Ed-Fi® REST API Design Guidelines

Ed-Fi Version 1.2

11/7/2013

Contents

Introduction

This document defines a set of design guidelines for an Ed-Fi® REST application programming interface (API). This document presents the conventions and design patterns to guide an Ed-Fi REST API specification, along with rationale for the guidelines.

As design guidelines, the content herein does not describe a specific implementation or particular API specification. Rather, the guidelines describe the properties to which an API specification and related implementation must adhere in order to be considered conformant to the Ed-Fi standard. Reference implementations and technical materials distributed as part of the Ed-Fi solution follow these guidelines.

Audience

This document is designed to be a guide for developers of an Ed-Fi REST API who would create a specification for their API implementation.

As a secondary audience, application developers may find this document useful when building client applications that interact with an Ed-Fi REST API.

Prerequisites

To gain the most benefit from this information, you should already be familiar with the following knowledge areas:

- OAuth2 (http://tools.ietf.org/pdf/draft-ietf-oauth-v2-12.pdf)
- Representational State Transfer (REST) services (http://en.wikipedia.org/wiki/Representational_state_transfer)
- JavaScript Object Notation (JSON) (http://www.json.org)

Scope

Information Scope

The Ed-Fi Unifying Data Model (UDM)[1] provides the basis for the resources transferred and manipulated by an Ed-Fi REST API implementation. The Ed-Fi UDM is a structured, conceptual model of common K–12 education data. The model includes entities that are easily recognized by people in the education field: schools, students, teachers, attendance, grades, assessment results and many others. These entities contain attributes (aka properties) that are also easily recognized. For example, assessment results contain data such as a score and a date the assessment was administered. The UDM also includes associations (aka relationships) between entities, such as the association between students and schools, and the association between student grades and sections.

REST interfaces are built around resources that define nouns. In the education domain, these nouns include schools, students, teachers, etc., all of which have been rigorously defined as entities, attributes and relationships in the Ed-Fi UDM. An Ed-Fi REST API defines resources as compositions of entities, attributes and associations, called domain aggregates, identified from the Ed-Fi UDM according to the principles of Domain-Driven Design (DDD)[2]. This concept is discussed in more detail later in this document.

An Ed-Fi REST API may cover a subset of the full Ed-Fi data model that is exposed and exchanged in a particular specification or implementation. The

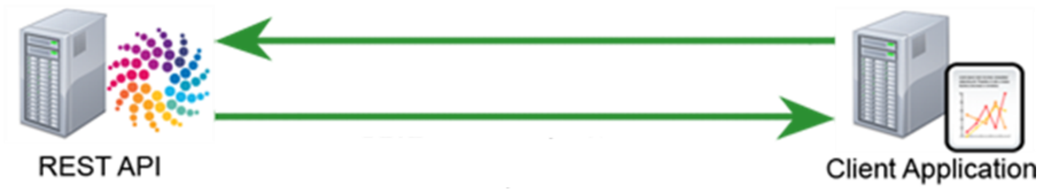
[1] *http://www.ed-fi.org/tech-docs/.*
[2] Evans, Eric et-al (2006), *Domain-Driven Design Quickly*, C4Media Inc., http://www.infoq.com/minibooks/domain-driven-design-quickly.

API need not be implemented for the entire scope of the Ed-Fi data model in order to be aligned.

Technical Scope

The REST (Representational State Transfer)[3] architectural style is a convention-based approach to define application programming interfaces (APIs). HTTP (Hypertext Transfer Protocol), using the HTTP operations (GET, PUT, POST, DELETE, etc.), is used as the application protocol.

REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of representations of resources. As depicted below, a server-based application or data store implements, exposes, or is wrapped with an Ed-Fi REST API to allow client applications to meaningfully exchange and manipulate education data.



APIs are typically a "contract" between providers of data and consumer applications, with the underlying platform and application choices treated as a black box[4]. An Ed-Fi REST API follows this pattern. An Ed-Fi REST API levies no requirements on how data is internally stored or how it is used

[3] Fielding, Roy Thomas (2000), *Architectural Styles and the Design of Network-based Software Architectures*, Doctoral dissertation, University of California, Irvine, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.
[4] http://en.wikipedia.org/wiki/Black_box

by client applications, only that the API provide a technical “contract” between a provider of data and its consumer applications, externally representing the exchanged data resources, aligned with the Ed-Fi data model as specified in this document. The same exact resource may be represented to different clients in different ways. For example, a resource might be represented as HTML to a web browser and JSON to an application. The key idea is that the representation is a way to interact with the resource, but is not the resource itself.

There may be circumstances where an Ed-Fi REST API would diverge from a “pure” REST approach to support specific use cases; for example, to support bulk load, as discussed later in this document.

Use Cases

An Ed-Fi REST API may be used in a variety of situations where different applications and/or data stores need to consume, exchange or manipulate education data. The following categories of use cases have been identified:

1. A data store implements an Ed-Fi REST API to allow multiple applications to share common education data. Sources of data will use the API to upload data for sharing among/between these client applications. Consumers of data will access and possibly manipulate portions of that data using the API.
2. An application implements an Ed-Fi REST API to allow other cooperating applications to interact in meaningful ways.
3. An Ed-Fi REST API is used to define interfaces to upload or request bulk data transfers. Because of the time to process/generate the bulk data, the interface includes queuing and notification.

Resources

Domain Aggregates as Resources

The resources that are transferred and manipulated by an Ed-Fi REST API are cohesive compositions of entities, attributes, and associations, called domain aggregates, identified from the Ed-Fi UDM according to the principles of Domain-Driven Design (DDD). Use cases and events in the domain typically center on individual domain aggregates.

Domain aggregates are organized along transactional boundaries, where the data contained should "live" and "die" together. For example, the Discipline Incident aggregate contains details about a discipline incident, but also captures data related to the behaviors and students involved. However, since a Discipline Action would not generally be captured at the same time as the Discipline Incident details, each has been separated into its own aggregate.

Each domain aggregate has an aggregate root. An aggregate root is an entity (and in some cases an association) that includes other entities, their attributes and associations. The subordinate artifacts of an aggregate are not directly accessible and can only be referenced through the aggregate root.

Most entities in the UDM are aggregate roots and contain no other entities (e.g., Student, School, Course). In some cases, associations that represent a significant domain concept are represented as aggregate roots (e.g., StudentSchoolAssociation reflects enrollment).

Resource Extensions

An Ed-Fi REST API continues the philosophy of extensibility behind the rest of the Ed-Fi solution. In this context, extensions may modify the structure of existing resources or create entirely new resources. Consumers of an Ed-Fi REST API interact with resource extensions just as they interact with other resources. For example, if the Student resource has been extended, an API consumer requesting a student will receive the extended resource.

The exact method for handling reads and writes of resource extensions is still under consideration and will be specified at a future date.

Resource Keys

Each resource exposed by an Ed-Fi REST API is referenced by an internally-assigned universally unique identifier (UUID). While the specific algorithm for generating these identifiers is not prescribed here, they should be generated using a UUID implementation such as Microsoft's GUID (Globally Unique Identifier) or similar[5].

Due to the Ed-Fi Operational Data Store's use of composite primary keys, all resources are also uniquely identifiable by one or more externally defined primary key values. For example, a Session is uniquely identified by the Session Name, Term and School Year. Resources can be accessed by primary key values using the standard HTTP GET query string search syntax (i.e. *{resourceUri}?keyField1=value1&keyField2=value2*). For POST, PUT, PATCH and DELETE operations, they must be identified using their universally unique identifiers.

[5] While clients are capable of generating unique identifiers, an Ed-Fi REST API should generate the identifiers for its clients, and should not accept client-generated identifiers when inserting or updating data.

HTTP Verbs

HTTP verbs communicate an action that should be taken against a resource. Depending on the verb, the request may include additional needed information in the body. In an Ed-Fi REST API, the following verbs are supported.

POST

An HTTP POST creates an individual subordinate resource. If successful, the URI to the new resource is returned in the “Location” HTTP header of the response.

GET

An HTTP GET returns existing resources.

PUT

An HTTP PUT performs an idempotent update of a resource. Only a full replacement of the existing resource is supported.

DELETE

An HTTP DELETE deletes an existing individual resource.

PATCH

An HTTP PATCH performs a partial update on an existing individual resource. For a partial update, only the properties that are submitted will be updated on the target resource. The entire patch will be applied, or none of it will. The new representation of the entire resource is returned in the response body.

URI Construction

The table below shows an example of how to use the verbs.

Resource	POST	GET	PUT/PATCH	DELETE
`/students`	Add a new Student	Get a collection of Students	Error	Error
`/students/{id}`	Error	Gets an individual Student	Updates an individual Student	Deletes an individual Student
`/students/{id}/studentSchoolAssociations`	Error	Gets a collection of Student School Associations	Error	Error

REST API Conventions and Features

REST Feature	Ed-Fi Implementation	Explanation
Authorization and Authentication	Three-Legged OAuth 2 (for user-based authentication/authorization), and Two-Legged OAuth (for "approved" partner application-based authentication/authorization).	Based on the OAuth 2 spec from January of 2011.
Case Sensitivity	`/users?firstName=JOHN` `/users?FIRSTNAME=John`	URIs, parameter names, and parameter values are not case sensitive. The two URI's to the left will produce the same results.
Encryption	HTTPS	All calls to the API must use SSL.
Hypermedia	See examples in the sections below.	If requested in the HTTP header, hypermedia URIs are provided during the response to a GET to give the caller a set of resources to access that are related to the originally requested resource.

Media types	JSON	JSON is the supported media type for all requests.
Version	`https://api.example.com/`**`v1`**`/users` or … `https://example.com/api/`**`v1`**`/users`	The version number is specified in the base URI.

URIs

For each resource, there are two base forms for the URI, one for a collection of resources and one for a specific resource in the collection. The collection is referred to by the pluralized name of the individual resource. A specific resource is referenced by the collection name, followed by a slash and the resource's unique identifier. For example:

- `/students` refers to a collection of Students
- `/students/ffc0…a272` refers to a specific student with an assigned identifier of "`ffc0…a272`"

A collection of resources associated with another resource may be referenced in a single URI, using the pattern "*associatedResource/{id}/resource*". For example:

- `/schools/abc7…123f/studentSchoolAssociations/students` refers to the collection of Students associated with a specific School

Note that the collection to be referenced is always the rightmost resource.

Ed-Fi Descriptors

Descriptors in the Ed-Fi data standard are a mechanism to support flexible enumerations or code tables. Each Descriptor has the following attributes:

- DescriptorId (primary key)
- URI
- Namespace

- CodeValue
- ShortDescription
- Description
- PriorDescription
- EffectiveBeginDate
- EffectiveEndDate
- a "map" back to an Ed-Fi enumeration value

The GET of a resource will return all of the component values of the descriptor enumerations, thus allowing the application to select which of the components to use.

PUTs of a resource must specify the URI for each descriptor value. The URI is constructed with the namespace followed by the URL-encoded ShortDescription.

Descriptors are also exposed as resources of an Ed-Fi REST API and can be accessed and manipulated as follows:

Resource	POST	GET	PUT	DELETE
`/[Subtype] Descriptors`	Adds a new descriptor	Gets all descriptors for the subtype	Error	Error
`/[Subtype] Descriptors/{id}`	Error	Gets all attributes for an individual descriptor	Updates an individual descriptor	Deletes an individual descriptor

Descriptor References

References to a descriptor value are a URI constructed with the namespace followed by the URL-encoded ShortDescription:

Namespace+"/"+ShortDescription

For example, to refer to the BehaviorDescriptor value in the Ed-Fi namespace with a short description of “School Violation”, the reference would be the following URI:

```
http://www.ed-
fi.org/Descriptor/BehaviorDescriptor.xml/School%20Violation
```

Query Operators

Search

An Ed-Fi REST API supports querying capabilities when searching a collection of resources. Query operators are applied to the query string using the following format: *{collectionURI}?{propertyName}{operator}{value}.* Currently the equals operator is the only operator supported. Support for other operators is a topic for future consideration.

Operator	Description
=	Equality

Examples

To search all available Students having the first name “John” as an exact match a name/value pairing is used as mentioned above:

```
https://api.example.com/v1/students?firstName=john
```

Selectors

Selectors allow application developers to be more selective about how much data is returned in the resource representations. Implementation of selectors in an Ed-Fi REST API is considered optional.

Parameter	Description
`includeFields`	Limits the response to the fields listed.
`excludeFields`	Limits the response to all fields except those listed.

Examples

To just retrieve a Student's first and last names:

```
https://api.example.com/v1/students/{id}?includeFields=firs
tName,lastSurname
```

To retrieve everything but a Student's middle name:

```
https://api.example.com/v1/students/{id}?excludeFields=midd
leName
```

Paging

When multiple records are being returned, the total count of all records is returned as an additional data member of the response body. The limit parameter can be used in the query string to set the maximum number of records returned. If no value is supplied, the limit parameter will default to 20. The offset parameter can be set to specify how many records to skip when getting the result set. The default value for offset is 0.

Example

To get the first name and last name of a collection of available Students from positions 31 to 40:

```
https://api.example.com/v1/students?fields=firstName,lastSu
rname&limit=10&offset=30
```

Views

Views provide the ability for a client to request predefined and/or custom views be returned as a resource. For example, a view could include student information, assessments, attendance records, gradebook entries and transcript information.

HTTP Responses

Response Codes

A core tenet of RESTful APIs is embracing HTTP response codes to communicate status information. An API consumer should be able to inspect the HTTP response code and understand the status of its request. The following response codes are used when responding to requests.

HTTP Response Code	Name	Reason(s)
200	OK	Returned after a successful operation when a response contains a body.
201	Created	Returned after a successful POST. The response from a POST will also include a Location in the Header pointing to the newly added resource. A POST response will not contain a body.

202	Accepted	Returned for a request that has been accepted for processing, but the processing has not yet been completed.
204	No Content	Returned when the server has fulfilled the request but does not return an entity body.
304	Not Modified	Returned when the client includes the “If-None-Match” header containing the requested resource’s last known ETag.
400	Bad Request	Returned if the request is malformed. The body of the response may contain a descriptive error message.
401	Unauthorized	Returned if the access token is invalid. The response will not contain a body.
403	Forbidden	Returned when the server is refusing to fullfill a request in situations such as: • Resource fails data validation (missing required properties, string lengths too long, etc.) • Resource fails referential validation • Resource fails uniqueness validation • HTTP headers required for the operation were not present (eg: Missing If-Match header on PATCH or DELETE)
404	Not Found	Returned if a resource is not found. The response will not contain a body.
412	Precondition Failed	Returned if an If-Match header pre-condition fails.
500	Internal Server Error	Returned if the server encountered an unexpected error during the operation.

Errors

If an error occurs on the server, a 500 (Internal Server Error) code will be returned along with the addition of a message in the body, containing the error details. For example:

```
{
    “code”: 500,
    “type”: “Internal Server Error”,
    “message”: “Unable to communicate with database”
  }
}
```

ETags

An Ed-Fi REST API supports optimistic concurrency and efficient bandwidth handling through the use of ETags (Entity Tag)[6].

ETags for Optimistic Concurrency

Concurrency becomes an issue in higher volume systems with multiple clients accessing the same data. An Ed-Fi REST API supports an “opt-in” optimistic concurrency model[7] using ETags. During PUT and DELETE operations, the API will verify that the resource has not been modified by another party since it was last obtained by the client. If the resource has not changed, the operation will continue normally. If, however, the resource has changed, the client will receive an error as notification that they must obtain the latest version of the resource before attempting further modifications. This approach can be used to prevent “last-in-wins” update scenarios and potential data loss.

When responding to a GET request for an individual resource (e.g. `/students/{id}`) the response header returned by the API must contain an ETag that uniquely identifies the version of the resource.

[6] http://en.wikipedia.org/wiki/HTTP_ETag
[7] http://en.wikipedia.org/wiki/Optimistic_concurrency_control

Example ETag in a GET response:

Response Header

```
ETag: “-8588261538364775808”
Content-Type: application/json; charset=utf-8
Cache-Control: private
Content-Length: 1398
```

Body

```
{
   "schoolId":12345,
   "classPeriodName":"4th Period",
   "classroomIdentificationCode":"abcde",
   "localCourseCode":"Math 101",
   "termTypeId":1,
   "schoolYear":2012,
   "uniqueSectionCode":"3FJ56",
   "sequenceOfCourse":1,
   "availableCredit":1.5
}
```

To opt-in to an optimistic update, the ETag value is added to an “If-Match” header of a subsequent PUT or DELETE request, and the operation will be processed only if the “If-Match” header value matches the latest ETag for the resource stored on the server. If, however, the ETags do not match, a 412 (Precondition failed) response code will be returned. If the “If-Match” header is not specified in the request, then the operation should be processed and the server should respond with a response code of 204 (No Content) if the operation succeeds. However, the API may be implemented to require optimistic concurrency for updates and deletes, and if no “If-Match” request header is supplied by the client, it may respond with a general 400 error status code.

Example header value in a PUT or DELETE request:

```
If-Match: -8588261538364775808
```

The ETag may be generated as a hashed representation of the resource, a version number, a timestamp representing the last modification to the resource, or a unique identifier that is refreshed after each modification to the resource.

ETags for Web Cache Validation

ETags can also be used to reduce bandwidth usage by preventing the contents of an unmodified resource from being returned. An Ed-Fi REST API should support such cache validation through the use of the “If-None-Match” request header. If the ETag value supplied in the request header still matches the existing resource, the API should respond with a 304 (Not Modified) status code with no response body rather than a 200 (OK) with the resource content.

Bulk Load and Transfer

There are cases that require bulk updates of Ed-Fi data. For example, a Student Information System (SIS) vendor may not support transactional updates to the Ed-Fi REST API from within their software, and will instead opt for a bulk transfer of Ed-Fi XML-based data interchanges on a regular interval.

As another example, at the start of a school year, an entire cohort of students will need to be promoted from one grade to the next. While this operation can be completed via a series of individual transactions, it would be less taxing on network traffic and overall execution time to execute a bulk request that would perform the entire batch.

The exact nature of an API for handling bulk submissions is still under consideration and will be specified at a future date.

Authentication

All resources in an Ed-Fi REST API must be protected using the OAuth 2 specification[8] from January 21, 2011. Prior to accessing the API, applications must register with the Ed-Fi REST API service provider and obtain a Client Id and Secret Key. Using the OAuth 2 protocol, an application must then obtain an access token to include in every call to the API.

Two-Legged OAuth

For applications that are fully trusted partners of an Ed-Fi REST API service provider, access tokens can be obtained for the application to interact with the API outside of the context of a particular user. For example, a SIS would have its own security model and would be using the API mostly to submit data in a transactional manner in the course of its users' operations within the SIS. For this scenario, the Ed-Fi REST API service provider can issue an access token without forcing each end user to log in with an external identify provider.

Three-Legged OAuth

Applications that are not fully trusted partners of an Ed-Fi REST API service provider must obtain access tokens that are bound to the context of a particular end user, and an Ed-Fi REST API must enforce user-level access permissions on all resources.

[8]Hammer-Lahav, et al. (2011), *The OAuth 2.0 Authorization Protocol*, IETF Draft, http://tools.ietf.org/pdf/draft-ietf-oauth-v2-12.pdf

While a full discussion of setting up a security infrastructure is beyond the scope of this document, here is the typical flow for a user logging in through Three-Legged OAuth to an application that uses an Ed-Fi REST API.

1. User attempts to access the application.
2. Since the user has not been authenticated, the user is redirected to the Ed-Fi REST API service provider's OAuth endpoint.
3. The Ed-Fi REST API service provider begins the process of authenticating the user using the SAML protocol by redirecting the user to a Security Token Service (STS). For example, this could be a server running Microsoft's Active Directory Federation Services (ADFS).
4. The STS either presents the user with a login page (if it is also the identity provider) or redirects them to the appropriate federated identity provider, such as an ADFS instance hosted by the user's school district.
5. The user logs in and is redirected back to the Ed-Fi REST API service provider's site (with a SAML token issued by the STS), whereby an authorization code is issued that associates the application with the authenticated user.
6. The user is redirected back to the original application with the authorization code.
7. The application makes an out-of-band API call back to the Ed-Fi REST API service provider with the authorization code, and receives an access token, which it captures and saves into the user's session.
8. The application then makes subsequent calls to the Ed-Fi REST API with the access token.

Authorization

Due to privacy concerns and FERPA regulations, it is critical that an Ed-Fi REST API correctly authorizes all resource requests.

Trusted Partner Applications

In a two-legged OAuth scenario with a trusted partner application (such as a SIS), resource requests would be secured by scoping requests to specific education organizations such as states, regional service centers, local education agencies or schools. Within the authorized scope, the applications would be able to access and modify all data.

Third Party Applications

For three-legged OAuth scenarios, each request would be authorized based on claims based security (see below) and Ed-Fi domain data to identify the students for which they have responsibility. For example, a superintendent would be granted access to student data for all students in their district, a principal would be granted access to all students in their school, and teachers would be granted access only to students enrolled in their sections.

Claims-Based Security

The resources in an Ed-Fi REST API are divided into groups, and secured through claims. For example, only system and data administrators would be issued claims allowing the modification of all types and descriptors, while teachers could view the section enrollments of their students, but perhaps only guidance counselors and registrars would be able to make changes.

API Resource Listing

The following table contains the resources exposed by an Ed-Fi REST API:

Resource	Contained Entities
AcademicHonorsType	AcademicHonorsType
AcademicSubjectDescriptor	AcademicSubjectDescriptor
AcademicSubjectType	AcademicSubjectType
AccommodationDescriptor	AccommodationDescriptor
AccommodationType	AccommodationType
Account	Account
	AccountCode
AccountCodeDescriptor	AccountCodeDescriptor
Actual	Actual
AddressType	AddressType
AdministrationEnvironmentType	AdministrationEnvironmentType
AdministrativeFundingControlDescriptor	AdministrativeFundingControlDescriptor
AdministrativeFundingControlType	AdministrativeFundingControlType
Assessment	Assessment
	AssessmentAssessmentFamily
	AssessmentScore
	AssessmentContentStandard
	AssessmentIdentificationCode
	AssessmentPerformanceLevel
	AssessmentSection
AssessmentCategoryType	AssessmentCategoryType
AssessmentFamily	AssessmentFamily
	AssessmentFamilyAssessmentPeriod
	AssessmentFamilyIdentificationCode
	AssessmentFamilyContentStandard
AssessmentIdentificationSystemType	AssessmentIdentificationSystemType
AssessmentItem	AssessmentItem

	AssessmentItemLearningStandard
AssessmentItemResultType	AssessmentItemResultType
AssessmentPeriodDescriptor	AssessmentPeriodDescriptor
AssessmentReportingMethodType	AssessmentReportingMethodType
AttendanceEventCategoryDescriptor	AttendanceEventCategoryDescriptor
AttendanceEventCategoryType	AttendanceEventCategoryType
BehaviorCategoryType	BehaviorCategoryType
BehaviorDescriptor	BehaviorDescriptor
Budget	Budget
CalendarDate	CalendarDate
CalendarEventDescriptor	CalendarEventDescriptor
CalendarEventType	CalendarEventType
CareerPathwayType	CareerPathwayType
CharterStatusType	CharterStatusType
ClassPeriod	ClassPeriod
ClassroomPositionDescriptor	ClassroomPositionDescriptor
ClassroomPositionType	ClassroomPositionType
Cohort	Cohort
	StudentCohortAssociation
	StudentCohortAssociationSection
	StaffCohortAssociation
	CohortProgram
CohortScopeType	CohortScopeType
CohortType	CohortType
CohortYearType	CohortYearType
CompetencyLevelDescriptor	CompetencyLevelDescriptor
ContentClassType	ContentClassType
ContinuationOfServicesReasonDescriptor	ContinuationOfServicesReasonDescriptor
ContinuationOfServicesReasonType	ContinuationOfServicesReasonType
ContractedStaff	ContractedStaff
CostRateType	CostRateType
CountryCodeType	CountryCodeType

Course	Course
	CourseCodeIdentificationCode
	CourseGradesOffered
	CourseLearningObjective
	CourseLearningStandard
	CourseLevelCharacteristic
	CourseCompetencyLevelDescriptor
CourseAttemptResultType	CourseAttemptResultType
CourseCodeSystemType	CourseCodeSystemType
CourseDefinedByType	CourseDefinedByType
CourseGPAApplicabilityType	CourseGPAApplicabilityType
CourseLevelCharacteristicType	CourseLevelCharacteristicType
CourseOffering	CourseOffering
CourseRepeatCodeType	CourseRepeatCodeType
CredentialFieldDescriptor	CredentialFieldDescriptor
CredentialType	CredentialType
CreditType	CreditType
DeliveryMethodType	DeliveryMethodType
Descriptor	Descriptor
DiagnosisDescriptor	DiagnosisDescriptor
DiagnosisType	DiagnosisType
DiplomaLevelType	DiplomaLevelType
DiplomaType	DiplomaType
DisabilityCategoryType	DisabilityCategoryType
DisabilityDescriptor	DisabilityDescriptor
DisabilityType	DisabilityType
DisciplineAction	DisciplineAction
	DisciplineActionDiscipline
	DisciplineActionStaff
DisciplineActionLengthDifferenceReasonType	DisciplineActionLengthDifferenceReasonType
DisciplineDescriptor	DisciplineDescriptor
DisciplineIncident	DisciplineIncident

	DisciplineIncidentBehavior
	DisciplineIncidentWeapon
	StudentDisciplineIncidentAssociation
	StudentDisciplineIncidentBehavior
DisciplineType	DisciplineType
EducationalEnvironmentType	EducationalEnvironmentType
EducationContent	EducationContent
	EducationContentSexTypeAppropriateness
	EducationContentGradeLevelTypeAppropriateness
EducationOrganization	EducationOrganization
	EducationOrganizationAddress
	EducationOrganizationCategory
	EducationOrganizationTelephone
	EducationOrgIdentificationCode
	AccountabilityRating
	EducationOrganizationInterventionPrescriptionAssociation
	EducationOrganizationNetworkAssociation
EducationOrganizationCategoryType	EducationOrganizationCategoryType
EducationOrganizationNetwork	EducationOrganizationNetwork
EducationOrganizationPeer	EducationOrganizationPeer
EducationOrgIdentificationSystemType	EducationOrgIdentificationSystemType
EducationPlanType	EducationPlanType
EducationServiceCenter	EducationServiceCenter
ElectronicMailType	ElectronicMailType
EmploymentStatusDescriptor	EmploymentStatusDescriptor
EmploymentStatusType	EmploymentStatusType
EntryType	EntryType
EntryTypeDescriptor	EntryTypeDescriptor
ExitWithdrawType	ExitWithdrawType
ExitWithdrawTypeDescriptor	ExitWithdrawTypeDescriptor

Grade	Grade
GradebookEntry	GradebookEntry
	StudentGradebookEntry
	GradebookEntryLearningObjective
	GradebookEntryLearningStandard
GradeLevelDescriptor	GradeLevelDescriptor
GradeLevelType	GradeLevelType
GradeType	GradeType
GradingPeriod	GradingPeriod
GradingPeriodDescriptor	GradingPeriodDescriptor
GradingPeriodType	GradingPeriodType
GraduationPlan	GraduationPlan
	GraduationPlanCreditsByCourse
	GraduationPlanCreditsBySubject
GraduationPlanType	GraduationPlanType
GraduationPlanTypeDescriptor	GraduationPlanTypeDescriptor
IncidentLocationType	IncidentLocationType
InstitutionTelephoneNumberType	InstitutionTelephoneNumberType
InteractivityStyleType	InteractivityStyleType
InternetAccessType	InternetAccessType
Intervention	Intervention
	InterventionStaff
	InterventionEducationContent
	InterventionGradeLevelTypeAppropriateness
	InterventionInterventionPrescription
	InterventionPopulationServedType
	InterventionSexTypeAppropriateness
	InterventionMeetingTime
InterventionClassType	InterventionClassType
InterventionEffectivenessRatingType	InterventionEffectivenessRatingType
InterventionPrescription	InterventionPrescription
	InterventionPrescriptionSexTypeApprop

	riateness
	InterventionPrescriptionGradeLevelTypeAppropriateness
	InterventionPrescriptionPopulationServedType
	InterventionPrescriptionDiagnosisType
	InterventionPrescriptionEducationContent
InterventionStudy	InterventionStudy
	InterventionStudySexType
	InterventionStudyGradeLevelType
	InterventionStudyInterventionEffectiveness
	InterventionStudyPopulationServedType
	InterventionStudyEducationContent
	InterventionStudyStateAbbreviationType
ItemCategoryType	ItemCategoryType
LanguageDescriptor	LanguageDescriptor
LanguagesType	LanguagesType
LEACategoryType	LEACategoryType
LearningObjective	LearningObjective
LearningStandard	LearningStandard
	LearningStandardIdentificationCode
LeaveEvent	LeaveEvent
LeaveEventCategoryType	LeaveEventCategoryType
LevelDescriptor	LevelDescriptor
	LevelDescriptorGradeLevel
LevelOfEducationDescriptor	LevelOfEducationDescriptor
LevelOfEducationType	LevelOfEducationType
LimitedEnglishProficiencyDescriptor	LimitedEnglishProficiencyDescriptor
LimitedEnglishProficiencyType	LimitedEnglishProficiencyType
LocalEducationAgency	LocalEducationAgency
Location	Location

MagnetSpecialProgramEmphasisSchoolType	MagnetSpecialProgramEmphasisSchoolType
MediumOfInstructionType	MediumOfInstructionType
MeetingDaysType	MeetingDaysType
MethodCreditEarnedType	MethodCreditEarnedType
NetworkPurposeType	NetworkPurposeType
ObjectiveAssessment	ObjectiveAssessment
	ObjectiveAssessmentLearningObjective
	ObjectiveAssessmentLearningStandard
	ObjectiveAssessmentItem
	ObjectiveAssessmentPerformanceLevel
OldEthnicityType	OldEthnicityType
OpenStaffPosition	OpenStaffPosition
	OpenStaffPositionAcademicSubject
	OpenStaffPositionInstructionalGradeLevel
OperationalStatusType	OperationalStatusType
OtherNameType	OtherNameType
Parent	Parent
	ParentAddress
	ParentInternationalAddress
	ParentElectronicMail
	ParentOtherName
	ParentTelephone
Payroll	Payroll
PerformanceBaseType	PerformanceBaseType
PerformanceLevelDescriptor	PerformanceLevelDescriptor
Person	Person
PopulationServedType	PopulationServedType
PostingResultType	PostingResultType
PostSecondaryEvent	PostSecondaryEvent
PostSecondaryEventCategoryType	PostSecondaryEventCategoryType
Program	Program

	ProgramService
ProgramAssignmentDescriptor	ProgramAssignmentDescriptor
ProgramAssignmentType	ProgramAssignmentType
ProgramSponsorType	ProgramSponsorType
ProgramType	ProgramType
PublicationStatusType	PublicationStatusType
RaceType	RaceType
ReasonExitedDescriptor	ReasonExitedDescriptor
ReasonExitedType	ReasonExitedType
ReasonNotTestedType	ReasonNotTestedType
RecognitionType	RecognitionType
RelationType	RelationType
RepeatIdentifierType	RepeatIdentifierType
ReportCard	ReportCard
	ReportCardGrade
	ReportCardStudentCompetencyObjective
ReporterDescriptionDescriptor	ReporterDescriptionDescriptor
ReporterDescriptionType	ReporterDescriptionType
ResidencyStatusDescriptor	ResidencyStatusDescriptor
ResidencyStatusType	ResidencyStatusType
ResponseIndicatorType	ResponseIndicatorType
ResponsibilityDescriptor	ResponsibilityDescriptor
ResponsibilityType	ResponsibilityType
RestraintEvent	RestraintEvent
	RestraintEventReason
	RestraintEventProgram
RestraintEventReasonsType	RestraintEventReasonsType
ResultDatatypeType	ResultDatatypeType
RetestIndicatorType	RetestIndicatorType
School	School
	SchoolCategory
	SchoolGradeOffered

	FeederSchoolAssociation
SchoolCategoryType	SchoolCategoryType
SchoolFoodServicesEligibilityDescriptor	SchoolFoodServicesEligibilityDescriptor
SchoolFoodServicesEligibilityType	SchoolFoodServicesEligibilityType
SchoolType	SchoolType
SchoolYearType	SchoolYearType
Section	Section
	SectionProgram
	SectionSectionCharacteristic
SectionCharacteristicsDescriptor	SectionCharacteristicsDescriptor
SectionCharacteristicsType	SectionCharacteristicsType
SeparationReasonDescriptor	SeparationReasonDescriptor
SeparationReasonType	SeparationReasonType
SeparationType	SeparationType
ServiceDescriptor	ServiceDescriptor
Session	Session
	SessionGradingPeriod
	AcademicWeek
SexType	SexType
SpecialEducationSettingDescriptor	SpecialEducationSettingDescriptor
SpecialEducationSettingType	SpecialEducationSettingType
Staff	Staff
	StaffAddress
	StaffInternationalAddress
	StaffRace
	StaffElectronicMail
	StaffIdentificationCode
	StaffOtherName
	StaffTelephone
	Credential
StaffClassificationDescriptor	StaffClassificationDescriptor
StaffClassificationType	StaffClassificationType
StaffEducationOrgAssignmentAssociati	StaffEducationOrgAssignmentAssociati

on	on
	StaffEducationOrgAssignmentEmploymentAssociation
StaffEducationOrgEmploymentAssociation	StaffEducationOrgEmploymentAssociation
StaffIdentificationSystemType	StaffIdentificationSystemType
StaffProgramAssociation	StaffProgramAssociation
StateAbbreviationType	StateAbbreviationType
StateEducationAgency	StateEducationAgency
Student	Student
	StudentCharacteristic
	StudentIndicator
	StudentLanguage
	Disability
	StudentTelephone
	StudentElectronicMail
	StudentCohortYear
	StudentRace
	StudentIdentificationCode
	StudentHomeLanguage
	StudentOtherName
	StudentAddress
	StudentInternationalAddress
	StudentLearningStyle
	StudentProgramParticipation
	StudentParentAssociation
StudentAcademicRecord	StudentAcademicRecord
	StudentAcademicRecordReportCard
	Diploma
	Recognition
	AcademicHonor
	ClassRanking
	CourseTranscript

	AdditionalCredit
StudentAssessment	StudentAssessment
	StudentAssessmentPerformanceLevel
	StudentAssessmentAccommodation
	StudentAssessmentScoreResult
	StudentAssessmentItem
	StudentObjectiveAssessment
	StudentObjectiveAssessmentScoreResult
	StudentObjectiveAssessmentPerformanceLevel
StudentCharacteristicType	StudentCharacteristicType
StudentCompetencyLearningObjective	StudentCompetencyLearningObjective
StudentCompetencyObjective	StudentCompetencyObjective
StudentCTEProgramAssociation	StudentCTEProgramAssociation
StudentEducationOrganizationAssociation	StudentEducationOrganizationAssociation
StudentIdentificationSystemType	StudentIdentificationSystemType
StudentInterventionAssociation	StudentInterventionAssociation
	StudentInterventionAssociationInterventionEffectiveness
StudentInterventionAttendanceEvent	StudentInterventionAttendanceEvent
StudentMigrantEducationProgramAssociation	StudentMigrantEducationProgramAssociation
StudentParticipationCodeType	StudentParticipationCodeType
StudentProgramAssociation	StudentProgramAssociation
	StudentProgramAssociationService
StudentSchoolAssociation	StudentSchoolAssociation
	StudentSchoolAssociationGraduationPlan
	StudentSchoolAssociationEducationPlan
StudentSchoolAttendanceEvent	StudentSchoolAttendanceEvent
StudentSectionAssociation	StudentSectionAssociation
StudentSectionAttendanceEvent	StudentSectionAttendanceEvent

StudentSpecialEdProgramAssociation	StudentSpecialEdProgramAssociation
	StudentSpecialEdProgramAssociationServiceProvider
StudentTitleIPartAProgramAssociation	StudentTitleIPartAProgramAssociation
TeacherSchoolAssociation	TeacherSchoolAssociation
	TeacherSchoolAssociationAcademicSubject
	TeacherSchoolAssociationInstructionalGradeLevel
TeacherSectionAssociation	TeacherSectionAssociation
TeachingCredentialBasisType	TeachingCredentialBasisType
TeachingCredentialDescriptor	TeachingCredentialDescriptor
TeachingCredentialType	TeachingCredentialType
TelephoneNumberType	TelephoneNumberType
TermType	TermType
TitleIPartAParticipantType	TitleIPartAParticipantType
TitleIPartASchoolDesignationType	TitleIPartASchoolDesignationType
WeaponDescriptor	WeaponDescriptor
WeaponType	WeaponType