Models

Overview

Models in FLUX enable you to perform data transformations using different processing classes. Each model class requires specific formatting in the **Model data** field to define how data should be parsed and transformed.

You can validate your model's behavior by entering sample input in the **Test data** field and running a test to preview the output results.

Model Classes

CLI

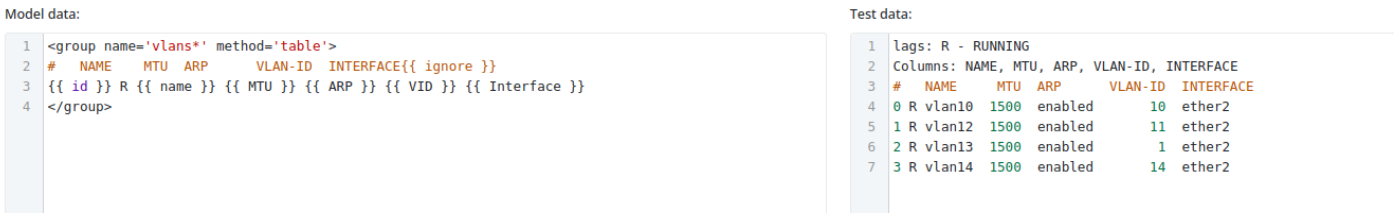
Execute command-line instructions on a device.

Model data format: A command string that will be executed.

Structured Text

Process text containing variables in a structured format.

Model data format: Structured text with embedded variables.



Python

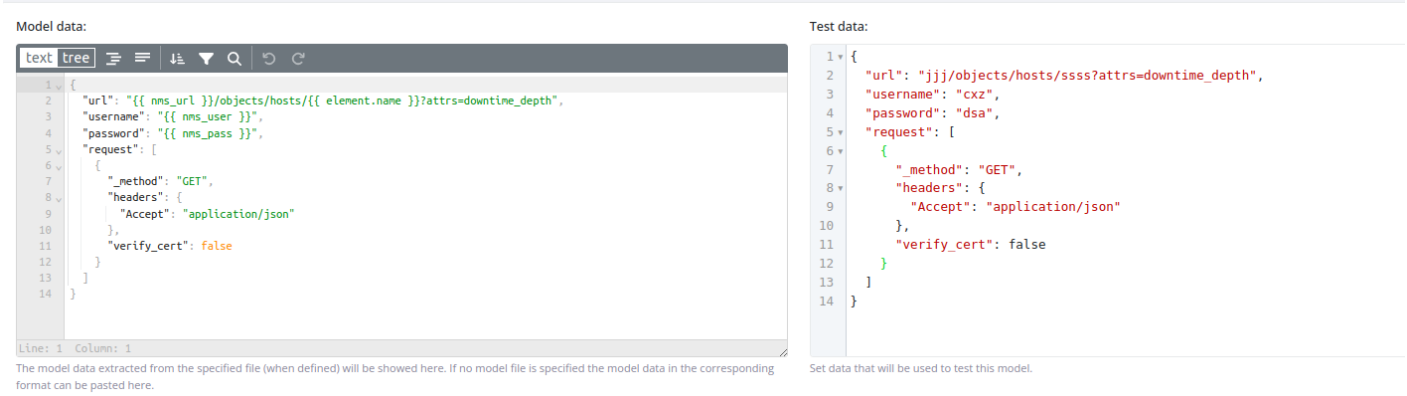
Execute Python code to transform data.

Model data format: Python code that uses a `data` variable, which will be populated with the content from the **Test data** input.

JSON & XML

Parse and transform JSON or XML data structures.

Model data format: JSON structure with variables.

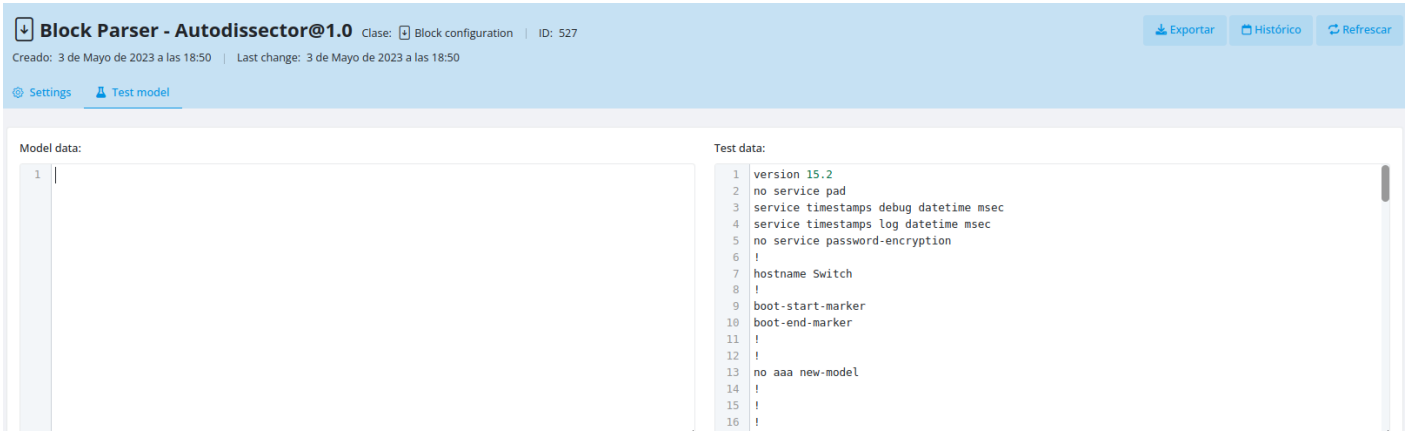


Block

Parse block-formatted files using custom or predefined rules.

Model data format: YAML parser configuration with multiple parsing rules.

Using predefined rules: Leave the **Model data** field empty to automatically detect the block type and parse it using FLUX's built-in rules.

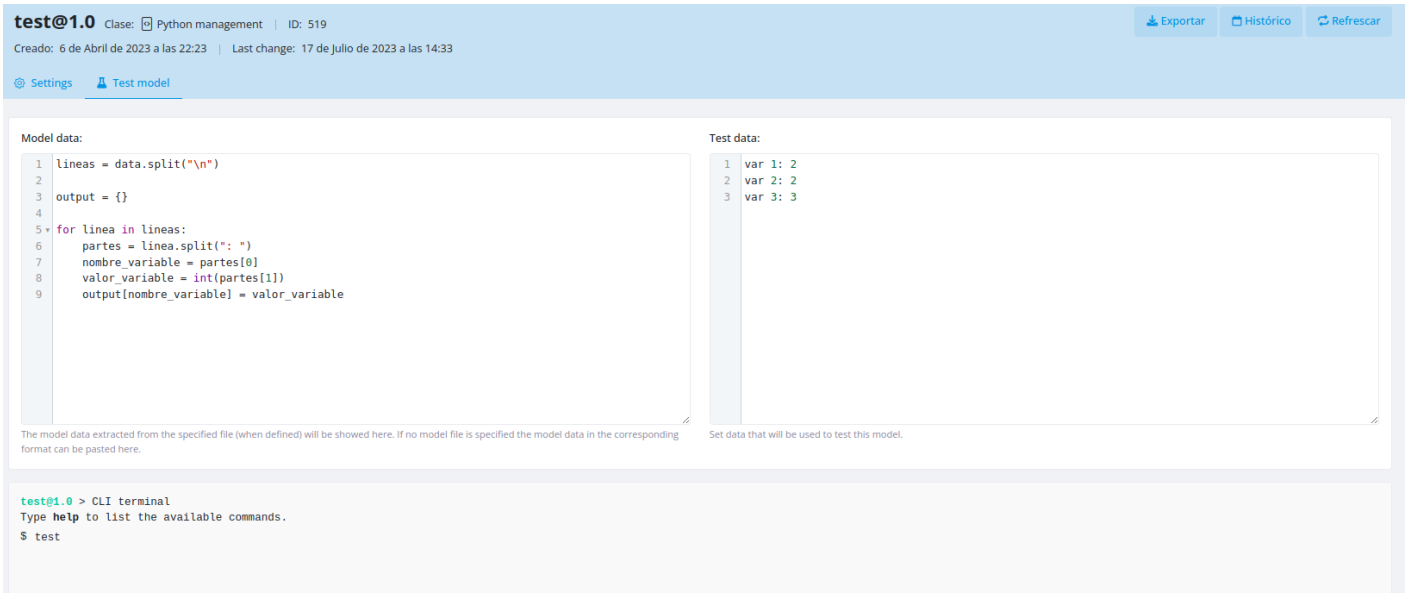


Testing Models

Basic Model Testing

1. Navigate to the **Test model** tab
2. Fill the **Model data** input with the parsing configuration
3. Fill the **Test data** input with sample data
4. Run `test` in the CLI to perform the transformation

Example:



Testing Against Element Context

You can test a model using the context of a specific element by entering the following in the **Test data** input:

```
id={x}
```

Replace `{x}` with the ID of the element whose context you want to use for testing.

Creating Model Templates with OpenAI

FLUX can generate model templates automatically using OpenAI based on your test data.

Prerequisites

1. Create an empty model with your desired class
2. Create an OpenAI helper element with the naming pattern: `openai-helper-{class}` (where `{class}` matches your model class)

Steps

1. Navigate to the **Test model** tab
2. Fill the **Test data** input with your sample data
3. Leave the **Model data** input empty
4. Run `template.create` in the CLI terminal

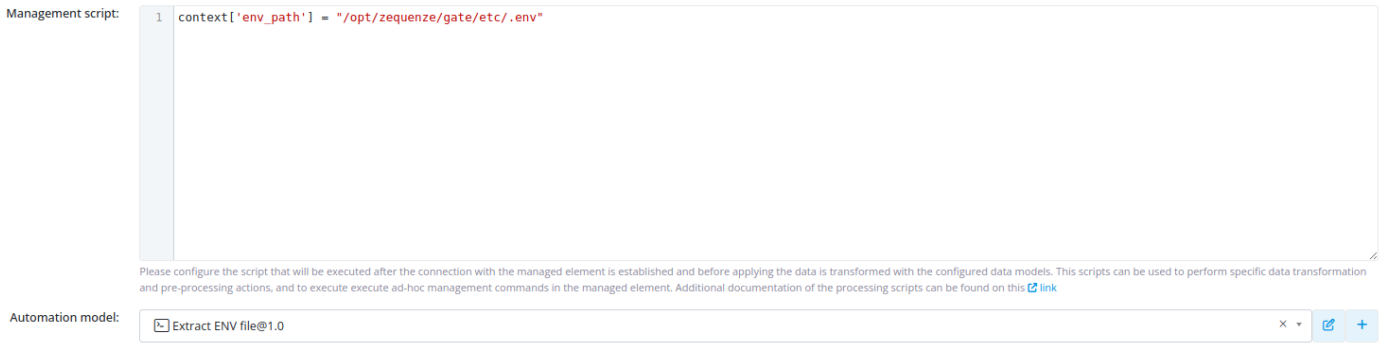
The system will call the `openai-helper-{class}` element, query the OpenAI API, print the generated template, and automatically evaluate it against your test data.

Dynamic Parameters

Models support dynamic variable substitution using element context data. Variables can be loaded in steps prior to model execution, allowing the model to replace them during transformation.

Example configuration:

Example usage:



Revision #2
Created 2026-02-13 23:07:12 UTC by ipena@zequenze.com
Updated 2026-03-05 22:30:35 UTC by ipena@zequenze.com