EasyStore Shipping Carrier Plugin Development Guide

Introduction

This guide explains how to create a new shipping carrier integration plugin for EasyStore. By following these instructions, you'll be able to integrate any shipping carrier's API with the EasyStore platform.

Plugin Structure

A shipping carrier plugin follows this standard structure:

```
yourcarrier/
├── assets/
│   └── images/
│       └── logo.svg
├── language/
│   └── en-GB/
│       ├── en-GB.plg_easystoreshipping_yourcarrier.ini
│       └── en-GB.plg_easystoreshipping_yourcarrier.sys.ini
├── services/
│   └── provider.php
├── src/
│   ├── Extension/
│   │   └── YourCarrierShipping.php
│   └── Helper/
│       └── YourCarrierApiClient.php
└── yourcarrier.xml
```

Step-by-Step Implementation

1. Create the Plugin XML Manifest

Create a file named yourcarrier.xml with the following structure:

```
<?xml version="1.0" encoding="utf-8"?>
<extension type="plugin" group="easystoreshipping" method="upgrade">
```

```
    <n>PLG_EASYSTORESHIPPING_YOURCARRIER</n>
    <version>1.0.0</version>
    <releaseDate>YYYY-MM-DD</releaseDate>
    <author>Your Name</author>
    <authorEmail>your.email@example.com</authorEmail>
    <authorUrl>https://www.example.com</authorUrl>
    <copyright>Copyright (C) YYYY Your Company. All rights reserved.</copyright>
    <license>GNU General Public License version 3</license>
    <description>PLG_EASYSTORESHIPPING_YOURCARRIER_DESC</description>
    <namespace
path="src">YourNamespace\Plugin\EasyStoreShipping\YourCarrier</namespace>
    <files>
        <folder plugin="yourcarrier">services</folder>
        <folder>assets</folder>
        <folder>src</folder>
        <folder>language</folder>
    </files>
    <languages>
        <language tag="en-GB">language/en-GB/en-
GB.plg_easystoreshipping_yourcarrier.ini</language>
        <language tag="en-GB">language/en-GB/en-
GB.plg_easystoreshipping_yourcarrier.sys.ini</language>
    </languages>
    <config>
        <fields name="params">
            <fieldset name="basic">
                <!-- Define your configuration fields here -->
                <field
                    name="title"
```

```
            type="text"
            label="PLG_EASYSTORESHIPPING_YOURCARRIER_TITLE"
            description="PLG_EASYSTORESHIPPING_YOURCARRIER_TITLE_DESC"
            default="Your Carrier"
        />
        <!-- Add environment settings (test/live) -->
        <field
            name="environment"
            type="radio"
            label="PLG_EASYSTORESHIPPING_YOURCARRIER_ENVIRONMENT_LABEL"
            description="PLG_EASYSTORESHIPPING_YOURCARRIER_ENVIRONMENT_DESC"
            default="mock"
            class="btn-group btn-group-yesno"
        >
            <option
value="mock">PLG_EASYSTORESHIPPING_YOURCARRIER_MOCK</option>
            <option value="test">PLG_EASYSTORESHIPPING_YOURCARRIER_TEST</option>
            <option value="live">PLG_EASYSTORESHIPPING_YOURCARRIER_LIVE</option>
        </field>
        <!-- Add API credentials fields -->
        <field
            name="api_key"
            type="text"
            label="PLG_EASYSTORESHIPPING_YOURCARRIER_API_KEY_LABEL"
            description="PLG_EASYSTORESHIPPING_YOURCARRIER_API_KEY_DESC"
            showon="environment:live"
        />
        <field
```

```
    name="api_secret"
    type="text"
    label="PLG_EASYSTORESHIPPING_YOURCARRIER_API_SECRET_LABEL"
    description="PLG_EASYSTORESHIPPING_YOURCARRIER_API_SECRET_DESC"
    showon="environment:live"
/>
<field
    name="test_api_key"
    type="text"
    label="PLG_EASYSTORESHIPPING_YOURCARRIER_TEST_API_KEY_LABEL"
    description="PLG_EASYSTORESHIPPING_YOURCARRIER_TEST_API_KEY_DESC"
    showon="environment:test"
/>
<field
    name="test_api_secret"
    type="text"
    label="PLG_EASYSTORESHIPPING_YOURCARRIER_TEST_API_SECRET_LABEL"
    description="PLG_EASYSTORESHIPPING_YOURCARRIER_TEST_API_SECRET_DESC"
    showon="environment:test"
/>
<!-- Add default package dimensions -->
<field
    name="default_weight"
    type="text"
    label="PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_WEIGHT_LABEL"
    description="PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_WEIGHT_DESC"
    default="1.0"
/>
```

```
<field
   name="default_length"
   type="text"
   label="PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_LENGTH_LABEL"
   description="PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_LENGTH_DESC"
   default="10"
/>
<field
   name="default_width"
   type="text"
   label="PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_WIDTH_LABEL"
   description="PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_WIDTH_DESC"
   default="10"
/>
<field
   name="default_height"
   type="text"
   label="PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_HEIGHT_LABEL"
   description="PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_HEIGHT_DESC"
   default="10"
/>
<!-- Add handling fee -->
<field
   name="handling_fee"
   type="text"
   label="PLG_EASYSTORESHIPPING_YOURCARRIER_HANDLING_FEE_LABEL"
   description="PLG_EASYSTORESHIPPING_YOURCARRIER_HANDLING_FEE_DESC"
   default="0"
```

```
            />
        </fieldset>
    </fields>
</config>
</extension>
```

2. Create the Main Extension Class

Create src/Extension/YourCarrierShipping.php:

```
<?php
/**
 * @package     EasyStore.Site
 * @subpackage  EasyStoreShipping.yourcarrier
 *
 * @copyright   Copyright (C) YYYY Your Company. All rights reserved.
 * @license     GNU General Public License version 3; see LICENSE
 */

namespace YourNamespace\Plugin\EasyStoreShipping\YourCarrier\Extension;

// phpcs:disable PSR1.Files.SideEffects
\defined('_JEXEC') or die;
// phpcs:enable PSR1.Files.SideEffects

use Joomla\Event\Event;
use JoomShaper\Component\EasyStore\Administrator\Plugin\CommonShippingPlugin;
use JoomShaper\Component\EasyStore\Site\Model\CartModel;
use YourNamespace\Plugin\EasyStoreShipping\YourCarrier\Helper\YourCarrierApiClient;

class YourCarrierShipping extends CommonShippingPlugin
```

```
{
    private const SANDBOX_API_URL = 'https://api-sandbox.yourcarrier.com/rates';
    private const PRODUCTION_API_URL = 'https://api.yourcarrier.com/rates';

    /**
     * Get the shipping methods for your carrier
     *
     * @param Event $event The event object
     *
     * @return void
     *
     * @since 1.0.0
     */
    public function onEasyStoreGetShippingMethods(Event $event)
    {
        $arguments = $event->getArguments();
        $shippingAddress = $arguments['subject']->shipping_address;
        $shippingMethods = [];

        $environment = $this->params->get('environment', 'mock');
        $handlingFee = $this->params->get('handling_fee', 0);

        $isMock = ($environment === 'mock');
        $isSandbox = ($environment === 'test');

        $apiUrl = $isSandbox ? self::SANDBOX_API_URL : self::PRODUCTION_API_URL;

        // Initialize API client
```

```
        if ($isMock) {
            $httpClient = new YourCarrierApiClient();
        } else {
            // Get API credentials from params
            $apiKey = $isSandbox
                ? $this->params->get('test_api_key')
                : $this->params->get('api_key');

            $apiSecret = $isSandbox
                ? $this->params->get('test_api_secret')
                : $this->params->get('api_secret');

            $httpClient = new YourCarrierApiClient($apiKey, $apiSecret, $apiUrl);
        }

        // Get shipping rates from API
        $response = (object) $httpClient->getShippingRates($this->getRequestData($isMock, $isSandbox, $shippingAddress));

        // Process response and format shipping methods
        if (!empty($response->products)) {
            foreach ($response->products as $rate) {
                $shippingMethods[] = $this->formatShippingMethod($rate, $rate['productName'], $handlingFee);
            }
        }

        // Set shipping methods in the event
        $event->setArgument('shippingMethods', $shippingMethods);
```

```
}

/**
 * Get the request data for API call
 *
 * @param bool $isMock Whether to use mock data
 * @param bool $isSandbox Whether to use sandbox environment
 * @param object $shippingAddress The shipping address object
 *
 * @return array The request data
 *
 * @since 1.0.0
 */
public function getRequestData($isMock, $isSandbox, $shippingAddress)
{
    $accountNumber = $isMock
        ? '123456789'
        : ($isSandbox
            ? $this->params->get('test_account_number')
            : $this->params->get('account_number'));

    $cart = new CartModel();
    $totalWeight = $cart->calculateTotalWeight();

    if ($totalWeight > 0) {
        $weight = $totalWeight;
    } else {
        $weight = $this->params->get('default_weight', 1.0);
```

```
    }

    $originAddress = $this->getOriginAddress();

    $data = [
        'accountNumber' => $accountNumber,
        'originCountryCode' => $originAddress->country,
        'originPostalCode' => $originAddress->postcode,
        'originCityName' => $originAddress->city,
        'destinationCountryCode' => $shippingAddress->country,
        'destinationPostalCode' => $shippingAddress->postcode,
        'destinationCityName' => $shippingAddress->city,
        'weight' => $weight,
        'length' => $this->params->get('default_length', 10),
        'width' => $this->params->get('default_width', 10),
        'height' => $this->params->get('default_height', 10),
        'unitOfMeasurement' => 'metric', // or 'imperial'
    ];

    return $data;
}

/**
 * Format shipping method from API response
 *
 * @param array $rate The rate data from API
 * @param string $name The shipping method name
 * @param float $handlingFee Additional handling fee
```

```
     *
     * @return array Formatted shipping method
     *
     * @since 1.0.0
     */
    private function formatShippingMethod($rate, $name, $handlingFee)
    {
        return [
            'id' => 'yourcarrier_' . $rate['serviceCode'],
            'title' => $name,
            'cost' => $rate['totalAmount'] + $handlingFee,
            'tax_class_id' => 0,
            'text' => $name
        ];
    }

    /**
     * Get the origin address from store settings
     *
     * @return object The origin address
     *
     * @since 1.0.0
     */
    private function getOriginAddress()
    {
        // Get store address from settings
        $settings = $this->app->bootComponent('com_easystore')
            ->getMVCFactory()
```

```
            ->createModel('Settings', 'Administrator')
            ->getSettings();

        return (object) [
            'country' => $settings->get('country', ''),
            'postcode' => $settings->get('postcode', ''),
            'city' => $settings->get('city', ''),
            'address' => $settings->get('address', ''),
        ];
    }
}
```

3. Create the API Client Helper

Create src/Helper/YourCarrierApiClient.php:

```
<?php
/**
 * @package     EasyStore.Site
 * @subpackage  EasyStoreShipping.yourcarrier
 *
 * @copyright   Copyright (C) YYYY Your Company. All rights reserved.
 * @license     GNU General Public License version 3; see LICENSE
 */

namespace YourNamespace\Plugin\EasyStoreShipping\YourCarrier\Helper;

// phpcs:disable PSR1.Files.SideEffects
\defined('_JEXEC') or die;
// phpcs:enable PSR1.Files.SideEffects
```

```
class YourCarrierApiClient
{
    private $baseUrl;
    private $authCredentials;
    private $curl;

    /**
     * Constructor
     *
     * @param string $apiKey API key
     * @param string $apiSecret API secret
     * @param string $baseUrl API base URL
     */
    public function __construct($apiKey = 'demo-key', $apiSecret = 'demo-secret', $baseUrl =
'https://api-mock.yourcarrier.com/rates')
    {
        $this->baseUrl = $baseUrl;
        $this->authCredentials = base64_encode($apiKey . ':' . $apiSecret);
        $this->curl = curl_init();
        $this->setDefaultCurlOptions();
    }

    /**
     * Destructor
     */
    public function __destruct()
    {
        if ($this->curl) {
```

```
        curl_close($this->curl);
    }
}

/**
 * Set default cURL options
 *
 * @return void
 */
private function setDefaultCurlOptions(): void
{
    curl_setopt_array($this->curl, [
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_ENCODING => '',
        CURLOPT_MAXREDIRS => 10,
        CURLOPT_TIMEOUT => 30,
        CURLOPT_FOLLOWLOCATION => true,
        CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
        CURLOPT_CUSTOMREQUEST => 'GET',
        CURLOPT_HTTPHEADER => $this->getDefaultHeaders(),
        CURLOPT_SSL_VERIFYPEER => true,
        CURLOPT_SSL_VERIFYHOST => 2,
    ]);
}

/**
 * Get default HTTP headers
 *
```

```
     * @return array
     */
    private function getDefaultHeaders(): array
    {
        return [
            'Authorization: Basic ' . $this->authCredentials,
            'Accept: application/json',
            'Content-Type: application/json',
        ];
    }

    /**
     * Update authentication credentials
     *
     * @param string $apiKey
     * @param string $apiSecret
     * @return void
     */
    public function updateCredentials(string $apiKey, string $apiSecret): void
    {
        $this->authCredentials = base64_encode($apiKey . ':' . $apiSecret);
        curl_setopt($this->curl, CURLOPT_HTTPHEADER, $this->getDefaultHeaders());
    }

    /**
     * Get shipping rates from API
     *
     * @param array $params Query parameters for the API call
```

```
     * @return array Decoded JSON response
     * @throws \Exception If API call fails
     */
    public function getShippingRates(array $params): array
    {
        try {
            // Validate parameters
            $this->validateShippingParams($params);

            // Build query string and set URL
            $queryString = http_build_query($params);
            curl_setopt($this->curl, CURLOPT_URL, $this->baseUrl . '?' . $queryString);

            // Execute request
            $response = curl_exec($this->curl);

            // Handle errors
            if ($response === false) {
                throw new \Exception('cURL error: ' . curl_error($this->curl));
            }

            $httpCode = curl_getinfo($this->curl, CURLINFO_HTTP_CODE);
            if ($httpCode >= 400) {
                throw new \Exception('HTTP error: ' . $httpCode);
            }

            // Parse and return response
            return json_decode($response, true);
```

```
    } catch (\Exception $e) {
        // Log the error
        error_log('YourCarrier API Error: ' . $e->getMessage());

        // Return mock data in case of error
        return $this->getMockRates();
    }
}

/**
 * Get mock shipping rates for testing
 *
 * @return array Mock rates
 */
private function getMockRates(): array
{
    return [
        'products' => [
            [
                'serviceCode' => 'express',
                'productName' => 'Express Delivery',
                'totalAmount' => 15.99
            ],
            [
                'serviceCode' => 'standard',
                'productName' => 'Standard Delivery',
                'totalAmount' => 9.99
            ],
```

```
            [
                'serviceCode' => 'economy',
                'productName' => 'Economy Delivery',
                'totalAmount' => 5.99
            ]
        ]
    ];
}

/**
 * Validate shipping parameters
 *
 * @param array $params Parameters to validate
 * @return void
 * @throws \Exception If parameters are invalid
 */
private function validateShippingParams(array $params): void
{
    $requiredParams = [
        'originCountryCode',
        'originPostalCode',
        'destinationCountryCode',
        'destinationPostalCode',
        'weight'
    ];

    foreach ($requiredParams as $param) {
        if (empty($params[$param])) {
```

```
                throw new \Exception('Missing required parameter: ' . $param);
            }
        }
    }
}
```

4. Create Language Files

Create language files in language/en-GB/:

en-GB.plg_easystoreshipping_yourcarrier.ini:

PLG_EASYSTORESHIPPING_YOURCARRIER="EasyStore Shipping - Your Carrier"

PLG_EASYSTORESHIPPING_YOURCARRIER_DESC="Shipping integration for Your Carrier"

PLG_EASYSTORESHIPPING_YOURCARRIER_TITLE="Your Carrier"

PLG_EASYSTORESHIPPING_YOURCARRIER_TITLE_DESC="The title displayed to customers"

PLG_EASYSTORESHIPPING_YOURCARRIER_ENVIRONMENT_LABEL="Environment"

PLG_EASYSTORESHIPPING_YOURCARRIER_ENVIRONMENT_DESC="Select the environment for API calls"

PLG_EASYSTORESHIPPING_YOURCARRIER_MOCK="Mock (Testing)"

PLG_EASYSTORESHIPPING_YOURCARRIER_TEST="Sandbox"

PLG_EASYSTORESHIPPING_YOURCARRIER_LIVE="Production"

PLG_EASYSTORESHIPPING_YOURCARRIER_API_KEY_LABEL="API Key"

PLG_EASYSTORESHIPPING_YOURCARRIER_API_KEY_DESC="Your production API key"

PLG_EASYSTORESHIPPING_YOURCARRIER_API_SECRET_LABEL="API Secret"

PLG_EASYSTORESHIPPING_YOURCARRIER_API_SECRET_DESC="Your production API secret"

PLG_EASYSTORESHIPPING_YOURCARRIER_TEST_API_KEY_LABEL="Test API Key"

PLG_EASYSTORESHIPPING_YOURCARRIER_TEST_API_KEY_DESC="Your sandbox API key"

PLG_EASYSTORESHIPPING_YOURCARRIER_TEST_API_SECRET_LABEL="Test API Secret"

PLG_EASYSTORESHIPPING_YOURCARRIER_TEST_API_SECRET_DESC="Your sandbox API secret"

PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_WEIGHT_LABEL="Default Weight (kg)"

PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_WEIGHT_DESC="Default weight in kilograms when product weight is not available"

PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_LENGTH_LABEL="Default Length (cm)"

PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_LENGTH_DESC="Default package length in centimeters"

PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_WIDTH_LABEL="Default Width (cm)"

PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_WIDTH_DESC="Default package width in centimeters"

PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_HEIGHT_LABEL="Default Height (cm)"

PLG_EASYSTORESHIPPING_YOURCARRIER_DEFAULT_HEIGHT_DESC="Default package height in centimeters"

PLG_EASYSTORESHIPPING_YOURCARRIER_HANDLING_FEE_LABEL="Handling Fee"

PLG_EASYSTORESHIPPING_YOURCARRIER_HANDLING_FEE_DESC="Additional fee added to shipping cost"

en-GB.plg_easystoreshipping_yourcarrier.sys.ini:

PLG_EASYSTORESHIPPING_YOURCARRIER="EasyStore Shipping - Your Carrier"

PLG_EASYSTORESHIPPING_YOURCARRIER_DESC="Shipping integration for Your Carrier"

5. Create the Service Provider

Create services/provider.php:

```
<?php

/**
 * @package     EasyStore.Site
 * @subpackage  EasyStoreShipping.yourcarrier
 *
 * @copyright   Copyright (C) YYYY Your Company. All rights reserved.
 * @license     GNU General Public License version 3; see LICENSE
 */


defined('_JEXEC') or die;
```

```
use Joomla\CMS\Extension\PluginInterface;
use Joomla\CMS\Factory;
use Joomla\CMS\Plugin\PluginHelper;
use Joomla\DI\Container;
use Joomla\DI\ServiceProviderInterface;
use Joomla\Event\DispatcherInterface;
use YourNamespace\Plugin\EasyStoreShipping\YourCarrier\Extension\YourCarrierShipping;

return new class implements ServiceProviderInterface {
    public function register(Container $container)
    {
        $container->set(
            PluginInterface::class,
            function (Container $container) {
                $dispatcher = $container->get(DispatcherInterface::class);
                $plugin = new YourCarrierShipping(
                    $dispatcher,
                    (array) PluginHelper::getPlugin('easystoreshipping', 'yourcarrier')
                );
                $plugin->setApplication(Factory::getApplication());
                return $plugin;
            }
        );
    }
};
```

6. Add Your Carrier Logo

Create assets/images/logo.svg with your carrier's logo in SVG format.

Testing Your Plugin

1. Install the plugin through the Joomla administrator interface
2. Enable the plugin in Extensions > Plugins
3. Configure the plugin settings
4. Test the shipping rates calculation in the checkout process

Best Practices

1. **Error Handling**: Always implement proper error handling and provide fallback options
2. **Logging**: Log API requests and responses for debugging
3. **Caching**: Consider caching shipping rates to improve performance
4. **Validation**: Validate all input data before sending to the carrier's API
5. **Security**: Never expose API credentials in client-side code

Common Integration Points

1. **Rate Calculation**: Implement the onEasyStoreGetShippingMethods event to calculate shipping rates

By following this guide, you can create a fully functional shipping carrier integration plugin for EasyStore that seamlessly integrates with the checkout process.