

The Case Window¹⁸²

A case window displays the structure of a case of a method.

Banner¹⁸²

The logo in the banner indicates the type of method (Plain, Get, Set, Initialization, or Local) and is followed by the name of the method. The number before the colon indicates the number of the case in the sequence of cases that constitute the method, while the number after the colon indicates the total number of cases in the method.

The banner also contains three case control buttons. They are from left to right: the Previous case arrow, Case list button, and Next case arrow.¹⁸²

pening¹⁸²

Double-clicking a method icon opens its first case window. Other actions that open case windows are discussed in the “Manipulation of Cases” section and the “Opening Operations: The Left and Right ‘Click Spots’ ” section, both below in this chapter.

Creating¹⁸³

Cmd-clicking in space creates an operation, Cmd-clicking just above the top of an operation creates a terminal, and Cmd-clicking just below the bottom of an operation creates a root. Terminals and roots cannot be created on certain operations with enforced arity (see the “Enforcement of Corresponding Arity” section below in this chapter).

Copying¹⁸³

Copying an operation also copies all its roots and terminals but not any associated datalinks or synchro links. If the operation is a Local, its method is also copied.

NOTE: Input and output bars, roots, and terminals cannot be copied, cut, pasted, or replicated.

Deleting⁸³

o

Deleting an operation deletes all its terminals and roots, and any synchro link beginning or ending on the operation.

o

Deleting a Local operation deletes its associated Local method.

o

Deleting a terminal or root causes all associated datalinks to be deleted.

o

Input and output bars cannot be deleted.

o

On certain operations in which arity is enforced, terminals and roots cannot be deleted (see the “Enforcement of Corresponding Arity” section below in this chapter).

Naming⁸³

When the name of an operation is changed, the operation is resized to accommodate the new name.

Specifying Execution Order⁸³

A synchro is a connector between two operation icons which specifies their order of execution. If operation A is selected, and operation B is Option-clicked, a synchro will be drawn between them, indicating that operation A executes before operation B does. This does not necessarily mean that operation B will execute immediately after A; other operation executions may intervene.

Connecting / Disconnecting⁸⁴

o

If a root A is selected, Option-clicking on a terminal B creates a datalink from A to B, or deletes an existing datalink from A to B.

o

If a terminal B is selected, Option-clicking on a root A creates a datalink from A to B, or deletes an existing datalink from A to B.

o

If an operation A is selected, Option-clicking on an operation B creates a synchro link from A to B, or deletes an existing synchro link from A to B.

Commenting⁸⁴

Operations, roots, and terminals can have comments.

Dragging⁸⁴

When an operation is dragged, its terminals and roots move with it.

Dragging of terminals and roots is restricted to the horizontal axis. When a terminal or root is dragged past the end of its operation, the operation grows symmetrically about its midpoint to accommodate the new position of the dragged terminal or root.

Similarly, if a terminal or root near the end of an operation is dragged towards the center, the operation shrinks to the extent allowed by its name and other terminals and roots.

Dragging a Loop annotated terminal (or root) moves the corresponding Loop annotated root (or terminal).

Certain roots and terminals cannot be dragged: namely, the roots of Constants or Partition-annotated operations, and the terminals of Matches.

Transformations¹⁸⁴

An operation can be transformed by change of name or by annotation.

If an operation is transformed to one of lesser arity, excess terminals and roots are deleted from right to the left. If an operation is transformed to one of greater arity, extra terminals and roots are added to the right.

For example, if a choose operation with inputs a, b, c and output A is changed to a Get operation, the Get operation has a single input (a) and two outputs (A and the new root added to the right of A).¹⁸⁴

Transformation by Name Change¹⁸⁵

Changing the name of an operation to that of a Mac Method enforces the proper arity.

Changing the name of an operation by changing its name to that of a user-defined method enforces arity in one of two ways, provided that the name editing is followed by pressing the Return key.

o

If the name of the operation is a universal reference, context-determined reference, or an explicit class method reference, the arity of the uniquely determined method is applied to the operation.

o

If the name of the operation is a data-determined reference, a dialog is opened displaying a scrolling list of classes in which methods of the given name are defined. Selecting one of these classes and clicking the Select button (or equivalently, double-clicking a class) dismisses the dialog and applies the arity of the method from the chosen class to the operation.¹⁸⁵

annotating Operations⁹⁸

Annotation Effect

Opers : Simple

Preserves previous arity.

Controls : Simple

Preserves previous arity. Removes Super or Repeat annotation and any associated control from operation, and all annotations from terminals and roots.

Constant

Enforces proper arity. The name becomes the value of the Constant. ⁹⁹

Match

Enforces proper arity. The name becomes the value of the Match.

Persistent

Reduces arity to allowable maximum, if necessary.

Instance

Enforces proper arity.

Get

⁹⁸ Enforces proper arity.

Set

Enforces proper arity.

Local

Preserves previous arity.

Evaluate

Reduces the name to an empty string if it is not a valid arithmetic expression, and forces the root arity to one.

Super

Preserves previous arity.

Repeat

Transforms operation into a multiplex while preserving previous arity.

In the following, verifying an operation name means checking that it is appropriate for an operation of a particular kind. If verification fails, the empty string replaces the erroneous name.

Mac Constant

Verifies name and enforces proper arity.

Mac Match

Verifies name and enforces proper arity.

Mac Global

Reduces arity to allowable maximum, if necessary.

Mac Address

Verifies name and enforces proper arity.

Mac Get Field

Verifies name and enforces proper arity.

Mac Set Field

Verifies name and enforces proper arity.

Annotating Roots and Terminals

Annotation

Effect

Simple

Preserves arity, and if no Loop or List root or terminal remains, the Repeat annotation is removed

from the operation.

¹⁸⁸

Inject

Adds a new Inject terminal to fixed-arity operations; otherwise, converts the selected terminal into an Inject. The name of the operation is removed. Only one terminal per operation can be annotated as Inject. Mac Methods cannot have Inject terminals.

¹⁸⁸

Partition

Enforces proper arity. The selected terminal is transformed into a List, and the two Partition roots True and False are created. This annotation can be applied only to a terminal of an operation with no roots.

¹⁸⁸

List

Preserves previous arity.

Loop

When applied to a terminal (or root), if the operation has no simple root (or terminal), a new root (or terminal) is added and the selected terminal (or root) and new root (or terminal) are annotated as a matching Loop.

¹⁸⁸

If the operation has simple roots (or terminals), the selected terminal (or root) and the horizontally closest simple root (or terminal) are annotated as a matching Loops. In both situations, the operation is annotated as Repeat if it is not already so annotated.

¹⁸⁸

Deleting Annotated Roots and Terminals⁹⁹

Deleting

Effect

Partition

Partition roots cannot be deleted. Deleting the List terminal simplifies the True and False roots, and removes the Repeat annotation from the operation.

⁹⁹

Loop

Deleting a Loop terminal (or root) simplifies the matching Loop root (or terminal). If it is the only Loop or List root or terminal, the Repeat annotation is removed from the operation.

List

Deleting a List terminal (or root) removes the Repeat annotation from the operation if there are no remaining List or Loop terminals or roots.

Combinations of Creating, Connecting, and Annotating⁸⁹

Whenever an action results in the creation of several elements including an operation, the new operation becomes the selected element, unless the Command and Option keys have been depressed at the same time (see the “General Rules in the Editor” section above in this chapter).

If a root (or terminal) is selected, Option-clicking near the top (or bottom) of an operation creates a terminal (or root) and a datalink connecting it with the selected root (or terminal). The originally selected root (or terminal) remains selected.

The effect of Option-clicking when a root (or terminal) is selected is further extended if the click is in space. Option-clicking in space creates an operation with a terminal (or root) and connecting datalink. ⁸⁹

If an operation is selected, Option-clicking in space creates a new operation and a synchro from the original operation to the new one.

A double-click on a terminal creates a new operation with a single root, annotates it as a Constant, and creates a datalink between the terminal and the root of the Constant.

A double-click on a root creates a new operation with a single terminal, annotates it as a Match, and creates a datalink between the root and the terminal of the Match.

A double-click near the top (or bottom) of an operation, but not on an existing terminal (or root), first creates a new terminal (or root), and then, as described in the preceding paragraph, a new Constant (or Match), root (or terminal), and datalink. ⁸⁹

Selection of Element Groups⁹⁰

Groups of operations or groups of roots and terminals can be selected. However, operations cannot be selected at the same time as roots or terminals. Marquee selection applies only to operations.

Manipulation of Groups⁹⁰

Locals⁹⁰

The complexity of a case can be reduced by grouping a collection of operations into a Local.⁹⁰

he annotation Oper to Local in the Oper menu applies to a group of selected operations. This group is transformed into a Local method associated with a Local operation that replaces the group in the case. The topology of the case is adjusted appropriately. All the datalinks (including synchros) that connect operations within the group are preserved.

The terminals and roots of the Local operation are linked, respectively, to the roots and terminals of the remaining portion of the case, preserving the original horizontal order. However, any synchros that connect operations in a group with those outside of it are removed.

omments attached to the roots from the remaining portion of the case, to which terminals of the Local operation are connected, are copied to the corresponding roots of the input bar of the Local method.

editing

Copy Object

The group of selected operations and all links within the group are copied into the Object clipboard.

Paste Object

The contents of the Object clipboard are copied into the case.

Replicate Object

Acts as Copy Object and Paste Object applied to the selected group, but does not

alter the contents of the Object clipboard.

Cut Object⁹²

Selecting this item is equivalent to copying and deleting the selected group of operations.

Dragging⁹²

Groups of terminals and roots cannot be dragged.

Enforcement of Corresponding Arity⁹²

Correspondence between the terminals of a Local operation and the roots of the input bars of the cases of its corresponding Local method is enforced. The same applies to correspondence between the roots of a Local operation and the terminals on the output bars of the cases of its Local method.

Input and output arity is also enforced between cases of a method. Creation or deletion of any root on an input bar (or terminal on an output bar) automatically creates or deletes the corresponding root (or terminal) in all the remaining cases.

Propagation of Comments⁹²

When a new method is created by a double-click on an operation, or an operation is annotated as a Local, the comments associated with the terminals (or roots) of the operation appear at the corresponding roots of the input bar (or terminals of the output bar) of the created method or Local method.⁹²

however, if a terminal of the operation does not have a comment, but is attached to a root with a comment, this comment is propagated to the corresponding root of the input bar of the new method or Local method, as shown in the picture.⁹³

omments are automatically propagated only for new methods and Locals. If you add comments to an existing method or Local and want them to be propagated, select the method or Local and choose Propagate Comments from the Info menu.

Opening Operations: The Left and Right “Click Spots”⁹³

Double-clicking either end of an operation produces an effect that depends on the annotation of the operation, the kind of method reference made by the name, and the end on which the click occurs.

If the operation references an existing element, clicking either end opens an appropriate window or dialog. Otherwise, double-clicking the left end creates the element referenced by the operation, and double-clicking the right end issues a warning to the user that the referenced element does not exist.

Below is a table summarizing the results of double-clicks on the ends of operations, according to the name and annotation of the operation.⁹³

Operation
Left End<
>Right End

Simple⁹³
< name>

Left End<	If <name> is the empty string, a dialog with a scrolling list of names of universal methods is displayed. Selecting a method transfers its name to the operation and opens the first case window of the method.
-----------	---

>Right End	Same as left end.
------------	-------------------

Left End<	Otherwise, if a universal method called <name> exists, its first case window is opened.
-----------	---

>Right End	Same as left end.
------------	-------------------

Left End<	Otherwise, if no universal method called <name> already exists, the method is created and its first case window is opened.
-----------	--

>Right End	An error message is produced.
------------	-------------------------------

Simple⁹⁴
/<name>

Left End<	Opens a dialog with a scrolling list of all classes that do not have a method <name>. On selecting a class, the method is created and its first case window is opened.
-----------	--

>Right End	Opens a dialog with a scrolling list of all classes that have a method <name>. On selecting a class, the first case window of the method is opened.
------------	---

Simple⁻⁹⁴
//<name>

Left End< If the class containing the method in which the operation is located
 has a method called <name>, the first case window of this method is
 opened.

>Right End Same as left end.

Left End< Otherwise, the method is created and its first case window is
 opened.

>Right End An error message is produced.

Simple⁻⁹⁴
<class>/<name>

Left End< If a class called <class> exists and has a method <name>, the first
 case window of this method is opened.

>Right End Same as left end.

Left End< If a class called <class> exists and does not have a method <name>,
 the method is created and its first case window is opened.

>Right End An error message is produced.

Left End< Otherwise an error message is produced.

>Right End Same as left end.

Evaluate⁻⁹⁴
No action. No action.

Local⁻⁹⁵
Opens first case window. Same as left end.

Constant⁻⁹⁵
Opens a Change Value dialog. Same as left end.

Match⁻⁹⁵
Opens a Change Value dialog. Same as left end.

Persistent⁻⁹⁵

Left End< If the operation has no name, a dialog is opened with a scrolling list

of persistents. Selecting a persistent transfers its name to the operation, and opens a Value window displaying its value.

>Right End Same as left end.

Left End< If the persistent exists, a Value window is opened, displaying its value.

>Right End Same as left end.

Left End< Otherwise, the persistent is created and a Value window opened.

>Right End An error message is produced.

Instance_{set}

Left End< If the operation name has no name, a dialog is opened with a scrolling list of classes. Selecting a class transfers its name to the operation, and opens the Attributes window of the class.

>Right End If operation name is the empty string, a dialog is opened with a scrolling list of classes. Selecting a class transfers its name to the operation and opens the Methods window of the class.

Left End< Otherwise, if the class exists, its Attributes window is opened.

>Right End Otherwise, if the class exists, its Methods window is opened.

Left End< Otherwise, if the operation has a name and the class does not exist, the Classes window is brought to the front and the class is created.

>Right End Otherwise, an error message is produced.

<name>_{set} Get or Set

Left End< No action if the operation has no name.

>Right End Same as left end.

Left End< Otherwise, opens a dialog with a scrolling list of all classes that do not have an attribute <name>. On selecting a class, dialogs are displayed (for specifying whether the attribute is to be a class or an instance attribute, and for setting its default value) as for runtime creation of attributes. (Refer to chapter 3, "The Interpreter Environment").

>Right End Opens a dialog with a scrolling list of all classes which have an attribute <name>. On selecting a class, a Value window is opened displaying the default value of the attribute.

/<name>⁹⁶
Get or Set

Left End< Opens a dialog with a scrolling list of all classes that do not have a
Get method <name>. On selecting a class, a Get method is created and
its first case window opened.

>Right End Opens a dialog with a scrolling list of all classes that have a Get
method <name>. On selecting a class, the first case of the Get method
is opened.

Operation
Left End Right End

Mac Constant⁹⁶
Opens the 'Info... dialog. Same as left end.

Mac Match⁹⁶
Opens the 'Info... dialog. Same as left end.

Mac Global⁹⁶
Opens the 'Info... dialog. Same as left end.

Mac Address⁹⁶
Opens the 'Info... dialog. Same as left end.

Mac Get Field⁹⁶
Opens the 'Info... dialog. Same as left end.

Mac Set Field⁹⁶
Opens the 'Info... dialog. Same as left end.

NOTE: With Mac Address, Mac Get Field, and Mac Set Field operations, if there exist multiple Mac
Structs containing a field with the specified name, a dialog with a scrolling list of all possible owning Mac
Structs is displayed.⁹⁶

Manipulation of Cases⁹⁷

The case-list button, previous-case arrow, and next-case arrow provide access to the different cases of a
method.

Case Controls ⁹⁷

o

Clicking on the previous-case (or next-case) arrow opens a window onto the previous (or next) case, if such a case exists.

o

Option-clicking on the previous-case (or next-case) arrow creates a new case, inserted in the case list directly to the left (or right) of the current case, and opens a window onto it.

o

Command-clicking on the previous-case (or next-case) arrow closes the current case window and opens a window onto the previous (or next) case, if such a case exists. ⁹⁷

o

Option-clicking in the case list button of a case window of a local will open its parent local or method. If the Command-key is also pressed, the current window will close.

o

Option-clicking the case list button of an execution (i.e., dotted background) window opens its calling method (or local method). If the current window is already at the top of the stack, the stack window will open. Pressing the Command-key along with the Option key will close the current execution window.

Case List Pane ⁹⁷

The Case List pane displays a numbered, draggable icon for each case of the method.

Opening ⁹⁸

Clicking on the case-list button opens the Case List pane.

Closing ⁹⁸

Clicking in body of the case window or on the case-list button closes the Case List pane.

Deleting^{98*}

When a case icon is deleted, the remaining icons are renumbered appropriately.

Commenting^{98*}

Case icons cannot be commented.

Actions^{98*}

0

Clicking in space (before, after, or between the case icons) in the Case List pane creates a case icon and renumbers the cases to the right of the click.

0

Double-clicking on a case icon opens the corresponding case window.

Dragging^{98*}

Dragging is restricted to the horizontal. Dragging case icons reorders the cases, which are then renumbered.

Closing Case Windows^{98*}

Command-clicking in the close box of a case window closes all case windows for that method.