Currency Tracker - Project Documentation

Lithara Perera

Table of Contents

1. Deployment Guide

1.1 Prerequisites

Before deploying the Currency Tracker solution, ensure you have:

- **Microsoft 365 Account** with Power Automate Premium license (or trial)
- **SharePoint Online** access
- **Office 365 Outlook** for email notifications
- **ExchangeRate-API** account and API key
- **Admin/Maker permissions** in Power Platform environment

1.2 Step-by-Step Deployment

Step 1: Create SharePoint List

1. Navigate to your SharePoint site
2. Click **New** → **List**
3. Name the list: **CurrencyRates**
4. Add the following columns:

Column Name	Type	Settings
Title (default)	Single line of text	Rename to "Currency"
Rate	Number	Decimal places: 2
Date	Date and Time	Include time
Threshold	Number	Decimal places: 2
AlertTriggered	Yes/No	Default: No
PreviousRate	Number	Decimal places: 2

5. Add initial data:

- Currency: **USD**, Threshold: **330**
- Currency: **EUR**, Threshold: **370**
- Currency: **GBP**, Threshold: **420**

Step 2: Obtain API Key

1. Visit ExchangeRate-API.com
2. Sign up for a free account
3. Copy your API key from the dashboard
4. Note: Free tier provides 1,500 requests/month

Step 3: Create Power Automate Flow

A. Create Main Currency Tracker Flow:

1. Go to Power Automate
2. Click **Create** → **Scheduled cloud flow**
3. Name: **Daily Currency Tracker**
4. Schedule: **Daily at 9:00 AM** (adjust time zone to IST)
5. Click **Create**

B. Add Actions in Order:

1. **HTTP Actions (3x - for each currency):**

- Add **HTTP** action
- Method: **GET**
- URI for USD: *https://v6.exchangerate-api.com/v6/YOUR_API_KEY/latest/USD*
- URI for EUR: *https://v6.exchangerate-api.com/v6/YOUR_API_KEY/latest/EUR*
- URI for GBP: *https://v6.exchangerate-api.com/v6/YOUR_API_KEY/latest/GBP*
- Replace YOUR_API_KEY with your actual API key

2. **Parse JSON Actions (3x - for each currency):**

- Add **Parse JSON** after each HTTP action
- Content: body('HTTP_-_USD') (adjust for EUR/GBP)
- Schema:

```
{
  "type": "object",
  "properties": {
    "result": {"type": "string"},
    "documentation": {"type": "string"},
    "terms_of_use": {"type": "string"},
    "time_last_update_unix": {"type": "integer"},
    "time_last_update_utc": {"type": "string"},
    "time_next_update_unix": {"type": "integer"},
    "time_next_update_utc": {"type": "string"},
    "base_code": {"type": "string"},
    "conversion_rates": {
      "type": "object",
      "properties": {
        "LKR": {"type": "number"}
      }
    }
  }
}
```

Figure 1 - Parse JSON Schema

3. Initialize Variables (9x total):

- **USDRate** (Float): body('Parse_JSON_-_USD')?['conversion_rates']?['LKR']
- **EURRate** (Float): body('Parse_JSON_-_EUR')?['conversion_rates']?['LKR']
- **GBPRate** (Float): body('Parse_JSON_-_GBP')?['conversion_rates']?['LKR']
- **CurrentDateUSD** (String): utcNow()
- **CurrentDateEUR** (String): utcNow()
- **CurrentDateGBP** (String): utcNow()
- **USDThreshold** (Float): 0 *(Initialize at top level)*
- **EURThreshold** (Float): 0 *(Initialize at top level)*
- **GBPThreshold** (Float): 0 *(Initialize at top level)*

4. Get Items from SharePoint (3x):

- Add **Get items** action (SharePoint)
- Site Address: Your SharePoint site
- List Name: **CurrencyRates**
- Filter Query for USD: Currency eq 'USD'
- Filter Query for EUR: Currency eq 'EUR'
- Filter Query for GBP: Currency eq 'GBP'

5. Apply to Each - USD Processing:

- Add **Apply to each** control
- Select output from: value (from Get items - USD)
- Inside the loop:
    - **Set variable** - USDThreshold: item()?['Threshold']
    - **Condition**: variables('USDRate') is greater than variables('USDThreshold')
    - **If Yes:**
        - **Send an email (V2)**:
            - To: Your email address
            - Subject: 🚨 USD Rate Alert - Threshold Crossed!
            - Body:
              USD Rate: @{variables('USDRate')}
              Threshold: @{variables('USDThreshold')}
              Date: @{variables('CurrentDateUSD')}
        - **Send an email (V2)**:
            - ID: `item()?['ID']`
            - Rate: `variables('USDRate')`
            - Date: `variables('CurrentDateUSD')`
            - AlertTriggered: Yes

6. **Repeat Step 5 for EUR and GBP**

C. Save and Test the Flow:

1. Click **Save**
2. Click **Test** → **Manually**
3. Verify all actions complete successfully
4. Check your email for alerts (if thresholds are crossed)
5. Verify data is updated in SharePoint list

Step 4: Create Data Cleanup Flow

1. Create another **Scheduled cloud flow**
2. Name: **Currency Data Cleanup**
3. Schedule: **Weekly on Sunday at 12:00 AM**
4. Add **Get items** action:
    - Site Address: Your SharePoint site
    - List Name: **CurrencyRates**
    - Filter Query: Date lt '@{addDays(utcNow(), -7)}'
5. Add **Apply to each**:
    - Select output: value
    - Inside loop: **Delete item** (SharePoint)
        - ID: item()?['ID']
6. Save and enable the flow

Step 5: Create Dashboard

Power Apps Canvas App

1. Go to Power Apps (make.powerapps.com)
2. Create → Canvas app from blank
3. Add data source: SharePoint CurrencyRates list

4. Design screens:
    - **Home Screen**: Display latest rates using Gallery control
    - **History Screen**: Show 7-day trend with charts
    - **Settings Screen**: Allow threshold updates
5. Publish and share the app

1.3 Configuration

Update Thresholds:

1. Go to your SharePoint **CurrencyRates** list
2. Click on any currency item
3. Edit the **Threshold** value
4. Save changes
5. Next flow run will use new threshold

Change Schedule:

1. Open the Power Automate flow
2. Click on the **Recurrence** trigger
3. Modify **Interval** and **Time zone**
4. Save the flow

Add More Currencies:

1. Add new row in SharePoint list with currency code and threshold
2. Duplicate HTTP/Parse/Variable/Get Items/Apply to each actions in flow
3. Update currency codes in API URLs and filter queries

1.4 Verification Checklist

- SharePoint list created with correct columns
- Initial currency data added to SharePoint
- API key obtained and tested
- Main flow created and saved successfully
- Flow runs without errors (check run history)
- Email alerts received when testing
- SharePoint data updates correctly
- Cleanup flow created and scheduled
- Dashboard created (if applicable)
- All connections authenticated properly

2. Deployment Guide

2.1 API Provider

Service: ExchangeRate-API
Website: https://www.exchangerate-api.com
Version: v6
Documentation: https://www.exchangerate-api.com/docs/overview

2.2 Authentication

Method: API Key in URL
Format: *https://v6.exchangerate-api.com/v6/{API_KEY}/{endpoint}*
Security: HTTPS only

2.3 Endpoints Used

Endpoint 1: Latest Rates

GET *https://v6.exchangerate-api.com/v6/{API_KEY}/latest/{BASE_CURRENCY}*

Description: Fetches latest exchange rates for specified base currency

Parameters:

- {API_KEY}: Your unique API key (string, required)
- {BASE_CURRENCY}: 3-letter currency code (e.g., USD, EUR, GBP)

Example Request:

GET *https://v6.exchangerate-api.com/v6/6bd51450faf7a83fc5111f06/latest/USD*

Response Format (JSON):


```
{
  "result": "success",
  "documentation": "https://www.exchangerate-api.com/docs",
  "terms_of_use": "https://www.exchangerate-api.com/terms",
  "time_last_update_unix": 1700049601,
  "time_last_update_utc": "Wed, 15 Nov 2025 00:00:01 +0000",
  "time_next_update_unix": 1700136001,
  "time_next_update_utc": "Thu, 16 Nov 2025 00:00:01 +0000",
  "base_code": "USD",
  "conversion_rates": {
    "USD": 1,
    "LKR": 330.75,
    "EUR": 0.92,
    "GBP": 0.79,
    "... (160+ currencies)": "..."
  }
}
```


Figure 2 - JSON Response of the External API

Key Fields Used:

- result: Status of API call ("success" or "error")
- base_code: Base currency code
- conversion_rates.LKR: Exchange rate to Sri Lankan Rupee
- time_last_update_utc: Timestamp of last rate update

Response Extraction:

LKR Rate = body('Parse_JSON')?['conversion_rates']?['LKR']

2.4 Rate Limits

Plan	Requests/Month	Cost
Free	1,500	$0
Basic	10,000	$9/month
Pro	50,000	$29/month

Current Usage:

- 3 API calls per day (USD, EUR, GBP) = 90 calls/month
- Well within free tier limits

2.5 Error Handling

Common Error Responses:

Figure 3 - Error Response

Error Types:

- unsupported-code: Invalid currency code
- malformed-request: Invalid API request format
- invalid-key: API key is incorrect or expired
- inactive-account: Account not active
- quota-reached: Monthly request limit exceeded

Implementation in Flow:

- Use **Configure run after** to handle HTTP failures
- Add **Condition** to check if result equals "success"
- Send error notification email if API call fails

2.6 API Best Practices

1. **Cache Results**: Store rates daily, don't call API multiple times per day
2. **Handle Errors Gracefully**: Always check response status
3. **Secure API Key**: Never expose in public repositories
4. **Monitor Quota**: Track API usage to avoid hitting limits
5. **Backup API**: Have alternative API configured for failover
6. **Rate Updates**: API updates rates once daily (typically midnight UTC)

3. Challenges Faced & Solutions

3.1 Challenge 1: Variable Initialization Inside Loops

Problem:

Attempted to initialize variables (USDThreshold, EURThreshold, GBPThreshold) inside "Apply to each" loops, resulting in error:

'The variable action 'Initialize_variable_-_USDThreshold' of type 'InitializeVariable' cannot be nested in an action of type 'For_each_-_USD'.'

Root Cause:
Power Automate does not allow Initialize Variable actions inside loop controls (Apply to each, For each, Do until). Variables must be initialized at the top/root level of the flow.

Solution Implemented:

1. Moved all Initialize Variable actions for thresholds to the top level (after the recurrence trigger)
2. Initialized variables with default value 0 (Float type)
3. Inside the loops, used Compose action instead of Initialize Variable
4. This allowed dynamic assignment of threshold values from SharePoint

Lesson Learned:
Always initialize all variables at the beginning of the flow, outside any control structures. Use Set Variable or Compose for dynamic value assignment.

3.2 Challenge 2: JSON Parsing Schema Definition

Problem:
Initially received error when trying to parse API response:

'The schema is invalid. Details: 'Unable to parse the content'

Root Cause:
The JSON schema provided didn't match the actual API response structure. ExchangeRate-API returns a complex nested object with conversion_rates containing all currency pairs.

Solution Implemented:

1. Made a test API call using browser/Postman to view actual response
2. Copied the complete JSON response

3. Used Power Automate's "Generate from sample" feature in Parse JSON action
4. Identified the exact path to LKR rate: conversion_rates.LKR
5. Updated all variable initializations to use correct JSON path

Working Schema:

```
{
  "type": "object",
  "properties": {
    "result": {"type": "string"},
    "base_code": {"type": "string"},
    "conversion_rates": {
      "type": "object",
      "properties": {
        "LKR": {"type": "number"}
      }
    }
  }
}
```

Figure 4 - Challenge 2 Solution Schema

Lesson Learned:
Always test API endpoints externally first and use the actual response to generate schema. Don't assume the structure without verification.

3.3 Challenge 3: SharePoint List Column Data Types

Problem:
Comparison condition failed with error:

'The template language expression 'variables('USDRate')' cannot be evaluated because the value is of type 'String' while expected type is 'Float'.'

Root Cause:
Initially created Threshold column as "Single line of text" instead of "Number", causing type mismatch in greater-than comparison.

Solution Implemented:

1. Deleted existing SharePoint list and recreated with correct column types
2. Ensured all numeric fields (Rate, Threshold, PreviousRate) were set to "Number" type with 2 decimal places

3. Changed variable types in Power Automate from String to Float for all rate and threshold variables
4. Updated variable initialization to use float conversion: float(item()?['Threshold'])

Correct Column Configuration:

Column	Type	Decimal Places
Rate	Number	2
Threshold	Number	2
PreviousRate	Number	2

Lesson Learned:
Pay careful attention to data types in both SharePoint and Power Automate. Type mismatches cause runtime errors that are hard to debug.

4. Conclusion

This Currency Tracker solution successfully demonstrates:

- Cloud-based automation using Power Platform

  External API integration with proper error handling
- Secure data storage with SharePoint Online
- Automated alerting mechanism via email
- Scheduled execution and data cleanup
- Scalable architecture for future enhancements

The challenges encountered during development provided valuable insights into Power Automate best practices, API integration patterns, and cloud solution design. The implemented solutions ensure a robust, maintainable, and production-ready system.

5. Appendix

A. Glossary

- **Power Automate**: Microsoft's cloud-based service for creating automated workflows
- **SharePoint Online**: Microsoft's cloud-based collaboration and document management platform
- **API**: Application Programming Interface for system integration
- **OData**: Open Data Protocol for querying and updating data
- **IST**: Indian Standard Time (UTC+5:30)
- **LKR**: Sri Lankan Rupee currency code

B. References

- Power Automate Documentation: https://docs.microsoft.com/en-us/power-automate/
- ExchangeRate-API Docs: https://www.exchangerate-api.com/docs/
- SharePoint REST API: https://docs.microsoft.com/en-us/sharepoint/dev/sp-add-ins/get-to-know-the-sharepoint-rest-service

C. Contact Information

Project Developer: Lithara Perera
Email: litharaperera2002@gmail.com
GitHub: https://github.com/Lithara
LinkedIn: https://www.linkedin.com/in/lithara-perera/