Person API

Introduction

The Person API takes into consideration the following standards and current implementations:

- University of Wisconsin Person SOAP Service
- CIFER API
- TIER API Working Group

Standard	Quick Review
UoW Person SOAP Service	uds_personv1.2.xsd
CIFER API	https://spaces.internet2.edu/display/cifer/SOR-Registry+Core+Schema+Specification
TIER API Working Group	TBD

Data Definition Comparison

This section focuses on defining the data that the API is meant to provide both in requests / responses. The current XML data definition provided in the UOW sample is shown below:

Conversion of this XML Schema to a sample XML message is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<p:Person xmlns:p="http://www.wisc.edu/xsd/UDS/uds_personv1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wisc.edu/xsd/UDS/uds_personv1
uds_personv1.2.xsd ">
  <p:Demographic>
    <p:Name>
      <p:Full>p:Full</p:Full>
      <p:First>p:First</p:First>
      <p:Middle>p:Middle</p:Middle>
      <p:Last>p:Last</p:Last>
      <p:Preferred_Middle>p:Preferred_Middle</p:Preferred_Middle>
      <p:Preferred_First>p:Preferred_First</p:Preferred_First>
    </p:Name>
    <p:Email>p:Email</p:Email>
    <p:PrivacyPolicies>
      <p:Policy>
        <p:Source>p:Source</p:Source>
        <p:Attributes>
          <p:Attribute>p:Attribute</p:Attribute>
        </p:Attributes>
      </p:Policy>
    </p:PrivacyPolicies>
  </p:Demographic>
  <p:Roles>
```

```
  <p:Role>p:Role</p:Role>
</p:Roles>
<p:Identifiers>
  <p:Identifier>
    <p:Source>p:Source</p:Source>
    <p:IdName>p:IdName</p:IdName>
    <p:Value>p:Value</p:Value>
  </p:Identifier>
</p:Identifiers>
<p:Employee>
  <p:Name>
    <p:Full>p:Full</p:Full>
    <p:First>p:First</p:First>
    <p:Middle>p:Middle</p:Middle>
    <p:Last>p:Last</p:Last>
    <p:Preferred_Middle>p:Preferred_Middle</p:Preferred_Middle>
    <p:Preferred_First>p:Preferred_First</p:Preferred_First>
  </p:Name>
  <p:Address>
    <p:Line1>p:Line1</p:Line1>
    <p:Line2>p:Line2</p:Line2>
    <p:Line3>p:Line3</p:Line3>
    <p:Line4>p:Line4</p:Line4>
    <p:City>p:City</p:City>
    <p:State>p:State</p:State>
    <p:Zip>p:Zip</p:Zip>
  </p:Address>
  <p:Email>p:Email</p:Email>
  <p:Phone>p:Phone</p:Phone>
  <p:Appointments>
    <p:Appointment>
      <p:AppointmentType>p:AppointmentType</p:AppointmentType>
      <p:AppointmentID>p:AppointmentID</p:AppointmentID>
      <p:TitleCode>p:TitleCode</p:TitleCode>
      <p:Title>p:Title</p:Title>
      <p:UDDS>
        <p:Code>p:Code</p:Code>
        <p:Abbreviation>p:Abbreviation</p:Abbreviation>
        <p:Unit>p:Unit</p:Unit>
        <p:Division>p:Division</p:Division>
        <p:Department>p:Department</p:Department>
        <p:Subdepartment>p:Subdepartment</p:Subdepartment>
      </p:UDDS>
    </p:Appointment>
  </p:Appointments>
  <p:Jobs>
    <p:Job>
      <p:Class>p:Class</p:Class>
      <p:TitleCode>p:TitleCode</p:TitleCode>
      <p:Title>p:Title</p:Title>
      <p:UDDS>
        <p:Code>p:Code</p:Code>
        <p:Abbreviation>p:Abbreviation</p:Abbreviation>
```

```
        <p:Unit>p:Unit</p:Unit>
        <p:Division>p:Division</p:Division>
        <p:Department>p:Department</p:Department>
        <p:Subdepartment>p:Subdepartment</p:Subdepartment>
      </p:UDDS>
    </p:Job>
  </p:Jobs>
</p:Employee>
<p:Student>
  <p:Name>
    <p:Full>p:Full</p:Full>
    <p:First>p:First</p:First>
    <p:Middle>p:Middle</p:Middle>
    <p:Last>p:Last</p:Last>
    <p:Preferred_Middle>p:Preferred_Middle</p:Preferred_Middle>
    <p:Preferred_First>p:Preferred_First</p:Preferred_First>
  </p:Name>
  <p:Email>p:Email</p:Email>
</p:Student>
<p:Special>
  <p:Name>
    <p:Full>p:Full</p:Full>
    <p:First>p:First</p:First>
    <p:Middle>p:Middle</p:Middle>
    <p:Last>p:Last</p:Last>
    <p:Preferred_Middle>p:Preferred_Middle</p:Preferred_Middle>
    <p:Preferred_First>p:Preferred_First</p:Preferred_First>
  </p:Name>
  <p:Address>
    <p:Line1>p:Line1</p:Line1>
    <p:Line2>p:Line2</p:Line2>
    <p:Line3>p:Line3</p:Line3>
    <p:Line4>p:Line4</p:Line4>
    <p:City>p:City</p:City>
    <p:State>p:State</p:State>
    <p:Zip>p:Zip</p:Zip>
  </p:Address>
  <p:Email>p:Email</p:Email>
  <p:Phone>p:Phone</p:Phone>
  <p:Sponsor>
    <p:UDDS>
      <p:Code>p:Code</p:Code>
      <p:Abbreviation>p:Abbreviation</p:Abbreviation>
      <p:Unit>p:Unit</p:Unit>
      <p:Division>p:Division</p:Division>
      <p:Department>p:Department</p:Department>
      <p:Subdepartment>p:Subdepartment</p:Subdepartment>
    </p:UDDS>
```

```
    </p:Sponsor>
  </p:Special>
</p:Person>
```

Conversion of the XML example to a student example is now shown:

```
{
    "demographics": {
        "name": {
            "first": "Crosby",
            "last": "Nash",
            "middle": "Stills",
            "preferred_Middle": "None",
            "preferred_First": "Crosby",
            "full": "Crosby Stills Nash"
        },
        "email": "csn@gmail.com",
        "policies": {
            "policy": null
        }
    },
    "identifiers": {
        "identifier": {
            "idValue": "someIdValue",
            "value": "SomeValue",
            "source": "SomeSource"
        }
    },
    "roles": {
        "role": "Student"
    },
    "student": {
        "name": {
            "first": "Crosby",
            "last": "Nash",
            "middle": "Stills",
            "preferred_Middle": "None",
            "preferred_First": "Crosby",
            "full": "Crosby Stills Nash"
        },
        "email": "csn@gmail.com"
    },
    "employee": null,
    "special": {
        "name": {
            "first": "Crosby",
            "last": "Nash",
            "middle": "Stills",
            "preferred_Middle": "None",
            "preferred_First": "Crosby",
            "full": "Crosby Stills Nash"
```

```
},
"address": {
    "state": "MA",
    "line1": "None",
    "line2": "Apt #2",
    "line3": "",
    "line4": "None",
    "city": "Worcester",
    "zip": "00000"
},
"email": "csn@gmail.com",
"phone": "444-444-4444",
"sponsor": {
    "udds": {
        "abbreviation": "abbrev1",
        "division": "division1",
        "department": "",
        "subdepartment": "subdepartment1",
        "code": "code1",
        "unit": "unit1"
    }
```

```
            }
        }
    }
```

Both the UDSPerson and CIFER Registry Core Schema cover similar details as shown below with varying detail in both. Where UDSPerson clearly organizes core elements, CIFER Registry Core is flatter with less organization.

UDS and CIFER Taxonomy Comparison

Core Elements UDS	Core Elements CIFER
Demographic	Person Core Attributes
Roles	Person Role Attributes
Identifiers	Person Core Attributes
Employee	HR Person Core Attributes
Student	Student Person Role Attributes
Special	N/A

CIFER Person Core Attributes Comparison

UDS Attributes	CIFER Person Core Attribute
Employee/Address or Special/Address	`address`
None	`address/country`
None	`address/formatted`
None	`address/language`
None	`address/locality`
Address/Zip	`address/postalCode`
Address/City, Address/State	`address/region`
None	`address/room`
Address/Line1,2,3,4	`address/street` ⚠ Changing to `streetAddress`
None	`address/type`
None	`address/verified`
None	`citizenship`
None	`dateOfBirth`
See below	`emailAddress`
Employee/Email, Student/Email, Special/Email	`emailAddress/address`

None	emailAddress/type
None	emailAddress/verified
None	ethnicity
None	gender
Identifiers/Identifier	identifier
TBD	identifier/identifier
TBD	identifier/type
TBD	identityProof
TBD	identityProof/dateOfBirth
TBD	identityProof/documentIssuer
TBD	identityProof/documentType
TBD	identityProof/fullName
TBD	identityProof/status
TBD	identityProof/timeVerified
TBD	identityProof/verifiedAddress
Demographic/Name, Employee/Name, Student/Name, Special/Name	name
Name/Last, Name/Full	name/family
None	name/formatted
Name/First	name/given
None	name/language
Name/Middle	name/middle
TBD	name/prefix
TBD	name/suffix
TBD	name/type
TBD	photo
TBD	photo/data
TBD	photo/encoding
TBD	photo/type
TBD	primaryAffiliation
TBD	primaryCampus
Roles/Role	role
Employee/Phone	telephoneNumber
Employee/Phone	telephoneNumber/number
None	telephoneNumber/type
None	telephoneNumber/verified
None	test
None	url

None	`url/type`
None	`url/url`
None	`visa`

CIFER Person Role Core Attributes Comparison

UDS Attributes	CIFER Person Role Attribute
TBD	`address`
TBD	`affiliation`
TBD	`campus`
TBD	`campusCode`
TBD	`department`
TBD	`departmentCode`
TBD	`displayTitle`
TBD	`emailAddress`
TBD	`identifier`
TBD	`leaveBegins`
TBD	`leaveEnds`
TBD	`manager`
TBD	`organization`
TBD	`organizationCode`
TBD	`percentTime`
TBD	`rank`
TBD	`rankSor`
TBD	`roleBegins`
TBD	`roleEnds`
TBD	`sor`
TBD	`sponsor`
TBD	`status`
TBD	`telephoneNumber`
TBD	`terminationReason`
TBD	`title`
TBD	`url`
TBD	`validFrom`
TBD	`validThrough`

CIFER HR Person Role Attributes Comparison

UDS Attributes	CIFER HR Person Role Attribute

Employee	`employeeType`

CIFER Student Person Role Attributes Comparison

UDS Attribute	CIFER Student Person Role Attribute
None	`classYear`
None	`courseAffiliation`
None	`courseMembership`
None	`degree`
None	`major`
None	`major/major`
None	`major/majorCode`
None	`major/type`
None	`residenceHall`
None	`studentType`
None	`term`

Data Definition Recommendations

From a data perspective, the following are recommendations for create a new Person data model

- New data model must be a superset of current XML Definition to maintain support for existing consumers.
- Use of grouping similar to current UDS provides the ability to ask for partial responses but also makes navigation by consumers more difficult (i.e. parameter vs. parent.child.parameter)
- Use of grouping makes data definition more readable / consumable by developers.
- Consider supporting both XML and JSON representations and have consumer specify this via the Accept header or within the URI. The following provide some examples:

```
Accept: application/json

?type=xml
```

Functional Definition Recommendations

This section will identify different activities that need to performed on the Person API.

PUT vs. POST vs. PATCH

PUT is idempotent, requires you know the resource identifier or pass it in upon creation, requires you send all the data for the resource. Effectively it replaces the resource that you are communicating with.

POST is non-idempotent, does not require you know the resource identifier but that you provide the Location header with its location and doesn't require that you send all the data for the resource. Effectively it is creating a resource or updating an existing resource.

PATCH is part of the newer HTTP specification and is not fully supported in all browsers. It is meant to provide partial updates similar to POST and is also non-idempotent. Recommend not using this feature until it is universally available.

Collections vs. Instance and Pagination

There are two types of resources that are utilized within REST. The first is an instance such as an individual person. The second is a collection which represents people that fit the filter criteria.

- Use pagination to limit results and allow user-agent to specify this as input in the URI (i.e. ?offset=50&limit=25)
- Offset is the starting point of the query, so in previous example start at record number 50 and return up to record 75.
- Limit is the total number of records, so in the previous example return 25 records.

Partial Representation

To limit results, one approach is to use query parameters to communicate a partial representation of a resource versus the entire resource or collection. The resource would then use this to filter the result.

- Only show me the roles and student information for person whose id is 123 (i.e /v1/person/123?fields=roles,student)

HTTP Verb and URI

When defining the URI the following are general recommendations:

- Nouns in the URI, not Verbs (i.e. person vs. getPerson, createPerson). HTTP Verbs identify the action being performed on resource.
- Keep URI to version, resource name, identifier, sub-category to keep URI reasonable

Verb	URI	Result	HTTP Response Codes
GET,POST	/v1/person	Retrieve a collection of people, creates a person and returns person identifier	200,201,404,500,503
GET,PUT,DELETE	/v1/person/:personId	Retrieve, replace / update or delete a person using personId	200,404,500
GET,PUT	/v1/person/:personId/demographic	Retrieve demographics for a person using personId	200, 404 ,500
GET,PUT	/v1/person/:personId/roles	Retrieve roles for a person using personId	200, 404 ,500
GET,PUT	/v1/person/:personId/identifiers	Retrieve identifiers for a person using personId	200, 404 ,500
GET,PUT	/v1/person/:personId/employee	Retrieve employee detail for a person using personId	200, 404 ,500
GET,PUT	/v1/person/:personId/student	Retrieve student detail for a person using personId	200, 404 ,500
GET,PUT	/v1/person/:personId/special	Retrieve special detail for a person using personId	200, 404 ,500
GET	/v1/person?name=value&name=value	Get people that match certain criteria TBD	200, 404 ,500