

Appendix A – Institutions in the Consortium

This appendix lists the 28 institutions within the Consortium, and the number of Full-Time Equivalents (FTEs) at each institution per year. One (1) FTE is defined as 60 ECTS credits produced.

Arkitektur- og designhøgskolen i Oslo	700 FTE
Dronning Mauds Minne Høgskole	700 FTE
Forsvarets høgskole	500 FTE
Høgskolen i Molde	2050 FTE
Høgskulen i Volda	2500 FTE
Høgskolen i Østfold	4350 FTE
Høgskulen på Vestlandet	11700 FTE
Høyskolen Kristiania*	9700 FTE
Kriminalomsorgens høgskole- og utdanningssenter	400 FTE
Lovisenberg diakonale høgskole	1050 FTE
MF vitenskapelig høyskole	500 FTE
NLA Høgskolen	2350 FTE
Nord universitet	6800 FTE
Norges Handelshøyskole	3300 FTE
Norges idrettshøgskole	1200 FTE
Norges miljø- og biovitenskaplige universitet	5700 FTE
NTNU	31700 FTE
OsloMet	16550 FTE
Politihøgskolen	1300 FTE
Sámi allaskuvla/Samisk høgskole	100 FTE
Universitet i Agder	10100 FTE
Universitetet i Bergen	14300 FTE
Universitetet i Innlandet	9200 FTE
Universitetet i Oslo	18550 FTE
Universitetet i Stavanger	8900 FTE
Universitetet i Sørøst-Norge	11600 FTE
Universitetet i Tromsø	10950 FTE
VID vitenskapelige høgskole	3750 FTE
Total	138 750 FTE

* Current number of FTEs. They might increase the scope of deliveries and increase the number of FTEs by approximately 2,400.

Source for the annual number of ECTS credits produced by each institution: The Norwegian Database for Higher Education (DBH).

Appendix B – List of current integrations with external systems

Present status, April 2026.

The list is meant to exemplify the variety of the current integrations between the reading list system and external systems, and to illustrate the need for the reading list system to be a robust component for integrations.

- The list is not meant to be comprehensive or exhaustive.
- Integration between the reading list system and the LSP and Discovery is not included.
- Names and abbreviations used in the table are explained at the bottom of the document.

No	Data	From <-> To	Protocol	Format	Volume	Frequency	Comments
1	Student records	FS -> LSP institution	FTP	XML	Thousands	Nightly	Integration profile in the LSP imports user data about students from student database hub (FS) to the LSP institution.
2	Student records	FS -> LSP institution	API	JSON/XML	Hundreds	Hourly	Sikt application imports user data about students from student database hub (FS) to the LSP institution.
3	Staff records	Mother institution -> LSP institution	FTP	TXT	Hundreds	On demand	Sikt application imports user data about staff directly from mother institutions to the LSP institution.
4	Copyright status	Reading list system <-> Kopinor	API	JSON	Hundreds	On demand	Bolk – Kopinor application used by Norwegian higher education institutions to manage and clear copyright permissions for course materials.

5	Courses	FS -> Reading list system	API + import-functionality	JSON/CSV	Thousands	Nightly	Course information are exported from the SIS-system, converted and imported to the Reading List system. The import is done each semester with potential update imports.
6	Reading lists	Reading list system -> FS	API	JSON	Thousands	Nightly	Permalink to the reading lists and the content of the reading lists are retrieved from the reading List System and imported into the SIS-system.
7	Reading lists	LMS <-> Reading list system	LTI		Thousands	On demand	The reading lists are visible/integrated directly into the LMS via an LTI-plugin.
8	Reading lists / Citations	Reading list system ->?	API	JSON	Thousands	On demand	Read access to reading lists and citations is provided under an open license, made available to all through a proxy interface to the underlying API.

FS = Felles Studentregister (The SIS-system used by the institutions).

LMS = Learning Management System

LSP = Library Services Platform

SIS = Student Information System. Contains information about the courses and the students at the institution

Appendix C – Bolk

Workflow in the current integration:

In the current reading list system, list creators can open a citation and choose to create a Bolk request from there. Only instructors with a specific role and librarians can see the integration and do so.

The reading list the citation is in must be associated with a course. If the number of participants is already registered on the course, this number will appear automatically in the Bolk request creation form; same thing with page range(s).

When the list creator press “Send”, the request is automatically transferred to Bolk, where a compendium/excerpt is automatically created. Bolk then returns a “Clearance Approved” or “Clearance - Action Required” status based on what percentage of the book the page range(s) represents or in case of an error message.

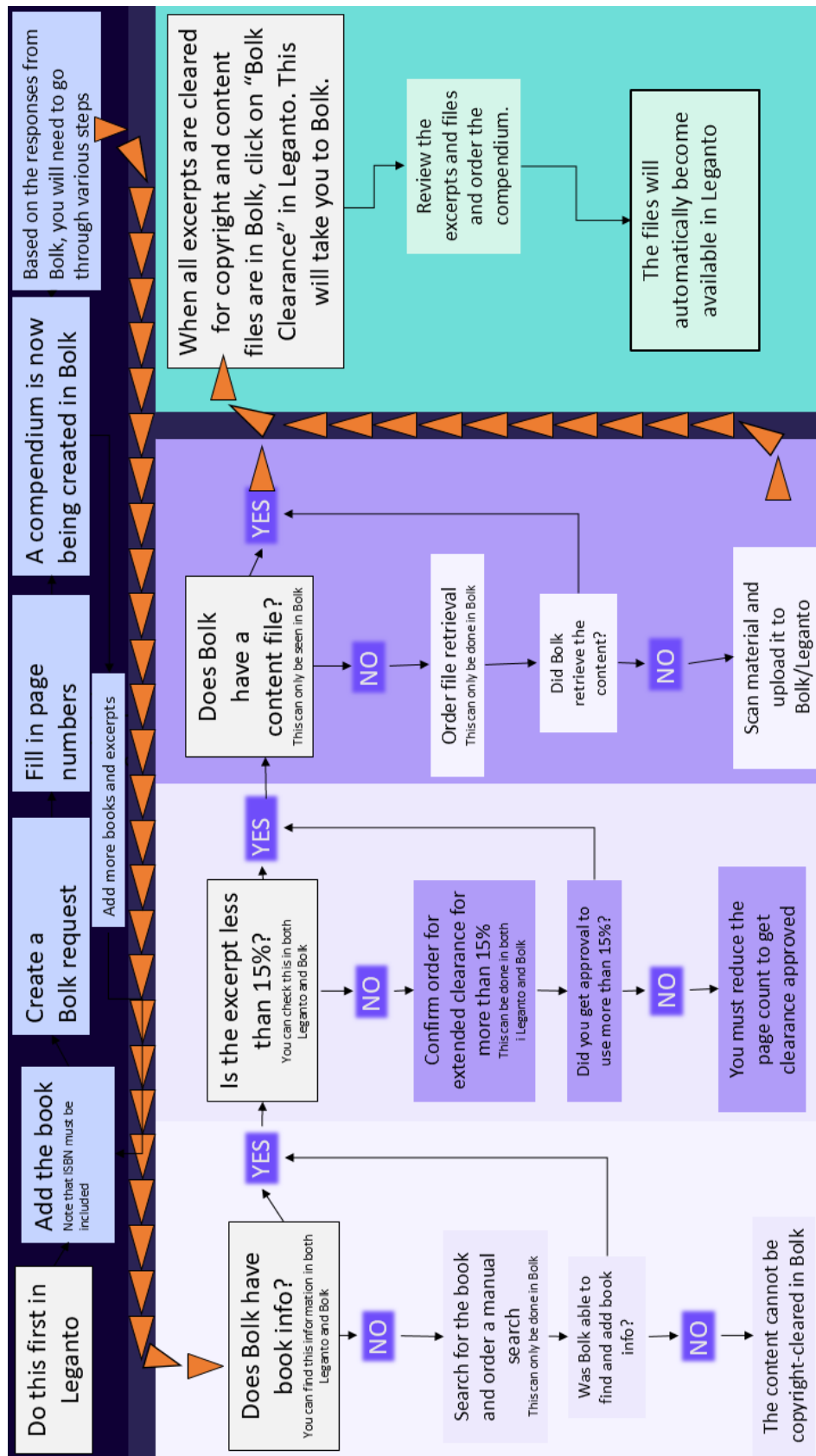
If “Action Required” appears, the list creator must open the citation again to find out what they must do next. If the percentage is above 15%, they need to click on “Extend copyright clearance”. The request will then get the status “Waiting for Approval” and needs to be approved by an administrator in Bolk.

After that, the status is synced either manually in Alma or Leganto, or through a nightly job. For some institutions there is an extra step after that where the list creator needs to approve the price. There has been a discussion on whether this step is necessary, and how the statuses for each step of the process could be improved.

Once all the Bolk requests on a list have been approved, the list creator can complete the process by clicking on “Bolk clearance”. They are then taken directly to Bolk, on a page where they can review the requests and the files before ordering the compendium. When this is done, files corresponding to each registered page range, when available, will appear on the citations.

After this, no other Bolk request can be registered on the reading list unless Kopinor unlocks the compendium in Bolk.

Flowchart: Copyright Clearance Process in the Leganto/Bolk integration



DD130 - BULK-API RIGHTS CLEARANCE

- **Version:** • 0.1
- **Status:** • Draft
- **Author:** • Siham S. Feklani

netcompany

Introduction

The intention of this document is to describe how the different user stories for the Bulk API is intended to be implemented and how the flow in the different processes are. This is a high-level sequence diagram for one excerpt and is intended to show how one request flows. The process may repeat itself. See the technical specification for the definition of the API.

Overview



Rights clearance

User stories

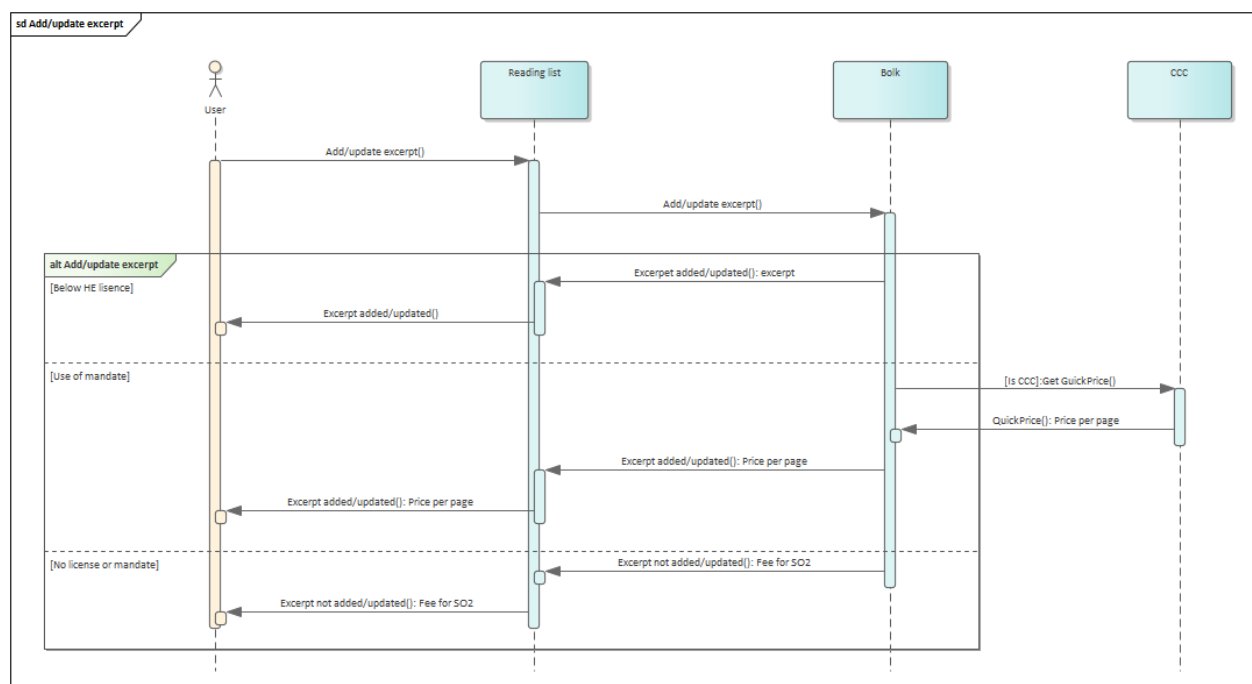
As...
(role)

...I shall...
(what)

...so that...
(why)

user	see information about existing mandate when I add an excerpt	I can use more from a publication than HE License
user	see information about existing mandate when I update the page range for an excerpt	I can use more from a publication than HE License
user	order extended rights clearance when I add excerpt that exceed existing license/mandate	I can use more from at publication than existing licenses and mandates allow
user	Decline offer for ordering extended rights clearance when I add excerpt that exceed existing license/mandate	I can reconsider the use of the excerpt in question
user	see the status of the extended rights clearance order	I can check when this goes through
user	I shall get information about the cost of extended rights clearance	I can decide whether to order or not.
user	I shall be able to approve offered extended rights	I can legally use the excerpt that exceeds existing licenses and mandates in Bolk
user	I shall be able to decline offered extended rights	I can edit page range or remove the excerpt(s) in my Reading List

Add/update excerpt



Adding excerpt

When adding an excerpt, the endpoint `api/curriculums/{curriculumId}/excerpts` is used. This will return a positive response with three different outcomes, the different responses are described below; “Using HE license”, “using mandate”, “need to order extended rights clearance”.

Update excerpt

When updating an existing excerpt, the endpoint `api/curriculums/{curriculumId}/excerpts/{excerptId}` is used. This will return a positive response with three different outcomes, the different responses are described below; “using HE license”, “using mandate”, “need to order extended rights clearance”.

Responses

Using HE license

The excerpt was added successfully and there is no extra cost connected to the usage. No further action is required.

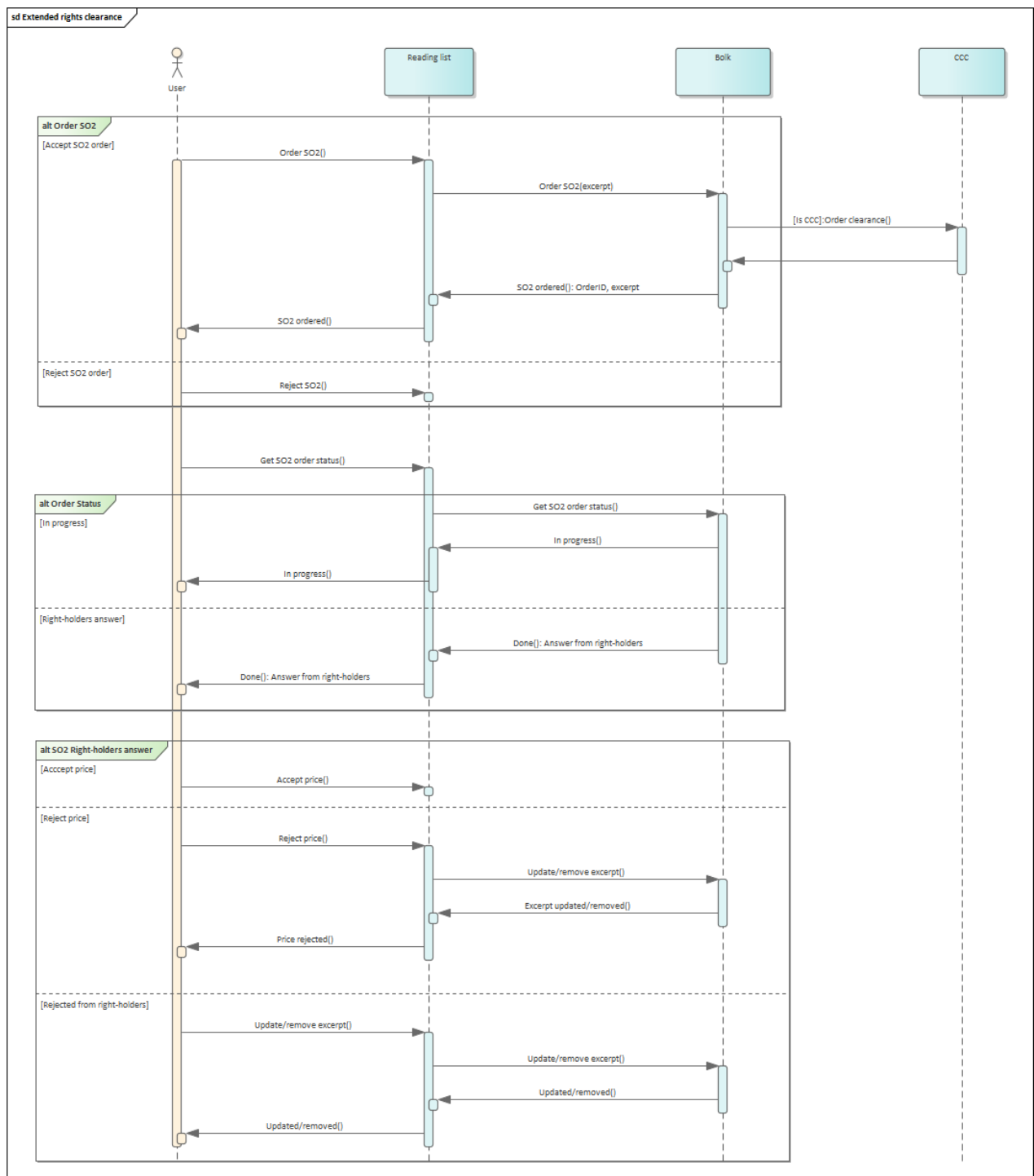
Using mandate

The excerpt was added successfully with the usage of an existing mandate. The user needs to see the information about usage per page.

Need to order extended rights clearance

The excerpt was not added because there is a need for extended rights clearance. This is a manual process and has a fee. The user is presented with the fee and need to decide if they want to order the extended rights clearance.

Extended rights clearance



Order extended rights clearance

When the usage of an excerpt extends either HE-license or existing mandates and extended rights clearance is needed. This has a fee and needs to be ordered by the user.

For ordering the extended rights clearance use the endpoint `api/curriculums/{curriculumId}/excerpts/clearance`.

If the excerpt already exists in Bolk please provide the excerpt-id to ensure that the correct excerpt is updated and that the extended clearance is ordered for that excerpt. If the excerpt-id is not provided the excerpt will be added to the curriculum.

See the "ordered" response for the return value.

Get order status

After an order has been placed there is a manual process for getting the rights from the right-holders.

For getting the status use `api/curriculums/{curriculumId}/excerpts/{excerptId}/orderStatus/{orderId}`

The response contains information about the order and the state that it is in. See "in progress" and "done" response.

Responses

Ordered

The extended rights clearance is ordered and returns an order-id, this is later used when fetching the status for the order.

In the response is also the added/updated excerpt.

In progress

The extended rights clearance is not done and is still in progress. No further action required.

Done

The extended rights clearance is done and there is a response from the right-holders.

If the response has a price per page, the price can either be accepted or rejected. For acceptance there is no need to update Bolk because the price is already added to the excerpt in the curriculum. If the price is rejected the excerpt needs to be updated or deleted.

If the response has a rejection from the right-holders the excerpt needs to be updated or removed from the curriculum.

SwaggerAPI

Definition

This API enables creation, editing and rights clearance of curriculums from Kopinor. To be able to use this API you must have:

1. OAuth token
2. An API-Key
3. A valid FeideID

4. Your FeideID must be connected to a valid Kopinor user.

When performing calls to the API, both API-key, FeideID and a valid OAuth token from Dataporten Which is the way to authorize access to Bolk, must be provided as headers. Only requests gone through Dateporten (GateKeeper) will be forwarded to Bolk.

Available authorizations ✕

curriculum_oauth (http, Bearer) ←

OAuth-token from Dataporten

Value:

AuthorizeClose

curriculum_api (apiKey) ←

The API-key from Kopinor

Name: X-API-KEY

In: header

Value:

AuthorizeClose

And the feideID should be added as parameter in the header. The request will be sent on behalf of the user having this feideID. Which again we use to identify the Tiril user in Bolk.

X-FEIDE-ID * required

Feide ID

string

(header)

Overview

Here is an overview of what we have in the API, which actions are implemented for curriculums, excerpts, ordering rights clearance and ordering/ submit a curriculum. We will go through alle the endpoints, what is expected as parameters and wich responses are returned.

Curriculum API 0.3.0 OAuth2

This API enables creation, editing and rights clearance of curriculums from Kopinor.

To be able to use this API you must have;

1. An API-key
2. A valid FeideID
3. Your FeideID must be connected to a valid Kopinor-user

This can be obtained with help from Kopinor.

When performing calls to the API, both an API-key, FeideID and a valid OAuth token from Dataporten must be provided as headers.

Authorize 

Curriculum Everything about curriculums. Add, edit and cancel a curriculum.

PUT /api/curriculums Add a new curriculum

GET /api/curriculums/{curriculumId} Get details for a curriculum.

POST /api/curriculums/{curriculumId} Edit details for a curriculum.

DELETE /api/curriculums/{curriculumId} Delete curriculum.

Ordering Submit an order.

POST /api/curriculum-orders/{curriculumId} Order a curriculum

Excerpt Everything about excerpts. Adding, removing or editing excerpts in a curriculum.

GET /api/curriculums/{curriculumId}/excerpts List of all excerpts in a curriculum.

PUT /api/curriculums/{curriculumId}/excerpts Add an excerpt

POST /api/curriculums/{curriculumId}/excerpts/{excerptId} Change page ranges or the order for excerpt.

DELETE /api/curriculums/{curriculumId}/excerpts/{excerptId} Delete an excerpt from curriculum.

Rights Clearance Ordering extended rights clearance when the usage of an excerpt exceeds either HE-licence and existing mandate, see the order status

PUT /api/curriculums/{curriculumId}/excerpts/clearance Order extended rights clearance

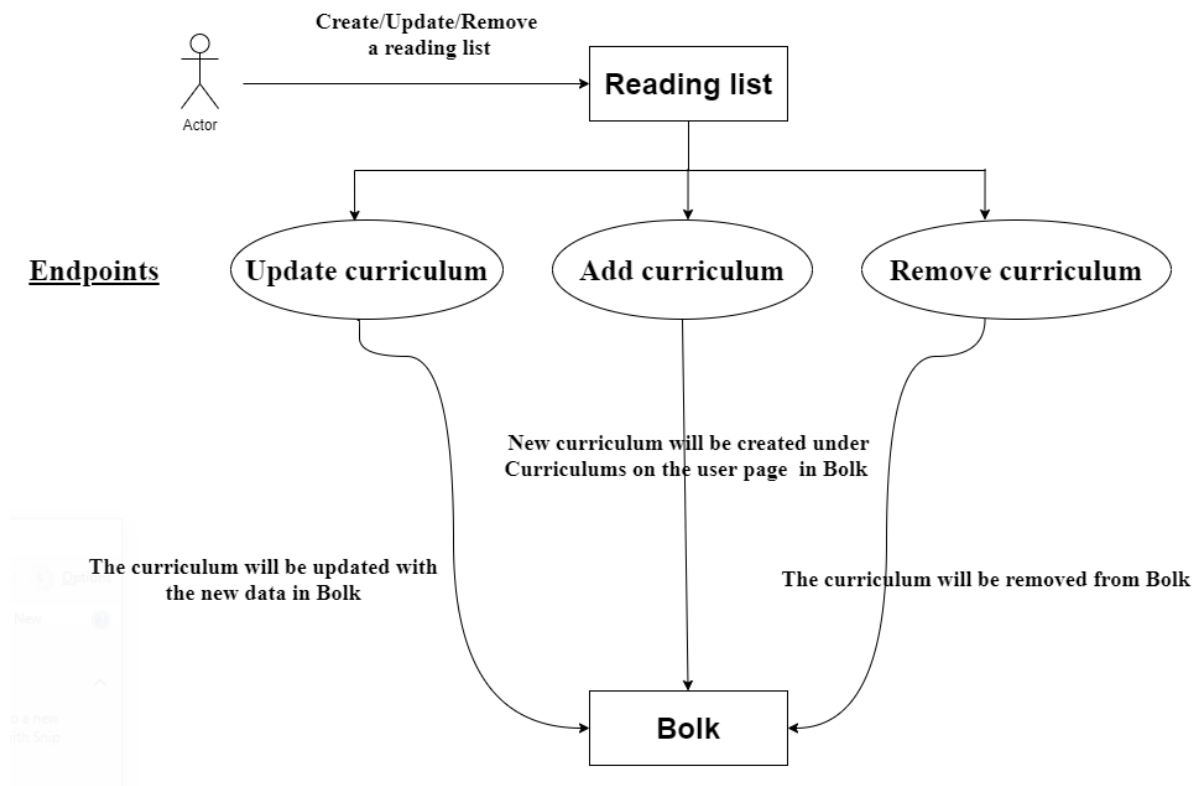
GET /api/curriculums/{curriculumId}/excerpts/{excerptId}/orderStatus/{orderId} Check status for ordered extended rights clearance

Schemas

Endpoints

Curriculum

Example of how the flow will look like:

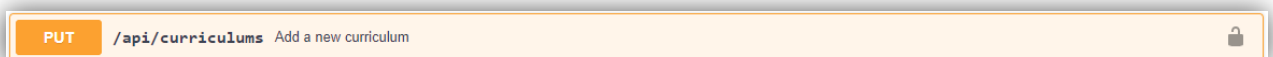


All results will be visible in Bolk under the user account the requests are sending on behalf of.

Adding a curriculum

For adding a curriculum, the endpoint `"/api/curriculums"` is used. This will return a curriculum details as a positive response, otherwise an error code with a message describing what went wrong.

The result of this will be a curriculum created in Bolk on behalf of the organization the user belongs to, the students will be able to use it when it is submitted. This curriculum must include elements which will present the reading list in Reading List systems, which can be added by using the endpoint for adding excerpt.



Parameters

Create a curriculum with the necessary fields which sends as a request body as the following:

```
{
  "courseCode": "SAMF-105",
  "courseName": "Curriculum for ISH3100 International Human Resource",
  "semester": "Autumn",
  "year": 2020,
  "teacher": "Leif",
  "semesterStart": "2020-08-30",
  "semesterEnd": "2020-11-30"
}
```

To create a curriculum, we need:

- a code and a name of the course, both are “strings”.
- The semester the course taking place, the semester should have one of the values “Autumn”, “Spring” or “Summer”.
- The year the course taking place, for example “2020”.
- The name of the teacher
- The start and end date of the semester. They are strings with date-time format.

Responses

Error code	Result
201	Curriculum created successfully. Returns a curriculum details: <pre>{ "curriculumId": 0, "courseCode": "SAMF-105", "semester": "2020-H", "semesterStart": "2020-08-30", "semesterEnd": "2020-11-30", "createdDate": "2020-10-22T10:30:55.137Z", "orderDate": "2020-10-22T10:30:55.137Z", "courseName": "string", "creatorId": 0, "lecturer": "string", "orderStatus": "DELIVERED" }</pre>
401	Unauthorized <pre>The client must authenticate itself to get the requested response</pre>
422	The request was well-formed but was unable to be followed du to: <pre>Not possible to process this request due to : - Missing mandatory fields. - Fields set wrongly (wrong start and/or end of semester). - Curriculum not Under Construction.</pre>

500	Internal server error The server has encountered a situation it doesn't know how to handle
503	Service Unavailable The server is unavailable or down for maintenance

Getting details for a curriculum

For getting the details for a curriculum, the endpoint “/api/curriculums/{curriculumId}” is used.

GET

/api/curriculums/{curriculumId} Get details for a curriculum.

Parameters

In addition to feideld, we need a curriculumID sent in path:

curriculumId * required

integer(\$int32)

(path)

Curriculum ID

curriculumId - Curriculum ID

Responses

Error code	Result
200	Curriculum found.Returns a curriculum details: <pre>{ "curriculumId": 0, "courseCode": "SAMF-105", "semester": "2020-H", "semesterStart": "2020-08-30", "semesterEnd": "2020-11-30", "createdDate": "2020-10-22T10:30:55.137Z", "orderDate": "2020-10-22T10:30:55.137Z", "courseName": "string", "creatorId": 0, "lecturer": "string", "orderStatus": "DELIVERED" }</pre>
400	Invalid request, due to:

	- curriculumId must be an integer and larger than 0. - excerptId must be an integer and larger than 0. - Missing content
401	Unauthorized The client must authenticate itself to get the requested response
404	Resource not found: Due to : - Curriculum not found. - Excerpt not found
500	Internal server error The server has encountered a situation it doesn't know how to handle
503	Service Unavailable The server is unavailable or down for maintenance

Update a curriculum

The user is allowed to change these fields related to a curriculum:

- Title
- Lecturer
- Semester start
- Semester end

Editing a curriculum is only allowed when it has order status 'UNDER_CONSTRUCTION'. The start and the end of semester should be within the semester period chosen.

For updating a curriculum the endpoint “/api/curriculums/{curriculumId}” is used .

POST	/api/curriculums/{curriculumId} Edit details for a curriculum.	
------	--	---

Parameters

In addition to feideld, we need a curriculumId sent in path and an object containing necessary fields to update a curriculum sent as a request body:

curriculumId * required	Curriculum ID
integer(\$int32)	
(path)	curriculumId - Curriculum ID

Curriculum details (se more details on create curriculum for fields description):

```
{
  "courseCode": "SAMF-105",
  "courseName": "Curriculum for ISH3100 International Human Resource",
  "semester": "Autumn",
  "year": 2020,
  "teacher": "Leif",
  "semesterStart": "2020-08-30",
  "semesterEnd": "2020-11-30"
}
```

Responses

Error code	Result
200	Curriculum updated successfully.Returns a curriculum details: <pre>{ "curriculumId": 0, "courseCode": "SAMF-105", "semester": "2020-H", "semesterStart": "2020-08-30", "semesterEnd": "2020-11-30", "createdDate": "2020-10-22T10:30:55.137Z", "orderDate": "2020-10-22T10:30:55.137Z", "courseName": "string", "creatorId": 0, "lecturer": "string", "orderStatus": "DELIVERED" }</pre>
400	Invalid request, due to: <pre>- curriculumId must be an integer and larger than 0. - excerptId must be an integer and larger than 0. - Missing content</pre>
401	Unauthorized <pre>The client must authenticate itself to get the requested response</pre>
404	Resource not found:

	Due to : - Curriculum not found. - Excerpt not found
422	The request was well-formed but was unable to be followed du to: Not possible to process this request due to : - Missing mandatory fields. - Fields set wrongly (wrong start and/or end of semester). - Curriculum not Under Construction.
500	Internal server error The server has encountered a situation it doesn't know how to handle
503	Service Unavailable The server is unavailable or down for maintenance

Remove a curriculum

Delete an existing curriculum. This action is only available when curriculum has order status 'UNDER_CONSTRUCTION'. To remove a curriculum the endpoint “/api/curriculums/{curriculumId}” is used.

DELETE /api/curriculums/{curriculumId} Delete curriculum.

Parameters

In addition to feideld, we need a curriculumID sent in path:

curriculumId * required
integer(\$int32)
(path)

Curriculum ID

Responses

Error code	Result
------------	--------

200	Curriculum deleted.Returns a curriculum details: <pre>{ "curriculumId": 0, "courseCode": "SAMF-105", "semester": "2020-H", "semesterStart": "2020-08-30", "semesterEnd": "2020-11-30", "createdDate": "2020-10-22T10:30:55.137Z", "orderDate": "2020-10-22T10:30:55.137Z", "courseName": "string", "creatorId": 0, "lecturer": "string", "orderStatus": "DELIVERED" }</pre>
400	Invalid request, due to: <pre>- curriculumId must be an integer and larger than 0. - excerptId must be an integer and larger than 0. - Missing content</pre>
401	Unauthorized <pre>The client must authenticate itself to get the requested response</pre>
404	Resource not found: <pre>Due to : - Curriculum not found. - Excerpt not found</pre>
500	Internal server error <pre>The server has encountered a situation it doesn't know how to handle</pre>
503	Service Unavailable <pre>The server is unavailable or down for maintenance</pre>

Excerpt

Adding an excerpt

This will return a positive response with three different outcomes, the different responses are described below

- “Using HE license”: The excerpt was added successfully and there is no extra cost connected to the usage. No further action is required.
- “Using mandate”: The excerpt was added successfully with the usage of an existing mandate. The user needs to see the information about usage per page.
- “Need to order extended rights clearance”: The excerpt was not added because there is a need for extended rights clearance. This is a manual process and has a fee. The user is presented with the fee and needs to decide if they want to order the extended rights clearance.

For adding an excerpt, the endpoint `/api/curriculums/{curriculumId}/excerpts` is used.

PUT
`/api/curriculums/{curriculumId}/excerpts` Add an excerpt
🔒

Parameters

In addition to `feidId`, we need a `curriculumId` sent in path and an object containing necessary fields to add an excerpt sent as a request body:

curriculumId * required
integer(\$int32)
(path)

Curriculum ID

curriculumId - Curriculum ID

Excerpt fields:

```
{
  "isbn": "9783662480274",
  "Pages": "20-50"
}
```

Responses

Error code	Result
201	The excerpt was successfully added. If mandates have been used, the fee of usage (price per page, per semester, per student) will be shown in addition to excerpt(s) information.. Returns an excerpt details: <pre> { "message": "Kopinor HE Licence", "price": "1,426 NOK", "excerpts": [{ "excerptId": 0, "publicationId": "8eca33ab-8ad3-498c-bb5a-f2d94e7b0c6f", "curriculumId": 0, "publication": 0, "pageFrom": 20, "pageTo": 50, "index": 1 }] }</pre>
400	Invalid request, due to: - NEGATIVE_CURRICULUM_ID: curriculumId must be an integer and larger than 0. - REQUEST_BODY_INFO_MISSING_OR_INCORRECT: Missing content or element(s) in the request. <pre> { "message": "Id must be an integer and larger than 0.", "error": "NEGATIVE_CURRICULUM_ID" }</pre>
401	Unauthorized <pre> The client must authenticate itself to get the requested response</pre>
404	Resource not found: Curriculum not found. <pre> { "message": "Curriculum not found.", "error": "CURRICULUM_NOT_FOUND" }</pre>

422	Not possible to add/make a change to excerpt(s) in a curriculum. The request was well-formed but was unable to be followed due to: - CURRICULUM_NOT_UNDER_CONSTRUCTION: Status is not 'UNDER CONSTRUCTION'. - OVERLAPPING_PAGE_RANGE: Range overlapping. - INVALID_ISBN: Invalid ISBN. - ISBN_NOT_FOUND: No content for this ISBN. - INVALID_PAGE_RANGE: Invalid page ranges for an excerpt. - USAGE_ABOVE_LICENSE: Rights clearance required. <pre>{ "message": "Range overlapping.", "error": "OVERLAPPING_PAGE_RANGE" }</pre>
500	Internal server error <pre>The server has encountered a situation it doesn't know how to handle</pre>
503	Service Unavailable <pre>The server is unavailable or down for maintenance</pre>

List of all excerpts in a curriculum

For getting the list of all excerpts in a curriculum, the endpoint `/api/curriculums/{curriculumId}/excerpts` is used. That will return the list of excerpts added to the curriculum if exists. Otherwise, a message about what went wrong.

GET	<code>/api/curriculums/{curriculumId}/excerpts</code> List of all excerpts in a curriculum.	🔒
-----	---	---

Parameters

In addition to feidId, we need a curriculumId sent in path:

curriculumId * required
integer(\$int32)
(path)

Curriculum ID

Responses

Error code	Result
200	The excerpt(s) in a curriculum. <pre>[{ "excerptId": 0, "publicationId": "8eca33ab-8ad3-498c-bb5a-f2d94e7b0c6f", "curriculumId": 0, "publication": 0, "pageFrom": 20, "pageTo": 50, "index": 1 }]</pre>
204	No Content. No excerpt(s) was added to this curriculum <pre>There is no content to return</pre>
400	Invalid request, due to: <pre>- curriculumId must be an integer and larger than 0. - excerptId must be an integer and larger than 0. - Missing content</pre>
401	Unauthorized <pre>The client must authenticate itself to get the requested response</pre>
404	Resource not found: <pre>Due to : - Curriculum not found. - Excerpt not found</pre>
500	Internal server error <pre>The server has encountered a situation it doesn't know how to handle</pre>
503	Service Unavailable <pre>The server is unavailable or down for maintenance</pre>

Update an excerpt

Changing the page ranges or the order for excerpts.

- If the new percentage exceed both HE license and the existing mandates, then there is a need for ordering extended rights clearance, and no changes will be affected, otherwise, the page range will be updated.
- Only allowed to change page range and index
- Changing order for excerpts by sending the excerpt detail with the new index.

For updating a curriculum the endpoint

“/api/curriculums/{curriculumId}/excerpts/{excerptId}” is used .

POST /api/curriculums/{curriculumId}/excerpts/{excerptId} Change page ranges or the order for excerpt. 

Parameters

In addition to feideld, we need a curriculumId and excerptId sent in path and an object containing necessary fields to update an excerpt sent as a request body:

Curriculum Id:

curriculumId * required Curriculum ID
integer(\$int32)
(path)
curriculumId - Curriculum ID

Excerpt Id:

excerptId * required Excerpt ID
integer(\$int32)
(path)
excerptId - Excerpt ID

Excerpt details:

```
{
  "excerptId": 0,
  "publicationId": "8eca33ab-8ad3-498c-bb5a-f2d94e7b0c6f",
  "curriculumId": 0,
  "publication": 0,
  "pageFrom": 20,
  "pageTo": 50,
  "index": 1
}
```

Responses

Error code	Result
200	Excerpt was successful updated with the new page range/order. If mandates have been used, the fee of usage (price per page, per semester, per student) will be shown in addition to excerpt information. Returns excerpt details: <pre>{ "message": "Kopinor HE Licence", "price": "1,426 NOK", "excerpts": [{ "excerptId": 0, "publicationId": "8eca33ab-8ad3-498c-bb5a-f2d94e7b0c6f", "curriculumId": 0, "publication": 0, "pageFrom": 20, "pageTo": 50, "index": 1 }] }</pre>
400	Invalid request due to: - NEGATIVE_CURRICULUM_ID: curriculumId must be an integer and larger than 0. - NEGATIVE_EXCERPT_ID: excerptId must be an integer and larger than 0. - REQUEST_BODY_INFO_MISSING_OR_INCORRECT: Missing content or element(s) in the request. <pre>{ "message": "Id must be an integer and larger than 0.", "error": "NEGATIVE_EXCERPT_ID" }</pre>
401	Unauthorized <pre>The client must authenticate itself to get the requested response</pre>
404	Resource not found: - Curriculum not found. - Excerpt not found. <pre>{ "message": "excerpt not found.", "error": "EXCERPT_NOT_FOUND" }</pre>

422	Not possible to add/make a change to excerpt(s) in a curriculum. The request was well-formed but was unable to be followed due to: - CURRICULUM_NOT_UNDER_CONSTRUCTION: Status is not 'UNDER CONSTRUCTION'. - OVERLAPPING_PAGE_RANGE: Range overlapping. - INVALID_ISBN: Invalid ISBN. - ISBN_NOT_FOUND: No content for this ISBN. - INVALID_PAGE_RANGE: Invalid page ranges for an excerpt. - USAGE_ABOVE_LICENSE: Rights clearance required. <pre>{ "message": "Range overlapping.", "error": "OVERLAPPING_PAGE_RANGE" }</pre>
500	Internal server error <pre>The server has encountered a situation it doesn't know how to handle</pre>
503	Service Unavailable <pre>The server is unavailable or down for maintenance</pre>

Remove an excerpt

Delete an existing excerpt. This action is only available when curriculum has order status 'UNDER_CONSTRUCTION'. To remove an excerpt the endpoint “/api/curriculums/{curriculumId}/excerpts/{excerptId}” is used.

DELETE
/api/curriculums/{curriculumId}/excerpts/{excerptId}
Delete an excerpt from curriculum.

Parameters

In addition to feidId, we need a curriculumId and excerptId sent in path :

Curriculum Id:

curriculumId
* required

Curriculum ID

integer(\$int32)

(path)

Excerpt Id:

excerptId * required

integer(\$int32)

(path)

Excerpt ID

excerptId - Excerpt ID

Responses

Error code	Result
200	Excerpt was successful deleted.Returns excerpt details: <pre>[{ "excerptId": 0, "publicationId": "8eca33ab-8ad3-498c-bb5a-f2d94e7b0c6f", "curriculumId": 0, "publication": 0, "pageFrom": 20, "pageTo": 50, "index": 1 }]</pre>
400	Invalid request du to: <pre>- curriculumId must be an integer and larger than 0. - excerptId must be an integer and larger than 0. - Missing content</pre>
401	Unauthorized <pre>The client must authenticate itself to get the requested response</pre>
404	Resource not found: <pre>Due to : - Curriculum not found. - Excerpt not found</pre>
500	Internal server error <pre>The server has encountered a situation it doesn't know how to handle</pre>
503	Service Unavailable <pre>The server is unavailable or down for maintenance</pre>

Ordering

Ordering a curriculum is available when curriculum has order status 'UNDER_CONSTRUCTION'. That changes the curriculum status to ordered and update the orderdate to the day's date. To order a curriculum the endpoint `"/api/curriculum-orders/{curriculumId}"` is used.

POST `/api/curriculum-orders/{curriculumId}` Order a curriculum 

Parameters

In addition to feideld, we need a curriculumID sent in path:

curriculumId * required Curriculum ID
integer(\$int32)
(path)

curriculumId - Curriculum ID

Responses

Error code	Result
200	The curriculum was successfully ordered. <pre>The content can be downloaded from Bolk.</pre>
400	Invalid request, due to: <pre>- curriculumId must be an integer and larger than 0. - excerptId must be an integer and larger than 0. - Missing content</pre> <pre>{ "message": "Id must be an integer and larger than 0.", "error": "NEGATIVE_CURRICULUM_ID" }</pre>
401	Unauthorized <pre>The client must authenticate itself to get the requested response</pre>
404	Resource not found:

	<pre> Due to : - Curriculum not found. - Excerpt not found { "message": "Curriculum not found.", "error": "CURRICULUM_NOT_FOUND" } </pre>
422	The request was well-formed but was unable to be followed <pre> Order right clearance for curriculum. Some elements/excerpts need to be approved by an admin. </pre>
500	Internal server error <pre> The server has encountered a situation it doesn't know how to handle </pre>
503	Service Unavailable <pre> The server is unavailable or down for maintenance </pre>

Rights clearance

Order extended rights clearance

- When the usage of an excerpt extends both HE-license or existing mandates and extended rights clearance is needed.
- This has a fee and needs to be ordered by the user.
- If the excerpt already exists in Bolk please provide the excerpt-id to ensure that the correct excerpt is updated and that the extended clearance is ordered for that excerpt. If the excerpt-id is not provided the excerpt will be added to the curriculum.

For ordering extended rights clearance, the endpoint “/api/curriculums/{curriculumId}/excerpts/clearance” is used.

PUT	/api/curriculums/{curriculumId}/excerpts/clearance	Order extended rights clearance	
-----	--	---------------------------------	---

Parameters

In addition to feideld, we need a curriculumID sent in path and an object containing necessary fields to add or update an excerpt sent as a request body:

curriculumId * required

integer(\$int32)

(path)

Curriculum ID

curriculumId - Curriculum ID

Excerpt fields: excerptId can be null if the excerpt is not an element of the curriculum.

```
{
  "excerptId": null,
  "isbn": "9783662480274",
  "Pages": "20-50"
}
```

Responses

Error code	Result
200	The rights clearance was successfully ordered. Check status to know when it can go through. Returns orderId and excerpt details: <pre>{ "status": "Ordered", "orderId": 12345, "excerpt": { "excerptId": 0, "publicationId": "8eca33ab-8ad3-498c-bb5a-f2d94e7b0c6f", "curriculumId": 0, "publication": 0, "pageFrom": 20, "pageTo": 50, "index": 1 } }</pre>
400	Invalid request, due to: - Negative_Curriculum_Id: curriculumId must be an integer and larger than 0. - Request_Body_Info_Missing_or_Incorrect: Missing content or element(s) in the request. <pre>{ "message": "Id must be an integer and larger than 0.", "error": "NEGATIVE_CURRICULUM_ID" }</pre>
401	Unauthorized

	The client must authenticate itself to get the requested response
404	Resource not found: Curriculum not found. <pre>{ "message": "Curriculum not found.", "error": "CURRICULUM_NOT_FOUND" }</pre>
422	Not possible to add/make a change to excerpt(s) in a curriculum. The request was well-formed but was unable to be followed due to: - CURRICULUM_NOT_UNDER_CONSTRUCTION: Status is not 'UNDER CONSTRUCTION'. - OVERLAPPING_PAGE_RANGE: Range overlapping. - INVALID_ISBN: Invalid ISBN. - ISBN_NOT_FOUND: No content for this ISBN. - INVALID_PAGE_RANGE: Invalid page ranges for an excerpt. - USAGE_ABOVE_LICENSE: Rights clearance required. <pre>{ "message": "Range overlapping.", "error": "OVERLAPPING_PAGE_RANGE" }</pre>
500	Internal server error The server has encountered a situation it doesn't know how to handle
503	Service Unavailable The server is unavailable or down for maintenance

Status for ordered rights clearance

Check status for ordered extended rights clearance to know when it can go through. To get status the endpoint `"/api/curriculums/{curriculumId}/excerpts/{excerptId}/orderStatus/{orderId}"` is used. If the order is approved, the response will include the price of usage and the fee of ordering rights clearance.

GET	<code>/api/curriculums/{curriculumId}/excerpts/{excerptId}/orderStatus/{orderId}</code>	Check status for ordered extended rights clearance	
-----	---	--	---

Parameters

In addition to feideld, we need a curriculumId, excerptId and orderId sent in path :

Curriculum Id:

curriculumId * required	Curriculum ID
integer(\$int32)	
(path)	curriculumId - Curriculum ID

Excerpt Id:

excerptId * required	Excerpt ID
integer(\$int32)	
(path)	excerptId - Excerpt ID

Order Id:

orderId * required	Order ID
integer(\$int32)	
(path)	orderId - Order ID

Responses

Error code	Result
200	The status of ordered rights clearance: <pre>{ "status": "Done", "price": "1,426 NOK", "rightsClearnceFee": "112,14 NOK", "excerpt": { "excerptId": 0, "publicationId": "8eca33ab-8ad3-498c-bb5a-f2d94e7b0c6f", "curriculumId": 0, "publication": 0, "pageFrom": 20, "pageTo": 50, "index": 1 } }</pre>
400	Invalid request du to:

	• <code>NEGATIVE_CURRICULUM_ID</code>: curriculumId must be an integer and larger than 0. • <code>NEGATIVE_EXCERPT_ID</code>: excerptId must be an integer and larger than 0. • <code>NEGATIVE_ORDER_ID</code>: orderId must be an integer and larger than 0. • <code>REQUEST_BODY_INFO_MISSING_OR_INCORRECT</code>: Missing content or element(s) in the request. <pre>{ "message": "Id must be an integer and larger than 0.", "error": "NEGATIVE_ORDER_ID" }</pre>
401	Unauthorized <pre>The client must authenticate itself to get the requested response</pre>
404	Resource not found: • Curriculum not found. • Excerpt not found. • Order not found. <pre>{ "message": "Order not found.", "error": "ORDER_NOT_FOUND" }</pre>
500	Internal server error <pre>The server has encountered a situation it doesn't know how to handle</pre>
503	Service Unavailable <pre>The server is unavailable or down for maintenance</pre>

Schemas

The schemas used for this API:

Schemas		▼
Curriculum	>	↑
ExcerptDetails	>	↑
ExcerptInfo	>	↑
OrderedRC	>	↑
RCorderStatus	>	↑

Curriculum

Curriculum: includes all the necessary fields in a curriculum.

```

Curriculum ▾ {
  curriculumId      integer($int32)
                    Curriculum Id

  courseCode        string
                    example: SAMF-105
                    Name of the course

  semester          string($year-V/H)
                    example: 2020-H
                    The semester course takes place

  semesterStart     string($date-time)
                    example: 2020-08-30
                    The date the semester starts

  semesterEnd       string($date-time)
                    example: 2020-11-30
                    The date the semester ends

  createDate        string($date-time)
                    The date this curriculum have been created

  orderDate         string($date-time)
                    The date this curriculum have been ordered

  courseName        string
                    Course name

  creatorId         integer($int32)
                    Id for the creator. Curriculum owner

  lecturer          string
                    Lecturers name

  orderStatus       string
                    The curriculum has several order-statuses:
                    • DELIVERED -> The curriculum-files has been created and can be retrieved
                    • IN_ORDER -> The curriculum har been ordered
                    • UNDER_CONSTRUCTION -> The curriculum is under construction and can be edited
                    • WAITING_APPROVAL -> The curriculum is awaiting approval before ordering
                    • APPROVED -> The curriculum has been approved for ordering
                    • CANCELED -> The curriculum has been canceled

                    Enum:
                    ▾ [ DELIVERED, IN_ORDER, UNDER_CONSTRUCTION, WAITING_APPROVAL, APPROVED, CANCELED ]
}

```

ExcerptDetails

Excerpt Details: this detail is returned for each excerpt created

ExcerptDetails ▾ {

```
excerptId      integer($int32)
publicationId   string
                example: 8eca33ab-8ad3-498c-bb5a-f2d94e7b0c6f
                Publication id is a combination of letters and digits

curriculumId   integer($int32)
                The curriculum this element belongs to

publication     integer($int32)
                The publication this excerpt is from

pageFrom       integer($int32)
                example: 20
                Start getting the content from this page

pageTo         integer($int32)
                example: 50
                Getting content until this page

index          integer($int32)
                example: 1
                The index of the excerpt in the curriculum. Must be a postivite number

}
```

ExcerptInfo

Excerpt Information: when an excerpt is created, and mandates is used the user gets information about price and mandate used in addition to excerpt details.

```

ExcerptInfo ▾ {
  message      string
               example: Kopinor HE Licence
               A message indicating which licence have been used.

  price        string
               example: 1,426 NOK
               price per page per student per term

  excerpts     ▾ [
               Excerpt(s) details

               ExcerptDetails ▾ {
                 excerptId    integer($int32)
                 publicationId string
                               example: 8eca33ab-8ad3-498c-bb5a-f2d94e7b0c6f
                               Publication id is a combination of letters and digits

                 curriculumId integer($int32)
                               The curriculum this element belongs to

                 publication   integer($int32)
                               The publication this excerpt is from

                 pageFrom     integer($int32)
                               example: 20
                               Start getting the content from this page

                 pageTo       integer($int32)
                               example: 50
                               Getting content until this page

                 index        integer($int32)
                               example: 1
                               The index of the excerpt in the curriculum. Must be a postivite number
               }
             ]

}

```

OrderedRC

Ordered Rights Clearance: when rights clearance is ordered, we return this information to the user including status, order ID and excerpt details.

```

OrderedRC v {
  status      string
              example: Ordered

  orderID     integer($int32)
              example: 12345

              orderId for the ordered rights clearance

  excerpt     ExcerptDetails v {
                excerptId      integer($int32)
                publicationId   string
                              example: 8eca33ab-8ad3-498c-bb5a-f2d94e7b0c6f
                              Publication id is a combination of letters and digits

                curriculumId   integer($int32)
                              The curriculum this element belongs to

                publication     integer($int32)
                              The publication this excerpt is from

                pageFrom       integer($int32)
                              example: 20
                              Start getting the content from this page

                pageTo         integer($int32)
                              example: 50
                              Getting content until this page

                index          integer($int32)
                              example: 1
                              The index of the excerpt in the curriculum. Must be a postivite number

              }
}

```

RCorderStatus

Rights Clearance Status: when checking rights clearance order, the status, price of usage, rights clearance fee and excerpt details is returned to the user.

```

ROrderStatus {
  status string
    • In Progress -> The extended rights clearance is not done and is still in progress. No further action required.
    • Done -> The extended rights clearance is done and there is a response from the right-holders/CCC. It can be "Accepted" or "Rejected", If the response has a price per page, the price can either be accepted or rejected. For acceptance there is no need to update Bolk because the price is already added to the excerpt in the curriculum. If the price is rejected the excerpt needs to be updated or deleted.

    Enum:
      > Array [ 2 ]
  price string
    example: 1,426 NOK
    price per page per student per term when the order is approved.

  rightsClearanceFee string
    example: 112,14 NOK
    The fee for ordering rights clearance

  excerpt ExcerptDetails {
    excerptId integer($int32)
    publicationId string
      example: 8eca33ab-8ad3-498c-bb5a-f2d94e7b0c6f
      Publication id is a combination of letters and digits

    curriculumId integer($int32)
      The curriculum this element belongs to

    publication integer($int32)
      The publication this excerpt is from

    pageFrom integer($int32)
      example: 20
      Start getting the content from this page

    pageTo integer($int32)
      example: 50
      Getting content until this page

    index integer($int32)
      example: 1
      The index of the excerpt in the curriculum. Must be a postivite number

  }
}

```

DD130 - BULK API CONTENT DELIVERY

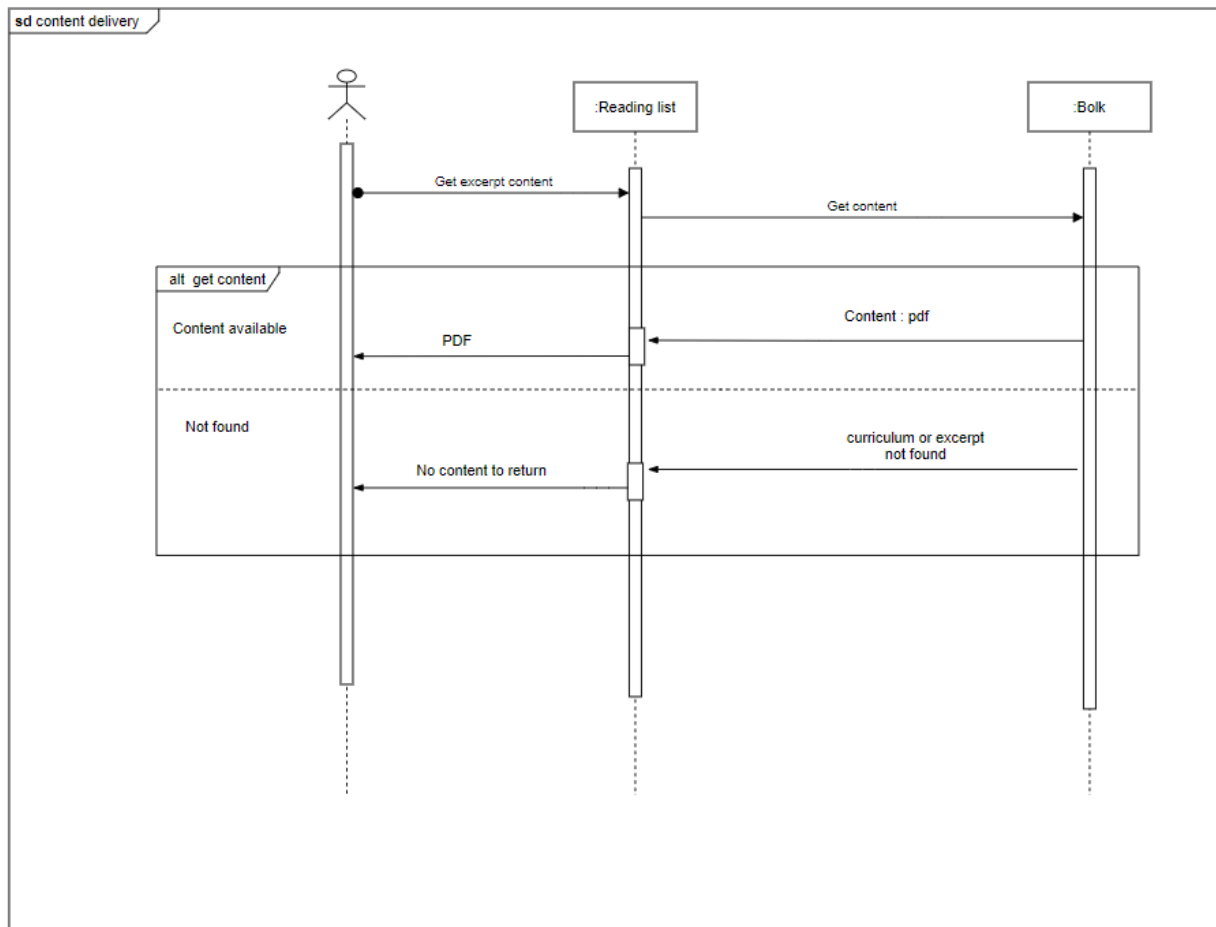
Version: 1.0
Status: 03 - Completed
Author: Siham Sidali Fekiani

netcompany

Introduction

This document is a continuation for the Bulk API documentation. It will just contain a sequence diagram for delivering content for one excerpt and how the request flows.

Overview



Content delivery

User stories

As... (role)	...I shall... (what)	...so that... (why)
user	download the content when the order has been confirmed	I can distribute it to my students

Swagger API

When performing calls to the API, both API-key, FeideID and a valid OAuth token from Dataporten which is the way to authorize access to Bolk, must be provided as headers. Only requests gone through Dateporten (GateKeeper) will be forwarded to Bolk.

Available authorizations

curriculum_oauth (http, Bearer) ←

OAuth-token from Dataporten

Value:

Authorize

Close

curriculum_api (apiKey) ←

The API-key from Kopinor

Name: X-API-KEY

In: header

Value:

Authorize

Close

And the feideID should be added as parameter in the header. The request will be sent on behalf of the user having this feideID, which again we use to identify the Tiril user in Bolk.

X-FEIDE-ID * required

string

(header)

Feide ID

X-FEIDE-ID - Feide ID

Check point 4 in the Bolk API documentation for more details.

Get content delivery endpoint

For getting the content use “api/curriculums/{curriculumId}/excerpts/{excerptId}/content”. That downloads a pdf with the excerpt content. The curriculum needs to be delivered to download the content.

GET

/api/curriculums/{curriculumId}/excerpts/{excerptId}/content

Get the excerpt content as pdf

▼

🔒

parameters

In addition to feideld, we need a curriculumId and excerptId sent in path:

Curriculum Id:

curriculumId * required
integer(\$int32)
(path)

Curriculum ID

curriculumId - Curriculum ID

Excerpt Id:

excerptId * required
integer(\$int32)
(path)

Excerpt ID

excerptId - Excerpt ID

Responses

Error code	Result				
200	The content is successfully downloaded <table><tr><th>Code</th><th>Details</th></tr><tr><td>200</td><td> Response body Download file</td></tr></table>	Code	Details	200	Response body Download file
Code	Details				
200	Response body Download file				
400	Invalid request du to: - NEGATIVE_CURRICULUM_ID: curriculumId must be an integer and larger than 0. - NEGATIVE_EXCERPT_ID: excerptId must be an integer and larger than 0. <pre>{ "message": "Id must be an integer and larger than 0.", "errorCode": "NEGATIVE_EXCERPT_ID" }</pre>				
401	Unauthorized <pre>{ "message": "The user is not authorized.", "errorCode": "UNAUTHORIZED_USER" }</pre>				

404	Resource not found: • Curriculum not found. • Excerpt not found. • File not found <pre>{ "message": "The file for excerpt 223177 was not found!", "errorCode": "EXCERPT_FILE_NOT_FOUND" }</pre>
422	The curriculum is not delivered: <pre>{ "message": "The curriculum 223177 is not delivered!", "errorCode": "CURRICULUM_NOT_DELEVERED" }</pre>
500	Internal server error

DD130 - BULK API - RETRIEVE STAGE3 URL

Version: 1.0
Status: 02 - Under development
Author: Siham Sidali Fekiani

netcompany

Introduction

This document is an extension of the Bulk API documentation. It details the request flow for accessing the Get Stage 3 URL endpoint.

Retrieve Stage3 URL

Swagger API

When making API calls, the request headers must include an API key, FeideID, and a valid OAuth token from Dataporten for authorization. Only requests authenticated through Dataporten (GateKeeper) will be forwarded to Bolk.

Available authorizations ✕

curriculum_oauth (http, Bearer) ←

OAuth-token from Dataporten

Value:

Authorize **Close**

curriculum_api (apiKey) ←

The API-key from Kopinor

Name: X-API-KEY

In: header

Value:

Authorize **Close**

The FeideID should be included as a parameter in the request header. The request will be processed on behalf of the user associated with this FeideID, which is used to identify the Tiril user in Bolk.

X-FEIDE-ID * required

Feide ID

string

(header)

Check point 4 in the Bolk API documentation for more details.

Get to stage3 endpoint

To access Stage 3 for a given curriculum, use the endpoint “api/curriculums/{curriculumId}/stage3”. This endpoint retrieves the URL that can be used to open Stage 3 in a browser. The user will then need to enter their Bolk username and

password to access the page. If the curriculum is not owned by the entered user, an error message will be displayed.

GET

/api/curriculums/{curriculumId}/stage3

Go to stage3.

Parameters

In addition to feideld, the curriculumId must be included in the path parameters.

Curriculum Id:

curriculumId

* required

integer(\$int32)

(path)

Curriculum ID

curriculumId - Curriculum ID

Responses

Error code	Result												
302	Ready for redirect. <table><tr><th>Code</th><th>Description</th><th>Links</th></tr><tr><td>302</td><td>Ready for redirect. Headers:</td><td>No links</td></tr><tr><th>Name</th><th>Description</th><th>Type</th></tr><tr><td>Location</td><td>URL to the Stage 3 page</td><td>string</td></tr></table>	Code	Description	Links	302	Ready for redirect. Headers:	No links	Name	Description	Type	Location	URL to the Stage 3 page	string
Code	Description	Links											
302	Ready for redirect. Headers:	No links											
Name	Description	Type											
Location	URL to the Stage 3 page	string											
400	Invalid request du to: NEGATIVE_CURRICULUM_ID: curriculumId must be an integer and larger than 0. <pre>{ "message": "Id must be an integer and larger than 0.", "errorCode": "NEGATIVE_CURRICULUM_ID"}</pre>												
401	Unauthorized												

	<pre>{ "message": "The user is not authorized.", "errorCode": "UNAUTHORIZED_USER" }</pre>
404	Resource not found: Curriculum not found. <pre>{ "message": "Curriculum not found.", "errorCode": "CURRICULUM_NOT_FOUND" }</pre>
422	The curriculum has multiple orders: <pre>{ "message": "Curriculum with id = XXX has multiple order. Not treated for this version!", "errorCode": "TOO_MANY_CURRICULUM_ORDERS" }</pre>
500	Internal server error <pre>{ "message": "error message from excerption message.", "errorCode": "UNKNOWN_ERROR" }</pre>

Appendix D – User stories

This appendix incorporates several user stories to complement the specifications outlined in the requirements list. While the requirements target the identification of key components on a more detailed and specific scale, the user stories provide a broader perspective. They aim to show workflows and user interactions, interlinking various elements of the requirements. Even though the user stories included in this document are not exhaustive, they are intended to demonstrate practical examples of system usage.

We want these user stories to help bridge the gap between technical specifications and real-world application. Most of these stories are written from the perspective of end users, to make sure that we don't lose track of the most important part of any solution - the end users and how they can effectively interact with the system.

The appendix is divided into categories to better illustrate which key areas are being described. However, it is important to note that many of these stories naturally overlap, as these areas are all part of the reading list system:

- User interface & communication
- BOLK, digitization and copyright
- Students
- Instructors
- Librarians
- Administrators

User interface & communication

U01 – Inline Citation Editing for Instructors

As an instructor, when I have opened a reading list and found a citation I'd like to edit, I want to edit and save my changes where I am, without being navigated away, so that I can update reference information quickly and efficiently without interrupting my workflow.

U02 – Active Lists Foregrounded

As a list creator, I want only active or in-progress lists to be displayed in the default working view, so that the workspace is kept clean and focused. Inactive lists and courses should be hidden in the GUI, but remain findable in the system and accessible online.

U03 – Consistent Alphabetical Sorting

As an instructor, I want to automatically sort references within each section of my reading list alphabetically, so that students can reliably find each reference in the expected place, regardless of the contributor's role in the publication (e.g. author, editor, chapter author).

The sort order treats contributor names consistently regardless of their role in the publication. An editor, a chapter author, and a book author are all sorted by their surname in the same way, so that Eriksen (editor) appears before Hylland (chapter author) and Moe (book author) as expected.

U04 – Messaging Between List Collaborators

As a list creator, I want to send and receive direct messages and change notifications with other list creators — including library staff, instructors, and co-instructors — in a timely and role-relevant way, so that I can collaborate effectively on reading lists, respond quickly when needed, and refer back to the full communication history within the system at any time.

U05 – Actionable Library Messaging

As a librarian, I want to send information to instructors and students directly from the library system, and receive their input and requests as actionable prompts on my task list that link directly to the relevant location in the system, so that I can communicate

efficiently with all parties and resolve issues such as digitization requests, purchase requests, and broken links without unnecessary back-and-forth or context switching.

U06 – Configurable, Role-Based Notification Management

As a system administrator, I want to configure which notifications are sent to users, when they are sent, and whether they are triggered automatically or manually by designated roles, so that users receive timely and relevant information without being overwhelmed by unnecessary or excessive notifications.

U07 – Controlled Notifications with Optional Comments for List Changes

As a list creator, I want to control whether changes I make to a reading list trigger notifications to students or other list creators, with the option to add a comment explaining the change, so that relevant parties are kept informed about updates in a timely and targeted way, without generating unnecessary noise for minor or routine changes.

When I make a specific change to a reading list (e.g. adding or removing a citation, editing or deleting a note), a popup or inline prompt appears asking whether I want to send a notification about the change. Within the notification prompt, I can add a comment to provide context or explanation for the change (e.g. "Replaced with a more recent edition" or "This chapter is no longer required").

BOLK, digitization and copyright

U08 – Copyright Clearance via Bolk Integration

As an instructor, I want to make several book chapters available for my students. I create Bolk requests directly from the relevant citations in Leganto, without having to create them again from scratch in Bolk.

The first chapter is under 15% of the entire book, so the request automatically gets the status "Approved". In the next citation, I register two different page ranges. The total is above 15%, so I get the message that an action is required. I click on "Extend copyright clearance".

A few days later, I get the message that my request was approved. (Note: I also get notified if a request is declined.) I can now complete the process for the entire list and click on "View and complete in Bolk". I am taken to the correct page in Bolk where I can review the excerpts I have ordered and preview the files. I click on "Order compendium"

in Bolk and a few minutes later, “View/download PDF” links appear under my two citations: one file for the first one, and one file per chapter for the second one.

U09 – Librarian Review of Bolk Copyright Requests

As a librarian, I need to approve or decline copyright clearance requests I receive in Bolk. Once I take an action on these requests, the instructor is informed/can get the information directly in the system without having to involve the library any further.

U10 – Digitization Requests from the Reading List

As an instructor, I want to submit a digitization request directly from the reading list, so that the library receives all necessary information to upload the correct file without needing to contact me.

U11 – Complete Metadata in Digitization Requests

As a librarian, I want to receive digitization requests from instructors with complete metadata (course name, course code, title, ISBN, page numbers, and comments), so that I can efficiently process the request and upload the correct file without additional follow-up.

U12 – Complete Metadata in Copyright Clearance Requests for Bolk Processing

As a librarian, I want to receive copyright clearance requests from instructors with complete metadata (course name, course code, title, ISBN, page numbers, and comments), so that I can process the request in Bolk and upload the correct, cleared file without needing additional information.

U13 – Digitization Request Overview

As a librarian, I want digitization requests to be clearly organized—either within the reading list or in a dedicated request area—so that they are easy to locate and manage.

U14 – Direct Copyright Clearance Requests from the Reading List

As an instructor, I want to submit a copyright clearance request directly from the reading list, so that the library can clear the correct excerpt and I can share legally approved materials with my students.

U15 – Copyright Clearance Request Overview

As a librarian, I want copyright clearance requests to be clearly organized—either within the reading list or in a dedicated request area—so that they are easy to locate and manage.

Students

U16 – Reading List Access for Prospective Students

As a prospective student, I am curious about the curriculum of the program I have applied for, and I go to the program plan page on the website. There I get access to the reading list from the reading list system. I do not get access to the full text, but I can still see what is available in the library.

U17 – LMS-Integrated Reading Lists with Library Availability

As an active student, I log in to the LMS and find the reading lists for the current semester so that I get an overview of the required course literature. Here I can easily see what is available in the library. I can reserve physical material if it is on loan and get access to digital and electronic content that is available.

U18 – Personal Reading List History with Progress Tracking

As an active student, I want an overview of all the reading lists I have, and have had, at the institution. Furthermore, I want to be able to tick off in the reading list what I have read, so that I have the best possible overview of what I have gone through and what I have not.

U19 – Reading List Access for Former Students

As a former student, I want to find the reading list for a course I studied five years ago, so that I can access materials from my studies for reference, further learning, or professional purposes.

Instructors

U20 – LMS Access to Course Reading Lists for Instructors

As an instructor, I want to access the existing reading list for my course directly from the LMS or be guided on how to create or request one if none exists, so that I can efficiently continue working on the reading list regardless of who created it.

U21 – Word Document Import for Reading List Draft

As an instructor, I want to import a draft reading list I have prepared in Word into the system, so that I can avoid duplicate work and save time by not having to manually re-enter all the information.

U22 – Import of References from External Sources

As an instructor, I want to add references from external sources such as digital bookshops or websites directly into the system, so that I can avoid manually entering bibliographic details and reducing the risk of errors.

U23 – New Edition Requests to the Library from the Reading List

As an instructor, I want to easily notify the library that I wish to update a reference to a newer edition once it is published, so that the reading list is kept up to date and the library can acquire the new edition for its collection.

U24 – Automatic New Edition Alerts for Reading List Titles

As an instructor, I want to be automatically alerted when a new edition of a title on my reading list is registered in the library catalogue, so that I can decide whether to update the reference without having to monitor new publications myself.

U25 – Shared Reading List Linked Across Multiple Courses

As an instructor, I want to have a single shared reading list that is linked to multiple courses and displayed in each of their respective LMS course rooms, so that I can maintain one list in one place without having to duplicate or manually synchronize content across courses.

Librarians

U26 – Complete Reading List Processing Within the System

As a library staff member responsible for processing reading lists, I want to be able to perform all actions related to processing, finalizing, and publishing reading lists from within the reading list system, so that I can complete my work entirely within one system without switching between tools.

U27 – Reading List-Driven Book Acquisition

As a librarian, I want to make book acquisition decisions based on the contents of reading lists in a rational and efficient way, so that course materials are available for students and the library collection is kept up to date without unnecessary time and resources being spent.

U28 – Unified Display of All Available Formats

As a librarian, I want to ensure that both electronic and physical versions of the same publication are visible under a single reference in the reading list, and that all available formats are clearly indicated, so that end users can easily orient themselves with what is available through the library, regardless of their accessibility needs.

U29 – Overview of Assigned Reading Lists and Workflow Status

As a librarian, I want to easily see which reading lists I am responsible for and what status they have in the workflow, so that I have a clear overview of what requires my attention and can prioritize my work accordingly.

U30 – Predefined Public and Internal Tags for Categorization and Workflow Tracking

As a librarian/list manager, I want to define and manage tags for use in reading lists, allowing instructors and course coordinators to easily select from predefined tags (public or internal). This enables citations to be categorized with labels such as "Mandatory Reading" or "Recommended Reading," helping students prioritize their reading and focus on the most essential materials.

Additionally, as a librarian, I want to use internal tags to track the status and progression of citations through different stages of the workflow. These internal tags should only be visible to librarians, ensuring coordination without causing confusion for instructors or students.

Administrators

U31 – Bulk Course Import and Reading List Rollover for New Semesters

As a system administrator, I want to bulk import courses from FS for a given semester and roll over reading lists from previous semesters by linking copies to the newly imported courses, so that I can efficiently prepare the system for an upcoming semester without having to recreate or manually reassign reading lists from scratch.

After the import, I am presented with a clear summary showing which courses received a rolled-over list, which could not be automatically matched, and which were imported without a linked list. Edge cases such as courses that have been discontinued, merged, or renamed should be handled gracefully.

Once a rolled-over list has been linked to a course, the relevant course responsables are notified and prompted to review and update the list for the upcoming semester.

U32 – Configurable Rollover

As a system administrator, I want to control which information is carried over from old reading lists during rollover, and be able to append a suffix to the list title (e.g. year and semester), so that it is clear which lists are originals and which are rolled-over copies, and that only relevant information is transferred to the new lists.

Appendix E – Online Transaction Types Definitions and expected response time

"Transaction Time" is defined as the time a transaction takes place (in milliseconds) from when it is received by the system, until the system processes and returns a response, measured on the server side.

The transactions measured are specifically end-user transactions (from the moment a user triggers a request until it is generated and sent back to the browser).

Expected response time (SLA/performance targets) for online transactions in the reading list system:

- 50% of all online transactions are performed in less than one second.
- 75% of all online transactions are performed in less than two seconds.
- 95% of all online transactions are performed in less than two and a half seconds.

Transaction Type	Definition
Search	<i>The transaction type is characterized by the user entering a search criterion and producing a list of result records.</i> <i>In some cases, no explicit search criteria is entered by the user, but the system will provide an implicit search criteria (e.g. display a list of reading lists or citations).</i> <i>The transaction starts when the user hits the appropriate control to execute the search and completes when the result list is displayed on the screen.</i>
Display	<i>The transaction type is characterized by the user selecting an item from a result set and displaying the full detail of that selected record.</i> <i>Other 'Display' transactions may be initiated by the user selecting controls from a screen that cause a full detail page to be displayed.</i> <i>The transaction starts when the user selects the item to be displayed and completes when the full record has been displayed on the screen.</i> <i>For analysis of response times, we separate Display transactions into two categories depending on the volume of data being displayed in a record:</i>

	• Short record display transactions – displaying records where the volume of data in the record is less than 0.5KB • Long record display transactions – displaying records where the volume of data in the record is greater than 0.5KB.
Update	The transaction type is characterized by the user requesting that some data changes that they have created be saved to the database. The transaction starts when the user selects to save the changes they have made and is complete when application control has returned to the user following executing the transaction. It should be noted that, to achieve very high-throughput and responsiveness transactions will be committed to the database asynchronously, meaning that control will return to the user extremely quickly potentially before the actual transaction has been committed. For analysis of response times, as for Display Transactions we separate Update Transactions into two categories depending on the volume of data being updated in a record: • Short record update transactions – the volume of data being updated is less than 0.5KB • Long record update transactions – the volume of data being updated is greater than 0.5KB.

Example use cases

Use Case 1: Searching for a reading list

1. The user enters search criteria for a reading list (e.g. course code, keyword, or title) and executes a search. (**Search transaction**)
2. From the resulting list the user selects the appropriate reading list and opens it. (**Short display transaction**)

Use Case 2: Accessing full text / Full text view

The user clicks the full text link on a citation, opening the full text material (e.g. PDF or ePub) within the reading list system, or opening the external source in a new browser window. (**Short display transaction**)

Use Case 3: Adding a citation to a reading list

1. The user opens the relevant reading list and selects the functionality for adding a citation via search. **(Short display transaction)**
 2. The user searches for a specific title within the system. **(Search transaction)**
 3. From the results list the user selects the desired title and the system displays the full bibliographic details. **(Short display transaction)**
 4. The user chooses to add the item to the reading list and customizes the citation as needed. **(Short record update transaction)**
 5. The user saves the citation in the reading list. **(Short record update transaction)**
-

Use Case 4: Editing a citation

1. The user selects the relevant citation and opens it in full view. **(Short display transaction)**
 2. The user activates edit mode. **(Short display transaction)**
 3. The user makes the necessary changes to the citation (e.g. adding a tag or updating a note) and saves. **(Short record update transaction)**
-

Use Case 5: Editing a reading list

1. The user opens the relevant reading list and selects the functionality for editing the list. **(Short display transaction)**
 2. The user edits the reading list (e.g. updates the title, description, or publication status) and saves the changes. **(Short record update transaction)**
-

Use Case 6: Changing system configurations

1. The user logs in to the reading list system and opens the configuration interface. **(Short display transaction)**
 2. The user navigates to a specific system configuration and makes the necessary change (e.g. toggling a feature on or off, updating a label) and saves. **(Short record update transaction)**
-

Use Case 7: Generating statistics reports

1. The user logs in to the system and navigates to the statistics module.
(Short display transaction)
 2. The user selects the relevant reporting parameters and executes a search.
(Search transaction)
 3. The system retrieves and displays an overview of the available usage data.
(Long Record Display Transaction)
 4. The user refines the report by applying additional filters and the system updates the displayed data accordingly. **(Long Record Display Transaction)**
 5. The user exports the report in the desired format (e.g. CSV or PDF) for further analysis or reporting purposes. **(Long Record Display Transaction)**
-

Appendix F: FS import file format

Each institution in the reading list consortium uses a version of the Norwegian student information system FS.

There is no direct integration between FS and the reading list system, so we use a program created by Sikt that retrieves data for a specific institution based on an input file placed on our servers.

The input file contains, among other things, the semester and year that the institution wants to retrieve, as well as where in FS the data should be retrieved from (different institutions register their data in different ways).

Here are the parameters contained in the input file and their possible values:

Parameter	Possible values	Explanation
Year	Ex: 2026	
Term	Spring, Autumn, entire year ('STÅR')	'STÅR 2026' will retrieve data for the entire university year starting with the autumn semester (Autum 2026 + Spring 2027).
Operation	IMPORT, ROLLOVER, DELETE	Import: courses are created. Rollover: courses are created and lists from the previous year are copied and associated to the new courses. Delete: used to delete courses in the import file.
<i>include_ua</i>	Yes/no	What level of data is retrieved from FS
Title format	1 or 2	1) Course name - course code term year. Scientific methods - BI224F Spring 2026 2) Course code - Course name (term year). Ex: ABC123 - Introduction to Social Sciences (Spring 2026)
Department	Yes/no	Yes: department code. No: faculty code
Feide domain	ex: @ntnu.no	The domain is added to instructors' IDs present in the file.

<i>brukere_fra_nivaa</i>	Courses, UE/UA, both	Refers to where in FS the data is retrieved from.
Campus and number of participants	[Checkbox] / [file name]	If enabled, campus and/or number of participants will be retrieved for each course. The data can be provided via a dedicated file uploaded alongside the input file, or estimated based on previous years in FS.
Language order	Ex: nb,nn,en, sme	nb = Bokmål, nn = Nynorsk, en = English, sme = Sami
Role codes IMS codes	Ex: TEACHER or * '2, 3, 7'	'*' will retrieve all the person names registered on each course. Specific codes (ex: TEACHER) can also be retrieved instead of everything. IMS is just another type of code that can be used instead.
Start/end date modifier	Ex: -1	The start and end dates are not retrieved from FS. A default date is added to the file, but institutions can adjust them. Ex: By writing -1, the semester start date will be June 1st instead of July 1st.

Based on this input file, a job runs nightly to retrieve and compile data into an import file. The resulting .csv contains information about each course: its code, name and academic department; the term, year and start/end dates; additional IDs the course should be searchable by; instructors responsible for the course; and, where available, campus location and number of participants.

Before each semester, Sikt uses this file to import the relevant courses for the upcoming semester into the reading list system for each institution. The reading lists are then connected to the relevant course, either manually or as part of the rollover process. This also enables the LTI integration between the learning management system (LMS) in use at the institution and the reading list system, allowing students to access the reading list associated with a course directly within the corresponding course in the LMS.