ED-FI RFC 3 - ASSESSMENT API (Inactive)

Ed-Fi Request for Comment: 3
Product: Ed-Fi Data Standard v2.0
Affects: Ed-Fi API Design & Implementation Guidelines (and Ed-Fi ODS / API)
Obsoletes: --
Obsoleted By: --
Status: Inactive
Ed-Fi Alliance
June 7, 2016

Synopsis

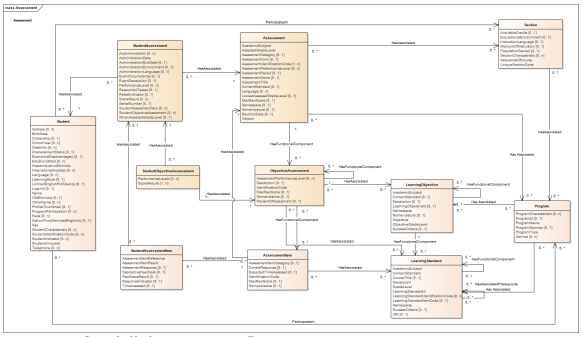
The RFC describes a REST API for systems to transfer assessment metadata and assessment results from one system to another. The API is intended to provide options useful for a Web-based architecture in which a client system uses HTTP GETs as the mechanism to implement an on-demand data exchange that allows a target system to pull data from a source system that hosts the API surface. This approach supports the transfer of detailed resource collections using only a few API calls. The API design goals are further supported through resource modeling that provides aggregates of related data entities, as well as field selection capabilities. The API specification includes resources that support discovery as well as data transfer. The resource models for the data (in JSON) are designed to be compatible with the Ed-Fi Unifying Data Model v2.0.

This RFC is complemented by ED-FI RFC 2 - ASSESSMENT DATA COLLECTION API (Inactive), which provides REST APIs intended for use in a "push" model of data exchange relying on HTTP POSTs and PUTs, in which the client system calls APIs on a target host system to initiate the data exchange.

Discussion

The Assessment API is a read-only API designed to allow clients efficiently to discover and access assessment metadata and assessment results. Access can be limited to small sets of data (e.g., results for a particular quiz for a particular class section) or can encompass transfer of large datasets between systems (e.g., all student results for a particular district-wide benchmark exam).

The resource models are consistent with the Ed-Fi Unifying Data Model v2.0 (UDM). Consistent with the structure and intent of the Ed-Fi UDM, the API focuses on the transfer of assessment results: the assessment resource itself is designed to provide sufficient metadata to interpret assessment results in typical operational contexts in the K–12 education enterprise (e.g., in classrooms or school principal offices), but not necessarily to move the assessment instrument itself from one system to another. The API is not intended to support portability of assessments themselves, as there are other existing technology standards that provide such support.



Ed-Fi UDM 2.0 Assessment Domain *(click to expand)*

The API has two distinct parts:

1. A "discovery" interface, that allows for a client to determine what assessments are available in the system, and to access assessment metadata
2. A "data access" interface, that allow for transfer of student assessment results

These two functions correspond to the `assessment` and `studentAssessment` API resources, respectively. The `assessment` and `studentAssessment` resources include many (mostly optional) sub-resources, such as `sections`, `programs`, `contentStandards`, and so forth. These resources are associated with assessments or student assessment results. All resources defined by the API specification are consistent with the Ed-Fi UDM 2.0.

Discovery Endpoint: assessment Resource

The purpose of the discovery interface is to allow for discovery of assessments in the source system. There are supports for search by natural key and an alternate, Resource ID (see below for more on Ed-Fi keys). The API supports a set of queries that return `assessment` resources for which there are corresponding `studentAssessments` resources. The API supports search by properties of the entities `localEducationAgency`, `school` and `administrationDate` of `studentAssessment`.

Some examples follow.

Discovery Examples

Discovery Example 1. Return all available assessments in the source system, limiting returned data to the fields provided

```
GET /assessments?fields=id,assessmentTitle,revisionDate,version,
academicSubjectDescriptor
```

Sample return:

```
200 {"Content-Type":"application/json; charset=utf-8"}
[
    {
        "revisionDate": "2012-10-23T00:00:00",
        "version": 5221,
        "assessmentTitle": "2nd Grade ELA 2nd Six Weeks 2012-2013",
        "id": "55b544f6-ef9b-4eb4-804d-306b8e4abacf",
        "academicSubjectDescriptor": "English Language Arts"
    },
    {
        "revisionDate": "2012-09-26T00:00:00",
        "version": 4874,
        "assessmentTitle": "2nd Grade Math 1st Six Weeks 2012-2013",
        "id": "0d434811-4458-4b0f-9bd7-a0d0719e1ecf",
        "academicSubjectDescriptor": "Mathematics"
    }
]
```

Other example "Discovery" requests:

Discovery Example 2. Return an `assessment` resource using its natural key.

```
GET /assessments?
assessmentTitle=ACT&academicSubject=English&assessedGradeLevel=Twelfth%
20grade&version=1
```

Discovery Example 3. Return all `assessment` resources for an associated `school` resource. Note that assessments have no direct associations with education organizations, so the query is asking for all `assessments` for which there is one or more `studentAssessment` results for any students associated with (i.e., enrolled in) that school.

```
GET /schools/CEAA310E-2967-4CE2-9A15-8AD4F2D487A7/assessments
```

Discovery Example 4. Return `assessment` resources for which there are `studentAssessment` results within a particular time period.

```
 GET /assessments?studentAssessment.administrationDate=[2015-01-01..2016-
01-01]
```

Data Access Endpoints: studentAssessment Resource

The general pattern for these endpoints is to name a group and an assessment for which data is being collected. Each of these endpoints returns a `studentAssessment` resource.

Data Access Examples

Data Access Example 1. Return all `studentAssessments` for a school. Note that requiring the `assessment_id` on the querystring seems to go against standard REST practice of putting IDs in the path. However, this syntax reflects that assessments are only associated with users and groups through enrollment associations (as modeled by the Ed-Fi UDM).

```
GET /schools/CEAA310E-2967-4CE2-9A15-8AD4F2D487A7/studentAssessments?
        assessment_id=4343CE89-60AF-436A-967B-E49CCD2CAE01
        &fields=student,administrationDate,scoreResults,studentUniqueId
```

Sample return:

```
200 {"Content-Type":"application/json; charset=utf-8"}
[
    {
        "administrationDate": "2012-10-29T00:00:00",
        "scoreResults": [
            {
                "resultDatatypeType": "Integer",
                "result": "4",
                "assessmentReportingMethodType": "Raw score"
            }
        ],
        "student": [
            {
                "firstName": "Sonya",
                "lastSurname": "Gross",
                "birthDate": "1997-03-11T00:00:00"
            }
       ]
    }, // etc.
]
```

Other example "Data Access" requests:

Data Access Example 2. Return all `studentAssessments` associated with a `student` resource

```
GET /students/{id}/studentAssessments?assessment_id=55B544F6-EF9B-4EB4-
804D-306B8E4ABACF
```

Data Access Example 3. Return all `studentAssessments` for an `assessment` resource, either using natural key

```
GET /sections/1d3eeead9223442a98f7061e8c5f5330/studentAssessments?
        assessmentTitle=ACT&academicSubject=English
        &assessedGradeLevel=Twelfth%20grade&version=1
```

Data Access Example 4. Return all studentAssessments for a particular assessment and school for a particular time period

```
GET /schools/255901001/studentAssessments?assessment_id=55B544F6-EF9B-
4EB4-804D-306B8E4ABACF
        &administrationDate=[2015-06-01..2015-06-30]
```

Resource Keys

Each resource can be referenced by two keys, a natural key and Resource ID. The Resource ID appears as the top level 'id' field in the resource model, and is a string. The Resource ID provides for compatibility with standard REST design where a simple resource identifier (URI) is the core mechanism for identifying individual resources. The natural key serves to define uniqueness through business values, often through a key composed of multiple fields. The uniqueness provided by the natural key assists in several ways, including supporting data quality through key unification, as key fields are shared across entities, and through providing robust lookup values.

The natural keys for each resource are as follows:

Resource	Natural Keys
`assessment`	• `title` • `assessedGradeLevelDescriptor` • `academicSubjectDescriptor` • `version`
`studentAssessment`	• `title` • `assessedGradeLevelDescriptor` • `academicSubjectDescriptor` • `version` • `studentUniqueId` • `administrationDate`

As is evident in this structure, the studentAssessment key includes the key for the assessment resource. This pattern is common in data implementations based on the Ed-Fi UDM.

Specification Content

Implementations of the Assessment Data Collection API must follow the REST API Design & Implementation Guidelines v2.0. The key terms under the "Requirement Levels" section from that document carryover to this document.

API Specification

Implementations *must* expose the all resources and operations (HTTP verbs), with all the configurations of required parameters and return values, as indicated in the Open API specification attached to this document.

A visual display of that specification is also attached. Please note that the visual specification is only for ease of reference and the Open API specification (in JSON) is authoritative if there are differences.

Note: HTTP response codes usage follows requirements in the REST API Design & Implementation Guidelines v2.0 under the section "Response Codes." Because Open API files requires at least one response code to validate, the specification attached usually contains only one response for each API path. To eliminate duplication, the response codes were removed from the Open API spec even though they are required for conformance

Field Selection

Field selection is *required* in the API specification to minimize the overall size of the returned entities. It is especially important when querying metadata to see if there are new assessments (e.g., when syncing two systems) or when there are many `studentAssessments`, but only a subset of data is needed.

Selection works by appending the fields required in the URI querystring using the fields parameter.

- Values for the fields parameter are comma-separated

- If the `fields` parameter is missing, all data is returned. If included, only included fields are returned.
- References must use a '.' to create a path to elements deeper in the model hierarchy
- If a value in the fields parameter references a JSON object or collection, all fields are returned by default. To return particular fields, fields must be listed in parentheses '()'. For clarity, an asterisk '*' can be used within parentheses as a wildcard to return all fields for that resource or collection of resources.

Field Selection Examples

Field Selection Example 1. Returns only the elements `assessmentTitle` and `version` for the `assessment` resource

```
/assessment/assessments?fields=assessmentTitle,version
```

Field Selection Example 2. Returns two fields (`assessmentTitle` and `version`) and a collection of `objectiveAssessment` resources with only the `description` field provided

```
 /assessment/assessments?fields=assessmentTitle,version,
objectiveAssessments(description)
```

Field Selection Example 3. Returns two fields and a collection of `objectiveAssessment` objects with all fields provided

```
 /assessment/assessments?fields=assessmentTitle,version,
objectiveAssessments(*)
```

Pagination

Pagination is implemented by appending two parameters to the querystring: `limit` and `offset`.

- `limit` specifies how many resources to return
- `offset` specifies the starting position for the collection of resource results to return, often using in supporting pagination for clients

The API does not require ordering, but does require that identical queries return resources in the same order, unless there have been changes to the underlying data between subsequent calls.

Pagination Example

Pagination Example 1. Returns 25 items located at positions 51-75 in the collection

```
 ?limit=25&offset=50
```

Additional Information and Requirements

- Implementations *must* ensure that the Resource IDs provided are unique across resources of that type within the implementing (source) system. It is *recommended* that the Resource IDs returned be a GUID or other guaranteed unique value so as to reduce possible issues with lookups and to improve security.
- All item definitions *must* follow the definitions according to the Ed-Fi UDM Element listed. Those definitions can be found in the Data Handbook for the Core XSD v2.0
    - If the element has a parent element, the cardinality of the child element is to be understood as applicable if and only if the parent element is included in the resource model.
- Dates must conform to a YYYY:MM:DD format, consistent with ISO 8601 format: http://tools.ietf.org/html/rfc3339.

Additional Examples

This section contains additional illustrative examples of the API specification.

Additional Examples

Additional Example 1. Return all `assessment` resources for a school for which there are `studentAssessments` between the dates provided. Return only the fields `assessmentTitle` and `id`.

```
GET /schools/CEAA310E-2967-4CE2-9A15-8AD4F2D487A7/assessments?
        fields=assessmentTitle,id&studentAssessment.administrationDate=
[2015-06-01..2015-06-30]
```

Sample return:

```
 200 {"Content-Type":"application/json; charset=utf-8"}

[
        {
             "assessmentTitle": "ACT",
             "id": "87d1634a-2d3c-4086-a1ca-beb397d97c0a"
        },
        {
            "assessmentTitle": "AP - English Literature and Composition",
            "id": "81e7cc24-4d82-443b-9953-cf07bd48d54d"
        },
        {
            "assessmentTitle": "AP - Government and Politics:
Comparative",
            "id": "6826cc87-0e2f-4125-b28d-c5723ded6024"
        },
        {
            "assessmentTitle": "English I 2nd six weeks 2012-2013",
            "id": "4343ce89-60af-436a-967b-e49ccd2cae01"
        }
    ]
```

Additional Example 2. Return all `studentAssessment` resources for an `assessment` resource by assessment id, between dates provided. Return only `studentUniqueId`, `administrationDate`, `scoreResults`, and `accommodations`.

```
GET /localEducationAgencies/E7242CA4-BB65-453D-92E0-DC1887CDB1ED/
                studentAssessments?fields=studentUniqueId,
administrationDate,scoreResults,accomodations
                &q=AdministrationDate:[2009-03-07..2016-03-10]
                &assessment_id=D5BC38D3-B2D4-4E5B-B0BA-1B32FDE6C44F
```

Sample return:

```
200 {"Content-Type":"application/json; charset=utf-8"}

[
       {
            "administrationDate": "2010-12-01T00:00:00",
            "studentUniqueId": "605750",
            "scoreResults": [
                {
                    "resultDatatypeType": "Integer",
                    "result": "16",
                    "assessmentReportingMethodType": "Scale score"
                }
            ],
            "accommodations": [],
        },
        {
            "administrationDate": "2010-12-01T00:00:00",
            "studentUniqueId": "607778",
            "scoreResults": [
                {
                    "resultDatatypeType": "Integer",
                    "result": "14",
                    "assessmentReportingMethodType": "Scale score"
                }
            ],
            "accommodations": [],
       },
       \\ etc...
    ]
```