# Common Lisp Object System Specification

# 3. Functions in the Meta-Object Protocol

This document was written by Daniel G. Bobrow, Linda G. DeMichiel, Richard P. Gabriel, Sonya E. Keene, Gregor Kiczales, and David A. Moon.

Contributors to this document include Patrick Dussud, Kenneth Kahn, Jim Kempf, Larry Masinter, Mark Stefik, Daniel L. Weinreb, and Jon L White.

## CONTENTS

# Introduction

This chapter describes some of the functions provided by the Common Lisp Object System Meta-Object Protocol. This document is preliminary; it is not to be construed as presenting a complete description of Meta-Object Protocol.

The description of each function includes its purpose, its syntax, the semantics of its arguments and returned values, and often an example and cross-references to related functions. This chapter is reference material that requires an understanding of the basic concepts of the Common Lisp Object System. The functions are arranged in alphabetic order for convenient reference.

# allocate-instance *Standard Generic Function*

## Purpose:

The generic function **allocate-instance** allocates storage for an instance of the given class.

## Syntax:

**allocate-instance** *class* `&key &allow-other-keys` [*Generic Function*]

## Method Signatures:

**allocate-instance** (*class* `standard-class`) `&key &allow-other-keys` [*Primary Method*]

## Arguments:

The *class* argument is a class object.

## Values:

The result is the newly-allocated instance.

## Remarks:

The keyword name of each keyword parameter specifier in the lambda-list of any method defined for **allocate-instance** becomes an initialization argument for all classes for which that method is applicable.

Programmers can customize the behavior of **allocate-instance** by defining additional methods for it or replacing the system-supplied method on **standard-class**.

## See Also:

"Object Creation and Initialization"

# check-initargs *Standard Generic Function*

## Purpose:

The generic function **check-initargs** is called by **make-instance** to test the validity of initialization arguments.

The function **check-initargs** implements the rules for validity described in the section "Object Creation and Initialization."

## Syntax:

**check-initargs** *class initarg-list* [*Generic Function*]

## Method Signatures:

**check-initargs** (*class* `standard-class`) *initarg-list* [*Primary Method*]

## Arguments:

The *class* argument is a class object.

The *initarg-list* argument is an initialization argument list.

## Values:

The value returned by **check-initargs** is ignored by **make-instance**.

## Remarks:

Programmers can customize the behavior of **check-initargs** by defining additional methods for it or by replacing the system-supplied method on **standard-class**.

## See Also:

"Object Creation and Initialization"

# class-all-initargs *Standard Generic Function*

## Purpose:

The generic function **class-all-initargs** returns a list of the names of all initialization arguments that apply to a given class, including the names of inherited initialization arguments.

This result is the union of the following two sets of the initialization argument names: the initialization argument names contained in the result of `(class-all-slot-initargs` *class*`)` and the initialization argument names that are defined by all the applicable methods for **allocate-instance** and **initialize-instance**.

## Syntax:

**class-all-initargs** *class* [*Generic Function*]

## Method Signatures:

**class-all-initargs** (*class* `standard-class`) [*Primary Method*]

## Arguments:

The *class* argument is a class object.

## Values:

The result is a list of the names of all initargs that apply to a given class, including the names of inherited initargs.

## Remarks:

The results are undefined if the value returned by this function is modified.

It is permitted, but not required, for an implementation to return values that share with internal data structures.

## See Also:

"Object Creation and Initialization"

**make-instance**

**initialize-instance**

# class-all-initarg-defaults *Standard Generic Function*

## Purpose:

The generic function **class-all-initarg-defaults** returns a list, each of whose elements is a list consisting of the name of an initialization argument defined for a slot in that class or any of its superclasses followed by the default value function and the default initial value form for that initialization argument. The default value function is the function whose body is the default initial value form; this function is created in the lexical environment of the **defclass** form that contains the default initial value form. The purpose of this function is to capture the environment.

## Syntax:

**class-all-initarg-defaults** *class* [*Generic Function*]

## Method Signatures:

**class-all-initarg-defaults** (*class* `standard-class`) [*Primary Method*]

## Arguments:

The *class* argument is a class object.

## Values:

The result is a list of the form

((*initialization-argument-name default-value-function default-initial-value-form*) ... ).

The default initial value form is the form that was originally specified. It is retained only for documentation. The results are undefined if the default initial value form is evaluated.

The default value function is what actually gets called; its effect is equivalent to enclosing the default initial value form in the appropriate lexical environment. The default value function takes no arguments.

## Remarks:

The results are undefined if the value returned by this function is modified.

It is permitted, but not required, for an implementation to return values that share with internal data structures.

# class-all-slot-initargs *Standard Generic Function*

## Purpose:

The generic function **class-all-slot-initargs** returns a list, each of whose elements is a list consisting of the name of an initialization argument defined for a slot in that class or any of its superclasses followed by the names of all slots defined in the **defclass** forms for that class or any of its superclasses for which that initialization argument is defined.

## Syntax:

**class-all-slot-initargs** *class* [*Generic Function*]

## Method Signatures:

**class-all-slot-initargs** (*class* `standard-class`) [*Primary Method*]

## Arguments:

The *class* argument is a class object.

## Values:

The result is a list of the form `((`*initialization-argument-name slot-name* `...) ... )`.

## Remarks:

The results are undefined if the value returned by this function is modified.

It is permitted, but not required, for an implementation to return values that share with internal data structures.

# class-direct-initargs *Standard Generic Function*

## Purpose:

The generic function **class-direct-initargs** returns a list of the names of all initialization arguments corresponding to a given class. These initialization arguments include those that were specified using the **:initarg** slot option in the **defclass form** for the given class, as well as those that were defined in the lambda-lists for methods on **allocate-instance** and **initialize-instance** that are defined for the exact given class.

## Syntax:

**class-direct-initargs** *class* [*Generic Function*]

## Method Signatures:

**class-direct-initargs** (*class* `standard-class`) [*Primary Method*]

## Arguments:

The *class* argument is a class object.

## Values:

The result is a list of initialization argument names.

## Remarks:

The results are undefined if the value returned by this function is modified.

It is permitted, but not required, for an implementation to return values that share with internal data structures.

## See Also:

"Object Creation and Initialization"

**make-instance**

**allocate-instance**

**initialize-instance**

# class-direct-initarg-defaults *Standard Generic Function*

## Purpose:

The generic function **class-direct-initarg-defaults** returns a list, each of whose elements is a list consisting of the name of an initialization argument defined for a slot in that exact class followed by the default value function and the default initial value form for that initialization argument. The default value function is the function whose body is the default initial value form; this function is created in the lexical environment of the **defclass** form that contains the default initial value form.

## Syntax:

**class-direct-initarg-defaults** *class* [*Generic Function*]

## Method Signatures:

**class-direct-initarg-defaults** (*class* `standard-class`) [*Primary Method*]

## Arguments:

The *class* argument is a class object.

## Values:

The result is a list of the form

(*initialization-argument-name default-value-function default-initial-value-form*) ... ).

The default initial value form is the form that was originally specified. It is retained only for documentation. The results are undefined if the default initial value form is evaluated.

The default value function is what actually gets called; its effect is equivalent to enclosing the default initial value form in the appropriate lexical environment. The default value function takes no arguments.

## Remarks:

The results are undefined if the value returned by this function is modified.

It is permitted, but not required, for an implementation to return values that share with internal data structures.

# class-direct-slot-initargs *Standard Generic Function*

## Purpose:

The generic function **class-direct-slot-initargs** returns a list, each of whose elements is a list consisting of the name of an initialization argument defined for a slot in that exact class followed by the names of all slots defined in the **defclass** form for that class for which that initialization argument is defined.

## Syntax:

**class-direct-slot-initargs** *class* [*Generic Function*]

## Method Signatures:

**class-direct-slot-initargs** (*class* `standard-class`) [*Primary Method*]

## Arguments:

The *class* argument is a class object.

## Values:

The result is a list of the form `((`*initialization-argument-name slot-name* `...) ... )`.

## Remarks:

The results are undefined if the value returned by this function is modified.

It is permitted, but not required, for an implementation to return values that share with internal data structures.

# compute-applicable-methods *Standard Generic Function*

## Purpose:

The generic function **compute-applicable-methods** returns a list of the methods of the generic function that are applicable for the given arguments.

## Syntax:

**compute-applicable-methods** *generic-function argument-list* [*Generic Function*]

## Method Signatures:

**compute-applicable-methods** (*generic-function* `standard-generic-function`) *argument-list* [*Primary Method*]

## Arguments:

The *generic-function* argument is a generic function object.

The *argument-list* argument is the argument list for that generic function for which the applicable methods are to be determined.

## Values:

The result is a list of method objects.

## Remarks:

The results are undefined if the value returned by this function is modified.

It is permitted, but not required, for an implementation to return values that share with internal data structures.

# default-initargs *Standard Generic Function*

## Purpose:

The system-supplied method for **default-initargs** implements the **:default-initargs** class option by appending initialization arguments that do not appear in the *initarg-list* argument to the *initarg-list* argument and returning the result.

The order of the initialization arguments appended to the list is determined by the rules for duplicate initialization arguments presented in the section "Rules for Duplicate Initialization Arguments."

The *initarg-list* argument is not modified.

## Syntax:

**default-initargs** *class initarg-list* [*Generic Function*]

## Method Signatures:

**default-initargs** (*class* `standard-class`) *initarg-list* [*Primary Method*]

## Arguments:

The *class* argument is a class object.

The *initarg-list* argument is an initialization argument list.

## Values:

The result is an initialization argument list.

## Remarks:

Programmers can customize the behavior of **default-initargs** by defining additional methods for it or replacing the system-supplied method on **standard-class**.

## See Also:

"Object Creation and Initialization"

"Rules for Duplicate Initialization Arguments"

# finalize-inheritance *Standard Generic Function*

## Purpose:

The generic function **finalize-inheritance** is called at least once before a class is instantiated and is called again whenever the class or any of its superclasses is redefined in any way or whenever an initialization method is defined, redefined, or undefined.

The generic function **finalize-inheritance** is not intended to be called by programmers. Programmers are expected to write methods for it.

## Syntax:

**finalize-inheritance** *class* `&key :slots :methods :initargs` [*Generic Function*]

## Method Signatures:

**finalize-inheritance** (*class* `standard-class`) [*Primary Method*]
`&key :slots :methods :initargs`

## Arguments:

The *class* argument is a class object.

The **:slots**, **:methods**, and **:initargs** arguments are boolean values that are true when the specified type of inheritance needs to be recomputed.

## Values:

The value returned by **finalize-inheritance** is ignored.

## Remarks:

The system-supplied method on **standard-class** conspires with methods for **make-instance**, **default-initargs**, **check-initargs**, **allocate-instance**, and **initialize-instance** to speed up object creation by precomputing information and storing it in slots of the class. This optimization is implementation dependent, but the **finalize-inheritance** mechanism that makes such optimizations possible is standardized.

Users with special optimization needs can write methods for **finalize-inheritance** to precompute information based on inherited information and to update the precomputed information whenever changes occur.

**See Also:**

**make-instance**

**check-initargs**

# method-keyword-names *Standard Generic Function*

## Purpose:

The generic function **method-keyword-names** returns a list of symbols indicating the keyword names of the keyword parameter specifiers in the given method's lambda-list. The result is the symbol **&allow-other-keys** instead of a list if the method's lambda-list contains that symbol.

## Syntax:

**method-keyword-names** *method* [*Generic Function*]

## Method Signatures:

**method-keyword-names** (*method* `standard-method`) [*Primary Method*]

## Arguments:

The *method* argument is a method object.

## Values:

The result is a list of symbols or the symbol **&allow-other-keys**.

# slot-boundp-using-class *Standard Generic Function*

## Purpose:

The generic function **slot-boundp-using-class** tests whether a specific slot in an instance of a given class is bound.

## Syntax:

**slot-boundp-using-class** *class instance slot-name* [*Generic Function*]

## Method Signatures:

**slot-boundp-using-class** (*class* `standard-class`) *instance slot-name* [*Primary Method*]

## Arguments:

The arguments are the class of the instance, the instance itself, and the name of the slot. The *class* argument is a class object.

## Values:

The generic function **slot-boundp-using-class** returns true or false.

## Remarks:

The set of arguments (including the class of the instance) facilitates defining methods on the metaclass for **slot-boundp-using-class**.

The generic function **slot-boundp-using-class** is intended for use by **slot-boundp**.

## See Also:

**slot-boundp**

# slot-makunbound-using-class *Standard Generic Function*

## Purpose:

The generic function **slot-makunbound-using-class** restores a slot in an instance to the unbound state.

## Syntax:

**slot-makunbound-using-class** *class instance slot-name* [*Generic Function*]

## Method Signatures:

**slot-makunbound-using-class** (*class* `standard-class`) *instance slot-name* [*Primary Method*]

## Arguments:

The arguments are the class of the instance, the instance itself, and the name of the slot. The *class* argument is a class object.

## Values:

The instance is returned as the result.

## Remarks:

The set of arguments (including the class of the instance) facilitates defining methods on the metaclass for **slot-makunbound-using-class**.

The generic function **slot-makunbound-using-class** is intended for use by **slot-makunbound**.

## See Also:

**slot-makunbound**

# slot-value-using-class *Standard Generic Function*

## Purpose:

The generic function **slot-value-using-class** returns the value contained in the slot *slot-name* of the given object of the given class. If there is no slot with that name, an error is signaled.

The macro **setf** can be used with **slot-value-using-class** to change the value of a slot.

## Syntax:

**slot-value-using-class** *class object slot-name* [*Generic Function*]

## Method Signatures:

**slot-value-using-class** (*class* `standard-class`) *object slot-name* [*Primary Method*]

## Arguments:

The arguments are the class of the object, the object itself, and the name of the slot. The *class* argument is a class object.

## Values:

The result is the value contained in the given slot.

## Remarks:

The set of arguments (including the class of the instance) facilitates defining methods on the metaclass for **slot-value-using-class**.

The generic function **slot-value-using-class** is intended for use by **slot-value**.

## See Also:

**slot-value**