The Oomnitza Connector

Oomnitza has created a unified connector, lovingly crafted using Python, which is a single application that can be used to pull data from multiple sources and push it to your Oomnitza application. The connector can presently pull data from the following sources, with more planned in the future.

- Airwatch http://www.air-watch.com
- BambhooHR http://www.bamboohr.com
- Casper http://www.jamfsoftware.com/products/casper-suite/
- Jasper http://www.jasper.com
- LDAP e.g., http://www.openldap.org, Active Directory
- MobileIron http://www.mobileiron.com
- Okta https://www.okta.com
- OneLogin https://www.onelogin.com
- SCCM http://www.microsoft.com
- ZenDesk https://www.zendesk.com

The Oomnitza Connector can be hosted on Oomnitza's server cloud, free of charge, if the third party server is or can be made accessible from the Oomnitza Cloud. Contact us for more details! Organizations with dedicated internal services may prefer to run this connector in-house, behind the same firewall that prevents outside access.

Getting Started

The most current version of this documentation can always be found on GitHub.

Since Oomnitza is highly customizable, there are many possibilities with the connector. Because of this, it is important to think ahead about what data you want to bring in and how you want to store it. Before we begin, take time to think about what information you want, and what Oomnitza fields you want filled out with Casper data. If the fields you want to map in haven't been created yet, now is a good time to do so. (Refer to our Guide to creating custom fields in Oomnitza to get started.)

Getting the Connector

The Oomnitza Connector code is hosted at https://github.com/Oomnitza/oomnitza-connector.

The Oomnitza Connector can also be downloaded from within your Oomnitza instance. Log into your instance and navigate to the System Settings page. Scroll to the bottom of the Integrations page and download either the correct binary or the “Source Code” Package.

- If you will be hosting the connector on a Windows or Mac server, we recommend using the binary version.
- The Source Code package can be use on a Linux server, as well as Windows and Mac. This package requires that a python environment be setup properly, which the binary version avoids.

Runtime Environment Setup

If you choose to run the binary version of the connector, you can skip this section. If you choose to install and run the python code, you will need to install Python 2.7.X as well as the packages which the connector relies upon. Some of the python packages may require build tools to be installed.

Linux Environment

On Ubuntu, the build tools are installed using:

```
> sudo apt-get install build-essential
```

We suggest you setup a virtual environment and use pip to install the requirements. This can be done as follows (See our documentation on installing additional Python modules for use in Oomnitza.):

```
> cd /path/to/connector
> virtualenv .
> source bin/activate
> pip install --upgrade pip
> pip install -r requirements.txt
```

Windows Environment

ActiveState has an excellent Python package for Windows. It can be downloaded from http://www.activestate.com/activepython/downloads. Once this has been downloaded and installed, the remaining setup steps can be performed using PowerShell as an administrator. So, open PowerShell and do the following:

```
> cd c:\
> mkdir oomnitza-connector
> cd oomnitza-connector
> virtualenv venv --no-setuptools
> venv\Scripts\activate
> Invoke-WebRequest https://raw.github.com/pypa/pip/master/contrib/get-pip.py -Ou
tFile .\get-pip.py
> python get-pip.py
> pip install --upgrade pip
> pip install requests pyodbc pyparsing
> Invoke-WebRequest https://github.com/Oomnitza/oomnitza-connector/archive/master
.zip -OutFile connector-master.zip
> Add-Type -A System.IO.Compression.FileSystem
> [IO.Compression.ZipFile]::ExtractToDirectory('c:\oomnitza-connector\connector-m
aster.zip', 'c:\oomnitza-connector\')
> cd oomnitza-connector-master
```

Connector Configs

Now you should be able to generate a default config file. Running `python connector.py generate-ini` will regenerate the config.ini file, and create a backup if the file already exists. When you edit this file, it will have one section per connection. You can safely remove the section for the connections you will not be using to keep the file small and manageable. An example generated config.ini follows.

```
[oomnitza]
url = https://example.oomnitza.com
api_token =
```

```
api_token =
username = oomnitza-sa
password = ThePassword

[airwatch]
enable = False
url = https://apidev.awmdm.com
username = username@example.com
password = change-me
api_token = YOUR AirWatch API TOKEN
sync_field = 24DCF85294E411E38A52066B556BA4EE

[bamboohr]
enable = False
url = https://api.bamboohr.com/api/gateway.php
system_name = YOUR BambooHR SYSTEM NAME
api_token = YOUR BambooHR API TOKEN
default_role = 25

[casper]
enable = False
url = https://jss.jamfcloud.com/example
username = username@example.com
password = change-me
sync_field = 24DCF85294E411E38A52066B556BA4EE
sync_type = computers
update_only = False

[jasper]
enable = False
wsdl_path = http://api.jasperwireless.com/ws/schema/Terminal.wsdl
username = username@example.com
password = change-me
storage = storage.db
api_token = YOUR Jasper API TOKEN
sync_field = 24DCF85294E411E38A52066B556BA4EE
```

```
sync_field = 24DCF85294E411E38A52066B556BA4EE
update_only = False

[ldap]
enable = False
url = ldap://ldap.forumsys.com:389
username = cn=read-only-admin,dc=example,dc=com
password =
base_dn = dc=example,dc=com
protocol_version = 3
filter = (objectClass=*)
default_role = 25
default_position = Employee

[mobileiron]
enable = False
url = https://na1.mobileiron.com
username = username@example.com
password = change-me
partitions = ["Drivers"]
sync_field = 24DCF85294E411E38A52066B556BA4EE

[okta]
enable = False
url = https://example-admin.okta.com
api_token = YOUR Okta API TOKEN
default_role = 25
default_position = Employee

[onelogin]
enable = False
url = https://app.onelogin.com/api/v2/users.xml
api_token = YOUR OneLogin API TOKEN
default_role = 25
default_position = Employee
```

```
[sccm]
enable = False
server = server.example.com
database = CM_DCT
username = change-me
password = change-me
authentication = SQL Server
sync_field = 24DCF85294E411E38A52066B556BA4EE

[zendesk]
enable = False
system_name = oomnitza
api_token = YOUR Zendesk API TOKEN
username = username@example.com
default_role = 25
default_position = Employee
```

The `[oomnitza]` section is where you configure the connector with the URL and login credentials for connecting to Oomnitza. You can use an existing user's credentials for username and password, but best practice is to create a service account using your standard naming convention. (See the (documentation)[http://docs) for managing user accounts in Oomnitza.)

The remaining sections each deal with a single connection to an external service. The "enable" field is common to all connections and if set to "True" will enable this service for processing. Some fields are common to a type of connection. For example, "default_role" and "default_user" are fields for connections dealing with loading People into the Oomnitza app.

Each section can end with a list of field mappings. Simple mappings which just copy a field from the external system to a field inside Oomnitza can be defined here or in the System Settings within Oomnitza. Simple mappings are as follows:

```
mapping.[Oomnitza Field] = {"source": "[external field]"}
```

For fields which require processing before being brought into Oomnitza must be defined in the

INI. These mappings are more involved. Please contact support@oomnitza.com for more information. The format is:

```
mapping.[Oomnitza Field] = {"source": "[external field]", "converter": "[converte
r name]"}
```

Oomnitza Configuration

`url` : the url of the Oomnitza application. For example: `https://example.oomnitza.com`

`username` : the Oomnitza username to use

`password` : the Oomnitza password to use

`api_token` : The API Token belonging to the Oomnitza user. If provided, `password` must be left blank.

Airwatch Configuration

`url` : the url of the Airwatch server

`username` : the Airwatch username to use

`password` : the Airwatch password to use

`api_token` : HPDL

`sync_field` : The Oomnitza field which contains the asset's unique identifier (we typically recommend serial number).

Default Field Mappings

```
To Be Determined
```

BambooHR Configuration

`url` : the url of the BambooHR server

`system_name` : hpdl

`api_token` : hpdl

`default_role` = 25

Default Field Mappings

```
mapping.USER =            {'source': "workEmail"}
mapping.FIRST_NAME' =     {'source': "firstName"}
mapping.LAST_NAME' =      {'source': "lastName"}
mapping.EMAIL' =          {'source': "workEmail"}
mapping.PHONE' =          {'source': "mobilePhone"}
mapping.POSITION' =       {'source': "jobTitle"}
mapping.PERMISSIONS_ID' = {'setting': "default_role"}
```

Casper Configuration

The `[casper]` section contains a similar set of preferences; your JSS URL, and the login credentials for an auditor account in Casper (See the Casper Suite Administrator's Guide , pg. 42).

The identifier section of the config.ini file should contain a mapping to a unique field in Oomnitza, which you want to use as the identifier for an asset. Serial Number is the most commonly used identifier since no two assets should share one. This will determine if the script creates a new record for a given serial number on its next sync, or if it updates an existing record that has new information.

`url` : the url of the Casper server

`username` : the Casper username to use

`password` : the Casper password to use

`sync_field`: The Oomnitza field which contains the asset's unique identifier (we typically recommend serial number).

`sync_type`: Sets the type of data to pull from Casper. Options are `computers` or `mobiledevices`. When syncing mobile devices a second section should be added to your config.ini file named `[Casper.MDM]` and this value should be set to `mobiledevices`.

`verify_ssl`: set to false if the Casper server is running with a self signed SSL certificate.

`update_only`: set this to True to only update records in Oomnitza. Records for new assets will not be created.

Default Field Mappings

```
To Be Determined
```

Jasper Configuration

`wsdl_path`: The full URL to the Terminal.wsdl. Defaults to: http://api.jasperwireless.com/ws/schema/Terminal.wsdl.

`username`: the Jasper username to use

`password`: the Jasper password to use

`storage`: The path to the storage file used to maintain state about the connector. Defaults to: `storage.db`

`api_token`: The Jasper API Token.

`sync_field`: The Oomnitza field which contains the asset's unique identifier.

`update_only`: set this to True to only update records in Oomnitza. Records for new assets will not be created.

Default Field Mappings

```
To Be Determined
```

LDAP Configuration

`url` : The full URI for the LDAP server. For example: `ldap://ldap.forumsys.com:389`

`username` : the LDAP username to use. Can be a DN, such as `cn=read-only-admin,dc=example,dc=com` .

`password` : the LDAP password to use

`base_dn` : The Base DN to use for the connection.

`protocol_version` : The LDAP Protocol version to use. Defaults to: 3.

`filter` : The LDAP filter to use when querying for people. For example: `(objectClass=*)`

`default_role` : The numeric ID of the role which will be assigned to imported users. For example: `25` .

`default_position` : The position which will be assigned to the user. For example: `Employee` .

MobileIron Configuration

`url` : The full URI for the MobileIron server. For example: `https://na1.mobileiron.com`

`username` : the MobileIron username to use.

`password` : the MobileIron password to use.

`partitions` : The MobileIron partitions to load. For example: `["Drivers"]` or `["PartOne", "PartTwo"]`

`sync_field`: The Oomnitza field which contains the asset's unique identifier.

Default Field Mappings

```
To Be Determined
```

Okta Configuration

`url`: The full URI for the Okta server. For example: `https://oomnitza-admin.okta.com`

`api_token`: The Jasper API Token.

`default_role`: The numeric ID of the role which will be assigned to imported users. For example: `25`.

`default_position`: The position which will be assigned to the user. For example: `Employee`.

Default Field Mappings

```
mapping.USER =           {'source': "login"},
mapping.FIRST_NAME =     {'source': "firstName"},
mapping.LAST_NAME =      {'source': "lastName"},
mapping.EMAIL =          {'source': "email"},
mapping.PHONE =          {'source': "mobilePhone"},
mapping.PERMISSIONS_ID = {'setting': "default_role"},
mapping.POSITION =       {'setting': "default_position"},
```

OneLogin Configuration

`url`: The full URI for the OneLogin server. For example: `https://app.onelogin.com/api/v2/users.xml`

`api_token`: The OneLogin API Token.

`default_role`: The numeric ID of the role which will be assigned to imported users. For

example: `25`.

`default_position`: The position which will be assigned to the user. For example: `Employee`.

Defualt Field Mappings

```
mapping.USER =           {'source': "username"}
mapping.FIRST_NAME =     {'source': "firstname"}
mapping.LAST_NAME =      {'source': "lastname"}
mapping.EMAIL =          {'source': "email"}
mapping.PHONE =          {'source': "phone"}
mapping.PERMISSIONS_ID = {'setting': "default_role"}
mapping.POSITION =       {'setting': "default_position"}
```

SCCM Configuration

Note: The SCCM connector currently requires a Windows host. While it should be possible to run the connector on a non-Windows host, such as Linux, we do not provide support for this configuration at this time.

`server`: The server hosting the SCCM database.

`database`: The SCCM database from which to pull data.

`username`: The username to use when connecting to the server.

`password`: The password to use when connecting to the server.

`authentication`: Sets the type of authentication to use when connecting to the server. Options are `SQL Server` or `Windows`. The default is to use SQL Server Authentication.

`sync_field`: The Oomnitza field which contains the asset's unique identifier (we typically recommend serial number).

Default Field Mappings

```
TO Be Determined
```

Zendesk Configuration

`system_name`: The Zendesk system name to use. For example: `oomnitza`

`api_token`: The Zendesk API Token.

`username`: the Zendesk username to use.

`default_role`: The numeric ID of the role which will be assigned to imported users. For example: `25`.

`default_position`: The position which will be assigned to the user. For example: `Employee`.

Default Field Mappings

```
mapping.USER =           {'source': "email"}
mapping.FIRST_NAME =     {'source': "name", 'converter': "first_from_full"}
mapping.LAST_NAME =      {'source': "name", 'converter': "last_from_full"}
mapping.EMAIL =          {'source': "email"}
mapping.PHONE =          {'source': "phone"}
mapping.PERMISSIONS_ID = {'setting': "default_role"}
mapping.POSITION =       {'setting': "default_position"}
```

Running the connector

The connector is meant to be run from the command line and as such as multiple command line options:

```
$ python connector.py
usage: connector.py [-h] [--show-mappings] [--testmode] [--save-data]
                    [--ini INI] [--logging-config LOGGING_CONFIG]
                    [--record-count RECORD_COUNT]
                    {upload,generate-ini,gui} [connectors [connectors ...]]
connector.py: error: too few arguments
(connector)daniels-mbp:Connector daniel$ python connector.py --help
usage: connector.py [-h] [--show-mappings] [--testmode] [--save-data]
                    [--ini INI] [--logging-config LOGGING_CONFIG]
                    [--record-count RECORD_COUNT]
                    {upload,generate-ini,gui} [connectors [connectors ...]]

positional arguments:
  {upload,generate-ini,gui}
                        Action to perform.
  connectors            Connectors to run.

optional arguments:
  -h, --help            show this help message and exit
  --show-mappings       Show the mappings which would be used by the
                        connector.
  --testmode            Run connectors in test mode.
  --save-data           Saves the data loaded from other system.
  --ini INI             Config file to use.
  --logging-config LOGGING_CONFIG
                        Use to override logging config file to use.
  --record-count RECORD_COUNT
                        Number of records to pull and process from connection.
```

The available actions are:

- `gui` (default): launch the config gui.
- `generate-ini`: generate an example config.ini file.
- `upload`: uploads the data from the indicated connectors to Oomnitza. The connector

values are taken from the section names in the ini file.

`--ini` is used to specify which config file to load, if not provided, `config.ini` from the root directory will be used. This option can be used with the `generate-ini` action to specify the file to generate.

`--logging-config` is used to specify an alternate logging config file.

`--show-mappings` is used to print out the loaded mappings. These mappings can be a combination of the built-in mappings, config.ini mappings, and mappings setup via the website.

`--testmode` will print out the records which would have been sent rather than pushing the data to the server. This can be used to see what, exactly, is getting sent to the server.

`--record-count` is used to limit the number of records to process. Once this number have been processed, the connector will exit. This can be used with `--testmode` to print out a limited number of records then exit cleanly.

`--save-data` is used to save the data loaded from the remote system to disk. These files can then be used to confirm the data is being loaded and mapped as expected.

Setting the connector to run as an automated task

There are many ways to automate the sync, here are a few:

- OS X: http://www.maclife.com/article/columns/terminal_101_creating_cron_jobs
- OS X: http://superuser.com/questions/126907/how-can-i-get-a-script-to-run-every-day-on-mac-os-x
- OS X: http://launched.zerowidth.com/
- Linux: http://www.cyberciti.biz/faq/how-do-i-add-jobs-to-cron-under-linux-or-unix-oses/
- Windows: http://bytes.com/topic/python/answers/32605-windows-xp-cron-scheduler-python

Advanced usage

Logging

The Oomnitza Connector uses the standard python `logging` module. This modules is configured via the `logging.json` file. This file can be edited, or copied to a new file, to change the logging behavior of the connector. Please see the python docs for information of configuring python logging.

SSL Protocol Version

If the service to be connected to requires a particular SSL protocol version to properly connect, the connection's section in the ini file can include a `ssl_protocol` option. The value can be one of: `ssl`, `sslv23`, `sslv3`, `tls`, `tls1`.

Record Filtering

Support has been added for filtering the records passed from the connector to Oomnitza. By default, all records from the remote system will be sent to Oomnitza for processing. To limit the records based on values in those records, a special `recordfilter` value can be added to a connector section in the ini file. This filter is written using the Python programming language.

For example, the following filter will only process records with the `asset_type` field set to "`computer`":

```
recordfilter:
    return record.asset_type == "computer"
```

This is a very new feature, with many options, and we are still working on the documentation. If you are interested in using this feature, please contact support@oomnitza.com for assistance.

The GUI

This section is under construction