Zapier Integration Guide for Your WordPress Plugin

This document provides a step-by-step guide to integrate Zapier with your WordPress plugin for two main features:

1. **AI Chat Webhook:** Connect external chat clients (like Facebook Messenger) to your plugin's AI.
2. **Lead Information Webhook:** Send lead data collected by your plugin to external services.

Part 1: AI Chat Webhook Integration

Goal: Allow Zapier to send user questions from external chat clients to your AI plugin and receive answers back.

Steps:

1. **Create `class-zapier-integration.php` file:**
    - Create a new file named `class-zapier-integration.php` in the `includes` directory of your plugin (create the `includes` directory if it doesn't exist).
    - Copy the following code into `your-plugin/includes/class-zapier-integration.php`:

```
<?php
/**
 * Class to handle Zapier integration for AI Chat.
 */
declare(strict_types=1);

class Zapier_Integration {

    /**
     * Initialize Zapier integration.
     */
    public function __construct() {
        add_action('rest_api_init', [$this,
'register_zapier_webhook_route']);
    }

    /**
     * Register REST API route for Zapier webhook.
     */
    public function register_zapier_webhook_route(): void {
        register_rest_route(
            'your-plugin/v1', // Namespace
            '/ai-chat-webhook', // Route
            [
                'methods'             => 'POST',
                'callback'            => [$this,
'handle_ai_chat_webhook'],
                'permission_callback' => '__return_true', // Adjust
```

```
permission callback as needed
            ]
        );
    }

    /**
     * Handle the incoming webhook request from Zapier for AI chat.
     *
     * @param WP_REST_Request $request The REST request object.
     * @return WP_REST_Response|WP_Error
     */
    public function handle_ai_chat_webhook(WP_REST_Request $request) {
        $params = $request->get_params();
        $user_question = isset($params['question']) ?
sanitize_text_field($params['question']) : '';

        if (empty($user_question)) {
            return new WP_Error('missing_question', 'No question
provided in the webhook request.', ['status' => 400]);
        }

        // ... existing code ...
        // Process the user question with your AI plugin's core logic
here.
        // Replace this with the actual function call to your AI
processing.
        $ai_response = $this->process_ai_question($user_question);
        // ... existing code ...

        if (is_wp_error($ai_response)) {
            return $ai_response;
        }

        return rest_ensure_response(['answer' => $ai_response]);
    }

    /**
     * Process the AI question and get the response from your AI model.
     *
     * @param string $question User's question.
     * @return string|WP_Error AI response or WP_Error on failure.
     */
    private function process_ai_question(string $question) {
        // ... existing code ...
        // This is a placeholder. Replace with your actual AI
processing logic.
        // For example, calling a function from your main AI class.
        // Assuming you have a class named 'AI_Chat_Core' with a method
'get_ai_answer'.
        if (class_exists('AI_Chat_Core') &&
method_exists('AI_Chat_Core', 'get_ai_answer')) {
            $ai_core = new AI_Chat_Core(); // Instantiate your AI core
class.
            $response = $ai_core->get_ai_answer($question);
```

```
            if (is_string($response)) {
                return $response;
            } else {
                return new WP_Error('ai_processing_failed', 'AI
processing failed to generate a response.', ['status' => 500]);
            }
        } else {
            return new WP_Error('ai_core_missing', 'AI Core class or
method not found.', ['status' => 500]);
        }
        // ... existing code ...
    }
}

new Zapier_Integration();
```

- **Edit your main plugin file** (e.g., `your-plugin/your-plugin.php`) and add the following line to include the new file:

```
<?php
/**
 * Plugin Name: Your Plugin Name
 * ...
 */
declare(strict_types=1);

// ... existing code ...

require_once plugin_dir_path(__FILE__) . 'includes/class-zapier-
integration.php';

// ... existing code ...
```

2. **Implement `process_ai_question()` function:**

- **Edit:** `your-plugin/includes/class-zapier-integration.php`
- **Locate:** the `process_ai_question()` function in the `Zapier_Integration` class.
- **Replace:** the placeholder code within this function with the actual code that processes the `$question` using your AI plugin's core logic. **Adapt the example below to match your plugin's AI processing mechanism.**

```
// ... existing code ...
{{ edit_1 }}
    /**
     * Process the AI question and get the response from your AI model.
     *
     * @param string $question User's question.
```

```
     * @return string|WP_Error AI response or WP_Error on failure.
     */
    private function process_ai_question(string $question) {
        // ... existing code ...
        // Replace this with your actual AI processing logic.
        // Example: Assuming you have an AI class instance in your
plugin.
        global $ai_plugin_instance; // Replace with how you access your
AI instance
        if (isset($ai_plugin_instance) &&
method_exists($ai_plugin_instance, 'get_ai_response')) {
            $response = $ai_plugin_instance-
>get_ai_response($question); // Replace 'get_ai_response' with your
actual method
            if (is_string($response)) {
                return $response;
            } else {
                return new WP_Error('ai_processing_failed', 'AI
processing failed to generate a response.', ['status' => 500]);
            }
        } else {
            return new WP_Error('ai_core_missing', 'AI Core instance or
method not found.', ['status' => 500]);
        }
        // ... existing code ...
    }
{{ edit_2 }}
// ... existing code ...
```

3. **Test the Webhook Endpoint (using Postman or similar):**
   - Use a tool like Postman, Insomnia, or `curl` to send a POST request to your webhook URL: `https://your-wordpress-site.com/wp-json/your-plugin/v1/ai-chat-webhook` (replace with your actual site and plugin path).
   - Request Body (JSON):

```
{
  "question": "Hello AI, how are you?"
}
```

   - Verify you receive a JSON response containing the AI's answer.
   - Check WordPress debug logs (`wp-content/debug.log`) for errors if needed.

4. **Create a Zap in Zapier (for AI Chat):**
   - Go to Zapier and create a new Zap.
   - **Trigger:** Choose your desired chat client (e.g., "Facebook Messenger") and the "New Message" trigger. Connect your account and set up the trigger.
   - **Action:** Choose "Webhook by Zapier" and select "POST".

        - **URL:** `https://your-wordpress-site.com/wp-json/your-plugin/v1/ai-chat-webhook` (your webhook URL).
        - **Payload Type:** "JSON".
        - **Data:** Key: `question`, Value: Select the field from your trigger app containing the user's message.
        - Test the action.
    - **(Optional) Response Action:** Add another action to send the AI's response back to the chat client (e.g., "Facebook Messenger" - "Send Message"). Configure it to use the "answer" field from the Webhook action's output.
    - Name and turn "On" your Zap.

5. **Test the Full AI Chat Flow:**
    - Send a message to your configured chat client.
    - Confirm your AI plugin processes the question and sends a response back via Zapier.

Part 2: Lead Information Webhook Integration

Goal: Send lead information from your plugin to external sources via Zapier.

Steps:

1. **Add Settings for Lead Webhook URL:**
    - **Edit:** `your-plugin/admin/settings.php` (or your plugin's admin settings file).
    - Add the following code to create a settings section and field for the Zapier Lead Webhook URL:

```php
<?php
// ... existing code ...

{{ edit_1 }}
/**
 * Add settings section for Zapier integration.
 */
add_settings_section(
    'your_plugin_zapier_settings_section',
    'Zapier Integration',
    'your_plugin_zapier_settings_section_callback',
    'your-plugin-settings-page'
);

/**
 * Zapier settings section callback.
 */
function your_plugin_zapier_settings_section_callback(): void {
    echo '<p>Configure Zapier integration settings here.</p>';
}

/**
 * Register Zapier Lead Webhook URL setting.
```

```
 */
register_setting('your-plugin-settings-group',
'your_plugin_zapier_lead_webhook_url', 'esc_url_raw');
add_settings_field(
    'your_plugin_zapier_lead_webhook_url',
    'Lead Webhook URL',
    'your_plugin_zapier_lead_webhook_url_field',
    'your-plugin-settings-page',
    'your_plugin_zapier_settings_section'
);

/**
 * Lead Webhook URL field callback.
 */
function your_plugin_zapier_lead_webhook_url_field(): void {
    $webhook_url =
esc_attr(get_option('your_plugin_zapier_lead_webhook_url'));
    echo "<input type='url' name='your_plugin_zapier_lead_webhook_url'
value='{$webhook_url}' class='regular-text'>";
    echo '<p class="description">Enter the Zapier Webhook URL to send
lead information.</p>';
}

// ... existing code ...
```

2. **Implement `trigger_lead_webhook()` function:**
   - **Edit:** `your-plugin/includes/class-zapier-integration.php`
   - **Locate:** the `Zapier_Integration` class and ensure the `trigger_lead_webhook()` function is present:

```
// ... existing code ...
{{ edit_1 }}
    /**
     * Action to trigger when lead information is collected.
     */
    public function trigger_lead_webhook(array $lead_data): void {
        $zapier_webhook_url =
get_option('your_plugin_zapier_lead_webhook_url', ''); // Get webhook
URL from settings

        if (empty($zapier_webhook_url)) {
            return; // No webhook URL set, do nothing.
        }

        $payload = wp_json_encode($lead_data);

        $response = wp_remote_post(
            $zapier_webhook_url,
            [
                'headers' => ['Content-Type' => 'application/json'],
```

```
                'body'    => $payload,
                'blocking' => false, // Non-blocking request
            ]
        );

        if (is_wp_error($response)) {
            error_log('Zapier Lead Webhook Error: ' . $response-
>get_error_message());
        }
    }
{{ edit_2 }}
// ... existing code ...
```

3. **Identify Lead Collection Point and Implement Action Hook:**
   - **Locate:** the code in your plugin where lead information is collected.
   - **Edit:** the relevant file (e.g., `your-plugin/includes/your-lead-collection-module.php`).
   - **Implement:** the `do_action('your_plugin_lead_collected', $lead_data)` hook and the `send_lead_data_to_zapier()` function. **Adapt the `collect_lead_data()` example to your plugin's lead collection process.**

```
<?php
declare(strict_types=1);

// ... existing code ...

{{ edit_1 }}
function collect_lead_data(string $name, string $email, string
$other_info): void {
    // ... existing code ...
    // Your lead data collection logic here (e.g., saving to database).
    // ... existing code ...

    $lead_data = [
        'name'       => $name,
        'email'      => $email,
        'other_info' => $other_info,
        'timestamp'  => current_time('mysql'),
    ];

    // Trigger Zapier webhook to send lead data.
    do_action('your_plugin_lead_collected', $lead_data);
}

// Hook into the lead collection action and send data to Zapier.
add_action('your_plugin_lead_collected', 'send_lead_data_to_zapier');
function send_lead_data_to_zapier(array $lead_data): void {
    $zapier_integration = new Zapier_Integration(); // Instantiate
Zapier integration class.
    $zapier_integration->trigger_lead_webhook($lead_data);
```

```
}
{{ edit_2 }}
```

4. **Create a Zap in Zapier (for Lead Information):**
    - Go to Zapier and create a new Zap.
    - **Trigger:** Choose "Webhook by Zapier" and select "Catch Hook".
    - **Get Webhook URL:** Zapier will provide a webhook URL. **Copy this URL.**
    - **Paste into Plugin Settings:** In your WordPress admin, go to your plugin's settings and paste the Zapier webhook URL into the "Lead Webhook URL" field. Save settings.
    - **Test Trigger:** In Zapier, click "Continue" and "Test Trigger".
    - **Trigger Lead Collection:** Perform the action in your plugin that collects lead data.
    - **Verify Trigger Success:** Check Zapier to see if the test trigger was successful and received lead data.
    - **Action:** Add an action to send lead data to your desired app (e.g., Google Sheets, CRM). Configure it to use the data from the Webhook trigger.
    - Test the action.
    - Name and turn "On" your Zap.
5. **Test the Full Lead Information Flow:**
    - Trigger lead collection in your plugin.
    - Verify lead information is sent to your chosen destination app via Zapier.

Part 3: Testing and Refinement

1. **Thorough Testing:** Test both AI Chat and Lead Information integrations in various scenarios.
2. **Error Handling:** Review and improve error handling in your code. Log errors appropriately.
3. **Security Review:** Ensure security, especially for the AI Chat webhook. Use HTTPS.
4. **Performance:** Verify performance, especially for webhook requests. Non-blocking requests are recommended.
5. **User Experience:** Make setup easy for users. Provide clear instructions.

Part 4: Documentation

1. **Create User Documentation:** Write clear documentation for your plugin users on setting up Zapier integration.
2. **Include in Documentation:**
    - Step-by-step Zap creation instructions for both features.
    - How to get webhook URLs from Zapier.
    - Where to paste webhook URLs in plugin settings.
    - Troubleshooting tips and common issues.
    - Consider screenshots or videos for clarity.

This guide should help you integrate Zapier into your WordPress plugin. Remember to adapt code examples to your plugin's specific structure and logic.