

Airwallex & VGS Implementation Guide

Table of Contents

Table of Contents.....	2
Project Overview.....	3
Data Flow Diagram.....	3
Implementation Steps.....	4
VGS Outbound Integration with Airwallex.....	5
Testing with Sample Code (Python).....	7
Resources.....	8

Project Overview

When integrating with Airwallex, customers may choose to use VGS as their tokenization provider. In this setup, VGS securely collects sensitive PCI data on behalf of Airwallex customers and reveals it to Airwallex during payment processing, thus eliminating any direct exposure to that data. The guide below offers a high-level overview of how to integrate Airwallex with VGS.

Data Flow Diagram

The diagram below illustrates how Airwallex customers would typically integrate with the VGS Vault:





Implementation Steps

- 1 An end user submits their card number, CVV, and any other fields deemed sensitive using the VGS iFrame built with VGS Collect. Once the submission is complete, VGS Collect automatically passes the data to the VGS Reverse Proxy.

Documentation and Guides: [VGS Collect](#)

- 2 The VGS Reverse Proxy (also known as Inbound Route) tokenizes sensitive data and passes it on to the Airwallex customer's server-side. For example, the card number **4111 1111 1111 1111** becomes **tok_live_1234**. This setup can be completed in the VGS Dashboard.

Documentation and Guides: [Reverse Proxy](#)

- 3 The Airwallex customer initiates a request to Airwallex and passes VGS aliases to Airwallex APIs through the VGS Forward Proxy. See more details in the [Outbound Integration with Airwallex](#) section below.

- 4 The VGS Forward Proxy intercepts a request from an Airwallex customer and replaces VGS aliases with the raw sensitive data initially collected from the customer. Thus, when the data reaches Airwallex, its API is able to interact with the raw data.

Documentation and Guides: [Forward Proxy](#)



The World's Leader in
Payment Tokenization



Airwallex

VGS Outbound Integration with Airwallex

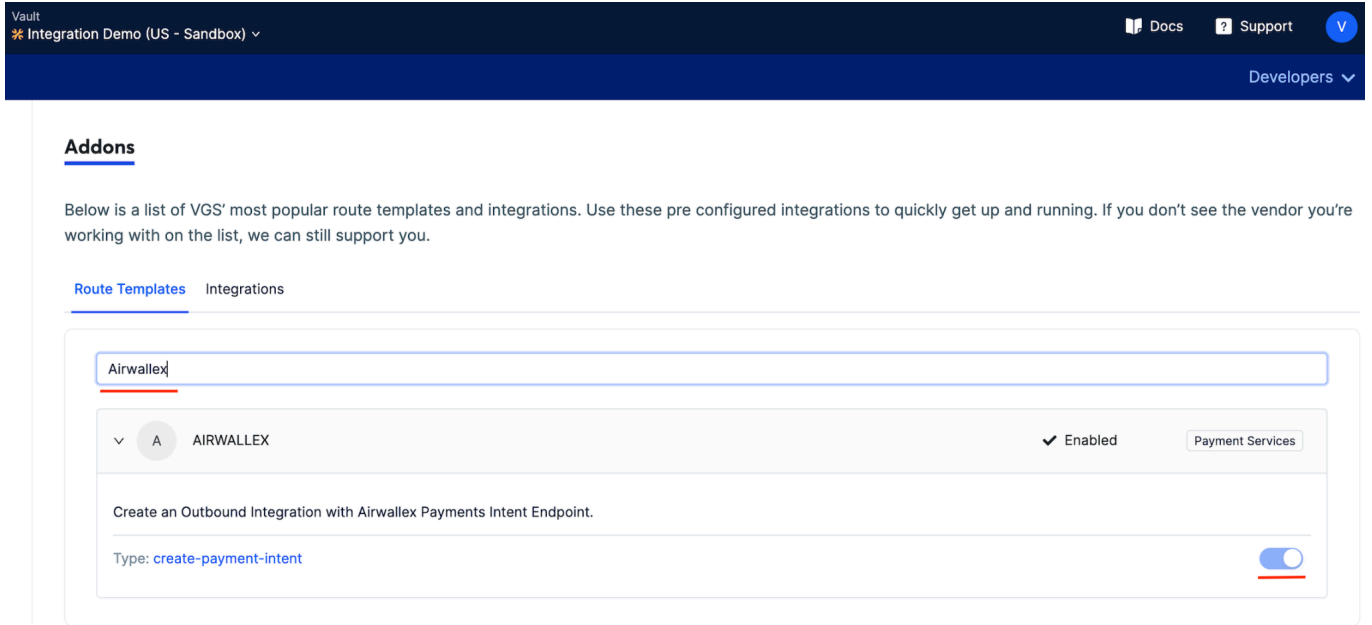
To initiate a transaction, the customer's server-side will need to submit VGS aliases to Airwallex and proxy the request through the VGS Forward Proxy. VGS will intercept this request and swap out VGS Aliases with sensitive data before passing it on to Airwallex.

To accomplish this, customers need to set up an Outbound Route in their VGS Dashboard by enabling an Airwallex add-on:

- Navigate to the "Addons" section within VGS Dashboard

The screenshot shows the VGS Dashboard interface. At the top, there's a header with the VGS logo, 'Organization Demo Organization', and 'Vault Integration Demo (US - Sandbox)'. Below the header is a navigation bar with 'Home' and 'Vault' tabs. The left sidebar contains a list of menu items: 'Vault', 'Getting Started', 'Dashboard', 'Tokenization', 'Routes', 'Form Builder', 'Logs', 'Addons' (highlighted with a red arrow), 'Network Tokens', 'ADMINISTRATION', 'Vault settings', 'Organization', and 'Usage Reports'. The main content area shows two status cards: 'Learn about VGS' (In progress) and 'Get started with inbound routes' (Complete). Below these is a section titled 'Get started with Issue cards' with a list of links: 'Integrate VGS Collect with your application', 'Configure your inbound route', 'Configure your outbound routes for KYC', 'Configure your outbound routes to your issuer platform', and 'Integrate VGS Show with your application'.

- Search “Airwallex” in the search bar
- Press the “Enable” button



Addons

Below is a list of VGS' most popular route templates and integrations. Use these pre configured integrations to quickly get up and running. If you don't see the vendor you're working with on the list, we can still support you.

[Route Templates](#) [Integrations](#)

Airwallex

▼ A AIRWALLEX ✓ Enabled Payment Services

Create an Outbound Integration with Airwallex Payments Intent Endpoint.

Type: [create-payment-intent](#) ☒

As a result, an Airwallex Outbound Route should appear in the “Routes” section of the dashboard:

 airwallex create payment intent 	
UPSTREAM api-demo.airwallex.com	UPDATED 2025-04-07T20:53:52

This route is a pre-built VGS outbound configuration that works with the Airwallex [Create Payment Intent API](#). It will target “**number**” and “**cvc**” fields in the payload to intercept VGS Aliases and replace them with raw PAN and CVC numbers. Given that this Airwallex endpoint is intended to be used for the integration, the setup can be considered complete.

If a customer intends to use another Airwallex endpoint, the integration can be reconfigured or adjusted manually in the settings of this Outbound Route.

For custom Outbound Route configuration setup, see the documentation and guides on the [VGS Forward Proxy](#).

Testing with Sample Code (Python)

The sample code below is a server-side request to Airwallex through VGS. To better understand how this request is constructed, please see the [VGS Forward Proxy](#) documentation.

- Make sure to populate script variables with your VGS and Airwallex credentials and certificates and pass VGS Aliases in the body of the payload.
- To generate VGS Aliases for testing, see the [Vault API](#) and/or the [Reverse Proxy](#) documentation and guides.

```
Python
import os
import tempfile
import requests
from requests import utils
import json

#CONSTANTS
PATH_TO_VGS_CA = <path_to_vgs_certificate>
VGS_USERNAME = <vgs_proxy_username>
```



```
VGS_PASSWORD = <vgs_proxy_password>
VGS_VAULT_ID = <vgs_vault_id>
AIRWALLEX_CLIENT_ID = <airwallex_client_id>
AIRWALLEX_API_KEY = <airwallex_api_key>

#VARIABLES
vgs_card_number_alias = <vgs_card_number_alias>
vgs_card_cvc_alias = <vgs_card_cvc_alias>

#FUNCTIONS
def read_file(path):
    with open(path, mode='rb') as file:
        return file.read()

def airwallex_auth(client_id, api_key):
    url = "https://api-demo.airwallex.com/api/v1/authentication/login"
    headers = {
        "x-client-id": AIRWALLEX_CLIENT_ID,
        "x-api-key": AIRWALLEX_API_KEY,
        "Content-Type": "application/json",
    }
    payload = {}
    response = requests.post(url, json=payload, headers=headers)
    data = response.json()
    token = data.get("token")
    return token

def create_payment_intent(auth_token, vgs_card_number_alias,
vgs_card_cvc_alias):
    url =
"https://api-demo.airwallex.com/api/v1/pa/payment_intents/create"
    proxy = {'https':
f'https://{VGS_USERNAME}:{VGS_PASSWORD}@{VGS_VAULT_ID}.sandbox.verygo
odproxy.com:8443' }
```

```
headers = {
    "Content-Type": "application/json",
    "Authorization": f"Bearer {auth_token}"
}
data = {
    "request_id": "<random_guid>",
    "amount": 10,
    "currency": "USD",
    "merchant_order_id": "Merchant_ID<random_guid>",
    "payment_method": {
        "type": "card",
        "card": {
            "expiry_month": "12",
            "expiry_year": "2030",
            "number": vgs_card_number_alias,
            "cvc": vgs_card_cvc_alias
        }
    }
}
```

#Certificate managment code block. Since VGS certificate is self-signed, it needs to be appended to the client's default CA Bundle.

```
path_to_lib_ca = utils.DEFAULT_CA_BUNDLE_PATH
with tempfile.NamedTemporaryFile() as ca_file:
    ca_file.write(read_file(PATH_TO_VGS_CA))
    ca_file.write(str.encode(os.linesep))
    ca_file.write(read_file(path_to_lib_ca))
    read_file(ca_file.name)
    try:
        response = requests.post(url, json=data, headers=headers,
proxies=proxy, verify=ca_file.name)
        parsed_json = json.loads(response.text)
        formatted_json = json.dumps(parsed_json, indent=4)
        print(formatted_json)
```



```
except Exception as e:
    print(e)

print("Status Code:", response.status_code)

#MAIN
auth_token = airwallex_auth(AIRWALLEX_CLIENT_ID, AIRWALLEX_API_KEY)
create_payment_intent(auth_token, vgs_card_number_alias,
vgs_card_cvc_alias)
```



Resources:

- [VGS Docs](#)
- [Vault API](#)
- [VGS Collect](#)
- [Reverse Proxy](#)
- [Forward Proxy](#)
- [Airwallex Payment Docs](#)
- [Airwallex Payment Intent API](#)
- [Airwallex Test Card Numbers](#)
- [Airwallex Device Fingerprint](#)
- [Airwallex Enhanced Fraud Protection](#)
- [Going Live](#)