

Sample External Service V3.5

Introductions

External Service v3.5 is a sample project that allows you to quickly implement an External Service V3.5 APIs to allow the Vitrium software to talk to your systems for authentication and authorization.

This project is written in **PHP with Laravel Framework** and dependencies can be installed with Composer.

This project is compatible with v3.5 of the Vitrium External Service specification. Contact your Vitrium Sales Rep or Technical Contact if you require information on the External Service V3.5 specification or instructions on configuring your Vitrium account to use this service.

For more details on Laravel Framework visit the following link <https://laravel.com/docs/10.x>

Installing Dependencies

This example supports PHP Version 8.2.0 +

Install the latest version of Composer from the following link:

<https://getcomposer.org/Composer-Setup.exe>

To install the dependencies navigate to the root folder, run the following command:

```
composer update
```

Running the project

To run the project, run the following command:

- default(Local Environment)
 php artisan serve
- custom(host with a specific port 23456)

```
php artisan serve --host=0.0.0.0 --port=23456
```

Project Configuration

The **default port** this project runs on is **8000**. To host the external service application on your local machine, please run the project on a **custom port** & **configure port-forward** setting from your router. Also, make sure to **allow the port in firewall's** inbound rule if you have an active windows firewall.

****IMPORTANT****

DO NOT PORT-FORWARD 80, 8080 and other common ports. You might be vulnerable to **Port Scanning Attack** and this could put your entire network and associated computers at **critical risk**.

Connecting the project to Vitrium

Inside Vitrium, navigate to the “Settings” tab, and in the “Account Settings” subtab, enable external service. Select API Version “3.5” in the dropdown that appears underneath and set the “Service URL” to the url of the server that this external service is running on. For details, **please refer to the following**.

The screenshot shows the Vitrium Settings interface. On the left is a sidebar with navigation links: DASHBOARD, CONTENT, USERS, REPORTS, and SETTINGS. Under SETTINGS, there's a sub-menu with Account Settings, Security Settings, Content Settings, Watermark Settings, DRM Policy Settings, Web Viewer Settings, Portal Settings, Staff Users, My Profile, SSO Settings, Integration Settings (which is highlighted), and SMTP Settings. The main content area is titled 'SAVE SETTINGS' and contains two sections. The first section, 'Enable External Service', has a checked checkbox, an 'API Version' dropdown set to '3.0', a 'Service URL' text field with the value 'http://174.43.21.6:2345/vitrium/service/3.0/', a 'Service Timeout' text field with '30', a 'Service Headers' text area, a 'Web Viewer SSO Token Name' text field with 'token', and a 'Custom Watermark Tokens' text area. Below this is a 'Show External Readers' checkbox which is unchecked. A small asterisk indicates required fields. The second section, 'SSO Configuration', has an 'Enable SSO' checked checkbox, an 'SSO Required' unchecked checkbox, an 'Authentication URL' text field, a 'Force Authentication' unchecked checkbox, a 'Query Parameters' text field with 'token', a 'Cookie Names' text field, and an 'Authentication From Query Parameter' text field. A legend at the bottom of the SSO section states '* indicates a required field'.

Enable External Service

API: 3.0

Service Url: http://YourPublicIpAddress:PortNumber/vitrium/service/3.5/

Service Timeout: 30

Web Viewer SSO Token Name: token

Enable SSO

Query Parameters: token

Enable **Show External Readers, if you would like to load users through external service*

Endpoints

Permissions [GET]

`http://YourPublicIpAddress:PortNumber/vitrium/service/3.5/permissions?userid={id}`

Readers [GET]

`http://YourPublicIpAddress:PortNumber/vitrium/service/3.5/readers?page={"index":1,"size":20}&filter={"contains":{"keywords}}&sort={"Username":1}`

Web Viewer Document Authentication [POST]

`https://view.protectedpdf.com/YourDocumentAlias?token=RjRJUXJDd1VzZmlyTEZVTnNpQittSjRXa1BHdnMwSFJRWXA vRjB1cz0=`

Get Token [GET]

`http://YourPublicIpAddress:PortNumber/gentoken?email=user01@example.com`

Database

Example user database for this sample project is stored in JSON File

For adding/updating users, check: /database/userdata.json

```
[
  {
    "Id": "8d241bea-e714-4182-8257-01abfebff134",
    "Username": "user01@example.com",
    "IsActive": true,
    "AccessPolicy": null
  },
]
```

Laravel Structure

For API endpoint routing, check: /route/web.php

```
// route for /authenticate endpoint v3.5. it will load AuthenticateController class then authenticate_3_5() function
Route::post('/vitrium/service/3.5/authenticate', [AuthenticateController::class, 'authenticate_3_5']);
```

The above code will call `authenticate_3_5()` function inside `AuthenticateController`

For Controllers, check: /app/Http/Controllers/

ServiceAuthenticate.php

```
if(!empty($request['Token']) && !is_null($request['Token'])) {
    // Decrypt Token to obtain username
    $username = VitriumUtility::tokenHasher("decrypt", $request['Token']);
} else {
    // Obtain username from the request
    $username = $request['Username'];
}

// validate access by username
if($this->validateAccess($username)){
    $message = 'Authentication success with '.$username;
    $request['consolelog'] === false ? '' : $out->writeln($message);
    // *if you don't send $success = true; in the response it will display error
    $success = true;
} else {
    $message = 'Authentication failed. Invalid username or email';
    $request['consolelog'] === false ? '' : $out->writeln($request);
}
```

This code example demonstrates a basic authentication process. It verifies user access by comparing their provided username against a list of authorized users stored in a JSON file named 'userdata.json'.

By connecting to your database, you can retrieve the necessary user data for validation. Upon successful validation, setting `$success = true;` signifies that the access has been authenticated.

**You can use `$out->writeln($message);` to output messages to console.*

PermissionsController.php

```
// http://vitriumservice.com/api/3.5/permissions?userid={userid}
// to receive the fields from payload, use $request['FIELD_NAME']
$out = new \Symfony\Component\Console\Output\ConsoleOutput();

// data is here, i.e) make an external API request or DB query
$response = [
    "DocIds"      => [],
    "FolderIds"   => [],
    "docExternalKeys" => [],
    "FolderExternalKeys" => [],
];
```

Permissions endpoint is called when you try to access portal. It calls authentication endpoint then permissions endpoint to filter out specified documents in the above response array. Easiest way to handle this will be using docExternalKeys.

ReaderController.php

```
// load user database to $data array
$file = base_path('/database/userdata.json');
$jsonData = file_get_contents($file);
$data = json_decode($jsonData, true);

/*
    define paging and filtering rule
*/

// return filtered user array
if(!empty($data)){
    $response = [

        "Results"    => array_values($data), // use array_values remove index keys from data array
        "TotalRecords" => $totalRecords

    ];
    $message = 'GET - Request successful ~/readers/ #Total Records: '.$totalRecords.', Pages: '.$totalPages.', Records per Page: '.$size.', Applied Filter: '.$filterBy;
```

*You must Enable **Show External Readers**, if you would like to load users through readers endpoint.*

Above shows example of required response. Other than this part of this code is just for pagination and filtering out the result.