## 1 Introduction

This is a specification for the package-local nicknames extension in Common Lisp.

### 1.1 Rationale

Package-local nicknames allow to use short and easy-to-use names for packages locally without potentially introducing name conflict as with normal nicknames.

### 1.2 Current state

Package-local nicknames are implemented in some form in `SBCL`, `CCL`, `ECL`, `Clasp`, `ABCL`, `Allegro CL`, `LispWorks`. There also exists a pending PR for the `CLISP` implementation.

Unfortunately, there are multiple inconsistencies between implementations, all implementations lose the **print-read** consistency to some extent, and there are multiple edge cases that aren't always implemented correctly.

### 1.3 Goal

The purpose of this document is to standardize the package-local nicknames extension and to address some existing issues.

[TODO] This CDR also aims to provide an extensive test suite for this extension.

## 2 Description

*Package-local nickname* (or *local nickname*) has the same effects as a usual *package nickname* (later *global nickname*), except that these effects only apply when `*package*` is bound to a package for which the nickname has been defined.

That means that calls to `find-package` with a *local nickname* defined in the *current package* should return the package nicknamed by this nickname.

This also affects all implied calls to `find-package`, including those performed by the lisp reader.

In addition, to maintain **print-read** consistency, the lisp printer is affected by *local nicknames* defined in the *current package*. For details see Issue 2.

*Local nickname* is allowed to shadow a *package name* or a *global nickname*, except for the names `#:CL`, `#:COMMON-LISP` and `#:KEYWORD` which should always refer to their packages.

The consequences of adding *local nicknames* to the packages `#:COMMON-LISP` and `#:KEYWORD` are undefined.

## 3 API

### 3.1 defpackage

`defpackage` options are extended to include *local-nicknames-option*:

```
local-nicknames-option ::= (:local-nicknames (nickname package)*)
```

Each pair specifies a *local nickname* `nickname` for the corresponding `package`.

This option may appear more than once.

### 3.1.1 Arguments and Values:

`nickname` must be a *string designator*.

`package` must be a *package designator*.

### 3.1.2 Exceptional situations

An error of type `package-error` is signaled when a package designated by `package` does not exists.

Name conflict errors are handled by the underlying calls to `add-package-local-nickname`.

See add-package-local-nickname: exceptional situations.

### 3.1.3 Implementation dependent

The behaviour is unspecified when a *local nickname* is specified for the package that is being defined. (See Issue 4.)

The behaviour is unspecified when supplied *local nicknames* are at variance with the current state of the package that is being defined. An implementation might choose to remove all present *local nicknames* at the begining of each redefinition of the package.

## 3.2 make-package

(**PROPOSAL**: see Issue 6.)

`make-package` lambda list is extended to include an additional keyword argument `:local-nicknames`:

```
local-nicknames ::= ((nickname package)*)
```

`local-nicknames` defaults to an *empty list*.

`local-nicknames` must be a *list* each element of which must be a *list* of form `(nickname package)`. Specifies *local nicknames* in the new *package*.

### 3.2.1 Arguments and Values:

`local-nicknames` must be a a *list* of pairs `(nickname package)`.

`nickname` must be a *string designator*.

`package` must be a *package designator*.

### 3.2.2 Exceptional situations

An error of type `package-error` is signaled when a package designated by `package` does not exists.

Name conflict errors are handled by the underlying calls to `add-package-local-nickname`.

See add-package-local-nickname: exceptional situations.

### 3.2.3 Implementation dependent

The behaviour is unspecified when a *local nickname* is specified for the package that is being defined. (See Issue 4.)

## 3.3 add-package-local-nickname

```
(add-package-local-nickname nickname actual-package &optional designated-package)
  => designated-package-object
```

`designated-package` defaults to the *current package*.
Adds a *package-local nickname* `nickname` for the `actual-package` in the `designated-package`.
Returns the package designated by `designated-package`. (But also see Issue 1.)
If a *nickname* is already defined, checks that it is defined for the package designated by `actual-package`.

### 3.3.1 Arguments and Values

`nickname` must be a *string designator*.
`actual-package` and `designated-package` must be *package designators*.
`designated-package-object` is of type *package*.

### 3.3.2 Exceptional situations

If a package designated by `actual-package` or a package designated by `designated-package` does not exists, an error of type *package-error* must be signaled.

If `nickname` is one of the names `#:CL`, `#:COMMON-LISP` or `#:KEYWORD`, an error of type *package-error* must be signaled.

If `nickname` is a *local nickname* for a package different from `actual-package`, an error of type *package-error* must be signaled.

### 3.3.3 Implementation dependent

The consequences are undefined when `designated-package` designates one of the packages `#:COMMON-LISP` or `#:KEYWORD`.

(**PROPOSAL**: see Issue 5.)

If `nickname` shadows the `designated-package`'s *package name* or one of its *global nicknames*, a style warning might signaled.

## 3.4 remove-package-local-nickname

```
(remove-package-local-nickname old-nickname &optional designated-package)
  => nickname-removed-p
```

`designated-package` defaults to the *current package*.
If `designated-package` has `old-nickname` as a *local nickname*, it is removed.
Returns *true* if the `old-nickname` existed and was removed, and `NIL` otherwise. (But also see Issue 1.)

### 3.4.1 Arguments and Values

`old-nickname` must be a *string designator*.
`designated-package` must be a *package designator*.
`nickname-removed-p` is a *generalized boolean*.

### 3.4.2 Exceptional situations

If a package designated by `designated-package` does not exists, an error of type *package-error* must be signaled.

## 3.5 package-local-nicknames

```
(package-local-nicknames package)
  => local-nicknames-alist
local-nicknames-alist ::= ((nickname . package)*)
```

Returns an *alist* describing *local nicknames* defined in the package designated by `package`.

### 3.5.1 Arguments and Values

`package` must be a *package designator*.

`local-nicknames-alist` is an *alist* with keys of type *string* and values of type *package*.

`nickname` must be a *string*.

`package` must be a *package*.

### 3.5.2 Exceptional situations

An error of type `package-error` is signaled when a package designated by `package` does not exists.

### 3.5.3 Notes

The returned *alist* must be safe to be modified by the user.

## 3.6 package-locally-nicknamed-by-list

```
(package-locally-nicknamed-by-list package)
  => packages-list
```

Returns a *list* of packages that have a *local nickname* defined for the package designated by `package`.

### 3.6.1 Arguments and Values

`package` must be a *package designator*.

`packages-list` is a *list* with elements of type *package*.

### 3.6.2 Exceptional situations

An error of type `package-error` is signaled when a package designated by `package` does not exists.

### 3.6.3 Notes

The returned *list* must be safe to be modified by the user.

# 4 Affected symbols

## 4.1 defpackage

See defpackage.

## 4.2 make-package

See make-package.

## 4.3 find-package

When argument to `find-package` is a *local nickname* that is defined in the *current package*, it returns the package named by this nickname.

This also affects all implied calls to `find-package`, including but not limited to those performed by the lisp reader as well as those performed by `export`, `find-symbol`, `import`, `rename-package`, `shadow`, `shadowing-import`, `delete-package`, `with-package-iterator`, `unexport`, `unintern`, `in-package`, `unuse-package`, `use-package`, `do-symbols`, `do-external-symbols`, `do-all-symbols`, `intern`, `package-name`, `package-nicknames`, `package-shadowing-symbols`, `package-use-list`, `package-used-by-list`.

`add-package-local-nickname`, `remove-package-local-nickname`, `package-local-nicknames` and `package-locally-nicknamed-by` are also affected.

(**PROPOSAL**: see Issue 8.)

The only exception is the `format`'s *tilde slash* directive, which should **not** use *local nicknames* of any package when looking up the symbol specified.

## 4.4 rename-package

When a package is renamed with `rename-package` it maintains all *local nicknames* it is nicknamed by, as well as all *local nicknames* it has defined.

### 4.4.1 Implementation dependent

(**PROPOSAL**: see Issue 5.)

If the *new-name* or one of the *new-nicknames* is shadowed by one of the *local nicknames* of the package being redefined, a warning might be signaled.

## 4.5 delete-package

When a package is deleted with `delete-package` all *local nicknames* defined in other packages that it was nicknamed by must be removed, as well as all *local nicknames* defined in the package that is being deleted.

This also means that a deleted package must not be available via calls to `package-locally-nicknamed-by-list` and `package-local-nicknames`.

## 4.6 format

See Issue 8.

### 4.7 *features*

If an implementation supports package-local nicknames it should add symbols `:package-local-nicknames` and `:cdr-15` (per CDR 14) to `*features*`.

## 5 Examples

[TODO]