Coursedog CSV Import Guide

Table of Contents

1. Introduction
2. General CSV Formatting Guidelines
3. Entity-Specific CSV Templates
    - Users
    - Courses
    - Departments
    - Programs
    - Degree Maps
    - Free Form Requirements
    - Simple Requirements
    - Meeting Patterns
    - Learning Outcomes
    - Document Items
    - Rooms
    - Instructors
    - Sections
4. CSV to JSON Mapping
5. Common Issues and Troubleshooting
6. Best Practices

Introduction

This guide provides comprehensive information on CSV (Comma-Separated Values) formatting and import requirements for the Coursedog platform. Whether you're importing course data, program information, or user details, following these guidelines will ensure smooth data integration and minimize errors.

CSV files are a common format for exchanging data between systems due to their simplicity and universal support. However, proper formatting is crucial for successful data imports into Coursedog.

General CSV Formatting Guidelines

File Format Requirements

- **File Extension**: Save files with `.csv` extension
- **Encoding**: UTF-8 (without BOM)
- **Line Endings**: Standard line endings (LF or CRLF consistently)
- **Delimiter**: Comma (`,`)
- **Text Qualifier**: Double quotes (`"`) for fields containing commas, quotes, or line breaks

Header Row

- The first row must contain column headers
- Header names should match Coursedog's expected field names
- Headers are case-sensitive
- Avoid special characters in headers
- Use underscores instead of spaces in header names

Data Rows

- Each row represents one record
- All rows must have the same number of columns as the header row
- Empty values should be represented as empty strings (two commas with nothing between them)
- Text containing commas, quotes, or line breaks must be enclosed in double quotes
- Double quotes within quoted text must be escaped by doubling them (`""`)

Example of Well-Formatted CSV

```
id,name,description,start_date,is_active
101,"Computer Science Department","Department focused on computer science, programming, and
102,Mathematics Department,"Department of mathematics",2023-09-01,true
```

Entity-Specific CSV Templates

Users

Users represent individuals who interact with the Coursedog system, including faculty, staff, and administrators.

Required Fields

Field	Description	Example	Format
email	User's email address (unique identifier)	john.doe@university.edu	Valid email format
firstName	User's first name	John	Text
lastName	User's last name	Doe	Text
role	User's role in the system	Faculty	Text (valid roles: Faculty, Staff, Admin)

Optional Fields

Field	Description	Example	Format
department	User's department	Computer Science	Text
title	User's title	Professor	Text
phone	User's phone number	555-123-4567	Text
status	User's status	Active	Text (Active/Inactive)

Example CSV

```
email,firstName,lastName,role,department,title,phone,status
john.doe@university.edu,John,Doe,Faculty,Computer Science,Professor,555-123-4567,Active
jane.smith@university.edu,Jane,Smith,Admin,Registrar,Director,555-987-6543,Active
```

Courses

Courses represent academic offerings within the curriculum.

Required Fields

Field	Description	Example	Format
code	Unique course code	CS101	Text (typically department code + nu
subject	Subject code	CS	Text (typically 2-4 characters)
courseNumber	Course number	101	Text or Number
title	Course title	Introduction to Programming	Text
credits	Number of credits	3	Number (can be decimal, e.g., 3.5)

Optional Fields

Field	Description	Example	Format
description	Course description	"This course introduces basic programming concepts"	Text
department	Department offering the course	Computer Science	Text
effectiveDate	Date when course becomes effective	2023-09-01	YYYY-MM-DD
endDate	Date when course is no longer offered	2028-08-31	YYYY-MM-DD
prerequisites	Course prerequisites	CS100	Text (see Prerequisites section)
corequisites	Course corequisites	CS101L	Text

Example CSV

```
code,subject,courseNumber,title,credits,description,department,effectiveDate,endDate
CS101,CS,101,Introduction to Programming,3,"This course introduces basic programming concept
MATH201,MATH,201,Calculus I,4,"Introduction to differential calculus",Mathematics,2023-09-01
```

Departments

Departments represent academic or administrative units within the institution.

Required Fields

Field	Description	Example	Format
code	Unique department code	CS	Text (typically 2-4 characters)
name	Department name	Computer Science	Text

Optional Fields

Field	Description	Example	Format
description	Department description	"Department focused on computer science"	Text
parentDepartment	Parent department code	ENGR	Text
chair	Department chair's email	john.doe@university.edu	Email
status	Department status	Active	Text (Active/Inactive)

Example CSV

```
code,name,description,parentDepartment,chair,status
CS,Computer Science,"Department focused on computer science",ENGR,john.doe@university.edu,Ac
MATH,Mathematics,"Department of mathematics",,jane.smith@university.edu,Active
```

Programs

Programs represent degree programs, certificates, or other academic offerings.

Required Fields

Field	Description	Example	Format
code	Unique program code	BSCS	Text

Field	Description	Example	Format
title	Program title	Bachelor of Science in Computer Science	Text
type	Program type	Major	Text (Major, Minor, Certificate, et

Optional Fields

Field	Description	Example	Format
description	Program description	"Comprehensive program in computer science"	Text
department	Department offering the program	CS	Text
totalCredits	Total credits required	120	Number
level	Program level	Undergraduate	Text (Undergraduate, Graduate)
status	Program status	Active	Text (Active/Inactive)

Example CSV

```
code,title,type,description,department,totalCredits,level,status
BSCS,Bachelor of Science in Computer Science,Major,"Comprehensive program in computer scienc
MINMATH,Mathematics Minor,Minor,"Supplementary program in mathematics",MATH,18,Undergraduate
```

Degree Maps

Degree maps outline the recommended sequence of courses for completing a program.

Required Fields

Field	Description	Example	Format
programCode	Program code the map belongs to	BSCS	Text
name	Name of the degree map	Computer Science 4-Year Plan	Text

Optional Fields

Field	Description	Example	Format
description	Description of the degree map	"Recommended 4-year plan for CS majors"	Text
totalCredits	Total credits in the map	120	Number
version	Version of the degree map	2023	Text
status	Status of the degree map	Active	Text (Active/Inactive)

Term and Course Fields For each term in the degree map, include columns with this naming pattern:

Field	Description	Example	Format
termN.name	Name of the Nth term	Fall Year 1	Text
termN.courseM.code	Code of the Mth course in term N	CS101	Text
termN.courseM.credits	Credits for the Mth course in term N	3	Number

Example CSV

```
programCode,name,description,totalCredits,version,term1.name,term1.course1.code,term1.course
BSCS,Computer Science 4-Year Plan,"Recommended 4-year plan for CS majors",120,2023,Fall Year
```

Free Form Requirements

Free form requirements allow for flexible text-based requirement definitions.

Required Fields

Field	Description	Example	Format
programCode	Program code the requirement belongs to	BSCS	Text
name	Name of the requirement	Core Requirements	Text
description	Description of the requirement	"Students must complete the following courses"	Text

Optional Fields

Field	Description	Example	Format
credits	Credits associated with the requirement	30	Number
order	Display order of the requirement	1	Number
status	Status of the requirement	Active	Text (Active/Inactive)

Example CSV

```
programCode,name,description,credits,order,status
BSCS,Core Requirements,"Students must complete the following courses: CS101, CS201, CS301",3
BSCS,Elective Requirements,"Students must complete 15 credits of CS electives",15,2,Active
```

Simple Requirements

Simple requirements define structured course requirements with specific rules.

Required Fields

Field	Description	Example	Format
programCode	Program code the requirement belongs to	BSCS	Text
name	Name of the requirement	Core Requirements	Text
type	Type of requirement	AllOf	Text (AllOf, AnyOf, NOf)

Optional Fields

Field	Description	Example	Format
credits	Credits required	30	Number
courses	List of course codes	CS101|CS201|CS301	Pipe-delimited list
numberOfCourses	Number of courses required (for NOf type)	3	Number
order	Display order of the requirement	1	Number
status	Status of the requirement	Active	Text (Active/Inactiv

Example CSV

```
programCode,name,type,credits,courses,numberOfCourses,order,status
BSCS,Core Requirements,AllOf,30,CS101|CS201|CS301,,1,Active
BSCS,Elective Requirements,NOf,15,CS401|CS402|CS403|CS404|CS405,3,2,Active
```

Meeting Patterns

Meeting patterns define when and where courses meet.

Required Fields

Field	Description	Example	Format
code	Unique meeting pattern code	MP001	Text
name	Name of the meeting pattern	MWF Morning	Text
days	Days of the week	Monday|Wednesday|Friday	Pipe-delimited list
startTime	Start time	09:00	HH:MM (24-hour format)
endTime	End time	09:50	HH:MM (24-hour format)

Optional Fields

Field	Description	Example	Format
location	Location of the meeting	Room 101	Text
building	Building code	SCI	Text
startDate	Start date of the pattern	2023-09-01	YYYY-MM-DD
endDate	End date of the pattern	2023-12-15	YYYY-MM-DD
status	Status of the pattern	Active	Text (Active/Inactive)

Example CSV

```
code,name,days,startTime,endTime,location,building,startDate,endDate,status
MP001,MWF Morning,Monday|Wednesday|Friday,09:00,09:50,Room 101,SCI,2023-09-01,2023-12-15,Act
MP002,TR Afternoon,Tuesday|Thursday,14:00,15:15,Room 202,ENG,2023-09-01,2023-12-15,Active
```

Learning Outcomes

Learning outcomes define the expected knowledge or skills students should acquire.

Required Fields

Field	Description	Example	Format
code	Unique outcome code	LO001	Text
description	Description of the outcome	“Students will be able to write basic programs”	Text

Optional Fields

Field	Description	Example	Format
category	Category of the outcome	Technical Skills	Text
level	Level of the outcome	Introductory	Text
programCode	Associated program code	BSCS	Text
courseCode	Associated course code	CS101	Text
status	Status of the outcome	Active	Text (Active/Inactive)

Example CSV

```
code,description,category,level,programCode,courseCode,status
LO001,"Students will be able to write basic programs",Technical Skills,Introductory,BSCS,CS1
LO002,"Students will understand fundamental algorithms",Conceptual Knowledge,Intermediate,BS
```

Document Items

Document items represent content elements in catalog pages or other documents.

Required Fields

Field	Description	Example	Format
code	Unique item code	DI001	Text
type	Type of document item	Text	Text (Text, Image, Table, etc.)
content	Content of the item	“Welcome to the Computer Science Department”	Text

Optional Fields

Field	Description	Example	Format
title	Title of the item	Welcome Message	Text
order	Display order of the item	1	Number
pageCode	Associated page code	CS_HOME	Text
status	Status of the item	Active	Text (Active/Inactive)

Example CSV

```
code,type,content,title,order,pageCode,status
DI001,Text,"Welcome to the Computer Science Department",Welcome Message,1,CS_HOME,Active
DI002,Image,https://example.com/cs_image.jpg,Department Image,2,CS_HOME,Active
```

Rooms

Rooms represent physical spaces where classes or events can be held.

Required Fields

Field	Description	Example	Format
code	Unique room code	SCI101	Text
name	Room name	Science Building Room 101	Text
building	Building code	SCI	Text

Optional Fields

Field	Description	Example	Format
capacity	Room capacity	30	Number
type	Room type	Classroom	Text
features	Room features	Projector|Whiteboard	Pipe-delimited list
floor	Floor number	1	Number
status	Room status	Active	Text (Active/Inactive)

Example CSV

```
code,name,building,capacity,type,features,floor,status
SCI101,Science Building Room 101,SCI,30,Classroom,Projector|Whiteboard,1,Active
ENG202,Engineering Building Room 202,ENG,25,Lab,Computers|Projector,2,Active
```

Instructors

Instructors represent faculty members who teach courses.

Required Fields

Field	Description	Example	Format
email	Instructor's email (unique identifier)	john.doe@university.edu	Valid email format
firstName	Instructor's first name	John	Text
lastName	Instructor's last name	Doe	Text

Optional Fields

Field	Description	Example	Format
department	Instructor's department	Computer Science	Text
title	Instructor's title	Professor	Text
phone	Instructor's phone number	555-123-4567	Text
office	Instructor's office	SCI 301	Text
status	Instructor's status	Active	Text (Active/Inactive)

Example CSV

```
email,firstName,lastName,department,title,phone,office,status
john.doe@university.edu,John,Doe,Computer Science,Professor,555-123-4567,SCI 301,Active
jane.smith@university.edu,Jane,Smith,Mathematics,Associate Professor,555-987-6543,MATH 202,A
```

Sections

Sections represent specific offerings of a course in a term.

Required Fields

Field	Description	Example	Format
courseCode	Course code	CS101	Text
term	Term code	FALL2023	Text
sectionNumber	Section number	001	Text

Optional Fields

Field	Description	Example	Format
crn	Course Reference Number	12345	Text
instructorEmail	Instructor's email	john.doe@university.edu	Email
meetingPatternCode	Meeting pattern code	MP001	Text
roomCode	Room code	SCI101	Text
capacity	Section capacity	30	Number
enrollmentCount	Current enrollment	25	Number
waitlistCapacity	Waitlist capacity	5	Number
waitlistCount	Current waitlist count	2	Number
status	Section status	Active	Text (Active/Scheduled/Cance

Example CSV

```
courseCode,term,sectionNumber,crn,instructorEmail,meetingPatternCode,roomCode,capacity,enrol
CS101,FALL2023,001,12345,john.doe@university.edu,MP001,SCI101,30,25,5,2,Active
MATH201,FALL2023,002,12346,jane.smith@university.edu,MP002,MATH202,25,20,5,0,Active
```

CSV to JSON Mapping

Coursedog internally represents data as JSON objects. Understanding this mapping helps create better CSV imports.

Basic Mapping Principles

1. Each CSV row becomes a JSON object
2. CSV headers become JSON property names
3. Nested properties use dot notation in headers
4. Arrays use indexed notation or delimiters

Example Mapping

CSV:

```
code,title,credits,customFields.department,customFields.level
MATH101,College Algebra,3,Mathematics,Undergraduate
```

JSON:

```
{
  "code": "MATH101",
  "title": "College Algebra",
  "credits": 3,
  "customFields": {
    "department": "Mathematics",
    "level": "Undergraduate"
  }
}
```

Handling Complex Data Structures

Nested Objects Use dot notation in headers to represent nested objects:

```
code,instructor.firstName,instructor.lastName,instructor.email
MATH101,John,Doe,john.doe@university.edu
```

Arrays For simple arrays, use pipe-delimited values:

```
code,title,meetingDays
MATH101,College Algebra,Monday|Wednesday|Friday
```

For complex arrays (arrays of objects), use indexed notation:

```
code,sections[0].crn,sections[0].capacity,sections[1].crn,sections[1].capacity
MATH101,12345,30,12346,25
```

Common Issues and Troubleshooting

Common Import Errors

Error	Possible Causes	Solution
Missing required field	Header missing or misspelled	Check header names against required fields
Invalid format	Data doesn't match expected format	Verify data types and formatting
Duplicate key	Attempting to import duplicate records	Ensure unique identifiers are truly unique
Reference not found	Referencing non-existent entity	Import referenced entities first or check references
Encoding issues	File not saved as UTF-8	Resave file with UTF-8 encoding

Validation Strategies

1. **Pre-validation**: Use spreadsheet software to check for:
   - Missing values in required fields
   - Consistent data formats
   - Proper escaping of special characters
2. **Test Imports**: Test with a small subset of data before full import
3. **Post-import Verification**: After import, verify:
   - Record counts match expected numbers
   - Sample records contain correct data
   - Relationships are properly established

Best Practices

Preparation

1. **Understand Entity Relationships**: Know how different entities relate to each other
2. **Import in the Right Order**: Import referenced entities before entities that reference them
3. **Use Templates**: Start with the provided templates and modify as needed
4. **Validate Data**: Check data for consistency and completeness before importing

Formatting

1. **Consistent Formatting**: Use consistent formatting for all fields
2. **Proper Escaping**: Properly escape special characters
3. **UTF-8 Encoding**: Always save CSV files with UTF-8 encoding
4. **Avoid BOM**: Do not include Byte Order Mark (BOM) in your CSV files

Testing and Validation

1. **Test Small Batches**: Test with small batches before importing large datasets
2. **Validate Results**: Verify imported data in the Coursedog interface
3. **Document Process**: Document your import process for future reference
4. **Backup Data**: Always backup data before importing

This guide covers the fundamental aspects of CSV formatting and import requirements for Coursedog. For specific questions or advanced use cases, please contact Coursedog support.