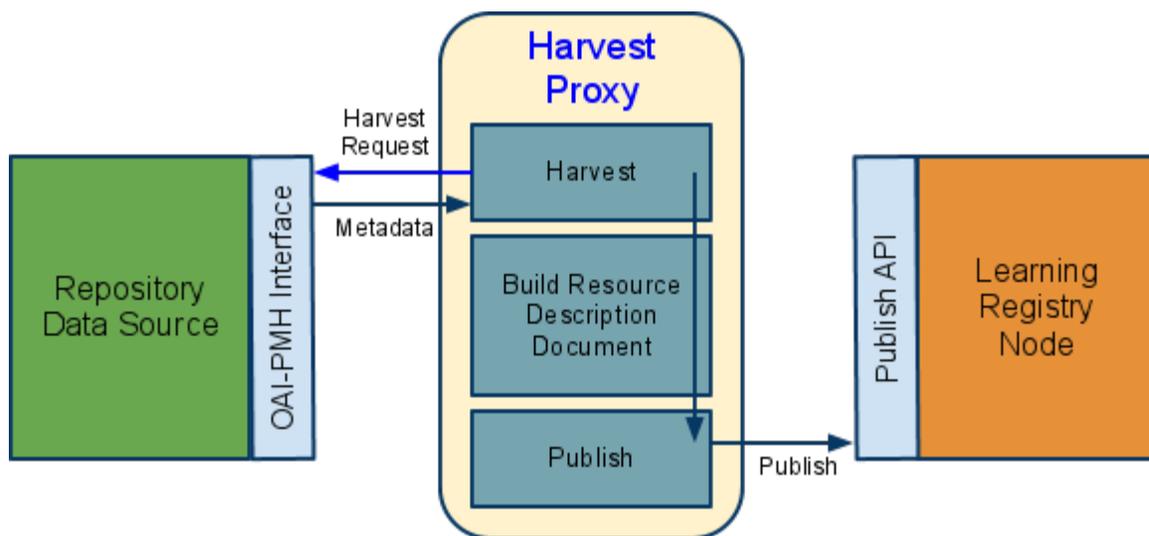


OAI-PMH to Learning Registry Publish Utility

If you have a repository with an OAI-PMH compliant interface, this utility will allow you to connect your repository with the Learning Registry to publish your metadata to the Learning Registry network.

Harvesting Intermediary

Content repositories may have an OAI-PMH interface that can be used to harvest their content. The Learning Registry does not provide a built-in API to harvest metadata from an OAI-PMH interface and publish it to a Learning Registry node. In the Learning Registry model a data source publishes (pushes) data into the network, rather than the Learning Registry harvesting (pulling) data from data sources. We want the data owner to control their data distribution, frequency of publishing, etc.. Rather than changing existing data sources that have an OAI-PMH interface to publish directly to the Learning Registry, intermediaries (publishing proxies) may be used, as illustrated.



The Learning Registry provides a simple utility described herein that can be used as such an intermediary. This document describes how to install, configure and use this utility. It lets you to connect an OAI-PMH interface to the publish API on any Learning Registry node. As the operator of the utility you control when the data is harvested. The utility creates a JSON formatted Learning Registry resource data description document with an inline payload string representing the harvested XML metadata for each record in the OAI-PMH results. Once the utility code is installed, all you need to do is set a few parameters to run it.

As provided, the utility can harvest a Dublin Core metadata dissemination from an OAI-PMH target, but the code can be adapted to harvest any metadata format that an OAI-PMH target is capable of disseminating.

NB: If you are harvesting a repository that you do not own, you should comply with all access conditions

and policies of the repository. Whomever operates the utility is considered to be the data provider.

Use of this utility is not a prerequisite to harvesting from an OAI-PMH target and publishing the data to a Learning Registry. Any software that implements the OAI-PMH harvesting interfaces and the Learning Registry publishing APIs may be used (or developed). For example, a more elaborate OAI-PMH harvesting interface with the Learning Registry is available from:

<https://github.com/LearningRegistry/LearningRegistry/blob/Sprint3Release/data-pumps/oai-pmh-data-pump.py>

The utility this document describes is a bit more basic and focused on explaining the basic concepts.

Harvest and Publish Process Overview

To setup and run the utility (which has been developed in Python), follow these steps (they are expanded below):

1. Install and configure the Python environment
2. Get the utility code -- contained in this document
3. Configure the utility code to connect the OAI-PMH interface to the Learning Registry node
4. Harvest and publish

1. Install and Configure Python Environment

1.1. Windows Instructions

1.1.1. Install Python

Download and install *Python 2.7.2* (later versions should also work but the following procedures have not been tested) from <http://python.org/download/>. You must install the 32 bit version (the required *setuptools* utility will not install with the 64 bit version). Launch the installer after download and follow the prompts. You do not need to install TCL/TK, the documentation, utility scripts or test suite.

Once installed, on the Windows *Start Menu* you will find the *Python 2.7* folder and a *Python (Command Line)* entry, Launch the command line interface to test in the install. Enter CTRL-Z followed by ENTER to exit the Python interpreter. Alternatively, you may open a *Command Prompt* window, go to the installation directory, usually C:\python27, and enter “python” to launch the Python interpreter.

1.1.2. Install Python Package Tools

Download and install the *setuptools 0.6c.11* utility to manage Python packages from <http://pypi.python.org/pypi/setuptools>. Select the Windows installer .exe file for Python 2.7, e.g., *setuptools-0.6c11.win32-py2.7.exe*. Launch the installer after download and follow the prompts.

1.1.3. Install Python OAI-PMH Harvester

Download and install *pyoai 2.4.4*, the Python OAI-PMH module from <http://pypi.python.org/pypi/pyoai>. You will need a utility such as 7-Zip (<http://www.7-zip.org/>) to extract the source code from the pyoai-

2.4.4.tar.gz file that you downloaded. Launch this utility and extract the Python source code. E.g. for 7-Zip:

- Download and install the 7-Zip installer
- Double click on the downloaded pyoai-2.4.4.tar.gz to launch 7-Zip
- Select pyoai-2.4.4.tar from the file pane and then click *File/Open*
- The files will be opened and the name will change to pyoai-2.4.4
- Drag and drop the pyoai-2.4.4 entry to a destination directory

Open a command line interface and navigate to the directory you extracted pyoai to. Run the following command: `python setup.py install`. If you have any trouble ensure that the directory you installed python to (C:\python27) is added to the system path and repeat the above steps.

1.2. Linux Instructions

1.2.1. Install Python

From a *Command Window*, enter `sudo apt-get install python`

1.2.2. Install Python Tools

From a *Command Window*, enter `sudo apt-get install python-setuptools`.

1.2.3. Install Python OAI-PMH Harvester

Download and install pyoai 2.4.4, the Python OAI-PMH module from <http://pypi.python.org/pypi/pyoai>. Unzip and extract the source files from the tarball pyoai-2.4.4.tar.gz.

Navigate to where you extracted the source files and find `setup.py`. From a *Command Window*, launch Python and run the setup file.

2. Download Harvesting Intermediary Utility Code

A copy of the code is included at the [end of this document](#). If you copy the code from this document, save it with the file name LR-harvest-and-publish.py.

3. Configure Harvesting Intermediary Utility Code

The harvesting code must be configured to connect to the repository to be harvested and the Learning Registry node where the data will be published. There are several values in the code that must be set. Open the code in your favorite text editor and provide values for the items below (these values shown in **reversed type** in the [code listing](#)).

3.1. Set Submitter Value

<<submitter>>: Provide the identity of the submitter persona (person or organization).

The identity value could be the email address of the submitter or the name (or URL) of the repository. *NB*: The script marks the submitter as being a *user*. If you are submitting on behalf of the repository (i.e., you are the repository manager), you should use the identity of the repository as the submitter, and change the value of `submitter_type` from `user` to `agent` (.).

3.2. Set Publish Service URL

<<Node Publish Service URL>>: Provide the URL of the Learning Registry node where the data will be published. You will need to have access rights to publish to this node.

To publish to the test network, use <http://lrdev1.learningregistry.org/publish> as the <<Node Publish Service URL>>

3.3. Set OAI-PMH Target URL

<<OAI-PMH Target URL>>: Provide the URL of the OAI-PMH interface of the repository.

The value for <<OAI-PMH Target URL>> should be the base URL to the OAI-PMH interface (e.g., “<http://example.org/fedora/oai>” rather than “http://10.100.30.170/fedora/oai?verb=ListRecords&metadataPrefix=oai_dc”).

3.4. Set Signing Values

This is a placeholder section for inclusion in a future version. Please ignore it.

Data submitted to the Learning Registry should include a digital signature, to insure that the submitter controls the identity under which the data is being submitted. If you are not signing your messages, delete the the signing portion of the script as indicated in the script. To sign the message you need to have a OpenPGP key pair and you need to publish your public key to one or more key servers. Details on generating and publishing a key and signing documents are available in [Signing Learning Registry Documents](#).

- <<signature>>: Document signature
- <<keyserver>>: Provide the location of the key servers where the signer’s public key can be found. Multiple values may be provided as a comma separated list of strings, e.g.,: [“<<keyserver1>>”, “<<keyserver2>>”, “<<keyserver3>>”]
- <<keyowner>>: Provide the identity (email address) of the owner of the key used to sign the message. If the <<keyowner>> value is the same as the <<submitter>> value , the <<keyowner>> value is not needed and the key-value pair can be removed (delete the entire line AND the comma at the end of the previous line).

3.5. Set Terms of Service

All information published in the Learning Registry must have an associated *Terms of Service* (ToS); an agreement covering the use of the provided data. See <http://www.learningregistry.org/tos/> for more details about the requirements for the ToS.

The script has a default ToS for the metadata you are publishing. It states,, in part, that the metadata is being released under the Creative Commons public domain dedication, CC0. The ToS is available at: <http://www.learningregistry.org/tos/cc0/v0-5/>

If you want to publish your metadadata under a different ToS, edit the value of the submission_TOS to be the URL of the ToS you are using. The list of available ToS for the public Learning Registry Network is

available at: <http://www.learningregistry.org/tos/>

4. Harvest and Publish

Once the script has been edited, you are ready to harvest and publish your data. You may want to test the process before publishing the data to the production Learning Registry network.

The procedure does a one time harvest of all Dublin Core metadata disseminations from the repository.

4.1. Publish to the Test Environment

To test your harvesting and settings, you may want to publish your data to the test network. Currently the test network will not validate your submitted identity, digital signature or specified terms of service. Your data may be deleted from the test node at any time.

To publish to the test network, use `http://lrdev1.learningregistry.org/publish` as the <<Node Publish Service URL>>

And run the following command: `python LR-harvest-and-publish.py`

NB: On linux machines, you will first need to make the script executable running `chmod +x LR-harvest-and-publish.py`.

4.2. Validate the Results

The script will output the document IDs that are published to the node in a file called `output.log` located in the same directory from which the `LR-harvest-and-publish.py` script is run. You can verify that the harvested records were published to the Learning Registry node by following the procedures below.

4.2.1. Open the output.log File

The `output.log` file will be created after the `LR-harvest-and-publish.py` script is run and will be located in the same directory from which the `LR-harvest-and-publish.py` script is run. The document ID for each harvested and published record is represented in the form of a JSON document like the one below:

```
{"doc_ID": "885205427ba14d6683a79edda0b807e1", "OK": true}
```

4.2.2. Select a doc_ID to Validate

Find a document id <doc_ID> in the `output.log` file.

4.2.3. Harvest the doc_ID from the LR Node

Assuming that you published your data to node `http://lrdev1.learningregistry.org/`, then in your browser, visit:

```
http://lrdev05.learningregistry.org/harvest/getrecord?by_doc_ID=True&request_id=<doc_ID>
```

4.3. Publish to the Production Learning Registry Network

You should only publish to the production Learning Registry network when you are sure that you are ready to make the harvested data available for distribution by the Learning Registry network. Once you publish the data, you cannot delete it from the network.

To publish to a production LR node, replace the <<Node Publish Service URL>> value in the script with the URL to the publish service of a production node and run the modified script again.

Bonus Section: Periodic Harvesting and Publishing

As noted, the script harvests all metadata. If you want to perform incremental harvesting (i.e., harvest metadata added to the repository between two dates), you need to use the incremental harvesting code below. You need to provide two additional values:

- <<harvest start date>>: Provide the start date for harvest
- <<harvest end date>>: Provide the end date for harvest

Alternatively, if you wish to harvest all data until a specific time, or all data from a specific time, delete the appropriate arguments from the code (highlighted in blue).

If you are performing repeated incremental harvesting to obtain new data from the repository, you must maintain the record of when you harvests and publish and update the script to change the dates for each new incremental harvest.

Harvesting Intermediary Utility Code: Base Code

The following code will harvest all OAI-PMH Public Core metadata from a specified OAI-PMH target and publish the data to a specified node in the Learning Registry. Values need to be provided for text in **reversed type**. Do not use the lines that have been redacted -- they are placeholders for a future version.

```
# OAI-PMH to Learning Registry Publish Utility
# Version 1.0 20110914
#
# Copyright 2011 US Advanced Distributed Learning Initiative
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
```

```

# Change Log
# V 1.0 -- initial public version

import urllib2
import json
from oaipmh.client import Client
from oaipmh.metadata import MetadataRegistry, oai_dc_reader
def convert_to_envelope(doc, rawMetadata):
    #add code here to create the document from the Dublin Core metadata
    doc = {
        "doc_type":          'resource_data',
        "doc_version":       "0.23.0",
        "active":            True,
        "resource_data_type": "metadata",
        "identity":{
            "submitter_type": "user",
            "submitter":      "wegrata",
            "curator":        "wegrata",
            "owner":          "wegrata",
        },
        "TOS": {
            "submission_TOS": "http://
www.learningregistry.org/tos/cc0/v0-5/"
        },
# -- signature values -- delete this block if not signing
# "digital_signature": {
#     "signature": "<<signature>>",
#     "key_server": ["<<keyserver>>"],
#     "key_owner": "<<keyowner>>"
# },
# -- end of signature values -- end of block to delete if not signing
        "resource_locator": 'location',
        "keys":              ["DC",],
        "payload_placement": "inline",
        "payload_schema":    ["oai_dc"],
        "resource_data":     rawMetadata,
        "publishing_node":   'local',
    }
    return doc

def acquire_and_publish_documents(oai_url, publish_url, reader, prefix):
    registry = MetadataRegistry()
    registry.registerReader(prefix, reader)
    client = Client(oai_url, registry)
    documents = []
    count = 0
    for record in client.listRecords(metadataPrefix=prefix):
        header = record[0]

```

```

        metadata = record[1]
        rawMetadata = urllib2.urlopen("{0}?
verb=GetRecord&metadataPrefix={1}
&identifier={2}".format(oai_url,prefix,header.identifier())).read()
        value = convert_to_envelope(metadata,rawMetadata)
        print(dir(header))
        if value != None:
            documents.append(value)
            count += 1
            if count % 10 == 0:
                publish_documents(publish_url,documents)
                documents = []
        publish_documents(publish_url,documents)

def publish_documents(publish_url,documents):
    data = {'documents':documents}
    headers = {"Content-Type":"application/json"}
    req = urllib2.Request(publish_url, json.dumps(data),headers)
    with open("output.log","a") as f:
        f.write(urllib2.urlopen(req).read())

def main():
    publish_url = '<<Node Publish Service URL>>'
    oai_url = '<<OAI-PMH Target URL>>'
    acquire_and_publish_documents(oai_url,publish_url,oai_dc_reader,'oai_dc')

if __name__ == '__main__':
    main()

```

Harvesting Intermediary Utility Code: Incremental Harvest

The following code will incrementally harvest OAI-PMH Dublic Core metadata from a specified OAI-PMH target and publish the data to a specified node in the Learning Registry. Values need to be provided for text in **reversed type**. Do not use the lines that have been redacted -- they are placeholders for a future version.

```

# OAI-PMH to Learning Registry Publish Utility
# Version 1.0 20110914
#
# Copyright 2011 US Advanced Distributed Learning Initiative
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0

```

```

# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
# Change Log
# V 1.0 -- initial public version

import urllib2
import json
from oaipmh.client import Client
from oaipmh.metadata import MetadataRegistry, oai_dc_reader
def convert_to_envelope(doc, rawMetadata):
    #add code here to create the document from the Dublin Core metadata
    doc = {
        "doc_type":          'resource_data',
        "doc_version":       "0.23.0",
        "active":            True,
        "resource_data_type": "metadata",
        "identity":{
            "submitter_type":    "user",
            "submitter":        "wegrata",
            "curator":          "wegrata",
            "owner":            "wegrata",
        },
        "TOS": {
            "submission_TOS":    "http://
www.learningregistry.org/tos/cc0/v0-5/"
        },
# -- signature values -- delete this block if not signing
#     "digital_signature": {
#         "signature":    "<<signature>>",
#         "key_server":   ["<<keyserver>>"],
#         "key_owner":    "<<keyowner>>"
#     },
# -- end of signature values -- end of block to delete if not signing
        "resource_locator":  'location',
        "keys":              ["DC",],
        "payload_placement": "inline",
        "payload_schema":    ["oai_dc"],
        "resource_data":     rawMetadata,
        "publishing_node":   'local',
    }
    return doc

def acquire_and_publish_documents(oai_url, publish_url, reader, prefix):

```

```

registry = MetadataRegistry()
registry.registerReader(prefix, reader)
client = Client(oai_url, registry)
documents = []
count = 0
for record in
client.listRecords(metadataPrefix=prefix, from_=start, until=end):
    header = record[0]
    metadata = record[1]
    rawMetadata = urllib2.urlopen("{0}?
verb=GetRecord&metadataPrefix={1}
&identifier={2}".format(oai_url,prefix,header.identifier())).read()
    value = convert_to_envelope(metadata,rawMetadata)
    print(dir(header))
    if value != None:
        documents.append(value)
        count += 1
        if count % 10 == 0:
            publish_documents(publish_url,documents)
            documents = []
    publish_documents(publish_url,documents)

def publish_documents(publish_url,documents):
    data = {'documents':documents}
    headers = {"Content-Type":"application/json"}
    req = urllib2.Request(publish_url, json.dumps(data),headers)
    with open("output.log","a") as f:
        f.write(urllib2.urlopen(req).read())

def main():
    publish_url = '<<Node Publish Service URL>>'
    oai_url = '<<OAI-PMH Target URL>>'
    start_date = datetime.datetime(2011,9,10)
    end_date = datetime.datetime(2011,9,14)
    acquire_and_publish_documents(oai_url,publish_url,oai_dc_reader,'oai_dc')

if __name__ == '__main__':
    main()

```

Change Log

Version	Date	Description
1.0	201105xx	Initial version. Still in draft
1.x	201105xx	Include signing documents

1.2	20110914	Updated code listings, reformatted document to include numbered headings for steps, fixed minor typos, and updated content to reflect what the code does.
-----	----------	---