# User Guide: Gradle Wrapper Updater Script

## 1. Purpose

This script automates the process of updating the Gradle wrapper in multiple Gradle projects within a directory structure. It traverses a specified root folder and its subfolders. In each subfolder that contains an existing Gradle wrapper setup, it executes the `gradle wrapper --gradle-version <target_version>` command.

The script can either use a globally installed Gradle (if its version matches the target) or temporarily download and use a specific Gradle version if instructed.

## 2. Prerequisites

- A Bash shell (common on Linux and macOS; available on Windows via WSL or Git Bash).
- If the script needs to download Gradle (using the `-i` option):
  - `curl` : For downloading files.
  - `unzip` : For extracting ZIP archives.
  - `mktemp` : For creating temporary directories securely.

These tools are typically pre-installed on Linux and macOS. Windows users using Git Bash might need to ensure `curl` and `unzip` are available in their PATH.

## 3. How it Identifies Gradle Projects

The script identifies a directory as a Gradle project with an existing wrapper if all the following files are present directly within that directory:

- `gradlew` (for Unix-like systems) OR `gradlew.bat` (for Windows)
- `gradle/wrapper/gradle-wrapper.jar`
- `gradle/wrapper/gradle-wrapper.properties`

It does **not** create a new wrapper if one doesn't exist; it only updates existing ones.

## 4. Usage

```
./update_gradle_wrappers.sh [-v <version>] [-i] [<start_directory>]
```

**Arguments:**

- `<start_directory>` (optional):
  - The directory from which the script will begin its traversal.
  - If not provided, it defaults to the current directory ( `.` ).

**Options:**

- `-v <version>` , `--target-version <version>` :
  - Specifies the Gradle version to which the wrapper should be updated. This is also the version the script will try to use for executing the `wrapper` task.
  - If not provided, defaults to `8.10.2` (this default is set within the script).
- `-i` , `--internal-gradle` :
  - If this option is provided, the script will download the specified `target-version` of Gradle and use it internally for this execution. This bypasses any checks against the system's installed Gradle.
  - The downloaded Gradle is stored in a temporary directory and cleaned up when the script finishes.
- `-h` , `--help` :
  - Displays a help message with usage instructions and options, then exits.

## 5. Behavior Regarding Gradle Versions

The script's behavior for choosing which Gradle executable to run the `wrapper` task depends on the `-i` option:

- **Without `-i` (Default Behavior):**
  1. The script checks for a `gradle` command on your system's PATH.
  2. It then attempts to determine the version of this system `gradle` .
  3. **If** the system `gradle` is found **and** its version exactly matches the `target-version` (specified by `-v` or the default), the script will use this system `gradle` command.
  4. **If** the system `gradle` is not found, its version cannot be determined, or its version does **not** match the `target-version` , the script will print an error message and **exit**. The error message will guide you to either install the correct Gradle version globally or use the `-i` option.
- **With `-i` (or `--internal-gradle` ):**
  1. The script will **ignore** any `gradle` command on your system's PATH.

2. It will proceed to download the Gradle distribution corresponding to the `target-version`.
3. This downloaded Gradle will be used to execute the `wrapper` task.
4. The downloaded files are placed in a temporary directory that is automatically cleaned up when the script exits.

## 6. What the Script Does

For each directory identified as a Gradle project with an existing wrapper:

1. It changes to that project's directory.
2. It executes the command: `<gradle_executable> wrapper --gradle-version <target_version>`
    - `<gradle_executable>` is either your system `gradle` or the path to the temporarily downloaded Gradle, based on the logic described in section 5.
    - `<target_version>` is the version specified by the `-v` option or the script's default.

## 7. Output and Logging

The script provides informative output, including:

- The starting directory for traversal.
- The target Gradle wrapper version.
- Which Gradle executable is being used (system or downloaded).
- The full command being run in each project.
- A message for each directory where a Gradle wrapper is found and processed.
- Success or error messages for the `gradle wrapper` command execution in each project.
- Error messages if prerequisites are missing or if system Gradle doesn't meet requirements (when `-i` is not used).
- Information about temporary file cleanup.

## 8. Cleanup

If the script downloads a Gradle distribution (due to the `-i` option), it creates a temporary directory (usually under `/tmp` or `$TMPDIR`). This temporary directory and its contents are automatically removed when the script finishes, whether it completes successfully, exits due to an error, or is interrupted (e.g., by Ctrl+C).

## 9. Example Usages

- **Update wrappers in the current directory and subdirectories to Gradle 8.10.2 (default), requiring system Gradle to be 8.10.2:**

```
./update_gradle_wrappers.sh
```

- **Update wrappers in `~/my-projects` to Gradle 8.9, requiring system Gradle to be 8.9:**

```
./update_gradle_wrappers.sh -v 8.9 ~/my-projects
```

- **Update wrappers in the current directory to Gradle 8.8, forcing a temporary download of Gradle 8.8:**

```
./update_gradle_wrappers.sh -i -v 8.8
```

or

```
./update_gradle_wrappers.sh --internal-gradle --target-version 8.8
```

- **Update wrappers in `./specific-group/java-apps` to Gradle 8.7, forcing download and use of Gradle 8.7:**

```
./update_gradle_wrappers.sh -i -v 8.7 ./specific-group/java-apps
```