The general idea of registration:

If the merchant dashboard user goes to the merchant dashboard website he has two options:
1. to register a new merchant account
2. if an invitation from the merchant to join is present, he is able to register an account and after registering view/update the merchant data.

So, the first screen, which the not logged-in user sees when opening the merchant dashboard webpage is some sign-up window, where he can choose either to register a new merchant or join a merchant.

If the merchant dashboard user is registering a new merchant after all sign-up steps a new window pops up, where he has to fill in the Merchant Details. (Either after the user is done with that WP BE sends API-key on FE for the merchant or before that WP BE will send FE API key to put it into one field of Merchant Details)

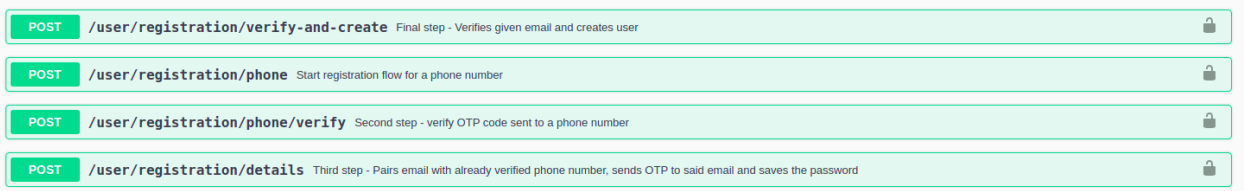
If merchant details are not properly filled in, the merchant will not be able to use his account.

Sign-up steps:
The same steps we use for registering a user with a change, that on the last fourth step of registration you have to pass a parameter to indicate what kind of user merchant dashboard FE is trying to register.

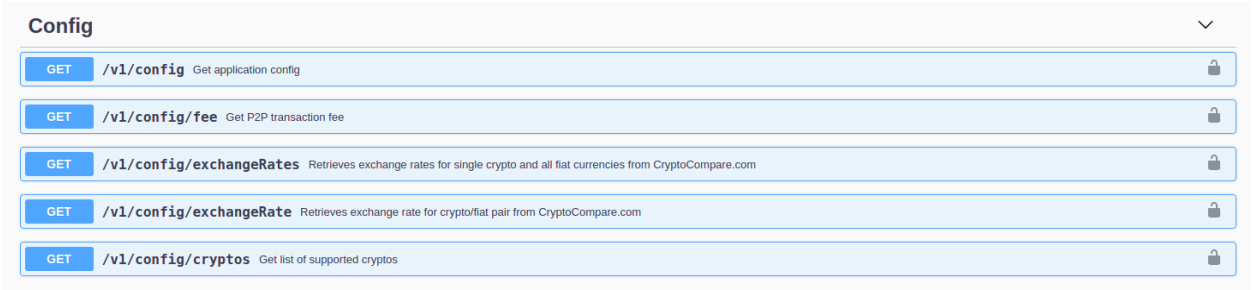
This parameter you should pass together with the other required parameters as a separate field '**isMerchantAdmin'** in the JSON payload of the fourth registration step. If the user was invited to join a merchant organization, then it will be handled by BE and the user will have access to the merchant organization after going through all 4 steps. If **isMerchantAdmin** was specified as true, then you'll be registered as merchant admin. But after it, you also have to log in and call **/merchantDashboard/admin/merchant** to finish the registration of a

merchant associated with the merchant admin user, which was just registered.

Current steps are documented in Swagger.
https://api.wadzpay.com/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config



Login steps:
You are authenticating via OAuth and Cognito using Amplify. After that, you receive a JSON with tokens (You take the value of **IdToken** from it) and you attach a header to each of your HTTP requests to WP BE ‘Authorization: Bearer ${value of IdToken}’
Endpoints:
Generally, endpoints that you can use will be stored with **/merchantDashboard** prefix. But you also can use config endpoints documented in Swagger. Also for sign-up, you have to use endpoints that start with **/user**
https://api.wadzpay.com/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config



/merchantDashboard/balances (accessible to both of the roles)

Example response:
{

‘BTC’ : 3.5,
‘ETH’ : 0.8,
‘USDT’: ‘10000.57’
}
/merchantDashboard/transactions
/merchantDashboard/transactions/{transactionId}
/merchantDashboard/transactionReports (accessible to both of the roles)

Returns a JSON list with a lot of transaction details.
Also, you can specify additional request parameters, like amountFrom, etc. Take a look at what you can specify there:
https://api.dev.wadzpay.com/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config#/User%20account%20and%20merchant%20methods/getTransactions_1

Transaction reports are basically the same as transactions but will throw a generated CSV file at you.

POST /merchantDashboard/admin/invite (accessible to ADMIN only)

Invites user with a specified email to join as ADMIN or READER
Payload:
JSON {
email: “user@example.com”,
role: “MERCHANT_READER” or “MERCHANT_ADMIN”
}

GET **/merchantDashboard/admin/invite** (accessible to everybody)

Request parameter: email

Response: merchant name and role if this email is invited.

API-keys endpoints:

Note that for security reasons, we don’t store API keys anywhere. We store only the hashes of them. So the merchant dashboard should emphasize that you won’t be able to get the API key from the server again if you lose the API key.

POST /merchantDashboard/admin/api-key (accessible to ADMIN only)

Issues new API-key

DELETE /merchantDashboard/admin/api-key (accessible to ADMIN only)

Deletes the API key based on the **username** field specified from the response of the previous endpoint.

POST /merchantDashboard/admin/enable

Reenables user

POST /merchantDashboard/admin/disable

Disables user

/merchantDashboard/admin/sendExternal

Sends transaction to an external wallet

/merchantDashboard/admin/p2p_transaction

Sends P2P transaction

/merchantDashboard/addresses

Get addresses for external receive

Roles:
There are two roles MERCHANT_READER, MERCHANT_ADMIN. Their usage was described in the previous paragraphs more or less.

P.S. Use Swagger to see the endpoint payloads. For **/merchantDashboard/**** they are almost the same as for **/user/**** or **/v1/merchant/**. If you have any questions, please feel free to ask them.**