CGS backend

APIs

|  |  |
| --- | --- |
| Document Owner: | Eran Shoucroun |
| Document File: | CGS\_backend\_APIs.docx |
| Document Version: | 1.4 |
| Document Location: | [link](http://turtle.timetoknow.local:8080/svn/dev/t2kdev/trunk/cgs/doc/design/architecture) |

# Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Author | Description | Date |
| 0 | Yoni Zohar | Assests APIs | 30.10.12 |
| 1 | Ophir Barnea | Course / Lessons APIs | 30.10.12 |
| 2 | Micha Shlain | Account API | 30.12 |
| 3 | Yoni Zohar | Update in Assets APIs | 30.10.12 |
| 4 | Yoni Zohar | Update in Assets APIs | 06.11.12 |
| 5 | Ron.Bresler | Lock ID, PUT vs POST, return values from save | 6.11.12 |
| 6 | Yoni Zohar | Update in Assets APIs | 22.11.12 |
| 7 | Yoni Zohar | Update in Assets APIs | 29.11.12 |
| 8 | Yoni Zohar | Update in Assets APIs | 06.12.12 |
| 9 | Ophir Barnea | Update Lessons/Sequences/Packages | 09.12.12 |
| 10 | Ophir Barnea | Move Sequences to course API(url), + validation util | 13.12.12 |
| 11 | Ophir barnea | Locks dates + publish params | 29.01.13 |
| 12 | Anya Grinberg | Applets | 31.01.13 |
| 13 | Ophir barena | Packages API changes | 31.01.13 |
| 14 | Anya Grinberg | Applets API changes | 13.02.13 |
| 15 | Micha Shlain | Standards API | 22.04.13 |
| 16 | Micha Shlain | Updated Standards API standard package version type changed to string | 25.04.13 |
| 17 | Micha Shlain | Added clone course API | 02.06.13 |
| 18 | Yoni Zohar | Added copy assets API for import lessons | 04.06.13 |
| 19 | Yoni Zohar | Admin APIs | 25.06.13 |
| 20 | Yoni Zohar | Update in Admin APIs | 30.06.13 |
| 21 | Anya Grinberg | Sequence Tree of Hidden Lessons | 14.11.13 |
| 22 | Anya Grinberg | Get applets for specific publisher | 25.11.13 |
| 23 | Anya Grinberg | Get text to speech providers | 11.12.13 |
| 24 | Anya Grinberg | Proxy Rest | 16.12.13 |
| 25 | Alex Zaikman | Remove Locale | 21.01.14 |
| 26 | Alex Zaikman | Send Email to CGS users | 21.01.14 |
| 27 | Alex Zaikman | NarrationService.ivona | 03.02.14 |

# Table of Contents

[Revisions 1](#_Toc379210539)

[Table of Contents 3](#_Toc379210540)

[1 Intro. 6](#_Toc379210541)

[2 APIs 6](#_Toc379210542)

[2.1 Publishers 6](#_Toc379210543)

[2.1.1 Get all publisher accounts 6](#_Toc379210544)

[2.1.2 Get publisher account 6](#_Toc379210545)

[2.1.3 Create publisher account 7](#_Toc379210546)

[2.1.4 Update publisher account 7](#_Toc379210547)

[2.1.5 Delete publisher account 7](#_Toc379210548)

[2.2 Users 8](#_Toc379210549)

[2.2.1 Get all users 8](#_Toc379210550)

[2.2.2 Get user 8](#_Toc379210551)

[2.2.3 Create user 8](#_Toc379210552)

[2.2.4 Update user 9](#_Toc379210553)

[2.2.5 Delete user 9](#_Toc379210554)

[2.2.6 Get all users of publisher 9](#_Toc379210555)

[2.2.7 Get user of publisher 10](#_Toc379210556)

[2.2.8 Create user in publisher 10](#_Toc379210557)

[2.2.9 Update user in publisher 10](#_Toc379210558)

[2.2.10 Delete user in publisher 10](#_Toc379210559)

[2.3 Assests management 11](#_Toc379210560)

[2.4 Course management 11](#_Toc379210561)

[2.4.1 Get publisher Courses properties list 11](#_Toc379210562)

[2.4.2 Get Course (TOC) 11](#_Toc379210563)

[2.4.3 Save Course (TOC) 12](#_Toc379210564)

[2.4.4 Clone Course 12](#_Toc379210565)

[2.4.5 Lock\Unlock Course (TOC) 12](#_Toc379210566)

[2.4.6 GET Lock Course (TOC) 13](#_Toc379210567)

[2.4.7 Create new edition for course 13](#_Toc379210568)

[2.4.8 get lesson sequences 14](#_Toc379210569)

[2.4.9 get lesson sequences thumbnails 15](#_Toc379210570)

[2.4.10 get sequences tree of hidden lessons 15](#_Toc379210571)

[2.4.11 remove Locale 15](#_Toc379210572)

[2.4.12 Send Email to CGS users 16](#_Toc379210573)

[2.4.13 NarrationService.ivona 16](#_Toc379210574)

[2.5 Lesson management 17](#_Toc379210575)

[2.5.1 Get lessons properties 17](#_Toc379210576)

[2.5.2 Get lesson 17](#_Toc379210577)

[2.5.3 Save lesson 18](#_Toc379210578)

[2.5.4 Save lesson Contents 18](#_Toc379210579)

[2.5.5 Lock\Unlock Lesson 19](#_Toc379210580)

[2.5.6 GET Lock Lesson 19](#_Toc379210581)

[2.5.7 Save lesson propertis 20](#_Toc379210582)

[2.5.8 Import Lesson – copy specific assets 21](#_Toc379210583)

[2.6 Package management 22](#_Toc379210584)

[2.6.1 Prepare for package 22](#_Toc379210585)

[2.6.2 Publish course 23](#_Toc379210586)

[2.6.3 Get CGSPackage 24](#_Toc379210587)

[2.6.4 Cancel Packaging Process 25](#_Toc379210588)

[*2.6.5* *Get CGSPackages By Publisher* 25](#_Toc379210589)

[*2.6.6* *Get CGSPackages By course* 26](#_Toc379210590)

[*2.6.7* *Get Active CGSPackages By userName* 26](#_Toc379210591)

[2.7 Utils services 27](#_Toc379210592)

[2.7.1 Validate course 27](#_Toc379210593)

[2.7.2 Validate lesson 27](#_Toc379210594)

[2.8 Applets management 27](#_Toc379210595)

[2.8.1 Get all allowed applets for specific publisher 27](#_Toc379210596)

[2.8.2 Get applet manifest of specific course 27](#_Toc379210597)

[2.8.3 Add applet to the course 28](#_Toc379210598)

[2.8.4 Update course applet to latest version 28](#_Toc379210599)

[2.9 Standards 28](#_Toc379210600)

[2.9.1 Get headers of all available standard packages 28](#_Toc379210601)

[2.9.2 Get a standard package 29](#_Toc379210602)

[2.10 Text To Speech Providers 29](#_Toc379210603)

[2.10.1 Get all text to speech providers 29](#_Toc379210604)

[2.11 Proxy service 29](#_Toc379210605)

[2.11.1 Propogate request to external domain 29](#_Toc379210606)

[2.12 Error handling 30](#_Toc379210607)

# Intro.

This document describes the entire APIs exposed from CGS server.  
All APIs are implemented using REST protocol.  
  
The APIs are grouped into same functional section (assests / Courses / lessons etc.)

The prefix for all API is : **http://{server-name}/cgs/rest/**

# APIs

## Publishers

### Get all publisher accounts

|  |  |
| --- | --- |
| **Description** | Returns a list of the publisher objects |
| **HTTP Method** | GET |
| **URL** | /publishers |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | /publishers |
| **Response JSON example** | http://turtle.timetoknow.local:8080/svn/dev/t2kdev/trunk/cgs/doc/design/architecture/data/publishers.json |

### Get publisher account

|  |  |
| --- | --- |
| **Description** | Returns the publisher object by user (currently , one default account(1) for all) |
| **HTTP Method** | GET |
| **URL** | /publishers/{publisherId} |
| **JAVA method** | getAccountData(userId) |
| **Errors** | "404 Not Found" - in case publisher account is not found "5xx Server " |
| **Example** | /publishers/1 |
| **Response JSON example** | http://turtle.timetoknow.local:8080/svn/dev/t2kdev/trunk/cgs/doc/design/architecture/data/publisher.json |

### Create publisher account

|  |  |
| --- | --- |
| **Description** | Creates the publisher object |
| **HTTP Method** | POST |
| **URL** | /publishers |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | /publishers |
| **Request JSON Example** | http://turtle.timetoknow.local:8080/svn/dev/t2kdev/trunk/cgs/doc/design/architecture/data/add\_publisher\_request.json |

### Update publisher account

|  |  |
| --- | --- |
| **Description** | Updates (overrides) an existing publisher object |
| **HTTP Method** | PUT |
| **URL** | /publishers |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | /publishers |
| **Request JSON Example** | http://turtle.timetoknow.local:8080/svn/dev/t2kdev/trunk/cgs/doc/design/architecture/data/update\_publisher\_request.json |

### Delete publisher account

|  |  |
| --- | --- |
| **Description** | Deletes the publisher object |
| **HTTP Method** | DELETE |
| **URL** | /publishers/{publisherId} |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | /publishers/1 |

## Users

### Get all users

|  |  |
| --- | --- |
| **Description** | Returns a list of the user objects |
| **HTTP Method** | GET |
| **URL** | /users?role={role} (the parameter "role" is not mandatory).  /users |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | /users  /users?role=ROLE\_ADMIN  /users?role=ROLE\_PUBLISHER\_ADMIN  /users?role=ROLE\_CONTENT\_DEVELOPER |
| **Result Example** | http://turtle.timetoknow.local:8080/svn/dev/t2kdev/trunk/cgs/doc/design/architecture/data/users.json |

### Get user

|  |  |
| --- | --- |
| **Description** | Returns a user object |
| **HTTP Method** | GET |
| **URL** | /users/{userId} |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | /users/3 |
| **Result example** | http://turtle.timetoknow.local:8080/svn/dev/t2kdev/trunk/cgs/doc/design/architecture/data/get\_user.json |

### Create user

|  |  |
| --- | --- |
| **Description** | Creates a user object whose authority is publisher admin |
| **HTTP Method** | POST |
| **URL** | /users |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | /users |
| **Request Example** | http://turtle.timetoknow.local:8080/svn/dev/t2kdev/trunk/cgs/doc/design/architecture/data/save\_pub\_admin\_request.json |

### Update user

|  |  |
| --- | --- |
| **Description** | Updates (overrides) a user object |
| **HTTP Method** | PUT |
| **URL** | /users |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | /users |
| **Request Examples** | http://turtle.timetoknow.local:8080/svn/dev/t2kdev/trunk/cgs/doc/design/architecture/data/update\_user\_request.json |

### Delete user

|  |  |
| --- | --- |
| **Description** | Deletes a user object |
| **HTTP Method** | DELETE |
| **URL** | /users/{userName} OR /users/{userId}?id=true |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | /users/Nir.Krevner OR /users/3?id=true |

### Get all users of publisher

|  |  |
| --- | --- |
| **Description** | Returns a list of the user objects that are members of the specified publishers |
| **HTTP Method** | GET |
| **URL** | /publishers/{publisherId}/users?role={role} (the parameter "role" is not mandatory).  /publishers/{publisherId}/users |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | Publishers/1/users  /users  /users?role=ROLE\_ADMIN  /users?role=ROLE\_PUBLISHER\_ADMIN  /users?role=ROLE\_CONTENT\_DEVELOPER |

### Get user of publisher

|  |  |
| --- | --- |
| **Description** | Returns a user object of the specified publisher |
| **HTTP Method** | GET |
| **URL** | /publishers/{publisherId}/users/{userId} |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | Publishers/1/users/3 |

### Create user in publisher

|  |  |
| --- | --- |
| **Description** | Creates a user object in the specified publisher. Its authority is content developer |
| **HTTP Method** | POST |
| **URL** | publishers/{publisherId}/users |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | Publishers/1/users |

### Update user in publisher

|  |  |
| --- | --- |
| **Description** | Updates (overrides) a user object in the specified publisher |
| **HTTP Method** | PUT |
| **URL** | Publishers/{publisherId}/users |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | /publishers/1/users |

### Delete user in publisher

|  |  |
| --- | --- |
| **Description** | Deletes a user object in the specified publisher |
| **HTTP Method** | DELETE |
| **URL** | /publishers/{publisherId}/users/{userName} OR /publishers/{publisherId}/users/{userId}?id=true |
| **JAVA method** |  |
| **Errors** |  |
| **Example** | /publishers/1/users/Nir.Krevner OR /publishers/1/users/3?id=true |

## Assests management

Get, Save and Delete are performed by sending a request to the same address. The first is by using a GET HTTP request, second is by using a POST HTTP request and the third is by using a DELETE HTTP request.

|  |  |
| --- | --- |
| **Description** | Get/Save a publisher account assest – assests that are common to a specific publisher |
| **HTTP Method** | GET / POST / DELETE |
| **URL** | /publishers/{publisherGUID}/courses/courseGUID/{destination}/\*\*  **Where:**   1. {destination} value is one of: skills, css, scheme, standards, media, src, cgsdata. 2. \*\* means any path (including subdirectories). 3. In POST, the expected request should contain the uploaded file as a form multipart field of a form. |
| **JAVA method** | getAsset / uploadAsset / deleteAsset |
| **Errors** | "404 Not Found" – In case the file doesn't exist (in get and delete).  "413 Request Entity Too Large" – In case the uploaded asset exceeds the upload size limit of its extension (or the default upload size limit)  "422 Unprocessable Entity" – In case the uploaded file is empty or wan't specified in a multipart field.  "500 Internal Server Error" |
| **Example** | /publishers/abcde765/courses/abcdef76543/media/bc/bcdef3f3fdsa2df3.jpg |

## Course management

### Get publisher Courses properties list

|  |  |
| --- | --- |
| **Description** | Returns a list of Courses that belongs to the publisher.  The Courses are headers only (with no TOC) |
| **HTTP Method** | GET |
| **URL** | /publishers/{publisherId}/courses |
| **JAVA method** | getPublisherCoursesProperites(publisherId) |
| **Errors** | "5xx Server " |
| **Example** | /publishers/1/courses |

### Get Course (TOC)

|  |  |
| --- | --- |
| **Description** | Returns a Course document (with toc) |
| **HTTP Method** | GET |
| **URL** | /publishers/{publisherId}/Courses/{CourseId}?&last-modified={lastModified} non-mandatory & version=yyyy-MM-dd'T'HH:mm:ss.SSSZ  *version* param : requested version. current “0.0.0“.  last-modified param : if empty returns the Course as requested , else returns the Course only in case of newer instance exsisting on server. |
| **JAVA method** | getCourse(id ,lastModified) |
| **Errors** | "404 Not Found" - in case Course is not found  "5xx Server " |
| **Example** | /publishers/{publisherId}/Courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/?version=0.0.0 & last-modified = 2012-11-13T14:04:27.92Z |

### Save Course (TOC)

|  |  |
| --- | --- |
| **Description** | Saves the Course content to Course repository. |
| **HTTP Method** | PUT |
| **URL** | /publishers/{publisherId}/courses/{CourseId} |
| **Response** | On success (code 200) response body will include the header section of the saved document:  { " last-modified ": "2012-11-13T14:04:27.92Z "} |
| **JAVA method** | saveCourse(Course) |
| **Errors** | "5xx Server " |
| **Example** | /publishers/{publisherId}/courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887 |

### Clone Course

|  |  |
| --- | --- |
| **Description** | Deep clone of a course content. |
| **HTTP Method** | POST |
| **URL** | /publishers/{publisherId}/courses/{CourseId}/clone |
| **Response** | On success (code 200) response body will include the new id of the cloned course  2a1dcb81-f918-be8e-3dbe-f2e7eed59887 |
| **JAVA method** | saveCourse(Course) |
| **Errors** | "5xx Server " |
| **Example** | /publishers/{publisherId}/courses/2a1dcb81-f918-be8e-3ffa-f2e7eed59887/clone |

### Lock\Unlock Course (TOC)

|  |  |
| --- | --- |
| **Description** | Tries to acquire Lock or to release a lock on a Course entity by the user (all versions) |
| **HTTP Method** | POST |
| **URL** | publishers/{publisherId}/courses/{CourseId}/lock?action= acquire/release?last-modified={ISO8601date}  params:  action: lock action – “release” will try tp release the lock on the course. “acquire” will try to acquire a lock on the course  last-modifoed: a date value , reflects the document last-modified value |
| **JAVA method** | lock (CourseId) |
| **Errors** | "5xx Server "  423 LOCKED – when the acquire or release request fails  "404 Not Found" - in case of Course not found |
| **Example** | /publishers/2/courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/lock? action= acquire |

### GET Lock Course (TOC)

|  |  |
| --- | --- |
| **Description** | Polls the locking status of the current object. Who locks the object |
| **HTTP Method** | GET |
| **URL** | publishers/{publisherId}/Courses/{CourseId}/lock |
| **JAVA method** | getLock(CourseId) |
| **Return** | {  "entityId": "69bb10b0-26a6-11e2-81c1-0800200c9a99",  "entityType": "COURSE",  "userName": "ophir.barnea",  "userEmail": "fName.lName@timetoknow.com",  "userPublisherId": 1,  “entityName”=”name”  "lockDate": 2012-11-13T14:04:27.92Z  } |
|  | "5xx Server " |
| **Example** | /publishers/{publisherId}/Courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/lock |

### Create new edition for course

|  |  |
| --- | --- |
| **Description** | Api for creating a course edition. The server will check forl any locks and if ok, will aquire a system lock over the course contents.  The course will be duplicated will a new edition from d.d.d to d+1.0.0   * Course will have new course Id   The CMS contenst are also duplicated |
| **HTTP Method** | POST |
| **URL** | publishers/{publisherId}/courses/{courseId}/createNewEdition |
| **JAVA method** | createNewCourseEdition (CourseId) |
| **Return** | New courseId as String |
|  | "5xx Server " |
| **Example** | /publishers/{publisherId}/courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887 |

### get lesson sequences

|  |  |
| --- | --- |
| **Description** | Returns all sequences of the lesson  \*\* note: the returned sequences might have some which are no longer attached to the lesson due to removal from resource table . the server does not check the resource table in the document , but only replies with the known (saved) sequences.  The client needs to use only those sequences which are binded to the lesson |
| **HTTP Method** | GET |
| **URL** | /publishers/{publisherId}/courses/{CourseId}/sequences?lessonId={lessonId}  lessonId: a mandatory request param , value of the lesson ID to get the sequences of. |
| **JAVA method** | getSequences (publisherId, courseId , lessonId) |
| **Errors** | "5xx Server " |
| **Response** | On success (code 200) response body will include the a JSON with an array of sequences  [{"seqId":"aaaaa","content":"aaaaaaaaaaa"},  {"seqId":"xxxxx",content":"somecontents...."}] |
| **Example** | /publishers/{publisherId}/courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/sequences?lessonId=dasd212edqsas |

### get lesson sequences thumbnails

|  |  |
| --- | --- |
| **Description** | Returns all sequences thumbnails of the lesson  \*\* note: the returned sequences thumbnails might have some which are no longer attached to the lesson due to removal from resource table . the server does not check the resource table in the document , but only replies with the known (saved) sequences.  The client needs to use only those sequences which are binded to the lesson |
| **HTTP Method** | GET |
| **URL** | /publishers/{publisherId}/courses/{CourseId}/thumbnails?lessonId={lessonId}  lessonId: a mandatory request param , value of the lesson ID to get the sequences of. |
| **JAVA method** | getThumbnails (publisherId, courseId , lessonId) |
| **Errors** | "5xx Server " |
| **Response** | On success (code 200) response body will include the a JSON with an array of sequences thumbnails  [{"seqId":"aaaaa","content":"aaaaaaaaaaa"},  {"seqId":"xxxxx",content":"somecontents...."}] |
| **Example** | /publishers/{publisherId}/courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/thumbnails?lessonId=dasd212edqsas |

### get sequences tree of hidden lessons

|  |  |
| --- | --- |
| **Description** | Returns the TOC tree from Course to Sequence all hidden lessons in the course |
| **HTTP Method** | GET |
| **URL** | /publishers/{publisherId}/courses/sequenceTreeOfHiddenLessons |
| **JAVA method** | ContentTreeController.getSequenceTreeOfHiddenLessons(publisherId, courseId , lessonId) |
| **Errors** | "5xx Server " |
| **Response** | See sequenceTree.json |
| **Example** | /publishers/{publisherId}/courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/ sequenceTreeOfHiddenLessons |

### remove Locale

|  |  |
| --- | --- |
| **Description** | Remove locale |
| **HTTP Method** | DELETE |
| **URL** | /publishers/{publisherId}/courses/{courseId}/locale/{locale} |
| **JAVA method** | CoursesController. removeLocale( publisherId courseId, locales) |
| **Errors** | "5xx Server " |
| **Response** | void |
| **Example** | /publishers/1/courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/locale/nl\_NL |

### Send Email to CGS users

|  |  |
| --- | --- |
| **Description** | Send Email to all CGS users or to specific addresses |
| **HTTP Method** | POST |
| **URL** | /mailService |
| **JAVA method** | MailController. sentMailToAll(MailBean) |
| **Errors** | "5xx Server " |
| **Response** | void |
| **Example** | / mailService  Body:  {  " hostname": "smtp.googlemail.com" //optional - by default smtp.googlemail.com  " port": 465 //optional - 465  "user":"t2kcgs" //optional - by default t2kcgs  " password": "1Qaz@wsx " //optional - by default "1Qaz@wsx"  " sSLOnConnect" : true //optional - by default true  " title": "some mail subject" //**required**  " body":"some body for this mail" //**required**  " toAddress":["avi@timetoknow.com", " lili@timetoknow.com" ]  //optional - by default will be sent to **ALL** CGC users if left empty  " bccAddress":[ moshe@timetoknow.com"]  //optional – empty by default  } |

### NarrationService.ivona

|  |  |
| --- | --- |
| **Description** | Ivona narration service |
| **HTTP Method** | POST |
| **URL** | /narrationService/ ivonaService |
| **JAVA method** |  |
| **Errors** | "5xx Server " |
| **Response** | byte[] |
| **Example** | {  "email" : "dewe@sds.com" //**required**  "apiKey" : "ef35423ht6ksdgdgsdfger4"//**required**  "codecId":"mp3/22050"//**required**  "voiceId": "ro\_carmen"//**required**  "contentType":"text/plain"//**required**  "params": {"Paragraph-Break":"400" , ... }//**required** – may be empty {}  } |

## Lesson management

### Get lessons properties

|  |  |
| --- | --- |
| **Description** | Returns an array of a Lesson-Properties , **ALL** lessons propertiess of the given course  , regardless of the lastModified value.  . |
| **HTTP Method** | GET |
| **URL** | /publishers/{publisherId}/courses/{courseId}/lessons |
| **JAVA method** | getLessonsProperties(int publisherId, String courseId) |
| **Errors** | "5xx Server "  "404 Not Found" - in case of lesson not found |
| **Example** | /publishers/{publisherId}/courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/lessons |

### Get lesson

|  |  |
| --- | --- |
| **Description** | Returns a lesson doc. |
| **HTTP Method** | GET |
| **URL** | publishers/{publisherId}/Courses/{CourseId}/lessons/{lessonId} ?  last-modified={lastModified} non-mandatory  last-modified param : if empty returns the lesson , if not, check if the server has a newer instance to return. if the not, returns NULL. |
| **JAVA method** | getLesson(CourseId , lessonId, lastModified) |
| **Errors** | "5xx Server "  "404 Not Found" - in case of lesson not found |
| **Example** | /publishers/{publisherId}/Courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/lessons/78adcb81-f918-be8e-3dbe-f2e7eed59887 |

### Save lesson

|  |  |
| --- | --- |
| **Description** | Saves the lesson. Saves a given lesson document |
| **HTTP Method** | PUT |
| **URL** | /publishers/{publisherId}/Courses/{CourseId}/lessons/{lessonId} |
| **JAVA method** | saveLesson(Lesson) |
| **Errors** | "5xx Server " |
| **Response** | On success (code 200) response body will include the header section of the saved document:  "header":{ "last-modified": "2345234523"} |
| **Example** | /publishers/{publisherId}/Courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/lessons/78adcb81-f918-be8e-3dbe-f2e7eed59887 |

### Save lesson Contents

|  |  |
| --- | --- |
| **Description** | Saves the lesson document and the associate components : sequences and thumnails.  The method reveives a wrapper json document which contains the lesson , course , publisher attributes and the lesson Json as string and a list of sequences and a list of thumbnails. |
| **HTTP Method** | PUT |
| **URL** | /publishers/{publisherId}/Courses/{CourseId}/lessons/{lessonId}/lessonContents |
| **JAVA method** | saveLessonContents (LessonContents) |
| **Errors** | "5xx Server " , “4xx” as json |
| **Response** | On success (code 200) response body will include the header section of the saved document:  "header":{ "last-modified": "2345234523"} |
| **Example** | /publishers/{publisherId}/Courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/lessons/78adcb81-f918-be8e-3dbe-f2e7eed59887/LessonContents  Body :  {    "courseId":"28",  "publisherId":1,  "lessonId":"2112",  "lesson": "lesson json - string ",  "sequences":[{"seqId":"111","lessonId":"2222","content":"seqContents string","lastModified":iso8601date},...],  "thumbnails":[{"seqId":"111","lessonId":"2222","content":"thumbnail string","lastModified":iso8601date},...]  } |
|  |  |

### Lock\Unlock Lesson

|  |  |
| --- | --- |
| **Description** | Tries to get an exclusive lock on a Lesson by the user (all versions) |
| **HTTP Method** | POST |
| **URL** | publishers/{publisherId}/courses/{CourseId}/lessons/{lessonId}/lock?action=aquire/relese &last-modified={ISO8601date}  params:  action: lock action – “release” will try tp release the lock on the course. “acquire” will try to acquire a lock on the course  last-modifoed: a date value , reflects the document last-modified value |
| **JAVA method** | lockLesson(lessonId) |
| **Errors** | "5xx Server "  423 LOCKED – when the acquire or release request fails |
| **Example** | /publishers/{publisherId}/Courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/lessons/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/lock?action=aquire |

### GET Lock Lesson

|  |  |
| --- | --- |
| **Description** | Polls the locking status of the current object. Who locks the object |
| **HTTP Method** | GET |
| **URL** | publishers/{publisherId}/courses/{CourseId}/lessons/{lessonId}/lock |
| **JAVA method** | acquireLock(CourseId) |
|  | {  "entityId": "69bb10b0-26a6-11e2-81c1-0800200c9a99",  "entityType": "LESSON",  "userName": "ophir.barnea",  "userEmail": "fName.lName@timetoknow.com",  "userPublisherId": 1,  “entityName”=”name”  "lockDate": 2012-11-13T14:04:27.92Z  } |
| **Errors** | "5xx Server " |
| **Example** | /publishers/{publisherId}/Courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/lessons/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/lock |

### Save lesson propertis

|  |  |
| --- | --- |
| **Description** | Updates only the given lesson properties.  Does not affect the lesson learning activities in the server DB |
| **HTTP Method** | POST |
| **URL** | /publishers/{publisherId}/Courses/{CourseId}/lessons/{lessonId} |
| **JAVA method** | saveLesson(publisherId, courseId , Lesson) |
| **Errors** | "5xx Server " |
| **Response** | On success (code 200) response body will include the header section of the saved document:  "header":{ "last-modified": "2345234523"} |
| **Example** | /publishers/{publisherId}/Courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/lessons/78adcb81-f918-be8e-3dbe-f2e7eed59887 |

### Import Lesson – copy specific assets

|  |  |
| --- | --- |
| **Description** | Copies a list of assets (specified in the request) from one course to another. This API is meant to be used during a lesson import process. |
| **HTTP Method** | POST |
| **URL** | /publishers/{destinationPublisherId}/Courses/{destinationCourseId}/import |
| **JAVA method** | copyAssets (publisherId, courseId , listOfPaths) |
| **Errors** | "5xx Server " |
| **Response** | On success (code 200) response body is empty. |
| **Example** | URL: <http://localhost/cgs/rest/publishers/1/courses/71fe61e2-26de-4464-9c7e-bcc9eaed1ecc/import>  BODY:  {  "sourceCourseId":"41d2a103-991e-4044-9a10-5474db7abe8a",  "pathsList":[  "media/0d/22/0d2297b4e8e3ae07ca7fdc62b3382f8f0e852b66a.PNG",  "media/32/09/32096be4a59914e601e2c9966b90233df47f062d.PNG",  "media/3e/43/3e4379bc73aaeede6fa65c7b620068777238d759.PNG",  "media/a5/43/a543523a61767573ee18c2ca394f3f37575cc0ac.PNG",  "media/d5/83/d5830207c51bef01de7fa93613aaa867245c5feb.PNG"  ]} |

## Package management

### Prepare for package

|  |  |
| --- | --- |
| **Description** | Request for the server to prepare the course for packaging. The server will lock  All the course contents . and on success , the package phase will be ‘PREPARED’ |
| **HTTP Method** | PUT |
| **URL** | /publishers/{publisherId}/packages/courses/{courseId} |
| **JAVA method** | preparePackaging (publisherId, courseId) |
| **Errors** | "5xx Server "  423 locks |
| **Response** | On success (code 200)  {   * "packageId":"04c49cc5-f0a6-4060-b03c-cb5bd54bedd4\_rev1-1\_1359636741995-5ef5", * “newVersion\_prod”: “1.2” * “newVersion\_preProd : “1.1.1” * "courseId":"04c49cc5-f0a6-4060-b03c-cb5bd54bedd4", * "publisherId":1, * "packagePhase":"PREPARED", * "userName":"t1", * "packStartDate":"2013-01-31T14:52:21.290+0200", * "packEndDate":*null*, * "packageOutputLocation":"c:\\tmp\\cgs\\packages\\output/04c49cc5-f0a6-4060-b03c-cb5bd54bedd4\_rev1-1\_1359636741995-5ef5.cgs", * "hostName":"TM-DT121", * "errors":[   ],   * "warnings":[   ],   * "pendingPlace":0, * "csi":*null*, * "description":"", * "releaseNote":"", * "packageModeEnum":"PRE\_PRODUCTION"   } |
| **Example** | /publishers/2/packages/courses/04c49cc5-f0a6-4060-b03c-cb5bd54bedd4 |

### Publish course

|  |  |
| --- | --- |
| **Description** | Request for start publishing a course , this call should be followed by the prepare request.   * The server will start to package the course from the prepared phase. |
| **HTTP Method** | POST |
| **URL** | /publishers/{publisherId}/packages/{packageId}? publishMode={pre-production/production}  Form params:  description : a string with the publish description  releaseNote: a string with the publish release-Note |
| **JAVA method** | packageCourse (packageId) |
| **Errors** | 404 |
| **Response** | On success (code 200)  CGS Package object (JSON)  {  "version": "1.1",  "courseId": "04c49cc5-f0a6-4060-b03c-cb5bd54bedd4",  "publisherId": 1,  "packagePhase": "PREPARED",  "userName": "t1",  "packStartDate": "2013-01-31T14:52:21.290+0200",  "packEndDate": "2013-01-31T15:02:11.206+0200",  "packageOutputLocation": "c:\\tmp\\cgs\\packages\\output/04c49cc5-f0a6-4060-b03c-cb5bd54bedd4\_rev1-1\_1359636741995-5ef5.cgs",  "hostName": "TM-DT121",  "errors": [],  "warnings": [],  "pendingPlace": 0,  "csi": null,  "description": "dasd",  "releaseNote": "fsdfdsf",  "packageModeEnum": "PRE\_PRODUCTION",  "packageId": "04c49cc5-f0a6-4060-b03c-cb5bd54bedd4\_rev1-1\_1359636741995-5ef5"  } |
| **Example** | /publishers/2/packages/2121212 |

### Get CGSPackage

|  |  |
| --- | --- |
| **Description** | Returns a CGS Package object (JSON) by ID |
| **HTTP Method** | GET |
| **URL** | /publishers/{publisherId}/packages/{packageId} |
| **JAVA method** | getCGSPackage (publisherId, packageId) |
| **Errors** | 404 |
| **Response** | On success (code 200)  CGS Package object (JSON)  {  "version": "1.1",  "courseId": "04c49cc5-f0a6-4060-b03c-cb5bd54bedd4",  "publisherId": 1,  "packagePhase": "PREPARED",  "userName": "t1",  "packStartDate": "2013-01-31T14:52:21.290+0200",  "packEndDate": "2013-01-31T15:02:11.206+0200",  "packageOutputLocation": "c:\\tmp\\cgs\\packages\\output/04c49cc5-f0a6-4060-b03c-cb5bd54bedd4\_rev1-1\_1359636741995-5ef5.cgs",  "hostName": "TM-DT121",  "errors": [],  "warnings": [],  "pendingPlace": 0,  "csi": null,  "description": "dasd",  "releaseNote": "fsdfdsf",  "packageModeEnum": "PRE\_PRODUCTION",  "packageId": "04c49cc5-f0a6-4060-b03c-cb5bd54bedd4\_rev1-1\_1359636741995-5ef5"  }  Possible Phases:  STARTING  PREPARED  PENDING  IN\_PROGRESS  FAILED  CANCELED  PUBLISHING  COMPLETED |
| **Example** | /publishers/2/packages/2121212 |

### Cancel Packaging Process

|  |  |
| --- | --- |
| **Description** | Cancel an execution of the packaging process.  The package task may be in a pending queue or already running. the request tries to cancel the job at all cases. |
| **HTTP Method** | PUT |
| **URL** | /publishers/{publisherId}/packages/{packageId}?action=cancel |
| **JAVA method** | cancelPackageProcess (publisherId, packageId) |
| **Errors** | 404 |
| **Response** | On success (code 200) |
| **Example** | /publishers/2/packages/2121212?action=cancel |

### *Get CGSPackages By Publisher*

|  |  |
| --- | --- |
| ***Description*** | *Returns a list of all published(packages) from the given publisher* |
| ***HTTP Method*** | *GET* |
| ***URL*** | */publishers/{publisherId}/packages* |
| ***JAVA method*** | *getCGSPackagesByPublisher (publisherId)* |
| ***Errors*** |  |
| **Response** | Array Of CGSPackage JSON |
| **Example** | /publishers/2/packages |

### *Get CGSPackages By course*

|  |  |
| --- | --- |
| ***Description*** | *Returns a list of all published(packages) from the given course* |
| ***HTTP Method*** | *GET* |
| ***URL*** | */publishers/{publisherId}/packages/* *courses/{courseId}* |
| ***JAVA method*** | *getCGSPackagesByCourse (publisherId,courseId)* |
| ***Errors*** |  |
| **Response** | Array Of CGSPackage JSON |
| **Example** | /publishers/2/packages/*courses/dasd2wddd1s2wecfef* |

### *Get Active CGSPackages By userName*

|  |  |
| --- | --- |
| ***Description*** | *Returns a list of all active published(packages) from the given user*  *The user name is the user that invoked the publish process*  *Active can be in the following states:*  *STARTING*  *PREPARED*  *PENDING*  *IN-PROGRESS*  *PUBLISHING* |
| ***HTTP Method*** | *GET* |
| ***URL*** | */publishers/{publisherId}/packages?userName={userName}* |
| ***JAVA method*** | *getActiveCGSPackagesByUseName (publisherId,username)* |
| ***Errors*** |  |
| **Response** | Array Of CGSPackage JSON |
| **Example** | /publishers/2/packages?*userName=dan.brown* |

## Utils services

### Validate course

|  |  |
| --- | --- |
| **Description** | A service to validate a course json against the schema (found in the t2kschema project) |
| **HTTP Method** | POST |
| **URL** | /utils/validations/course |
| **JAVA method** | validateCourseJson(course) |
| **Errors** |  |
| **Response** | A json response with a validation message(s) |
| **Example** |  |

### Validate lesson

|  |  |
| --- | --- |
| **Description** | A service to validate a lesson json against the schema (found in the t2kschema project) |
| **HTTP Method** | POST |
| **URL** | /utils/validations/lesson |
| **JAVA method** | validateLessonJson (lesson) |
| **Errors** |  |
| **Response** | A json response with a validation message(s) |
| **Example** |  |

## Applets management

### Get all allowed applets for specific publisher

|  |  |
| --- | --- |
| **Description** | Returns an array of Applet artifacts |
| **HTTP Method** | GET |
| **URL** | /publishers/{publisherId}/applets |
| **JAVA method** | getAppletsAllowedForPublisher (int publisherId) |
| **Errors** | "5xx Server "  "404 Not Found" - in case publisher is not found |
| **Example** | /publishers/1/applets |

### Get applet manifest of specific course

|  |  |
| --- | --- |
| **Description** | Returns an array of Applet manifest |
| **HTTP Method** | GET |
| **URL** | /publishers/{publisherId}/courses/{courseId}/applets?last-modified=<timestamp> |
| **JAVA method** | getAppletManifest(String courseId, long lastModified) |
| **Errors** | "5xx Server "  "404 Not Found" - in case course not found |
| **Example** | /publishers/1/Courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/applets?last-modified =123654 |

### Add applet to the course

|  |  |
| --- | --- |
| **Description** | Add applet to course |
| **HTTP Method** | POST |
| **URL** | /publishers/{publisherId}/courses/{courseId}/applets/{appletId} |
| **JAVA method** | addApplet(String courseId, String appletId) |
| **Errors** | "5xx Server "  423 locks  "404 Not Found" - in case course not found, applet not found |
| **Example** | /publishers/{publisherId}/courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887/applets/?5b1dcb81-f918-be8e-3dbe-f2e7eed59992 |

### Update course applet to latest version

|  |  |
| --- | --- |
| **Description** | Update course uplet to newer version |
| **HTTP Method** | PUT |
| **URL** | /publishers/{publisherId}/courses/{courseId}/applets |
| **Request Body** | { "appletIds": [ "77777789-5656-66565", "66666789-5656-66565", … ] } |
| **JAVA method** | updateApplet(String courseId, appletIds) |
| **Errors** | "5xx Server "  423 locks  "404 Not Found" - in case course not found, applet not found |
| **Example** | /publishers/{publisherId}/courses/2a1dcb81-f918-be8e-3dbe-f2e7eed59887 |

## Standards

### Get headers of all available standard packages

|  |  |
| --- | --- |
| **Description** | Returns headers of all latest standard packages (latest = highest version) |
| **HTTP Method** | GET |
| **URL** | /standards |
| **JAVA method** | getStandardPackagesHeaders() |
| **Errors** | "5xx Server " |
| **Example** | /standards |

### Get a standard package

|  |  |
| --- | --- |
| **Description** | Returns the standard package, including the complete standards tree |
| **HTTP Method** | GET |
| **URL** | standards/{packageName}/subjectAreas/{subjectArea}?version={version}  **version** – optional, if not provided, latest version will be returned |
| **JAVA method** | getStandardsPackage () |
| **Errors** | "5xx Server " |
| **Example** | Standards/nysls/subjectAreas/math?version=1.2 |

## Text To Speech Providers

### Get all text to speech providers

|  |  |
| --- | --- |
| **Description** | Returns all text to speech providers objects |
| **HTTP Method** | GET |
| **URL** | /ttsproviders |
| **JAVA method** | StaticContentController. getTextToSpeechProviders () |
| **Errors** | "5xx Server " |
| **Example** | / ttsproviders |

## Proxy service

### Propogate request to external domain

|  |  |
| --- | --- |
| **Description** | Propogates request to external service. Omits the following headers: authorization, cookie, set-cookie |
| **HTTP Method** | GET/POST/PUT/DELETE |
| **URL** | /proxy?<external encoded service URL> |
| **JAVA method** | ProxyController. proxyRequest () |
| **Errors** | "5xx Server " ; Same error as external domain response |
| **Example** | / proxy? %20http%3A%2F%2Fwww.google.co.il%2F%3Fgws\_rd%3Dcr%26ei%3DB8iuUrC7EuSYyQO74oGABA  For <http://www.google.co.il/?gws_rd=cr&ei=B8iuUrC7EuSYyQO74oGABA> page |

## Error handling

Rest API errors will be exposed in a unified format. This format holds the information about the error cause, relevant data and a message (as moreInfo) for developer usage.

Each of the errors must be in correlation to HTTP TYPE which points on the client-server interaction problem.

The Rest Error format is:

{

"errorId": int, -the error code, also the template Id to use

"data":Object, -the error Data, which the template use.

"httpStatus":String, -http status of the rest response(not exposed)

"message":String -extra info about the problem . (not exposed)

}

Sample Error :

{

"errorId":2000,

"data":[

{

"entityId":"234d23fsdfklyussd45",

"entityType":"COURSE",

"userName":"ophir.barnea",

"userEmail":"[ophir.barnea@timetoknow.com](mailto:ophir.barnea@timetoknow.com)",

"userPublisherId":2,

"lockDate": "2013-01-17T11:23:31.046+0200"

},

{

"entityId":"d2hafwsdasf234f23r23",

"entityType":"COURSE",

"userName":"t1",

"userEmail":"[t1@mail.com](mailto:t1@mail.com)",

"userPublisherId":2,

"lockDate": "2013-01-17T11:23:31.046+0200"

}

],

"httpStatus":"LOCKED",

"message":*”server found few lock users …. Please check the cache “* }

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Error Id** | **Http Status** | **Description** | **Error Data type** |  |
| 1000 | 422 | CONTENT\_IS\_NOT\_VALID | String |  |
| 1001 | 422 | CONTENT\_URL\_IS\_NOT\_VALID | String |  |
| 1002 | 422\* | SCHEMA\_IO\_ERROR | String |  |
| 2000 | 423 | CONTENT\_IS\_LOCKED | Array of Lock\*\* |  |
| 2001 | 423 | CONTENT\_IS\_NOT\_OWNED\_BY\_USER | Array of Lock\*\* |  |
| 3000 | 404 | RESOURCE\_NOT\_FOUND | String |  |
| 3001 | 413 | FILE\_TOO\_BIG | String |  |
| 3002 | 422 | FILE\_IS\_EMPTY\_OR\_NOT\_IN\_REQUEST | String |  |
| 4000 | 409 | CONTENT\_IS\_NOT\_IN\_SYNC | String |  |
| 0 | 500 | INTERNAL\_SERVER\_ERROR | String |  |

\*\*Lock Json:

{

"entityId":String,

"entityType":String/enum

"userName":String,

"userEmail":String,

"userPublisherId":int,

"lockDate": Date ISO8601

}