Autodesk University

Las Vegas, Nevada, December 3–6, 2002

Speaker Name: **Darren J. Young / dyoung@mcwi.com**

Course Title: **Introduction to Visual LISP's vl- functions**

Course ID: **CP42-3**

Course Outline: This course presents long-time AutoLISP® programmers with many of the new VL based functions provided in the Visual LISP™ programming environment. Many existing AutoLISP® programmers have yet to explore some of this powerful functionality and are still writing algorithms which can now be simplified with the use of just one or two VL functions. Discover new ways to upgrade your programming skills and capabilities with AutoLISP.

File/Directory Functions

(**vl-directory-files** *[directory pattern return-type]*)

Returns a list of files and/or directories.

directory =	String naming the directory to get files from. If absent or *nil* uses current directory.
pattern =	String naming the DOS directory & files extension to for the file and/or directory names. If absent or *nil* assumes *.*.
return-type =	-1 / directories only 0 / files & directories (default) 1 / files only

(**vl-file-copy** *source-file destination-file [append]*)

Copies or appends the contents of one file to another file. Returns an integer if successful or *nil* if unsuccessful.

source-file =	String name of file to be copied
destination-file =	String name of file to copy to
append =	If specified and not *nil* source file is appended to the end of the destination file

(**vl-file-delete** *filename*)

Deletes a file. Returns *T* if successful or *nil* if unsuccessful.

filename =	String name of file to delete

(**vl-file-directory-p** *filename*)

Determines if a file name is a directory. Returns *T* if string is a directory or *nil* if it's not a directory.

filename =	String containing a file/directory name

(**vl-file-rename** *old-filename new-filename*)

Renames a file. Return *T* if rename was successful or *nil* if rename failed.

old-filename =	String name of the file to rename
new-filename =	String of the new file name

(**vl-file-size** *filename*)

Returns an integer indicating the files size, *nil* if the file if unreadable or does not exist, and 0 if the file is empty.

filename =	String containing a file name

(**vl-file-systime** *filename*)

Returns a list of the last date and time the file was modified or *nil* if the file is not found.
(year month day-of-week day-of-month hours minutes seconds *[last data item should be ignored – bug]*)

filename =	String containing a file name

(**vl-filename-base** *filename*)

Returns the name of the file after stripping off the directory and file extension.

filename =	String containing a file directory/name

(**vl-filename-directory** *filename*)

Returns the name of the directory after stripping off the file name and file extension.

filename =	String containing a file directory/name

(**vl-filename-extension** *filename*)

Returns the name of the file extension after stripping off the directory and file name.

filename =	String containing a file directory/name

(**vl-filename-mktemp** *[pattern directory extension]*)

Returns a unique file name to be used as a temporary file.

pattern =	String containing the file name pattern. If *nil* or absent, "$VL~~" is used as the prefix.
directory =	String naming a directory for temporary files. If *nil* or absent, the following order is used…
	1st - The directory specified in pattern – if any
	2nd - The directory specified by the TMP environment variable
	3rd - The directory specified by the TEMP environment variable
	4th - The current Directory
extension =	String naming the extension used for the file. If *nil* or absent, uses the extension in pattern if any

(**vl-get-resource** *text-file*)

Returns the text stored in a *.txt file packaged within a VisualLISP application package (*.vlx)

t*ext-file =*	String naming the *.txt file packaged in a *.vlx. (Note: Do NOT specify the .txt extension)

AutoLISP/VisualLISP Variable Functions

(**vl-bb-ref** *'variable*)

Returns a value from a quoted variable from the black board namespace.

'variable =	Variable to get value from in black board namespace

(**vl-bb-set** *'variable value*)

Sets a value from a quoted variable to the black board namespace

'variable =	Variable name to store in black board namespace
value =	Value to store to variable in black board namespace

(**vl-doc-ref** *'variable*)

Returns a value from a quoted variable from the current documents namespace from a separate namespace VLX.

'variable =	Variable to get value from in current document namespace

(**vl-doc-set** *'variable value*)

Sets a value from a quoted variable in a separate namespace VLX to the current documents namespace.

'variable =	Variable name to store in current document's namespace
value =	Value to store to variable in current document's namespace

(**vl-propagate** *'variable*)

Copies the value of a variable into all open document namespaces and any subsequently opened documents.

'variable =	Variable to copy to all open document namespaces

(**vl-symbol-name** *'symbol*)

Returns a string containing the name of a quoted symbol.

(**vl-symbol-value** *'symbol*)

Returns the value bound to a quoted symbol.

(**vl-symbolp** *object*)

Indicates if the specified object is a symbol or not.

object =	Object to test if it's a symbol

AutoLISP/VisualLISP Function Functions

(**vl-acad-defun** *'symbol*)

Defines an AutoLISP function symbol (doesn't have "C:") as an external subroutine (accessible from an external ARX app)

'symbol =	A symbol identifying a function

(**vl-acad-undefun** *'symbol*)

Undefines an AutoLISP function symbol do it's no longer accessible from external ARX applications.

'symbol =	A symbol identifying a function

(**vl-arx-import** *['function | application]*)

Imports Object ARX/ADSRX applications into a separate namespace VLX application.

'function =	A symbol naming a function to be imported
application =	String naming an application whose functions are to be imported

(**vl-doc-export** *'function*)

Makes a function available to the current document.

'function =	A symbol naming the function to be exported

(**vl-doc-import** *application ['function…]*)

Imports a previously exported function into a VLX namespace.

application =	String naming a VLX application whose function are to be imported (don't include VLX extension)
'function =	One or more symbols naming function to be imported. If absent, all functions will be imported

(**vl-exit-with-error** *message*)

Passes control from a VLX error handler to the (*error*) function of the calling namespace.

message =	A string message

(**vl-exit-with-value** *value*)

Returns a value to the function that invoked the VLX from another namespace.

value =	Any value

(**vl-list-exported-functions** *[appname]*)

Lists exported functions.

appname =	String naming a loaded VLX application (don't include VLKX extension)

(**vl-list-loaded-vlx**)

Returns a list of separate namespace VLX files associated with the current document.

(**vl-load-all** *filename*)

Loads an AutoLISP file into all open AutoCAD drawings and any subsequently opened AutoCAD drawings.

filename =	A string naming an AutoLISP file to be loaded (always specify the extension LSP/FAS/VLX)

(**vl-load-com**)

Loads the extended VisualLISP functions that support ActiveX automation and reactors in VisualLISP.

(**vl-load-reactors**)

Same functionality as (vl-load-com) but maintained for backward compatibility.

(**vl-symbol-name** *'symbol*)

Returns a string containing the name of a quoted symbol.

(**vl-symbol-value** *'symbol*)

Returns the value bound to a quoted symbol.

(**vl-symbolp** *object*)

Indicates if the specified object is a symbol or not.

object =	Object to test if it's a symbol

(**vl-unload-vlx** *appname*)

Unloads a VLX application that's loaded in its own namespace.

appname =	String name of application to unload (do NOT include the VLX extension)

(**vl-vbaload** *filename*)

Loads a Visual Basic project

filename =	String name of Visual Basic project to load

(**vl-vbarun** *macroname*)

Runs a Visual Basic macro

macroname =	String name of a loaded Visual Basic macro to run

(**vl-vlx-loaded-p** *appname*)

Determines if a separate namespace VLX application is already loaded.

appname =	String naming a VLX application

(**vlax-add-cmd** *global-name 'func-sym [local-name cmd-flags]*)

Adds commands to the AutoCAD built-in command set.

global-name =	String
'func-sym =	Symbol naming an AutoLISP function with zero arguments
local-name =	String (defaults to *global-name*)
cmd-flags =	Integer bit flag (defaults to ACRX-CMD-MODAL + ACRX-CMD-REDRAW) ACRX-CMD-MODAL (0) ACRX-CMD-TRANSPARENT (1) ACRX-CMD-USEPICKSET (2) ACRX-CMD-REDRAW (4)

(**vlax-remove-cmd** *global-name*)

Removes a single command or a command group.

Registry Functions

(**vl-registry-delete** *reg-key [val-name]*)

Deletes the specified key or value from the Windows registry.

reg-key =	String specifying a registry key
val-name =	String containing the value of the *reg-key* entry

(**vl-registry-descendents** *reg-key [val-names]*)

Returns a list of subkeys or value names for the specified registry.

reg-key =	String specifying a registry key
val-name =	If specified the value names are listed. If *nil* or absent, subkeys are returned

(**vl-registry-read** *reg-key [val-name]*)

Returns data stored in the registry specified by the key/value pair.

reg-key =	String specifying a registry key
val-name =	String specifying the value of a registry entry

(**vl-registry-write** *reg-key [val-name val-data]*)

Creates a key or stores data in the Windows registry.

reg-key =	String specifying a registry key
val-name =	String specifying the value of a registry entry
val-data =	Data for the value of a registry entry

(**vlax-product-key**)

Returns the AutoCAD Window registry path.

List Functions

(**vl-catch-all-apply** *'function list*)

Passes a list of arguments to a function and traps any exceptions.

'function =	Function to pass list of arguments to
list =	List containing arguments to pass to the function

(**vl-catch-all-error-message** *error-object*)

Returns a string from a (vl-catch-all-apply) error object.

error-object =	An error object returned by (vl-catch-all-apply) when there's an error

(**vl-catch-all-error-p** *argument*)

Determines of the returned value (*argument*) from (vl-catch-all-apply) is an error object or not.

argument =	Value returned from (vl-catch-all-apply)

(**vl-consp** *list*)

Determines if a list is *nil*.

List =	A list

(**vl-list*** *object [object]…*)

Constructs and returns a list.

object =	Any AutoLISP object

(**vl-list-length** *list-or-cons-obj*)

Calculates the length of a true list.

list-or-cons-obj =	A true or dotted list

(**vl-position** *symbol list*)

Returns the index of the specified list item.

symbol =	Any AutoLISP symbol
list =	A true list

(**vl-remove** *remove list*)

Removes items from a list.

remove =	Item to remove from list
list =	List to remove item from

(**vl-remove-if** *predicate-fun list*)

Removes all items from a list that fail the test function

predicate-fun =	Any test function that accepts a single argument and returns T for any user specified condition
list =	List to be tested

(**vl-remove-if-not** *predicate-fun list*)

Removes all items from a list that pass the test function

predicate-fun =	Any test function that accepts a single argument and returns T for any user specified condition
list =	List to be tested

(**vl-sort** *list compare-fun*)

Sorts a list based on the comparison function and returns the sorted list.

list =	List to be sorted
compare-fun =	A comparison function that takes two arguments and returns *T* if the 1st precedes the 2nd in the sort order

(**vl-sort-i** *list compare-fun*)

Sorts a list based on the comparison function and returns a list of sorted index numbers.

list =	List to be sorted
compare-fun =	A comparison function that takes two arguments and returns *T* if the 1st precedes the 2nd in the sort order

String Functions

(**vl-string-elt** *string position*)

Returns the ASCII character code for the character in a string at the specified position.

string =	String to be examined
position =	Specified character position (0 based index) to return ASCII character code for

(**vl-string-left-trim** *char-set string*)

Removes the specified characters from the beginning of a string.

char-set =	String representing characters to trimmed
string =	String to remove characters from

(**vl-string-mismatch** *string1 string2 [position1 position2 ignore-case-p]*)

Returns the length of the longest common prefix for two strings starting at the specified positions.

string1 =	1st string to be matched
string2 =	2nd string to be matched
position1 =	Integer specifying start search position (0 based index) of string 1 (0 if absent)
position2 =	Integer specifying start search position (0 based index) of string 1 (0 if absent)
ignore-case-p =	If *T* match is not case sensitive

(**vl-string-position** *char-code string [start-pos [from-end-p]]*)

Searches for a character with the specified ASCII code in string, returning its position (0 based index).

char-code =	ASCII character code to search string for
string =	String to search
start-pos =	Starting position to begin search (0 based index)
from-end-p =	If *T* search starts from end of string continuing backwards until position is reached

(**vl-string-right-trim** *char-set string*)

Removes the specified characters from the end of a string.

char-set =	String representing characters to trimmed
string =	String to remove characters from

(**vl-string-search** *pattern string [start-pos]*)

Searches for the specified pattern in string returning the position it occurs (0 based index)

pattern =	String containing the pattern to be searched for
string =	String to search
start-pos =	Starting position to begin search (0 based index)

(**vl-string-subst** *new-string pattern string [start-pos]*)

Substitutes one string for another in a string.

new-string =	String to be substituted for pattern
pattern =	String containing the pattern to be replaced
string =	String to be searched for pattern
start-pos =	Starting position to begin search (0 based index)

(**vl-string-translate** *source-set dest-set string*)

Replaces characters in a string with a specified set of characters.

source-set =	A string of characters to be matched
dest-set =	String of characters to be substituted for those in source-set
string =	String to be searched and translated

(**vl-string-trim** *char-set string*)

Removes the specified characters from the beginning and end of a string.

char-set =	String representing characters to trimmed
string =	String to remove characters from

Conversion Functions

(**vl-list->string** *char-codes-list*)

Combines the characters associated with codes into a string.

char-codes-list =	A list of non-negative integers smaller than 256

(**vl-prin1-to-string** *data*)

Returns the string representation of AutoLISP data as if it were output by the (prin1) function.

data =	Any AutoLISP data

(**vl-princ-to-string** *data*)

Returns the string representation of AutoLISP data as if it were output by the (princ) function.

data =	Any AutoLISP data

(**vl-string->list** *string*)

Converts a string into a list of character codes.

string =	Any string

Conditional (Test) Functions

(**vl-every** *predicate-fun list [list]…*)

Checks if the predicate function is true for every element combination.

predicate-fun =	Any test function that accepts any number of argument as there are lists and returns T on a user specified condition
list =	A list to be tested

(**vl-member-if** *predicate-fun list*)

Determines of the predicate function is true (*T*) for one of the list members.

predicate-fun =	Any test function that accepts a single argument and returns T for any user specified condition
list =	A list to be tested

(**vl-member-if-not** *predicate-fun list*)

Determines of the predicate function is *nil* for one of the list members.

predicate-fun =	Any test function that accepts a single argument and returns T for any user specified condition
list =	A list to be tested

(**vl-some** *predicate-fun list [list]…*)

Checks if the predicate function is not *nil* for one element combination.

predicate-fun =	Any test function that accepts any number of argument as there are lists and returns T on a user specified condition
list =	A list to be tested

Miscellaneous Functions

(**vl-cmdf** *[arguments]…*)

Executes an AutoCAD command.

arguments =	AutoCAD commands and their options

(**vlisp-compile** *'mode filename [output-file]*)

Compiles AutoLISP source code into a FAS file.

VLAX-* ActiveX Object, Property, Method & Collection Functions

(**vlax-create-object** *prog-id*)

Creates a new instance of an application object.

(**vlax-dump-object** *object [T]*)

Lists an object's properties, and optionally, the methods that apply to the object.

(**vlax-erased-p** *object*)

Indicates if an object was erased.

(**vlax-for** *symbol collection [expression1 [expression2…]]*

Iterates through a collection of objects evaluating each expression.

(**vlax-get-acad-object**)

Gets the top level AutoCAD application object for the current AutoCAD session.

(**vlax-get-object** *program-id*)

Returns a running instance of an application object.

(**vlax-get-or-create-object** *program-id*)

Returns a running instance of an application object or creates a new instance if it's not running.

(**vlax-get-property** *object property*)

Gets a VLA-object's property

(**vlax-import-type-library** :tlb-filename *filename [*:methods-prefix *mprefix* :properties-prefix *pprefix* :constants-prefix *cprefix]*)

Imports information from a type library.

(**vlax-invoke-method** *object 'method argument [argument…]*)

Calls the specified ActiveX method'

(**vlax-map-collection** *object 'function*)

Applies a function to all objects in a collection.

(**vlax-method-applicable-p** *object 'method*)

Determines if an object supports the specified method.

(**vlax-object-released-p** *object*)

Determines if an object has been released.

(**vlax-property-available-p** *object 'property [check-modify]*)

Determines if an object has the specified property.

(**vlax-put-property** *object 'property argument*)

Set the property of an ActiveX object.

(**vlax-read-enabled-p** *object*)

Determines if an object can be read.

(**vlax-release-object** *object*)

Releases a drawing object.

(**vlax-typeinfo-available-p** *object*)

Determines whether TypeLib information is present for the specified type of object.

(**vlax-write-enabled-p** *object*)

Determines if an AutoCAD drawing object can be modified.

VLAX-* ActiveX Data & Variable Functions

(**vlax-ldata-delete** *dict key [private]*)

Erases LISP data from a drawing dictionary.

(**vlax-ldata-get** *dict key [default-data] [private]*)

Retrieves LISP data from a drawing dictionary or an object.

(**vlax-ldata-list** *dict [private]*)

Lists LISP data in a drawing dictionary.

(**vlax-ldata-put** *dict key data [private]*)

Stores LISP data in a drawing dictionary or an object.

(**vlax-ldata-test** *data*)

Determines if data can be saved over a session boundary.

(**vlax-make-safearray** *type '(lower-bounds . upper-bounds) ['(lower-bounds . upper-bounds)…]*)

Creates a safearray.

(**vlax-make-variant** *[value] [type]*)

Creates a variant data type.

(**vlax-safearray-fill** *safearray 'element-values*)

Stores data in the elements of a safearray.

(**vlax-safearray-get-dim** *safearray*)

Returns the number of dimensions in a safearray object.

(**vlax-safearray-get-element** *safearray element…*)

Returns an element from an array.

(**vlax-safearray-get-l-bound** *safearray dimension*)

Returns the lower boundary (starting index) of a dimension of an array.

(**vlax- safearray-get-u-bound** *safearray dimension*)

Returns the upper boundary (end index) of a dimension of an array.

(**vlax-safearray-put-element** *safearray index… value*)

Adds an element to an array.

(**vlax-safearray-type** *safearray*)

Returns the data type of a safearray.

(**vlax-variant-type** *variant*)

Determines the data type of a variant.

(**vlax-variant-value** *variant*)

Returns the value of a variant.

VLAX-* ActiveX Conversion Functions

(**vlax-3D-point** *list*) –or (**vlax-3D-point** *x y z*)

Creates an ActiveX Compatible (variant) 3d point structure.

(**vlax-ename->vla-object** *entity-name*)

Converts an entity name to an VisualLISP ActiveX object.

(**vlax-safearray->list** *safearray*)

Returns the elements of a safearray in list form.

(**vlax-tmatrix** *list*)

Returns a suitable representation for a 4 × 4 transformation matrix to be used in VLA methods.

(**vlax-variant-change-type** *safearray type*)

Returns the value of a variant after changing it from one data type to another.

(**vlax-vla-object->ename** *object*)

Converts a VLA-object to an AutoLISP entity name.

VLAX-* ActiveX Curve Functions

(**vlax-curve-getArea** *curve-obj*)

Returns the area inside the curve.

(**vlax-curve-getClosestPointTo** *curve-object given-point [extend]*)

Returns the point (in WCS) on a curve that is nearest to the specified point.

(**vlax-curve-getClosestPointToProjection** *curve-object given-point normal [extend]*)

Returns the closest point (in WCS) on a curve after projecting the curve onto a plane.

(**vlax-curve-getDistAtParam** *curve-object parameter*)

Returns the length of the curve’s segment from the curve’s beginning to the specified parameter.

(**vlax-curve-getDistAtPoint** *curve-object point*)

Returns the length of the curve's segment between the curve's start point and the specified point.

(**vlax-curve-getEndParam** *curve-object*)

Returns the parameter of the endpoint of the curve.

(**vlax-curve-getEndPoint** *curve-object*)

Returns the endpoint (in WCS) of the curve.

(**vlax-curve-getFirstDeriv** *curve-object parameter*)

Returns the first derivative (in WCS) of a curve at the specified location.

(**vlax-curve-getParamAtDist** *curve-object distance*)

Returns the parameter of a curve at the specified distance from the beginning of the curve.

(**vlax-curve-getParamAtPoint** *curve-object point*)

Returns the parameter of the curve at the point.

(**vlax-curve-getPointAtDist** *curve-object distance*)

Returns the point (in WCS) along a curve at the distance specified by the user.

(**vlax-curve-getPointAtParam** *curve-object parameter*)

Returns the point at the specified parameter value along a curve.

(**vlax-curve-getSecondDeriv** *curve-object parameter*)

Returns the second derivative (in WCS) of a curve at the specified location.

(**vlax-curve-getStartParam** *curve-object*)

Returns the start parameter on the curve.

(**vlax-curve-getStartPoint** *curve-object*)

Returns the start point (in WCS) of the curve.

(**vlax-curve-isClosed** *curve-object*)

Determines if the specified curve is closed.

(**vlax-curve-isPeriodic** *curve-object*)

Determines if the specified curve has an infinite range in both directions and there is a period value *dT,* such that a point on the curve at (*u* + *dT*) = point on curve (*u*), for any parameter *u*.

(**vlax-curve-isPlanar** *curve-object*)

Determines if there is a plane that contains the curve.

VLR-* Reactor Object Functions

(**vlr-acdb-reactor** *data callbacks*)

Constructs a reactor object that notifies when an object is added to, modified in, or erased from a drawing database.

(**vlr-command-reactor** *data callbacks*)

Constructs an editor reactor that notifies of a command event.

(**vlr-deepclone-reactor** *data callbacks*)

Constructs an editor reactor object that notifies of a deep clone event.

(**vlr-docmanager-reactor** *data callbacks*)

Constructs a reactor object that notifies of events relating to drawing-documents.

(**vlr-dwg-reactor** *data callbacks*)

Constructs an editor reactor object that notifies of a drawing event.

(**vlr-dxf-reactor** *data callbacks*)

Constructs an editor reactor object that notifies of an event related to reading or writing a DXF file.

(**vlr-editor-reactor** *data callbacks*)

Constructs an editor reactor object.

(**vlr-insert-reactor** *data callbacks*)

Constructs an editor reactor object that notifies of an event related to block insertion.

(**vlr-linker-reactor** *data callbacks*)

Constructs a reactor object that notifies your application every time an ARX application is loaded or unloaded.

(**vlr-lisp-reactor** *data callbacks*)

Constructs an editor reactor object that notifies of a LISP event.

(**vlr-miscellaneous-reactor** *data callbacks*)

Constructs an editor reactor object that does not fall under any other editor reactor types.

(**vlr-mouse-reactor** *data callbacks*)

Constructs an editor reactor object that notifies of a mouse event.

(**vlr-object-reactor** *owners data callbacks*)

Constructs an object reactor object.

(**vlr-sysvar-reactor** *data callbacks*)

Constructs an editor reactor object that notifies of a change to a system variable.

(**vlr-toolbar-reactor** *data callbacks*)

Constructs an editor reactor object that notifies of a change to the bitmaps in a toolbar.

(**vlr-undo-reactor** *data callbacks*)

Constructs an editor reactor object that notifies of an undo event.

(**vlr-wblock-reactor** *data callbacks*)

Constructs an editor reactor object that notifies of an event related to writing a block.

(**vlr-window-reactor** *data callbacks*)

Constructs an editor reactor object that notifies of an event related to moving or sizing an AutoCAD window.

(**vlr-xref-reactor** *data callbacks*)

Constructs an editor reactor object that notifies of an event related to attaching or modifying XREFs.

VLR-* Reactor Maintenence Functions

(**vlr-add** *object*)

Enables a disabled reactor object.

(**vlr-added-p** *object*)

Tests to determine if a reactor object is enabled.

(**vlr-beep-reaction** *[arguments]*)

Produces a beep sound.

(**vlr-current-reaction-name**)

Returns the name (symbol) of the current event, if called from within a reactor's callback.

(**vlr-data** *object*)

Returns application-specific data associated with a reactor.

(**vlr-data-set** *object data*)

Overwrites application-specific data associated with a reactor.

(**vlr-notification** *reactor*)

Determines whether or not a reactor will fire if its associated namespace is not active.

(**vlr-owner-add** *reactor owner*)

Adds an object to the list of owners of an object reactor.

(**vlr-owner-remove** *reactor owner*)

Removes an object from the list of owners of an object reactor.

(**vlr-owners** *reactor*)

Returns the list of owners of an object reactor.

(**vlr-pers** *reactor*)

Makes a reactor persistent.

(**vlr-pers-list** *[reactor]*)

Returns a list of persistent reactors in the current drawing document.

(**vlr-pers-p** *reactor*)

Determines whether or not a reactor is persistent.

(**vlr-pers-release** *reactor*)

Makes a reactor transient.

(**vlr-reaction-name** *reactor-type*)

Returns a list of all possible callback conditions for this reactor type.

(**vlr-reaction-set** *reactor event 'function*)

Adds or replaces a callback function in a reactor.

(**vlr-reactions** *reactor*)

Returns a list of pairs (event-name . callback_function) for the reactor.

(**vlr-reactors** *[reactor-type…]*)

Returns a list of existing reactors.

(**vlr-remove** *reactor*)

Disables a reactor.

(**vlr-remove-all** *[reactor-type]*)

Disables all reactors of the specified type.

(**vlr-set-notification** *reactor 'range*)

Defines whether or not a reactor's callback function will execute if its associated namespace is not active.

(**vlr-trace-reaction**)

A pre-defined callback function that prints one or more callback arguments in the Trace window.

(**vlr-type** *reactor*)

Returns a symbol representing the reactor type.

(**vlr-types**)

Returns a list of all reactor types.