Journal of Advances and Scholarly Researches in Allied Education
Vol. 18, Issue No. 5, August-2021, ISSN 2230-7540

A Study The Software Development Kit (SDK) Architecture For Payment Gateway Service On The Internet

Ravin Kumar[1]* P. K. Bharti[2]

[1] Research Scholar, Computer Science and Engineering, Shri Venkateshwara University, Gajraula, Uttar Pradesh, India

Email: ravinpal2009@gmail.com

[2] Professor, Shri Venkateshwara University, Gajraula, Uttar Pradesh, India

Email: padutt@gmail.com

Abstract- Payment Card Industry Data Security Standard (PCI DSS)certification makes Solinor the first software company in Finland. A primary motivation for this thesis research was Solinor's need to streamline the process of deploying & integrating the Payment Highway product. With a well-thought-out Software Development Kit that is simple to use and understand, the Payment Highway may be built more quickly for a client's unique project. Research & analysis of the Payment Highway product followed the present state analysis to illuminate the needs of the SDK. After collecting & comprehending user needs, the Software Development Kit's architecture and design were developed. The study's last stage involved the creation & distribution of the SDK for developers.

Keywords- Payment Highway, Software Development Kit's, Architecture, API

----------X----------

INTRODUCTION

The Payment Highway product's manufacturer is introduced initially in this chapter. The history of the thesis and its motivation are then outlined. Additionally, the study topic, thesis outline, and methodology are all presented here.

Solinor was established in 2002 as a software development firm. Innovators' Solutions is an acronym for the company's full name. Solinor utilizes cutting-edge technologies to create software & digital network services for businesses of all sizes & types. Solinor has extensive knowledge in the banking & payment processing sectors. Payment Card Industry Data Security Standard (PCI DSS) certification makes Solinor the first software company in Finland.

A primary motivation for this thesis research was Solinor's need to streamline the process of deploying & integrating the Payment Highway product. By acting as a gateway, Payment Highway streamlines the process of accepting credit card and debit card payments for e-commerce goods.

This study was carried out as follows. Research into the existing payment infrastructure was conducted to determine the Payment Highway's place within the greater payment infrastructure. Research & analysis of the Payment Highway product followed the present state analysis to illuminate the needs of the SDK. After collecting & comprehending user needs, the SDK architecture & design were developed.

OBJECTIVES

1. To study the software development kit (SDK) for payment gateway service on the internet

2. To study the Payment Highway in the payment systems.

RESEARCH METHODOLOGY

Methodology is the systematic, theoretical analysis of the methods applied to a field of study. It comprises the theoretical analysis of the body of methods and principles associated with a branch of knowledge. The term "research" refers to the process of gathering data and details on a specific topic or issue. To put it simply, research is the practice of conducting a thorough examination. It is said that inventions are born out of necessity, & person conducting this research could be considered a researcher. The term "research" would

be utilized in a technical meaning because it is a pedagogical effort.

- **RESEARCH DESIGN**

This study was carried out in the following manner. The current state of the payment system was analyzed to basic theoretical the function of the Payment Highway. A thorough investigation of the Payment Highway product was carried out after the present state study to gain a clear picture of the SDK needs for the product's API. Once the requirements were collected & understood, the SDK's design and architecture could be finalized. The SDK was developed & published in the final phase of this research.

- **SAMPLING DESIGN**

The sample will make use of Solinor, who has extensive knowledge in the financial services & payments sectors. Solinor was the first software company in Finland to be approved as PCI DSS compliant. As per Deloitte's FAST50, Solinor was the fastest-growing custom software firm in Finland in both 2016 & 2017.

- **DATA COLLECTION**

This work relies on both primary & secondary data for its research. The empirical study, which used semi-structured face-to-face interviews as the primary data source, represents the primary data. In the literature review, you'll find a depiction of secondary data. Given the deductive nature of this study, the answers to the study's research questions will be drawn from both primary data acquired during the study & secondary data drawn from the literature review.

Primary Data

In order to collect data, researchers used semi-structured face-to-face interviews. As Collis & Hussey (2014) stated, interviews are a suitable tool for collecting and exploring consumers' thoughts, actions, & feelings about e-payments. As a result of King's (2004) argument that semi-structured interviews can be used for qualitative research, they are nonstandard.

Secondary Data

E-payments research, both general & consumer, is included in the literature review. In the literature review, you may find the secondary data utilized in this survey.

PAYMENT HIGHWAY

This chapter provides a product overview for Payment Highway. Detailed descriptions of Payment Highway's features are provided. It is clarified why Payment Highway is a viable option for internet trade. As new clients get used to the service, the product's difficulties are also discussed.

A payment gateway is provided by the item called Payment Highway. The CHD is tokenized and stored securely by Payment Highway (Cardholder data). Additionally, it offers a safe way to display a web form where users can enter Cardholder information. In this way, the security of the online store is enhanced, and the merchant is relieved of having to complete the challenging PCI DSS assessment process. The merchant is still in charge of making secure use of this service. Payment Highway offers a wide variety of payment processors & acquirers, allowing the merchant to quickly select the least expensive option.

The CHD input form service was created to be a fully customizable HTML online form. The form was created with a responsive design, and the customer can brand it with their logo. This makes it simple to link the Payment Highway with current electronic commerce systems. [Abeysinghe 2008][14]

One of the most important parts of the Payment Highway product is the tokenization service. This implies that in the Payment Highway, the cardholder's information is securely kept before being replaced with a random hash encoding that identifies the card in the database. [Gomzin S. 2014]

The hash is then returned to the retailer so that it can be stored there and used to identify the buyer for future purchases. All of a merchant's tokens can be revoked & produced again if the merchant's service is compromised. With this move, the outdated tokens become useless & CHD is better protected.

PAYMENT HIGHWAY SDK

This chapter describes the Software Development Kit's (SDK) common usage patterns and provides a detailed explanation of the SDK's architecture. This chapter's final section describes how to integrate the SDK into either a fresh or continuing project. Prior to this study, integrating an existing E-commerce system with the Payment Highway required extensive work that was frequently repeated. To make the necessary integration work easier, the SDK project for the Payment Highway was initiated.

The time it takes to onboard new customers is shortened by a well-structured SDK. Additionally, it aids in reducing integration work expenses & raising Payment Highway product sales.

- **Architecture**

[14]Abeysinghe S. RESTful PHP Web Services: Packt Publishing Ltd; 2008. Internet Engineering Task Force (IETF). The JavaScript Object Notation (JSON)

The services & operations of the Payment Highway HTTP API are intuitively replicated by the SDK. The SDK verifies the input data & manages difficult operations including processing HTTP requests, sorting header information, & calculating authentication tokens.

The PHP programming language is used to create the SDK, which is the focus of this thesis. PHP 5.4 is the bare minimum version needed to execute the SDK. The SDK architecture was created with implementation independence in mind, making it possible to utilize it as a self-contained library with any PHP-written framework or program. The architecture was created with the intention of being adaptable to additional programming languages.

The architecture of the FormAPIService is in its most basic version. The sole duty of the FormAPIservice is to provide a form model with the appropriate URI, request headers, including a secure signature, and HTTP request method. The FormAPIService's sketched architecture is shown in Figure 1. Later in this chapter, implementation specifics are discussed.

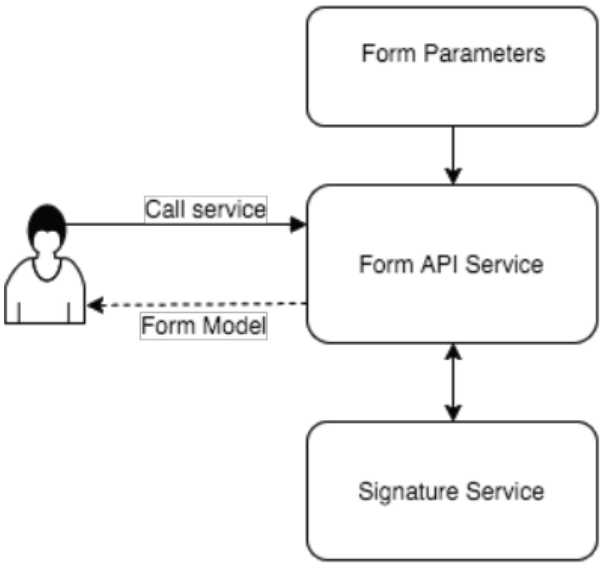

Figure 1. Form API Service architecture

When compared to the FormAPIService, the PaymentAPIService has a more intricate design. This is because of the PaymentAPIService's broader responsibilities, which include handling HTTP queries to the Payment Highway HTTP API & validating the API's responses. The PaymentAPIService's behavior & architecture are shown in Figure 2.

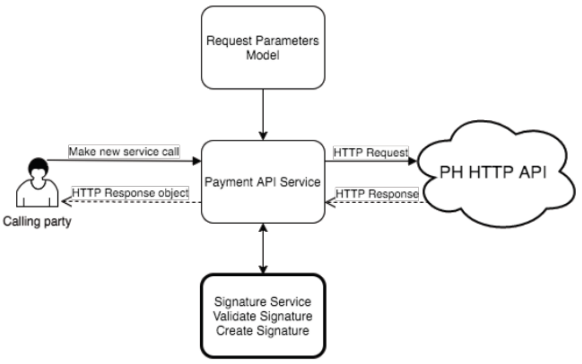

Figure 2. Payment API Service architecture

Table 1 below explains the project structure. The namespaces and their descriptions are displayed in Table 1. The namespaces adhere to the standards outlined in PSR-4. The outdated PSR-0 autoloading standard has been replaced by PSR-4 in PHP.

Table 1. Project structure by namespaces

Namespace	Description
\Solinor\PaymentHighway	Service classes for payment highway SDK. Mirroring the HTTP API Form and Payment API sides.
\Solinor\PaymentHihgway\Model	Models used to inject to PaymentAPIService calls and FormAPIServivce calls.
\Solinor\PaymentHihgway\Security	Security handling services, such as authentication signature calculator and signature validator.

The two primary service classes are contained in the base namespace SolinorPaymentHighway. The model classes utilized to create the request objects passed to the various methods utilized by the FormAPIService class or the PaymentAPIService class are included in the namespace SolinorPayment HighwayModel. PaymentHighway in Solinor Classes that are utilized when security services are needed are included in the security namespace. Either the PaymentAPIService class or the FormAPIService class internally accesses the security services.

- **FormAPIService**

The FormAPIService class facilitates the construction of the HTTP form parameters. The constructor is where the common parameters for all methods are provided when creating a new FormAPIService. In order to obtain a list of the

parameters for each Form API call, it first generates an instance of the service and then utilizes the produced methods. A new instance of the service is created when the new FormAPIService method is called, as seen in Figure 3.

```
use \Solinor\PaymentHighway\FormAPIService

$method = "POST";
$signatureKeyId = "testKey";
$signatureSecret = "testSecret";
$account = "test";
$merchant = "test_merchantId";
$baseUrl = "https://v1-hub-staging.sph-test-solinor.com";
$successUrl = "https://example.com/success";
$failureUrl = "https://example.com/failure";
$cancelUrl = "https://example.com/cancel";
$language = "EN";

$formService = new FormAPIService($method, $signatureKeyId, $signatureSecret, $account,
                                  $merchant, $baseUrl, $successUrl, $failureUrl,
                                  $cancelUrl, $language);
```

Figure 3. Calling new *FormAPIService*.

Now it is possible to use the methods that correspond to Form API HTTP calls. The alternate parameters required for each of the various method calls can be injected using these techniques. The techniques for creating unique parameters for each API action call are shown in Figure 4.

```
$formService->generateAddCardParameters( $accept_cvc_required = false );
$formService->generatePaymentParameters($amount, $currency, $orderId, $description);
$formService->generateAddCardAndPaymentParameters($amount, $currency, $orderId, $description);
$formService->generatePayWithTokenAndCvcParameters( $tokenId, $amount, $currency, $orderId, $description);
```

Figure 4. Techniques for creating distinct parameters for each Form API action calls

Each argument is provided with a string value. The $accept cvc required variable, which takes a Boolean true or false value, is the only exception. The value of the $accept cvc required variable is false by default.

When one of the methods is called, a Form object is returned. The method, URI, & HTTP headers to be utilized for redirecting to the Form API are all included in this object. The calculated authentication signature is included in the HTTP headers together with the signature header with the proper format.

```
// $form variable is the returned Form object.
$form = new Solinor\PaymentHighway\Model\Form();

// returns value as string.
$httpMethod = $form->getMethod();
$actionUrl = $form->getAction();

// Header parameters are stored as key => value paired array
$parameters = $form->getParameters();
```

Figure 5. Form model returned by service method invocation

The FormAPIService differs from the PaymentAPIService in that no HTTP queries are made by the service. Developers are left in charge of the redirection to the FormAPIService. The Software Development Kit doesn't alter the program flow while the developer handles redirection implementation. Additionally, there are no implementation-specific assumptions made in the Software Development Kit.

- **PaymentAPIService**

With regards to interactions with the Payment API, the PaymentAPIservice class is helpful. With the addition of HTTP capability to handle the actual request to the Payment Highway HTTP API, the PaymentAPIservice functions similarly to the FormAPIservice. It first builds a service object, after which it calls the payment API using the request methods. The common parameters for each of the methods are fed into the constructor when establishing a new PaymentAPIService. The new PaymentAPIService is created in Figure 6 using the constructor's parameters.

```
use Solinor\PaymentHighway\PaymentAPIService

$serviceUrl = "https://v1-hub-staging.sph-test-solinor.com";
$signatureKeyId = "testKey";
$signatureSecret = "testSecret";
$account = "test";
$merchant = "test_merchantId";

$paymentApi = new PaymentAPIService($serviceUrl, $signatureKeyId, $signatureSecret, $account, $merchant);
```

Figure 6. Creating new *PaymentAPIService*object

The methods for invoking the HTTP API become available after the PaymentAPIService instance has been established. As seen in Figure 7, each method accepts the additional parameters required to complete a successful HTTP API call.

```
$paymentApi->initTransaction();
$paymentApi->commitFormTransaction( $transactionId, Transaction $transaction );
$paymentApi->debitTransaction( $transactionId, Transaction $transaction );
$paymentApi->revertTransaction($transactionId, $amount);
$paymentApi->statusTransaction( $transactionId );
$paymentApi->tokenize( $tokenizeId );
$paymentApi->getReport( $date );
```

Figure 7. All PaymentAPIService object's publicly callable methods

The commit & debit transaction methods really use a payload model as the transaction model injection. This model is an exact replica of the JSON object that is sent along with the HTTP request to the HTTP API for Payment Highway.

In the illustration in Figure 8, the constructor of the Transaction model accepts a Token model. When utilizing the Transaction model with the commitFormTransaction method, the Token model can be disregarded. This is so that only the transaction number, dollar amount, and currency are required by the commit operation. The token id is passed as a string to the Token model. The token id is a string in UUIDv4 format.

```
use Solinor\PaymentHighway\Model\Request\Transaction;
use Solinor\PaymentHighway\Model\Token;

$token = new Token( $id );
$transaction = new Transaction( $token, $amount, $currency );

$json = json_encode( $transaction );
```

Figure 8. Transaction model forming

The Token object, amount as a numeric string, & currency are all accepted by the Transaction model. The currency is specified as a string that is formatted using ISO-4217, such as "EUR". Euros are the only currency currently accepted on the Payment highway.

The Transaction model can be converted to a JSON string once it has been created. To do this, a JsonSerialize interface is implemented within the Transaction model. The json encode function in the Transaction model executes the JsonSerialize interface's method, which is a standard PHP library interface. The HTTP request body receives the Transaction model once it has been converted to a JSON string. The HTTP request is processed by the PaymentAPISerivce, which then sends the caller a JSON in the response body. Figure 9 displays how to get a result object's values in the JSON that was returned & decoded into a PHP object.

```
// evaluates to "100"
$response->result->code

// evaluates to "OK"
$response->result->msg
```

Figure 9. The caller then receives the response object.

A typical PHP object is returned to the caller as the response object. Any values returned in JSON can be obtained using the same method shown in the previous illustration.

- **Installation**

As a Composer package, the Software Development Kit is made available. To index & list the available packages, the Composer by default utilizes a service called Packagist. Although it is not necessary for the packages utilized by the Composer to be listed on Packagist, doing so makes package adoption & discovery more effective.

Typically, packages are hosted on a platform like Github or Bitbucket. These services provide as platforms for software-as-a-service version control programs like GIT. On Github, the Payment Highway SDK is housed. The system requires a composer.json file to be present in the root of the project folder in order to list a package on Packagist. A summary of the parameters available in the composer.json file is shown in Figure 10.

```
{
  "name": "solinor/paymenthighwayio",
  "description": "Paymenthighway SDK",
  "type": "library",
  "homepage" : "https://paymenthighway.fi/dev/",
  "licence" : "MIT",
  "authors" : [
    {
      "name" : "Jonni Larjomaa",
      "email" : "jonni.larjomaa@solinor.com"
    }
  ],
  "require": {
    "php": ">=5.4.0",
    "ramsey/uuid" : "^2.8",
    "nategood/httpful": "*",
    "respect/validation": "^1.0"
  },
```

Figure 10. Illustration of composer.json file

The Packagist uses the meta-data in this file to index & display the package contents. The require part of this file also contains a list of the dependant packages for this project.

One must add the prerequisite for this library to their project's composer.json file in order to use this library with Composer. The SDK package is included as a dependency in the composer.json files' required section, as seen in Figure 11.

```
"require" : {
  "solinor/paymenthighwayio" : "1.0.0-RC1"
}
```

Figure 11. Requirement added to composer PH SDK for JSON

To install or update the new packages, Composer must be run once the package has been added to the composer.json file. The command shown in Figure 12 can be used to accomplish this on the command line.

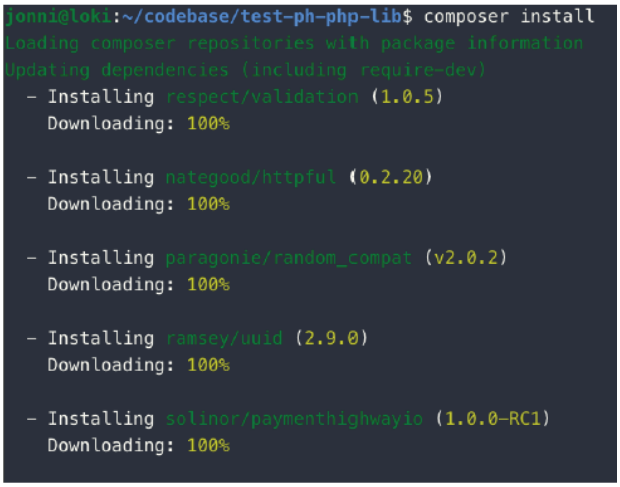

Figure 12. Installing the PH SDK and all of its dependencies using composer.

This example assumes that the Composer is set up and that the executable can be located using the PATH environment variable. The SDK is prepared for usage with the current project once the composer has completed the installation.

RESULTS & DISCUSSION

The major findings of the thesis are described in this chapter. The result is addressed generally, and conclusions regarding the thesis's success are distilled. The components & roles of payment systems are numerous. The payment sector has expanded to become a sizable ecosystem, and numerous players have entered the market for payment technologies. By working together through the PCI-SSC, the card brands manage and govern how the systems involved in the payment scheme interact. The Council is in charge of revising the security regulations applied to the payment systems. The Payment Highway payment gateway system was introduced in this thesis. The Payment Highway was created by SolinorOy, a software business. These payment gateways serve as a middleman for authorizations, credit & debit transactions, and they manage & retain customer credit card data safely. The purpose of this thesis was to create a Software Development Kit to make it easier to integrate the Payment Highway as a payment method for electronic commerce.

Two HTTP APIs are built into The Payment Highway. Payments, authorizations, tokenization, and reporting are all handled through these APIs. The form API processes & collects Cardholder data via an HTML-rendered form. The payment API is utilized as a backend function to commit the transactions entered into the HTML form, retrieve card tokens, create tokenized payments, and retrieve transaction & settlement status information.

A significant amount of specialized integration work was required when new clients were introduced to the Payment Highway, and numerous comparable procedures had to be performed. Since there was no simple way to use APIs programmatically, Solinor was forced to create a Software Development Kit in order to reduce the amount of labor required.

The produced Software Development Kit was created with convention over configuration in mind, and it has a straightforward architecture. Two service classes—one for each API—along with a number of models & utility programs make up the SDK. The SDK is extremely user-friendly and simple to install when all these capabilities are combined. A distribution management tool called Composer that was created for PHP-based projects is used for installation. The primary difficulty encountered throughout SDK development was the constant change in criteria & specifications. This led to the issue of "not getting it done"-conundrum. The SDK was ultimately decided to be released, and the initial version is now accessible through Packagist. The SDK's distribution via Packagist makes it simple for developers to update to the newest releases. Due to the speed with which new security features may be introduced, this SDK has increased the quality of integrations & security of deployments.

CONCLUSION

Payment Gateways are essential for enabling online commerce.Without the development of a secure mechanism for transferring sensitive or secret data, like the details of a credit card, it was challenging to conduct non-face-to-face (impersonal) business transactions in the past at a time when there was no clear way to stop cybercrimes. But things have changed now. Today, there are numerous payment gateways that offer a variety of services & enable credit card transactions online.The purpose of this thesis was to create a Software Development Kit to make it easier to integrate the Payment Highway as a payment method for electronic commerce.In order to include bug fixes & newest functionality, the SDK requires further development. On top of this SDK, the subsequent development phases could involve solution-specific libraries or plugins. The SDK has accelerated accompanying new customers & enhanced integration efforts, which was its intended use.

BIBLIOGRAPHY

1. Aaina Khan.et.al (2017), “Safer E-Wallets”, International Journal of Scientific & Engineering Research Volume 8, Issue 5, pp: 102-104.
2. Aastha Gupta and Munish Gupta (2013), “Electronic Mode of Payment – A Study of Indian Banking System”, International Journal of Enterprise Computing And Business Systems, Volume. 2 issue 2.
3. Abeysinghe S. RESTful PHP Web Services: Packt Publishing Ltd; 2008. Internet Engineering Task Force (IETF). The JavaScript Object Notation (JSON)

4. AbhishekWaghmare (2016). "The year cashless payments trended" upwards Business Stadard.
5. Abid, S. (2016), "Electronic Payment System- An Evolution in Indian Banking System", IOSR Journal of Economics and Finance, Volume. 7, issue 2. pp 25-30.
6. Abrazhevich, D. (2001). "Classification and Characteristics of Electronic Payment Systems". Proceedings of 2nd International Conference on Electronic Commerce and Web Technologies (EC-Web 01). Springer, Heidelberg, LNCS 2115. pp. 81-90.
7. Abrazhevich, Dennis (2004). 'Electronic Payment Systems: A User Centered Perspective and Interaction Design'. Thesis. J.F. Schouten School for UserSystem Interaction Research. The Eindhoven University of Technology. Netherland.
8. Adeniyi, Alao A.; Abeokuta, Sapon and Olutayo, Sorinola O. (2015). "Cashless Policy & Customer's Satisfaction: A study of Commercial Banks in Ogun State". Nigeria' Research Journal of Finance & Accounting. Vol. 6 No. 2. pp. 37-47.
9. Agboola, A. A (2006). "Electronic Payment Systems and Tele-Banking Services in Nigeria". Journal of Internet Banking and Commerce. Vol. 11. No. 3.
10. Al Khatib, Adnan M. (2012). "Electronic Payment Fraud Detection Techniques". World of Computer Science and Information Technology Journal. Vol. 2. No. 4. pp. 137-141.
11. Alaknanda Lonare.et.al (2018) "E-Wallets: Diffusion and Adoption in Indian Economy", Indian Journal of Commerce & Management Studies, Volume IX Issue 2, pp- 9-16.
12. Alam, S.S.; Ali, M.H.; Omar, N.A.; Hussain, W.M.H.W. Customer satisfaction in online shopping in growing markets: An empirical study. Int. J. Asian Bus. Inf. Manag. 2020, 11, 78–91. [CrossRef]

Corresponding Author

Ravin Kumar*

Research Scholar, Computer Science and Engineering, Shri Venkateshwara University, Gajraula, Uttar Pradesh, India