NIFI

SIC PROJECT

Summary

We generate sample JSON order files in ~/sample_orders/input, then use a NiFi flow — ListFile → FetchFile → UpdateAttribute → PutFile — to move those files to ~/sample_orders/output while adding a timestamp attribute.

Prepare environment (on each teammate's VM)

Create the folders used by the flow:

Example:

mkdir -p ~/sample_orders/input

mkdir -p ~/sample_orders/output

Ensure write permissions so NiFi can read/write files:

Example:

chmod -R 755 ~/sample_orders

chmod -R 777 ~/sample_orders

Generate sample JSON orders (copy & run)

Use this Python one-liner/script to create 100 JSON files with nested items arrays (each file = one order). Run directly and output should be in your input directory to the NiFi ListFile input directory:

```
mkdir -p ~/sample_orders/input
python3 - <<'EOF'
import json, os, random, datetime

input_dir = os.path.expanduser('~/sample_orders/input')

products = [101, 102, 103, 104, 105]

for i in range(1, 101):  # 100 files
    order = {
        "order_id": i,
        "customer_id": random.randint(1000, 2000),
        "order_date": datetime.date.today().isoformat(),
        "total_value": random.randint(20, 500),
        "payment_status": random.choice(["PAID", "PENDING", "FAILED"]),
        "items": [
            {
                "product_id": random.choice(products),
                "sales_quantity": random.randint(1, 5)
            }
            for _ in range(random.randint(1, 3))  # 1-3 items per order
        ]
    }
    file_path = os.path.join(input_dir, f"order_{i}.json")
    with open(file_path, "w") as f:
        json.dump(order, f)
EOF
```

(Script will be provided separately)

Verify files exist:

ls -l ~/sample_orders/input | head

wc -l ~/sample_orders/input/*.json

NiFi flow

Add processors (drag & add)

On the NiFi canvas, add these processors (search and add each):

- ListFile (org.apache.nifi.processors.standard.ListFile)
- FetchFile (org.apache.nifi.processors.standard.FetchFile)
- UpdateAttribute (org.apache.nifi.processors.attributes.UpdateAttribute)
- PutFile (org.apache.nifi.processors.standard.PutFile)

Place them in a row or left→right order.

Configure properties (double-click each processor → Properties tab)

ListFile

- Input Directory (exact):

/home/<username>/sample_orders/input (The output directory you created)

Leave scheduling and concurrency at defaults unless needed.

FetchFile

- No directory property needed (it uses the filename/path from ListFile). Leave defaults.

UpdateAttribute (configuration is optional)

- Add property ingestion_time with value:

${now():format("yyyy-MM-dd'T'HH:mm:ss'Z'")}

PutFile

- Directory (exact):

/home/<username>/sample_orders/output (The output directory you created)

Conflict Resolution Strategy: replace (or ignore depending on your preference)

Wire connections (create relationships)

For each processor, drag an arrow from the source to the destination and choose the appropriate relationship(s):

1. ListFile → FetchFile
    - Choose relationship: success
2. FetchFile → UpdateAttribute
    - Choose relationship: success
3. UpdateAttribute → PutFile
    - Choose relationship: success

Important: for relationships you do **not** handle (e.g., failure, not.found, etcAuto-terminate those relationships so NiFi stops complaining:

    - Right-click the processor → Configure → **Settings** tab → check the relationships to **Automatically Terminate Relationships** (e.g., failure, not.found) → Apply

For **PutFile**, because it is the final sink, you can auto-terminate its success (or just leave it unconnected and also auto-terminate success in settings). If you leave success un-terminated and un-connected, NiFi will mark the processor invalid.

Validate and fix any yellow warnings

- Hover over each processor. If you see a **yellow triangle**, open the processor and check required properties. Fix the missing property (usually directory path) or auto-terminate unused relationships.

Start the flow

Select all, right click and select *Start*. If everything is correct you should see 4 on the start symbol on top. With no yellow marks on any of the processors

Verify ingestion

- Count files in output:

ls -1 ~/sample_orders/output | wc -l

(make sure it's the name of your output directory configured in the PutFile Processor)

Expected: 100 (if you generated 100 orders)

7. What we added/what each processor does (clear explanations)

- **ListFile**
    - Purpose: lists files in a directory and generates FlowFile metadata (filename, path). It does NOT read file contents.
    - Key props: Input Directory, File Filter.
    - Output relationship used: success.
- **FetchFile**
    - Purpose: reads the file content referenced by ListFile and attaches content to the FlowFile.
    - Key note: uses filename/path attributes passed by ListFile.
    - Relationships: success, not.found, failure. not.found or failure can be auto-terminated if not handled.
- **UpdateAttribute**
    - Purpose: set FlowFile attributes (metadata) without changing content. We use it to inject ingestion_time.
    - Example attribute: ingestion_time = ${now()}.
- **PutFile**
    - Purpose: write FlowFile content to a local file system directory.

- Key props: Directory, File Name, Conflict Resolution.
- After PutFile, the FlowFile job is typically finished.

---

8. Auto-terminate vs connect — rules of thumb

- Auto-terminate a relationship only if you **do not** plan to process that outcome.
  - Example: FetchFile:not.found → if you do not have a recovery step, auto-terminate.
- Do **not** auto-terminate success on a processor you intend to chain — connect success to the next processor.
- A processor with required relationships unhandled (not connected and not auto-terminated) will be considered invalid — NiFi prevents starting.

Common problems & fixes:

- **Start button greyed** → at least one processor is invalid (yellow triangle). Hover the triangle and fix missing required properties or auto-terminate unused relationships.
- **ListFile: Input Directory invalid** → ensure the exact path exists on the node NiFi runs on (absolute path), correct ownership/permissions.
- **PutFile writing fails** → check write permissions and that the Directory exists. Use chmod to grant access or change owner.
- **Curl upload returns 401 Unauthorized** → NiFi is secured. Use the UI or provide valid user credentials with curl -u user:pass -k.
- **Connection refused on 8080** → NiFi may run on HTTPS/8443. Use the correct port.
- **Files are not appearing** → check NiFi processor bulletins (red circle) and NiFi logs in nifi-app.log.
- **Files being re-read repeatedly** → ListFile has state; if you used GetFile instead of ListFile + FetchFile, GET removes files. Use ListFile+FetchFile so ListFile tracks what's been listed and FetchFile reads content.