# **ASU Ring Leader LTI 1.3 Library**

The ‘rl-server-lib’ and ‘rl-client-lib’ packages are published as private NPM packages that can be included within tools. These libraries will be used to expedite the integration with LTI 1.3 capable LMSs.

The ‘rl-tool-example-client’ and ‘rl-tool-example-server’ packages are used to stand up an example usage of the ‘rl-server-lib’ and ‘rl-client-lib’ libraries.

## Table of content

1. [NameAndRoles Services](#_NamesAndRoles_Service)
   1. [Retrieving members from a platform](#_Retrieving_members_from)
2. [AssignmentAndGrading Service](#_AssignmentAndGrading_Service)
   1. [Sample grade JSON](#_Sample_grade_JSON)
   2. [Sending grades to a platform](#_Sending_grades_to_1)
   3. [Retrieving grades from a platform](#_Retrieving_grades_from)
   4. [Retrieving assignments from a platform](#_Retrieving_assignments_from)
   5. [Create assignments in platform](#_Create_assignments_in)
   6. [Delete assignments from platform](#_Delete_assignments_from)
3. [DeepLinking Service](#_Deep_Linking_Service)
   1. [Deep linking message](#_Deep_linking_message)
   2. [Items parameter](#_Items_parameter)
   3. [Options](#_options)
4. [Sample JSON for each LTI services](#_Sample_JSON_for)
   1. [Names and Roles Provisioning Service](#_Names_and_Roles)
   2. [External Tool Click -Instructor View](#_External_Tool_Click)
   3. [External Tool Click -Student View](#_External_Tool_Click_1)
   4. [Assignment Service](#_Assignment_Service.json)
   5. [CreateAssignmentInsideTool](#_CreateAssignmentInsideTool.json)
   6. [SubmitStudentGrade](#_SubmitStudentGrade.json)
   7. [GetGradeFromPlatform Service](#_GetGradeFromPlatform_Service.json)
   8. [DeepLinking - Response](#_DeepLinking_–_Response)
5. [Launch and LTI Advantage Service End Points](#_Launch_and_LTI)
   1. [rlLtiLaunchExpressEndpoints](#_rlLtiLaunchExpressEndpoints)
      1. [Initiating OIDC flow](#_Initiating_OIDC_flow)
      2. [External Tool launched directly from Platform UI](#_External_Tool_launched)
      3. [External Tool launched via assignments in Platform UI](#_External_Tool_launched_1)
      4. [DeepLinking request from Platform](#_DeepLinking_request_from)
   2. [rlLtiServiceExpressEndpoints](#_rlLtiServiceExpressEndpoints)

## **NamesAndRoles Service**

This library implements the [**LTI® 1.3 Names and Roles Provisioning Service**](https://www.imsglobal.org/spec/lti-nrps/v2p0) in the form of the **NamesAndRoles Class**.

The Names And Roles Provisioning Service can be used by a tool to access a list of platform's users (referred to as members) and their roles within the context of a course, program or other grouping.

This class provides a single method for retrieving membership information, and offers the specified filters and functionalities.

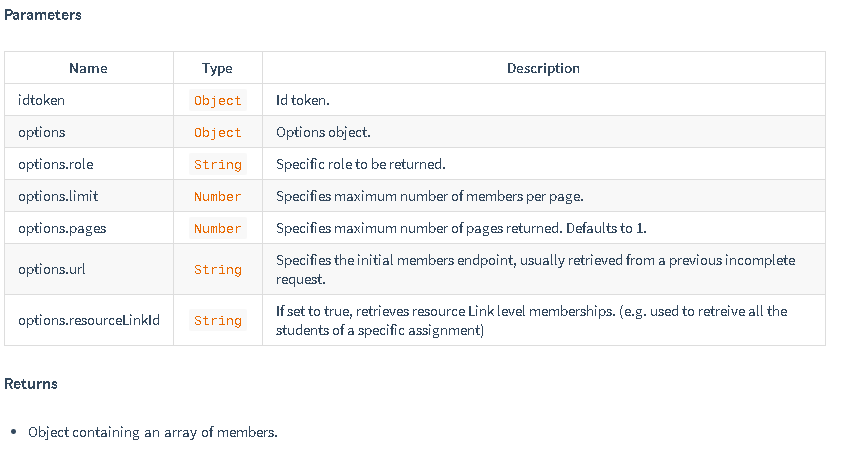
## [Retrieving members from a platform](https://cvmcosta.me/ltijs/" \l "/namesandroles?id=retrieving-members-from-a-platform)

All members of a platform within the context can be retrieved simply by calling the getMembers() method:

Since the information necessary to make request is present in the idtoken i.e. *platform*, it is necessary to pass it along as the first parameter**.**

const results = await new NamesAndRoles().getMembers(platform, {role: ‘Learner’ });// Gets context members

standard response:



## **AssignmentAndGrading Service**

This library implements the [**LTI® 1.3 Assignment and Grading Service Specification**](https://www.imsglobal.org/spec/lti-ags/v2p0/) in the form of the **Grade Class**.

### [Sample grade JSON](https://cvmcosta.me/ltijs/" \l "/grading?id=sending-grades-to-a-platform)

### The library is able to send grades to a platform in the [**application/vnd.ims.lis.v1.score+json**](https://www.imsglobal.org/spec/lti-ags/v2p0/#score-publish-service) LTI® standard:

{

"scoreGiven" : 83,

"comment" : "This is exceptional work.",

"activityProgress" : "Completed",

"gradingProgress": "FullyGraded"

}

This excludes the fields timestamp*,*userId*and*scoreMaximum of the specification, because the library function fills them automatically using the platform passed

### Sending grades to platform

app.post('/putgrade', async (req, res) => {

let grade = {

scoreGiven: 50,

activityProgress: 'Completed',

gradingProgress: 'FullyGraded'

}

// Sends a grade to a platform's grade line

Const results = await new Grade().putGrade (platform, grade)

return res.send(results)

})

### [Retrieving grades from platform](https://cvmcosta.me/ltijs/" \l "/deeplinking?id=deep-linking-message)

app.get('/grade', async (req, res) => {

// Retrieves grades from a platform

Const results = await new Grade().getGrades(platform)

return res.send(results)

})

### [Retrieving assignments from platform](https://cvmcosta.me/ltijs/" \l "/deeplinking?id=deep-linking-message)

The library is able to retrieve assignment from a platform:

app.get('/assignment, async (req, res) => {

// Retrieves lineitems from a platform

const result = await new Grade().getLineItems(platform)

return res.send(result)

})

**Note -** Library does provide the ability to create / delete assignments i.e. line items but currently we are not using this in current integration.

### [Create assignments in platform](https://cvmcosta.me/ltijs/" \l "/deeplinking?id=deep-linking-message)

The library is able to create assignments in platform.

app.post('/createassignment', async (req, res) => {

const lineItem = {

scoreMaximum: 100,

label: 'Grade',

tag: 'grade'

}

// Sends lineitem to a platform

const result = await new Grade().createLineItem(platform, lineItem)

return res.send(result )

})

### [Delete assignments from platform](https://cvmcosta.me/ltijs/" \l "/deeplinking?id=deep-linking-message)

The library is able to delete assignments from platform.

**Note-** Library can delete only those assignments which are created from tool:

// Deleting lineitem

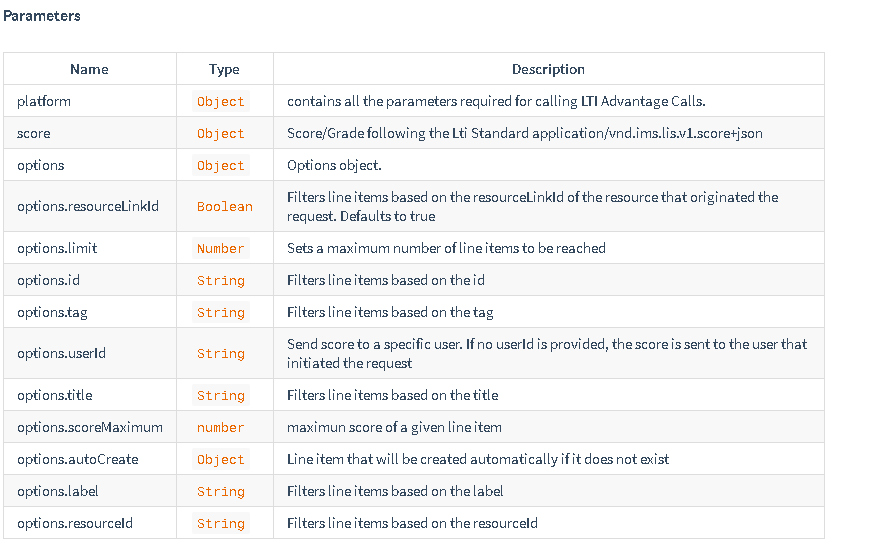
app.delete('/lineitem', async (req, res) => {

// Deleting lineitem on a platform

const result = await new Grade().deleteLineItems(platform,{id : litemItemId} )

return res.send(result)

})



# **Deep Linking Service**

Library implements the [**LTI® 1.3 Deep Linking Service Specification**](https://www.imsglobal.org/spec/lti-dl/v2p0/) in the form of the **DeepLinking Class**.

Deep Linking consists of using LTI® launch to select specific resources within a tool to be displayed to the user.

This class provides two methods for generating the deep linking messages, and a callback method called whenever there is a successful deep linking launch.

Following the official specification, the usual working flow is:

* Platform initiates deep linking launch.
* redirects to the resource selection view.
* In the resource selection view, the user selects resource and makes a POST request with the result.
* The tool then uses the DeepLinking class to generate the self-submitting form with the signed JWT message of the deep linking request and returns the form to the client.
* The client then embeds the form in the page, submitting the deep linking request.

**Deep linking launches uses the same endpoint as regular launches.**

### [Deep linking message](https://cvmcosta.me/ltijs/" \l "/deeplinking?id=deep-linking-message)

After resources are selected, the tool can now create a deep linking request message and send to the platform, this can be done through two methods:

* createDeepLinkingForm(idtoken, items, options)

Returns a self-submitting Deep Linking form containing the signed JWT message, this form should be embedded in the html of the tool's client in order to finish the deep linking request.

// Handles deep linking request generation with the selected resource   
app.post('/deeplink', async (req, res) => {

const resource = req.body

const items = [

{

type: 'ltiResourceLink',

title:'Assignment 1',

url:'https://my-tool/launch',

resourceId: assignment.resource\_id,

lineItem: {

scoreMaximum: assignment.lineitem\_score\_maximum,

label: assignment.lineitem\_label,

resourceId: assignment.lineitem\_resource\_id,

tag: assignment.lineitem\_tag

}

}

]

// Creates the deep linking request form

const form = await new DeepLinking().createDeepLinkingForm(platform, items,

{ message: 'Successfully registered resource!' })

return res.send(form)

})

* createDeepLinkingMessage(idtoken, items, options)

Returns a signed JWT message, that the client has to submit in a POST form in order to finish the deep linking request.

// Handles deep linking request generation with the selected resource

lti.app.post('/deeplink', async (req, res) => {

const resource = req.body

const items = [

{

type: 'ltiResourceLink',

title:'Assignment 1',

url:'https://my-tool/launch',

resourceId: assignment.resource\_id,

lineItem: {

scoreMaximum: assignment.lineitem\_score\_maximum,

label: assignment.lineitem\_label,

resourceId: assignment.lineitem\_resource\_id,

tag: assignment.lineitem\_tag

}

}

]

// Creates the deep linking request JWT message

const message = await new DeepLinking(). (platform, items,

{ message: 'Successfully registered resource!' })

return res.send(message)

})

### [Items parameter](https://cvmcosta.me/ltijs/" \l "/deeplinking?id=contentitems-parameter)

The Items parameter is either an content item object or an array of content item objects, following the [**LTI® 1.3 content item specification**](https://www.imsglobal.org/spec/lti-dl/v2p0/#content-item-types).

Passing the parameter **does not guarantee that all content items will be sent in the request**, to avoid errors, Library only sends content items that fit within the platform's accepted item types and allowed quantity.

If a platform only allows one content item per deep linking request, only the first content item passed in the parameter will actually be sent to the platform.

### [options](https://cvmcosta.me/ltijs/" \l "/deeplinking?id=options)

The [**Deep Linking specification**](https://www.imsglobal.org/spec/lti-dl/v2p0) allows us to set custom messages that should be displayed to the user in case of success or failure:

* options.message

ex: 'Successfully registered the resources!'

* options.errMessage

ex: 'Resource registration failed!'

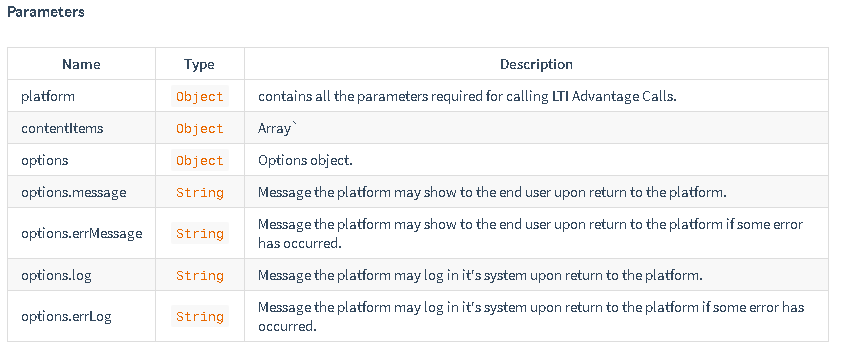
We can also set the messages that the platform will log in their systems in case of success or failure.

* options.log

ex: 'registered\_lti\_resource'

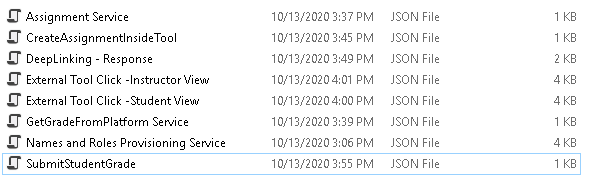
* options.errLog

ex: 'resource\_registration\_failed'



# Sample JSON for each LTI Advantage services

The document folder contains multiple JSON file that has sample JSON that is received via platform or sent to platform via tool.



#### Names and Roles provisioning Service.json

* 1. This file contains the json that is returned from platform when tool makes a request to fetch the roster.
  2. Roster can be fetched at course level as well as assignment level.

#### External Tool Click -Instructor View.json

* 1. This file contains the json that is returned by platform when an instructor launce the external tool inside platform.
  2. Library has a property named “isInstructor” that gets set when an instructor launce the tool.

#### External Tool Click -Student View.json

1. This file contains the json that is returned by platform when student launce the external tool inside platform.
2. Library has a property named “isStudent” that gets set when student launce the tool.

#### CreateAssignmentInsideTool.json

1. This file contains the json that represent how the assignment is created in the tool that will be later link to assignment via deep linking – resource link selection

#### Assignment Service.json

* 1. This file contains the json that represents the assignment that is sent by the platform to the tool when tool makes a request for list of assignments

#### SubmitStudentGrade.json

1. This is sample json for sending grade of a student to platform using AssignmentAndGrades LTI advantage service

#### GetGradeFromPlatform Service.json

* 1. This file contains the json that represents the grades for each assignment that is sent by the platform to the tool when tool makes a request for grades of each assignments

#### DeepLinking – Response.json

1. This file contains the json that is sent by the tool when platform initiates a deep link request to the tool
2. This happens when instructor is creating assignment in platform and then starts process of linking that assignment to the external tool link.
3. All the resource link (i.e. assignments created inside tool) are displayed inside the platform and then instructor selects the resource link that he wants to associate with the platform assignment

## Launch and LTI Advantage Service End Points

There are two separate files for handling the tool launch and calling the LTI Advantage Services.

### rlLtiLaunchExpressEndpoints

These endpoints are responsible for handling the launch process.

#### Initiating OIDC flow

When user clicks on the external tool from courses navigation in platform, the request is received by the OIDC\_LOGIN\_INIT\_ROUTE endpoints. As per the specification, we need to support both GET and POST request.

 // OIDC GET initiation

*app*.get(OIDC\_LOGIN\_INIT\_ROUTE, requestLogger, async (*req*, *res*) *=>* {

    initOidcGet(*req*, *res*);

  });

*app*.post(OIDC\_LOGIN\_INIT\_ROUTE, requestLogger, async (*req*, *res*) *=>* {

    initOidcPost(*req*, *res*);

  });

#### External Tool launched directly from Platform UI

After we initiate the OIDC flow, platform sends the id\_token which is handled by the LTI\_ADVANTAGE\_LAUNCH\_ROUTE endpoint. This function is redirects user (depending upon Student or Instrutor) to their defined respective endpoints.

**Note:** - this function does not get called when the OIDC flow is initiated on clicking assignment. This gets called only when we click external tool.

  // post to accept the LMS launch with idToken

*app*.post(LTI\_ADVANTAGE\_LAUNCH\_ROUTE, requestLogger, async (*req*, *res*) *=>* {

    ltiLaunchPost(*req*, *res*);

  });

#### External Tool launched via assignments in Platform UI

The OIDC flow is same for both the launches i.e. via direct external tool launch or via assignment however when it is launch from the assignment, the end point that handles the redirect is LTI\_ASSIGNMENT\_REDIRECT.

*app*.post(LTI\_ASSIGNMENT\_REDIRECT, requestLogger, async (*req*, *res*) *=>* {

    assignmentRedirectPost(*req*, *res*);

  });

#### DeepLinking request from Platform

Platform send a deeplinking request to the tool when user starts process of associated the resource link to an assignment. This request is handled by LTI\_DEEPLINK\_REDIRECT end point

*app*.post(LTI\_DEEPLINK\_REDIRECT, requestLogger, async (*req*, *res*) *=>* {

    console.log(`LTI\_DEEPLINK\_REDIRECT:${LTI\_DEEPLINK\_REDIRECT}`);

    deepLinkRedirect(*req*, *res*);

  });

### rlLtiServiceExpressEndpoints

These endpoints are responsible for calling LTI Advantage Services. Different end points available are:

1. CREATE\_ASSIGNMENT\_ENDPOINT – Creates an assignment that is saved in tool. These assignments are nothing but a resource link that gets associated with the platform’s assignment
2. GET\_ASSIGNMENT\_ENDPOINT – fetching the assignment link from the platform
3. GET\_UNASSIGNED\_STUDENTS\_ENDPOINT – getting a list of students not assigned to a specific assignment
4. PUT\_STUDENT\_GRADE\_VIEW – auto-submitting grades for logged-in user i.e. student.
5. DEEP\_LINK\_RESOURCELINKS\_ENDPOINT – it sends list of all the assignments created inside the tool as resource selection item list.
6. DEEP\_LINK\_ASSIGNMENT\_ENDPOINT – handles the deep linking request when platform initiates the deep linking flow.
7. PUT\_STUDENT\_GRADE – submitting grades of students as instructor
8. DELETE\_LINE\_ITEM – deleting a line item created by tool. This does not delete the assignments created from platform UI
9. GET\_GRADES