## RPA Solution Documentation: Yahoo Finance Data Management

### 1. Solution Overview

The primary goal of this automation is to address the challenges of manual data extraction and analysis from Yahoo Finance. It automates:

* Accessing Yahoo Finance to extract trending tickers and market data.
* Processing individual stock pages for specific details like currency and earnings dates.
* Storing, organizing, and analyzing the extracted data into a structured Excel report.
* Generating an HTML formatted email summary with the Excel report attached.

The solution leverages the RE-Framework for its robustness, including built-in error handling, transaction management, and logging capabilities.

### 2. Architecture & RE-Framework Modifications

The solution is based on the standard RE-Framework, with specific modifications to enhance flexibility, robustness, and deployment:

* **Dynamic Orchestrator Folder and Queue Management in** InitAllSettings.xaml:
  + The InitAllSettings.xaml workflow includes new In arguments:
    - in\_OrchestratorQueueFolder: Passed from Main.xaml. If the AssetsFolder column in Config.xlsx (Assets sheet) is empty, the automation uses this path to retrieve Orchestrator assets. This allows deploying as a .nupkg package without modifying Config.xlsx post-deployment, as the asset folder can be set directly as an Orchestrator process argument.
    - in\_OrchestratorQueueName: Passed from Main.xaml. This argument is used to programmatically access the specified queue, primarily to clear any existing items before the Dispatcher begins adding new items. This ensures a clean start for each run, preventing reprocessing of old items from a previous crashed or incomplete run.
* **Orchestrator Folder Configuration**:
  + The primary Orchestrator folder for both queues and assets is determined by the in\_OrchestratorQueueFolder input argument of Main.xaml.
  + If in\_OrchestratorQueueFolder is not provided, a default folder name **"YahooFinanceReporter"** is assumed.
* **Orchestrator Queue Name Configuration**: The queue name used by the automation is determined in the following order of precedence:
  + Value provided to the in\_OrchestratorQueueName input argument of Main.xaml.
  + Value set in the OrchestratorQueueName setting within the 'Settings' sheet of Config.xlsx.
  + If both the above are empty/not set, a default queue name **"YahooStocks"** is used.
* **Modularity**: The automation is designed with modular components for easier maintenance:
  + **Dispatcher**: Fetches initial data and populates the queue.
  + **Performer**: Processes each queue item (stock).
  + **Custom Logger (**Logger.xaml**)**: Manages detailed logging (see Section 8.2).
  + **Report (**Report.xaml**)**: Generates the Excel report.
  + **Object Repository**: UI elements are managed using a centralized Object Repository from a custom UI Library.

### 3. Configuration

#### 3.1. Orchestrator Assets

The following assets are configured in UiPath Orchestrator:

|  |  |  |
| --- | --- | --- |
| Asset Name | Type | Description |
| CurrencyExchangeEndpoint | Text | API endpoint (e.g., https://api.frankfurter.app/latest?from=USD) returning JSON exchange rates, used for market cap normalization to USD. |
| LogOutputPath | Text | Folder path for saving execution log files (used by the custom Logger.xaml). |
| ReportOutputPath | Text | Folder path for saving the generated Excel report. |
| Tables | Text | Comma-separated list of table names (e.g., "Trending Tickers") from Yahoo Finance. Names must match button text on the site. |
| Top N Market Caps | Number | Number of top market cap stocks for the "Highest Market Cap" sheet. |
| MailFrom | Text | Sender's email address for reports. (Note: Email sending infrastructure requires separate setup, see Section 7) |
| MailRecipients | Text | Semicolon-separated list of email addresses for report distribution. (Note: Email sending infrastructure requires separate setup, see Section 7) |

#### 3.2. Config.xlsx File

Key configurations in Config.xlsx:

* **Settings Sheet**:
  + logF\_BusinessProcessName: Standard RE-Framework entry, used in report naming.
  + OrchestratorQueueName: Fallback queue name if not set by in\_OrchestratorQueueName argument.
* **Assets Sheet**: Defines assets to be fetched from Orchestrator. The Folder column can be empty if in\_OrchestratorQueueFolder is used.
* **Constants Sheet**: Contains fixed values.

### 4. Data Extraction

* The automation accesses Yahoo Finance to extract financial data.
* It focuses on trending tickers and market data from tables specified in the Tables asset.
* For individual stocks identified, the robot navigates to their respective pages to retrieve specific details such as currency and earnings dates.

### 5. Data Storage & Organization (Excel Report)

The extracted and analyzed data is stored in a well-organized Excel workbook.

* **Excel File Naming Convention**: The report file is named dynamically: "[Config("logF\_BusinessProcessName")] Report - [Now.ToString("dd-MM-yyyy - HH-mm-ss")].xlsx". Example: YahooFinanceAutomation Report - 26-05-2025 - 11-55-00.xlsx.
* **Sheet Structure**: The Excel workbook contains:
  1. **All Data**: Raw stock data gathered during the run.
  2. **Top Movers**: Stocks with absolute percentage change > 2%.
  3. **USD**: Stocks in USD currency.
  4. **Non-USD**: Stocks in non-USD currencies.
  5. **Highest Market Cap**: Top N stocks by market capitalization (normalized to USD).
  6. **Earnings Today!**: Stocks with earnings date today or within a current range. (Sheet names are hardcoded due to tight coupling with their specific generation logic.)
* **Column Structure (Common across relevant sheets)**: Data is text. Columns include: Symbol, Symbol Url, Name, Change, Change %, Volume, Avg Vol (3M), Market Cap, P/E Ratio (TTM), 52 Wk Change %, Currency, Earnings Date. A temporary "Numeric Market Cap" column may be used internally during processing for calculations.

### 6. Data Analysis & Reporting (Sheet Generation Logic)

The Report.xaml workflow analyzes collected data and populates Excel sheets.

* **All Data**: Master list of all processed stocks.
* **Top Movers**: Stocks where absolute "Change %" > 2%.
* **USD Stocks**: Stocks where "Currency" is "USD".
* **Non-USD Stocks**: Stocks where "Currency" is not "USD".
* **Highest Market Cap**:
  + **Normalization**: Market capitalizations are normalized to a common currency (USD) for accurate comparison. An Invoke Code activity performs this:
    1. It fetches exchange rates from the CurrencyExchangeEndpoint (assuming rates are relative to USD, e.g., EUR per USD).
    2. A temporary numeric column for market cap is used or created if the original is text.
    3. For each stock not in USD, its numeric market cap is converted to USD equivalent by dividing by its currency's exchange rate against USD (e.g., if 1 USD = 0.92 EUR, an EUR market cap is divided by 0.92).
    4. Stocks whose currency exchange rate is not found are tracked and may be excluded from this specific sorting.
    5. The DataTable is then sorted based on these normalized USD market cap values in descending order.
  + The top N stocks are included. Business Rule Exceptions are thrown for items with missing/invalid market cap data during earlier processing stages.
* **Earnings Today!**:
  + This sheet is populated by filtering the "All Data" based on the "Earnings Date" column using a LINQ query. The logic is as follows:
    1. If the "Earnings Date" string contains " - ", it's treated as a date range (e.g., "May 26, 2025 - May 28, 2025"). The system checks if DateTime.Today falls within this parsed range (inclusive).
    2. If the "Earnings Date" string does not contain " - ", it's treated as a single date (e.g., "May 26, 2025"). The system checks if DateTime.Today is equal to this parsed date.
    3. Dates are parsed using "MMM d, yyyy" format (e.g., "May 26, 2025") with invariant culture.
  + If no stocks meet these criteria, the sheet will be empty or indicate no data.

### 7. Automated Report Distribution (Email)

* **Email Content**: HTML formatted email includes: greeting, process name, report date, top market cap stock details, "Earnings Today!" status, reference to attachment, and robot signature.
* **Attachment**: The full Excel report.
* **Implementation Status & Configuration**:
  + Email content generation is implemented. Assets MailFrom and MailRecipients define sender/recipients. An eml file is created.
  + The actual sending mechanism (SMTP, Outlook, etc.) **is not fully implemented** and requires customer-specific setup (server details, credentials, choice of UiPath activities) in the deployment environment.

### 8. Error Handling & Logging

#### 8.1. Error Handling

* **RE-Framework**: Standard exception handling and retries.
* **UI Interactions**: Object Repository activity verifications (relying on other screen elements rather on timeouts).
* **Custom Business Rule Exceptions**: Try-catch blocks for sensitive operations (e.g., market cap evaluation). A New BusinessRuleException("Relevant message") is thrown for issues like invalid/missing market cap, allowing RE-Framework to track these in Orchestrator.

#### 8.2. Logging

* A custom Logger.xaml is used:
  + **Inputs**: in\_message (string, required), in\_log\_filename (string, required), in\_output\_folder\_path\_str (string), in\_level (string, e.g., "Info", "Error"; defaults to "Info").
  + **Functionality**: Writes to UiPath logs (Log Message activity) and appends a timestamped, leveled message ("[timestamp] - [LEVEL] - [message]") to a custom text file in LogOutputPath.

### 9. Parameterization & Scalability

* **Parameterization**: Key parameters via Orchestrator assets and Config.xlsx.
* **Scalability**: Queues support complex transactions. UI interaction volume is internet-dependent; APIs would be better for massive scale. Solution is Yahoo Finance-specific.

### 10. Detailed User Guide

#### 10.1. Prerequisites

1. **UiPath Studio & Robot**:
   * Development and local testing can be performed using the free UiPath Community Edition.
   * For production runs, especially those scheduled or initiated from UiPath Orchestrator, appropriate UiPath licenses (e.g., Studio, Unattended Robot) are required.
2. **UiPath Orchestrator Connection**: Studio should be connected for full functionality (queues, assets, deployment, scheduling).
3. **Microsoft Edge Browser**: Installed, with the UiPath browser extension enabled and configured to "Allow in Incognito." The automation runs in incognito mode.

#### 10.2. Setup Instructions

1. **Orchestrator Folder**: Create (default: "YahooFinanceReporter"). If different, use in\_OrchestratorQueueFolder argument.
2. **Orchestrator Queue**: Create in the folder (default: "YahooStocks"). If different, set via in\_OrchestratorQueueName argument or Config.xlsx.
3. **Orchestrator Assets**: Create and configure as per Section 3.1.
4. **Project Deployment**: Publish from Studio to Orchestrator; create a Process.
5. **(If implementing email sending)**: Configure chosen email activities (e.g., SMTP server, Outlook) and secure necessary credentials (e.g., as Orchestrator assets).

#### 10.3. Running the Automation

* **From UiPath Orchestrator (Intended for production/scheduled runs)**: Start a job from the Process. Provide runtime arguments (in\_OrchestratorQueueFolder, in\_OrchestratorQueueName) if defaults/Config settings are not used or need overriding. (Note: Full Orchestrator-based execution path has not been tested due to license constraints during development).
* **From UiPath Studio (Development/Testing)**: Run Main.xaml. Set input arguments in Studio if needed.

#### 10.4. Maintenance

* **Configuration**: Update Orchestrator assets.
* **Selectors**: Update Object Repository if Yahoo Finance UI changes.
* **Log Monitoring**: Review Orchestrator and custom file logs.
* **Dependencies**: Keep UiPath components/packages updated.

### 11. Scheduling & Automation

* Use UiPath Orchestrator triggers for scheduled runs (requires appropriate licensing).