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, $
    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, $d
    $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.