

## Delta Tables: Understanding the Two Patterns in Fabric Spark for creating Delta Tables

### 1. Save as a table (MANAGED table only)

#### **Example**

```
df.write.format("delta").saveAsTable("managed_products")
```

#### Result

- A managed table is created in the Lakehouse metastore
- Delta files are stored under the default managed table location

The screenshot shows the Microsoft Fabric Data Explorer interface. On the left, the 'OneLake' data item is selected in the 'Data items' section of the Explorer sidebar. In the main workspace, a PySpark (Python) notebook cell is running, showing the command to save a DataFrame as a managed Delta table:

```
1 df.write.format("delta").saveAsTable("managed_products")
```

The execution status shows it was successful in 11 seconds. Below the notebook, a Spark SQL query is run to describe the managed\_delta table:

```
1 %%sql
2 DESCRIBE FORMATTED managed_products;
```

The execution status shows it was successful in 32 seconds. To the right, the 'Inspect' pane displays the schema of the managed\_delta table:

ABC col_name	ABC data_type
ProductID	int
ProductName	string
Category	string
ListPrice	double
5	
6 # Detailed Tab...	
7 Name	spark_catalog.chimcobldhq2ajhp41musjbedo62or54l2monqcc5lmaq3fe pma9b4c9ng.managed_products
8 Type	MANAGED
9 Location	abfss://05dd6b73-f930-4b8d-a2fb-34d884c23a8d@onelake.dfs.fabric.microsoft.com/d8b1dbae-b810-409f-ac22-44fda04b18b1/Tables/dbo/managed_products
10 Provider	delta

### 2. Save as a table and write Delta files to specified path (MANAGED table and Delta Files)

#### **Example**

```
df.write.format("delta").saveAsTable("external_products",
path="abfs_path/external_products")
```

#### Result

- A managed table is created in the Lakehouse metastore
- Delta files are stored under the default managed table location
- Executes the write operation and writes Delta files to the path provided

col_name	data_type	con
ProductID	int	NULL
ProductName	string	NULL
Category	string	NULL
ListPrice	double	NULL
5		
6	# Detailed Tab...	
7	Name	spark_catalog.chimcobldhq2ajbp41rmusjbedo62or54l2monqcc5lmaq3felpma9b4c9ng.external_products
8	Type	MANAGED
9	Location	abfss://05dd6b73-f930-4b8d-a2fb-34d884c23a8d@onelake.dfs.fabric.microsoft.com/d8b1dbae-b810-409f-ac22-44fd0a04b18b1/tables/dbo/external_prod...
10	Provider	delta

Fabric does NOT allow external tables created via Spark. It ignores the path= argument in saveAsTable() and does not link the table to the folder created in the specified path.

Whereas, in **Databricks** it works. Databricks interprets this as:

- Create a metastore table
- Store its data on the custom path
- → Result, TYPE: external table

Eventhough the Microsoft Learn says external tables should be created this way.

#### Create an external table

You can also create external tables, which may be stored somewhere other than the lakehouse, with the schema metadata stored in the lakehouse.

1. In the Explorer pane, in the ... menu for the **Files** folder, select **Copy ABFS path**. The ABFS path is the fully qualified path to the lakehouse Files folder.
2. In a new code cell, paste the ABFS path. Add the following code, using cut and paste to insert the abfs\_path into the correct place in the code:

```
Code Copy
df.write.format("delta").saveAsTable("external_products", path="abfs_path/external_products")
```

3. The full path should look similar to this:

```
Code Copy
%%sql
DESCRIBE FORMATTED external_products;
```

4. Run the cell and in the results, view the Location property for the table. Widen the Data type column to see the full path and notice that the OneLake storage locations ends with /Files/external\_products.

Why Fabric behaves differently?

Fabric Spark enforces:

- **Managed tables only** inside a Lakehouse
- No external tables are created via Spark
- No custom locations for Spark tables

This is because Fabric uses a **unified metastore** and tightly controls table locations for governance, lineage, and OneLake consistency.