

## **Complete Step-by-Step Guide to Creating the MT5 Backtest Flow Script in Power Automate Desktop**

I'll guide you through creating the entire MT5 Backtest Flow script in Power Automate Desktop, explaining each step in detail as if teaching a 12-year-old. I won't skip any parts this time.

### **Step 1: Install and Open Power Automate Desktop**

1. First, we need to install Power Automate Desktop:
  - Open your web browser
  - Go to <https://flow.microsoft.com/en-us/desktop/>
  - Click "Download Power Automate Desktop"
  - Run the installer and follow the on-screen instructions
2. After installation, open Power Automate Desktop:
  - Click on the Windows Start menu
  - Type "Power Automate Desktop"
  - Click on the Power Automate Desktop app

### **Step 2: Create a New Flow**

1. When Power Automate Desktop opens, you'll see the main screen:
  - Click on the "New flow" button (it has a + symbol)
  - A dialog will appear asking for a flow name
  - Type "Backtest flow script V1.1.5" as the name
  - Click "Create" button
2. Now you'll see the flow designer screen with a blank canvas

### **Step 3: Initialize Default Configuration Variables**

1. First, we'll create our default configuration object:
  - Click the "+" button to add an action
  - In the search box, type "Set variable"
  - Select "Set variable" from the results

- In the "Variable name" field, type defaultConfig
- For "Variable type", select "Object"
- In the "Variable value" field, paste the entire JSON object from lines 5-36 of the script:

```

2. {
3.   "mt5Path": "C:\\Program Files\\MetaTrader 5 EXNESS\\terminal64.exe",
4.   "eaPath":
      "C:\\Users\\kigundu\\AppData\\Roaming\\MetaQuotes\\Terminal\\53785E099C927
      DB68A545C249CDBCE06\\MQL5\\Experts\\Custom EAs\\Moving Average",
5.   "reportPath": "D:\\FOREX\\FOREX DOCUMENTS\\MT5 STRATEGY TESTER
      REPORTS\\Reports",
6.   "startDate": "2019.01.01",
7.   "endDate": "2024.12.31",
8.   "reportCounter": 1,
9.   "maxWaitTimeForTest": 180,
10.  "initialLoadTime": 15,
11.  "maxRetries": 3,
12.  "skipOnError": true,
13.  "autoRestartOnFailure": true,
14.  "maxConsecutiveFailures": 5,
15.  "adaptiveWaitEnabled": true,
16.  "baseWaitMultiplier": 1.0,
17.  "maxAdaptiveWaitMultiplier": 5,
18.  "systemLoadCheckInterval": 300,
19.  "lowMemoryThreshold": 200,
20.  "verboseLogging": false,
21.  "logProgressInterval": 10,

```

```
22. "detailedSystemCheckInterval": 600,  
23. "logFilePath": "D:\\FOREX\\FOREX DOCUMENTS\\MT5 STRATEGY TESTER  
    REPORTS\\automation_log.json",  
24. "errorScreenshotsPath": "D:\\FOREX\\FOREX DOCUMENTS\\MT5 STRATEGY  
    TESTER REPORTS\\Reports\\errors",  
25. "checkpointFile": "D:\\FOREX\\FOREX DOCUMENTS\\MT5 STRATEGY TESTER  
    REPORTS\\Reports\\checkpoint.json",  
26. "configFilePath": "D:\\FOREX\\FOREX DOCUMENTS\\MT5 STRATEGY TESTER  
    REPORTS\\Reports\\backtest_config.json",  
27. "performanceHistoryFile": "D:\\FOREX\\FOREX DOCUMENTS\\MT5 STRATEGY  
    TESTER REPORTS\\Reports\\performance_history.json",  
28. "retryBackoffMultiplier": 1.5,  
29. "maxRetryWaitTime": 60  
30. }
```

Copy

- Click "Save" to add this action

#### **Step 4: Add Error Handling Structure**

1. Now we'll add a Try/Catch block to handle errors:
  - Click the "+" button to add an action
  - Search for "Try" and select it
  - This will create a Try block with Catch and Finally sections
2. Move your cursor to inside the Try block (between "Try" and "Catch")

#### **Step 5: Initialize Individual Variables**

Inside the Try block, we'll initialize all our individual variables:

1. For each configuration setting, add a "Set variable" action:
  - Click "+" inside the Try block
  - Search for "Set variable" and select it

- For the first variable:
  - Name: mt5Path
  - Type: Text
  - Value: %defaultConfig.mt5Path%
- Click "Save"

2. Repeat this process for each variable (one by one):

- eaPath = %defaultConfig.eaPath%
- reportPath = %defaultConfig.reportPath%
- startDate = %defaultConfig.startDate%
- endDate = %defaultConfig.endDate%
- reportCounter = %defaultConfig.reportCounter%
- maxWaitTimeForTest = %defaultConfig.maxWaitTimeForTest%
- initialLoadTime = %defaultConfig.initialLoadTime%
- maxRetries = %defaultConfig.maxRetries%
- skipOnError = %defaultConfig.skipOnError%
- autoRestartOnFailure = %defaultConfig.autoRestartOnFailure%
- maxConsecutiveFailures = %defaultConfig.maxConsecutiveFailures%
- consecutiveFailures = 0
- adaptiveWaitEnabled = %defaultConfig.adaptiveWaitEnabled%
- baseWaitMultiplier = %defaultConfig.baseWaitMultiplier%
- maxAdaptiveWaitMultiplier = %defaultConfig.maxAdaptiveWaitMultiplier%
- currentAdaptiveMultiplier = 1.0
- systemLoadCheckInterval = %defaultConfig.systemLoadCheckInterval%
- lastSystemLoadCheck = 0
- lowMemoryThreshold = %defaultConfig.lowMemoryThreshold%
- availableMemory = 1000

- verboseLogging = %defaultConfig.verboseLogging%
- logProgressInterval = %defaultConfig.logProgressInterval%
- detailedSystemCheckInterval = %defaultConfig.detailedSystemCheckInterval%
- logFilePath = %defaultConfig.logFilePath%
- errorScreenshotsPath = %defaultConfig.errorScreenshotsPath%
- checkpointFile = %defaultConfig.checkpointFile%
- eaIndex = 0
- currencyIndex = 0
- timeframeIndex = 0
- resumeFromCheckpoint = false
- configFilePath = %defaultConfig.configFilePath%
- performanceHistoryFile = %defaultConfig.performanceHistoryFile%
- retryBackoffMultiplier = %defaultConfig.retryBackoffMultiplier%
- maxRetryWaitTime = %defaultConfig.maxRetryWaitTime%

### **Step 6: Check if Paths Exist**

Still inside the Try block, after setting all variables:

#### **1. Add an If condition to check if MT5 path exists:**

- Click "+" to add an action
- Search for "If" and select it
- In the condition field, type: FOLDER NOT EXISTS "%mt5Path%"
- Inside this If block:
  - Add a "Throw exception" action
  - Set the message to: "MetaTrader 5 path does not exist: %mt5Path%"
- Click "Save"

#### **2. Add another If condition to check if EA path exists:**

- Click "+" after the End If
  - Search for "If" and select it
  - In the condition field, type: FOLDER NOT EXISTS "%eaPath%"
  - Inside this If block:
    - Add a "Throw exception" action
    - Set the message to: "EA path does not exist: %eaPath%"
  - Click "Save"
3. Add an If condition to create reports folder if it doesn't exist:
- Click "+" after the End If
  - Search for "If" and select it
  - In the condition field, type: FOLDER NOT EXISTS "%reportPath%"
  - Inside this If block:
    - Add a "Create folder" action
    - Set the path to: "%reportPath%"
  - Click "Save"
4. Add an If condition to create error screenshots folder if it doesn't exist:
- Click "+" after the End If
  - Search for "If" and select it
  - In the condition field, type: FOLDER NOT EXISTS "%errorScreenshotsPath%"
  - Inside this If block:
    - Add a "Create folder" action
    - Set the path to: "%errorScreenshotsPath%"
  - Click "Save"

## **Step 7: Initialize Performance History**

Still inside the Try block:

1. Create a dictionary for performance history:

- Click "+" to add an action
  - Search for "Create dictionary" and select it
  - Set the variable name to: performanceHistory
  - Click "Save"
2. Add an If condition to check if performance history file exists:
- Click "+" to add an action
  - Search for "If" and select it
  - In the condition field, type: FILE EXISTS "%performanceHistoryFile%"
  - Inside this If block:
    - Add a "Try" action to create a nested try/catch
    - Inside this nested Try:
      - Add a "Read text from file" action
      - Set the file path to: "%performanceHistoryFile%"
      - Set the variable to store the result in: historyData
      - Add a "Parse JSON" action
      - Set the JSON text to: "%historyData%"
      - Set the variable to store the result in: performanceHistory
    - Inside the nested Catch:
      - Add an "Append text to file" action
      - Set the file path to: "%logFilePath%"
      - Set the text to append: "Error reading performance history file. Initializing new history.\r\n"
  - Click "Save"

### **Step 8: Log Start of Execution**

Still inside the main Try block:

1. Create a log entry object:

- Click "+" to add an action
- Search for "Set variable" and select it
- Set the variable name to: logEntry
- Set the variable type to: Object
- Set the value to:

```

2. {
3.   "timestamp": "%CURRENT DATE% %CURRENT TIME%",
4.   "level": "INFO",
5.   "message": "Started execution",
6.   "details": {
7.     "mt5Path": "%mt5Path%",
8.     "eaPath": "%eaPath%",
9.     "reportPath": "%reportPath%"
10.  }
11.}

```

Copy

- Click "Save"

12. Append the log entry to the log file:

- Click "+" to add an action
- Search for "Append text to file" and select it
- Set the file path to: "%logFilePath%"
- Set the text to append: "%logEntry TO JSON%\r\n"
- Click "Save"

### Step 9: Set Up Catch Block

Now move to the Catch section (after the Try block):

1. Create an error log entry:



- Click "+" inside the Catch block
  - Search for "Set variable" and select it
  - Set the variable name to: errorLog
  - Set the variable type to: Object
  - Set the value to:
2. {
  3. "timestamp": "%CURRENT DATE% %CURRENT TIME%",
  4. "level": "ERROR",
  5. "message": "Error during initialization",
  6. "error": "%ERROR MESSAGE%"
  7. }

Copy

- Click "Save"
8. Append the error log to the log file:
    - Click "+" to add an action
    - Search for "Append text to file" and select it
    - Set the file path to: "%logFilePath%"
    - Set the text to append: "%errorLog TO JSON%\r\n"
    - Click "Save"
  9. Display an error message:
    - Click "+" to add an action
    - Search for "Display message" and select it
    - Set the message to: "Error during initialization: %ERROR MESSAGE%"
    - Click "Save"
  10. Add an If condition to check skipOnError:
    - Click "+" to add an action

- Search for "If" and select it
- In the condition field, type: %skipOnError%
- Inside this If block:
  - Create a continue log entry:
    - Add a "Set variable" action
    - Set the variable name to: continueLog
    - Set the variable type to: Object
    - Set the value to:
      - {
      - "timestamp": "%CURRENT DATE% %CURRENT TIME%",
      - "level": "WARN",
      - "message": "Continuing despite initialization error"
      - }

Copy

- Append the continue log to the log file:
  - Add an "Append text to file" action
  - Set the file path to: "%logFilePath%"
  - Set the text to append: "%continueLog TO JSON%\r\n"
- In the Else block:
  - Add an "Exit flow" action
- Click "Save"

## Step 10: Create Helper Functions

Now we'll create all the helper functions. For each function:

### 10.1. LogMessage Function

1. Click "+" after the Try/Catch block
2. Search for "Function" and select it

3. Set the function name to: LogMessage

4. Add parameters:

- Click "Add parameter" and add:
  - level (Text)
  - message (Text)
  - details (Text)

5. Inside the function:

- Add a "Set variable" action:
  - Variable name: logEntry
  - Variable type: Object
  - Value:
- {
- "timestamp": "%CURRENT DATE% %CURRENT TIME%",
- "level": "%level%",
- "message": "%message%"
- }

Copy

- Add an "If" condition:
  - Condition: "%details%" != ""
  - Inside If:
    - Add a "Set variable" action:
      - Variable name: logEntry.details
      - Value: %details%
- Add another "If" condition:
  - Condition: %verboseLogging% OR "%level%" = "ERROR" OR "%level%" = "WARN" OR "%level%" = "INFO"

- Inside If:
  - Add an "Append text to file" action:
    - File path: "%logFilePath%"
    - Text to append: "%logEntry TO JSON%\r\n"

6. Click "Save"

## 10.2. VerifyAndSetValue Function

1. Click "+" to add a new function
2. Set the function name to: VerifyAndSetValue
3. Add parameters:
  - fieldName (Text)
  - currentValue (Text)
  - targetValue (Text)
4. Inside the function:
  - Add an "If" condition:
    - Condition: "%currentValue%" != "%targetValue%"
    - Inside If:
      - Call the LogMessage function:
        - level: "DEBUG"
        - message: "Changing %fieldName% from '%currentValue%' to '%targetValue%'"
        - details: ""
      - Add a "Return" action:
        - Return value: false
    - Inside Else:
      - Call the LogMessage function:
        - level: ` "DEBUG

Please provide the remaining information since I have saved this part

PromptsClaude 3.7 Sonnet

**Context— none fetched**

## **Continuing the Complete Step-by-Step Guide to Creating the MT5 Backtest Flow Script**

Let's continue from where we left off with creating the helper functions:

### **10.2. VerifyAndSetValue Function (continued)**

Inside the Else block:

- Call the LogMessage function:
  - level: "DEBUG"
  - message: "%fieldName% already set to '%targetValue%', skipping"
  - details: ""
- Add a "Return" action:
  - Return value: true
- Click "Save"

### **10.3. LoadConfiguration Function**

1. Click "+" to add a new function
2. Set the function name to: LoadConfiguration
3. Inside the function:
  - Add an "If" condition:
    - Condition: FILE EXISTS "%configFilePath%"
    - Inside If:
      - Add a "Try" action (nested try/catch)
      - Inside the nested Try:
        - Call LogMessage:
          - level: "INFO"

- message: "Loading configuration from %configFilePath%"
- details: ""
- Add a "Read text from file" action:
  - File path: "%configFilePath%"
  - Variable to store result: configData
- Add another nested "Try" for JSON parsing:
  - Add a "Parse JSON" action:
    - JSON text: "%configData%"
    - Variable to store result: config
  - Add multiple "If" conditions to check and set each configuration property:
    - For each property in the config object, add an "If" condition:
      - Condition: PROPERTY "propertyName" EXISTS IN config
      - Inside If: Set the corresponding variable
    - Repeat for all properties (mt5Path, eaPath, reportPath, etc.)
  - Call LogMessage:
    - level: "INFO"
    - message: "Configuration loaded successfully from JSON"
    - details: ""
- Add a "Catch" for the JSON parsing:
  - Call LogMessage:
    - level: "WARN"

- message: "Failed to parse JSON config, falling back to text format"
  - details: ""
- Add text parsing logic for legacy format:
  - Split the config data by line breaks
  - Loop through each line
  - Parse key-value pairs
  - Set variables based on keys
- Add a "Catch" for the overall file reading:
  - Call LogMessage:
    - level: "ERROR"
    - message: "Error loading configuration: %ERROR MESSAGE%. Using default settings."
    - details: ""
- Inside Else (if config file doesn't exist):
  - Call LogMessage:
    - level: "INFO"
    - message: "No configuration file found at %configFilePath%. Using default settings."
    - details: ""

4. Click "Save"

#### 10.4. AdaptiveWait Function

1. Click "+" to add a new function
2. Set the function name to: AdaptiveWait
3. Add parameters:
  - waitTime (Number)
  - isRetry (Boolean)

- retryCount (Number)

#### 4. Inside the function:

- Add an "If" condition:
  - Condition: %isRetry%
  - Inside If (for retries):
    - Add "Set variable" actions to calculate backoff:
      - $\text{backoffFactor} = \text{MIN}(\text{\%maxRetryWaitTime\%} / \text{\%waitTime\%}, \text{POWER}(\text{\%retryBackoffMultiplier\%}, \text{\%retryCount\%}))$
      - $\text{adjustedWaitTime} = \text{\%waitTime\%} * \text{\%backoffFactor\%}$
      - $\text{adjustedWaitTime} = \text{MIN}(\text{\%adjustedWaitTime\%}, \text{\%maxRetryWaitTime\%})$
  - Inside Else If:
    - Condition: %adaptiveWaitEnabled%
    - Inside this Else If:
      - $\text{adjustedWaitTime} = \text{\%waitTime\%} * \text{\%currentAdaptiveMultiplier\%}$
  - Inside final Else:
    - $\text{adjustedWaitTime} = \text{\%waitTime\%}$
- Add another "If" condition for logging:
  - Condition: %adjustedWaitTime% > %waitTime% \* 1.5 AND %verboseLogging%
  - Inside If:
    - Call LogMessage:
      - level: "DEBUG"
      - message: "Adjusted wait time from %waitTime% to %adjustedWaitTime% seconds"
      - details: ""



- Add a "Wait" action:
  - Duration: %adjustedWaitTime%
- 5. Click "Save"

### **10.5. LegacyAdaptiveWait Function**

1. Click "+" to add a new function
2. Set the function name to: LegacyAdaptiveWait
3. Add parameters:
  - waitTime (Number)
4. Inside the function:
  - Call AdaptiveWait:
    - waitTime: %waitTime%
    - isRetry: false
    - retryCount: 0
5. Click "Save"

### **10.6. CaptureErrorState Function**

1. Click "+" to add a new function
2. Set the function name to: CaptureErrorState
3. Add parameters:
  - errorContext (Text)
4. Inside the function:
  - Add a "Try" action
  - Inside Try:
    - Format timestamp for filename
    - Take screenshot
    - Log screenshot details
    - Check if Strategy Tester window exists and save partial results

- Inside Catch:
  - Log error

5. Click "Save"

### **10.7. SaveCheckpoint Function**

1. Click "+" to add a new function
2. Set the function name to: SaveCheckpoint
3. Inside the function:
  - Add a "Try" action
  - Inside Try:
    - Create checkpoint data object
    - Write to checkpoint file
    - Log success
  - Inside Catch:
    - Log error
4. Click "Save"

### **10.8. PerformMemoryCleanup Function**

1. Click "+" to add a new function
2. Set the function name to: PerformMemoryCleanup
3. Inside the function:
  - Add an "If" condition to check memory threshold
  - Inside If:
    - Log warning
    - Kill explorer.exe process
    - Restart explorer.exe
    - Check if cleanup helped
    - Log results

4. Click "Save"

### **10.9. DetailedSystemCheck Function**

1. Click "+" to add a new function
2. Set the function name to: DetailedSystemCheck
3. Inside the function:
  - Get current time
  - Check if it's time to run a system check
  - Get available memory
  - Get CPU usage if memory is low
  - Create system metrics object
  - Log system status if needed
  - Adjust wait multiplier based on system metrics
  - Call memory cleanup if needed
4. Click "Save"

### **10.10. EstimateTestDuration Function**

1. Click "+" to add a new function
2. Set the function name to: EstimateTestDuration
3. Add parameters:
  - currency (Text)
  - timeframe (Text)
  - eaName (Text)
4. Inside the function:
  - Check if historical data exists for this combination
  - Calculate estimated duration based on parameters
  - Return the estimated duration
5. Click "Save"

### **10.11. UpdatePerformanceHistory Function**

1. Click "+" to add a new function
2. Set the function name to: UpdatePerformanceHistory
3. Add parameters:
  - currency (Text)
  - timeframe (Text)
  - eaName (Text)
  - actualDuration (Number)
4. Inside the function:
  - Create key for this combination
  - Update or add the entry in performance history
  - Save to file
  - Log update
5. Click "Save"

### **10.12. UI Interaction Helper Functions**

Create these UI interaction helper functions:

1. NavigateToField - Tabs to a specific field position
2. EnterValue - Enters a value in the current field
3. SelectDropDownItem - Selects an item from a dropdown
4. ConfigureField - Configures a field with retry logic
5. ConfigureDropDownField - Configures a dropdown field with retry logic
6. OptimizeMT5ForBacktesting - Optimizes MT5 settings
7. SaveCurrentSettings - Saves Strategy Tester settings
8. LoadSettings - Loads saved Strategy Tester settings

### **Step 11: Define Lists for Timeframes and Currencies**

1. Create the timeframes list:

- Click "+" after all functions
  - Search for "Create list" and select it
  - Set the variable name to: timeframes
  - Click "Save"
2. Add all timeframes to the list:
- For each timeframe (M1, M2, M3, etc.), add an "Add item to list" action:
    - List variable: timeframes
    - Item to add: "M1" (then M2, M3, etc.)
  - Repeat for all 21 timeframes
3. Create the currencies list:
- Add a "Create list" action
  - Set the variable name to: currencies
  - Click "Save"
4. Add all currency pairs to the list:
- For each currency pair, add an "Add item to list" action:
    - List variable: currencies
    - Item to add: "EURUSD" (then GBPUSD, etc.)
  - Repeat for all currency pairs

## **Step 12: Get List of EAs**

1. Create the EA list:
- Add a "Create list" action
  - Set the variable name to: eaList
  - Click "Save"
2. Add a "Try" action to get EA files:
- Inside Try:
    - Add a "Get files in folder" action:

- Folder path: "%eaPath%"
- File pattern: "\*.ex5"
- Recursive: No
- Variable to store result: eaList
- Add an "If" condition to check if any EAs were found:
  - Condition: COUNT OF LIST eaList = 0
  - Inside If:
    - Add an "Add item to list" action:
      - List variable: eaList
      - Item to add: "Moving Average"
    - Call LogMessage to warn about no EAs found
  - Inside Else:
    - Log found EAs
- Inside Catch:
  - Log error
  - Add default EA to prevent errors

### **Step 13: Load Configuration and Check for Checkpoint**

1. Call the LoadConfiguration function
2. Add an "If" condition to check for checkpoint file:
  - Condition: FILE EXISTS "%checkpointFile%"
  - Inside If:
    - Add a "Try" action
    - Inside Try:
      - Read checkpoint file
      - Parse checkpoint data
      - Set variables from checkpoint

- Set resumeFromCheckpoint to true
- Log resuming from checkpoint
- Inside Catch:
  - Log error
  - Set resumeFromCheckpoint to false

#### **Step 14: Initial System Check and Launch MT5**

1. Call the DetailedSystemCheck function
2. Add a "Set variable" action:
  - Variable name: retryCount
  - Value: 0
3. Add a "While" loop for MT5 launch retries:
  - Condition: %retryCount% < %maxRetries%
  - Inside While:
    - Add a "Try" action
    - Inside Try:
      - Log MT5 launch attempt
      - Run MT5 program
      - Wait for MT5 to load
      - Check if MT5 window exists
      - Log success
      - Call OptimizeMT5ForBacktesting
      - Configure MT5 to use unlimited bars
      - Break out of loop
    - Inside Catch:
      - Increment retry counter
      - Log error

- Capture error state
- Check if max retries reached
- Wait before retrying

### **Step 15: Create Templates Directory**

Add an "If" condition to check if templates directory exists:

- Condition: FOLDER NOT EXISTS "%reportPath%\templates"
- Inside If:
  - Create the templates folder

### **Step 16: Main Nested Loops for Testing**

1. Add a "For each" loop for EAs:
  - List variable: eaList
  - Loop variable: eaFile
  - Index variable: eaIndex
  - Inside loop:
    - Check if should skip to checkpoint position
    - Extract EA name
    - Log starting tests for this EA
2. Inside the EA loop, add a "For each" loop for currencies:
  - List variable: currencies
  - Loop variable: currency
  - Index variable: currencyIndex
  - Inside loop:
    - Check if should skip to checkpoint position
    - Add a "Try" action
    - Inside Try:
      - Log starting tests for this currency



- Check system resources
- Open Strategy Tester with retry logic
- Configure Strategy Tester
- Save settings

3. Inside the currency loop, add a "For each" loop for timeframes:

- List variable: timeframes
- Loop variable: timeframe
- Index variable: timeframeIndex
- Inside loop:
  - Check if should skip to checkpoint position
  - Add a "Try" action
  - Inside Try:
    - Log setting timeframe
    - Check system resources
    - Load saved settings
    - Configure timeframe
    - Save settings
    - Estimate test duration
    - Save checkpoint
    - Start test
    - Wait for test to complete with progress monitoring
    - Record actual test duration
    - Save report
    - Check if need to restart MT5
  - Inside Catch:
    - Log error

- Capture error state
- Increment consecutive failures
- Check if need to restart MT5

### **Step 17: Cleanup After All Tests**

1. Add a "Try" action after all loops
2. Inside Try:
  - Close MT5
  - Handle "Save changes" dialog
  - Remove checkpoint file
  - Generate summary report
  - Log completion
  - Display completion message
3. Inside Catch:
  - Log error
  - Display completion message with warning

### **Step 18: Save and Test the Flow**

1. Click the "Save" button at the top of the screen to save your flow
2. Click the "Run" button to test your flow:
  - Start with a small subset of data (few EAs, currencies, timeframes)
  - Monitor the execution
  - Check logs for errors
  - Adjust wait times and error handling as needed

### **Step 19: Make Adjustments and Finalize**

1. Review the flow execution results
2. Make any necessary adjustments to:
  - Wait times

- Error handling
  - UI interaction logic
3. Save the final version of your flow

### **Important Tips for Success**

1. **Test in Stages:** Don't try to run the entire flow at once. Test each function and section separately.
2. **Customize Paths:** Update the paths in the defaultConfig to match your system.
3. **UI Interaction:** The most challenging part is UI interaction. Make sure your MT5 window is visible and accessible.
4. **Error Handling:** This script has robust error handling. Don't skip implementing the try/catch blocks.
5. **System Resources:** For long runs, the system resource monitoring is crucial. Implement it carefully.
6. **Checkpoints:** The checkpoint system allows resuming long runs. Test it by intentionally stopping the flow.
7. **Logging:** Implement all logging to help troubleshoot issues.
8. **Adaptive Waits:** The adaptive wait system helps handle varying system loads. Implement it as shown.

This completes the step-by-step guide for implementing the entire MT5 Backtest Flow script in Power Automate Desktop. Take your time with each step, and don't hesitate to test sections as you build them.