

# ***uffi* Reference Guide**

**Kevin M. Rosenberg**  
Heart Hospital of New Mexico

kevin@rosenberg.net  
504 Elm Street N.E.  
Albuquerque  
New Mexico  
87102



## ***uffi* Reference Guide**

by Kevin M. Rosenberg

Copyright © 2002 by Kevin M. Rosenberg

- The *uffi* package was designed and written by Kevin M. Rosenberg.
- Allegro CL® is a registered trademark of Franz Inc.
- Lispworks® is a registered trademark of Xanalys Inc.
- Microsoft Windows® is a registered trademark of Microsoft Inc.
- Other brand or product names are the registered trademarks or trademarks of their respective holders.



# Table of Contents

<b>1. Introduction.....</b>	<b>7</b>
Purpose.....	7
Background .....	7
Supported Implementations .....	7
Installation.....	7
<b>2. Programming Reference .....</b>	<b>9</b>
Design Overview.....	9
Declarations .....	9
Overview .....	9
uffi-declare.....	9
slot-type .....	9
Immediate Types .....	10
def-constant .....	10
def-type.....	10
null-char-p .....	10
Aggregate Types.....	10
def-enum.....	10
def-struct.....	10
get-slot-value .....	10
get-slot-pointer .....	11
def-array .....	11
deref-array .....	11
Objects .....	11
allocate-foreign-object.....	11
free-foreign-object.....	11
pointer-address.....	11
deref-pointer .....	11
make-null-pointer .....	12
null-pointer-p.....	12
+null-c-string-ptr+ .....	12
Strings .....	12
convert-from-c-string.....	12
convert-to-c-string .....	12
free-c-string .....	12
with-c-string .....	12
convert-from-foreign-string.....	13
convert-to-foreign-string.....	13
allocate-foreign-string .....	13

Routine .....	13
def-function .....	13
Libraries .....	13
load-foreign-library .....	13

# Chapter 1. Introduction

## Purpose

This reference guide describes *uffi*, a Lisp package that provides persistent cross-implementation support of C-language compatible libraries.

## Background

Every Common Lisp implementation has a method for interfacing to C-language compatible libraries. Unfortunately, these methods vary widely amongst implementations. Currently, to support multiple implementations, developers must write a different interface library for each Common Lisp implementation.

*uffi* gathers a common subset of functionality between Common Lisp implementations. *uffi* wraps this common subset of functionality with its own syntax and provides macro translation of *uffi* functions into the specific syntax of supported Common Lisp implementations.

Developers who use *uffi* to interface with C libraries will automatically have their code function in each of *uffi*'s supported implementations.

## Supported Implementations

The primary tested and supported platforms for *uffi* are:

- AllegroCL v6.1 on Redhat Linux 7.2 and Microsoft Windows.
- Lispworks v4.2 on Redhat Linux 7.2 and Microsoft Windows.
- CMUCL 18c on Redhat Linux 7.2.

## Installation

Installation is fairly simple. The main requirement is that you have a copy of *defsystem*. You can download the latest version of *defsystem* from the *CLOCC* (<http://www.sourceforge.net/projects/clocc>) CVS tree. After installing *defsystem*, simply *push* the directory containing *uffi* into

mk:\*central-registry\*. Whenever you want to load the *uffi* package, use the function `(mk:os :uffi 'load)`.



# Chapter 2. Programming Reference

## Design Overview

*uffi* was designed as a cross-implementation compatible *Foreign Function Interface*. Necessarily, only a common subset of functionality can be provided. Likewise, not every optimization for that a specific implementation provides can be supported. Wherever possible, though, implementation-specific optimizations are invoked.

## Declarations

### Overview

Declarations are used to give the compiler optimizing information about foreign types. Currently, only CMUCL supports declarations. On AllegroCL and Lispworks, these expressions declare the type generically as `T`

### uffi-declare

This is used wherever a `declare` expression can be placed. For example:

```
(let ((my-structure (uffi:allocate-foreign-object 'a-struct)))
  (uffi:uffi-declare a-struct my-structure))
```

### slot-type

This is used inside of `defclass` and `defstruct` expressions to set the type for a field. Because the type identifier is not evaluated in ANSI Common Lisp, the expression must be backquoted for effect. For example:

```
(eval
```

```
`(defclass a-class ()  
  ((char-ptr :type ,(uffi:slot-type (* :char))))))
```

## Immediate Types

### **def-constant**

This is a thin wrapper around `defconstant`. It also exports the symbol from the package.

### **def-type**

This is the main function for creating new types.

### **null-char-p**

A predicate testing if a pointer object is `NULL`

## Aggregate Types

### **def-enum**

Declares a C enumeration. It generates constants for the elements of the enumeration.

### **def-struct**

Declares a structure.

## **get-slot-value**

Accesses a slot value from a structure.

## **get-slot-pointer**

This is similar to `get-slot-value`. It is used when the value of a slot is a pointer type.

## **def-array**

Defines an array.

## **deref-array**

Accesses an element of an array.

# **Objects**

## **allocate-foreign-object**

Allocates an instance of a foreign object.

## **free-foreign-object**

Frees the memory used by a foreign object.

## **pointer-address**

Returns the address as an integer of a pointer.

## **deref-pointer**

Returns the object to which a pointer points.

## **make-null-pointer**

Creates a `NULL` pointer of a specified type.

## **null-pointer-p**

A predicate testing if a pointer is has a `NULL` value.

## **+null-c-string-ptr+**

A constant returning a `NULL` character pointer;

# **Strings**

## **convert-from-c-string**

Converts a Lisp string to a `c-string`.

## **convert-to-c-string**

Converts a Lisp string to a `c-string`. These `c-string`'s should be freed with `free-c-string`.

## **free-c-string**

Frees any memory possibly allocated by `convert-to-c-string`.

## **with-c-string**

Binds a lexical variable to a newly allocated `c-string`. Automatically frees `c-string`.

## **covert-from-foreign-string**

Returns a Lisp string from a foreign string. Has parameters to handle ASCII versus binary strings.

## **convert-to-foreign-string**

Converts a Lisp string to a foreign string. Memory should be freed with `free-foreign-object`.

## **allocate-foreign-string**

Allocates space for a foreign string. Memory should be freed with `free-foreign-object`.

## **Routine**

### **def-function**

This macro generates a C routine definition.

## **Libraries**

### **load-foreign-library**

This function loads foreign libraries. It has checks to ensure that a library is loaded only once during a session.

