Build a New User Registration Form with the Users API

Tamara Marnell, Central Oregon Community College

Description

Build a web form that new community and Summit visiting patrons can use to create internal accounts in Alma.

APIs

Users: https://developers.exlibrisgroup.com/alma/apis/users/

Languages

Examples use PHP for server-side processing. The front-end form can be written manually in HTML, CSS and JavaScript, or built in a content management system (CMS) that can send submissions to a server-side script that uses the API calls described below.

Server-side or client-side?

Both

How To

This application consists of two parts:

1) A public-facing HTML form for new patrons to submit.
2) A server-side script that creates an internal user account in Alma by submitting an XML User object in the body of a POST request to the Users API.*

* Because only the Analytics API allows requests from other domains (see CORS), the processing cannot be done client-side.

Step 1: Create the HTML Form

Create a public-facing form with fields for all information you require or would like to store in Alma internal accounts. The documentation for the User object lists the following fields as mandatory to create accounts through the API:

- record_type
- primary_id
- account_type
- email (email_address and email_types)
- phones (phone_number and phone_types)
- address (line1, city, and address_types)

Log in to your Developer Network account to see the possible values your institution can submit for record_type, account_type, user_groups, etc.

At Central Oregon Community College, we also ask Summit Visiting patrons to indicate their home institution and ID number. We save Summit affiliations to the User Statistics section and ID numbers as External Identifiers, so those patrons can use their own school ID cards to check out COCC materials.

Here are sample files for a plain front-end form in PHP. The application policies and group codes are specific to COCC.

- new_patron_example.php
    - Requires summit_institutions.php
- new_patron_script.js
- new_patron_style.css

Step 2: Write the Form Processing Script

The server-side script that processes the form data should accomplish the following:

1) Sanitize the data for safe submission to Alma.
2) Create a unique Primary ID for the new account.
3) Construct a User Object to submit to the API.
4) Submit the POST request to create the new internal account.
5) Forward the user to a confirmation page, and optionally notify library staff.

Here is a sample PHP file to process information submitted through the sample form above: new_patron_process.php.

In the following snippets, the constant EXL_URL refers to https://api-na.hosted.exlibrisgroup.com/almaws/v1/users/ set in line 6. An APIKEY must be defined in line 7 for the script to run.

Sanitize the data

The sample code assumes all submitted fields are strings. If you wish to sanitize the data more thoroughly by individual field, see Sanitize Filters in the PHP manual.

```
$post_data = $_POST;
$sanitized_data = array();
foreach ($post_data as $name => $value) {
  $sanitized_data[$name] = filter_var($value, FILTER_SANITIZE_STRING);
}
```

Create a Unique Primary Identifier

At COCC, primary identifiers for community patron and Summit Visiting patron accounts follow the format “p.First.Last” (e.g., p.Luke.Skywalker). Create an identifier that follows your own institution’s policies and procedures.

```
// Get name
$name_first = ucwords($sanitized_data['name_first']);
$name_middle = ucwords($sanitized_data['name_middle']);
$name_last = ucwords($sanitized_data['name_last']);
```

```
$name_full = trim($name_first . ' ' . $name_middle . ' ' . $name_last);
$primary_id = 'p.' . $name_first . '.' . str_replace('-', '', $name_last);
```

Before proceeding, submit a call to the Users API with the created identifier (e.g., /v1/users/p.Luke.Skywalker). If the call returns a user record, modify the primary identifier and try again until the call fails to match an existing account. The sample below accomplishes this by adding a number to the end and rechecking (e.g., p.Luke.Skywalker1, then p.Luke.Skywalker2, etc.)

```
function id_exists($primary_id) {
  $ch1 = curl_init();
  curl_setopt($ch1, CURLOPT_URL, EXL_URL . urlencode($primary_id) . '?apikey='
. APIKEY);
  curl_setopt($ch1, CURLOPT_RETURNTRANSFER, TRUE);
  curl_setopt($ch1, CURLOPT_HTTPHEADER, array('Content-Type:
application/xml'));
  curl_setopt($ch1, CURLOPT_CUSTOMREQUEST, 'GET');
  $response1 = curl_exec($ch1);
  curl_close($ch1);
  if (!stristr($response1, 'errorsExist')) {
    return true;
  }
  return false;
}

// Check for existence of primary ID
$i = 1;
$original_primary_id = $primary_id;
while (id_exists($primary_id)) {
  $primary_id = $original_primary_id.$i;
  $i++;
}
```

Construct the User Object

The User Object is the same structure used for SIS imports. If your institution has an SIS integration profile set up, you can look at the ZIP files on the server used for that job and model your own User Object after that XML.

At COCC we set the expiry and purge dates for community and Summit visiting patrons to 180 days in the future. You can modify the date calculated to align with your institution's policies on line 87.

The PIN field on line 94 and password field on line 107 are both blank because COCC creates Active Directory accounts for all community and Summit Visiting patrons to sign in to Primo. You may need to modify and test PIN/password submissions to work with the Ex Libris Identity Service, social login, or other authentication methods used at your institution.

By default, our script sets the status of the new record to "inactive" on line 109. The patron must present identification at the Circulation Desk before staff activate the account for borrowing privileges.

```
$expiry_date = date('Y-m-d', strtotime('+180 days')) . 'Z';

// Construct user object
$user = '<user>
```

```
  <record_type desc="Public">PUBLIC</record_type>
  <primary_id>' . $primary_id . '</primary_id>
  <first_name>' . $name_first . '</first_name>
  <middle_name>' . $name_middle . '</middle_name>
  <last_name>' . $name_last . '</last_name>
  <full_name>' . $name_full . '</full_name>
  <pin_number/>
  <job_category desc=""/><job_description/>
  <gender desc=""/>
  <user_group desc="' . $group_desc . '">' . $group_code . '</user_group>
  <campus_code desc=""/>
  <web_site_url/>
  <cataloger_level desc="[00] Default Level">00</cataloger_level>
  <preferred_language desc="English">en</preferred_language>
  <birth_date>' . $birth_year . '-'. $birth_month . '-' . $birth_day .
'Z</birth_date>
  <expiry_date>' . $expiry_date . '</expiry_date>
  <purge_date>' . $expiry_date . '</purge_date>
  <account_type desc="Internal">INTERNAL</account_type>
  <external_id/>
  <password/>
  <force_password_change/>
  <status desc="Inactive">INACTIVE</status>
  <contact_info>
    <addresses>
      <address segment_type="External" preferred="true">
        <line1>' . $address_line1 . '</line1>
        <line2>' . $address_line2 . '</line2>
        <city/>
        <state_province/>
        <postal_code/>
        <country/>
        <address_note/>
        <start_date>' . date('Y-m-d') . '</start_date>
        <address_types>
          <address_type desc="Home">home</address_type>
        </address_types>
      </address>
    </addresses>
    <emails>
      <email segment_type="External" preferred="true">
        <email_address>' . $email_address . '</email_address>
        <email_types>
          <email_type desc="' . ucfirst($email_type) . '">' . $email_type .
'</email_type>
        </email_types>
      </email>
    </emails>
    <phones>
      <phone segment_type="External" preferred="true" preferred_sms="false">
        <phone_number>' . $phone_number . '</phone_number>
        <phone_types>
          <phone_type desc="' . ucfirst($phone_type) . '">' . $phone_type .
'</phone_type>
        </phone_types>
      </phone>
    </phones>
```

```
  </contact_info>';

// Summit Visiting Patron info
if ($sanitized_data['summit_affiliation'] == 1) {
  $user .= '<user_identifiers>
    <user_identifier segment_type="External">
      <id_type desc="Additional ID 1">OTHER_ID_1</id_type>
      <value>' . $summit_id . '</value>
      <status>ACTIVE</status>
    </user_identifier>
  </user_identifiers>
  <user_statistics>
    <user_statistic>
      <category_type desc="Institution">INSTITUTION</category_type>
      <statistic_category desc="' . $summit_name . '">' . $summit_code .
'</statistic_category>
      <statistic_note/>
    </user_statistic>
  </user_statistics>';
}

// End user
$user .= '</user>';
```

Create the New Account

Submit the user object to the API in a POST request, following the documentation for Create user.

```
// Create new user
$ch2 = curl_init();
curl_setopt($ch2, CURLOPT_URL, EXL_URL . '?social_authentication=false&apikey='
. APIKEY);
curl_setopt($ch2, CURLOPT_RETURNTRANSFER, TRUE);
curl_setopt($ch2, CURLOPT_HEADER, FALSE);
curl_setopt($ch2, CURLOPT_CUSTOMREQUEST, 'POST');
curl_setopt($ch2, CURLOPT_POSTFIELDS, $user);
curl_setopt($ch2, CURLOPT_HTTPHEADER, array('Content-Type: application/xml'));
$response2 = curl_exec($ch2);
curl_close($ch2);
```

Forward the user to a confirmation message

The following snippet employs the $_SESSION “superglobal” variable to pass specific information about the user’s account to the confirmation page. In PHP, you can pass any of the submitted form data this way to personalize the message.

At this step, you can also send confirmation emails to the new patron and to library staff, using your preferred server-side library.

```
// Forward to confirmation page
if ($response2 !== false && !stristr($response2, 'errorsExist')) {
  $_SESSION['group'] = $group_desc;
  $_SESSION['summit_institution'] = $summit_name;
  header("Location: new_patron_confirm.php");
}
else {
  echo '<p>An error occurred.</p>';
```

```
  preg_match('/<errorMessage>(.*)<\/errorMessage>/', $response2, $matches);
  echo $matches[1];
}
```

A sample confirmation page: new_patron_confirm.php