RFID_app Documentation – 31/8/2022

Contents

<u>Node – red Installation</u>
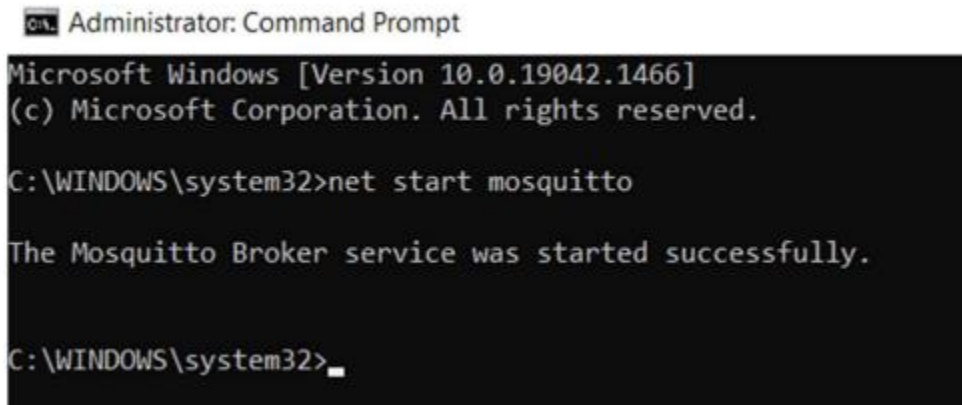
Prerequisite: Node.js is require to be installed.

If node.js is not installed, Install the .msi installer for node.js.

1. To install on Windows, open Command Prompt and enter
   npm install -g –unsafe-perm node-red
2. Once installation is complete, run Node red by typing the follow command in Command Prompt
   node-red
3. Node – red editor can be accessed by entering the follow into a browser link.
   https://<ip-address>:1880 OR https://localhost:1880/
4. Navigate to the three lines on the top right corner of Node-Red and select "Import".
5. Click "select a file to import" then choose the click Import. Download and import MainFLow from
   https://github.com/msf4-0/RFID-ERPNext-System
6. Navigate to the three lines on the top right corner of Node-Red and select "manage Palette". And install any unrecognized types. Deploy to check if any palette is still missing.

MQTT Setup and Installation

MQTT, in simple terms, is a messaging protocol through either network or internet

1. To install MQTT follow this link
   https://mosquitto.org/download/
2. To start Mosquitto MQTT service, run Command Prompt as **Administrator** and enter
   net start mosquito

3. To verify if MQTT is connected the MQTT nodes will turn green and show connected

   Eg:
4. If not you may need to change the mosquito.conf file
   Add this 2 lines
   listener 1883

   and allow_anonymous true

   Restart mosquito service
5. If still not connected, go to firewall and change its setting

ERPNext
Prerequisite: Docker Desktop is required to be installed

1. Follow this link
   GitHub - msf4-0/Integrated-Resource-Planning-System-IRPS: Integrated using ERPNext, Frepple,
   Metabase, Telegram, and Barcode Scanning System

Building RFID_app

1. Install Android Studio
   https://developer.android.com/studio
2. Download the RFID_app.ZIP from the github link
   https://github.com/msf4-0/RFID-ERPNext-System
3. Open project in Android Studio
4. An option to upgrade gradle file is presented, do not upgrade to the latest and set the gradle version at the minimum possible version with no build error.
5. Set your IP address at MainActivity – myIP then click run to download the app onto selected device.

Notes for RFID_app:

1. The App is developed on top of an SDK provided, fully in Java only.
2. The App was developed with none prior knowledge of Java and OOP.
3. The reason most of the code is in RED font is due to those functions being deprecated therefore not being used in the latest version of Android Studio anymore. However, the file can still be built and used during the time it is being developed.
4. The main interface of the app is hidden and shown using setVisibility instead of Viewpager due to inexperience. Listview is used instead of RecycleView due to inexperience.
5. The main app interface developed includes: MainActivity, new_ERP_1, new_ERP_2_A, new_ERP_3, new_ERP_3_2.

Node-red Flow

Introduction

Node-red handles most of the data formatting, where it extracts data from the API payload from ERPNext and formats it into string to be sent through MQTT to the app. Vice versa from the app to ERPNext, string to object and array data.
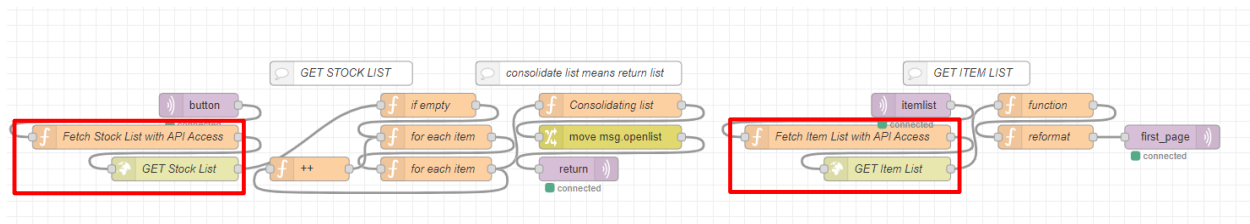
Token

Token is the "key" or "password" which is unique to your ERPNext account, that allows access to your data in the account.
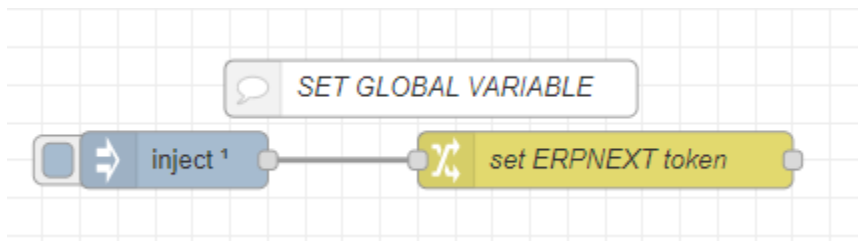

Get Item/Stock List

API Access node defines the type and content of the message to be sent attached with a payload, the https request then send the message to the ERPNext host. In this case the payload does not matter.
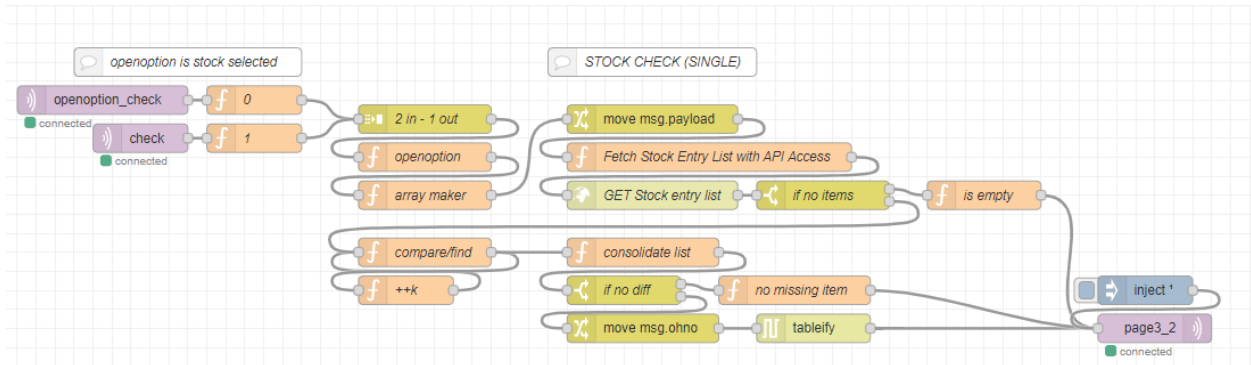
Get Stock Entry List

1. Node **1** puts aside the topic which the message is received so it can be used at step 4.
2. As soon as the node **2 in - 1 out** receives 2 payload messages, in which the 2 messages are stock name and scanned items, then only an output will be passed.
3. The **Item List** is fetched and compared for any missing items and identical items.
4. The **Stock Entry** is fetched and compared for any missing stocks and identical stocks.
5. They are labeled according to the topic which the message is received initially, then all items is combined, and finally returned to the App to be displayed.
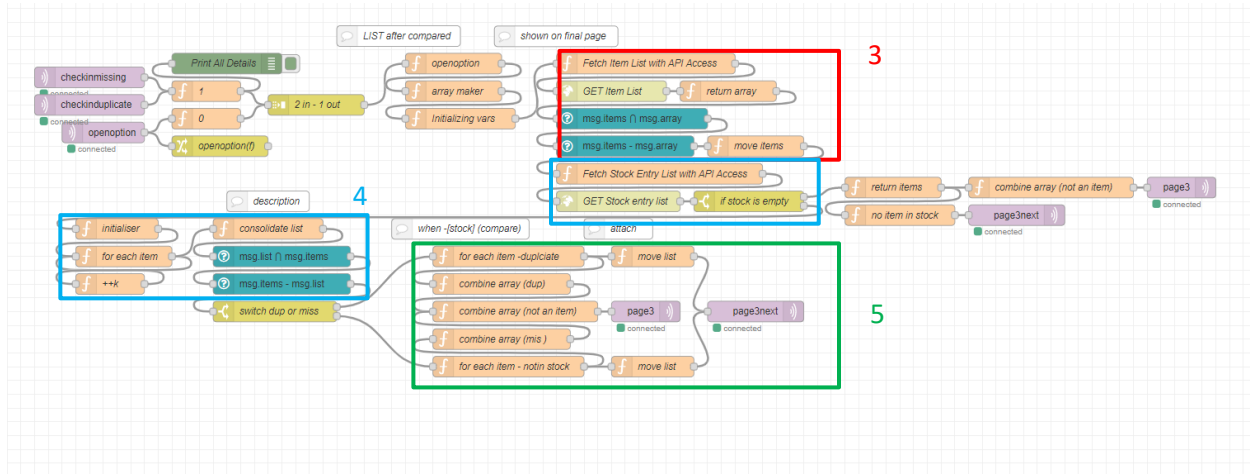
Check missing

Fetch Stock Entry List and **compare** against received scanned items. Returns HTML string instead of json string, the app then displays it using WebView.

Check in

1. Replace extra data with empty. Compare against original items. Items with no extra data is moved forward.
2. Original Stock Entry List is fetched and new item is appended to the end of it. The new Stock Entry List is sent to ERPNext which replaces the old Stock Entry list using **PUT API**. Then the reply is sent to the App.
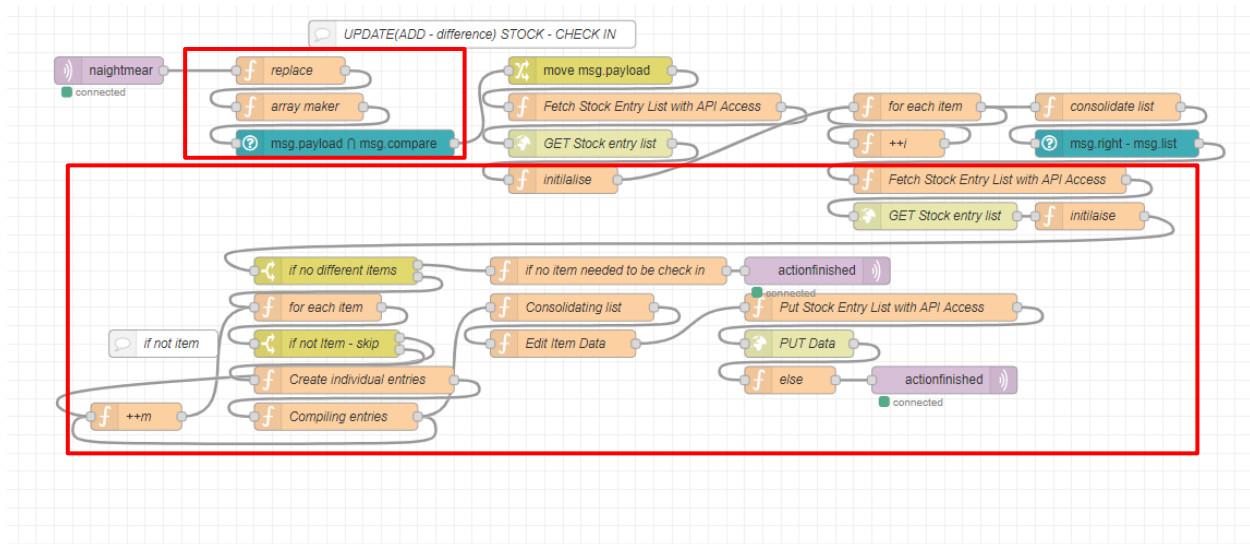
Check out

1. Replace extra data with empty. Compare against original items. Items with no extra data is moved forward.
2. Compare against original Stock Entry List and check for any same entries and remove them.
3. The new Stock Entry List is sent to ERPNext which replaces the old Stock Entry list using **PUT API**. Then the reply is sent to the App.