# **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.

## **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.

## [Usage](https://cvmcosta.me/ltijs/#/namesandroles?id=usage)

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

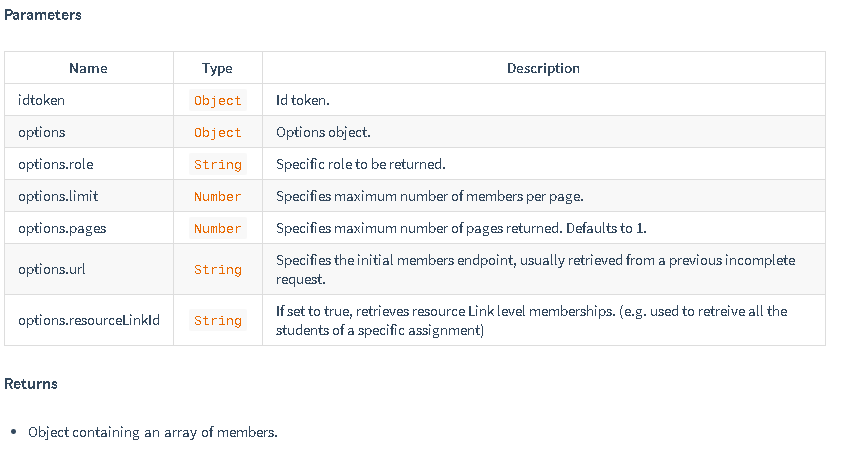
All members of a platform within the context can be retrieved simply by calling the getMembers() method:

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**.

##### [Sending grades to a platform](https://cvmcosta.me/ltijs/#/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 the grade:

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 a platform](https://cvmcosta.me/ltijs/#/grading?id=retrieving-grades-from-a-platform)

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 a platform](https://cvmcosta.me/ltijs/#/grading?id=retrieving-grades-from-a-platform)

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/#/grading?id=retrieving-grades-from-a-platform)

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/#/grading?id=retrieving-grades-from-a-platform)

The library is able to delete assignments from platform.

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

// Deleting lineitem

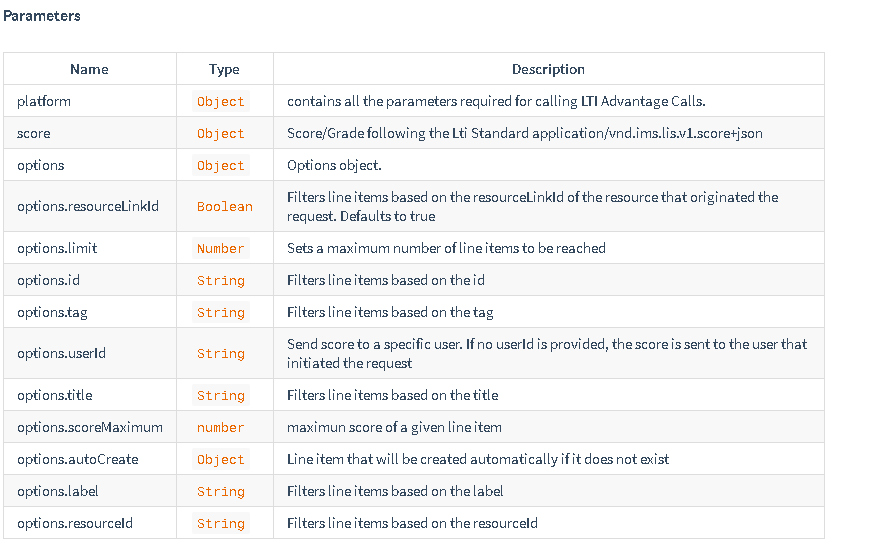
app.delete('/lineitem', async (req, res) => {

// 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/#/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/#/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/#/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'

