

Vigilo OneRoster implementation

Table of content:

Required:	1
Restrictions (might change):	1
Endpoints	2
Additional Metadata	5
Implementation guide	7

Required:

TenantId

IP of caller (for ip-filter pr Tenant).

Use header for authentication: Key: <tenantId>, value: <token given by Vigilo>

Restrictions (might change):

- Due to performance, some of the most resource-heavy resources have been disabled, and will return “403 Forbidden”. In that case, use /schools/<schoolId>/xxx endpoint instead.
- dateLastModified will always be todays date at midnight
- Some SourceIds contains a composite key. They will contain ‘\’ to separate ids and it will be urlEncoded.
- Filters will be added by request/need.

We HIGHLY recommend using the resources pr. organization. (/schools/) instead of root as most of the data makes more sense in context of an organization. Most of the data is organization specific and are not to be used across several organizations.

Example of pagination: /students?limit=50&offset=250

Example of filter: school/<schoolId>/students?filter=status='active'

Example of both; school/<schoolId>/students?filter=status='active'&limit=50&offset=250

Endpoints

Base url: https://dataporten-api.vigilo.no:4004/api/<tenantId>ims/oneroster/v1p1

Action	URI	Supported filter *
getAllOrgs	/orgs	
getOrg	/orgs/<orgId>	
getAllSchools	/schools	
getSchool	/schools/<schoolId>	
getAllAcademicSessions	/academicSessions (restricted due to heavy load)	type: schoolYear term status: active tobedeleted
getAcademicSessionsForSchool	/schools/<schoolId>/academicSessions	type: schoolYear term status: active tobedeleted
getAcademicSession	/academicSessions/<academicSessionId> Alt: /schools/<schoolId>/academicSessions/<academicSessionId>	
getAllUsers	/users	Identifier role: parent

getUsersForSchool	/schools/<schoolId>/users	Identifier status: active tobedeleted
getUser	/users/<userId> Alt: /schools/<schoolId>/users/userId	
getAllStudents	/students	identifier
getStudentsForSchool	/schools/<schoolId>/students	identifier status: active tobedeleted
getStudentsForClassInSchool	/schools/{school_id}/classes/{class_id}/students	Identifier status: active tobedeleted
getStudent	/students/<userId> Alt: /schools/<schoolId>/students/<userId>	
getAllTeachers	/teachers	identifier
getTeachersForSchool	/schools/<schoolId>/teachers	identifier
getTeachersForClassInSchool	/schools/{school_id}/classes/{class_id}/teachers	identifier
getTeacher	/teachers/<userId> Alt: /schools/<schoolId>/teachers/<userId>	
getAllClasses	/classes (restricted due to heavy load)	schoolYearId: <schoolYearId> classType: homeroom scheduled - OR - groupType: Vigilo-internal value which is presented in metadata tag
getClassesForSchool	/schools/<schoolId>/classes/	status: N/A active tobedeleted schoolYearId: <schoolYearId> classType: homeroom scheduled - OR - groupType: Vigilo-internal value which is presented in metadata tag

getClassesForStudent	/school/<schoolId>/student/<studentId>/classes	schoolYearId: <schoolYearId> classType: homeroom scheduled - OR - groupType: Vigilo-internal value which is presented in metadata tag
getClass	/classes/<classId> Alt: /schools/<schoolId>/classes/<classId>	
getAllCourses	/courses (restricted due to heavy load) In development	
getCoursesForSchool	/schools/<schoolId>/courses/ In development	
getCourse	In development	
getAllEnrollments	/enrollments (restricted due to heavy load unless schoolYearId is specified; /enrollments)	schoolYearId: <schoolYearId> status: active tobedeleted role: teacher student
getEnrollmentsForSchool	/schools/<schoolId>/enrollments	schoolYearId: <schoolYearId> status: active tobedeleted role: teacher student
getEnrollmentsForClassInSchool	/schools/{school_id}/classes/{class_id}/enrollments	status: active tobedeleted role: teacher student
getEnrollment	/enrollments/<enrollmentId> /schools/<schoolId>/enrollments/<enrollmentId>	
getTerms	/terms	
getTermsForSchool	/schools/<schoolId>/terms	
getTerm	/terms/<termId> Alt: /schools/<schoolId>/terms/<termId>	

Additional Metadata

The “metadata”-tag of the payloads consist of additional data which can be useful. It’s basically a key-value map which does not have any specific parameters according to the standard but are populated on-demand with data which can be useful for the consumer of the API. The table below lists some of the values currently available. Please note - when there’s an empty value, the key might also not come.

Payload	Metadata
Org	<p>academicSessions: Link to AcademicSessions for this org. classes: Link to Classes for this org. students: Link to Students (user-objects) for this org. teachers: Link to Teachers (user-objects) for this org. enrollments: Link to Enrollments for this org. courses: Link to Courses for this org. externalOrganizationNumber: OrganizationUnit - externalOrganizationNumber addressline1: Organization addressline1 addressline2: Organization addressline2 postalCode: Organization postalCode city: Organization city latitude: Organization latitude longitude: Organization longitude phoneNumber: Organization phoneNumber email: Organization email homePageUrl: Organization homePageUrl gsild: Organization gsild managerFirstName: Organization managerFirstName managerLastName: Organization managerLastName vigold: Organization vigold nationalRegisterId: Organization nationalSchoolRegisterId shortName: Organization pasCode key: Only for tenants: Tenant “key”.</p>

<p>User</p> <p>Note: 1 person might be several users with the same sourceId, but with different roles and orgs. (student/teacher/parent). 1 user might also come as several students or teachers - one pr org where they belong. E.g: 1 user is a parent + 50% teacher at two schools + taking a course at a third school = 1 parent + 2 teachers + 1 student occurrence.</p>	<p>formOfWrittenNorwegian: nn nb. gender: male female. birthDate: Date of birth (format: yyyy-MM-dd). fromDate: (student) fromDate (Not currently in use!) toDate: (student) toDate (Not currently in use!) Only returned when getting in the context of an organization: inactiveDate: (student) inactiveDate positions: (teacher) Array of positions for the teacher. Contains the following data: positionName: (teacher) positionName positionDescription: (teacher) positionName status: (teacher) employee position status. endDate: (teacher) positionEndDate</p> <p>DEPRECATED: positionName: (teacher) positionName (if several positions; each name divided by character ' ') positionDescription: (teacher) positionName (if several positions; each name divided by character ' ') endDate: (teacher) positionEndDate (if several positions; each name divided by character ' ') status: (teacher) employee position status. (if several positions; each name divided by character ' ')</p> <p>userIds (not in metadata-tag, but documented here as it's dynamic): adUserIdentifier: ActiveDirectory username. feideUserIdentifier: Feide username. upnUserIdentifier: UPN username.</p>
<p>AcademicSession</p>	<p>status: status of the schoolYear directly from Vigilo. "Nåværende Utgått Pre_registered" org: Link to the Organization of which this academicSession belongs.</p>
<p>Class</p>	<p>schoolYear: Link to the academicSession (type: schoolYear) this class belongs to. pasPeriod: If this instance is a PAS-group, this will display PAS Period. pasSubjectCode: If this instance is a PAS-group, this will display PAS SubjectCode. groupType: GroupType from Vigilo. Possible values to date: (Class, LearningSupport, SubjectGroup, SnoGroup, GnoGroup, SfoGroup, SfaGroup, TpoGroup, NativeLanguage, TofGroup, OtherUsers, PasTestGroup) pasTransferRequestedByPersonId: If this instance is a PAS-group, this will display the user which transferred the group to PAS.</p>

Enrollment	<p>schoolYear: Link to the academicSession (type: schoolYear) this enrollment belongs to.</p> <p>inactiveDate: (student) inactiveDate</p> <p>positions: (teacher) Array of positions for the teacher. Contains the following data:</p> <ul style="list-style-type: none"> positionName: (teacher) positionName positionDescription: (teacher) positionName status: (teacher) employee position status. endDate: (teacher) positionEndDate <p>DEPRECATED:</p> <p>positionName: (teacher) positionName (if several positions; each name divided by character ' ')</p> <p>positionDescription: (teacher) positionName (if several positions; each name divided by character ' ')</p> <p>endDate: (teacher) positionEndDate (if several positions; each name divided by character ' ')</p> <p>status: (teacher) position status. (if several positions; each name divided by character ' ')</p>
Course	N/A - Under development.

Implementation guide

1. Get parents: /users?filter=role='parent'
 - a. Get schools: /schools/

For each school:

 1. Get students /schools/<orgId>/students
 2. Get teachers /schools/<orgId>/teachers
 3. Get present active school year: /schools/<orgId>/academicSessions?filter=type='schoolYear' AND status='active'
 4. Use 'sourceld' from active school year (unique pr. school) to get classes:
/schools/<orgId>/classes?filter=schoolYearId='<schoolYearId (sourceld) in academicSessions>'
 5. Use sourceld from active school year to get enrollments:
/schools/<orgId>/enrollments?filter=schoolYearId='<schoolYearId (sourceld) in academicSessions>'
 6. Use enrollments to filter which teachers and students are active in the different classes.
 7. Find active employees which are not enrolled (e.g admin staff), by finding the teachers (step 2) which has metadata status "active" and are not in the list of teachers found in 6.