Hands-On Lab: Lookup Activity in Azure Data Factory

Author: Dr. Sandeep Kumar Sharma

Lab overview

In this lab we will learn about the **Lookup activity** in Azure Data Factory (ADF). The Lookup activity is essential when you need to fetch a single row or small dataset from a data source and use that value later in the pipeline — for example, to decide whether to run another activity or to pass a configuration value. This is a beginner-level, step-by-step exercise designed so that you can follow along in a live classroom.

Learning objectives

By the end of this lab you will be able to:

- Explain what the Lookup activity does and when to use it.
- Configure a Linked Service and Dataset for a simple source (CSV file / Azure SQL / Azure Blob Storage).
- Add and configure a Lookup activity in a pipeline to read a single row/value.
- Use the Lookup activity output with subsequent activities (For example: If Condition activity or Set Variable activity).
- Troubleshoot common lookup errors.

Prerequisites

Please ensure the following before you begin:

1. An Azure subscription and access to an Azure Data Factory instance. (If you don't have one, ask your instructor to provide sandbox access.)
2. Basic familiarity with ADF Studio (how to open Author -> Pipelines, how to create linked services and datasets).
3. Sample data prepared: a CSV file named `config.csv` with the following content placed in an Azure Blob container or a small table in Azure SQL:

```
settingKey,settingValue
maxRetry,3
isFeatureXEnabled,true
```

1. A resource (Azure Storage account + container) or Azure SQL database where the sample file/table resides.

What is the Lookup Activity?

The Lookup activity in ADF is used to read a row (or a small set of rows) from a source dataset. It does not move data — it only reads and returns the result to the pipeline runtime. You typically use Lookup when:

- You need a configuration value (for example, a date range or a flag) stored in a small file or table.
- You want to retrieve a control record (single row) that influences downstream logic.
- You need to retrieve a list of values and then iterate over them (limited use — for larger lists use another approach).

Important points:

- Lookup by default reads the first row. You can optionally provide a query to control what is returned.
- The output is available in pipeline expressions as `activity('<lookupName>').output.value` (or `.firstRow` for convenience when expecting a single row).
- Lookup is synchronous: the pipeline waits for it to finish.

High-level flow for this lab

1. Create or verify Linked Service to your source (Azure Blob Storage or Azure SQL).
2. Create Dataset pointing to `config.csv` (or to the SQL table).
3. Create a new Pipeline and add a Lookup activity.
4. Configure the Lookup to use the dataset and an optional query.
5. Add a Set Variable or If Condition activity to consume the Lookup result.
6. Debug / Trigger the pipeline and validate the results.
7. Inspect outputs and troubleshoot if needed.

Step-by-step hands-on exercise (very detailed)

Step 1 — Create Linked Service

1. In ADF Studio, click **Manage** (left pane) → **Linked services** → **New**.
2. Choose **Azure Blob Storage** as the connector.
3. Provide:
4. **Name:** `LS_Blob_Sample`
5. **Subscription / Storage account / Authentication method** — fill according to your environment (SAS / Account Key / Managed Identity).
6. Click **Test connection** → ensure it succeeds → **Create**.

Why this matters: The Lookup activity needs a dataset, and the dataset needs a linked service to connect to the actual storage.

Step 2 — Create Dataset for the config file

1. In ADF Studio, go to **Author** → **+ (plus)** → **Dataset**.
2. Choose **Azure Blob Storage** → **DelimitedText**.
3. Configure dataset properties:
4. **Name:** `DS_Config_CSV`
5. **Linked service:** `LS_Blob_Sample` (select from dropdown)
6. **File path:** point to container and folder where `config.csv` is located.
7. **First row as header: Checked** (because our `config.csv` has headers)
8. Schema: Click **Import schema** (optional for clarity). ADF will show the two columns `settingKey`, `settingValue`.
9. **Save** the dataset.

Trainer tip: If using Azure SQL, create a table `Config` with columns `settingKey` and `settingValue`, and create a Dataset of type Azure SQL Table.

Step 3 — Add a Pipeline and Lookup activity

1. In **Author**, click **+** → **Pipeline** → choose **Pipeline**.
2. Rename the pipeline: `PL_Lookup_Config`.
3. From the Activities pane (left side), drag **Lookup** into the pipeline canvas.
4. Click the Lookup activity and set the properties (Settings tab):
5. **Name:** `Lookup_Config`
6. **Source dataset:** `DS_Config_CSV`
7. **Use query:** Optional. For example, to get a specific key:

```
SELECT settingValue FROM Config WHERE settingKey = 'maxRetry'
```

- If using CSV and you don't have a SQL engine, use dataset filters or limit the rows in the CSV and then process `firstRow` in expressions.
- **First row only:** Check this if you are sure you need a single row (recommended for control records).
- Save the pipeline.

Trainer explanation: If you leave the query blank, the Lookup will read the full file and return an array in `output.value`. If `firstRow` is checked, you can safely use `.firstRow.settingValue`.

Step 4 — Add a Set Variable activity to consume Lookup output

1. In the pipeline, add a variable: On the pipeline's canvas top toolbar, click **Variables** → **New**.
2. **Name:** `varMaxRetry`
3. **Type:** String (or int — but ADF pipeline variables are often strings; you can cast later).
4. From Activities, drag **Set Variable** onto canvas, connect Lookup → Set Variable (use the green arrow to indicate success path).
5. Configure Set Variable:
6. **Name:** `SetVar_MaxRetry`

7. **Variable name:** `varMaxRetry`
8. **Value:** Use expression: `@activity('Lookup_Config').output.firstRow.settingValue`

Trainer tip: Use the expression builder to avoid syntax mistakes. You can click the small `Add dynamic content` link under Value and pick from the Lookup output.

Step 5 — Add an If Condition activity to make a decision

1. From Activities, drag **If Condition** to canvas and connect Set Variable → If Condition.
2. Configure the If Condition `Expression` to check the value, for example:

```
@greater(int(variables('varMaxRetry')), 1)
```

1. In the **If true** branch, add a simple activity (like **Web** or **Execute Pipeline**) or **Wait** activity to simulate action. In **If false** branch, add a different activity or leave empty.

Trainer explanation: This demonstrates how Lookup values can drive pipeline control flow.

Step 6 — Debug / Trigger the pipeline and validate

1. Click **Debug** at the top to run the pipeline in interactive mode.
2. Monitor the run in the **Output** pane. Expand the Lookup activity to see its `output` JSON.
3. Example output snippet you might see:

```
"output": {
  "value": [
    {"settingKey": "maxRetry", "settingValue": "3"},
    {"settingKey": "isFeatureXEnabled", "settingValue": "true"}
  ],
  "firstRow": {"settingKey": "maxRetry", "settingValue": "3"}
}
```

1. Verify that the Set Variable activity was assigned the expected value and that the If Condition branched correctly.

Step 7 — Publish and trigger (optional)

1. When satisfied, click **Publish all** and create a trigger if you want scheduled runs.
2. Validate a full production run and monitor via **Monitor** blade.

Example: Using Lookup with Azure SQL (alternate)

If your configuration is in an Azure SQL table named `Config`, you can set the Lookup dataset to an Azure SQL dataset and in the Lookup settings use a query such as:

```
SELECT settingValue FROM Config WHERE settingKey = 'isFeatureXEnabled'
```

Then read the value in pipeline expressions as `@activity('Lookup_Config').output.firstRow.settingValue`.

Common mistakes & troubleshooting (Trainer voice)

- **Lookup returns empty output** — confirm path and file name in dataset, and check access permissions on the linked service. For SQL, verify the query returns rows when run in SQL Management Studio.
- **Schema mismatch** — if `First row as header` is set incorrectly for CSV, column names will shift. Use **Import schema** to validate.
- **Using `vs \*\*` — `.value` returns an array; if you expect a single row prefer `.firstRow` to avoid indexing errors.
- **Expression syntax errors** — use the expression builder and test small expressions first.
- **Large files** — Lookup is not meant for large datasets. If your file has many rows you should use Copy activity or Data Flow.

Validation checklist (what the trainer will look for during lab)

- Linked Service created and tested.
- Dataset pointing to the sample file/table and schema imported.
- Lookup activity configured and returns expected JSON in output.
- Pipeline variable correctly set from Lookup output.
- If Condition branch executed correctly based on the Lookup value.
- Pipeline published (optional).

End of Lab