Migrating to Meteor 2.3

Most of the new features in Meteor 2.3 are either applied directly behind the scenes (in a backwards compatible manner) or are opt-in. For a complete breakdown of the changes, please refer to the changelog.

The above being said, there are a few breaking changes that you might need to apply migration for.

Node.js v14

As Node.js version was upgraded to a new major version we recommend that you review if your npm dependencies are compatible with Node.js 14.

- If we receive reports from breaking changes we are going to list them here but so far we are not aware of any.
- We recommend that you read Node.js release notes though.
- We recommend that you remove your `node_modules` folder (`rm -rf node_modules`) and run `meteor npm i` to be sure you compile all the binary dependencies again using the new Node.js version.
- Maybe you also want to recreate your lock file.
- If you get an error try `meteor reset` which will clear caches, beware that this will also remove your local DB for your app.

`deprecated` option for packages

In `Package.description`, there is a new `deprecated` option. If set to `true` it will inform user when installing that the package has been deprecated. Additionally you can provide a string that will be displayed, where you can direct the users where to go next.

All official packages that have been deprecated have now the deprecated flag and will inform you about that if you install or update them.

Removal of deprecated package API

Old API for packages definitions has been removed. The old underscore method names (`Package.on_use`, `Package.on_test`, `Package._transitional_registerBuildPlugin` and `api.add_files`) have been removed and will no longer work, please use the camel case method names (e.g. `api.addFiles()`).

Accounts 2.0

- `accounts-base@2.0.0`
  - Deprecated backward compatibility function `logoutOtherClients` has been removed.
- `accounts-password@2.0.0`
  - Deprecated backward compatibility functionality for `SRP` passwords from pre-Meteor 1.0 days has been removed.
  - Enroll account workflow has been separated from reset password workflow (the enrollment token records are now stored in a separate db field `services.password.enroll`).
- `oauth@2.0.0`
  - Removed deprecated `OAuth.initiateLogin` and other functionality like the addition of `?close` in return URI for deprecated OAuth flow pre Meteor 1.0

If you are maintaining a package that depends on one of the accounts packages which had a major version bump you will either need to set the new version manually or set `api.versionsFrom('2.3')`. You can also have it reference its current version and 2.3 like this: `api.versionsFrom(['1.12', '2.3'])`, for specific package it can be like this: `api.use('accounts-base@1.0.1 || 2.0.0')`.

HTTP v2

Internally `http` package has replaced the use of http for fetch, should still work as previous version, but edge cases might be different. This is to aid you in transition to fetch. Note that this means that the `npmRequestOptions` parameter to `HTTP.call` has been removed, as `request` is no longer used internally. You should migrate to the use of `fetch`. You can install polyfill package via:

```
meteor add fetch
```

Removal of deprecated APIs

In addition to the above mentioned removal of deprecated package API, other long deprecated APIs have been removed and will no longer work.

- Removed deprecated `mobile-port` flag
- Removed deprecated `raw` name from `isobuild`
- `ddp-client@2.5.0`
  - Removed deprecated backward compatibility method names for Meteor before 1.0
- `ddp-server@2.4.0`
  - Removed deprecated backward compatibility method names for Meteor before 1.0
- `meteor-base@1.5.0`
  - Removed `livedata` dependency which was there for packages build for 0.9.0
- `minimongo@1.7.0`

  - Removed the `rewind` method that was noop for compatibility with Meteor 0.8.1
- `mongo@1.12.0`
  - Removed the `rewind` method that was noop for compatibility with Meteor 0.8.1
- `socket-stream-client@0.4.0`
  - Remove IE8 checks

Migrating from a version older than 2.2?

If you're migrating from a version of Meteor older than Meteor 2.2, there may be important considerations not listed in this guide (which specifically covers 2.2 to 2.3). Please review the older migration guides for details:

- Migrating to Meteor 2.2 (from 2.0)
- Migrating to Meteor 2.0 (from 1.12)
- Migrating to Meteor 1.12 (from 1.11)
- Migrating to Meteor 1.11 (from 1.10.2)
- Migrating to Meteor 1.10.2 (from 1.10)
- Migrating to Meteor 1.10 (from 1.9.3)
- Migrating to Meteor 1.9.3 (from 1.9)
- Migrating to Meteor 1.9 (from 1.8.3)
- Migrating to Meteor 1.8.3 (from 1.8.2)
- Migrating to Meteor 1.8.2 (from 1.8)
- Migrating to Meteor 1.8 (from 1.7)
- Migrating to Meteor 1.7 (from 1.6)
- Migrating to Meteor 1.6 (from 1.5)
- Migrating to Meteor 1.5 (from 1.4)
- Migrating to Meteor 1.4 (from 1.3)
- Migrating to Meteor 1.3 (from 1.2)