

NLA Party Identifier Application

System Administrator's Manual

Creative Commons License

This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Australia License](http://creativecommons.org/licenses/by-nc-sa/3.0/au/deed.en).

Contents

[Overview 3](#_Toc354734864)

[License 3](#_Toc354734865)

[Acknowledgement 3](#_Toc354734866)

[Components 3](#_Toc354734867)

[Architecture 4](#_Toc354734868)

[Prerequisites 4](#_Toc354734869)

[Softwares: 4](#_Toc354734870)

[OAI-PMH -2 4](#_Toc354734871)

[Configuration for OAI-PMH Data Provider 4](#_Toc354734872)

[Database Design 6](#_Toc354734873)

[SQL Scripts for OAI-PMH Data Provider 6](#_Toc354734874)

# Overview

This document lists and explains the steps required to deploy and maintain an instance of NLA Party Identifier application

# License

The Use of NLA Application identifier is governed by the GNU GPL3 license.

# Acknowledgement

This project is supported by the Australian National Data Service (ANDS). ANDS is supported by the Australian Government through the National Collaborative Research Infrastructure Strategy Program and the Education Investment Fund (EIF) Super Science Initiative.

# Components

The NLA Party Identifier Application has three main components:

1. **The NLA Party Identifiers Web Application**  
   This is the major component of the system which has these capabilities:
2. Display and store the Internal staffs personal information and allow users to modify or insert their personal data.
3. Allow user to view other Internal ANU researchers’ personal information

c. Display *research topics* of each ANU researcher and provides relevant information of each topic.

d. Insert new publication records into the system using *bibtex* formatted data.

1. **The OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting**) **Application**  
   This component enables communication between an external system (for e.g:Trove) with the NLA Party Identifier Web Application via OAI-PMH. The component is a *data provider*  and repository that exposes structured metadata via OAI-PMH and that the *Service Provider* is able to make OAi-PMH service request to harvest the metadata.
2. **The Data Harvester**

This component enables data harvesting from various sources including *The Metadata Store Java Service* and The NLA’s  *Trove*  application. This data harvester component is developed using unix shell scripts and Java program which run regularly to update and store any new data from the source systems into the MySQL Database. The harvested metadata includes the following:

1. Person’s personal Information such as : address, email, fax, phone number,etc.
2. Publication records
3. Grants records
4. NLA ID from trove

# Architecture



# Prerequisites

## Softwares:

1. Operating System : The application has been running on a *linux red hat* operating system.
2. PHP version 5 or later: for detailed instructions please see the following:

<http://php.net/manual/en/install.php>

1. Apache Web Server version 2 or later: for detailed installation instructions please see the following:

<http://httpd.apache.org/docs/2.2/platform/windows.html>

1. MYSQL version 5 or later: please see the following:

<http://dev.mysql.com/doc/refman/5.5/en/installing.html>

1. PDO (PHP Data Objects) :

<http://php.net/manual/en/book.pdo.php>

## 

## *OAI-PMH -2* Software

**Description:**

This open source software is an implementation for an OAI-PMH 2.0 Provider written in PHP. This implementation completely complies to OAI-PMH 2.0, including the support of on-the-fly output compression which may significantly reduce the amount of data being transferred.

This package has been inspired by PHP OAI Data Provider developed by Heinrich Stamerjohanns at University of Oldenburg. Some of the functions and algorithms used in this code were transplanted from his implementation at <http://physnet.uni-oldenburg.de/oai/>.

**Installation Steps :**

1. Download the PHP files in : <http://code.google.com/p/oai-pmh-2/>
2. Extract all the files.
3. Store the files in these locations:
4. Linux/UNIX OS: store the PHP files in the /var/www/html/oai

Allow your webserver to write to the token directory

1. Windows OS: store the PHP files in the htdocs directory.
2. Check your oai site through a web browser: http://localhost/oai/

## The files of the OAI-PMH Data Provider

The application consists of these following PHP files:

|  |  |  |
| --- | --- | --- |
| **No** | **File name** | **Description** |
| 1. | identify.php | identifies the data provider. Responses to Identify |
| 2. | listmetadataformats.php | lists supported metadata formats, e.g. dc or rif-cs. Responses to ListMetadataFormats. |
| 3. | listsets.php | lists supported sets, e.g. Activity, Collection or Party. Responses to List Sets. |
| 4. | listrecords.php | lists a group of records without details. Responses to ListRecords. It also serves to ListIdentifiers which only returns identifiers. getrecord.php: gets an individual record. Responses to GetRecord. |
| 5. | Utility classes | xml\_creater.php which includes classess ANDS\_XML, ANDS\_Error\_XML, ANDS\_Response\_XML |
| 6. | oaidp-util.php | Support to different metadataformats in your own systems. Two examples provided with the package are: record\_dc.php and record\_rif.php. They are helpers and need information from the real records. They need to be devleoped for your particular system. |
| 7. | oaidp-config.php | Configurations |

The configuration file named **oaidp-config.php** needs to be edited accordingly:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Configuration file** :   |  |  | | --- | --- | | **Attribute Name** | **Value** | | *Repository Name:* | $identifyResponse["repositoryName"] = 'Australian National University'; | | *Admin Email:* | $adminEmail = array(‘krisna.irwan@gmail.com'); | | *EarliestDateStamp:* | $identifyResponse["earliestDatestamp"] = '2000-01-01'; | | *Deleted record* | $identifyResponse["deletedRecord"] = 'no'; | | *repository Identifier* | $repositoryIdentifier = 'anu.edu.au'; | | *oai identifier* | $SQL['identifier'] = 'oai\_identifier'; | | *Oai metadata prefix* | $SQL['metadataPrefix'] = 'oai\_metadataprefix'; | | *ID Prefix* | $idPrefix = ''; | |  | $oaiprefix = "";  $SQL['datestamp'] = 'datestamp'; | |  | $SQL['deleted'] = 'delete // change according to your local | | |  | | **The MySQL Database setup**:  Variables to be set accordingly:   |  |  |  | | --- | --- | --- | | **No.** | **Variable** | **Description** | | 1. | $DB\_HOST | Hostname | | 2. | $DB\_USER | User name | | 3. | $DB\_NAME | ‘oaidb’ | | 4. | $SQL['set'] | `oai\_set` | | 5. | $METADATAFORMATS | The metadataformats such as: **rif-cs** or **oai-dc** | | 6. | $DSN | Domain Source Name for the MySQL. mysql://$DB\_USER:$DB\_PASSWD@$DB\_HOST/$DB\_NAME" |  Database Design | | NLA Party Identifier System Core Database Schema   |  |  |  | | --- | --- | --- | | **No.** | **Table name** | **Description of the table** | | 1. | oai\_records | stores the oai header records of each person metadata. | | 2. | useraccount | stores the detailed information | | 3. | pub\_to\_authors | stores the linking between the publication and authors | | 4. | publication | stores the detailed information of each publication | | 5 | grant\_detail | stores the detailed information of each grant | |  Data HarvesterPrerequisites:  * Operating System: Linux Red Hat. * Java     The Data Harvester has four components:   |  |  |  | | --- | --- | --- | | **No** | **Name** | **Description** | | 1. | People Data Harvester | * Retrieve the people information from the Java Service * Update to the System Database | | 2. | Publication Data Harvester | * Retrieve publication information * Update to the System Database | | 3. | Grant Data Harvester | * Retrieve grant information * Update to the System Database | | 4. | NLA ID Harvester | * Retrieve NLA ID information from the Trove |  SQL Scripts for OAI-PMH Data Provider Create table:  **oai\_records –** the table to store the headers of the party records.  SQL Script:  CREATE TABLE oai\_records (  serial INT(11) PRIMARY KEY auto\_increment,  provider varchar(255),  url varchar(255),  enterdate datetime,  oai\_identifier varchar(255),  oai\_set varchar(255),  datestamp datetime,  deleted enum('false','true'),  dc\_title varchar(255),  dc\_creator text,  dc\_subject varchar(255),  dc\_description varchar(255),  dc\_contributor varchar(255),  dc\_publisher varchar(255),  dc\_date date,  dc\_type varchar(255),  dc\_format varchar(255),  dc\_identifier varchar(255),  dc­\_source varchar(255),  dc\_language varchar(255),  dc\_coverage varchar(255),  dc\_right varchar(255),  oai\_metadataprefix varchar(255),  ori\_table\_name varchar(255),  ori\_id varchar(255),  set\_type varchar(255),  ori\_table\_name2 varchar(255),  oai\_identifier\_url varchar(255)  )        Create table: **useraccount -** the table to store the content (body) of the party records.  SQL Script:  CREATE TABLE useraccount (  id\_org varchar(30) PRIMARY KEY auto\_increment,  title varchar(30),  first\_name varchar(100),  family\_name varchar(100),  tel varchar(100),  fax varchar(100),  email varchar(100),  www varchar(100),  address varchar(250),  post\_code varchar(30),  city varchar(30),  state varchar(30),  country varchar(30),  duty varchar(30),  id\_rep varchar(30),  subject varchar(255),  description varchar(2000),  relatedinfo varchar(255),  idpub varchar(255),  imagepicture blob,  nlaid varchar(100),  for1 varchar(30),  for1\_pct varchar(30),  for2 varchar(30),  for2\_pct varchar(30),  for3 varchar(30),  for3\_pct varchar(30),  staffnumber varchar(50)  )  Create table: **publication**  SQL Script:  Create Table: CREATE TABLE publication (  pubid varchar(30) PRIMARY KEY auto\_increment,  title varchar(500),  yearpublished varchar(40),  source varchar(50),  authorid varchar(40),  ori\_id varchar(50),  PubAuthorPK varchar(40),  Included varchar(30),  Ands\_pub\_id varchar(100),  Orig\_author\_id varchar(100)  ) |