Coding Style Sheet

Variable Names

Single Value Variables	*x*	Single valued x.
	x_value	Single valued x.
Multivalue Variables	*xs*	List of xs.
	xs_values	List of xs.

** The names of variables is discretionary. However, the name of the variable should show purpose and application. (Example: tb is not description enough to imply purpose, messageTextbox would be a more appropriate variable name.)

Functions

Data Access	*(getX)*	Acquires the data that the *x* variable contains.
	(setX x)	Sets the value for *x*. *** ACL2 is side-effect free, use let**
Network	*(receiveX)*	Acquires *x* from network resources.
	(sendX x)	Sends *x* to be posted via network resources.
Threading	*(startX x)*	Starts the execution of thread *x*.
	(stopX x)	Stops the execution of thread *x*.
	(sleepX x time)	Halts the execution of thread *x* for *time* milliseconds.
Events	*(getXListener)*	Acquires the listener (handler) assigned to the element *x*.
	(addXListener x)	Adds a listener (handler) to the element *x*.
GUI	*(constructX)*	Visually constructs the rendering for *x.*

** Functions defining so-called "utility functions" are to be named for their purpose with Camelback notation. Example would include (multiplexData xs ys).

Commenting

File Commenting

Commenting headers that are contained within files should be in the following notation:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; /server/utilities.lisp
;
; package server
; Created on January 25, 2013 by Matthew A. Crist.
; Team Dijkstra
;
; This file contains the functions that are required for data processing
; that may not normally be considered a part of the server technology
; but may be classified as a utility function.
;
; FUNCTION
;--------------------------------------------------------------------------
; (multiplexData xs ys) - multiplexes the data together into a single ;
;                         string.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; CHANGE LOG:
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; 0.0.1_20130124 - Initial file conception.
; 1.0.0_20130125 - Release candidate reversioning.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Function Commenting

Commenting functions should echo the function usage, purpose, arguments and return value:

```
; (multiplexData xs ys)
; Multiplexes the list of xs with the list of ys and returns a singular
; list that contains xs and ys.  If xs is greater than ys or ys is greater
; than xs, the remaining members of either list after one reaches length 0
; is appended to the end of the return list.
;
; xs - the first set to be multiplexed.
; ys - the second set to be multiplxed.
;
; returns - the set of xs and ys where (x1 y1 x2 y2 … xn yn).
```

Inline Commenting

Inline comments are discretionary for the programmer. Comments should be concise and to the point. Over-commenting functions should not occur and combining the purpose for blocks of code into a single comment should suffice.

Appropriate inline commenting:

```
; Assigns the first value of the list and rest of the list
(let* (x_value (car xs))
      (the_rest (cdr xs)))
```

File Naming Convention

Folder Names	Folder names should be named according to the client in which they support. For instance, server files should be located in a subfolder labeling "server" and client files should be in a sub folder labeled "client". Outside resources such as images should be contained within their respective folders in which they apply. Images related to the client will be located in "*/client/images/".* Likewise for server image files: *"/server/images"*.

** For multipart folder names, subfolders should be used: *"/client/gui/images" or "/client/buffer/logs/".*

File Names	The naming convention of files should be done according to the purpose of the functions that it contains. If functions are of utility purpose, we would contain those within a file labeled *"utilities.lisp"*. Likewise, GUI functions should be contained within a file labeled *"gui.lisp"*.

** For multipart file names, underscores should be used to separate words: *"io_utilities.lisp"*. It is recommended, however to use the multipart folder structure instead.

For all code, maximum with should be limited to 75 characters before a new line. New line breaks within the code are at the programmer's discretion, but readability should be maintained.