Textit.biz RECORDS API Integration Guide

Step 1: Understand the API

Note : *The output of the RECORDS API is cached and updated every 15 minutes for optimal performance.*

Endpoint : The API endpoint is `https://recordsapi.textit.biz/ `.

Parameters : `id` : Your User ID. [Required]

`pw` : Your Password. [Required]

`month` : Query Month [Optional]

Expected Values for `month` parameter :

If the parameter is 0 or omitted: Select the current month.
-1: Select the last month.
-2: Select the month before last.
-3: Select two months before last

Response Format : The API returns data in XML format with the following structure:

```
<root>
  <item>
    <time>2024-08-01 09:20:40</time>
    <to>772823050</to>
    <text>This is an example text</text>
    <schedule/>
    <cost>0.8424</cost>
    <stat>delivered | 2024-08-01 09:20:43</stat>
  </item>
  <!-- More items -->
</root>
```
```

## **Step 2:** Access the API

Use PHP or any other programming language of your choice to send a request to the API and retrieve the XML data. Below is an example using PHP `file_get_contents`.
```

Step 3: Parse the XML Data

Once the data is fetched, use `SimpleXMLElement` to parse the XML and extract the information.

Step 4: Display the Data in an HTML Table

Create an HTML table and populate it with the data extracted from the XML.

Example PHP Script

```
<?php
// Replace with your actual User ID and Password
$userId = 'yourUserID';
$password = 'yourPassword';
$month=0;
$apiUrl = "https://recordsapi.textit.biz/?id=$userId&pw=$password&month=$month";

// Fetch the XML data from the API
$xmlData = file_get_contents($apiUrl);

// Check if data was fetched successfully
if ($xmlData === FALSE) {
    die("Error fetching data from API.");
}

// Parse XML data
try {
    $xml = new SimpleXMLElement($xmlData);
} catch (Exception $e) {
    die("Error parsing XML: " . $e->getMessage());
}

// Start the HTML table
echo "<table border='1'>";
echo "<tr><th>Time</th><th>To</th><th>Text</th><th>Status</th></tr>";

// Extract data and populate the table
foreach ($xml->item as $item) {
    echo "<tr>";
    echo "<td>" . htmlspecialchars($item->time) . "</td>";
    echo "<td>" . htmlspecialchars($item->to) . "</td>";
    echo "<td>" . htmlspecialchars($item->text) . "</td>";
    echo "<td>" . htmlspecialchars($item->stat) . "</td>";
    echo "</tr>";
}

// End the HTML table
echo "</table>";
?>
```

Explanation of the Code

1. **API URL Construction**: The URL is dynamically created by replacing `yourUserID` and `yourPassword` with your credentials and `month` with the required month.

2. **Fetching Data**: The `file_get_contents` function is used to retrieve data from the API. If the retrieval fails, the script will terminate with an error message.

3. **Parsing XML**: The `SimpleXMLElement` class is used to parse the XML string. An error is caught and displayed if parsing fails.

4. **Generating the HTML Table**:

   - The table begins with headers for each data field (`Time`, `To`, `Text`, `Status`).

   - The script iterates over each `<item>` element in the XML, extracting the `time`, `to`, `text`, and `stat` elements. You can add or remove elements as you like

   - Each field is safely escaped using `htmlspecialchars` to prevent HTML injection.

   - The data is then populated into rows within the table.

Additional Considerations

- **Security**: Securely store and handle API credentials, ideally using environment variables or a separate configuration file.

- **Error Handling**: Expand error handling to cover more scenarios, such as network failures or unexpected data formats.

- **Styling**: Enhance the HTML table's appearance using CSS for better readability and user experience.