

# Drupal 7 to 10 Migration: Custom Migrate YAML & Overrides Guide (v2)

---

## 14. How upgrade\_d7\_node Works + Full Consultant Examples

---

Understanding upgrade\_d7\_node

---

You do NOT manually create upgrade\_d7\_node by default.

When you enable migrate\_drupal + migrate\_drupal\_ui and connect your D7 database in settings.php, Drupal scans the D7 site and auto-generates migration plugin IDs at runtime.

Common examples:

- upgrade\_d7\_user
- upgrade\_d7\_node
- upgrade\_d7\_taxonomy\_term
- upgrade\_d7\_file

They exist temporarily inside memory during drush migrate-import.

Where to find upgrade\_d7\_node?

---

There is no physical file by default.

You see upgrade\_d7\_node listed ONLY when you run:  
drush migrate-status

How to customize upgrade\_d7\_node?

---

You have two safe consultant methods:

1■■■ Easiest: hook\_migration\_plugins\_alter()

---

Recommended for real projects. No full override needed.

Example:

```
function mymodule_migration_plugins_alter(array &$definitions) {
  if (isset($definitions['upgrade_d7_node'])) {
    $definitions['upgrade_d7_node']['process']['title'] = [
      'plugin' => 'my_custom_process',
      'source' => 'title',
    ];
  }
}
```

2■■■ Full custom YAML override using migrate\_plus

---

Create a custom module, example: my\_migration

Folder structure:

- my\_migration/config/install/migrate\_plus.migration.upgrade\_d7\_node.yml
- my\_migration/config/install/migrate\_plus.migration.upgrade\_d7\_taxonomy\_term.yml

Example 1: Node migration YAML override

---

```
id: upgrade_d7_node
label: 'Migrate D7 nodes'
```

```

migration_group: migrate_drupal_7
source:
  plugin: d7_node
process:
  title:
    plugin: my_custom_process
    source: title
  body/value:
    plugin: clean_body_text
    source: body
  created: created
  changed: changed
destination:
  plugin: entity:node
migration_dependencies: {}

```

#### Example 2: Taxonomy migration YAML override

```

-----
id: upgrade_d7_taxonomy_term
label: 'Migrate D7 taxonomy terms'
migration_group: migrate_drupal_7
source:
  plugin: d7_taxonomy_term
process:
  name: name
  description: description
  vid: vid
destination:
  plugin: entity:taxonomy_term
migration_dependencies: {}

```

#### Where to create custom Process Plugin?

Create inside your module folder:  
 mymodule/src/Plugin/migrate/process/MyCustomProcess.php

#### Example code:

```

namespace Drupal\mymodule\Plugin\migrate\process;
use Drupal\migrate\Plugin\migrate\process\ProcessPluginBase;
use Drupal\migrate\MigrateExecutableInterface;
use Drupal\migrate\Row;

/**
 * @MigrateProcessPlugin(id = "my_custom_process")
 */
class MyCustomProcess extends ProcessPluginBase {
  public function transform($value, MigrateExecutableInterface $migrate_executable, Row $row, $current_step) {
    return strtoupper($value) . ' - MIGRATED';
  }
}

```

#### Summary of full steps

1. Enable migrate, migrate\_drupal, migrate\_drupal\_ui
2. Add D7 DB connection to settings.php
3. Run drush migrate-status (you will see upgrade\_d7\_node)
4. Create custom module for YAML or plugin overrides
5. Use hook\_migration\_plugins\_alter() to safely inject plugins
6. Run drush migrate-import upgrade\_d7\_node
7. Your custom plugin will run automatically

This is the full real-world consultant workflow.

## ----- 15. Real Consultant Examples: Why Create Custom Process Plugins? -----

### Case 1: Clean legacy body field text -----

Client D7 body fields had random control characters that broke display.

Input:

"This is a \x00 messy \x07 body text."

Plugin: clean\_body\_text

Removes all control chars.

Output:

"This is a messy body text."

### Case 2: Add migration trace suffix to titles -----

Client wanted all imported nodes to append "- Migrated" to title.

Input:

"Summer Festival 2024"

Plugin: my\_custom\_process

Output:

"Summer Festival 2024 - Migrated"

### Case 3: Map legacy category codes -----

D7 site used custom codes (A, B, C). Client wanted friendly labels.

Input:

"old\_category\_code\_B"

Plugin: category\_code\_mapper (mapping array)

Output:

"Blog"

### Case 4: Format dates to ISO format -----

Legacy D7 site stored event dates in random formats.

Input:

"7th April 2024"

Plugin: format\_date\_to\_iso

Output:

"2024-04-07"

#### Case 5: Enrich records with external API data

-----  
Client wanted every migrated company node to include Google Places rating.

Input field:

"OpenAI HQ, San Francisco, CA"

Plugin: enrich\_company\_rating

External API returns rating: 4.8

Output:

field\_company\_rating = 4.8

Summary

-----  
Custom Process Plugins solve real-world project needs:

- Clean bad legacy content
- Append info or transform strings
- Map internal codes to readable labels
- Call external APIs to enhance data
- Standardize formats for consistent output

## ----- 14. End-to-End Understanding: Custom Migrate YAML + Plugins -----

### Example 1: Node migration YAML override -----

This example shows how a real D7 → D10 migration would use both core source plugins AND your own custom Process Plugins.

Your custom module: my\_migration

Place this file at:

my\_migration/config/install/migrate\_plus.migration.upgrade\_d7\_node.yml

YAML content:

```
id: upgrade_d7_node
label: 'Migrate D7 nodes'
migration_group: migrate_drupal_7
source:
  plugin: d7_node
process:
  title:
    plugin: my_custom_process
    source: title
  body/value:
    plugin: clean_body_text
    source: body
  created: created
  changed: changed
destination:
  plugin: entity:node
migration_dependencies: {}
```

What happens?

- Core pulls node data from D7
- title field runs through my\_custom\_process (adds suffix etc.)
- body field runs through clean\_body\_text (removes bad chars)
- Nodes are saved in Drupal 10

### Example 2: Taxonomy term migration YAML override -----

File: my\_migration/config/install/migrate\_plus.migration.upgrade\_d7\_taxonomy\_term.yml

```
id: upgrade_d7_taxonomy_term
label: 'Migrate D7 taxonomy terms'
migration_group: migrate_drupal_7
source:
  plugin: d7_taxonomy_term
process:
  name: name
  description: description
```

```
vid: vid
destination:
  plugin: entity:taxonomy_term
```

#### Summary

-----

This is a real-world D7 → D10 consultant workflow:

1. Create custom module (my\_migration)
2. Add override YAML files under config/install
3. Create any custom Process Plugins under src/Plugin/migrate/process
4. Run standard:  
    drush migrate-import upgrade\_d7\_node  
    drush migrate-import upgrade\_d7\_taxonomy\_term
5. Your Process Plugins run automatically inside normal migrate-import

#### Result:

Client gets enhanced, cleaned, enriched Drupal 10 content without touching core migrate behavior.

## ----- 16. End-to-End Flow: How d7\_taxonomy\_term Works with Plugins -----

When you run:

```
drush migrate-import upgrade_d7_taxonomy_term
```

What happens?  
-----

Your YAML file:

```
id: upgrade_d7_taxonomy_term
source:
  plugin: d7_taxonomy_term
process:
  name:
    plugin: clean_term_name
    source: name
  description: description
  vid: vid
destination:
  plugin: entity:taxonomy_term
```

Flow:

1. Migrate API loads YAML migration definition.
2. Source Plugin d7\_taxonomy\_term reads D7 taxonomy\_term\_data table.
3. Each term row is processed:
  - clean\_term\_name Process Plugin is called to clean name field.
4. Drupal 10 taxonomy\_term entity is created.

Your Custom Process Plugin:  
-----

```
namespace Drupal\mymodule\Plugin\migrate\process;
```

```
use Drupal\migrate\Plugin\migrate\process\ProcessPluginBase;
use Drupal\migrate\MigrateExecutableInterface;
use Drupal\migrate\Row;
```

```
/**
```

```
 * Cleans taxonomy term name.
```

```
 *
```

```
 * @MigrateProcessPlugin(id = "clean_term_name")
```

```
 */
```

```
class CleanTermName extends ProcessPluginBase {
```

```
  public function transform($value, MigrateExecutableInterface $migrate_executable, Row $row, $
```

```
    $value = preg_replace('/[^A-Za-z0-9 ]/', '', $value);
```

```
    return trim($value);
```

```
  }
```

```
}
```

Example Execution:



-----		
	Input (D7 term)	Output (D10 term)
	-----	-----
	"Event *** Name!!! "	"Event Name"
	"Summer #\$\$ Festival "	"Summer Festival"
	" Drupal Rocks!!! "	"Drupal Rocks"

Summary:

-----

drush migrate-import triggers:

→ Source Plugin to get data

→ Your Process Plugin runs to clean name

→ Taxonomy term entity is created

This is true real-world consultant workflow.

You modify only YAML + plugin.

Queue, batch & core migrate system remain untouched.