Identity Registration Web Service V2 - Resources - Person

This document describes the different resources available through the Person Resource.

Sources

The Person resource supports several source record types. A source record is information that describes a part of a person's overall identity (e.g. for a student employee, their student source record would contain their student information, and their employee source record would contain their employee information). The source records are described here:

Source Record Type	Record Identifier	Institutional Identifier	Description	Notes
MyUW Giving Sponsor	`mugs`	Advance ID	Low-assurance identities for UW Donors.	
Office of Educational Assessment	`oea`	OEA source system record identifier	Office of Educational Assessment	
HEPPS	`hepps`	Employee Identification Number	EID sourced from payroll database	Available in Dev, Eval, and Prod environments.
UWHR	`uwhr`	Employee Identification Number	EID (Identity Registry is System of Record)	Only available in the HRP-Dev environment.
SDB	`sdb`	Student Identification Number	Student ID sourced from registrar	
Test	`test`	Test Identifier	Test Identifier specified as needed	For Testing and QA

Attributes

IRWS Person Attributes describes all the attributes of the Person resource, as well as which attributes must be defined when performing certain operations (create, update, etc.) on a record.

Actions

Note: Your application may not be authorized for all of these actions.

Read a Record (GET)

You can read a source record with the following HTTP request:

`GET` *{baseurl}*`/person/`*{source}*`/`*{validid}*

where *{baseurl}* is the base URL for the service, *{source}* is the IRWS source record type from the list of sources, and *{validid}* is the institutional identifier for this record. No payload is required, and it will be ignored if provided. Details on parameters and schemas are available in the API Specification.

IRWS Person Attributes provides detailed descriptions of attributes.

The response always includes an ETag header, which can be used on subsequent updates or deletes.

Create a Record (PUT)

You can create a source record with the following HTTP request:

`PUT` *{baseurl}*`/person/`*{recordidentifier}*`/`*{identifier}*

where *{baseurl}* is the base URL for the service, *{recordidentifier}* is the IRWS record type from the list of sources, and *{identifier}* is the institutional identifier for this record. A JSON payload with the appropriate attributes is required.

If you are working with the UWHR resource in the hrp-dev environment, do not specify an identifier. Instead use:

*PUT {baseurl}/person/uwhr/**

and include the appropriate UWHR attributes. The identity registry will generate an EID and include it in the response payload.

Details on parameters and schemas are available in the API Specification .

The response always includes an ETag header, which can be used on subsequent updates or deletes.

Auto Matching (duplicate record prevention)

If certain attributes are specified in the payload, the Identity Registry will attempt to match the new record to an existing identity. This process is described in detail in Auto Matching. IRWS will return a 409 error if you attempt to add a source record to an identity, and that identity is already associated with a source record of the same type (e.g. you could add an sdb record to an identity that has a uwhr record, but you couldn't add a second uwhr record to that identity).

Update a Record (POST)

You can update a source record with the following HTTP request:

`POST {baseurl}/person/`*`{recordidentifier}`*`/{identifier}`

where *{baseurl}* is the base URL for the service, *{recordidentifier}* is the IRWS identifier from the list of sources, and *{identifier}* is the unique identifier for this record. A JSON payload with the appropriate attributes is required.

Details on parameters and schemas are available in the API Specification .

The response always includes an ETag header, which can be used on subsequent updates or deletes.

Create or Update a PAC (POST)

A PAC is *not* automatically generated when a record is created. A PAC can be created or updated by a POST operation as follows:

`POST `*`{baseurl}`*`/person/`*`{recordidentifier}/{identifier}`*`/pac`

where *{baseurl}* is the base URL for the service, *{recordidentifier}* is the IRWS record type from the list of sources, and *{identifier}* is the institutional identifier for this record. No payload is required, and it will be ignored if provided.

Details on parameters and schemas are available in the API Specification .

Searching for a Record

You can search for a record within a resource by executing a GET operation as follows:

GET *{baseurl}*/person

and providing search parameters via querystring or JSON payload (where *{baseurl}* is the base URL for the service) . You must specify one (and only one) validid as the primary search term. The validid should be typecast into one of the following:

Source identifier validid types:

- sdb (validid=sdb={syskey})
- hepps (validid=hepps={eid})
- advance (validid=advance={advanceid})
- clinician (validid=clinician={centralid})
- many other smaller source populations (ask if you need help)

A shorthand notation that drops the source name will usually work (e.g. for SDB, validid={syskey}), but since there is no guarantee that identifiers in different sources use unique namespaces, it is best to be explicit and include the source name.

Special validid types (not source dependent):

- lname (validid=lname={lname})
- regid (validid=regid={regid})
- ssn (validid=ssn={ssn})
- studentid (validid=studentid={studentid})
- uwnetid (validid=uwnetid={uwnetid})

A shorthand notation that drops "validid=" will work, but in some cases (lname, ssn) it may lead to ambiguities about which term you intend to use as the primary search term and which are secondary search terms. Best to be explicit.

In addition to the primary validid search attribute, you may specify 0 or more of the following secondary search attributes to refine the result set:

- birthdate (birthdate={birthdate})

- fname (fname={fname}) - only in combination with lname as primary
- lname (lname={lname}) - lname can be used as a primary or secondary search term
- ssn (ssn={ssn}) - ssn can be used as a primary or secondary search term

> A last name search will consider last name and prior last name for those record types that support prior last name (Advance). For example, if a person's last name was Smith and it changed to Jones, a last name search on Smith will return the Jones record.

Failure to follow the search rules will return an HTTP 400 response.

See the Attribute Descriptions page for further information on these attributes.

Search Algorithm

Searches are not case-sensitive and do not, by default, do substring matching. However, the asterisk (*) and question mark (?) regular expression operators are supported in some cases.

> Reminder: The question mark operator must be urlencoded if used in a query string.

Wildcards are not supported on primary (validid) searches, except when using lname. So a search for "validid=lname=john*" would return search results for people with last names of "John", "Johnson", or "Johnston", but a search for "validid=ssn=509*" wouldn't return any records.

Wildcards are supported on all secondary search attributes, including ssn.

> Searches for lname and fname are normalized by removing spaces and punctuation marks, and then compared to a normalized version of the name in the registry. For example a search for lname "O'Malley" would be normalized to "omalley", and then return results for all people with an lname of "O'Malley", "OMalley", or "O Malley".

> The attribute referred to as "fname" in the API is actually a givenName (all parts of a name except the last name). A person may have records in multiple sources with a first name in fname, or a first name and middle initial in fname, or a first name and middle name in fname. Due to this variation across sources, it is recommended that you use a wildcard on fname searches.

Examples

Most of these examples demonstrate query strings but a JSON payload example is provided at the end of this section.

Search Only by Social Security Number

```
{baseurl}/person?validid=ssn=123456789
```

Search by Last Name and First Name

```
{baseurl}/person?validid=lname=spud&fname=jo*
```

Search by Last Name and Birth Date

```
{baseurl}/person?validid=lname=spud&birthdate=1234567
```

JSON Search Payload

```
{
        "person": [
                {
                        "lname": "Spud"
                        ,"fname": "Joe"

           }
 ]
}
```