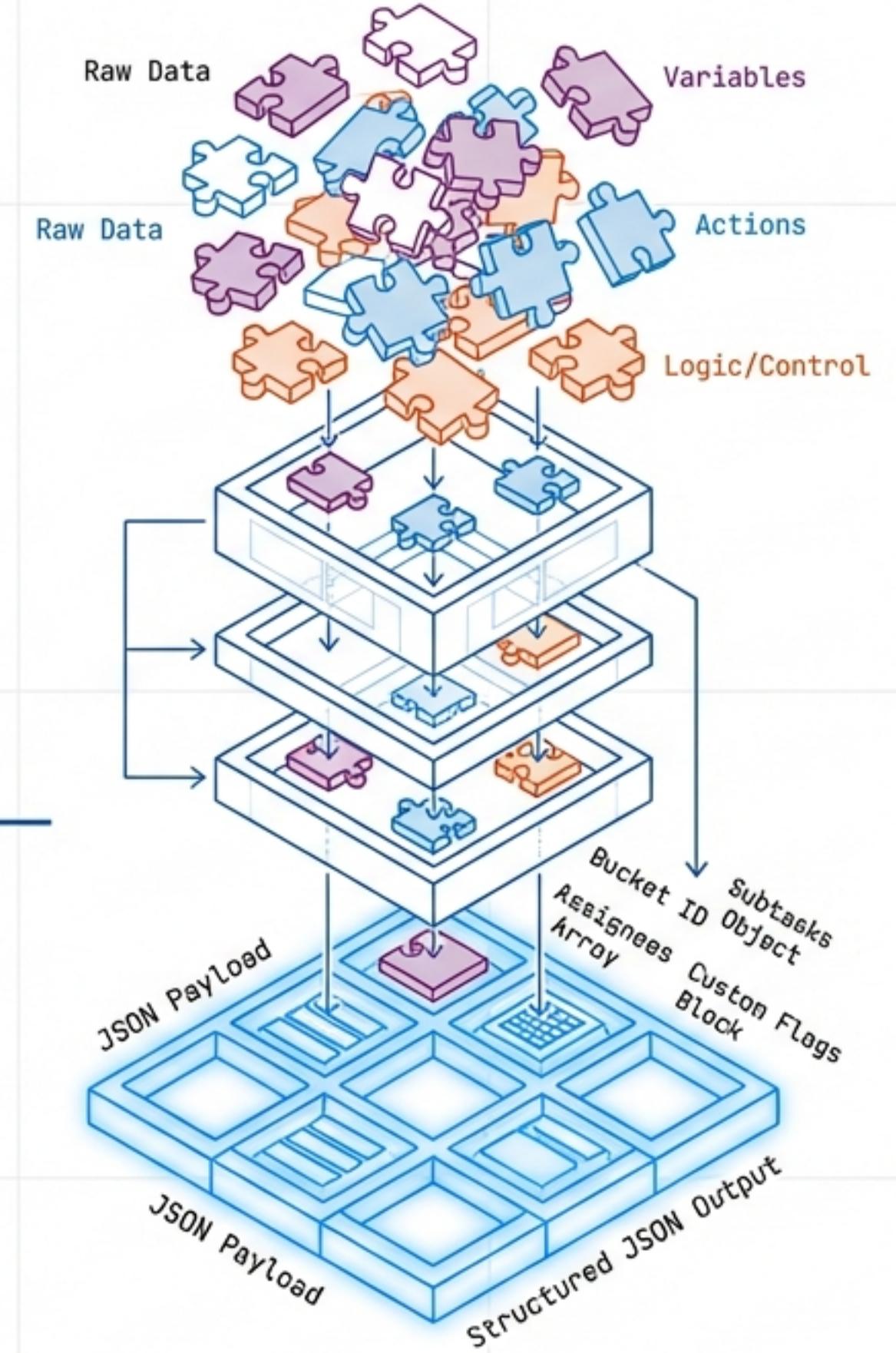


Mastering Planner Data Extraction

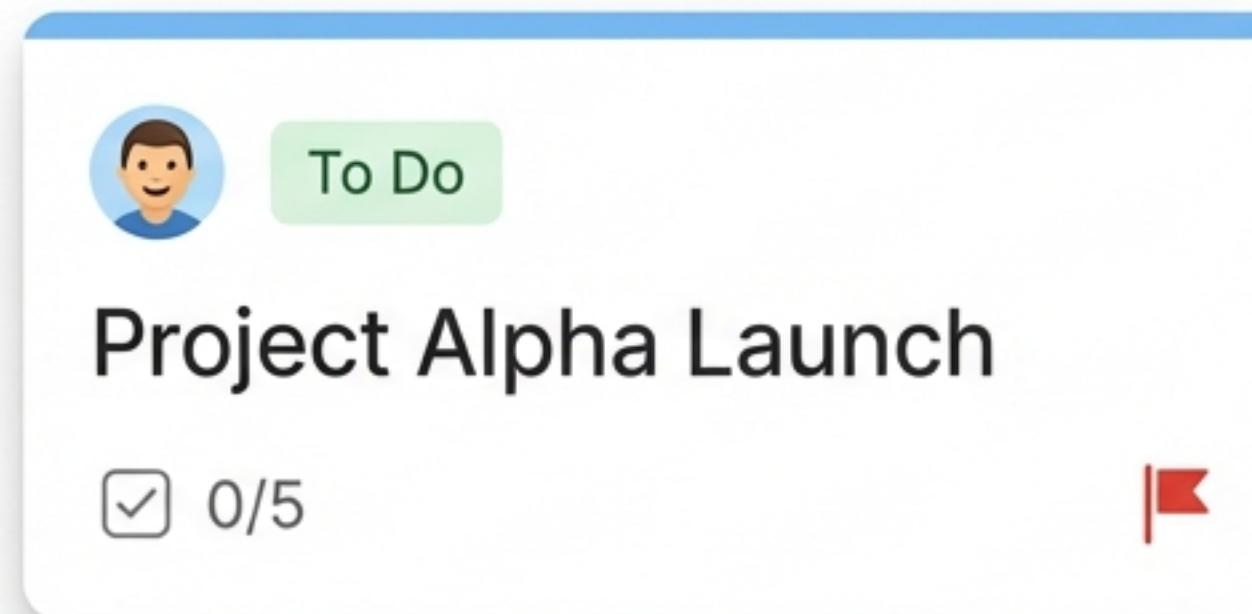
A Power Automate Deep Dive

A step-by-step architectural guide to extracting Assignees, Subtasks, Buckets, and Custom Flags into a unified JSON dataset.



The Data Gap: What ‘List Tasks’ Actually Gives You

What You See in Planner



What You Get (Raw JSON)

```
1 {  
2   "title": "Project Alpha Launch",  
3   "percentComplete": 50, <- "Status Code, not Math"  
4   "bucketId": "28374-ad73", <- "Where is the Name?"  
5   "checklistItemCount": 5, <- "Where are the items?"  
6   "assignment":  
7     "8473-3d22" } <- "Where is the User Name?"  
8 }  
9 }
```

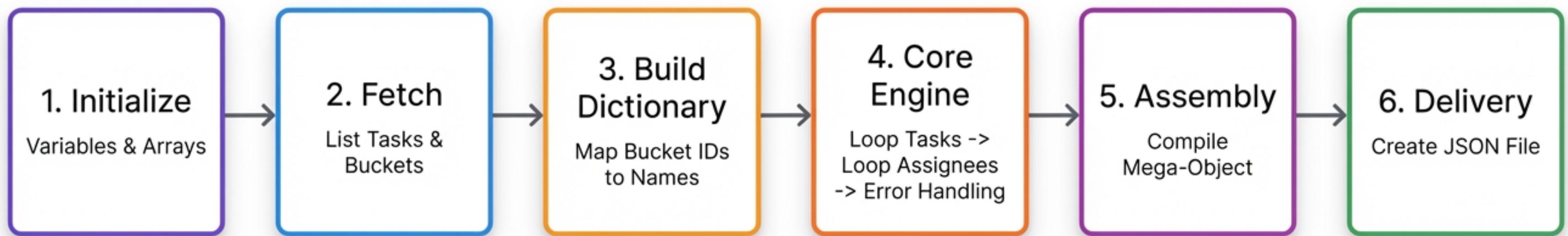
MISSING NAME

MISSING BUCKET

MISSING DETAILS

A code editor window displaying raw JSON data. The JSON object contains several fields: 'title', 'percentComplete', 'bucketId', 'checklistItemCount', 'assignment', and a closing brace. Three large red stamps are overlaid on the JSON: 'MISSING NAME' covers the 'percentComplete' field, 'MISSING BUCKET' covers the 'bucketId' field, and 'MISSING DETAILS' covers the 'assignment' field. The JSON is numbered from 1 to 9 on the left.

The Logic Architecture



Step 1: The Trigger & Schedule

Action: Scheduled Cloud Flow

Frequency: Daily

Context: Acts as a live data source

Context: Acts as a live data source
for Power BI refresh cycles.

Recurrence

Interval

1

Frequency

Day

Time zone

(UTC-08:00) Pacific Time

Step 2: Initializing Storage Containers

We must initialize three specific **Array** variables. These containers will hold our fragmented data as we process it.

1. **PlannerData**: The final consolidated output.
2. **Assignees**: Temporary buffer for user profiles.
3. **Buckets**: The ID-to-Name directory.

Initialize Variable

Name: PlannerData

Type: **Array**

Initialize Variable

Name: Assignees

Type: **Array**

Initialize Variable

Name: Buckets

Type: **Array**

Step 3: Fetching the Raw Lists

List Tasks

Group Id:
Marketing Team

Plan Id:
Q4 Campaign Launch

List Buckets

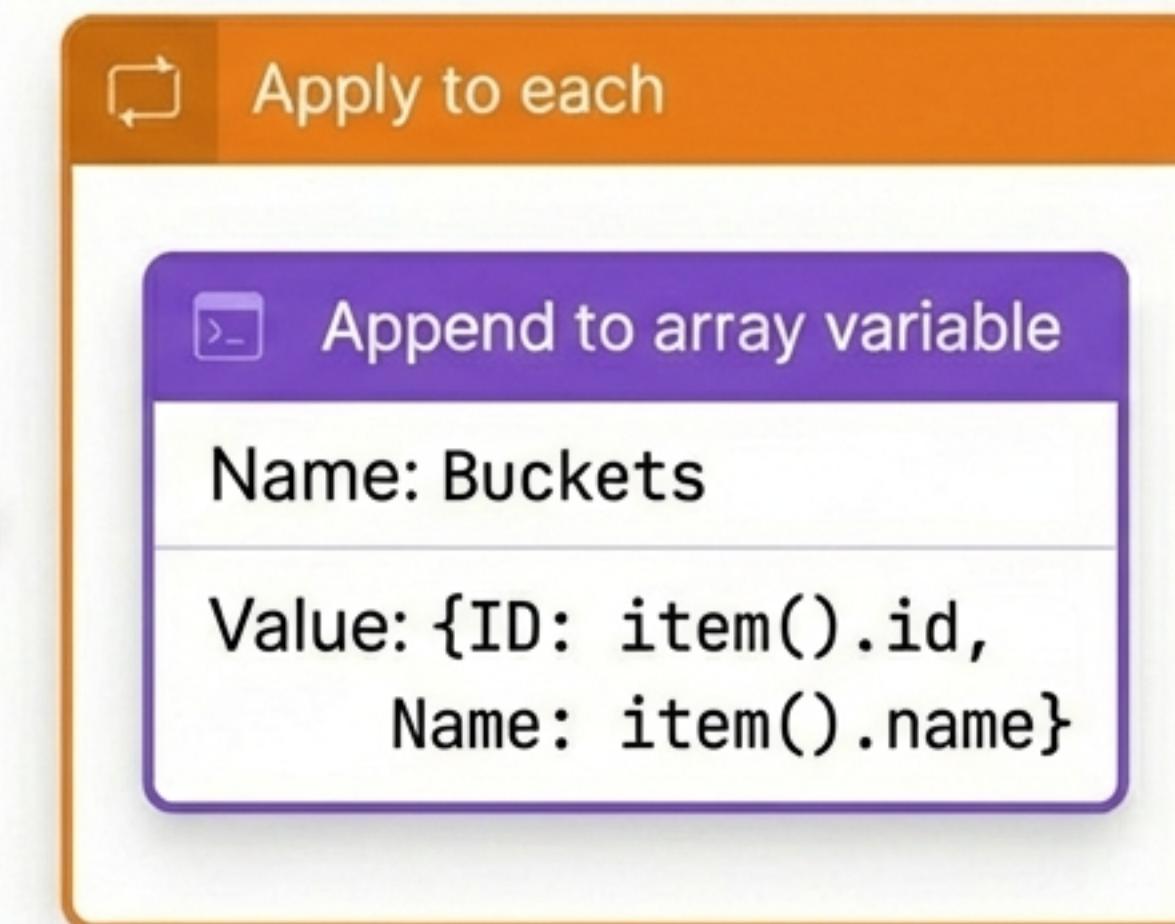
Group Id:
Marketing Team

Plan Id:
Q4 Campaign Launch

Pro Tip: The 'Plan ID' field accepts Custom Values. Use this to dynamically iterate through multiple plans.

Step 4: The Bucket Strategy

Tasks only contain a 'BucketID'. To report on meaningful names (e.g., '**Backlog**'), we must first build a reference directory.



Variable: Buckets



Step 5: Constructing the Bucket Directory

The screenshot shows a step in a workflow editor titled "Append to array variable". The "Name" field is set to "Buckets". The "Value" field contains a JSON object with two properties: "BucketID" and "BucketName". Each property is a dynamic expression: "@{items('Apply_to_each')?['id']} for BucketID and "@{items('Apply_to_each')?['name']} for BucketName". A callout box labeled "Select Dynamic Content: Value ID & Value Name" has arrows pointing to the "id" and "name" fields in the JSON value.

Append to array variable

Name
Buckets

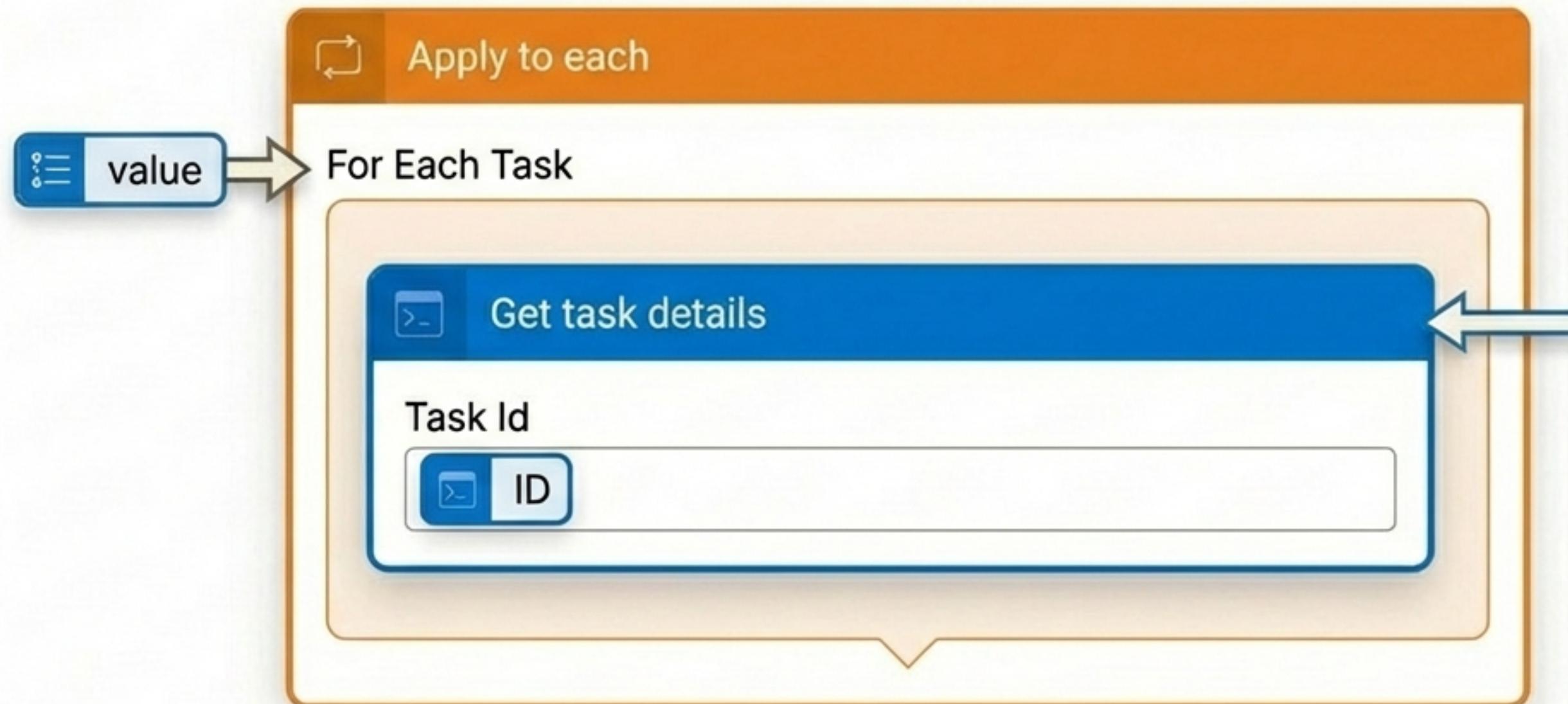
Value

```
{  
    "BucketID": @{items('Apply_to_each')?['id']},  
    "BucketName": @{items('Apply_to_each')?['name']}
```

Select Dynamic Content:
Value ID & Value Name

Step 6: The Main Loop & Detail Retrieval

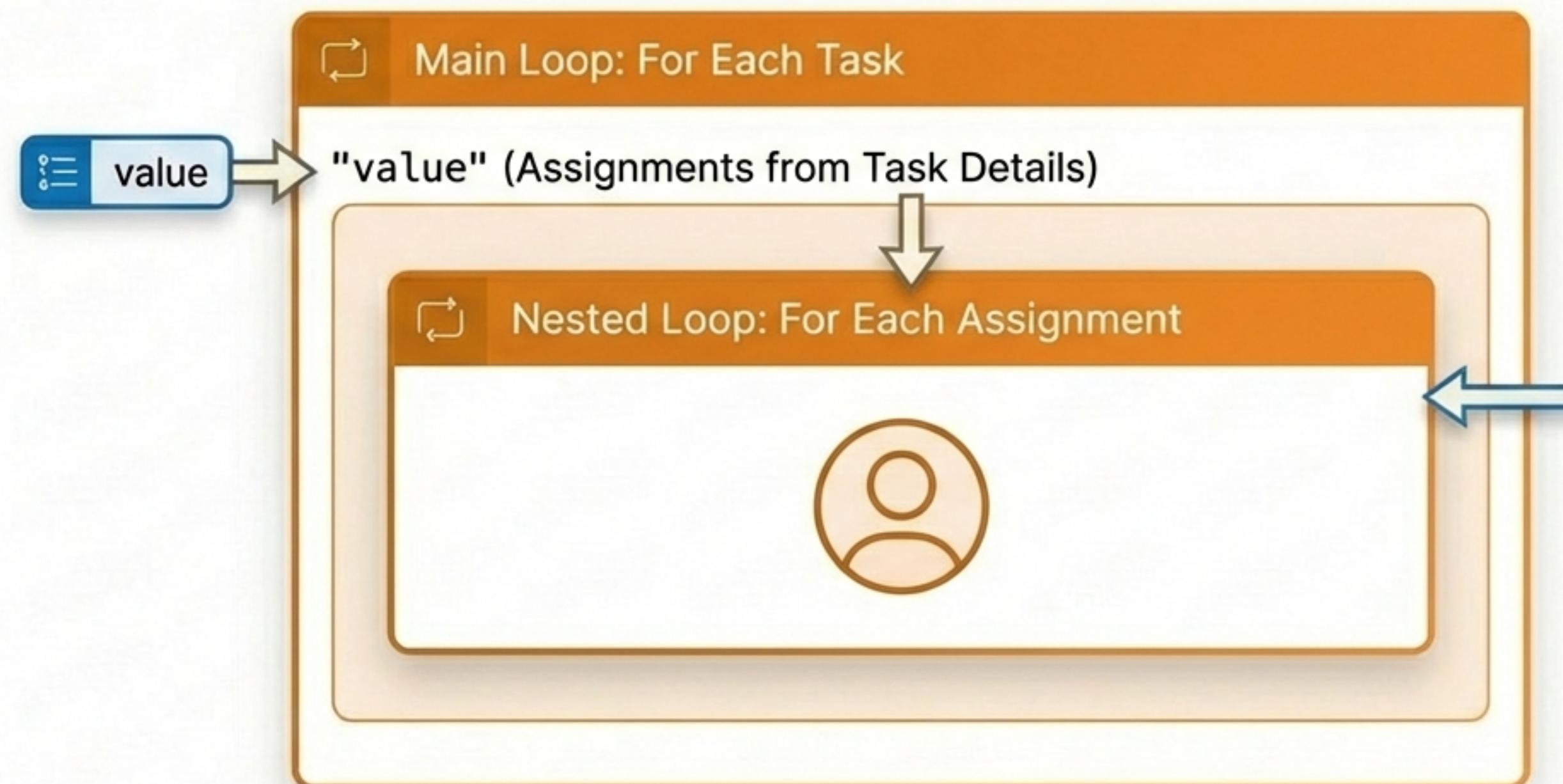
We now begin the core processing engine. We iterate through every task and immediately fetch its full details.



Crucial: “List Tasks” gives the skeleton.
“Get Task Details” gives the meat
(Checklists, Descriptions).

Step 7: The Assignee Challenge

Tasks can have multiple assignees. This requires a nested loop structure.



We must iterate
“Assignments”,
not “Assignees”.

Step 8: Resolving User Profiles

Get user profile (V2)
(Office 365 Users)

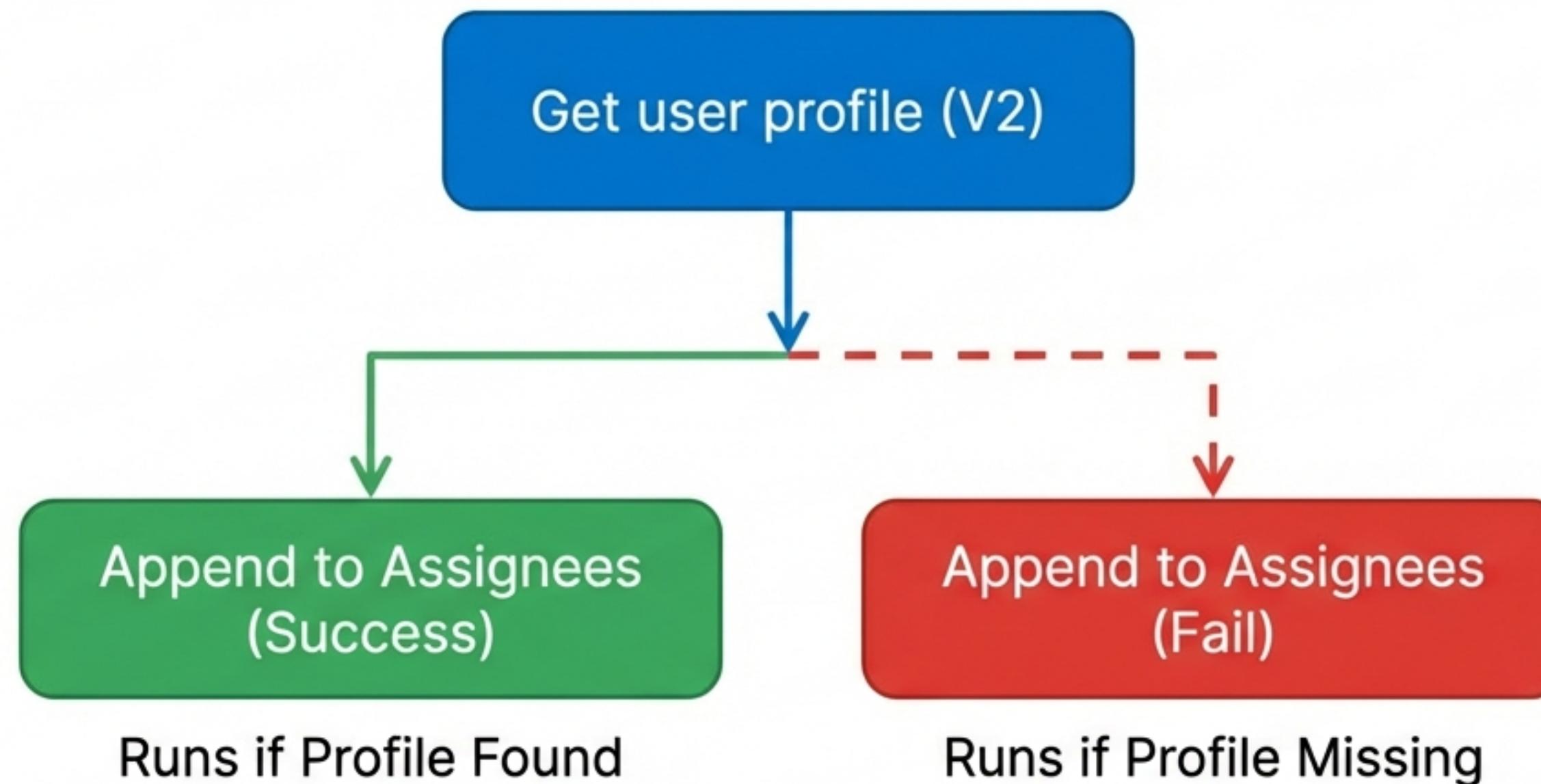
User (UPN): Assignee User ID



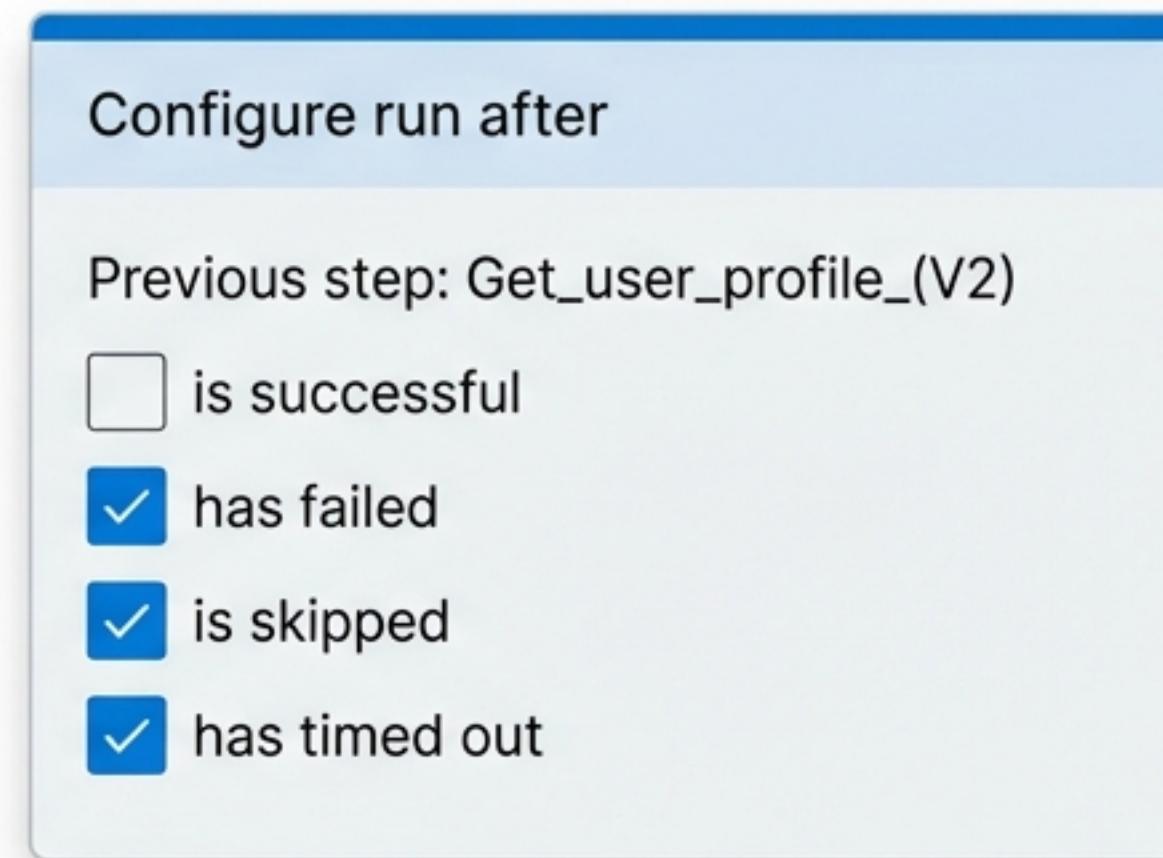
FAILURE POINT: If a user has left the organization, this action will fail. Without error handling, the entire flow crashes here.

Step 9: Advanced Error Handling (Parallel Branches)

We split the timeline to handle success and failure gracefully.



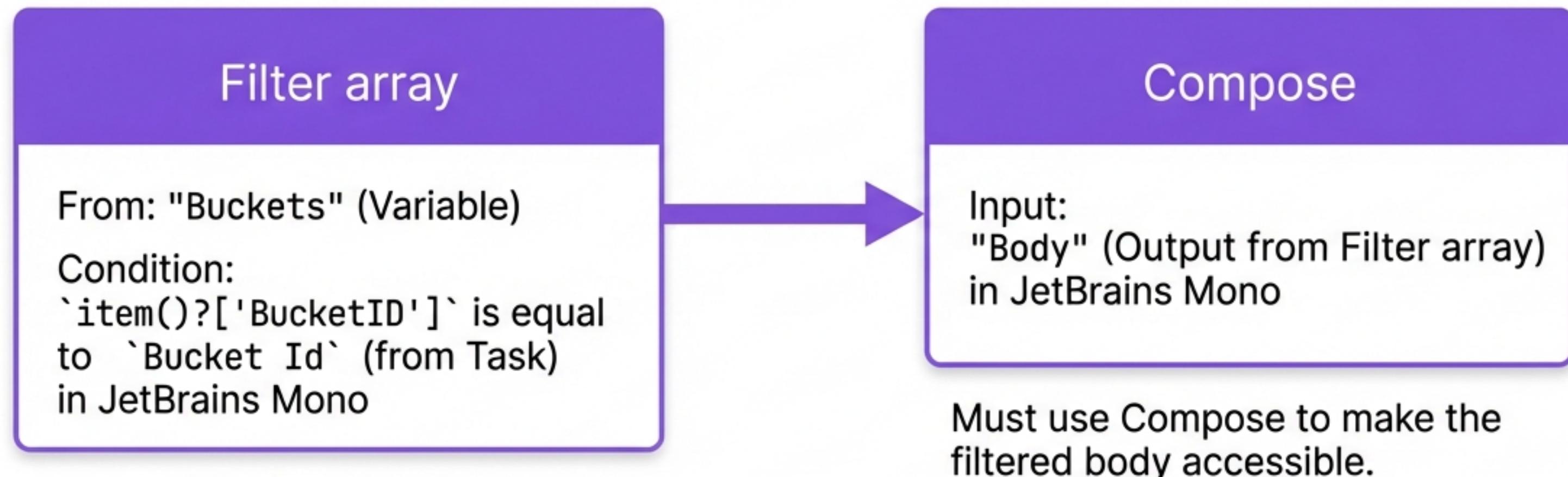
Step 10: Handling the “Ghost User”



```
{  
  "AssigneeName": "Unavailable",  
  "AssigneeID": @{items('Apply_to_each_assignment')?['userId']}  
}
```

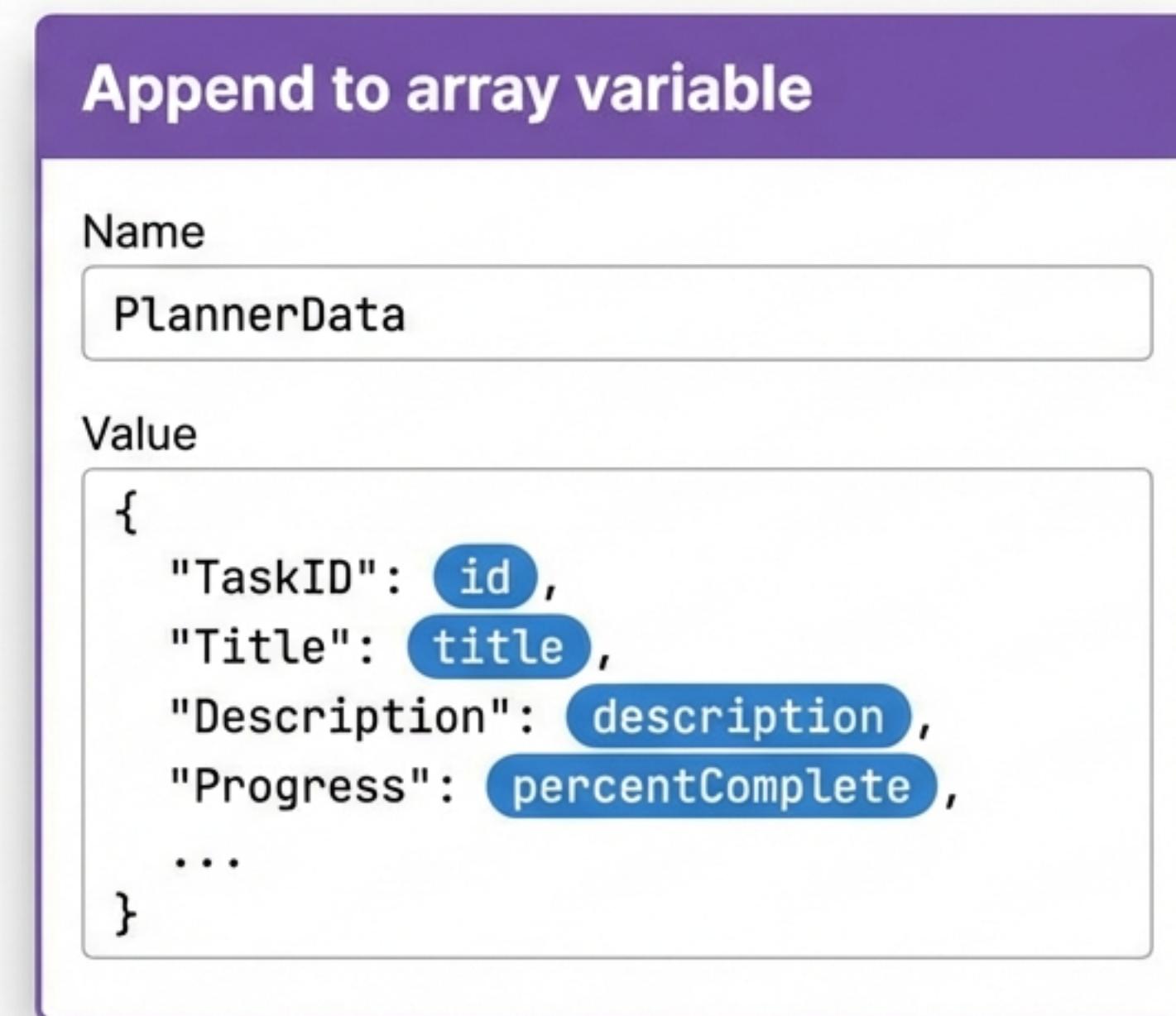
Step 11: Filtering the Buckets

Back in the Main Loop, we match the current task's Bucket ID against our Directory variable.



Step 12: Mapping the Mega-Object (Part 1)

We begin constructing the single JSON object that holds all task data.



Step 13: Mapping the Mega-Object (Part 2)

Injecting the complex arrays and dates.

Append to array variable

Value

```
...
"StartDate": "startDateTime",
"DueDate": "dueDateTime",
"Assignees": @{variables('Assignees')},
"Subtasks": @{outputs('Get_task_details')['checklist']},
"Bucket": @{outputs('Compose_filtered_buckets')}
}
```

Step 14: Decoding Colored Flags

Power Automate sees colors. You see meanings. You must manually map these keys.

Power Automate Value	Your Report Key
	pending
	urgent
	approved

```
"Urgent": @{items('Apply_to_each')?['appliedCategories']?['category3']}
```

WARNING: The Critical Reset

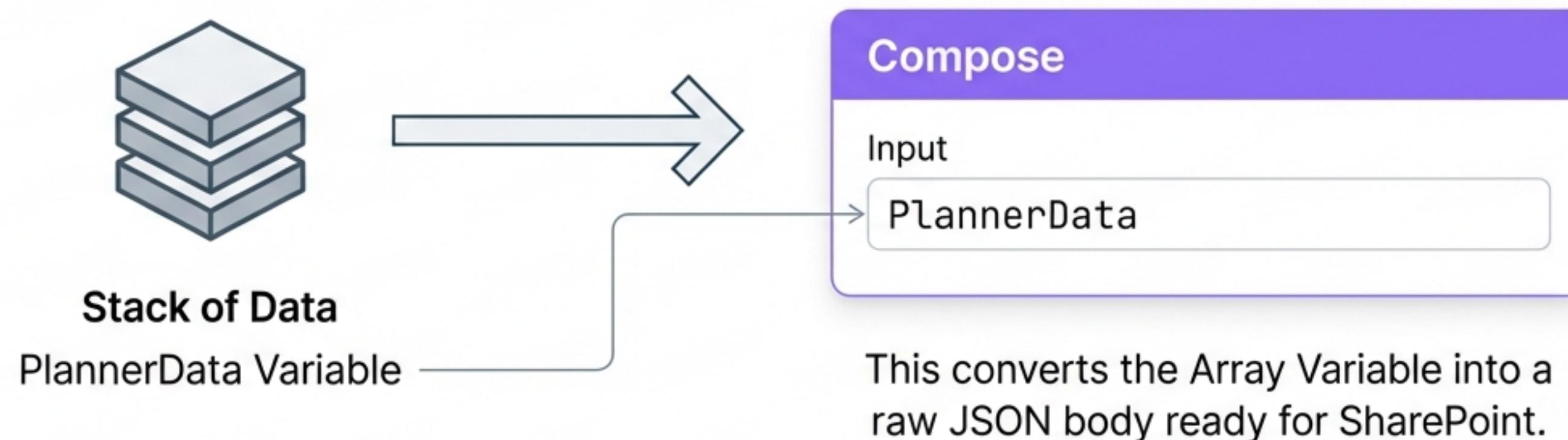
High-contrast alert is anspacily to achieve the elstricolation providers.



You must **reset the Assignees variable to NULL at the end of the loop**. If you skip this, Task 2 will inherit Task 1's **assignees**, creating a snowball effect of incorrect data.

Step 15: Composing the Final Payload

Outside the loops, we prepare the data for file creation.



Step 16: Creating the JSON File

Final step to generate the file on SharePoint.

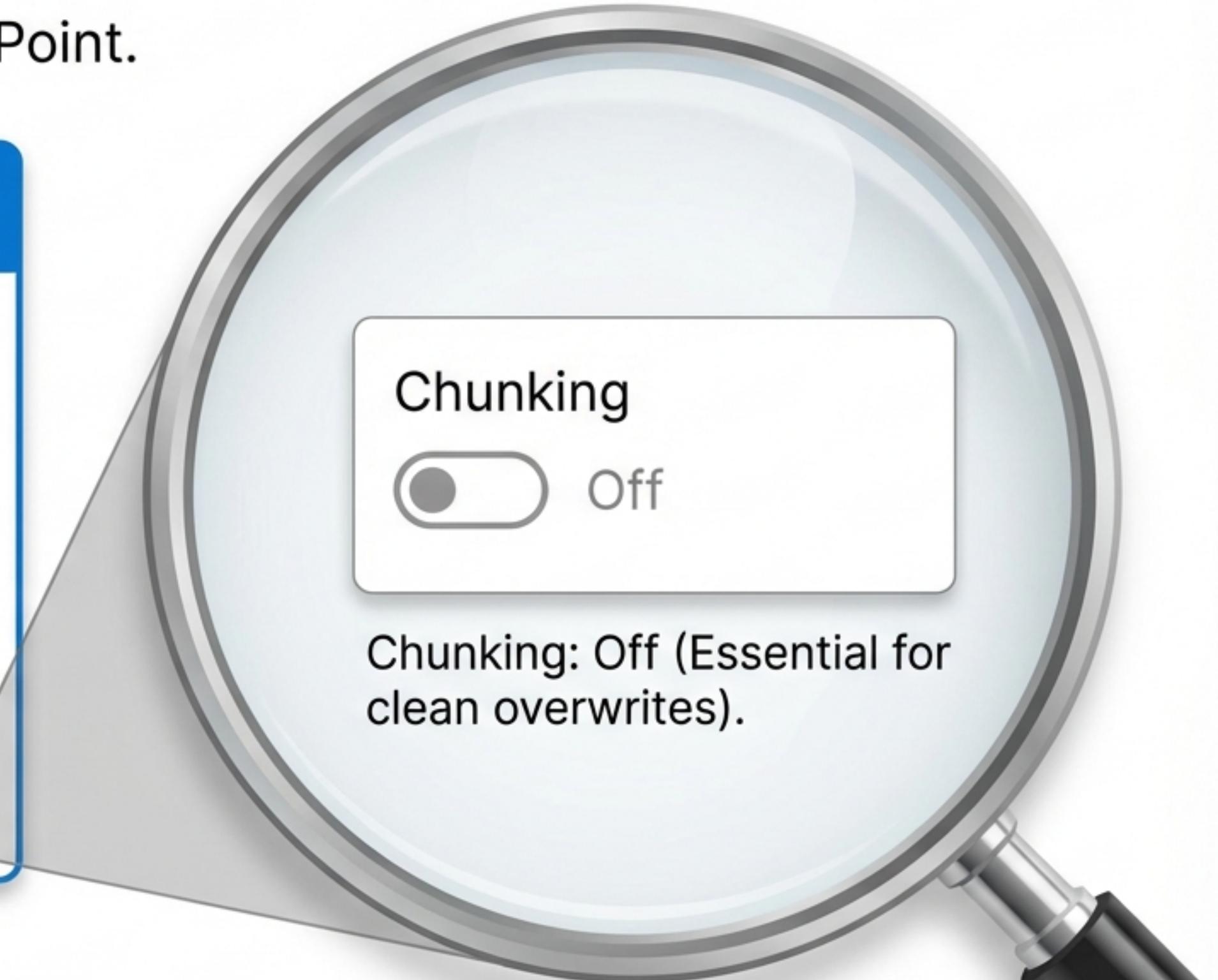
Create file

Folder Path:
/Shared Documents/

File Name:
planner_data.json

File Content:

Settings 



The Result: Reporting Ready

- Resolved Assignee Names
- Proper Bucket Names
- Subtask Details
- Custom Flag Statuses

