


API Source

 This plugin is experimental, and interfaces may change across minor releases.

- Modes
- Installation
 - Poll Mode (Apache Kafka)
- Configuration
 - Push Mode
 - Pull Mode
 - Poll Mode
 - Poll Mode (Apache Kafka)
- Endpoints and Actions
 - Push Mode
 - Pull Mode
 - Poll Mode
- Message Format
 - PUT (Request) / GET (Response)
 - sorAttributes
 - returnURL
 - PUT (Response)
 - Poll (Apache Kafka)
- Multiple Roles
 - Message Meiosis
 - Message Format
 - GET
 - DELETE
 - Updating Records
 - Use With Bulk Load
 - Additional Considerations
- See Also

The API Organizational Identity Source Plugin is designed to integrate using RESTful APIs and message buses. The API implemented is based on, but not identical to, the [CIFER SOR-Registry Strawman Write API](#).

Modes

Org Identity Source Mode	Support
Manual Search and Linking	Supported in Pull Mode
Enrollment, Authenticated	Not supported
Enrollment, Claim	Not supported
Enrollment, Search	Supported in Pull Mode
Enrollment, Select	Supported in Pull Mode

Org Identity Sync Mode	Support
Full	Supported in Pull Mode
Query	Supported in Pull Mode
Update	Supported in Pull Mode
Manual	Supported in Pull Mode

Installation

This is a non-core plugin, see [Installing and Enabling Registry Plugins](#) for more information.

Poll Mode (Apache Kafka)

Use of the Apache Kafka Poll Mode requires the installation of the [PHP-rdkafka](#) library. This library is *not* distributed with Registry and must be installed separately by following the library's installation instructions. PHP-rdkafka v4.0.0 or later is required.

Configuration

ApiSource supports three modes of operations:

- **Push Mode:** The System of Record sends records to ApiSource via ApiSource's API
- **Pull Mode:** ApiSource queries the System of Record via the SoR's API (*not yet implemented*)
- **Poll Mode:** ApiSource polls an endpoint (typically a message bus) for records to process (*implemented for Kafka on an experimental basis*)

All three modes may be used concurrently, so long as a single SORID space is in effect. (ie: All three modes will share the same unique key for the same record subject.)

ApiSource must be instantiated once per System of Record.

Push Mode

Push Mode requires the creation of an [API User](#). The specified API User will have read/write access to the ApiSource API, the endpoint of which is made available via the ApiSource configuration page. It is recommended to create an *Unprivileged CO API User* for this purpose, though any defined API User may be used.

Push Mode does not support [Org Identity Sync](#) via Job Shell or at Login. Technically, no errors will be thrown, but because Push Mode does not support calling out to the System of Record, no updates will be made.

Pull Mode

Pull Mode requires the System of Record to offer an API with a predictable interface to specific records. In other words, given a specific SOR ID the API must allow for the retrieval of a specific record.

Not yet implemented.

Poll Mode

Poll Mode consumes event-oriented messages, such as those that may be placed on a Message Queue or Bus. As distinct from Pull Mode, it is typically not possible for a specific record to be retrieved on demand, but only when the System of Record sends out an update.

While Poll Mode can be used concurrently with Push and Pull Modes, only one messaging technology can be used for Poll Mode within a given ApiSource instantiation.

Because Poll Mode is dependent on updates being sent through the message system, it does not support [Org Identity Sync](#) via Job Shell or at Login.

Polling is implemented via the [Registry Job Shell](#), using a Job provided by ApiSource.

- Job Name: `ApiSource.Poll`
- Job Parameters
 - `api_source_id`: The ID of the API Source instance to run polling for.
 - Required
 - `max`: The maximum number of records to process before exiting.
 - Default: 100 (Registry v4.0.2 and earlier) or 10 (Registry v4.1.0 and later)
 - As of Registry v4.1.0, for Apache Kafka this is the number of *loops* that may run trying to consume messages. The number of messages that may be consumed is configured by the *Batch Size* setting on the *KafkaServer* object. Thus the actual maximum number of messages that may be consumed is $(max * batch_size)$.
 - This parameter may be refactored in a future minor release.

Sample Usage

```
./Console/Cake job ApiSource.Poll -s -c 2 --max 10 --api_source_id 5
```



Each instantiation of ApiSource in Poll Mode requires a separate Job instance with the associated `api_source_id`, each of which must be manually configured in cron.




The ApiSource Job Shell operates independently of the [Organizational Identity Source](#) `syncorgsources` Job. This is subject to change in a future release.

Poll Mode (Apache Kafka)

[Apache Kafka](#) support is available as of Registry v4.0.0. ⚠️ Kafka support is *Experimental* and may change across minor releases.

Add a new Kafka Server, via `CO > Servers > Add a New Server`. On the next page, set the configuration information as provided by the Kafka administrators.

Instantiate ApiSource and set *Poll Mode* to *Apache Kafka*. Select the *Server* created above. For Registry v4.0.x, set the Consumer Group ID and Kafka Topic as provided by the Kafka administrators. (For Registry v4.1.0 and later, these settings are configured via the Kafka Server.)

 Registry v4.0.x uses the [high-level consumer](#). Registry v4.1.0 and later uses the [low-level consumer](#) and batch consumption of messages.

Endpoints and Actions

Push Mode

The URL prefix for ApiSource operating in Push Mode is

`https://server.org/registry/api_source/coid/v1/sorPeople/sorlabel/sorid`

where

- **coid**: The CO ID for this instance of ApiSource
- **sorlabel**: SOR Label, as configured for the Organizational Identity Source (v4.1.0 and later) or ApiSource (v4.0.2 and earlier) instance
- **sorid**: The System of Record's unique ID for the record being presented

coid and **sorlabel** are used to find the correct instantiation of ApiSource.

The following actions are supported in Push Mode:

- **DELETE**: Remove the specified record from the set of records associated with this SoR. No body is expected.
- **GET**: Obtain the current record for the specified SOR ID. The response will be a record in the same format as was **PUT**.
- **PUT**: Add a record for the specified SOR ID, or update an existing record. The body of the request is in the message format described below.

The following HTTP Response Codes may be returned:

- 200: An existing record was found and deleted, returned, or updated (as appropriate for the action).
- 201: The record was successfully stored and processed (**PUT** only).
- 202: The record was accepted, but further processing (and possibly administrator action) is required.
- 401: Unauthenticated / authorization failed.
- 404: The specified record does not exist (**DELETE** or **GET** only).
- 500: An error occurred.

Pull Mode

Not yet implemented.

Poll Mode

The endpoint is determined by the message system.

When a message is processed for an SORID that has not been seen before, the message will be processed as an add action. If the SORID has been seen before, the message will be processed as on update (or a delete, if the message metadata indicates such, see *Message Format*, below).

Message Format


PUT (Request) / GET (Response)

The message format is a JSON object with two possible members: `sorAttributes` and `returnUrl`.

The message should be sent with a `Content-Type` header of `text/json`.

sorAttributes

`sorAttributes` is a required object whose members are from the following list of available attributes, as defined in the [TAP Attribute Dictionary](#):

- `addresses*`
- `adhoc`
- `affiliation`
- `dateOfBirth`
- `department`
- `emailAddresses*`
- `identifiers*`
- `managerIdentifier†`
- `names*`
 -  The first name found will be designated the Primary Name
- `organization`
- `sponsorIdentifier†`
- `telephoneNumbers*`

- title
- urls*
- validFrom / validThrough

**Plural attributes may have multiple values, provided via a JSON array*

†See [Pipeline Relationship Syncing](#) for more information

returnURL

returnUrl is optional, and consists of a single string containing a URL. This URL is used as a [Petition-Specific Redirect Target](#), if API Source is connected to a [Pipeline](#) that in turn is [connected to an Enrollment Flow](#). Make sure the returnUrl is configured in the Enrollment Flow's *Return URL Whitelist* configuration.

Sample Request Message

```
{
  "sorAttributes": {
    "names": [
      {
        "type": "official",
        "given": "Pat",
        "middle": "X",
        "family": "Lee"
      }
    ],
    "affiliation": "faculty",
    "organization": "School of Philosophy and Biopharmacology",
    "department": "Department of Metaphysics",
    "title": "Associate Professor of Metaphysical Microbiology",
    "dateOfBirth": "1990-04-25",
    "validFrom": "2019-09-01T00:00:00Z",
    "validThrough": "2020-08-31T23:59:59Z",
    "managerIdentifier": "E79725186",
    "sponsorIdentifier": "E69813130",
    "identifiers": [
      {
        "type": "national",
        "identifier": "541-00-3732"
      }
    ],
    "emailAddresses": [
      {
        "type": "personal",
        "address": "patxlee@email.nil",
        "verified": true
      }
    ],
    "addresses": [
      {
        "type": "home",
        "streetAddress": "3593 Red Maple Drive",
        "locality": "Los Angeles",
        "region": "CA",
        "postalCode": "90046",
        "country": "US"
      }
    ],
    "telephoneNumbers": [
      {
        "type": "home",
        "number": "323-555-1208"
      }
    ],
    "urls": [
      {
        "type": "personal",
        "url": "https://metaphysics.spb.ac.nil/plee"
      }
    ],
    "adhoc": [
      {
        "tag": "flavor",
        "value": "chocolate"
      }
    ]
  },
  "returnUrl": "https://registry.myvo.org/localapps/post-enrollment?regid=12345"
}
```

PUT (Response)

The message format is a JSON object, currently with a single member: `identifiers`, representing a list of identifiers associated with the CO Person created by or attached to the SOR record. Additional attributes may be returned in the future.

Sample Response Message

```
{
  "identifiers": [
    {
      "identifier": "1049f0d5-04cc-4ad5-8ab4-b6e056227dcb",
      "type": "reference"
    },
    {
      "identifier": "pxl28",
      "type": "network"
    }
  ]
}
```

Poll (Apache Kafka)

The message format uses the same `sorAttributes` object used for PUT requests and described above, with an additional `meta` section. The expected metadata attributes are

- `resource`: Must be the string `sorPersonRole`
- `version`: Must be the string `1`
- `sor`: A string matching the SOR Label configured for the API Source instance
- `sorid`: The System of Record's unique ID for the record being presented

The JSON document must be stored in the `payload` attribute of the Kafka Message.

There is no response to a polled message. To send a response back to the requesting system, use a provisioner such as the [API Provisioning Plugin](#).

Sample Request Metadata

```
{
  "meta": {
    "resource": "sorPersonRole",
    "version": "1",
    "sor": "kafka",
    "sorid": "K2345987"
  },
  "sorAttributes": {
    ...
  }
}
```

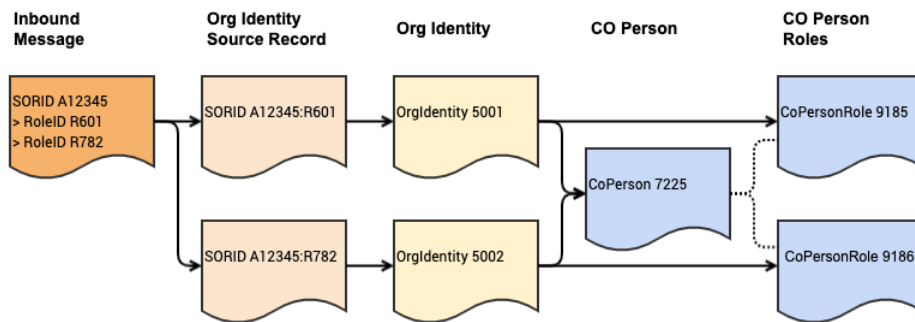
Multiple Roles

Message Meiosis

As of Registry v4.1.0, ApiSource supports an experimental format for transferring multiple roles in a single request. In order to handle these messages, ApiSource converts each inbound message into multiple Org Identity Source Records, in a process referred to as *Message Meiosis*. Multiple roles are conveyed using a `roles` attribute, each of which has a unique `roleIdentifier` attribute.

During Message Meiosis,

1. For each role, the `roleIdentifier` is appended to the `sorid` with a colon (`:`) to create a *Compound SORID*.
 - a. ⚠ The Compound SORID must be unique within the source, which may be an issue if the source uses colons in its System of Record IDs.
2. The non-role attributes are copied to the message for each role.
3. Each Meiosized message is treated by the Org Identity Source and Pipeline infrastructure as if it were a complete, standalone message.



Message Format

ApiSource puts an inbound message through Meiosis based on the presence of the `roles` key within `sorAttributes`, which is instead specified as

- `dateOfBirth`
- `emailAddresses*`
- `identifiers*`
- `names*`
 - The first name found will be designated the Primary Name
- `urls*`
- `roles*`
 - `roleIdentifier`
 - `adhoc`
 - `affiliation`
 - `addresses*`
 - `department`
 - `managerIdentifier†`
 - `organization`
 - `sponsorIdentifier†`
 - `telephoneNumbers*`
 - `title`
 - `validFrom / validThrough`

*Plural attributes may have multiple values, provided via a JSON array

†See [Pipeline Relationship Syncing](#) for more information

where `roleIdentifier` is an identifier used to identify the unique role within the record.

Sample Request Message

```

{
  "sorAttributes": {
    "names": [
      {
        "type": "official",
        "given": "Pat",
        "middle": "X",
        "family": "Lee"
      }
    ],
    "dateOfBirth": "1990-04-25",
    "identifiers": [
      {
        "type": "national",
        "identifier": "541-00-3732"
      }
    ],
    "emailAddresses": [
      {
        "type": "personal",
        "address": "patxlee@email.nil",
        "verified": true
      }
    ],
    "urls": [
      {

```

```

        "type": "personal",
        "url": "https://metaphysics.spb.ac.nil/plee"
    }
],
"roles": [
    {
        "roleIdentifier": "R250001",
        "affiliation": "faculty",
        "organization": "School of Philosophy and Biopharmacology",
        "department": "Department of Metaphysics",
        "title": "Associate Professor of Metaphysical Microbiology",
        "validFrom": "2019-09-01T00:00:00Z",
        "validThrough": "2020-08-31T23:59:59Z",
        "managerIdentifier": "E79725186",
        "sponsorIdentifier": "E69813130",
        "addresses": [
            {
                "type": "home",
                "streetAddress": "3593 Red Maple Drive",
                "locality": "Los Angeles",
                "region": "CA",
                "postalCode": "90046",
                "country": "US"
            }
        ],
        "telephoneNumbers": [
            {
                "type": "home",
                "number": "323-555-1208"
            }
        ],
        "adhoc": [
            {
                "tag": "flavor",
                "value": "chocolate"
            }
        ]
    },
    {
        "roleIdentifier": "R782510",
        "affiliation": "affiliate",
        "organization": "School of Culinary Arts",
        "department": "Recipe Development",
        "title": "Consultant",
        "sponsorIdentifier": "E11109783"
    }
]
},
{
    "returnUrl": "https://registry.myvo.org/localapps/post-enrollment?regid=12345"
}

```

GET

Because the original message is not stored before Meiosis, GET requests will only work with the Compound SORID.

DELETE

DELETE requests must currently use the Compound SORID to delete roles individually.

Updating Records


Because ApiSource performs Meiosis on the inbound message before processing it, update messages need not (but may) contain all role records. However, because of this, omitting a role record in an update will *not* cause the role to be deleted. The role must be deleted using the Compound SORID, as described above.

Use With Bulk Load

When using Multiple Roles with [Registry BulkLoad Shell](#), follow the instructions for Linking to Organizational Identity Sources *once for each role*. Use the Compound SORID as the SORID. The *source record* should represent the post-Meiosis message, not the original (multiple role) message.

Additional Considerations

Because ApiSource "copies" attributes to create multiple records, a single ApiSource message may create multiple Org Identities. However, ApiSource will cache the CO Person ID created for the first role and apply it to later roles. If the Org Identity Source is connected to a Pipeline calling out to an external match service, the match service will therefore receive only one request per inbound message.

 If an OrgIdentity is initially created, and then a new Role is added later, the new Role should be added *after* the existing Role(s) in the JSON `roles` section in order to ensure that the existing CO Person ID is correctly linked. If the new Role is processed first, a new CO Person ID may be assigned (depending on the Match configuration).

The implementation is likely to change in a future release, as support for Organizational Identities with multiple roles is added to the core codebase ([CO-2215](#)).

See Also

- [cm_api_sources](#)
- [cm_api_source_records](#)
- [API Provisioning Plugin](#)