

Jason – W4 – 2022.07.09

# WebSocket

- 1. WebUI 中 WsNotificationDispatcher.js 中需要新增CASE, 尚待實作中的項目 (如果const 中沒有設定，是要自己新增const)
- 2. Websocket 是從 q8server(伺服器後端) 往 WebUI (前端畫面) 傳送  
主要任務是要告訴 WebUI 這個 Websocket 內容是做什麼用的
- 3. 例如：type = TerminalStatus 就是用來改變 terminal 狀態
- 4. 目前的Status 中，沒有能夠反應terminal 出現錯誤的Status，所以需要建立一個新的Status，在terminal出現錯誤時，及時反應在WebUi的畫面上，來即時提醒terminal 管理者進行錯誤排除。

```
31      switch (wsType) {  
32      case TERMINAL_STATUS: {  
33          store.dispatch({  
34              type: WS_NOTIFICATION_TERMINAL_STATUS,  
35              payload: wsPayload,  
36          });  
37          break;  
38      }
```

# WebSocket

5. 目前在WsNotificationDispatcher.js中完成的，如下圖  
(但大多數Websocket的資料進來的方式都長一樣)

```
WsNotificationDispatcher.js M X
src > middleware > WsNotificationDispatcher.js > default > <function> > <function> > payload > Con
23 export default (store) => (next) => (action) => {
24   let state = store.getState();
25   let { type, payload } = action;
26   let rst = next(action);
27
28   switch (type) {
29     case WS_NOTIFICATION: {
30       let { type: wsType, payload: wsPayload } = payload;
31       switch (wsType) {
32         case TERMINAL_STATUS: {
33           store.dispatch({
34             type: WS_NOTIFICATION_TERMINAL_STATUS,
35             payload: wsPayload,
36           });
37           break;
38         }
39         case RDS_SERVER_STATUS: {
40           store.dispatch({
41             type: WS_NOTIFICATION_RDS_SERVER_STATUS,
42             payload: wsPayload,
43           });
44           break;
45         }
46         case APPLICATION_STATUS: {
47           store.dispatch({
48             type: WS_NOTIFICATION_APPLICATION_STATUS,
49             payload: wsPayload,
50           });
51           break;
52         }
53       }
54     }
55   }
56 }
```

```
WsNotificationDispatcher.js M X
src > middleware > WsNotificationDispatcher.js > default > <function> > <function>
53
54   case PENDING_TERMINALS: {
55     store.dispatch({
56       type: WS_NOTIFICATION_PENDING_TERMINALS,
57       payload: wsPayload,
58     });
59     break;
60   }
61   case TERMINAL: {
62     store.dispatch({
63       type: WS_NOTIFICATION_TERMINAL,
64       payload: wsPayload,
65     });
66     break;
67   }
68   case TERMINAL_LIST: {
69     store.dispatch({
70       type: WS_NOTIFICATION_TERMINAL_LIST,
71       payload: wsPayload,
72     });
73     break;
74   }
75   case TERMINAL_ERROR: {
76     store.dispatch({
77       type: WS_NOTIFICATION_ERROR_TERMINALS,
78       payload: wsPayload,
79     });
80     break;
81   }
82 }
```

# Reducer

- `src/reducers/TerminalReducer.js`  
(reducer, 會改變 state, 後續引發 `render()`, 意思就是可讓圖示發生變化)  
例如：以下的雷達例子

```
190     case WS_NOTIFICATION_PENDING_TERMINALS: {  
191         return update(state, {  
192             pendingTerminals: { $set: payload },  
193         });  
194     }
```

代表 component 那要用到 `pendingTerminals` 的資料，是由 Websocket 的 `payload` 傳進來的

[上述對應的 component 可以參考 `src/components/Header/index.js`  
`pendingTerminals: state.terminals.pendingTerminals`]

# Reducer

可以研究

WS\_NOTIFICATION\_TERMINAL\_STATUS

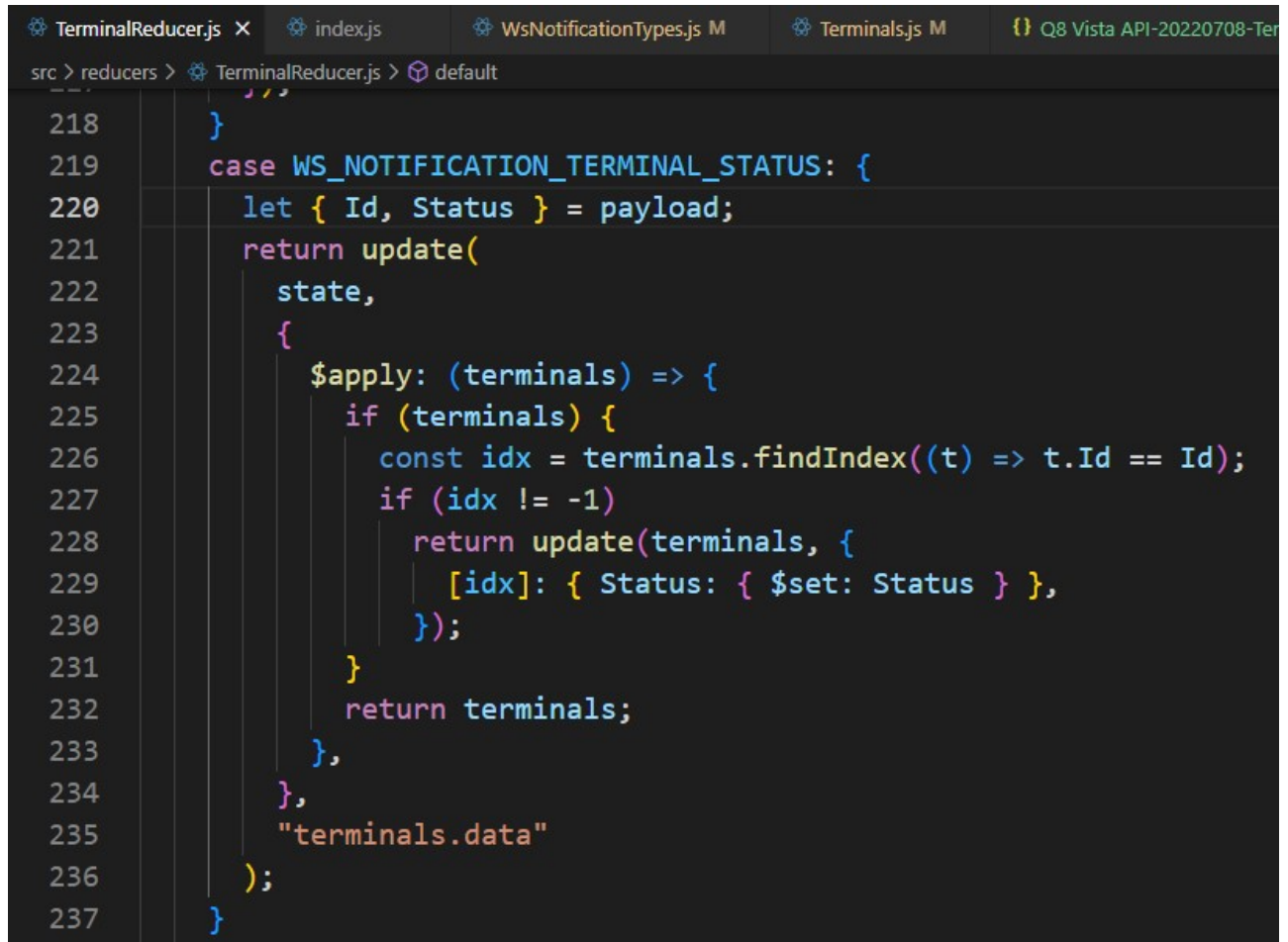
這其中，當websocket將資料傳送進來時，要先找到是哪一個terminal

(因為terminal 有多個，id=1、id=2、id=3...等等的terminal)

右圖，第226行

就是在判斷，要變更資料的websocket 傳送進來資料中的terminal id 是否存在

第227~229行，然後才能夠變更terminal 的status資料



```
218 }
219 case WS_NOTIFICATION_TERMINAL_STATUS: {
220   let { Id, Status } = payload;
221   return update(
222     state,
223     {
224       $apply: (terminals) => {
225         if (terminals) {
226           const idx = terminals.findIndex((t) => t.Id == Id);
227           if (idx != -1)
228             return update(terminals, {
229               [idx]: { Status: { $set: Status } },
230             });
231         }
232         return terminals;
233       },
234     },
235     "terminals.data"
236   );
237 }
```

# postman

- 通常右邊這種API.json 的文件，是給人類看的，就是前端與後端溝通，並照著文件中的內容來實作，也可以視作共同開發專案的規範
- 可以用postman 來打開這個api.json 檔來檢視其內容



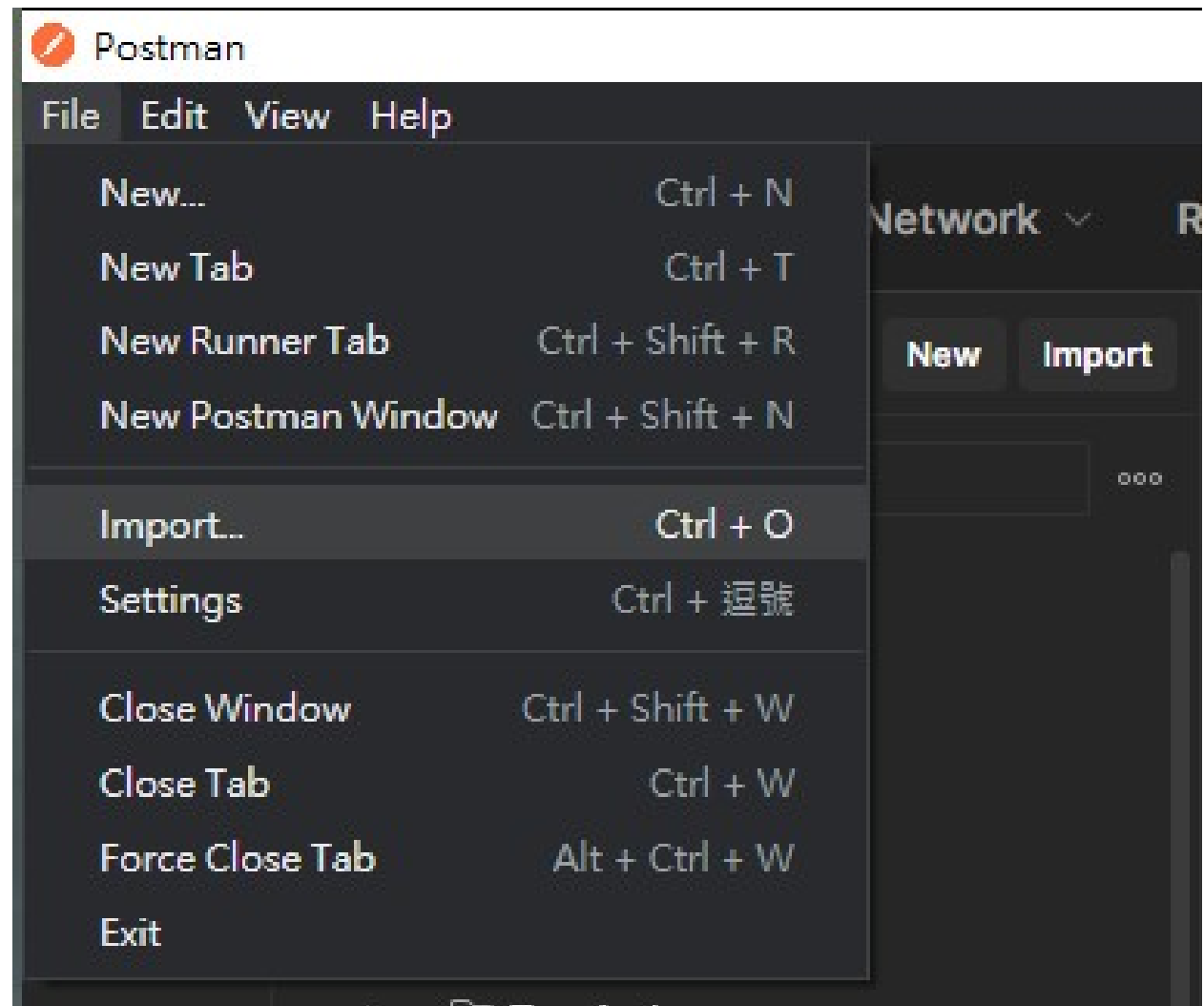
Q8 Vista API-20220708-TerminalError.json

# postman

使用匯入的方式

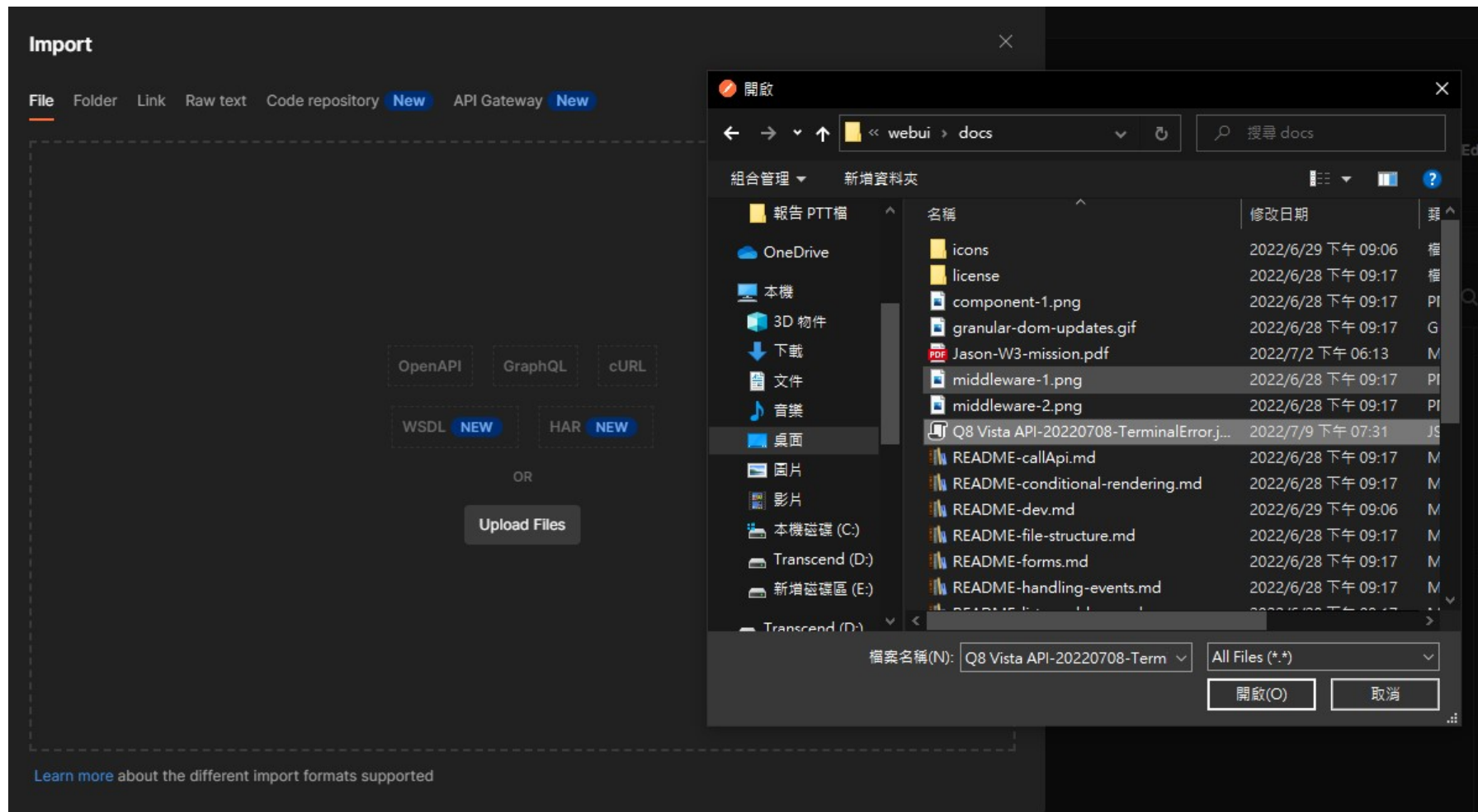
1. 點選 File

2. 點選import



# postman

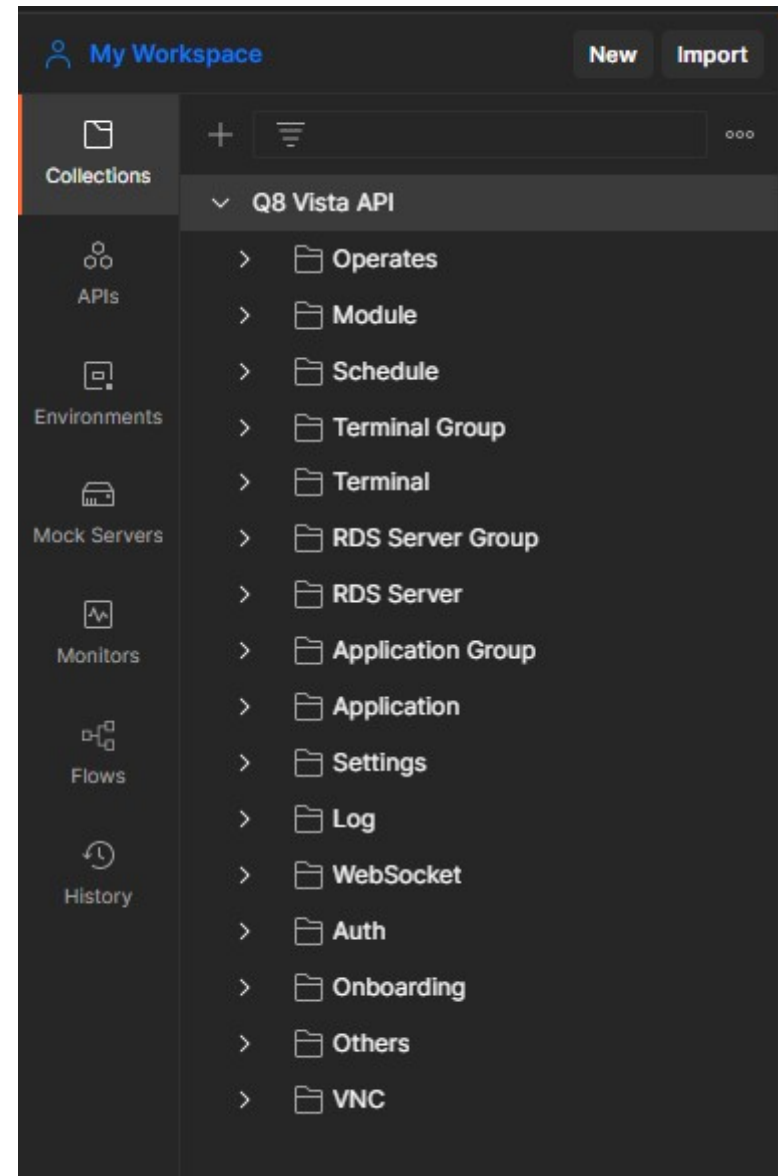
## 3. Upload Files，並且點選要瀏覽的API.json 檔





# postman

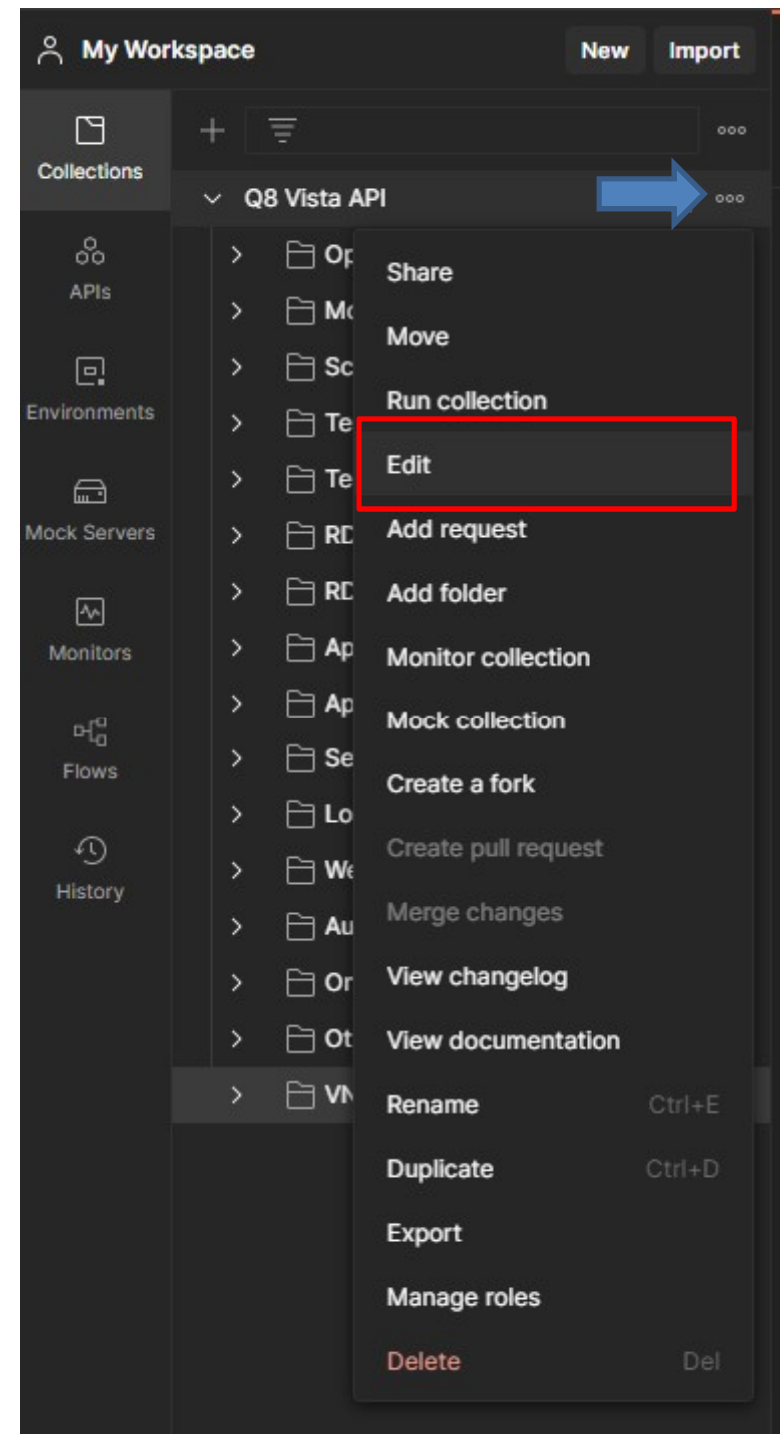
4. Import 之後，就能夠看到如右圖的list



# postman

5. 點擊API list 旁邊3個點的按鈕

6. 點擊 Edit



# postman

7. 選擇Variables

8. 可以注意到  
列表中有一個  
server\_ip:port 的項目

The screenshot shows the Postman interface with the 'Q8 Vista API' collection selected. The 'Variables' tab is active, displaying a list of variables. A blue arrow points to the 'Variables' tab. The 'server\_ip:port' variable is highlighted with a red box.

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist
<input checked="" type="checkbox"/>	base_url	https://192.168.20.68:8085	https://192.168.20.68:8085		
<input checked="" type="checkbox"/>	terminal_id	1	1		
<input checked="" type="checkbox"/>	app_id	1	1		
<input checked="" type="checkbox"/>	rds_server_id	1	1		
<input checked="" type="checkbox"/>	manufacturer	Arista	Arista		
<input checked="" type="checkbox"/>	model	BoxPC138G	BoxPC138G		
<input checked="" type="checkbox"/>	schedule_id	1	1		
<input checked="" type="checkbox"/>	module_id	1	1		
<input checked="" type="checkbox"/>	screen_id	1	1		
<input checked="" type="checkbox"/>	log_start	2021-05-14	2021-05-14		
<input checked="" type="checkbox"/>	log_end	2021-05-17	2021-05-17		
<input checked="" type="checkbox"/>	log_level	Info	Info		
<input checked="" type="checkbox"/>	log_class	terminal	terminal		
<input checked="" type="checkbox"/>	dhcp_id	1	1		
<input checked="" type="checkbox"/>	ad_server_id	1	1		
<input checked="" type="checkbox"/>	user_id	1	1		
<input checked="" type="checkbox"/>	server_ip:port	192.168.1.119:8085	192.168.1.119:8085		
<input checked="" type="checkbox"/>	terminal_client_ip	192.168.20.169	192.168.20.169		
<input checked="" type="checkbox"/>	terminal_group_id	1	1		
<input checked="" type="checkbox"/>	rds_server_group_id	1	1		
<input checked="" type="checkbox"/>	app_group_id	1	1		

# postman

9. 選擇Q8 Vista API 主資料夾

10. 選擇WebSocket資料夾

11. 選擇 Notifaction

12. 選擇  
wss://ws/notifications

13. 選擇TerminalError  
就可以看到如右畫面

