dotnet list package

Article • 02/25/2023

This article applies to: ✓ .NET Core 3.1 SDK and later versions

Name

`dotnet list package` - Lists the package references for a project or solution.

Synopsis

.NET CLI

```
dotnet list [<PROJECT>|<SOLUTION>] package [--config <SOURCE>]
    [--deprecated]
    [-f|--framework <FRAMEWORK>] [--highest-minor] [--highest-patch]
    [--include-prerelease] [--include-transitive] [--interactive]
    [--outdated] [--source <SOURCE>] [-v|--verbosity <LEVEL>]
    [--vulnerable]
    [--format <console|json>]
    [--output-version <VERSION>]

dotnet list package -h|--help
```

Description

The `dotnet list package` command provides a convenient option to list all NuGet package references for a specific project or a solution. You first need to build the project in order to have the assets needed for this command to process. The following example shows the output of the `dotnet list package` command for the SentimentAnalysis project:

Output

```
Project 'SentimentAnalysis' has the following package references
   [netcoreapp2.1]:
   Top-level Package               Requested   Resolved
   > Microsoft.ML                  1.4.0       1.4.0
   > Microsoft.NETCore.App   (A)   [2.1.0, )   2.1.0

(A) : Auto-referenced package.
```

The **Requested** column refers to the package version specified in the project file and can be a range. The **Resolved** column lists the version that the project is currently using and is always a single value. The packages displaying an `(A)` right next to their names represent implicit package references that are inferred from your project settings (`Sdk` type, or `<TargetFramework>` or `<TargetFrameworks>` property).

Use the `--outdated` option to find out if there are newer versions available of the packages you're using in your projects. By default, `--outdated` lists the latest stable packages unless the resolved version is also a prerelease version. To include prerelease versions when listing newer versions, also specify the `--include-prerelease` option. To update a package to the latest version, use dotnet add package.

The following example shows the output of the `dotnet list package --outdated --include-prerelease` command for the same project as the previous example:

Output

```
The following sources were used:
   https://api.nuget.org/v3/index.json
   C:\Program Files (x86)\Microsoft SDKs\NuGetPackages\

Project `SentimentAnalysis` has the following updates to its packages
   [netcoreapp2.1]:
   Top-level Package      Requested   Resolved   Latest
   > Microsoft.ML         1.4.0       1.4.0      1.5.0-preview
```

If you need to find out whether your project has transitive dependencies, use the `--include-transitive` option. Transitive dependencies occur when you add a package to your project that in turn relies on another package. The following example shows the output from running the `dotnet list package --include-transitive` command for the HelloPlugin project, which displays top-level packages and the packages they depend on:

Output

```
Project 'HelloPlugin' has the following package references
   [netcoreapp3.0]:
   Transitive Package      Resolved
   > PluginBase            1.0.0
```

Arguments

`PROJECT | SOLUTION`

The project or solution file to operate on. If not specified, the command searches the current directory for one. If more than one solution or project is found, an error is thrown.

Options

- **`--config <SOURCE>`**

  The NuGet sources to use when searching for newer packages. Requires the `--outdated` option.

- **`--deprecated`**

  Displays packages that have been deprecated.

- **`-f|--framework <FRAMEWORK>`**

  Displays only the packages applicable for the specified target framework. To specify multiple frameworks, repeat the option multiple times. For example: `--framework net6.0 --framework netstandard2.0`. The short form of the option (`-f`) is available starting in .NET 9 SDK.

- **`-?|-h|--help`**

  Prints out a description of how to use the command.

- **`--highest-minor`**

  Considers only the packages with a matching major version number when searching for newer packages. Requires the `--outdated` or `--deprecated` option.

- **`--highest-patch`**

  Considers only the packages with a matching major and minor version numbers when searching for newer packages. Requires the `--outdated` or `--deprecated` option.

- **`--include-prerelease`**

  Considers packages with prerelease versions when searching for newer packages. Requires the `--outdated` or `--deprecated` option.

- **`--include-transitive`**

Lists transitive packages, in addition to the top-level packages. When specifying this option, you get a list of packages that the top-level packages depend on.

- **`--interactive`**

  Allows the command to stop and wait for user input or action. For example, to complete authentication. Available since .NET Core 3.0 SDK.

- **`--outdated`**

  Lists packages that have newer versions available.

- **`-s|--source <SOURCE>`**

  The NuGet sources to use when searching for newer packages. Requires the `--outdated` or `--deprecated` option.

- **`-v|--verbosity <LEVEL>`**

  Sets the verbosity level of the command. Allowed values are `q[uiet]`, `m[inimal]`, `n[ormal]`, `d[etailed]`, and `diag[nostic]`. The default is `minimal`. For more information, see LoggerVerbosity.

- **`--vulnerable`**

  Lists packages that have known vulnerabilities. Cannot be combined with `--deprecated` or `--outdated` options. Nuget.org is the source of information about vulnerabilities. For more information, see Vulnerabilities and How to Scan NuGet Packages for Security Vulnerabilities .

- **`--format <console|json>`**

  Sets the report output format. Allowed values are `console`, `json`. Defaults to `console`. Available starting in .NET SDK 7.0.200.

- **`--output-version <VERSION>`**

  Sets the report output version. Allowed value is `1`. Defaults to `1`. Requires the `--format json` option. When a new JSON version is available, the command will produce the new format by default. This option will let you specify that the command should produce an earlier format. Available starting in .NET SDK 7.0.200.

Examples

- List package references of a specific project:

  .NET CLI

  ```
  dotnet list SentimentAnalysis.csproj package
  ```

- List package references that have newer versions available, including prerelease versions:

  .NET CLI

  ```
  dotnet list package --outdated --include-prerelease
  ```

- List package references for a specific target framework:

  .NET CLI

  ```
  dotnet list package --framework netcoreapp3.0
  ```

- List package references in machine readable json output format:

  .NET CLI

  ```
  dotnet list package --format json
  ```

- List package references for a specific target framework in machine readable json output format:

  .NET CLI

  ```
  dotnet list package --framework netcoreapp3.0 --format json
  ```

- Save machine readable json output of package references, including transitive dependency and vulnerability details into a file:

  .NET CLI

  ```
  dotnet list package --include-transitive --vulnerable --format
  json >> dependencyReport.json
  ```

- List package references in machine readable json output format with output version 1:

.NET CLI

```
dotnet list package --format json --output-version 1
```