GS Fire Store– Integration Guide

Functionality Requirements

- Elements(iframe)
  - Ticket Intention
- CC Sale Ticket
  - Save_account
- Sales on Tokens
  - CC Sale Tokenized
- Recurrings (SaAs fees)
- Refund/Void API Partial Refunds

Fortis API Documentation https://docs.fortis.tech/v/1_0_0#/php/quick-start-guide/overview

PCI Compliance

Each integration requires PCI Compliance for certification to Fortis. Below is the PCI Compliance scope of the project:

Integration Method	PCI Compliance	Hosted Options	Gateway/API Features
Hosted	Out of Scope	Elements (CNP), Tokenization,	CC, Tokens, Recurring, One-Time Payments, Contacts, Webhooks

Merchant Onboarding

Each merchant GS FIRE STORE will be onboarding will be completed by the below method(s)

1. Sales Assist

API Credential Levels

Integration Level – Authentication

1. Developer ID – We issue a unique Developer ID for each partner integration. This ID is required in API call headers and hosted payment page data for authentication and should be hidden from end users.

Merchant Level – Authentication (variable per merchant)

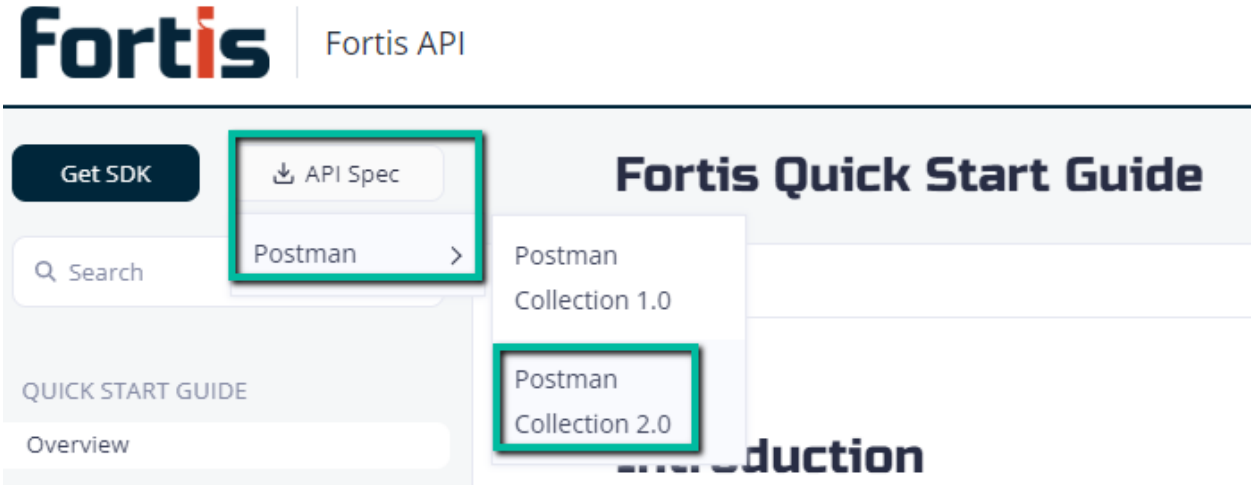
1. API User ID – When a merchant is onboarded, we will create a user specifically for the API integration, this allows us to differentiate from a regular user of the Fortis Merchant Portal.
2. User API Key – Each User has a unique API Key

Merchant Level – Transaction (variable per merchant)

1. Location ID – A merchant may have one or more Locations (physical office or payment channel) and each Location has a unique Location ID.
2. Product Transaction ID- A Location can have one or more settlement accounts (card or ACH). Each settlement account has a unique Product Transaction ID that can be used in a transaction request to process the payment on a specific account.

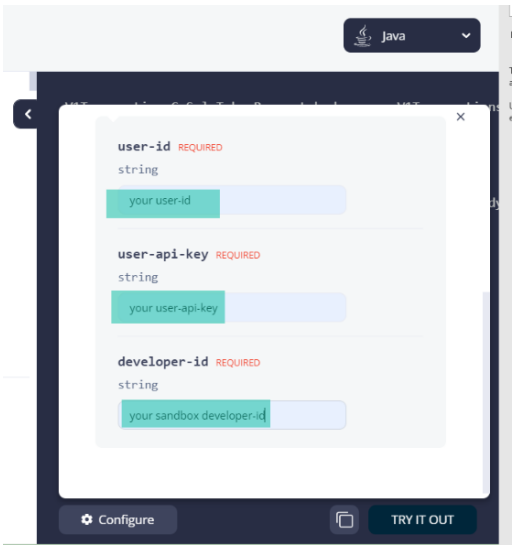
POSTMAN Collections

The API offers up a POSTMAN collection that can be downloaded and imported into your Rest API client tool.



API Code Playground Configuration

The documentation allows integrators to test out the various API calls from within the documentation by a simple configuration setup that needs to include (user-id, user-api-key, developer-id)



Testing Data

Sandbox: https://api.sandbox.fortis.tech/v1/

Testing Card numbers: Use the list of test Card numbers with your requests to trigger controlled error messages.

Transaction Status_id: Status codes are returned in transaction responses and let the integrator know if the transaction is approved or declined.

Reason_Code_id: Reason response codes clarify why a transaction is approved or declined. The verbiage from this list can be mapped to the integrators UI to let merchant know why a card is declined.

Recommended solution: Card not Present, Ecomm

- Elements(iframe)
  - Ticket Intention
- CC Sale Ticket
  - Save_account
- Sales on Tokens
  - CC Sale Tokenized
- Recurrings (SaAs fees)
- Refund/Void API Partial Refunds

How it Works:

GS FIRE STORE will be utilizing the hosted Elements iframe to allow customers to key in card details within the GS FIRE STORE app. GS FIRE STORE will be using the ticket_id's obtained from the Elements transactions to process transactions for the web store checkout.

Elements (hosted iframe used to collect card details)

Docs: https://docs.fortis.tech/v/1_0_0#/java/elements/overview

Elements allows integrators to provide a PCI validated GS Fire Store iframe for customers to input their cardholder details to accept payments within your website. The Elements form keeps the solution out of PCI scope. Along with the GS Fire Storeurity, the iframe is highly customizable to suit your business needs. Please see an example below of a simple payment form.

Elements use a Ticket Intention object to represent your intent to collect CC payments from a customer. Elements has Event Listeners for tracking charge attempts, and payment state changes throughout the process.

These are the steps of an Elements transaction:

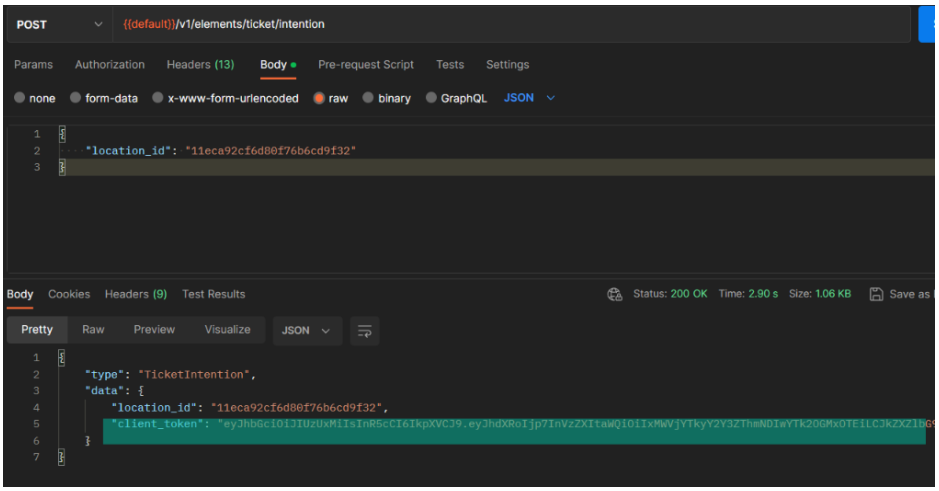
1. GS FIRE STORE sends a TicketIntention request (server to server) to initiate an Elements payment on Fortis.
2. TicketIntention Request Body Considerations
    a. Required Field
        i. Location_id

Ticket Intention

Docs: https://docs.fortis.tech/v/1_0_0.html#/rest/api-endpoints/elements/ticket-intention

POST /v1/elements/ticket/intention to create the one-time use client_token required for the Elements JS script to render the single in

Request/Response



3. Fortis responds to the TicketIntention request with a unique Client Token value for the transaction. This is a one-time use token and is good for 15 minutes.
4. GS FIRE STORE uses the Client Token to open the Elements payments page in an iFrame on the website or within the GS FIRE STORE application to collect the card details.
    a. Set the configuration option field ***view: card-single-field*** for the Single Input Box view in the script.

    b. Details on configuration options are found here: Configurations Options - Fortis API - 1.0.0

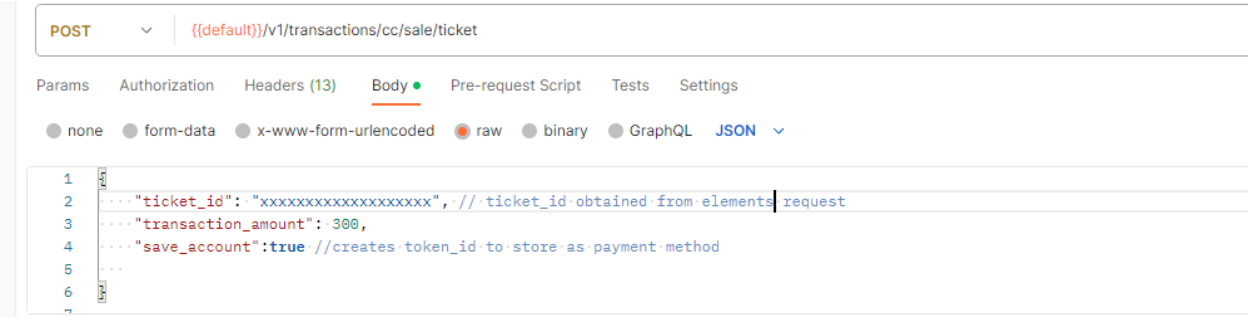
c. The listener for the transaction status is defined in the script. The Done event will contain the ticket_id needed to complete the CC Sale request. The response formats are found here: https://docs.fortis.tech/v/1_0_0.html#/rest/elements/events

5. The customer or staff completes their transaction on the Elements payments page.
   d. CVV, and Billing ZIP validation are applied to transactions if the Auto Decline settings are enabled on the merchant Location.
6. Fortis responds with the ticket_id to the GS FIRE STORE app on completion using the Done event listed in 4b above. The ticket_id is good for one time use and has a 15-minute expiration time.
7. GS FIRE STORE will use the ticket_id value in the done event of the Ticket intention to POST against the CC Sale Ticket endpoint from the server side to charge the card and to get the multi-use token needed for future one-time tokenized payments for the customer bids.

CC Sale Ticket- One time Payments (Server Side Request)

Docs: https://docs.fortis.tech/v/1_0_0#/rest/api-endpoints/transactions-credit-card/cc-sale-ticket

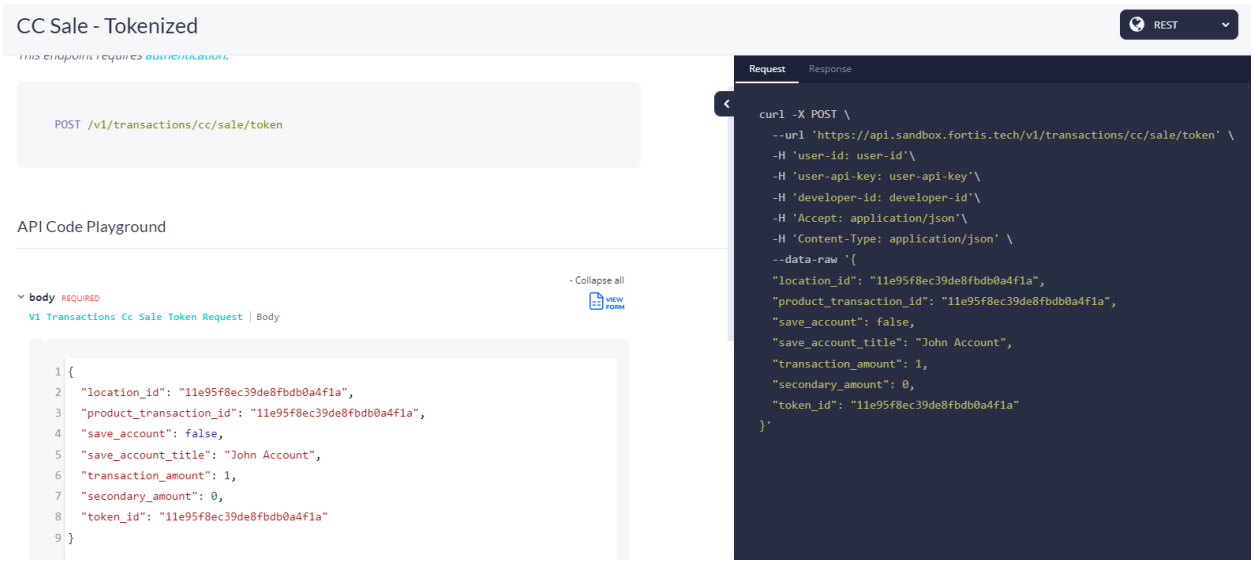
Using the single use ticket_id obtained by the Done Event from the Elements transaction, the solution will now need to process the credit card server side to initiate payments. The CC Sale Ticket request will be used to process the transaction amount on the card. The save_account field, if included will create the multi_use token used to create the token_id to store on the customers profile for future payments. In the response of the request will be the transaction_id along with the status_code(approved/declined) and the reason_code_id (why declined). These codes can be mapped to the error messaging to let users know if transactions are approved or declined.



CC Sale Tokenized- Stored Payment Method

Docs: https://docs.fortis.tech/v/1_0_0#/rest/api-endpoints/transactions-credit-card/cc-sale-tokenized

Once a CC is stored and the account vault token has been created, this token is available to run any transactions that are supported in the Transactions Enpdpoint. The integrators system would be able to POST sale request to the API directly without the need for the Elements form. Since there is no card data on tokens (account vaults), the account vault sale is out of PCI scope.



Recurring Payments(Collect SaAs fees)

Docs: https://docs.fortis.tech/v/1_0_0#/rest/api-endpoints/recurring/create-a-new-recurring-record

The recurrings endpoint is used to run a recurring transaction one or more times. There are two different types of recurrings:

ongoing (recurring_type_id="o") - a recurring that runs until it is deleted, or an end date has been set.

installment (recurring_type_id="i") - a recurring that runs a fixed number of times, regardless of approval or decline.

When setting up a reccuring, it isn't necessarily linked directly to a contact, it is linked to an account vault through the account_vault_id or account_vault_api_id field. The account vault is then in turn linked to a contact (if using contacts). So, to create a recurring, it is recommended to create a contact, then create an account vault for that contact. Then the id of the account vault can be used for the recurring as account_vault_id.

Reporting

Docs: https://docs.fortis.tech/v/1_0_0#/rest/api-endpoints/transactions-read/list-transactions

The Fortis API offers robust reporting features that are available for integrators looking to incorporate their transaction reporting and customer stored payment methods. The API has various filters and expands that can be included in queries to the various endpoints.

Transaction Reporting

Transactions can be looked up at any time with a GET request to the transactions read endpoint. The transaction_id is required to look up individual transactions.

Single Transaction: https://docs.fortis.tech/v/1_0_0#/rest/api-endpoints/transactions-read/get-transaction

List All Transactions

Docs: https://docs.fortis.tech/v/1_0_0#/rest/api-endpoints/transactions-read/list-transactions

This endpoint will allow integrators to pull all transactions from each location. You can include the various filters for each field when querying the data.

Example: Query yesterdays approved transactions for a location

GET /v1/transactions?filter[created_ts]=yesterdays&filter[status_code]=101&filter[location_id]={location_id}

Stored Token Lookup

Example: Query all customers stored Credit Card token details. The data can be used to display the last 4, exp_date and card type in the UI. Swap out cc for ACH to see ACH token details on the location.

GET /v1/tokens?filter[location_id]={location_id}&filter[payment_method]=cc