LAMS and 3rd Party App Integration APIs

LAMS Integrations

Introduction

LAMS provides the API described in this page to create rich and powerful interaction with any 3rd party system. The aim for this integration is to make LAMS appear into an LMS system like yet another native tool of the LMS.

LAMS2 has been integrated with:

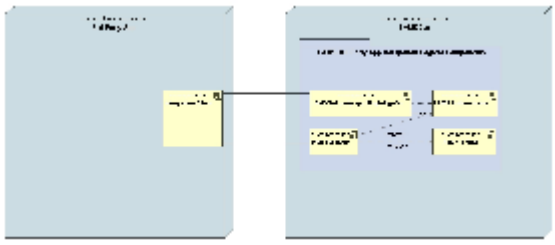
- Blackboard
- Moodle | Check out the LAMS-Moodle demo
- Sakai
- And any other that uses IMS LTI

In this document, we present a technical overview on how these integrations are implemented. Although each 3rd party applications are different (even they are in different programming languages) the principles that we follow are essentially the same.

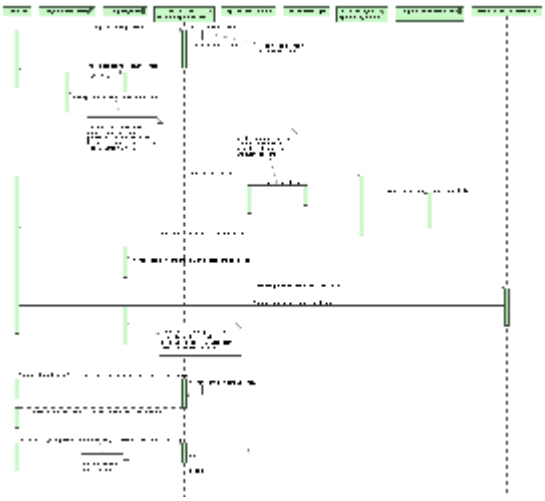
LAMS2 Side

The following diagrams give the overview and the details of the LAMS 2.x/3rd party app SingleSignOn implementation:

- overview diagram:



- sso sequence diagram:



User Information

Every time a LMS user attempts to access LAMS and there's no personal details passed, LAMS will check to see if the user exists in its database. If the user has not been created in LAMS, LAMS will make a call to the external application to retrieve user information and create the user on the fly. Therefore you will need to set up a user information servlet on the LMS side that will return user information to LAMS.

Note that the callback URL will only take place if the firstName, lastName parameters were **not** passed with the LoginRequest, LearningDesignRepository or LesssonManager (join method). See below.

If you are passing the user details as parameters in the requests, then there will be no call back to the LMS and the user will be created with the details passed.

Request format

The user information url is defined in the LAMS System Administration page and contains three parameters: username, timestamp and hash.

For example, the url for moodle is given by:

http://localhost/moodle/mod/lamstwo/userinfo.php?ts=%**timestamp**%&un=%**username**%&hs=%**hash**%

The hash value is generated using the SHA1 algorithm on the following (all in lower case)
[timestamp + username + serverID + serverKey]

Response Format

The response from the external application is a comma separated list of 14 values

<Title>,<First name>,<Last name>,<Address>,<City>,<State>,<Postcode>,<Country>,<Day time number>,<Mobile number>,<Fax number>,<Email>,
<Locale language>,<Locale country>

Group Information

In order to enable import and usage of the groups created on the integrated server (*Blackboard only*), one needs to go to the LAMS System Administration page and enter External groups URL property following the next format:

https://<BB_SERVER>:8443/webapps/lams-lamscontent-BBLEARN/GroupData

After this one be able to select and use external groups at the monitoring page of the lesson created via that integrated server.

External calls to LAMS

Login Requests

Request for author, learner and monitor are directed through the *LoginRequest* servlet. For example:

http://localhost/lams/LoginRequest?....

The parameters are:

Parameter	Description	Restrictions/conditions
uid	The username on the external system	Must be unique. Only alphanumeric characters and these symbols allowed "_" "-")
ts	Number of milliseconds since January 1, 1970, 00:00:00 GMT (known as "the epoch") till the time when the call is made	Required if according integrated server has "Enforce time limitation" setting ON. And if it's OFF it can be a random String.
sid	serverID (see integration configuration)	
method	either author, monitor, learner or learnerStrictAuth; it sets permissions and redirects user to requested content	Must match one of the methods' name
hash	A SHA1 hash of [ts + uid + method + serverID + serverKey] (Note: all lower case before hashing) If you are using the learnerStrictAuth, you must include the lsId in the hash (see below)	
mode	Additional directive: "preview" for previewing lesson in learner, "gradebook" to access user gradebook after logging in	
courseid	the id of the course from the LMS	Must be unique for each LMS course
lsid	Used for monitor, learner and gradebook to tell LAMS the lesson id you want to open	Required if requesting access to monitor, learner or gradebook UIs
firstName	Used if a new user is created	No numbers or special characters permitted
lastName	Used if a new user is created	No numbers or special characters permitted. Only single quotes and spaces allowed ie: O'Harries or Garcia Marquez)
email	Used if a new user is created	Correctly validated email address required
country	Country (must use 2 letter code as ISO 3166-1 alpha-2 code for country)	
lang	Language code or locale code. In case of just passing language use "es" for Spanish or locale as "es_ES" (see list of available locales in LAMS)	
isUpdateUserDetails	"true" in case user details should get updated with new information, "false" (or no parameter) otherwise	
requestSrc	String to identify who is making the call, this is only used in authoring to display on the close button.	
notifyCloseURL	Callback URL to refresh the page after a sequences has been added.	

You can use either http method POST or GET.

Mapping external roles to LAMS course roles

LAMS course roles are learner, monitor and author.

Therefore passing a method with one of these roles will assign the user the following LAMS course roles

Method	Grants roles
Author	learner, monitor and author
Monitor	monitor
learner	learner
learnerStrictAuth	learner

The learnerStrictAuth method

Important!

Use the **learnerStrictAuth** method when launching students into lessons

(only available for LAMS 2.5+)

```
While you can use the method=learner to launch learners into a LAMS lesson, it is strongly recommended that you use the method=learnerStrictAuth instead.
```

When you use **learnerStrictAuth**, you also include the lessonID (lsId) that the user is being launched to into the authentication hash. This prevents any tampering and ensures that the user always goes to the correct lesson.

The main difference within the LoginRequest as explained before is that you use the method "learnerStrictAuth" and include the lsId in the hash as follows:

```
hashText = ToLowerCase([ts + uid + method + lsId +serverID + serverKey])
```

Then use sha1 to hash the hashText.

Getting LAMS server time and Time to live for LoginRequests

Note!

This feature is only available for LAMS 2.5+

In order for LMS to provide correct timestamp when making a call to LoginRequest it can use this servlet to acquire current LAMS server time.

It can be achieved by making a request to the GetServerTime servlet on the LAMS side. For example:

http://localhost/lams/services/getServerTime

Servlet will return the number of milliseconds since January 1, 1970, 00:00:00 GMT POSIX time. For example, 1417535668074 (which stand for Dec 02 11:24:28 VET 2014).

No parameters are expected.

You can use either http method POST or GET.

Why do I need to know the LAMS server time?

Good question, in LAMS 2.5 we wanted to add another level of security in case a user attempts to hijack a LoginRequest URL. On LAMS 2.5 there are two new options when creating an integrated server:

- Enforce time limitation for integration requests (checkbox) and
- Time to live for integration requests (in minutes)

If you enforce a time limitation, then the LoginRequest have a set number (say 5 mins) for this LoginRequest URL "to live". After that time, the URL will be rejected.

So you use getServerTime to find out the POSIX time that the LAMS app is using and then you can calculate the delta with yours. So next time you send a LoginRequest, if you have the "Enforce time limitation" checked, then you send the **ts** with the LAMS server time.

Getting Learning Designs

You will need to get a List of existing learning designs so the LMS user can choose which ones they wish to start.
You can get a list of learning designs by making a request to the LearningDesignRepository servlet on the LAMS side. For example:

http://localhost/lams/services/xml/LearningDesignRepository?

The parameters are:

Parameter	Description
username	The username of the current user. Must be unique
datetime	timestamp
serverId	the server id
hashValue	SHA1 hash of [timestamp + username + serverID + serverKey] (Note: all lower case before hashing)
courseId	the id of the course from the LMS

country	Country (must use 2 letter code as ISO 3166-1 alpha-2 code for country)
lang	Language code or locale code. In case of just passing language use "es" for Spanish or locale as "es_ES" (see list of available locales in LAMS)
mode	"1" returns all learning designs being monitored by the current user, "2" returns all learning designs authored by the user
firstName	used if a new user is created
lastName	used if a new user is created
email	used if a new user is created

You can use either http method POST or GET.

Starting, Stopping, Scheduling, Removing or Cloning a Lesson

To start, stop, schedule or clone a lesson in LAMS you need request another servlet:

http://localhost/lams/services/xml/LessonManager?

The parameters are:

- username - the username of the current user
- serverId - the server id
- datetime - timestamp
- hashValue - SHA1 hash of [timestamp + username + serverID + serverKey] (Note: all lower case)
- courseId - the id of the course from the LMS *(excluding removeLesson)*
- country - Country (must use 2 letter code as ISO 3166-1 alpha-2 code for country)
- lang - Language code or locale code. In case of just passing language use "es" for Spanish or locale as "es_ES" (see list of available locales in LAMS)
- method - "start", "stop", "schedule", "removeLesson"/"removeAllLessons" or "clone"
- ldId - the id of the learning design *(for start and schedule methods)*
- lsId - the id of the lesson *(for stop, removeLesson, clone)*
- title - the title of the lesson *(for start and schedule)*
- desc - the description of the lesson *(for start and schedule)*
- learnerEnableExport - enables exporting portfolio by learners. Optional, default value: false *(for start and schedule)*
- learnerSeeOnline - enables learner presence in the lesson. Optional, default value: false *(for preview, start and schedule)*
- learnerInstantMessaging - enables learner instant messaging in the lesson. Optional, default value: false *(for preview, start and schedule)*
- enableNotifications - enables "email notifications" link for the lesson. Optional, default value: false *(for start and schedule)*
- allowLearnerRestart - whether learners are allowed to restart the lesson. Optional, default value: according LAMS ExtServer's default setting _(for start and schedule)

This servlet returns an xml response depending on the method ('start', 'schedule' or 'stop').

- Start - An xml node called "Lesson" with an attribute "lessonId" which contains the lesson id.
- Schedule - An xml node called "Lesson" with an attribute "lessonId" which contains the lesson id.
- Stop - An xml node called "Lesson" with an attribute "deleted" which informs you whether the lesson was successfully deleted in LAMS
- Clone - An xml node called "Lesson" with an attribute "lessonId" which contains the new lesson id.

You can use either http method POST or GET.

Pre-adding students and monitors to a lesson

By default, the LoginRequest servlet will add students/monitors to an existing lesson automatically. But in some cases you might want/need to pre-add the students/monitors to the lesson

For this, you can use the LessonManager servlet and make only one call passing all usernames for all the students and monitors:

http://localhost/lams/services/xml/LessonManager?

The parameters are:

- username - the username of the current user
- serverId - the server id
- datetime - timestamp
- hashValue - SHA1 hash of [timestamp + username + serverID + serverKey] (Note: all lower case)
- courseId - the id of the course from the LMS
- country - Country (must use 2 letter code as ISO 3166-1 alpha-2 code for country)
- lang - Language code or locale code. In case of just passing language use "es" for Spanish or locale as "es_ES" (see list of available locales in LAMS)
- lsId - the id of the lesson
- method - "join"
- learnerIds - a comma separated string with all the students' usernames (ie: johnd,ernieg,fei,sarah)
- monitorIds - a comma separated string with all the monitors' usernames

You can use either http method POST or GET.

If the learners and monitors are to be created in LAMS, you must pass their user details following a few conventions:

- firstNames, lastNames, emails - these are comma separated parameters;
- if there is a need to create even only one user it is required to supply comma separated list of all users going to be joined to lesson (i.e.: if learnerIds=user1,user1). Then each of the new parameters should have 2 values, like firstNames=name1,name2.
- in case if you want to join monitors as well during one call - learner's parameters should go first, before monitor's ones
- if any of these 3 parameters(firstNames, lastNames, emails) is provided it is expected that all of them have the **same number of comma separated elements**

LAMS will **not** create the users if the number of usernames, firstnames and lastnames is not the same.

While you can use GET to post this,

Removing users from a lesson

Removes user(s) completely from a lesson regardless of his/her role with it.

Call LessonManager servlet: http://localhost/lams/services/xml/LessonManager?

The parameters are:

- username - the username of the current user
- serverId - the server id
- datetime - timestamp
- hashValue - SHA1 hash of [timestamp + username + serverID + serverKey] (Note: all lower case)
- lsId - the id of the lesson
- method - "removeUser" or "removeAllUsers"
- userIds - a comma separated string with all the users' usernames (ie: johnd,ernieg,fei,sarah) *(for removeUser only)*

Previewing a lesson

This is the same as starting a lesson. The only thing that changes here is that the title and the method is to be set to **preview**. Once you get the lessonId, just compose the LoginRequest as usual with mode = **preview**. That's all 😅

Get Design Image

In LAMS version **2.3.5+** (that's the latest on the lams2_3_release branch), you are able to get a learning design image (the learning design activity drawing) from an integration.

http://localhost/lams/services/LearningDesignSVG?
The parameters are:

- serverId - the server id
- datetime - timestamp
- hashValue - SHA1 hash of [timestamp + username + serverID + serverKey] (Note: all lower case)
- username - the username of the current user
- ldId - the id of the learning design (for starting or scheduling)
- svgFormat - image format (svgFormat can be either 1 for SVG or 2 for PNG)

The response is the image content (bytes), so you can set it as "src" attribute of a <img> tag.

You need to use http method GET.

Get user progress for a lesson

To get a user progress for a lesson, you must also request the LessonManagerServlet. You can either get a specific progress for a user, or a list of progresses for an entire lesson.

The parameters are:

- username - the username of the current user
- serverId - the server id
- datetime - timestamp
- hash - SHA1 hash of [timestamp + username + serverID + serverKey] (Note: all lower case)
- method - singleStudentProgress or studentProgress for the list of progresses
- lsId - the lesson id
- courseId - the course id
  There are 3 additional parameters for single user progress (method=singleStudentProgress)
- firstName - first name of the user who's progress you wish to retrieve
- lastName - last name of the user who's progress you wish to retrieve
- email - email of the user who's progress you wish to retrieve

You can use either http method POST or GET.

You will receive an xml response in the following form:

For single user progress:

```
<LearnerProgress
    lessonComplete="true"
    activityCount="4"
    activitiesCompleted="4"
    attemptedActivities="4"
    studentId="304"
    courseId="MATH101"
    username="testUser"
    lessonId="13"
>
</LearnerProgress>
```

For multiple user progresses:

```
<LessonProgress>

<LearnerProgress
    lessonComplete="true"
    activityCount="4"
    activitiesCompleted="4"
    attemptedActivities="4"
    studentId="304"
    courseId="MATH101"
    username="testUser"
    lessonId="13"
>
</LearnerProgress>

<LearnerProgress
    lessonComplete="false"
    activityCount="4"
    activitiesCompleted="2"
    attemptedActivities="3"
    studentId="114"
    courseId="MATH101"
    username="stephenh"
    lessonId="13"
>
</LearnerProgress>


</LessonProgress>
```

Get lessons where an user has Monitor role

To get a list of lesson for which the given user has Monitor role you must request LessonManagerServlet.

The parameters are:

- username - the username of the current user
- serverId - the server id
- datetime - timestamp
- hash - SHA1 hash of [timestamp + username + serverID + serverKey] (Note: all lower case)
- method - use "listMonitor"
- courseId - the course id

You will receive an xml response in the following form:

```
<Lessons>
  <Lesson lessonId="5" name="myLesson1"       description="math lesson" state="3" />
  <Lesson lessonId="8" name="someOtherLesson" description="eng lesson"  state="5" />
</Lessons>
```

Get tool outputs (marks) from a lesson

At times you might want to get the marks that the student(s) got in a LAMS activity -for instance the mark of a multiple choice.

For that you need to change the methods as follows:

1. method=toolOutputsAllUsers: This one gets all outputs for all activities and for all users, see toolOutputsAllUsers.xml
2. method=authoredToolOutputsAllUsers: This one gets the requested outputs (which were pre-defined in author in the learning design) for each activity and prints them out for each user. See authoredToolOutputsAllUsers.xml
3. method=toolOutputsUser: This one gets all the outputs for an activity, but only for the specified user, see toolOutputsUser.xml
4. method=authoredToolOutputsUser: This one gets the requested outputs (which were pre-defined in author in the learning design)for each activity, but just for the specified user, see authoredToolOutputsUser.xml

You can use either http method POST or GET.

Calling:
http://<lams-server>:8080/lams//services/xml/LessonManager?
&serverId=xxx&courseId=xxx&username=test1&datetime=xxx&hashValue=xxx&lang=en&country=AU&method=toolOutputsAllUsers&lsId=1

you get:

toolOutputsAllUsers.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ToolOutputs lessonId="1" name="mc">
    <LearnerOutput completedLesson="false" firstName="Mary" lamsUserId="4" lamsUserName="mmm" lastName="Morgan"
userName="mmm">
        <Activity activityId="6" attempted="true" completed="true" title="MC1">
            <ToolOutput description="Are learner's answers all correct?" name="learner.all.correct" output="
true" type="boolean" />
            <ToolOutput description="Learner's total mark" name="learner.mark" output="1" type="long" />
        </Activity>
        <Activity activityId="5" attempted="true" completed="false" title="MC2" />
        <Activity activityId="7" attempted="false" completed="false" title="MC3" />
    </LearnerOutput>
    <LearnerOutput completedLesson="false" firstName="Two" lamsUserId="6" lamsUserName="test2" lastName="Test"
userName="test2">
        <Activity activityId="6" attempted="true" completed="false" title="MC1" />
        <Activity activityId="5" attempted="false" completed="false" title="MC2" />
        <Activity activityId="7" attempted="false" completed="false" title="MC3" />
    </LearnerOutput>
    <LearnerOutput completedLesson="true" firstName="Four" lamsUserId="8" lamsUserName="test4" lastName="Test"
userName="test4">
        <Activity activityId="6" attempted="true" completed="true" title="MC1">
            <ToolOutput description="Are learner's answers all correct?" name="learner.all.correct" output="
false" type="boolean" />
            <ToolOutput description="Learner's total mark" name="learner.mark" output="0" type="long" />
        </Activity>
        <Activity activityId="5" attempted="true" completed="true" title="MC2">
            <ToolOutput description="Are learner's answers all correct?" name="learner.all.correct" output="
false" type="boolean" />
            <ToolOutput description="Learner's total mark" name="learner.mark" output="0" type="long" />
        </Activity>
        <Activity activityId="7" attempted="true" completed="true" title="MC3">
            <ToolOutput description="Are learner's answers all correct?" name="learner.all.correct" output="
false" type="boolean" />
            <ToolOutput description="Learner's total mark" name="learner.mark" output="0" type="long" />
        </Activity>
    </LearnerOutput>
    <LearnerOutput firstName="Jacky" lamsUserId="3" lamsUserName="lamskh01" lastName="Fang" userName="lamskh01">
        <Activity activityId="6" attempted="false" completed="false" title="MC1" />
        <Activity activityId="5" attempted="false" completed="false" title="MC2" />
        <Activity activityId="7" attempted="false" completed="false" title="MC3" />
    </LearnerOutput>
    <LearnerOutput completedLesson="true" firstName="One" lamsUserId="5" lamsUserName="test1" lastName="Test"
userName="test1">
        <Activity activityId="6" attempted="true" completed="true" title="MC1">
            <ToolOutput description="Are learner's answers all correct?" name="learner.all.correct" output="
true" type="boolean" />
            <ToolOutput description="Learner's total mark" name="learner.mark" output="1" type="long" />
        </Activity>
        <Activity activityId="5" attempted="true" completed="true" title="MC2">
            <ToolOutput description="Are learner's answers all correct?" name="learner.all.correct" output="
false" type="boolean" />
```

```
            <ToolOutput description="Learner's total mark" name="learner.mark" output="0" type="long" />
        </Activity>
        <Activity activityId="7" attempted="true" completed="true" title="MC3">
            <ToolOutput description="Are learner's answers all correct?" name="learner.all.correct" output="
false" type="boolean" />
            <ToolOutput description="Learner's total mark" name="learner.mark" output="0" type="long" />
        </Activity>
    </LearnerOutput>
    <LearnerOutput firstName="Three" lamsUserId="7" lamsUserName="test3" lastName="Test" userName="test3">
        <Activity activityId="6" attempted="false" completed="false" title="MC1" />
        <Activity activityId="5" attempted="false" completed="false" title="MC2" />
        <Activity activityId="7" attempted="false" completed="false" title="MC3" />
    </LearnerOutput>
    <LearnerOutput firstName="Testing" lamsUserId="2" lamsUserName="test" lastName="LDAP" userName="test">
        <Activity activityId="6" attempted="false" completed="false" title="MC1" />
        <Activity activityId="5" attempted="false" completed="false" title="MC2" />
        <Activity activityId="7" attempted="false" completed="false" title="MC3" />
    </LearnerOutput>
</ToolOutputs>
```

Calling:
http://<your-lams>:8080/lams//services/xml/LessonManager?
&serverId=xxx&courseId=xxx&username=test1&datetime=xxx&hashValue=xxx&lang=en&country=AU&method=authoredToolOutputsAllUsers&lsId=1
You get:
authoredToolOutputsAllUsers.xml

You can use either http method POST or GET.

Calling:
http://<your-lams>:8080/lams//services/xml/LessonManager?
&serverId=xxx&courseId=xxx&username=test1&datetime=xxx&hashValue=xxx&lang=en&country=AU&method=toolOutputsUser&lsId=1&outputsUser=test
XYZ

You get:
toolOutputsUser.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ToolOutputs lessonId="1" name="mc">
    <LearnerOutput completedLesson="true" firstName="One" lamsUserId="5" lamsUserName="test1" lastName="Test"
userName="test1">
        <Activity activityId="6" attempted="true" completed="true" title="MC1">
            <ToolOutput description="Are learner's answers all correct?" name="learner.all.correct" output="
true" type="boolean" />
            <ToolOutput description="Learner's total mark" name="learner.mark" output="1" type="long" />
        </Activity>
        <Activity activityId="5" attempted="true" completed="true" title="MC2">
            <ToolOutput description="Are learner's answers all correct?" name="learner.all.correct" output="
false" type="boolean" />
            <ToolOutput description="Learner's total mark" name="learner.mark" output="0" type="long" />
        </Activity>
        <Activity activityId="7" attempted="true" completed="true" title="MC3">
            <ToolOutput description="Are learner's answers all correct?" name="learner.all.correct" output="
false" type="boolean" />
            <ToolOutput description="Learner's total mark" name="learner.mark" output="0" type="long" />
        </Activity>
    </LearnerOutput>
</ToolOutputs>
```

Calling:
http://<your-lams>:8080/lams//services/xml/LessonManager?
&serverId=xxx&courseId=xxx&username=test1&datetime=xxx&hashValue=xxx&lang=en&country=AU&method=authoredToolOutputsUser&lsId=1&outputs
User=testXYZ

You get:
authoredToolOutputsUser.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<ToolOutputs lessonId="1" name="mc">
    <LearnerOutput completedLesson="true" firstName="One" lamsUserId="5" lamsUserName="test1" lastName="Test"
userName="test1">
        <Activity activityId="6" attempted="true" completed="true" title="MC1">
            <ToolOutput description="Are learner's answers all correct?" name="learner.all.correct" output="
true" type="boolean" />
        </Activity>
        <Activity activityId="5" attempted="true" completed="true" title="MC2">
            <ToolOutput description="Learner's total mark" name="learner.mark" output="0" type="long" />
        </Activity>
        <Activity activityId="7" attempted="true" completed="true" title="MC3">
            <ToolOutput description="Learner's total mark" name="learner.mark" output="0" type="long" />
        </Activity>
    </LearnerOutput>
</ToolOutputs>
```

Get Gradebook for Lesson or Course

To get Gradebook for a particular lesson or a whole course you need to send a request Gradebook servlet:

http://localhost/lams/services/Gradebook?

The parameters are:

- uid - the username of the current user
- method - either author, monitor or learner; see description below
- ts - timestamp
- sid - the server id
- hash - SHA1 hash of [ts + uid + method + serverID + serverKey] (Note: all lower case)
- country - Country (must use 2 letter code as ISO 3166-1 alpha-2 code for country)
- lang - Language code or locale code. In case of just passing language use "es" for Spanish or locale as "es_ES" (see list of available locales in LAMS)
- courseid - the id of the course from the LMS
- lsid - the id of the lesson
- courseName - the LMS's course name

You can use either http method POST or GET.

Gradebook content can be obtained two ways:

- if the user is already logged in, a direct call to Gradebook servlet will return the desired content; "method" parameter is still required for hash, but has no meaning in processing
- Gradebook content can be also delivered by calling LoginRequest servlet, which will log the user in first; in this case the request to LoginRequest servlet needs to have "method" and "courseid" parameters set (same as in a plain call to this servlet), plus "mode" parameter set to "gradebook"

If ID of an external course is provided (it must be in case of calling LoginRequest, but it is not required when calling Gradebook servlet directly), it is translated into internal LAMS organisation ID and user is redirected to Gradebook for this course. If lsId is provided, it has higher priority than courseid and user is redirected to Gradebook for this particular lesson.

Get total user marks in a lesson

It's possible to get either a lesson total mark for one specified user, or lesson total marks for all users in a lesson, or lesson total marks for all users in all lessons in a course.

To get total user marks from LAMS you need to make a call to the following servlet: http://localhost/lams/services/xml/LessonManager?
The parameters are:

- username - the username of the current user
- serverId - the server id
- datetime - timestamp
- hashValue - SHA1 hash of [timestamp + username + serverID + serverKey] (Note: all lower case)
- method - gradebookMarksUser for lesson total mark for a specified user or gradebookMarksLesson for all users in a lesson or gradebookMarksCourse - for all users for all lessons in a course
- lsId - the lesson id *(only for method=gradebookMarksUser and method=gradebookMarksLesson)*
- courseId - the course id *(only for method=gradebookMarksCourse)*
- outputsUser - the username of the user who's progress you wish to retrieve *(only for method=gradebookMarksUser)*

You will receive an xml response in the following format:

For single user total mark:

```
<GradebookMarks>
    <Lesson
        lessonId="13"
        lessonName="Big lesson"
        lessonMaxPossibleMark="4">
        <Learner
            extUsername="alex"
            userTotalMark="3">
        </Learner>
    </Lesson>
</GradebookMarks>
```

In case of method=gradebookMarksLesson there will be multiple <Learner> elements. And in case of method=gradebookMarksCourse there will be multiple <Lesson> elements.

Easy integration example

Check out Integration example to get a quick overview on how you can integration your LMS to LAMS.