Meita/Zatca Package Demo Guide

This guide demonstrates how to use the **Meita/Zatca** PHP package under various scenarios: - Pure PHP environment versus Laravel projects. - Creating simplified (simplified tax) and standard (tax invoice) invoices. - Configuring for a single company and for multiple companies. - Normalizing and validating invoice data before sending. - Handling errors returned from ZATCA.

The examples below provide a reference for developers integrating e-invoicing solutions in Saudi Arabia.

---

1. Requirements and Installation

Before starting:

1. Install PHP 8.0+ and Composer.
2. Install the **Meita/Zatca** package via Composer:

```
composer require meita/zatca
```

This will add the package to your project and set up autoloading.

---

2. Configuration Overview

The package reads company-specific details from a configuration file. A default config template is provided inside the package under `src/config/zatca.php`. For Laravel, you can publish it using:

```
php artisan vendor:publish --tag=zatca-config
```

For pure PHP projects, copy the template to a `config` folder in your project. A typical configuration looks like this:

```
return [
    'default' => 'company1',
    'companies' => [
        'company1' => [
            'seller_name'      => 'Dania Air Control System Factory',
            'seller_vat'       => '123456789012345',
            'seller_crn'       => '1012345678',
            'currency'         => 'SAR',
            'invoice_type'     => 'simplified',  // default invoice type
            'invoice_types'    => ['simplified','standard'], // allowed types
            'address' => [
                'street'        => 'King Abdullah Road',
                'building_no'   => '1234',
                'city'          => 'Riyadh',
                'postal_code'   => '11564',
```

```
            'country'       => 'SA',
        ],
        'tax' => [
            'percent'       => 15,
            'category_code' => 'S',
        ],
        'certificate_path' => __DIR__.'/keys/company1_certificate.pem',
        'private_key_path' => __DIR__.'/keys/company1_private.pem',
        'client_id'        => getenv('ZATCA_COMPANY1_CLIENT_ID'),
        'client_secret'    => getenv('ZATCA_COMPANY1_SECRET'),
        'environment'      => 'sandbox',
    ],
    // Add other companies here...
],
];
```

Key points:

- You can define multiple companies under `companies`.
- `invoice_type` sets the default (e.g. ‘simplified’ or ‘standard’).
- `invoice_types` lists all allowed types for this company.
- For real submission, populate your CSID and secret.
- The `keys` folder should contain your private key and certificate files.

---

3. Using the Package in Pure PHP

Below is a step-by-step example of creating, validating and signing a simplified invoice in a pure PHP environment.

3.1. Basic Simplified Invoice (Single Company)

```
<?php

require __DIR__.'/vendor/autoload.php';

use Meita\Zatca\Invoice;
use Meita\Zatca\ZatcaClient;
use Meita\Zatca\Support\ZatcaConfig;
use Meita\Zatca\Support\InvoiceAdapter;
use Meita\Zatca\Support\InvoiceValidator;
use Meita\Zatca\Support\ZatcaException;

try {
    // Load configuration for the default company
    $config   = new ZatcaConfig();          // uses config('zatca') or
require config file
    $adapter  = new InvoiceAdapter();
    $validator= new InvoiceValidator();

    // Example input data (could come from your ERP)
    $input = [
```

```
        'seller_name'       => 'Dania Air Control System Factory',
        'seller_vat'        => '123456789012345',
        'invoice_total'     => 115.0,
        'vat_total'         => 15.0,
        'items' => [
            ['name' => 'Test Item', 'quantity' => 1, 'price' => 100.0],
        ],
    ];

    // Normalize input using the adapter
    $normalized = $adapter->adapt($input);

    // Validate required fields for a simplified invoice
    $missing = $validator->validate($normalized, 'simplified');
    if (!empty($missing)) {
        throw new Exception('Missing fields: '.implode(', ', $missing));
    }

    // Build Invoice object
    $invoice = new Invoice($normalized);

    // Instantiate the client with configuration
    $client = new ZatcaClient($config);

    // Prepare the invoice (XML, hash, signature, QR)
    $prepared = $client->prepareSignedInvoice($invoice);

    echo "Prepared invoice data:\n";
    print_r($prepared);

    // Optionally send to ZATCA (requires valid CSID, secret, and sandbox
endpoints)
    // $response = $client->sendInvoice($invoice);
    // print_r($response);

} catch (ZatcaException $e) {
    // Handle errors returned from ZATCA
    echo "ZATCA Error: [{$e->getCategory()}] {$e->getCode()} - {$e-
>getMessage()}\n";
} catch (Exception $e) {
    echo "General Error: {$e->getMessage()}\n";
}
```

This example demonstrates:

- Loading configuration via `ZatcaConfig`.
- Normalizing raw data with `InvoiceAdapter`.
- Validating required fields with `InvoiceValidator`.
- Generating an `Invoice` instance and signing it via `ZatcaClient`.
- Handling errors gracefully.

3.2. Standard Invoice

To create a standard invoice, set `invoice_type` to `'standard'` and ensure you include buyer details and additional fields:

```
$input['invoice_type'] = 'standard';
$input['buyer_name']   = 'Client Company';
$input['buyer_vat']    = '987654321012345';
$input['invoice_total_exclusive'] = 200.0;
$input['invoice_total_inclusive'] = 230.0;
// Include supply date, net amounts, etc. according to ZATCA specs

$normalized = $adapter->adapt($input);
$missing = $validator->validate($normalized, 'standard');
// Build and send as before
```

---

4. Multi-Company Support in PHP

If your application serves multiple companies, define them in the config file. You can instantiate `ZatcaConfig` for a specific company:

```
// Load config for 'company2'
$configCompany2 = new ZatcaConfig('company2');
$client2        = new ZatcaClient($configCompany2);

// Proceed with invoice preparation as above
```

By calling `new ZatcaConfig('company_key')`, you select the company and its settings.

---

5. Using the Package in Laravel

Laravel integration simplifies dependency resolution through a Service Provider and Facade.

5.1. Setup

1. Publish the configuration:

```
php artisan vendor:publish --tag=zatca-config
```

2. Configure your companies in `config/zatca.php` as shown earlier.
3. Optionally configure environment variables for secrets and keys.

5.2. Creating and Sending Invoices

You may use dependency injection or the facade.

Using Dependency Injection

```
use Meita\Zatca\Invoice;
use Meita\Zatca\ZatcaClient;
use Meita\Zatca\Support\InvoiceAdapter;
use Meita\Zatca\Support\InvoiceValidator;

public function sendInvoice(Request $request, ZatcaClient $zatcaClient)
{
    $adapter = new InvoiceAdapter();
    $validator = new InvoiceValidator();

    $data = $adapter->adapt($request->all());

    $missing = $validator->validate($data, $data['invoice_type']);
    if ($missing) {
        return response()->json(['error' => 'Missing fields: '.implode(', ',
$missing)], 400);
    }

    $invoice = new Invoice($data);
    $prepared = $zatcaClient->prepareSignedInvoice($invoice);

    // Optionally send to ZATCA
    // $response = $zatcaClient->sendInvoice($invoice);

    return response()->json($prepared);
}
```

Using the Facade

```
use Zatca; // Facade alias provided by ServiceProvider
use Meita\Zatca\Invoice;
use Meita\Zatca\Support\InvoiceAdapter;
use Meita\Zatca\Support\InvoiceValidator;

public function generate(Request $request)
{
    $adapter = new InvoiceAdapter();
    $validator = new InvoiceValidator();
    $data = $adapter->adapt($request->all());

    $missing = $validator->validate($data, $data['invoice_type']);
    if ($missing) {
        throw new \Exception('Missing fields: '.implode(', ', $missing));
    }

    $invoice  = new Invoice($data);
    $prepared = Zatca::prepareSignedInvoice($invoice);

    return view('invoice.show', ['result' => $prepared]);
}
```

Here, the facade resolves `ZatcaClient` from the service container and uses the default company. To switch companies, specify the company when instantiating `ZatcaConfig` or adjust the

`default` key in the config.

---

6. Handling Errors and Custom Hooks

ZATCA returns detailed error codes when validation fails. The package throws a `ZatcaException` containing the category, code, and message. You can register an error hook in the config:

```
'companies' => [
    'company1' => [
        // ...
        'on_error' => function ($error) {
            // Log the error or convert it to a custom response
            \Log::error('ZATCA error:', $error);
        },
    ],
],
```

Whenever `ZatcaClient` encounters an error, it invokes this callback before throwing the exception.

---

7. Summary

This demonstration covered:

- Installing and configuring the **Meita/Zatca** package.
- Creating simplified and standard invoices in pure PHP.
- Supporting multiple companies by selecting different configuration profiles.
- Integrating with Laravel using dependency injection or a facade.
- Normalizing and validating invoice data with `InvoiceAdapter` and `InvoiceValidator`.
- Handling errors via hooks and exceptions.

By following these examples and customizing the configuration, you can adapt the package to various invoicing scenarios and ensure compliance with Saudi Arabia’s e-invoicing regulations.
