# INI File Cheatsheet

A practical reference covering the syntax, structure, and common conventions for working with INI configuration files.

## INI Basics & Syntax

### File Structure

| |
|---|
| INI files consist of sections and properties (key-value pairs). |
| Sections group properties logically. |
| There is typically a global section before the first named section (though not always explicitly used). |
| Order of sections and keys within sections is generally not guaranteed unless the parser explicitly supports it. |
| Example structure:<br>`; Global settings`<br>`key1 = value1`<br><br>`[SectionName1]`<br>`key2 = value2`<br>`key3 = value3`<br><br>`[SectionName2]`<br>`key4 = value4` |
| Blank lines are usually ignored and used for readability. |
| Leading/trailing whitespace around section names, keys, and values is often trimmed by parsers. |

### Sections

| | |
|---|---|
| Define a section using square brackets `[]`. | `[SectionName]` |
| Section names are typically case-insensitive, but this depends on the parser. | `[Database]`<br>`User = root`<br>`[database] ; Same section?`<br>`Password = secure`<br><br>(Depends on parser) |
| A section applies to all subsequent key-value pairs until the next section is defined. | `[Settings]`<br>`Timeout = 30`<br>`Retries = 3`<br><br>`[Logging]`<br>`Level = INFO ; Logging settings begin here` |
| The default (global) section has no header. | `AppName = MyApp`<br>`[Database]`<br>`DBName = production`<br><br>`AppName` is in the global section. |
| Avoid special characters in section names unless specifically supported by your parser. | Stick to alphanumeric characters, hyphens, and underscores. |
| Empty sections are valid but usually don't have an effect. | `[EmptySection]`<br><br>`[AnotherSection]`<br>`Key = Value` |

## Key-Value Pairs

| | |
|---|---|
| Properties are defined as `key = value`. | `key = value` |
| Whitespace around the `=` sign is typically ignored. | `key = value`<br>`key=value`<br>`key =value`<br>`key= value`<br><br>All usually parsed as `key = value`. |
| Keys within the same section should be unique. | If duplicated, the parser might use the first, the last, or report an error. |
| Key names are often case-insensitive. | `[Settings]`<br>`TIMEOUT = 60`<br>`timeout = 120 ; Might override or conflict`<br><br>(Depends on parser) |
| Values can contain various characters. Support for special characters (`=`, `;`, `#`, newline) in values varies by parser. | Quoting values (`"value"` or `'value'`) is a common convention to include special characters or spaces, but it's not part of the official INI spec and parser support varies. |
| Trailing comments on a key-value line are usually not supported in standard INI. Comments must start at the beginning of a line. | `key = value ; This comment is invalid in strict INI` |

## Comments

| | |
|---|---|
| Lines starting with `;` or `#` are treated as comments. | `; This is a comment`<br>`# This is also a comment` |
| Comments must be on their own line. Trailing comments on value lines are non-standard. | `[Section]`<br>`key = value ; Invalid comment placement` |
| Comments are ignored by the parser. | Use comments to explain sections, keys, or provide context. |
| You can comment out lines to temporarily disable them. | `#key = value` |
| Blank lines can also function as visual separators, enhancing readability, and are ignored by parsers. | `[Section1]`<br>`KeyA = ValA`<br><br>`[Section2]`<br>`KeyB = ValB` |

### Case Sensitivity & Whitespace

| | |
|---|---|
| Section names and key names are *conventionally* case-insensitive, but this is parser-dependent. | `[Settings]`<br>`TIMEOUT = 60`<br><br>`[settings]`<br>`timeout = 120 ; Potentially same key/section` |
| Values are typically case-sensitive unless the parser applies specific type-casting rules (e.g., for booleans). | `Status = Active`<br>`status = inactive ; Different values` |
| Leading and trailing whitespace around section names, keys, and values is usually trimmed. | `[ Section ]`<br>`key = value`<br><br>Often parsed as `[Section]` with `key = value`. |
| Whitespace *within* values is generally preserved, especially if the value is quoted. | `Description = This has spaces inside`<br>`Path = "C:\Program Files\MyApp"` |
| Whitespace around the `=` separator is ignored. | `Key = Value`<br>`Key=Value`<br>`Key =Value`<br>`Key= Value`<br><br>All treated equally. |

## INI Data Types & Best Practices

### Values & Data Types

INI values are inherently strings.

Parsing libraries interpret values into specific data types (integers, booleans, floats, etc.) based on conventions or explicit type hints (less common).

Common conventions for Booleans:

- `true`, `yes`, `on`, `1` -> True
- `false`, `no`, `off`, `0` -> False
  (Case-insensitivity for booleans is common).

Common conventions for Numbers:

- Integers: `123`, `-45`
- Floats: `1.23`, `-4.5e-2`
  (Parsing depends on locale and parser support).

Strings:

- Default: Raw text until comment or end of line.
- Quoted Strings (`"..."` or `'...'`): Used to include spaces or special characters. Parser support and handling of escaped quotes within strings varies greatly.

Array-like values:

- No standard way. Common workarounds:
  - Comma-separated: `list = item1, item2, item3`
  - Repeated keys (parser dependent): `item = A item = B` (usually last one wins)
  - Indexed keys (parser dependent): `item[0] = A item[1] = B`

## Quoting and Escaping

| | |
|---|---|
| Standard INI doesn't define quoting or escaping. | Parser behavior is key here. |
| Many parsers support quoting values with `"` or `'`. | `Path = "C:\Program Files\App Name"`<br>`Message = 'This is a long message with spaces.'` |
| How to include quotes within a quoted string? Parser-dependent. | • Use the other quote type: `'He said "Hello"'`<br>• Escape the quote: `"She said \"Hi\""` (backslash escaping is non-standard but common). |
| How to include `=` or `;` or `#` in a value? | • If at the start of the value, quoting might work: `Value = "=Start With Equals"`<br>• If within the value, parser must support it or it might terminate the value/start a comment. Quoting helps here too. |
| Newlines in values:<br>• Non-standard.<br>• Some parsers use escape sequences (`\n`) or allow multi-line values with indentation (rare in standard INI). | Stick to single-line values for maximum compatibility. |

## Best Practices & Tips

| |
|---|
| Keep it simple: INI is designed for simple key-value pairs and sections. Avoid complex nested structures. |
| Use consistent casing: Even if the parser is case-insensitive, use consistent casing for keys and sections (e.g., camelCase, snake_case) for readability. |
| Add comments: Explain sections, keys, and non-obvious values. Use `;` or `#` at the start of lines. |
| Use blank lines: Separate sections and groups of related keys for better readability. |
| Avoid special characters: Limit section and key names to alphanumeric, hyphens, and underscores for best compatibility. |
| Be mindful of your parser: INI parsing is not strictly standardized. Features like quoting, escaping, case-sensitivity, and array handling vary. Consult your library's documentation. |
| Prefer simple values: If you need complex data structures (lists, nested objects), consider formats like JSON, YAML, or XML. |

## Common Pitfalls

| |
|---|
| **Trailing Comments:** Placing comments after a key-value pair (`key = value ; comment`) is non-standard and likely not parsed correctly. |
| **Case Sensitivity:** Assuming case-insensitivity for keys/sections without checking the parser's behavior. |
| **Whitespace:** Relying on significant leading/trailing whitespace in keys or values without using quotes (and checking parser support for quotes). |
| **Duplicate Keys:** Defining the same key multiple times within a section can lead to unpredictable results (first wins, last wins, error). |
| **Special Characters in Values:** Using `=` or `;` or `#` at the start of a value without quoting may break parsing. |
| **Complex Data:** Trying to represent lists, dictionaries, or nested structures directly in standard INI syntax. INI is flat. |
| **Non-standard Features:** Using features like includes, variable interpolation, or specific escape sequences unless explicitly supported by the target parser. |