# Accurate Player 3 Monorepo

This monorepo contains all packages related to Accurate Player 3. Packages are located in the `packages` folder:

- `core` - Core player functionality used by all players
- `progressive` - Player that plays videos using [progressive download](https://en.wikipedia.org/wiki/Progressive_download)
- `hls` - Player that plays videos using [HTTP Live Streaming](https://en.wikipedia.org/wiki/HTTP_Live_Streaming)
- `dash` - Player that plays videos using [Dynamic Adaptive Streaming over HTTP](https://en.wikipedia.org/wiki/Dynamic_Adaptive_Streaming_over_HTTP)
- `abr` - Player that plays both HLS and DASH videos
- `cutlist` - Player that plays videos from a cut list using progressive download
- `plugins` - Player plugins
- `timecode` - Utilities for parsing/manipulating video timecodes
- `license-client` - License validation client
- `demo` - Player and plugin demos

`core` contains the basic player logic, interfaces and playback controllers.
It relies heavily on logic from `timecode`.

`core` itself doesn't contain much logic, playback is handled by playback controllers and `plugins` empower the player.

## Commands

- `yarn` - Installs dependencies.
- `yarn build` - Builds all packages. Production build. Minified source code and no source maps.
- `yarn build:dev` - Builds all packages. Development build. No minification of source code and includes source maps.
- `yarn build:watch` - Builds all packages but `demo` in watch mode (rebuilds on file changes). Development build.
- `yarn build:docs` - Builds documentation.
- `yarn build:demo` - Builds demos.
- `yarn test` - Runs all tests
- `yarn test:coverage` - Runs all tests and generate [istanbul](https://istanbul.js.org/) coverage reports.
- `yarn lint` - Checks all source for lint errors
- `yarn lint:fix` - Checks all source for lint errors and tries to fix all errors that can automatically be fixed.
- `yarn start` - Starts builds in watch mode in all packages and starts the `demo` -package on http://localhost:5000
- `yarn clean` - Clears all build folders and removes all installed packages.

See complete list of commands in `package.json`

Note IDE will not find related packages before they've been built.

# Development

To start only a subset of the packages with `yarn start` (or any other command) you can leverage `lerna`s
filter options, full documentation [here](https://github.com/lerna/lerna/tree/main/core/filter-options).

Some examples:

Start plugins and the demo package:

```
yarn start --scope "*/*plugins" --scope "*/*demo"
```

Start the progressive player package with all it's dependencies:

```
yarn start --scope "*/*progressive" --include-dependencies
```

This feature is useful to not start every package during development to speed things up.

JIT Demo

There are some environment variables used for the Jit demo

- `JIT_BACKEND`: Specify backend for Jit, can either be a complete url (eg `http://127.0.0.1:8080`) or
  or just an ip (eg `127.0.0.1`), in which case `http://` and `:8080` is added automatically.
- `AP_KEY`: Specify ap license key to use

Example command to start: `env JIT_BACKEND=127.0.01 AP_KEY=ABCD yarn start`

Test without publishing

We can test changes locally without the need of publishing changes to nexus npm repository.

Using npm pack

Run `npm pack` inside the appropriate package (e.g. packages/core) to create a local archive `codemill-accurate-player-core-x.y.z.tgz`.
Install it where needed by running `npm install path/to/*.tgz`.

Release

Test release

To create a prerelease unique to your current branch, run:

```
yarn release:pre
```

Normal release

Release using
`yarn increment -- pre{major|minor|patch}`
or just
`yarn increment`
selecting version manually

See RELEASE.md