Migrating legacy localization IDs

Why this migration is necessary

The Angular translation system works by matching a message ID to a translated message in a different language. One option for creating these IDs is to have the translation system generate them for you. Previously, the format for these IDs are not stable when there are insignificant changes, such as to whitespace. The new format is much more robust.

This topic describes how to migrate old localization IDs to help you future-proof your application in the event that the old format is removed.

What this migration does

Angular version 11 introduced a new format for generating localization IDs. These new IDs are more robust than the previous legacy format. However, applications created before version 11 still used the legacy format for their IDs.

You do not need to migrate custom localization IDs to new IDs.

With the release of Angular version 12, you now have how tools available to help you migrate any legacy localization IDs to IDs that use the latest algorithms.

You have two options for migrating legacy IDs. The first method uses the Angular CLI to locate legacy IDs in your application. The second method uses a standalone script, `localize-extract`, to locate the legacy IDs.

Migrating legacy IDs using the CLI

To migrate legacy localization IDs using the CLI:

1. Run the `ng extract-i18n` command.

   ng extract-i18n --format=legacy-migrate

   After running this command, you have a migration file, `messages.json`, containing a mapping between legacy localization IDs and new IDs that you can use to update your application.

2. Update the IDs in your application using the `npx localize-migrate` command.

   npx localize-migrate --files=*.xlf --mapFile=messages.json

   You can also specify other formats in the `files` parameter, such as `*.xmb`.

3. Commit the updated files to your source control system.

After you complete the migration, set the Angular Compiler option, `enableI18nLegacyMessageIdFormat`, to `false`. For more information about this option, see Angular Compiler Options.

Migrate legacy IDs using `localize-extract`

If you are not using the Angular CLI, you can migrate legacy localization IDs using `localize-extract`:

1. Run the `npx localize-extract` command.

   npx localize-extract --format=legacy-migrate --source=./path/to/bundles/**/*.js --outputPath=./messages.json

In this command, `./path/to/bundles/` represents the path to your distributable files. You can set the `outputPath` parameter to any directory in your system.

Your distributable files must not have been translated, such as by using `localize-translate`, as doing so strips the `$localize` tagged strings that the extractor requires.

After running this command, you have a migration file, `messages.json`, containing a mapping between legacy localization IDs and new IDs that you can use to update your application.

2. Update the IDs in your application using the `npx localize-migrate` command.

   npx localize-migrate --files=*.xlf --mapFile=messages.json
   You can also specify other formats in the `files` parameter, such as `*.xmb`.

After you complete the migration, set the Angular Compiler option, `enableI18nLegacyMessageIdFormat`, to `false`. For more information about this option, see Angular Compiler Options.

{@searchKeywords i18n}