


Registry BulkLoad Shell

BulkLoad Shell allows for faster loading of large initial datasets. BulkLoad Shell is available as of Registry v3.3.0. The benefits of BulkLoad Shell are typically seen for datasets of at least 50k records.

- [BulkLoad Shell Constraints](#)
- [Preparing Data For BulkLoad](#)
 - [Supported Tables](#)
 - [Plugin Tables](#)
 - [Inbound JSON File Format](#)
 - [File Metadata Object](#)
 - [Data Objects](#)
 - [Cross References](#)
 - [Record Metadata](#)
 - [CO Person Records](#)
 - [Linking to Organizational Identity Sources](#)
 - [CO Group Records](#)
 - [Plugin Configuration Records](#)
 - [Examples](#)
- [Running BulkLoad](#)
 - [BulkLoad Usage](#)
 - [Performance Considerations](#)
 - [Steps](#)
 - [Follow Up Steps](#)

BulkLoad Shell Constraints

It is important to understand that BulkLoad bypasses the normal Registry data processing engine, and as such is only suitable for an initial bulk load of data.

- BulkLoad currently only supports Postgres databases. Support for MySQL/MariaDB may be added in a future release. ([CO-1906](#))
- BulkLoad will disable database indexes, meaning the entire Registry application should be unavailable during the actual load process. In a multi-tenant deployment, bulk loading data for a single CO will require all COs to be offline.
- BulkLoad supports core data model attributes only, which should be sufficient for most use cases.
 -  As of Registry v4.0.0, plugin models are supported as well.
- BulkLoad bypasses data validation, [data normalization](#), [identifier assignment](#), duplicate identifier checks, [record matching](#), [pipeline](#) processing, and [provisioning](#). Prior to loading, data must be formatted and cleansed properly. (Some of these steps, such as identifier assignment and provisioning, may be manually run after the load is completed.)
- [Pooled Org Identities](#) are not supported.
- Additional data model constraints are described below.

Preparing Data For BulkLoad

1. Configure the CO. In order to use [Extended Types](#), populate [COU](#) assignments (via CO Person Roles), [CO Group Memberships](#), and linking to [Organizational Identity Sources](#), the appropriate configurations must first be made in the target CO.
2. Cleanse the inbound data.
 - a. Make sure there are no duplicate identifiers.
 - b. Make sure there are no duplicate CO Group Memberships (ie: that a given CO Person is not added to the same CO Group twice).
 - c. Currently, tabs (\t) are not supported in attribute values.
 - d. Newlines (\n) are supported for fields that support multiple lines, but must be double escaped (\\n).
3. Prepare the inbound file, as described below.
 - a. [Automatic Group Memberships](#) will be established by BulkLoad. Do not include these in the inbound file.

Supported Tables

The following tables are supported. In general, any field specified in the [Registry Data Model](#) documentation is supported. Foreign keys and other attribute metadata will be automatically inserted by BulkLoad, and so should be omitted. CoOrgIdentityLinks will also be automatically inserted.

- [ad_hoc_attributes](#)
- [addresses](#)
- [co_groups](#)
- [co_group_members](#)
- [co_org_identity_links](#)
- [co_people](#)
- [co_person_roles](#)
- [email_addresses](#)
- [history_records](#)
- [identifiers](#)
- [names](#)
- [org_identities](#)

- [org_identity_source_records](#)
- [telephone_numbers](#)
- [urls](#)

Plugin Tables

As of Registry v4.0.0, BulkLoad Shell supports three types of Plugin tables: *CoGroup*, *CoPerson* and *Configuration*. *CoPerson* tables are those with a foreign key to CoPerson (which will automatically be inserted), *CoGroup* tables operate similarly. *Configuration* tables have no dependent foreign keys. The plugin models must be declared in the JSON File metadata (described below).

Inbound JSON File Format

The inbound file consists of multiple JSON objects, one to a line. The first object is a File Metadata object, which is described below. Subsequent lines each hold one object record, as described below.

Data is generally represented as a complete primary object (CoGroup, CoPerson), with related data (EmailAddress, Identifier, CoPersonRole, etc) nested within. Foreign keys are automatically inserted by BulkLoad Shell.

Values must be their actual database values, not application enums. For example, a valid `CoPerson:status` value is "A", not "Active". In general, the values are documented in the [Data Model](#).

File Metadata Object

The following Metadata attributes are available:

- **pluginModels:** Available as of Registry v4.0.0, this specifies records to load that are associated with plugin models. *pluginModels* is specified as an object with up to two keys: *CoPerson* and *Configuration*. The former is a list of models that have foreign keys to CoPerson, the latter is a list of models that do not have foreign keys. Models must be specified using Cake Plugin notation, eg *MyPlugin.MyModel*. See the examples section, below.
- **local:** A JSON object that may be locally defined. This object is intended to allow annotations, comments, and other information to be encoded in the inbound file for locally defined purposes. The entire object is ignored by BulkLoad Shell.

The Metadata line may be an empty JSON object (`{ }`) if no Metadata is required.

The JSON Schema definition of the File Metadata Object is available [here](#).

Data Objects

The JSON Schema definition for Data Objects is available [here](#).


Cross References

It is possible to reference primary objects defined earlier in the inbound file using *Cross References*. This enables scenarios such as

- Bulk Loading CO Groups, then creating CO Group Memberships to those groups within a later CO Person record
- Assigning a Sponsor to a CO Person Role

When preparing the inbound data, each relevant record is given a *cross reference label* (identified in the record metadata via the `xref` attribute). Subsequent objects may refer to the Registry internal foreign key assigned for the newly created object, via `@{label}` notation, which can be used anywhere in any non-metadata attribute of a suitable type (eg: not boolean, etc).

 Cross Reference labels must be alphanumeric.

 Labels are parsed *after* the JSON document is parsed, and so must not create invalid JSON. This mostly means that they must be placed in quotes when used to create a foreign key (eg: `co_group_id`). This will technically convert the value to a JSON string rather than an integer, but PHP will coerce it back to an integer when needed.

Cross References are available as of Registry v4.0.0.

Record Metadata

Each primary object (CO Group, CO Person) may have Metadata attributes, of which the following are available:

- **objectType:** The primary object type, either `CoGroup` or `CoPerson`. This attribute is not currently required.
- **xref:** The cross reference label used to subsequently identify this object. For example, if the xref label `abc1` is assigned to a CoPerson object, the object could be referred to in a later object via `{ "co_person_id": "@{abc1}" }`.
- **local:** A JSON object that may be locally defined. This object is intended to allow annotations, comments, and other information to be encoded in the inbound file for locally defined purposes. The entire object is ignored by BulkLoad Shell.

Record Metadata is available as of Registry v4.0.0.

CO Person Records

Each subsequent line consists of a single JSON object representing a single CO Person. The members of the object are labeled using the model name (CamelCase, singular) in this layout:

- CoPerson (object)
 - [CO Person Status](#) will not be recalculated, so set it to the desired value
- CoGroupMember (array)
 - [Nested Groups](#) are not supported
- CoPersonRole (array)
 - Address (array)
 - AdHocAttribute (array)
 - TelephoneNumber (array)
 - As of Registry v4.0.0, [Sponsors and Managers](#) can be assigned using Cross References, as long as the Sponsor CO Person record is loaded before the Sponsored CO Person record.
- EmailAddress (array)
- HistoryRecord (array)
- Identifier (array)
- Name (array, exactly one Name should be flagged as primary)
- OrgIdentity (array)
 - OrgIdentity (object, OrgIdentity fields go here)
 - Address (array)
 - AdHocAttribute (array)
 - EmailAddress (array)
 - HistoryRecord (array)
 - Identifier (array)
 - Name (array, exactly one Name should be flagged as primary)
 - TelephoneNumber (array)
 - Url (array)
- OrgIdentitySourceRecord (array)
 - OrgIdentity (object)
 - OrgIdentity (object, OrgIdentity fields go here)
 - Address (array)
 - AdHocAttribute (array)
 - EmailAddress (array)
 - HistoryRecord (array)
 - Identifier (array)
 - Name (array, exactly one Name should be flagged as primary)
 - TelephoneNumber (array)
 - Url (array)
- Url (array)
- As of Registry v4.0.0, Plugin Models that have a foreign key into CoPerson (and are declared in the metadata) may also be included as an array.

Records must be "pre matched". If a CO Person has multiple Org Identities, they must be placed in the same JSON object.

Linking to Organizational Identity Sources

Records can be linked to [Organizational Identity Sources](#), to link them for purposes of future updates. To do so, include a suitable OrgIdentitySourceRecord, linked to the appropriate OIS configuration (via the `org_identity_source_id` foreign key). The OrgIdentity as would be returned by the appropriate OIS backend must be included. The OrgIdentity should include the SORID as one of its Identifiers.

Do not specify CO Person Roles that would be created via Pipelines attached to the OIS. BulkLoad will create them.

[CO Group Mappings](#) are *not* supported, because creating these mappings would require instantiating the OIS plugin backend, which would degrade performance. Instead, create the desired CoGroupMember records in the JSON record.

CO Group Records

Each subsequent line consists of a single JSON object representing a single CO Group. The members of the object are labeled using the model name (CamelCase, singular) in this layout:

- CoGroupMember (array)
- HistoryRecord (array)
- Identifier (array)
- Plugin Models that have a foreign key into CoGroup (and are declared in the metadata) may also be included as an array.

[Nested Groups](#) are not supported.

CO Group Records are supported as of Registry v4.0.0.

Plugin Configuration Records

Plugin Configuration Records may be included one per line. See the examples, below.

Examples

The later examples here are shown with newlines for readability, however the actual file should include no newlines, except to separate each record.

Simple records without newlines, including empty metadata

```
{
  {
    {CoPerson":{"status":"A"},"Name":[{"given":"Myrtle","family":"Jefferson","type":"official","primary_name":
true}}}
    {CoPerson":{"status":"A"},"Name":[{"given":"Novella","family":"Torres","type":"official","primary_name":true}}}
```

Simple single record, with newlines for readability

```
{
  "CoPerson": {
    "status": "A"
  },
  "Name": [
    {
      "given": "Myrtle",
      "family": "Jefferson",
      "type": "official",
      "primary_name": true
    }
  ],
  "Identifier": [
    {
      "identifier": "d3b5b15c-3ce2-4ce5-9752-acb941ed0e78",
      "type": "reference",
      "login": false,
      "status": "A"
    },
    {
      "identifier": "476-56-5741",
      "type": "national",
      "login": "false",
      "status": "A"
    }
  ],
  "EmailAddress": [
    {
      "mail": "MyrtleWJefferson@university.edu",
      "type": "official",
      "verified": true
    }
  ],
  "CoGroupMember": [
    {
      "co_group_id": 37,
      "member": true,
      "owner": false
    }
  ],
  "CoPersonRole": [
    {
      "affiliation": "employee",
      "title": "Employee",
      "o": null,
      "ou": "Biology",
      "TelephoneNumber": [
        {
          "country_code": "1",
          "number": "507-798-2339",
          "type": "campus"
        }
      ]
    }
  ],
  "OrgIdentity": [
```

```

{
  "OrgIdentity": {
    "affiliation": "member"
  },
  "Name": {
    "given": "Myrtle",
    "family": "Jefferson",
    "type": "official",
    "primary_name": true
  },
  "Identifier": [
    {
      "identifier": "24n9vBgj@social.com",
      "type": "eppn",
      "login": true,
      "status": "A"
    }
  ],
  "EmailAddress": [
    {
      "mail": "myrtle787@social.com",
      "type": "personal",
      "verified": true
    }
  ]
}
]
}

```

Single record attached to an OIS, with newlines added for readability

```

{
  "CoPerson": {
    "status": "A"
  },
  "Name": [
    {
      "given": "Novella",
      "family": "Torres",
      "type": "official",
      "primary_name": true
    }
  ],
  "Identifier": [
    {
      "identifier": "0dc187ef-5d10-412d-83d2-b55221b556b5",
      "type": "reference",
      "login": false,
      "status": "A"
    },
    {
      "identifier": "509-74-0614",
      "type": "national",
      "login": "false",
      "status": "A"
    }
  ],
  "EmailAddress": [
    {
      "mail": "NovellaDTorres@university.edu",
      "type": "official",
      "verified": true
    }
  ],
  "CoGroupMember": [
    {
      "co_group_id": 22,
      "member": true,
      "owner": false
    }
  ]
}

```

```

    }
  ],
  "OrgIdentitySourceRecord": [
    {
      "org_identity_source_id": 2,
      "sorid": "hrms2",
      "source_record": "{ \"0\": \"hrms2\", \"1\": \"\", \"2\": \"\", \"3\": \"Novella\", \"4\": \"\", \"5\": \"Torres\", \"6\": \"\", \"7\": \"\", \"8\": \"\", \"9\": \"\", \"10\": \"\", \"11\": \"\", \"12\": \"NovellaDTorres@university.edu\", \"13\": \"913-626-2317\", \"14\": \"1\", \"15\": \"509-74-0614\", \"16\": \"Employee\", \"17\": \"Linguistics\", \"18\": \"0dc187ef-5d10-412d-83d2-b55221b556b5\" }",
      "reference_identifier": "0dc187ef-5d10-412d-83d2-b55221b556b5",
      "OrgIdentity": {
        "OrgIdentity": {
          "affiliation": "employee",
          "title": "Employee",
          "o": null,
          "ou": "Linguistics"
        },
        "Address": [],
        "Name": [
          {
            "given": "Novella",
            "family": "Torres",
            "type": "official",
            "primary_name": true
          }
        ],
        "EmailAddress": [
          {
            "mail": "NovellaDTorres@social.com",
            "type": "personal",
            "verified": true
          }
        ],
        "Identifier": [
          {
            "identifier": "0dc187ef-5d10-412d-83d2-b55221b556b5",
            "type": "reference",
            "login": false,
            "status": "A"
          },
          {
            "identifier": "509-74-0614",
            "type": "national",
            "login": "false",
            "status": "A"
          },
          {
            "identifier": "hrms2",
            "type": "sorid",
            "login": "false",
            "status": "A"
          }
        ],
        "TelephoneNumber": [
          {
            "country_code": "1",
            "number": "913-626-9988",
            "type": "mobile"
          }
        ]
      }
    ]
  }
}

```

Metadata header, with newlines for readability

```
{
  "meta":{
    "pluginModels":{
      "CoPerson":[
        "SshKeyAuthenticator.SshKey"
      ],
      "Configuration":[
        "MyPlugin.GreenObject"
      ]
    }
  }
}
{ ... }
```

Record with Metadata

```
{ "meta":{ "pluginModels":{ "CoPerson":["SshKeyAuthenticator.SshKey"], "Configuration":["MyPlugin.GreenObject"]} },
  { "CoPerson":{ "status":"A", "SshKey":[{"ssh_key_authenticator_id":1, "type":"ssh-dss", "skey":"abc123"}] },
  { "GreenObject":{ "my_attribute":"my_value" } },
  { "GreenObject":{ "my_attribute":"my_other_value" } }
```

Cross Reference Example with Unix Clusters and Groups

```
{ "meta":{ "pluginModels":{ "CoPerson":["SshKeyAuthenticator.SshKey", "UnixCluster.UnixClusterAccount"], "
Configuration":["UnixCluster.UnixClusterGroup"]} } }
{ "meta":{ "objectType":"CoGroup", "xref":"gobj1"}, "CoGroup":{ "co_id":"2", "name":"Primary Unix Cluster", "
description":"Primary Unix Cluster users", "open":false, "status":"A", "group_type":"S", "auto":false, "
nesting_mode_all":false} }
{ "meta":{ "objectType":"CoPerson", "xref":"obj1"}, "CoPerson":{ "status":"A", "Name":[{"given":"Nelly", "family":"
Brasel", "type":"official", "primary_name":true}], "CoGroupMember":[{"co_group_id":"@{gobj1}", "member":true, "
owner":false}], "UnixClusterAccount":[{"unix_cluster_id":1, "sync_mode":"F", "status":"A", "username":"nbrasel", "
uid":10001, "gecos":"Nelly Brasel", "login_shell":"/bin/bash", "home_directory":"/home/nbrasel", "
primary_co_group_id":44}] } }
{ "meta":{ "plugin":"UnixCluster", "objectType":"UnixClusterGroup", "xref":"obj2"}, "UnixClusterGroup":{
"unix_cluster_id":1, "co_group_id":"@{gobj1}" } }
```

Running BulkLoad

BulkLoad Usage

```
cake bulk_load [-a actor] [-t dbtype] coid infile
```

where

- **actor**: Actor identifier to record to [Changelog](#) actor_identifier column, defaults to *Bulk Load Shell*
- **coid**: Target CO to load records to
- **dbtype**: Target database type, currently only the default *postgres* is supported
- **infile**: JSON file to process records from (In Registry v3.3.x only, multiple files may be specified)



Be careful with file ownerships here. You will need to run this command as the web server user or as root, depending on whether or not the web server user can read the input file. If running as root, make sure any files generated in the cache directory (wherever `app/tmp` points to) owned by root are subsequently deleted.

Performance Considerations

BulkLoad can process ~500k records on a single vCPU with 2GB of RAM and a local database in about 10 minutes. Larger datasets may require somewhat more memory, but additional vCPUs are unlikely to help much. Make sure the database server has sufficient disk space available. (A few GB should be sufficient, depending on the size of the dataset.) Communications to a database server over a network (vs local to the same server) may result in slower run times.

See also: [Registry Installation - PHP \(Memory Considerations\)](#)

Steps

1. Make sure access to the web interface is unavailable, if anyone else might try to use it.
2. Run the BulkLoad command, specifying the target CO ID to load the records to. Depending on the number of records to load, the process will take a few minutes to complete.
 - a. `./Console/cake bulk_load coid infile.json`
3. If the web interface is unusually slow after the load is completed, try [clearing caches](#) and running the `cake database` command to rebuild the indexes again.

Follow Up Steps

1. If desired, schedule Jobs to assign identifiers and reprovision all. These Jobs may take some time to complete.