12. BREAKING AND TRACING

A number a functions and methods are available in the LOOPS environment to facilitate the process of finding and correcting bugs in user-written LOOPS code. These give you the capability to interrupt or trace methods so that you can examine the state of the computations by using the Interlisp-D Break package; see the *Interlisp-D Reference Manual*.

In addition to being able to break and trace methods, you can break and trace accesses to data within objects. For example, you can determine when a process is attempting to change a class variable or is trying to read the value of an instance variable. This feature gives you a powerful tool to assist in the understanding of the behavior of a piece of code from both a functional view and a data view.

12.1 Breaking and Tracing Methods

The Interlisp-D environment provides a number of features for breaking and tracing functions. LOOPS methods are implemented as Lisp functions, so the breaking and tracing of method invocation is similar to Interlisp-D.

The following table describes the methods in this section.

Name	Type	Description
BreakMethod	Method	Breaks a method of a class.
TraceMethod	Method	Traces a method of a class.
UnbreakMethod	Method	Unbreaks or untraces a method of a class.
SelectorsWithBreak	Method	Returns a list of selectors whose implementations have a break.

(← *self* **BreakMethod** *selector*) [Method of Class]

Purpose: Breaks a method of a class.

Behavior: Varies according to the argument.

- If *selector* is NIL, a menu appears showing the selectors associated with the class *self* that have not already been broken. If you do not make a choice from the menu, this method returns the symbol **NothingBroken**.
- If *selector* is non-NIL and is not associated with *self*, an error occurs stating that *selector* was not found in *self*.

If a method is broken, this fact is printed in the Prompt Window. The broken method function is added to the list **BROKENFNS**. (See the *Interlisp-D Reference Manual* for more information on **BROKENFNS**.)

Arguments: *self* Must be bound to a class.

	selector	Must be a selector associated with *self* or NIL.

Returns: The symbol **NothingBroken** or NIL.

Categories: Class

Example: The following command causes a break when the message **Open** is sent to any window:

```
12←(← ($ Window) BreakMethod 'Open)
```

(← *self* **TraceMethod** *selector*) [Method of Class]

Purpose: Traces a method of a class.

Behavior: Varies according to the argument.

- If *selector* is NIL, a menu appears showing the selectors associated with the class *self* that have not already been broken. If you do not make a choice from the menu, this method returns the symbol **NothingTraced**.
- If *selector* is non-NIL and is not associated with *self*, an error occurs stating that *selector* was not found in *self*.

If a method is traced, this fact is printed in the Prompt Window. The traced method function is added to the list **BROKENFNS**. (See the *Interlisp-D Reference Manual* for more information on **BROKENFNS**.) Whenever the function is called a message will be printed to a trace window, when it is exited a message will be printed with the returned value.

Arguments:	*self*	Must be bound to a class.
	selector	Must be a selector associated with *self* or NIL.

Returns: The symbol **NothingTraced** or NIL.

Categories: Class

(← *self* **UnbreakMethod** *selector*) [Method of Class]

Purpose: Unbreaks or untraces a method of a class.

Behavior: Varies according to the argument.

- If *selector* is NIL, a menu appears showing the selectors associated with the class *self* that have been broken. If you do not make a choice from the menu, this method returns the symbol **NothingUnbroken**.
- If *selector* is non-NIL and is not associated with *self*, an error occurs stating that *selector* was not found in *self*.

If a method is unbroken, its method function is removed the list **BROKENFNS**. (See the *Interlisp-D Reference Manual* for more information on **BROKENFNS**.) The value return is a list containing the name of the unbroken method function.

Arguments:	*self*	Must be bound to a class.
	selector	Must be a selector associated with *self* or NIL.

Returns: The symbol **NothingUnbroken** or a list containing the name of the unbroken method function.

Categories: Class

(← *self* **SelectorsWithBreak**) [Method of Class]

Purpose:	Return a list of selectors whose implementations have a break.
Behavior:	Searches through the list **BROKENFNS** collecting all selectors of method functions that begin with the class name of *self*. (See the *Interlisp-D Reference Manual* for more information on **BROKENFNS**.)
Arguments:	*self* Must be bound to a class.
Returns:	A list of selectors of *self*.
Categories:	Class

12.2 Breaking and Tracing Data

Breaking or tracing functions or methods cause interruptions to occur in a computation when a function or method is entered. Breaks or traces on data can be made to occur when either the data is to be read or changed. Only data that is contained within objects can be broken; this feature is not available to arbitrary Lisp data. Breaks and traces on data are implemented through the mechanism of active values. The following **ActiveValue** classes contain this mechanism:

- **BreakOnPut**
- **BreakOnPutOrGet**
- **TraceOnPut**
- **TraceOnPutOrGet**

You can use the methods and functions in this section to place or remove breaks on data. You can also add and remove traces and breaks through the inspector interface. See Chapter 18, User Input/Output Modules, for more information on the inspector.

Note: Breaking or tracing a variable effectively breaks or traces any IndirectVariable that points to it.

The following table describes the items in this section.

Name	Type	Description
BreakIt	Method	Puts a break on data within an object.
BreakIt	Function	Sends the message **BreakIt** to *self*.
TraceIt	Method	Puts a trace on data within an object.
TraceIt	Function	Sends the message **TraceIt** to *self*.
UnBreakIt	Function	Unbreaks broken data; untraces traced data.
BrokenVariables	Global Variable	Contains a list of broken or traced variables.

(← *self* **BreakIt** *varName propName &OPTIONAL (type 'IV) breakOnGetAlsoFlg*) [Method of Object]

Purpose: Puts a break on data within an object.

Behavior: Adds an entry to the list **BrokenVariables**.

- If *breakOnGetAlsoFlg* is T, creates an instance of the class **BreakOnPutOrGet** and adds the active value to the data specified by *self*, *varName*, *propName*, and *type*.
- If *breakOnGetAlsoFlg* is NIL, the active value instance is of the class **BreakOnPut**.

When a break occurs, the break window shows the nature of the break and which object and what variable is broken. See examples below.

Arguments:

self	Points to the object that contains the data to be broken.
varName	The name of the variable.
propName	If a property access is to be broken, this is the name of the property.
type	The type of the data. This can be IV, CV, or METHOD; the default is IV.
breakOnGetAlsoFlg	If this is non-NIL, breaks will occur when data is read. If this is NIL, breaks will occur only on attempts to write the data.

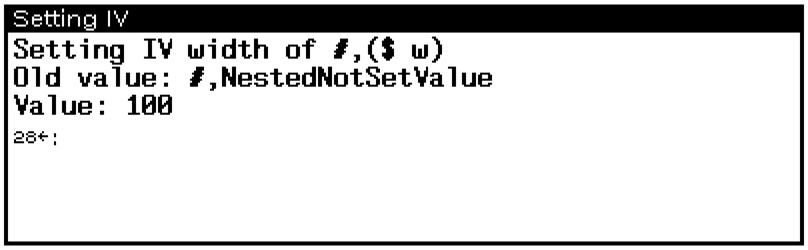
Returns: *self*

Categories: Object

Example: The following commands check if a window's width and height are going to change.

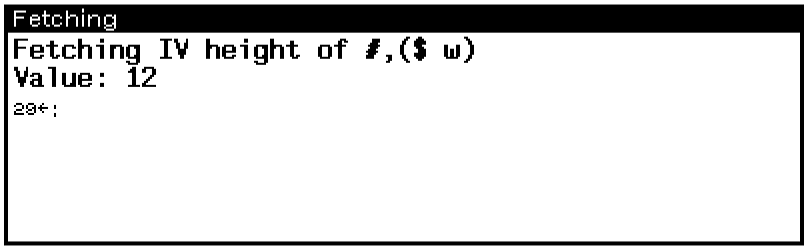
```
(← ($ Window) New 'w)
(← ($ w) BreakIt 'width)
(← ($ w) BreakIt 'height NIL NIL T)
```

Trying to change the width causes this break:



```
Setting IV
Setting IV width of #,($ w)
Old value: #,NestedNotSetValue
Value: 100
28←:
```

Trying to read the height causes this break:



```
Fetching
Fetching IV height of #,($ w)
Value: 12
29←:
```

(**BreakIt** *self varName propName type breakOnGetAlsoFlg*) [Function]

Behavior: Sends the message **BreakIt** to *self* passing the remainder of the arguments. See the method **BreakIt**, above, for details.

(← *self* **TraceIt** *varName propName &OPTIONAL (type 'IV) traceGetAlsoFlg*) [Method of Object]

Purpose: Puts a trace on data within an object.

Behavior: Adds an entry to the list **BrokenVariables**.

- If *traceGetAlsoFlg* is T, creates an instance of the class **TraceOnPutOrGet** and adds the active value to the data specified by *self*, *varName*, *propName*, and *type*.
- If *traceGetAlsoFlg* is NIL, the active value instance is of the class **TraceOnPut.**

When a trace occurs, a trace window appears if necessary, with the traced information printed in it. See examples below.

Arguments:

self	Points to the object that contains the data to be traced.
varName	The name of the variable.
propName	If a property access is to be traced, this is the name of the property.
type	The type of the data. This can be IV, CV, or METHOD; the default is IV.
traceOnGetAlsoFlg	If this is non-NIL, trace messages will occur when data is read. If this is NIL, trace messages will occur only on attempts to write the data.

Returns: *self*

Categories: Object

Examples: To monitor if a window's width and height are going to change, enter

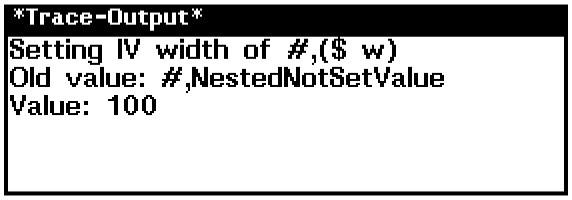
```
97←(← ($ Window) New 'w)
#,($& Window (NEW0.1Y%:.;h.eN6 . 495))

98←(← ($ w) TraceIt 'width)
#,($& Window (NEW0.1Y%:.;h.eN6 . 495))

99←(← ($ w) TraceIt 'height NIL NIL T)
#,($& Window (NEW0.1Y%:.;h.eN6 . 495))
```

Trying to change the width or the height causes a trace.

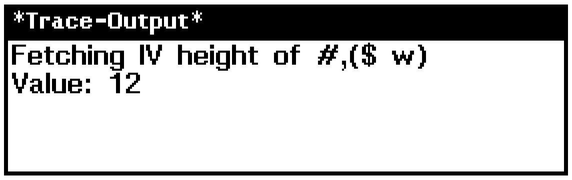
```
100←(change (@ ($ w) width) 100)
```



```
*Trace-Output*
Setting IV width of #,($ w)
Old value: #,NestedNotSetValue
Value: 100
```

```
100

101←(@ ($ w) height)
```



```
*Trace-Output*
Fetching IV height of #,($ w)
Value: 12
```

```
12
```

(**TraceIt** *self varName propName type breakOnGetAlsoFlg*) [Function]

Purpose/Behavior: Sends the message **TraceIt** to *self* passing the remainder of the arguments. See the method **TraceIt**, above, for details.

(**UnBreakIt** *self varName propName type*) [Function]

Purpose: Unbreaks broken data; untraces traced data.

Behavior: Varies according to the argument *self*.

- If *self* is NIL, iterates through the list **BrokenVariables** and removes the active values from the objects on that list. **BrokenVariables** is set to NIL.
- If *self* is not NIL, removes the active value from the data described by *self*, *varName*, *propName*, and *type*. The corresponding entry is removed from **BrokenVariables**. If there is no active value on the specified data, a break occurs saying that the specified data is not broken and type OK to continue.

Arguments:

self	Points to the object that contains the data to be traced.
varName	The name of the variable.
propName	If a property access is to be traced, this is the name of the property.
type	The type of the data. This can be IV, CV, or METHOD; the default is IV.

Returns: Value depends on the arguments.

- If *self* is NIL, the value of **BrokenVariables** before it was bound to NIL is returned.
- If *self* is non-NIL and there were no errors, the list containing *self*, *varName*, and *propName* is returned.

Example: The following command removes a break from the instance variable **id#** in the instance named **Datum12**:

```
(UnBreakIt ($ Datum12) 'id#)
```

BrokenVariables [Global Variable]

Purpose/Behavior: This is initialized to NIL. As data within objects is traced or broken, an entry is added to this list. Each entry contains the object, the variable name, the active value created to implement the break, the property name, and the type.

[This page intentionally left blank]