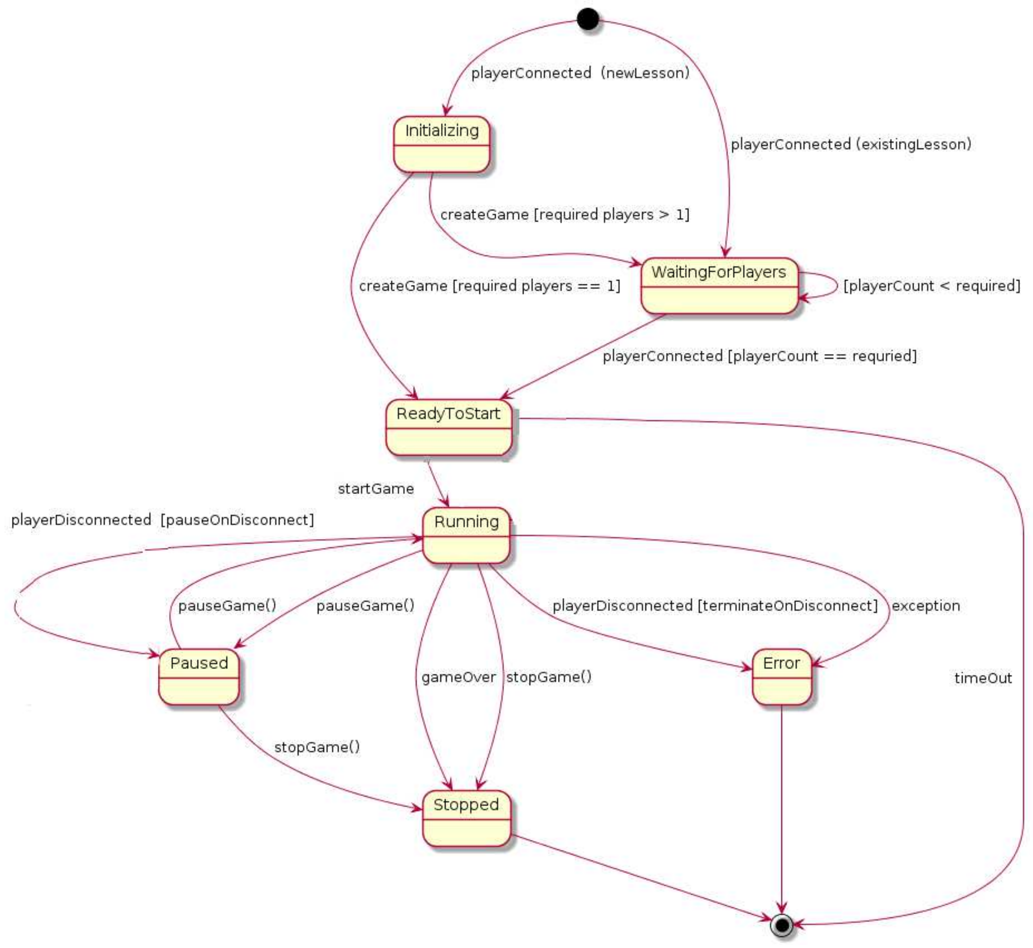

playerConnected (newLesson)
playerConnected (existingLesson)
Initializing
createGame [required players > 1]
createGame [required players == 1]
WaitingForPlayers
[playerCount < required]
playerConnected [playerCount == requried]
ReadyToStart
startGame
playerDisconnected [pauseOnDisconnect]
Running
pauseGame()
pauseGame()
playerDisconnected [terminateOnDisconnect]
exception
Paused
Error
gameOver
stopGame()
timeOut
stopGame()
Stopped

Figure 4 - Matches State Diagram

2.5 Interaction with sensors

Chapter 5.3.7 of the ProsocialLearn deliverable D2.4 [1] provides a detailed overview of the Player Affect Observation Subsystem. In the context of this deliverable we will focus on how gaming providers can easily integrate such functionality in their games.

Player affect observations are modelled in accordance with the OGC Observation and Measurement model according to the JSON implementation profile [15]. Valance-arousal dimensional space is used for all modes of measurement unit for classification of emotional state [16]. This allows a discretisation process to split the space and cover all the different emotions required for all the different input modalities and allows classifiers to use the same labels and apply basins of attraction around emotions concepts we can cluster them together.

Table 1 - Emotion Observation Mo

Observation	Features of Interest	Measurement	Measurement unit
Voice	Carious sound features (energy in frequency band,	Audio stream capture, feature extraction	Valance-Arousal

	pitch, duration)	classification	
Facial Expression	Low-level distance and angle measures associated with eyes, eyebrows and mouth	Video stream capture, feature extraction and classification	Valance-Arousal
Self-Reported Emotions	Feedback provided by the user on perception of valence and arousal	Selection of valance and arousal from in-game emoticons	Valance-Arousal

The sensor observation channels are incorporated into a data fusion pipeline using decision-level techniques to bring together multi-modal sources and analyse emotional affect. Decision-level fusion is used to determine events along a timeline at which learned decisions are extracted from all unimodal features and integrated into higher semantic concepts (e.g. valance arousal). The platform enables multichannel acquisition of observations. Each channel has its data processing pipeline related to the nature of the observation but are integrated into the platform using a consistent architectural model to allow for manageability. Each classifier implements a stream-based processing function that processes sensor data input and asserts player affect observation output for downstream processing and analytics (e.g. Adaptation and Learning Record Store). The distribution of the data processing function across different components is not dictated by the platform as it is dependent on resourcing and data protection constraints, and associated trade-offs.

A wiki page [7] describes the ProsocialLearn JavaScript game client API (almost the same information can be found in the Annex A of this deliverable).

This API allows gaming developers to:

- Access player and game instance meta-data from the ProsocialLearn platform
- Stream audio (voice) data to the ProsocialLearn platform for emotion analysis
- Stream visual (face) data to the ProsocialLearn platform for emotion analysis
- Access local time offset of your player relative to the ProsocialLearn platform time

2.6 Game adaptation

The aforementioned elements are measured through an observations acquisition framework that aims to transform raw signal data into player affect and engagement information. The information on user engagement [6] for a particular game is used to identify the game's elements that rank highest for a particular student's persistent *engagement profile* for that game. A separate *prosociality profile* is also maintained for each player to rank players according to their achievements with respect to a number of Prosocial Learning Objectives (PLOs), indicating player's ability to properly use and display specific prosocial behaviours or skills.

A dedicated Prosocial Adaptation Manager (PAM) component is therefore employed to enable the use of adaptive elements in a variety of prosocial games. Two modes for game adaptation are supported: a) offline adaptation; and b) online adaptation. Offline adaptation occurs prior to the game launch and aims to match the player's prosocial ability to a *game scenario*, i.e. pre-launch game conditions (such as difficulty setting, contextual data and story-driven interactions) that appropriately match the player's prosociality ranking. Game scenarios ranking is constantly updated as more and more players attempt, win or fail in achieving a particular scenario's prosocial objectives; offline adaptation therefore matches players to a game setting that is determined (by the total number of players) to be the most appropriate to their current prosociality ranking.

Online adaptation on the other hand is responsible for making dynamic adjustments to the game in order to guide players towards game content they'll most likely enjoy, thus maintaining their engagement high. The adaptive features in this mode of adaptation, e.g. the *game elements* are best represented in the case of prosocial games as the game's means for administering positive reinforcement and corrective feedback to commend or correct player in-game actions with respect to the prosocial skill they need to demonstrate. Such elements may include among others the use of in-game GUI messages, graphics, audio messages or tutoring avatars. The PAM uses the player's engagement profile for a particular game to determine the elements that are ranked highest in terms of in terms of engagement, and are likely to receive the most favourable response from the player, thus maintaining their engagement high and maximizing the potential of learning outcomes.

To address adaptation in multiplayer games, a single-winner election method computes each player's ranking score for each element based on its position in the player's preference list and then estimates the group's score for each scenario or element, thus choosing the most appropriate element based on the group, rather than the single player.

2.6.1 PAM Integration in short

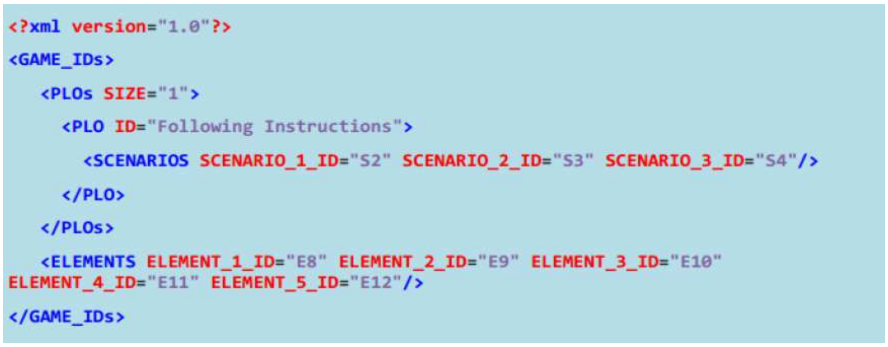
Introduction

- The Prosocial Adaptation Manager (PAM) is responsible for personalizing prosocial games to meet the students' needs and preferences.
- Any game provided with adaptive features can communicate with the PAM.
- Two phases of adaptation are supported: offline adaptation and online adaptation.
- Multiplayer games are supported through a special group adaptation mode, promoting group-specific social interactions.
- Communication with the games is provided through simple web socket messages, requiring only the designated identifiers for the games' adaptive features.

Usage Requirements

- Adaptive Features: Each game should include two sets of adaptive features, namely the game scenarios and elements. Scenarios are selected in offline adaptation to adjust game conditions before the game starts, and elements are used in online adaptation for feed-back and other means of keeping the players' engagement high while playing.
- XML file listing adaptive feature IDs: For every prosocial game registered, an XML file should be generated for the PAM, containing the IDs of the scenarios and elements offered by the game for offline and online adaptation, respectively. This XML must be placed into Game IDs folder.

Table 2 - PAM Configuration XML file



```
<?xml version="1.0"?>
<GAME_IDs>
   <PLOs SIZE="1">
     <PLO ID="Following Instructions">
       <SCENARIOS SCENARIO_1_ID="S2" SCENARIO_2_ID="S3" SCENARIO_3_ID="S4"/>
     </PLO>
   </PLOs>
   <ELEMENTS ELEMENT_1_ID="E8" ELEMENT_2_ID="E9" ELEMENT_3_ID="E10"
ELEMENT_4_ID="E11" ELEMENT_5_ID="E12"/>
</GAME_IDs>
```

- Client Synchronization Interface (CSI): Any single-player game can connect directly to the PAM via a web socket interface. A CSI should offer a service that combines two or more clients into a single game session as well as synchronization support, interconnecting PAM with multiple player clients.
- Web Socket: A web socket in to be able to connect to the IP and port that the PAM has been configured to listen to.
- Listener: A listener for PAM messages
- Communication with PAM involves only 3 types of messages for sending data to the PAM and 2 types for receiving input from the PAM. Reserved words (identifiers) are separated from update information or values with an underscore ("_") character and are formatted in capital letters.
- For multiplayer games, the CSI should open one web socket connection with the PAM for every game session.
- Only 3 types of messages are supported for initialization, engagement estimation, and prosocial outcome.
- At the start of the game: Client sends a message starting with the single reserved word GAME, followed by the exact name of the game, including spaces (i.e. Path of Trust), followed by the name of the PLO/skill, including spaces (i.e. Following Instructions), followed by the player's unique UUID. This tells the PAM which Game_IDs XML file to look for. PAM establishes the connection with a response using keywords PAM and OFFLINE, followed by preferred scenario ID (as designated by the developer in Game_IDs XML, see Section 2, Usage Requirements) and preferred element ID (again, as designated in the Game_IDs XML defined by the developer).
- Multiple times during the game, the client should send the player's engagement value to the PAM using the reserved keyword ENGAGEMENT, followed by the value (within the range 0 to 1). The PAM should respond with a message with reserved keywords PAM_ONLINE and the preferred element ID, as defined in the Game_IDs XML by the developer (see Section 2 Usage Requirements).
- At the end of the game, the client should send the player's prosocial outcome with respect to the skill being tested. The variable ProsocialOutcome can take the following values: PROSOCIAL, FAILURE or UNKNOWN, according to the outcome of the game with respect to the skill being tested (in this case, Following Instructions).

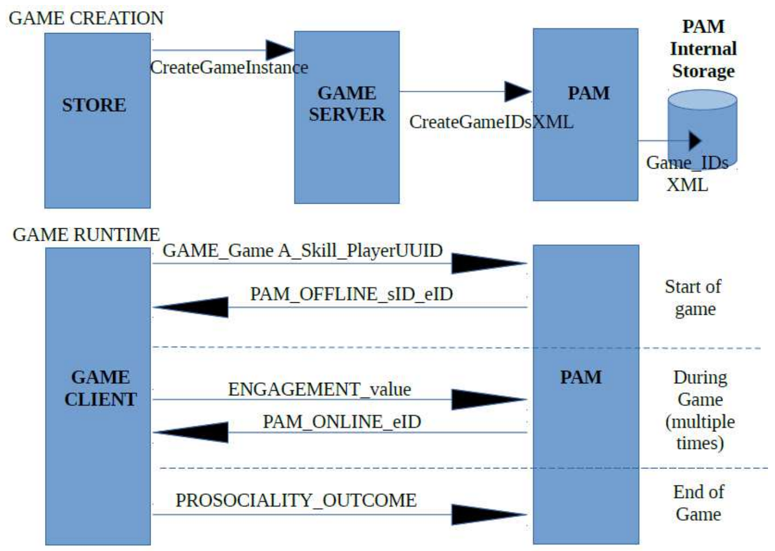


Figure 5 - Interaction ProsocialLearn Platform - PAM – Game Frontend

2.7 Prosocial Game Development Starter Kit

To simplify, even more, the work carried out by programmers want to implement games relying on the ProsocialLearn approach, the consortium implemented a "silly game" used as example to follow composed by two modules (a game server and a front-end component). The code can be easily downloaded logging-in the project Wiki [8]. Annex B provides a short guide on how to use the "silly game" code.

2.8 Tools to support games integration with the ProsocialLearn platform.

To successfully drive and support games integration with the ProsocialLearn platform the consortium defined a series of activities (organized in a well-defined roadmap) game developers have to implement to deliver their games integrated with the basic functionalities provided by the platform.

1. ID_1 Game Deployment in the ProsocialLearn staging environment
2. ID_2 Implementation of the GetLessonPlanTemplate interface and LessonPlanTemplate Json
3. ID_3 Integrate the API for the student observation (voice, audio and body)
4. ID_4 Integrate the Login (implementing the multiplayer mechanism)
5. ID_5 Integration with the PAM
6. ID_6 Game match governance/management
7. ID_7 Game interact with the Learning Docker
8. ID_8 Game Deployment with OpenShift

To effectively monitor these activities, "issues" (an easy way to keep track of tasks, enhancements, and bugs for the projects) have been created (see the Project GitLab: https://gitlab.atosresearch.eu/ari/prosociallearn/). Moreover the technical team created persistent chat rooms (channels) organized by topic (using the Slack Software). All content inside Slack is searchable, including files, conversations, and people. Slack integrates with a large number of third-party services and supports community-built integrations (as well as the project GitLab, so any commit in the GitLab has been notified to Slack users).

3 Deploy games in the ProsociaLearn infrastructure

As mentioned in previous reports (produced in the context of the WP5), the ProsocialLearn platform and infrastructure has been deployed in a protected infrastructure that the consortium virtualized to meet the platform requirements.

A detailed description of how the project has organized the infrastructure can be found in D5.5 [9].

On the other hand, since D5.5 is confidential, for the benefit of readers we repeat basic concepts driving the project operational approach.

The ProsocialLearn platform has been hosted by the Leaseweb (https://www.leaseweb.com/) infrastructure provider. Although Leaseweb has several data centers in Europe, U.S. and Asia, the ProsocialLearn physical machines are hosted by the Amsterdam data center.

ATOS as coordinator and responsible to operate the ProsocialLearn platform, has also the responsibility to govern the ProsocialLearn infrastructure (composed by several virtual machines and physical servers) to deliver the ProsocialLearn functionalities to end users (Gaming Providers, Budget Holders, Teachers and Students) under a Service Level Agreement (SLA) / Quality of Experience (QoE). The data center uses the latest technology to guarantee powerful performance and availability, it is reliable too. Multiple levels of redundancy have been included to ensure consistently high performance.

According to the analysis performed in the Section 5.4 "Scalability" of the report D2.4 [1], the ProsocialLearn infrastructure needs to be reliable but also robust and effective.

The Annex C: ProsocialLearn Infrastructure" of this report details the ProsocialLearn infrastructure.

Key characteristics are:

- Unmetered hosting: a hosting plan with unmetered traffic.

- Hard Disk: 2 Hard Disks of 4TB in RAID 1;

- CPU: 2x Intel Octa-Core Xeon E5-2650

- Memory RAM of 128GB in DDR3 type

Figure 6 gives a high-level overview of how the infrastructure is organized to host the components belonging to the ProsocialLearn platform.

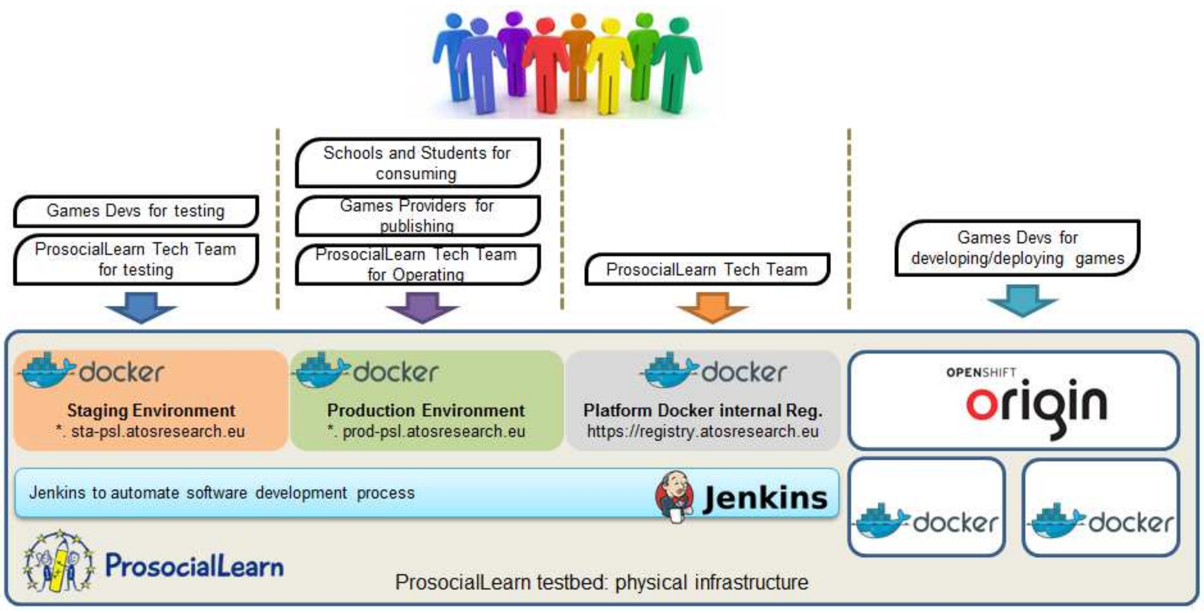

Games Devs for testing
ProsocialLearn Tech Team for testing
Schools and Students for consuming
Games Providers for publishing
ProsocialLearn Tech Team for Operating
ProsocialLearn Tech Team
Games Devs for developing/deploying games
docker
Staging Environment
*. sta-psl.atosresearch.eu
Production Environment
*. prod-psl.atosresearch.eu
Platform Docker internal Reg.
https://registry.atosresearch.eu
OPENSHIFT origin
Jenkins to automate software development process
Jenkins
ProsocialLearn
ProsocialLearn testbed: physical infrastructure

Figure 6 - ProsociaLearn Testbed

A physical data center supports the execution of several virtual machines representing separate environments for developing, testing and releasing prosocial games:

- ***Staging ProsocialLearn Environment***: The staging environment is a virtual machine that hosts all the components belonging to the ProsocialLearn platform, embedded in Docker images. It is going to be used by ProsocialLearn technicians and game developers to test the platform functionalities as well as the actual games. Only when the code is mature, will it be moved to the production environment.
- ***Production ProsocialLearn Environment***: The production environment is a virtual machine that follows the same approach of the staging environment. It represents the real-time staging of ProsocialLearn platform and is the place where the games are executed. It is going to be used by budget holders to acquire games, teachers and students as well.
- ***Platform Docker internal Registry***: This virtual machine hosts an internal Docker images registry used by platform technicians to release new versions of the ProsocialLearn components.
- ***Jenkins Service:*** This virtual machine hosts a Jenkins Service[9]. It helps platform technicians to automate the non-human part of the whole software development process, following a continuous integration / continuous delivery approach.
- ***OpenShift Origin***: This virtual machine hosts an OpenShift Origin PaaS solution[10]. OpenShift Origin is an application platform where gaming developers and teams can build, test, deploy, and run their games. OpenShift Origin also serves as the upstream code base upon which OpenShift Online and OpenShift Container Platform are built.

3.1 Local instance of OpenShift Origin PaaS Cloud

[9] http://jenkins.prosociallearn.eu

[10] https://pslgames.atosresearch.eu:8443/console/

One of the key functionalities the ProsocialLearn platform needed to bring was to develop and deliver technology (fully integrated with the rest of the platform components) supporting game developers with all the game implementation processes: develop the games, build, deploy on a secure and reliable infrastructure, testing etc.

The consortium also promised to implement a Game Platform as a Service (PaaS) infrastructure that removes all of the underlying service "pains" and allow developers to focus on application development only.

We finally chose OpenShift Origin (Red Hat's application platform) software which, from the gaming providers' perspective, can be easily used to govern apps and scale resources.

The consortium installed the OpenShift Origin software in the project premises to implement private cloud to provide the same basic benefits of public cloud but in a much more controlled environment. These benefits include self-service and vertical and horizontal scalability; multi-tenancy; the ability to provision resources on demand; but also changing computing resources on-demand; and creating multiple machines for complex computing jobs, such as big data.

OpenShift Origin also serves as the upstream code base upon which OpenShift Online and OpenShift Container Platform are built.

OpenShift Origin provides a quite complete and detailed documentation, where developers can find information and guides to help them learn about the OpenShift Origin software and start exploring its features [10].

OpenShift Origin dev guide [11] helps developers set up and configure a workstation to develop and deploy applications in an OpenShift Origin cloud environment with a command-line interface (CLI). The guide provides detailed instructions and examples to help developers:

1. Monitor and browse projects with the web console
2. Configure and utilize the CLI
3. Generate configurations using templates
4. Manage builds, images and webhooks
5. Define and trigger deployments
6. Integrate external services (databases, SaaS endpoints)

Besides that, although it is not mandatory, the ProsociaLearn technical team strongly suggests to deploy Games embedded in Dockers images.

An overview of the advantages of using Docker can be found in Redhat's customer portal [12].

Finally the following link provides best practices on writing and testing container images that can be used on OpenShift Origin. Once you have created an image, you can push it to the internal registry [13].

In the context of the ProsocialLearn project, a local instance of OpenShift Origin has been installed in the ATOS premises. The link [14] is the OpenShift Origin console gaming developers use to deploy their games (complex architectures) in the ProsocialLearn infrastructure.

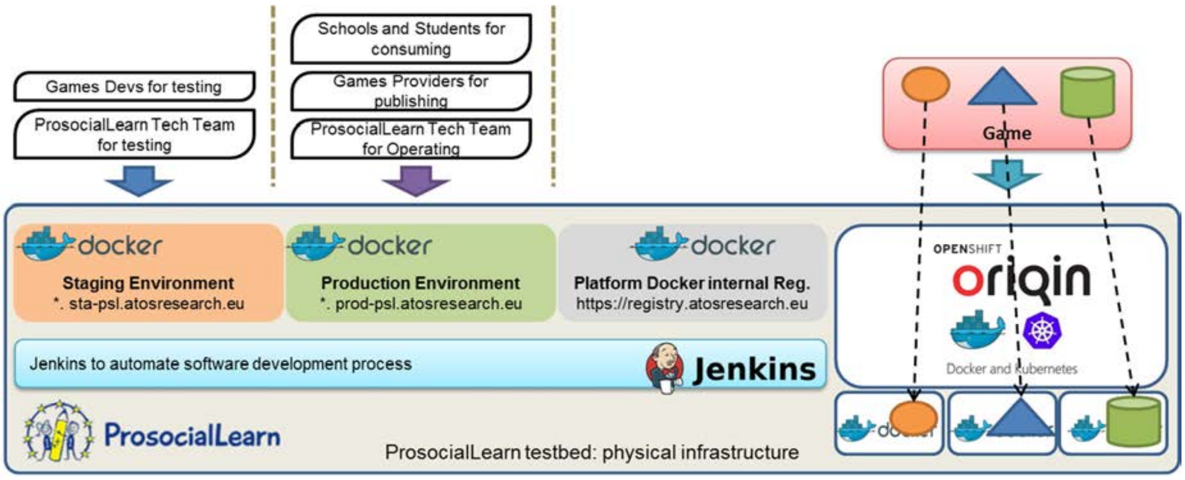

Games Devs for testing
ProsocialLearn Tech Team for testing
Schools and Students for consuming
Games Providers for publishing
ProsocialLearn Tech Team for Operating
Game
docker
Staging Environment
*. sta-psl.atosresearch.eu
Production Environment
*. prod-psl.atosresearch.eu
Platform Docker internal Reg.
https://registry.atosresearch.eu
OPENSHIFT origin
Docker and Kubernetes
Jenkins to automate software development process
Jenkins
ProsocialLearn
ProsocialLearn testbed: physical infrastructure

Figure 7 - Developers deploy their games

3.2 Testing Environment

To allow the 5 gaming providers belong to the ProsocialLearn consortium will quickly deploy and run their games (using a DEVOPS approach) the consortium also provided a way to deploy the games directly in the staging infrastructure.

At this scope Annex E provides a short and easy to follow guide to help gaming developers to deploy services in the staging infrastructure.

Once gaming developers successfully deployed their games (as one or more Docker images in the ProsocialLearn infrastructure), they can use the facilities provided by the platform to operatively test their games (while they interact with the services provided by the platform) before to publish them in the ProsocialLearn Marketplace.

Before to access to the aforementioned testing facilities, gaming providers have to register to the ProsocialLearn portal (see Figure 9 and Figure 10).

After a successful login, developers have to click on the Test button (see Figure 8).

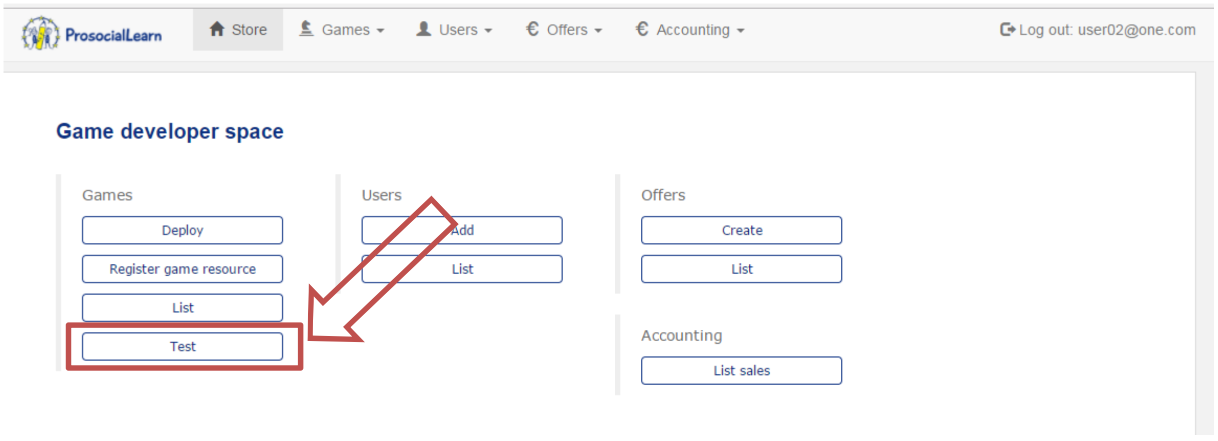


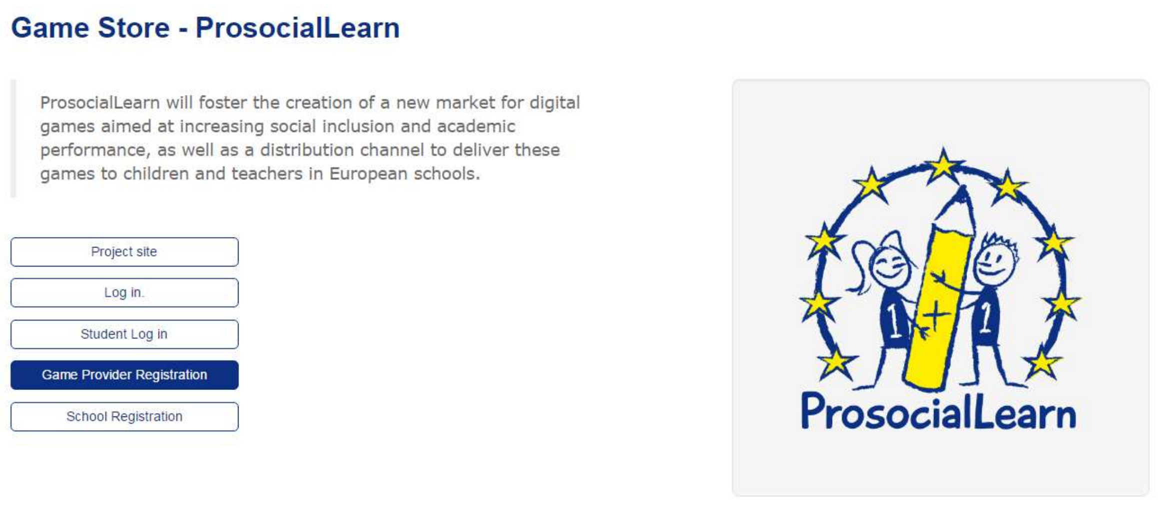
Figure 8 - Gaming Developers space

4 Game publication to the ProsocialLearn Marketplace

The ProsocialLearn Marketplace provides a Register Entity operation for registering gaming providers to the gaming market store. Gaming Providers to register to the store have to browse the ProsociaLearn Store Homepage[11] and click on "Game Provider Registration" (Figure 9 and Figure 10). The Platform Operator has the responsibility to validate Gaming Providers registration to allow them to operate through the portal.

Once Gaming Providers have been authorized to operate, they can log-in to the Store and access to the game testing page clicking on the Test button in Figure 8 (Section 3.2).

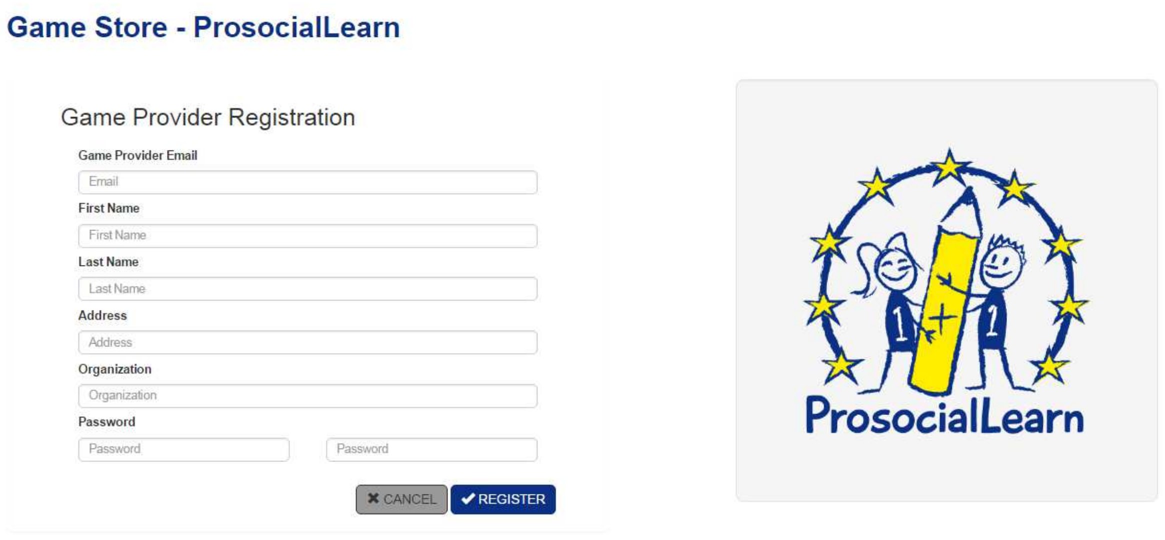
The page provide a testing environment that (Figure 11) simulates the work done by teachers (create Learning Group, Lesson Plan and Lesson) to allow gaming developers to easily/quickly test the interaction among the game and the platform during the runtime of the game.


Game Store - ProsocialLearn

ProsocialLearn will foster the creation of a new market for digital games aimed at increasing social inclusion and academic performance, as well as a distribution channel to deliver these games to children and teachers in European schools.

Figure 9 - ProsociaLearn Store Homepage

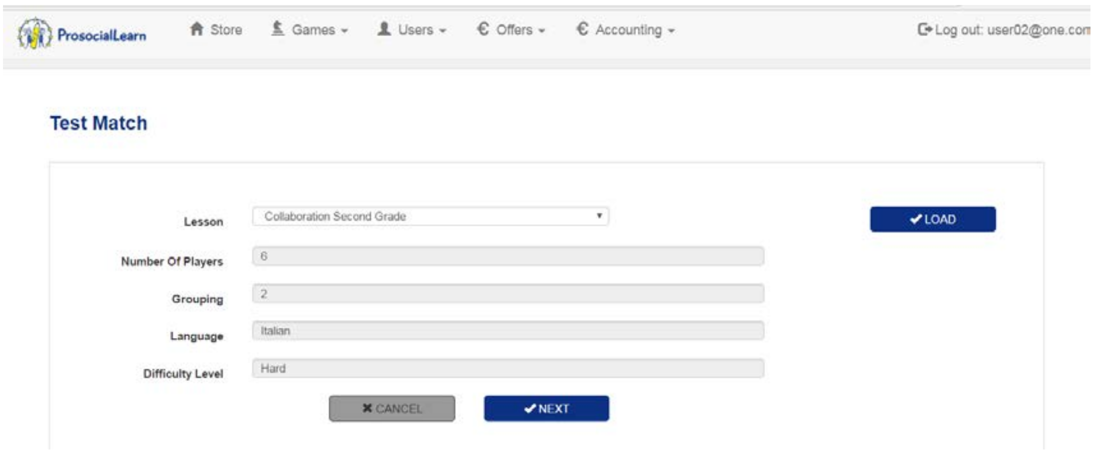
[11] http://sta-psl.atosresearch.eu/store/


Game Store - ProsocialLearn

Game Provider Registration

Game Provider Email	Email
First Name	First Name
Last Name	Last Name
Address	Address
Organization	Organization
Password	Password / Password

Figure 10 - Gaming Provider Registration page



Test Match

Lesson	Collaboration Second Grade
Number Of Players	6
Grouping	2
Language	Italian
Difficulty Level	Hard

Figure 11 - Game testing page

Once gaming developers have full tested games they can create an offer in the ProsocialLearn Store (MarketPlace). The Store provides to Game providers/developers easy to use space to register new resources (games) in the ProsocialLearn space and create (after receiving the formal authorization from the Platform Operator) personalize offers (game packages) to specific customers (Figure 12 and Figure 13).

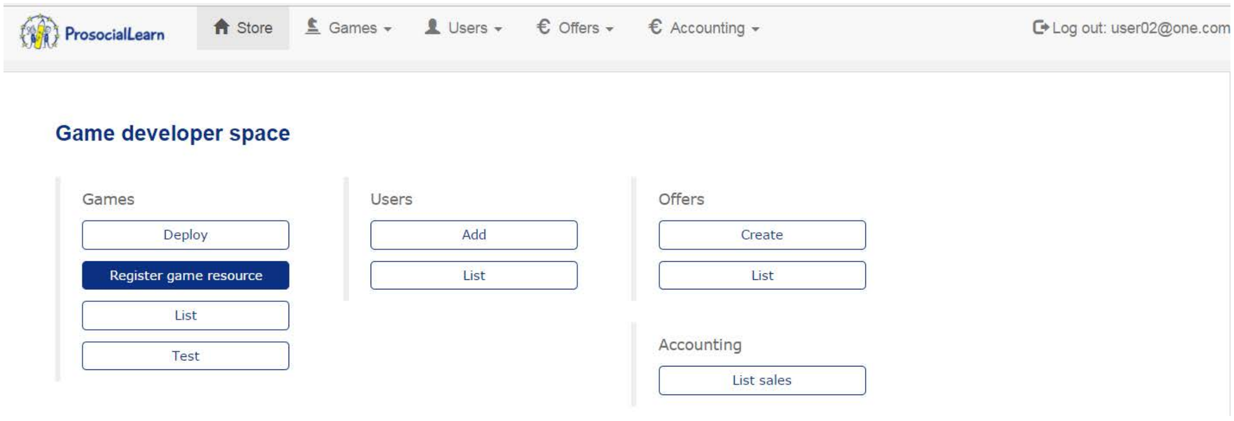
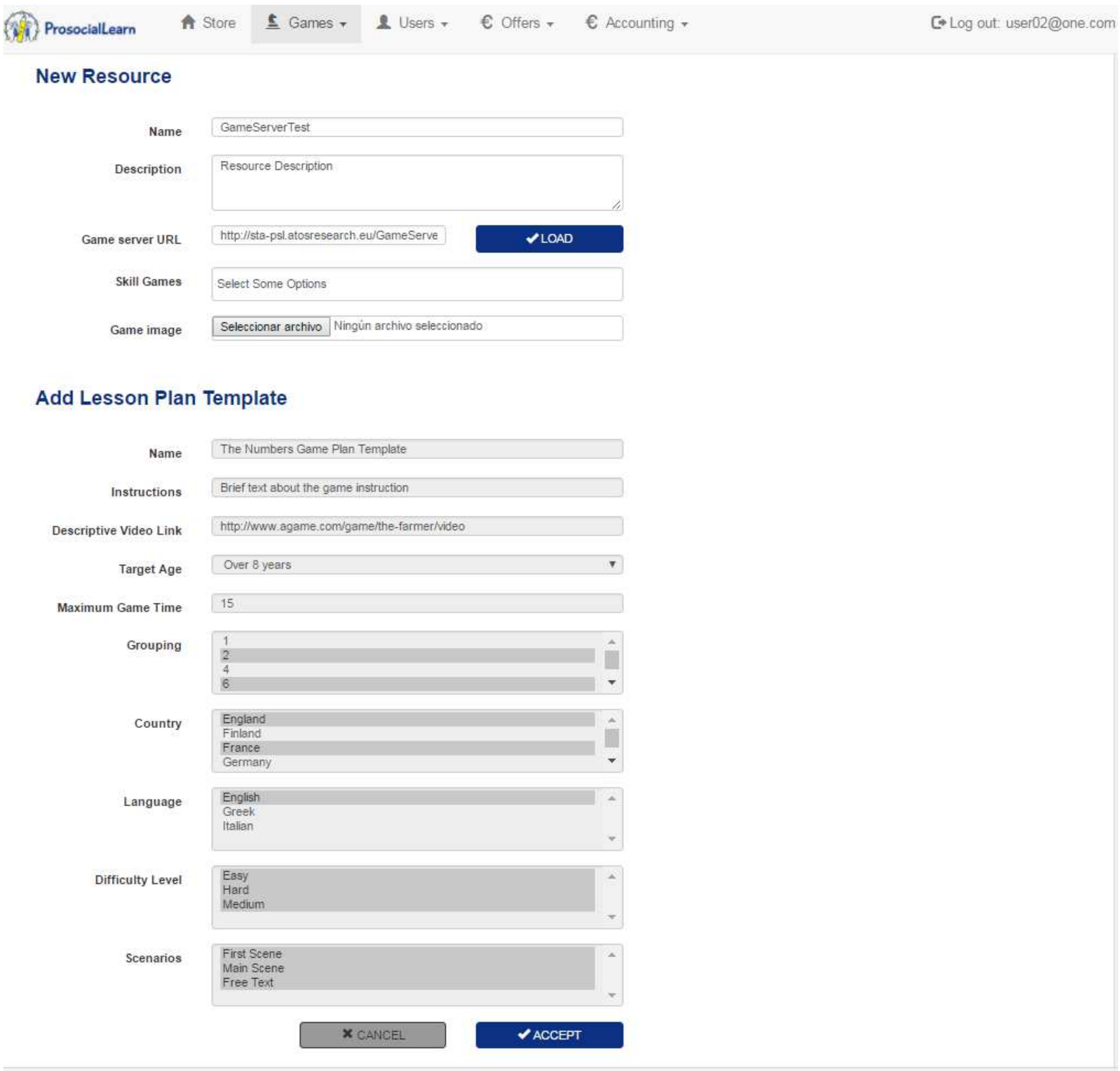


Figure 12 - Register Game Resource [1/2]



New Resource

Name	GameServerTest
Description	Resource Description
Game server URL	http://sta-psl.atosresearch.eu/GameServe
Skill Games	Select Some Options
Game image	Ningún archivo seleccionado

Add Lesson Plan Template

Name	The Numbers Game Plan Template
Instructions	Brief text about the game instruction
Descriptive Video Link	http://www.agame.com/game/the-farmer/video
Target Age	Over 8 years
Maximum Game Time	15
Grouping	1, 2, 4, 6
Country	England, Finland, France, Germany
Language	English, Greek, Italian
Difficulty Level	Easy, Hard, Medium
Scenarios	First Scene, Main Scene, Free Text

Figure 13 - Register Game Resource [2/2]

5 Conclusions

The main objective of the deliverable D5.6 “1st Platform operations report” has been to create a guide enabling gaming developer to learn about how to design, implement, deploy and test prosocial games relying on the ProsocialLearn approach.

Five gaming providers (PlayGen Ltd, REDIKOD AB, ANIWAY OY, HYPESLUGS SRL and MAD ABOUT PANDAS UG) have been adopting the ProsocialLearn platform and infrastructure both for integrating the main functionalities the platform provide and deploy their games in a secure and scalable cloud environment.

Intentionally, the report does not detail interactions with schools’ actors. This will be presented in the upcoming WP5 deliverable: D5.7 – 2nd Platform operations report. This report will be delivered by the end of the project (M36, December of 2017).

At this scope D5.6 contains a basic description of the ProsocialLearn platform and an easy-to-follow procedure gaming providers have to follow to create their games to be managed by the ProsocialLearn platform.

To successfully drive and support this integration work the consortium defined a series of activities (organized in a well-defined roadmap) game developers have to implement.

To effectively monitor the implementation of these activities, the consortium will rely on the issue tracking features of the Project GitLab.

Moreover the technical team created persistent chat rooms (channels) organized by topic (using the Slack Software). All content inside Slack is searchable, including files, conversations, and people.

6 Reference

[1] ProsocialLearn deliverable D2.4: 2nd System Requirements and Architecture (public): http://prosociallearn.eu/wp-content/uploads/2016/09/D2.4-2nd_System_Requirements_and_Architecture_Final_Version.pdf

[2] ProsocialLearn deliverable D3.2: 1st Prosocial affect fusion and player modelling: http://prosociallearn.eu/wp-content/uploads/2016/02/D3.2-1st%20Prosocial%20affect%20fusion%20and%20player%20modelling-Final%20version.pdf

[3] ProsocialLearn deliverable D3.3: 2nd Prosocial affect fusion and player modelling: http://prosociallearn.eu/wp-content/uploads/2016/09/D3.3_2nd_Prosocial_affect_fusion_and_player_modelling-final_version.pdf

[4] K. Kaza , A. Psaltis , K. Stefanidis , K. Apostolakis , S. Thermos , K. Dimitropoulos, P. Daras, "Body motion analysis for emotion recognition in serious games", HCI International, Toronto, Canada, July 2016.

[5] A. Psaltis, K. Kaza, K. Stefanidis, S. Thermos, K. Apostolakis, K. Dimitropoulos, P. Daras, "Multimodal affective state recognition in serious games applications", IEEE International Conference on Imaging Systems and Techniques, October 4-6, 2016.

[6] K. Apostolakis, K. Dimitropoulos, K. Stefanidis, A. Psaltis and P. Daras, D4.2 Intelligent Adaptation and Personalization, Feb. 2016

[7] ProsocialLearn JavaScript game client API: https://gitlab.atosresearch.eu/ari/prosociallearn/tree/master/WP3/PsL_JSClientAPI

[8] ProsocialLearn silly game: https://gitlab.atosresearch.eu/ari/prosociallearn/tree/master/WP5/GameServer/WebContent/game

[9] ProsocialLearn Deliverable D5.5 “3rd Prosocial platform release”.

[10] OpenShift Origin documentation: Developers Web Console Walkthrough https://docs.openshift.org/latest/getting_started/developers_console.html#getting-started-developers-console

[11] https://docs.openshift.org/latest/dev_guide/index.html

[12] Advantages of using docker: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/7.0_Release_Notes/sect-Red_Hat_Enterprise_Linux-7.0_Release_Notes-Linux_Containers_with_Docker_Format-Advantages_of_Using_Docker.html

[13] https://docs.openshift.org/latest/creating_images/index.html

[14] https://pslgames.atosresearch.eu:8443/console/

[15] Observations and Measurements standard: http://www.opengeospatial.org/standards/om

[16] Russell, J.A. (1980). A Circumplex Model of Affect. Journal of Personality and Social Psychology 39 (6): 1161–78

Annex A: ProsociaLearn JavaScript game client API

Introduction

This Annex describes the ProsocialLearn JavaScript game client API - it is intended for game developers to use. Within this package is a demonstration page (index.html) that contains active code to demonstrate the basic application of the API.

What does this API do for you?

This API allows you to:

- Access player and game instance meta-data from the ProsocialLearn platform
- Stream audio (voice) data to the ProsocialLearn platform for emotion analysis
- Stream visual (face) data to the ProsocialLearn platform for emotion analysis
- Access local time offset of your player relative to the ProsocialLearn platform time

Licence

The terms of use of this API are defined by the licence file entitled "ProsocialLearn Game Client API Licence.txt" found in the licences folder of this package.

Known issues

This is an alpha release of the API and has the following known issues:

- ProsocialLearn back-end services that support this API are partially implemented and may be unavailable
- Time-stamps included in voice/face emotion data streams are not yet synchronized
- Validated & secured communications between the client and the platform are not yet implemented
- Browser support is currently limited to Chrome v55+

These issues will be resolved in a future release.

Prerequisites

The following important prerequisites must be met for the correct operation of this web page and the ProsocialLearn game client API:

- Your computer can access the Internet and ProsocialLearn platform
- Your computer has either sound recording facilities (microphone) or video recording facilities (a webcam)
- This web page is being served to you by a web server (such as NGINX)
- You are viewing this page in Chrome

Getting started

Follow these steps to get started:

- Acquire the API from ProsocialLearn
- Copy the 'pslLibs' folder into the Javascript development folder you are working in.
- Add the following script reference to your web page: `<script src="<..your js folder..>/pslLibs/PSLClientAPI.js"></script>`

- Load your web page and set up some test user data by executing the following on the web page console: __createTestUserCookie();

You are now ready to start working with the ProsocialLearn JavaScript game client API.

Initialising the API

To initialise the API, you should instansiate a new instance of the API and then initialise it with the appropriate configuration for voice and/or face emotion capture (as required).

- Initialising for voice emotion capture

```
initialise( voiceConfig, null, initialiseCallback );
{
pslClient = new PSLClient();
var vConfig = { trackingCallback : onVoiceData };
pslClient.initialise( vConfig, null, onAPIInitialised );
...
function onAPIInitialised( result )
{
// result === 'OK'?
}
...
function onVoiceData( result )
{
// result === Float32Array
}
}
```

In the code above we see a new PSLClient instance is created and then initialised just for voice emotion detection. For voice configuration, you only need supply one parameter: 'trackingCallback' which points to a function allowing you to access captured audio data locally - this callback is optional and can be null. Above we supply 'null' for the second parameter to indicate that face based emotion detection is not needed. Finally, we point to our initialisation call-back function 'onAPIInitialised(..)' that gets called when the API has finished getting started - success is indicated with an 'OK' string.

- Initialising for face emotion capture

```
initialise( null, faceConfig, initialiseCallback );
pslClient = new PSLClient();
var fConfig = { videoID : 'myVideoInputDIV_ID',
                canvasID : 'myVideoOutputDIV_ID',
                width: 320,
                height: 240,
                trackingCallback : onFaceData };
pslClient.initialise( null, fConfig, onAPIInitialised );
...
function onAPIInitialised( result )
{
// result === 'OK'?
}
```

```
...
function onFaceData( result )
{
// result === array of ANEW-VA scores
}
```

In this example, we set up just for detecting emotion in the face via an attached web camera. Here configuration requires a few more elements, namely:

Table 3. Video Configuration Parameters

config param(s)	specification
videoID	ID of the `<video>...</video>` element used to render input captured from the web cam. A typical example of this would be: `<video id="videoInput" width="320" height="240"><source></video>`
canvasID	ID of the `<canvas>...</canvas>` element used to render the augmented face feature analysis elements. A typical example of this would be: `<canvas id="faceTrackOutput" width="320" height="240"></canvas>`
width, height	The width and height in pixels of the video stream input (in our example, this would be 320 and 240).
trackingCallback	A function that receives the emotion classification values as defined by the ANEW V-A space values. See the local callback function section below. This parameter can be null.

In this case, when the initialisation function is called, we supply null for the voice configuration to indicate that voice based emotion capture is not required; again we use the 'onAPIInitialised(..)' function to get news of the result of the initialisation.

Note: if you wish to use both voice and face emotion channels, simply supply both configuration objects in the same initialisation call.

Starting and stopping capture

trackVoice(true); trackFace(true); shutdown();

Once the API has been initialised, you can start and stop capture easily using the following methods:

pslClient.trackVoice(<boolean>);

Use this method to start tracking voice: true to start tracking, false to stop tracking.

pslClient.trackFace(<boolean>);

Use this method to start tracking the face: true to start tracking, false to stop tracking.

pslClient.shutdown();

Use this method to shutdown the client.

Game information

getGameInfo()

It is expected that your game will be deployed within the PsL platform and that players will gain access to it via the PsL portal. By the time a user has reached your game web page, the PsL platform will have information about who they are and which run-time instance of your game they will be linked with. The PsL Client API provides you with this information when you call the 'gameGameInfo()' method, returning:

```
{
playerNickName  : nick-name of player (string),
playerID        : unique identifier of player (UUID),
gameInstanceID  : unique identifier of game instance (UUID)
localTimeOffset : offset (in milliseconds) of your local clock relative to
the PsL server
}
```

Note: if you are running this page locally or on a third party web server, this information is not directly provided to you via the platform. There are two work-arounds to generate some dummy data for this purpose:

Call the __createTestUserCookie() function provided in our API

or

Visit the PsL game store and then
Click 'Game Provider'
Click 'Test Game'
Enter the public URL of your game front-end
Enter a nick name
Click start

Local callbacks

Local callbacks for locally accessing data generated for both voice and face are supported; you supply these callbacks during initialisation. The callback signatures are described below.

onVoiceCBFunc(result)
The result parameter is a Float32Array containing normalised floating-point PCM values (sampled at 44.1Khz).

onFaceCBFunc(result)
The result parameter is an array of ANEW-VA scores.

Annex B: Prosocial Game Development Starter Kit

Introduction

This page describes the steps Game Developers have to pass through in order to create a compatible game to be managed by the ProsocialLearn platform.

Code

Source code of the example web game:
https://gitlab.atosresearch.eu/ari/prosociallearn/tree/master/WP5/GameServer/WebContent/game

Steps

1. Implement a GetLessonTemplate interface to return the Lesson Template Plan Json
2. Put the game in GameServer/WebContent/game
3. Import the PSLClientAPI.js in our game:
4. Initialise the sensors and call this function from your game:

```
function initialiseSensors() {
  pslClient = new PSLClient();
  var fConfig = {
    videoID: 'videoInput',
    canvasID: 'faceTrackOutput',
    width: 320,
    height: 240,
    trackingCallback: onFaceData
  };
  var vConfig = {
    trackingCallback: onVoiceData
  };
  pslClient.initialise(vConfig, fConfig, onBothAPIInitialised);
}
```

5. Use the data collected. (in this case we are only showing it on console)

```
function onBothAPIInitialised(result) {
  console.log("Initialising feedback: " + result);

  if (result === "OK") {
    timedText("Initialising feedback: [" + result + "]",50);
```

```
        __createTestUserCookie();
        var gameInfo = pslClient.getGameInfo();
        console.log(gameInfo);

        pslClient.trackFace(true);
        pslClient.trackVoice(true);
    } else {
        timedText("(" + result + ")",20);
    }
}

function onFaceData(result) {
    console.log(result);
}

function onVoiceData(result) {
    console.log(result);
}
```

Annex C: ProsocialLearn Infrastructure

Chapter 5 of [1] details the scalability requirements of the ProsocialLearn platform.

Annex C details the Hardware and Software characteristics of the infrastructure support the runtime of the ProsocialLearn platform.

Hardware

RACKSPACE

19" Secure Rackspace (2U)

Location: AMS-01

DEDICATED EQUIPMENT LEASE

HP DL380pG8 (12xLFF)

CPU: 2x Intel Octa-Core Xeon E5-2650

RAM: 128GB DDR3

Hard Disk: 2x4TB SATA2;

RAID 1 (Mirror)

Network

IP CONNECTIVITY

1 x 1000Mbps Full-Duplex

100Mbps Unmetered

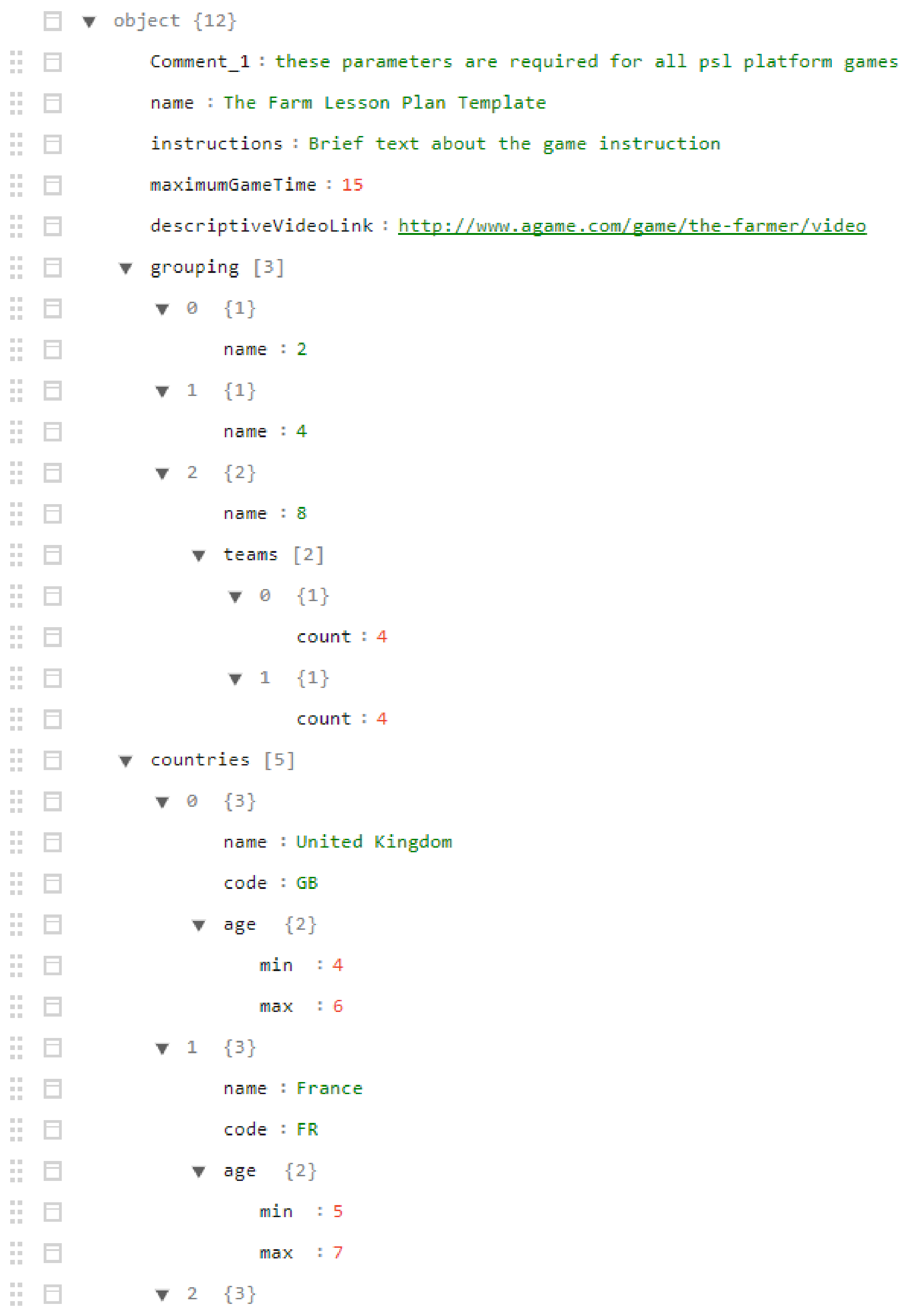
Public IPv4: 8

SOFTWARE

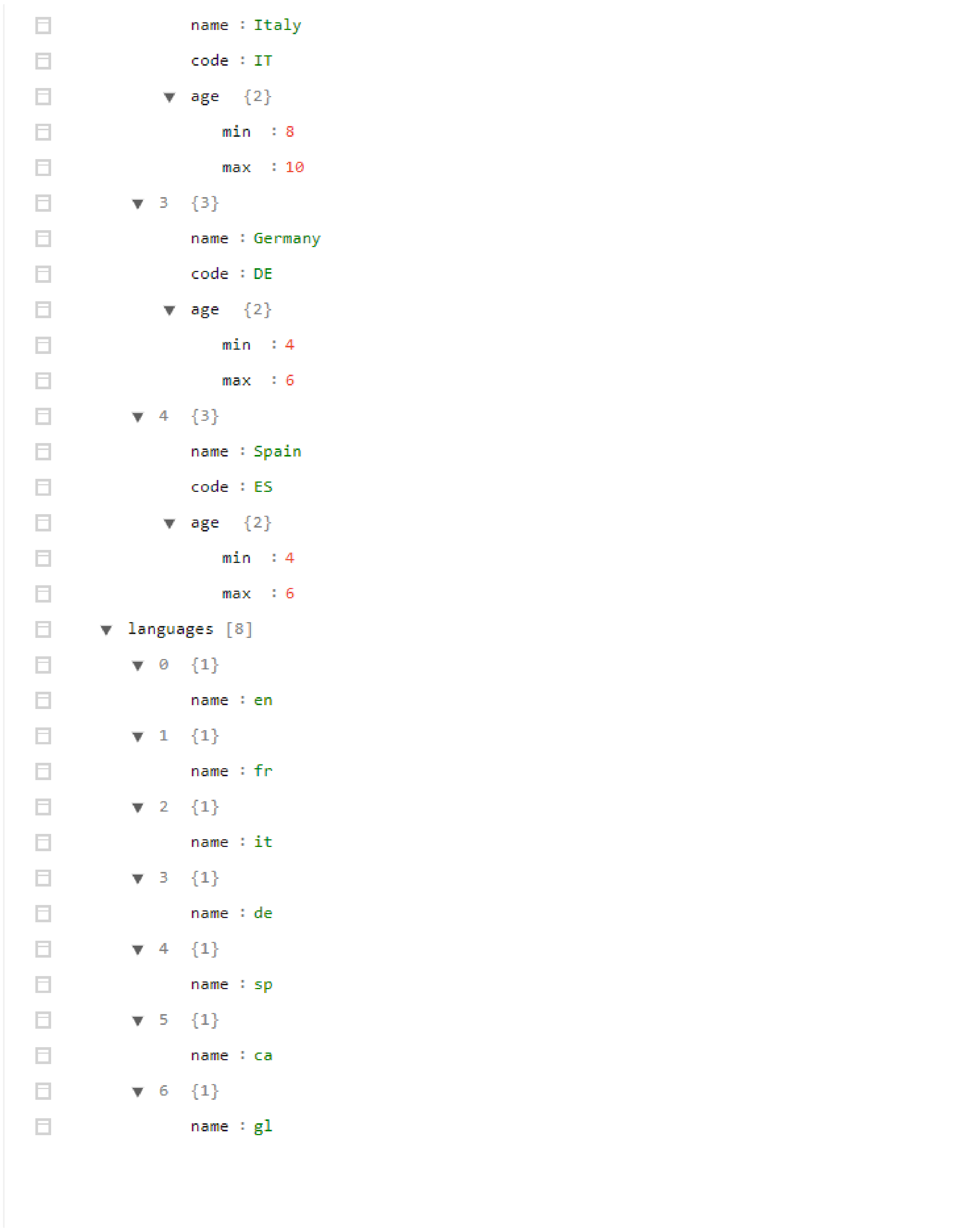
CentOS 7 64bit

Annex D: Lesson Plan Template Json

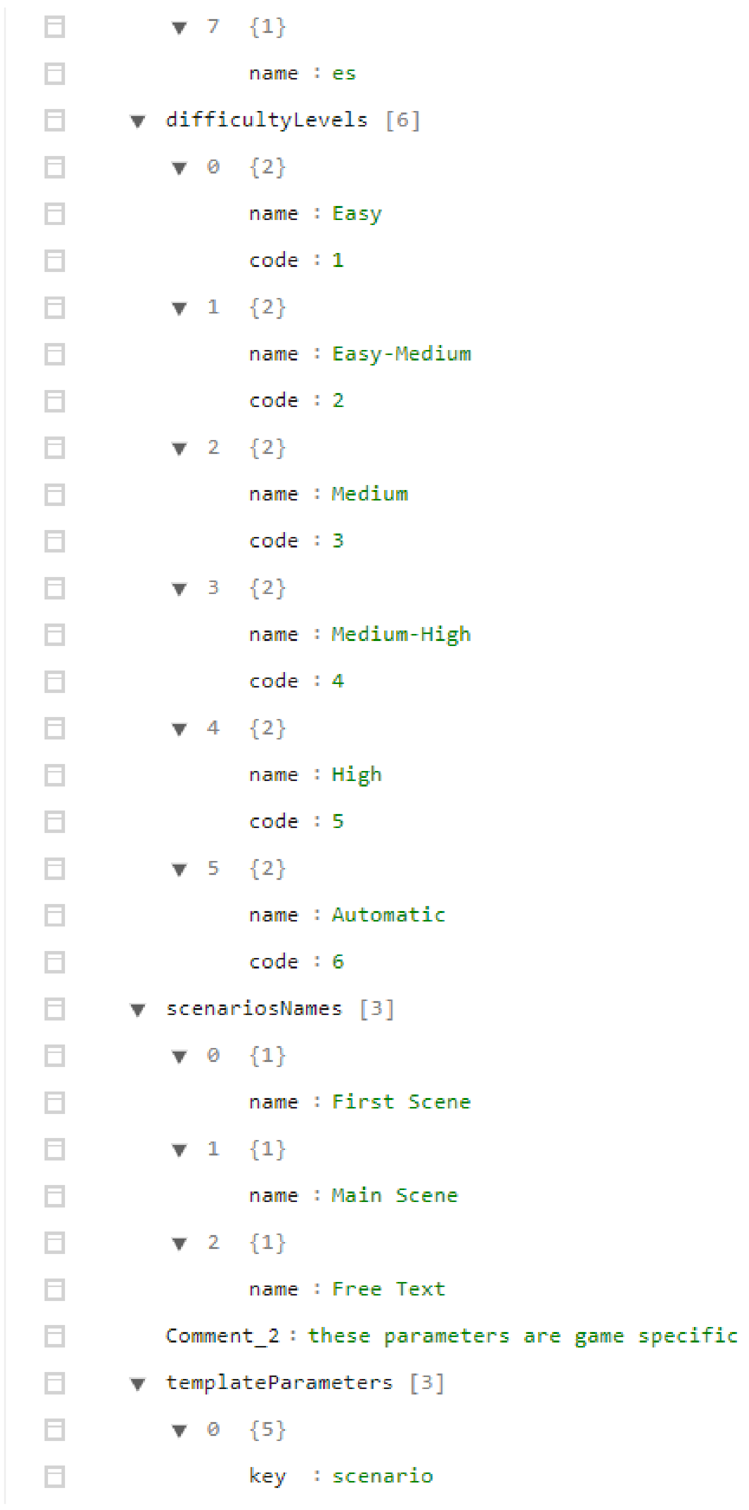
Below, the JS code to implement the getLessonPlanTemplate interface on game server side.



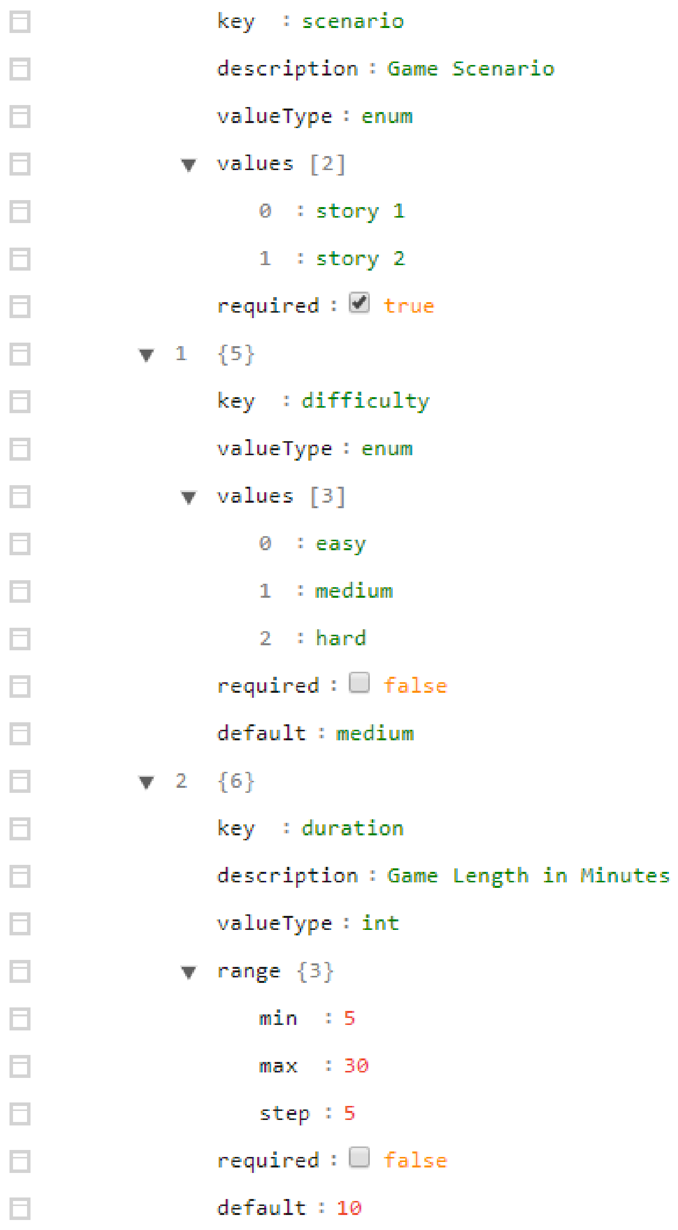
```
▼ object {12}
    Comment_1 : these parameters are required for all psl platform games
    name : The Farm Lesson Plan Template
    instructions : Brief text about the game instruction
    maximumGameTime : 15
    descriptiveVideoLink : http://www.agame.com/game/the-farmer/video
  ▼ grouping [3]
    ▼ 0  {1}
        name : 2
    ▼ 1  {1}
        name : 4
    ▼ 2  {2}
        name : 8
      ▼ teams [2]
        ▼ 0  {1}
            count : 4
        ▼ 1  {1}
            count : 4
  ▼ countries [5]
    ▼ 0  {3}
        name : United Kingdom
        code : GB
      ▼ age  {2}
          min  : 4
          max  : 6
    ▼ 1  {3}
        name : France
        code : FR
      ▼ age  {2}
          min  : 5
          max  : 7
    ▼ 2  {3}
```



```
            name : Italy
            code : IT
          ▼ age  {2}
              min  : 8
              max  : 10
      ▼ 3  {3}
            name : Germany
            code : DE
          ▼ age  {2}
              min  : 4
              max  : 6
      ▼ 4  {3}
            name : Spain
            code : ES
          ▼ age  {2}
              min  : 4
              max  : 6
  ▼ languages [8]
      ▼ 0  {1}
            name : en
      ▼ 1  {1}
            name : fr
      ▼ 2  {1}
            name : it
      ▼ 3  {1}
            name : de
      ▼ 4  {1}
            name : sp
      ▼ 5  {1}
            name : ca
      ▼ 6  {1}
            name : gl
```



```
        ▼ 7  {1}
              name : es
    ▼ difficultyLevels  [6]
        ▼ 0  {2}
              name : Easy
              code : 1
        ▼ 1  {2}
              name : Easy-Medium
              code : 2
        ▼ 2  {2}
              name : Medium
              code : 3
        ▼ 3  {2}
              name : Medium-High
              code : 4
        ▼ 4  {2}
              name : High
              code : 5
        ▼ 5  {2}
              name : Automatic
              code : 6
    ▼ scenariosNames  [3]
        ▼ 0  {1}
              name : First Scene
        ▼ 1  {1}
              name : Main Scene
        ▼ 2  {1}
              name : Free Text
      Comment_2 : these parameters are game specific
    ▼ templateParameters  [3]
        ▼ 0  {5}
              key  : scenario
```



```
          key  : scenario
          description : Game Scenario
          valueType : enum
        ▼ values [2]
             0  : story 1
             1  : story 2
          required : ☑ true
    ▼  1  {5}
          key  : difficulty
          valueType : enum
        ▼ values [3]
             0  : easy
             1  : medium
             2  : hard
          required : ☐ false
          default : medium
    ▼  2  {6}
          key  : duration
          description : Game Length in Minutes
          valueType : int
        ▼ range {3}
             min  : 5
             max  : 30
             step : 5
          required : ☐ false
          default : 10
```

Annex E: How-to create a Docker image of a service to be deployed in the ProsocialLearn Infrastructure

Annex E gives an overview to how-to create a Docker image of a service to be deployed in the ProsocialLearn Infrastructure. The following steps needs to be completed in order to create the environment to integrate the new games to Prosociallearn Game platform.

- "Dockerize" your game
- Push the container to Prosociallearn internal Docker registry
- Manage Jenkins job to deploy the game in Prosociallearn staging platform

Prologue

ProsocialLearn is a secure environment, so it is protected by certificates. To allow the ProsocialLearn security layer works fine you need to install/deploy (first of all) your game in the ProsocialLearn infrastructure. Moreover to follow the ProsocialLearn approach games have to be deployed in a Docker image. The follow guide will teach you how to deploy your game (first of all) in the ProsocialLearn infrastructure. The second and final step of the development will be to use the OpenShift tool.

"Dockerize" your game

To "dockerize" your game, we have prepared a guideline depending on the technology used in the frontend/backend of your game.

Requirements:

- Docker Community Engine. Installation tutorial[12]

How to:

First create a file with the name "**Dockerfile**" inside your project folder with the following information:

1. Unity WebGL + Nodejs

```
FROM node:6.10.3
MAINTAINER <developer_name_or_organization> contact@email.com
EXPOSE 80
COPY . /usr/src/app
WORKDIR /usr/src/app
CMD ["node", "your-daemon-or-script.js"]
```

In this link you can see the **best practises** to create your Dockerfile.

2. Phaser.io + Apache

```
FROM nginx
MAINTAINER <developer_name_or_organization> contact@email.com
EXPOSE 80
WORKDIR /usr/share/nginx/html
ADD . /usr/share/nginx/html/
```

In this link you can see the **best practises** to create your Dockerfile.

[12] https://store.docker.com/editions/community/docker-ce-server-ubuntu

Second, build your container. In the same folder where the **Dockerfile** is located, run the following command:

```
docker build -t registry.prosociallearn.eu:5043/container_game_name .
```

Push the container to Prosociallearn internal Docker registry

Once you have created the container, the next step is push it to prosociallearn docker internal registry.

```
Docker push registry.prosociallearn.eu:5043/container_game_name
```

In case of you get this error: "**unauthorized: authentication required**", please run the following command to login prosociallearn docker registry.

```
Docker login -u=pslregistry -p=JALQbpBvmz6cv3dW registry.prosociallearn.eu:5043
```

Container versioning **(Optional)**. To keep and deploy different versions of your Docker containers, the strategy is to **tag** the container and push it to prosociallearn Docker internal registry. The command to add the version tag to containers is the following:

```
Docker tag registry.prosociallearn.eu:5043/container_game_name:latest registry.prosociallearn.eu:5043/container_game_name:version_tag
```

After that you can push the new tagged container to prosociallearn Docker internal registry with the command:

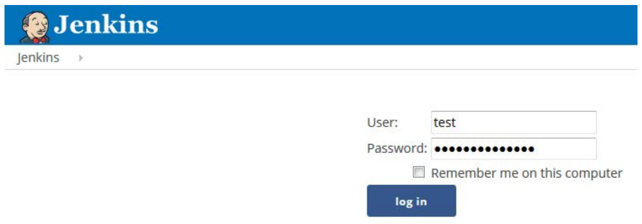
```
Docker push registry.prosociallearn.eu:5043/container_game_name
```

Manage Jenkins job to deploy the game in Prosociallearn staging platform

1. How to login?

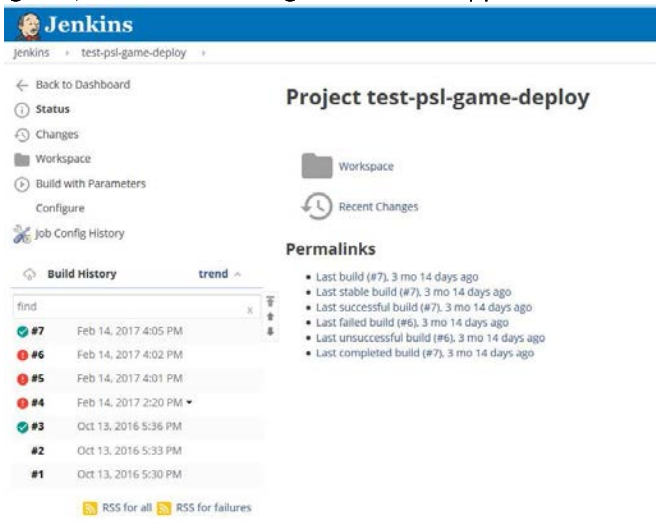
Go to the webpage: http://jenkins.prosociallearn.eu/

Insert your User and Password and click on "**log in**".



2. How to deploy a new version of the game in staging?

It is simple, click on the game, and the following window will appear:



Jenkins

Jenkins › test-psl-game-deploy ›

- Back to Dashboard
- Status
- Changes
- Workspace
- Build with Parameters
- Configure
- Job Config History

Build History trend

find

#7	Feb 14, 2017 4:05 PM
#6	Feb 14, 2017 4:02 PM
#5	Feb 14, 2017 4:01 PM
#4	Feb 14, 2017 2:20 PM
#3	Oct 13, 2016 5:36 PM
#2	Oct 13, 2016 5:33 PM
#1	Oct 13, 2016 5:30 PM

RSS for all RSS for failures

Project test-psl-game-deploy

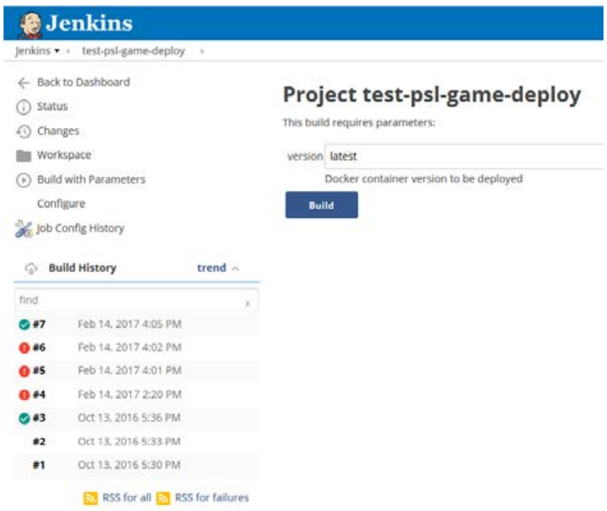
Workspace

Recent Changes

Permalinks

- Last build (#7), 3 mo 14 days ago
- Last stable build (#7), 3 mo 14 days ago
- Last successful build (#7), 3 mo 14 days ago
- Last failed build (#6), 3 mo 14 days ago
- Last unsuccessful build (#6), 3 mo 14 days ago
- Last completed build (#7), 3 mo 14 days ago

3. Next click on "**Build with Parameters**" and write in the version field the Docker container tag version to be used and push the Build button.



Jenkins

Jenkins › test-psl-game-deploy ›

- Back to Dashboard
- Status
- Changes
- Workspace
- Build with Parameters
- Configure
- Job Config History

Build History trend

find

#7	Feb 14, 2017 4:05 PM
#6	Feb 14, 2017 4:02 PM
#5	Feb 14, 2017 4:01 PM
#4	Feb 14, 2017 2:20 PM
#3	Oct 13, 2016 5:36 PM
#2	Oct 13, 2016 5:33 PM
#1	Oct 13, 2016 5:30 PM

RSS for all RSS for failures

Project test-psl-game-deploy

This build requires parameters:

version: latest

Docker container version to be deployed

Build

4. Afterwards, you can go to the url assigned to your game and play with it. The url will be something like this: **https://sta-psl.atosresearch.eu/gamename.**