WASP

WASP Gateway™ Product Services Guide for EC Merchants and Processors

WASP™ OEM Payment Platform

Product Services Guide for EC Merchants & Processors ver 6.08 Rel 12-08-24

2024

IMPORTANT NOTICES

Stratsol owns the intellectual property in this document exclusively. You acknowledge that you must not perform any act which infringes the copyright or any other intellectual property rights of Stratsol and cannot make any copies of this Manual unless in accordance with these terms and conditions.

Without our express written consent, you must not:

- Distribute any information contained in this Manual to the public media or quote or use such information in the public media; or

- Allow access to the information in this Manual to any company, firm, partnership, association, individual, group of individuals or other legal entity other than your officers, directors and employees who require the information for purposes directly related to your business.

The software described in this Manual is supplied under a license agreement and may only be used in accordance with the terms of that agreement.

Stratsol, WASP Gateway and the WASP and Stratsol logos are trademarks of Stratsol.

All third-party product and service names are trademarks or registered trademarks of their respective owners.

SUMMARY OF CHANGES

Not Applicable

RELATED DOCUMENTATION

The following documents and manuals provide information related to the subjects described in this guide.

- *The Merchant Agreement for Processing Payments (Merchant Agreement)*
- *Stratsol Merchant Administration User Interface Guide (Merchant Admin Guide)*
- *Stratsol Onboarding Guide for EC Merchants and Processors (Setup Guide)*
- *Stratsol Guide to Using UnionPay Cards (Client information)*
- *Stratsol Merchant Enablement Information Form (Merchant Information)*

1. OVERVIEW

Stratsol has implemented a Payment Gateway facility to enable merchants, payment processors and payment service providers to accept common payment cards for Electronic Commerce (EC) and Point of Sale (POS). This document is specific to EC merchants only.

The service is known as the **WASP Gateway** (or WASP) and provides internet front-end payment and account management functions and an on-line administrative platform to a web hosted application.

The purpose of the WASP gateway is to provide a high function payment service that simplifies the payment acceptance process for merchants.

This document provides a high level overview of the WASP system and should be read in conjunction with the detailed online available API documentation.

WASP GATEWAY BENEFITS

The WASP Gateway provides an outsourced proposition to acquirers, merchants and payment processors (collective referred to as *Payment Service Providers* or PSPs in this document) for this type of service providing:

Significant cost savings – costs incurred for redundant hardware, software, telecommunications and internal operational overheads for an in-house system are avoided.

Faster time to market – WASP makes it possible for a user to be "live" with their service within a much shorter time frame than usual.

Reduced risk – WASP clients avoid many of the risks associated with an in-house system such as high initial and on-going costs; recruitment of skilled technical resources; data security requirements and ability to say up to date with the latest industry requirements, compliance and trends.

Business transparency – WASP is an "out-sourced" packaged service running on Amazon Web Services. All costs are clear to the merchant or processor and agreed up front.

Reduced burden of merchant administration - WASP provides a web-based merchant administration portal for merchants and processors to manage their payment transactions. The system allows merchants to report on transaction types and perform financial transactions on-line such as refunds, voids and reconciliation.

Wide applicability - WASP can be used to enable individual merchants regardless of size and number of transactions. It is an ideal solution for the small to medium enterprise (SME) but can also be readily tailored to suit larger enterprises that operate call centres and web sites to service their clients.

Rapid merchant payment enablement - WASP deploys a flexible connection mechanism based on 'web-services' communication.

Range of payment methods - WASP currently supports credit and debit card payments from many popular card brands and wallets including UnionPay cards and Alipay mobile applications. Transaction types include Purchases, Void and Refunds.

Secure payments - Data passed between the merchant and WASP is encrypted using industry standard cryptography. WASP can reduce the transaction risk by storing card details and other sensitive data on behalf of the merchant.

Future proofing - WASP will continue to be compliant with new industry initiatives.

WASP GATEWAY DESCRIPTION

The following diagram describes the WASP Gateway system and interfaces.

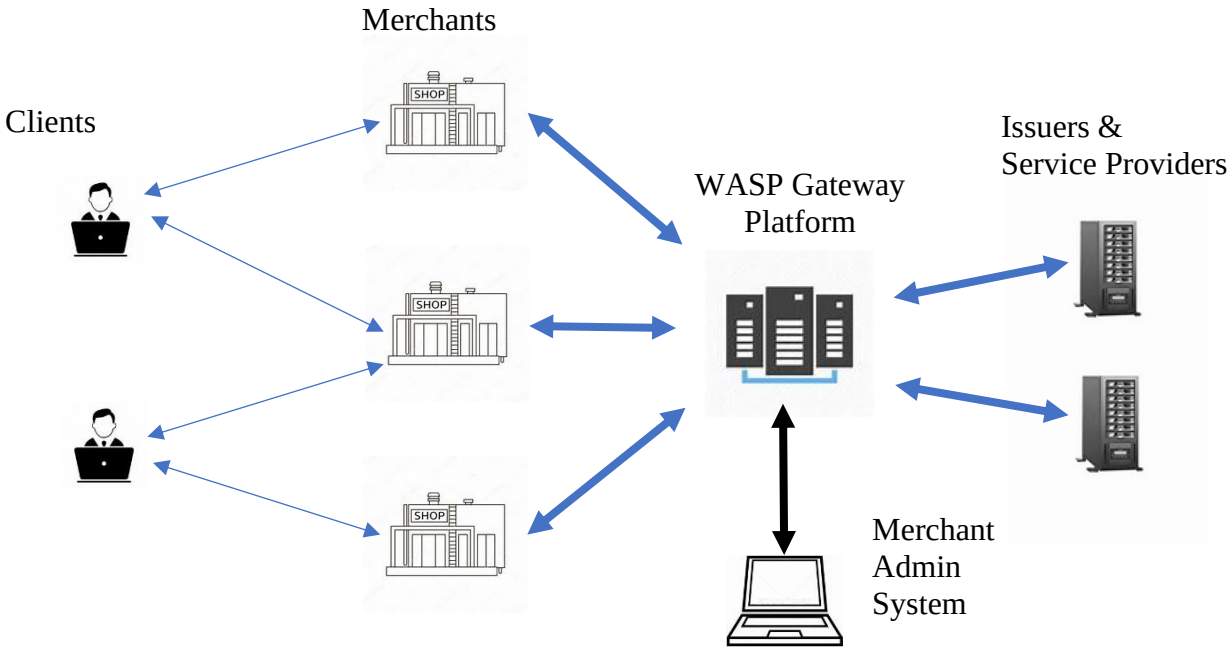

Figure 1. WASP Architecture

Services Provided to the Merchants

The services provided to merchants and processors by WASP through the hosted platform are functionally divided into the following groups:

1. Client Account Management
2. Payment Enablement and Processing
3. Merchant Administration Functions

2. WASP ONLINE PORTAL

The *Merchant Administration* functions are supplied by providing the merchant or processor with secure browser access to the WASP central transaction database to perform day-to-day merchant functions. The user is provided with tools for searching, reporting and editing of client accounts and transactions.

The user documentation for the Merchant Administration functions is in a separate document, the *WASP Gateway Merchant Administration User Interface Guide*.

3. THE WASP API INTERFACE

The *Client Account Management* and *Payment Management and Processing* services are accessible to the merchant and processors (for EC merchants) using a Web-Services style connection mechanism, where messages are exchanged using JSON encoded resources via RESTful APIs over an encrypted channel. This ensures a simpler integration and allows WASP to support several integration environments, such as Microsoft COM and .NET, JSP and PHP, etc.

The following sections describe the API requests PSPs and merchants can initiate to manage their clients on the WASP *Client Account Management* platform, and enable payments through the WASP *Payment Management and Processing* service.

WASP maintains two separate systems – test and production, with different URLs. The endpoint for your API request should be set accordingly. While this document will use the web address *my-gateway.net*, the actual endpoint address may be different and will be provided by your PSP.

API requests are made by

- sending data to the WASP Gateway for action, for example to create new clients on the merchant client management system or to initiate payments. These API calls are made using POST or PUT methods.

- requesting data from the WASP Gateway, for example to retrieve client details from the merchant client management system or query outstanding payment transactions. These API calls are made using the GET method.

WASP is a comprehensive API designed to simplify payment processing by allowing merchants, and aggregators, to initiate payment requests and enabling aggregators to manage their merchant base effectively.

The system provides a robust set of interfaces and subsystems required to support a payment gateway. It facilitates API integration for aggregators, Payment Service Providers (PSPs), and their merchants across multiple payment and banking networks. Additionally, WASP offers a complete back-office reporting, settlement, and reconciliation subsystem, coupled with a merchant-facing portal for administration and visualization of business outcomes.

To monitor the system's health and performance, Prometheus is used to gather and store metrics from the applications and infrastructure. These metrics are visualized in Grafana through customizable dashboards, providing valuable insights into system operations. Furthermore, alerting mechanisms are implemented using AWS SNS to send SMS notifications for critical issues, ensuring prompt response to any potential problems.

The WASP API is RESTful, accepting form-encoded and/or JSON request bodies while using standard HTTP response codes, headers, authentication, and verbs. This ensures a simpler integration and allows WASP to support several integration environments, such as Microsoft COM and .NET, JSP and PHP, etc.

WASP maintains two separate systems – test and production, with different URLs. The endpoint for your API request should be set accordingly.

As an extension to the payment gateway, Wasp also offers an additional API set for dispute management, enhancing its capabilities in handling payment-related disputes.

THE WASP SECURITY KEYS

The WASP Gateway maintains unique security keys for each merchant connected to the system.

When merchants are provisioned onto the system, they are provided with an *API key*, a unique *GUID identifier*, and a *private key phrase*. Both the merchant identifier and API key remain the same across test and production environments; however, the key phrase differs in each system to prevent inadvertent interactions. Notably, interactions in the test environment have no effect on production data and do not interact with banking networks.

Authentication

The WASP API uses API keys to authenticate all administrative requests. While a merchant's API key allows access to the system, it is not used to validate the content of payment payloads.

Importantly, payment requests do not use API keys for authentication, and API keys must not be included in these requests. Instead, payments need to be signed to validate their content, ensure at-most-once delivery, and verify the merchant's identity. Payment requests should never be initiated directly from the merchant's website, as embedding the API key or the key phrase on the website is a direct security violation.

Signature Calculation

The signature is calculated through the following stages:

Stage 1:
Concatenate the UTC Date and Time (e.g., "2023-12-20T21:51:40.197Z"), followed by a new line ("\n"), and the MD5 Hash of the payload/body.

Stage 2:
Perform HMAC256 signing of the Stage 1 result using the merchant's unique key phrase.

Stage 3:
Base64 encode the result of Stage 2.

Authentication Headers

The following headers are used to authenticate requests:

Header Element	Description
X-Api-Key	The API Key

X-Date	The Date and Time of the request in UTC, used to prevent replay attacks. (Must be exactly the same as that used in the signature calculation)
X-Content-MD5	The MD5 Hash of the payload/body as supplied. This is an optional header component used to assist in debugging.
Authorization	The Signature of the request. This value **must** begin with **Wasp** (including the trailing space) followed by the result of Stage 3, above.

API requests lacking proper authentication will fail with a **403 Forbidden** response.

PAGINATION

All API resources that fetch arrays of JSON objects using list API methods use a consistent pagination methodology. For example, you can list Aggregators, Merchants, and Transactions. All list API methods use cursor-based pagination and share a common structure, accepting the following two optional query parameters: *limit* and *next-token*.

Query Parameters

limit *(optional, default is 100)* - Specifies a limit on the number of items to return, ranging between 1 and 100.

next-token *(optional)* – Defines the next item to retrieve from the list being queried. When making a list request, the response will contain a *next-token* field that can be used to fetch the next page of the list. To retrieve the first page, ensure that the *next-token* is not present. If the *next-token* is absent in the response, the list is complete.

VERSIONING

When backwards-incompatible changes are made to the API, a new version is released. The current version is V1. To explicitly request a specific version, use the **X-Accept-Version** header.

WASP API RESPONSES

All WASP API Responses include a status code, signifying whether the call succeeded. These are conventional HTTP response codes to indicate the success or failure of an API request. In general

- Codes in the **2xx** range indicate success
- Codes in the **4xx** range indicate a failure due to the provided information (e.g., a required parameter was omitted)
- Codes in the **5xx** range indicate an error within the processing environment.

Additionally, the response will usually include relevant data or a status message and potentially more detailed information.

THE WASP MODULES

WASP includes modules to enable PSPs to manage Aggregators, Merchants, Transactions, Users and Clients as well as initiate payments and administer the platform.

All of these modules can be accessed via API calls as described above or optionally from the web hosted secure user portal.

The detailed description of the API calls, their parameters and possible return status and values is in the online WASP API documentation, available at our website.