4. CHANGES TO INTERLISP-D IN LYRIC/MEDLEY

NOTE: Chapter 4 is organized to correspond to the original *Interlisp-D Reference Manual*, and explains changes that have occurred in Interlisp-D with the Lyric and Medley releases. To make it easy to use this chapter with the *IRM,* information is organized by *IRM* volume and section numbers. Section headings from the *IRM* are maintained to aid in cross-referencing.

Lyric information as well as Medley release enhancements are included. Medley additions are indicated with revision bars in the right margin.

VOLUME I—LANGUAGE

Chapter 3 Lists

Section 3.2 Building Lists From Left To Right

(I:3.7)

The functions **DOCOLLECT** and **ENDCOLLECT** are no longer supported.

(I:3.8)

The description of the **ADDTOSCRATCHLIST** function has been revised to read:

(ADDTOSCRATCHLIST *VALUE)* [Function]

For use inside a **SCRATCHLIST** form. *VALUE* is added on to the end of the value being collected by **SCRATCHLIST.** When the **SCRATCHLIST** returns, its value is a list containing all of the things that **ADDTOSCRATCHLIST** has added.

Section 3.10 Sorting Lists

(I:3.17)

(**SORT** *DATE COMPAREFN*) [Function]

There is no safe interrupt to **SORT**—if you abort a call to **SORT** by *any* means the possibility exists for losing elements from the list being sorted.

Chapter 6 Hash Arrays

(I:6.1)

(**HASHARRAY** *MINKEYS OVERFLOW HASHBITSFN EQUIVFN RECLAIMABLE REHASH-THRESHOLD*) [Function]

The function **HASHARRAY** has two new optional arguments, *RECLAIMABLE* and *REHASH-THRESHOLD*. If *RECLAIMABLE* is true, then entries in the hash table are considered "reclaimable" in the sense that the system is permitted to remove any key and its associated value from the hash table at any time. In practice, the contract is less severe: the system only removes keys when a hash table fills and is about to be rehashed, and then it only removes keys whose reference count is one, and to which there are thus no pointers outstanding except possibly from the stack (local variables). This is useful for hash tables that serve to cache information about Lisp objects to avoid recomputation; for example, the system hash table CLISPARRAY is now reclaimable. Discarding keys keeps the table from necessarily needing to grow, and potentially allows the storage consumed by both the key and value to be reclaimed.

Section 6.1 Hash Overflow

(I:6.3)

You should note changes to the wording of two of the possibilites for the overflow method:

The first sentence for **NIL** should read: The array is automatically enlarged by *at least* a factor of 1.5 every time it overflows.

The explanation for "a positive integer N" should read: The array is enlarged to include *at least* N more slots than it currently has.

Chapter 7 Integer Arithmetic

(I:7.5)

The variables **MIN.INTEGER** and **MAX.INTEGER** have been removed from the *Interlisp-D Reference Manual*. Therefore, calling **(MIN)** and **(MAX)** is an error.

(***I:7.7***)

(**FIXR** *N*) [Function]

When N is exactly half way between two integers, FIXR rounds it to the even number. For example (FIXR 1.5) ⇒ 2 and (FIXR 2.5) ⇒ 2.

Section 7.3 Logical Arithmetic Functions

The function **INTEGERLENGTH** does *not* coerce floating point numbers to integers; rather, it signals an error, "Arg not Integer". (This was true in Koto as well.)

Section 7.5 Other Arithmetic Functions

(I:7.13)

The algorithms for **SIN**, **COS**, and other trigometric functions have been tuned and are now accurate to at least six significant figures.

Chapter 8 Record Package

(I:8.11)

When using **BLOCKRECORD**, it is an error to try to declare a record with a zero-length field. Previously, the system would go into an infinite loop. In the Medley release, the system will now detect this and signal an error.

Chapter 9 Conditionals and Iterative Statements

Section 9.2 Equality Predicates

(I:9.3)

(**EQUALALL** *X Y*) [Function]

Add the following NOTE to the **EQUALALL** function:

Note: In general, **EQUALALL** descends all the way into all datatypes, both those defined by the user and those built into the system. If you have a data structure with fonts and pointers to windows, **EQUALALL** will descend into those also. If the data structures are circular, as windows are, **EQUALALL** can cause a Stack Overflow error.

Section 9.8.3 Condition I.s. oprs

UNTIL *N* (*N* a number) [I.S. Operator]

REPEATUNTIL *N* (*N* a number) [I.S. Operator]

These descriptions were included in the *Interlisp-D Reference Manual* in error and should be removed. **UNTIL** and **REPEATUNTIL** work *only* with predicate expressions, not numbers.

Chapter 10 Function Definition, Manipulation , and Evaluation

Section 10.2 Defining Functions

(I:10.11)

In the definition of the **MOVD** function, the sentence "**COPYDEF** is a higher-level function that only moves expr definitions, but..." should be revised to read:

COPYDEF is a higher-level function that not only moves expr definitions, but also works for variables, records, etc.

Section 10.5 Functional Arguments

(I:10.19)

FUNARG functionality (non-**NIL** second argument to **FUNCTION**) has been withdrawn. Most of the uses for Interlisp **FUNARG**'s are better written using the lexical closure functionality of Common Lisp.

Section 10.6.2 Interpreting Macros

The variables **SHOULDCOMPILEMACROATOMS** and **UNSAFEMACROATOMS** no longer exist.

Chapter 11 Variable Bindings and the Interlisp Stack

(II:11.2)

In Lisp there is a fixed amount of space allocated for the stack. When this space is exhausted, the **STACK OVERFLOW** error occurs. However, if the system waited until the stack were *really* exhausted, there wouldn't be room to run the debugger. Thus, a portion of the stack space is reserved; when the stack intrudes into the reserved area, it causes a stack overflow interrupt, and subsequently a call to the debugger.

In order not to get a **STACK OVERFLOW** error while inside the debugger, this intrusion into the reserved area is only noted once. If the reserved area is exhausted, then a "hard" stack overflow occurs (a 9319 MP halt), from which the only recourse is a hard reset via STOP (or Ctrl-D from TeleRaid). Following a hard reset, the stack is cleared, stack overflow detection is reenabled, and all processes are restarted.

The implications of this are that you should not attempt any deep computations while inside the debugger for a stack overflow error, and you should call **(HARDRESET)** as soon as possible in order that subsequent stack overflows can again be caught in the debugger before they advance to the MP halt. In order to make this more convenient, the system automatically calls **(HARDRESET)** if you exit the debugger via the **^** or **OK** commands, or abort with a Ctrl-D. The only way to exit the debugger without having a

(HARDRESET) occur is by using the **RETURN** command. You can disable this feature by setting **AUTOHARDRESETFLG** to **NIL**, in which case you must be sure to perform the **(HARDRESET)** yourself if you want the next stack overflow to be detected early enough to enter the debugger.

Section 11.2.1 Searching the Stack

(**STKPOS** *FRAMENAME N POS OLDPOS*) [Function]

(STKPOS ’STKPOS) does not cause an error; it merely returns NIL. (This was true in Koto as well.) It is still not permissible to create a pointer to the active frame; however, **STKPOS** never attempts to, as it starts searching for the specified frame by looking first at its caller.

Section 11.2.2 Variable Bindings in Stack Frames

(I:11.7)

(**STKARG** *N POS —*) [Function]

(**STKNARGS** *POS —*) [Function]

The functions **STKARG** and **STKNARGS** will now return the number of arguments supplied to a Lambda Nospread when there is a break. The **?=** command will show all the arguments.

(**SETSTKARGNAME** *N POS NAME*) [Function]

The function **SETSTKARGNAME** does not work for interpreted frames.

Section 11.2.5 Releasing and Reusing Stack Pointers

(**CLEARSTK** *FLG*) [Function]

(CLEARSTK NIL) is a no-op—the ability to clear all stack pointers is inconsistent with the modularity implicit in a multi-processing environment.

CLEARSTKLST [Variable]

NOCLEARSTKLST [Variable]

The variables CLEARSTKLST and NOCLEARSTKLST are no longer used. (More precisely, they are used only by the Old Interlisp Executive, which means that programs can no longer depend on them.)

Section 11.2.7 Other Stack Functions

(II:11.13)

In the **REALFRAMEP** function, the *INTERPFLG* argument description has been corrected to read:

If *INTERPFLG*=**T** returns **T** if *POS* is not a dummy frame. For example, if (**STKNAME** *POS*)**=COND**, (**REALFRAMEP** *POS*) is **NIL**, but (**REALFRAMEP** *POS* **T**) is **T**.

Chapter 12 Miscellaneous

Section 12.2 Idle Mode

The following properties in **IDLE.PROFILE** are new or have meanings different from the documentation in the *Interlisp-D Reference Manual*:

ALLOWED.LOGINS The authentication aspects of this property have been separated into the **AUTHENTICATE** property. The value of this property now speaks specifically to who is allowed to exit idle mode: If the value is **NIL** (or any other non-list), no login at all is required to exit Idle mode. Otherwise, the value is a list composed of any of the following:

***** Require login, but let anyone exit idle mode. This will overwrite the previous user's name and password each time idle mode is exited.

T Let the previous user (as determined by **USERNAME**) exit idle mode. If the user name has not been set, this is equivalent to ***.**

A user name Let this specific user exit idle mode.

A group name Allow any members of this group (an NS Clearinghouse group name) to exit idle mode.

The initial value for **ALLOWED.LOGINS** is **(T *)**, i.e., anyone is allowed to exit idle mode.

AUTHENTICATE The value of this property determines what mechanism is used to check passwords. If **T**, use the NS authentication protocol (requires the presence of an NS Authentication server accessible via the network). If **NIL**, do not check the password at all—accept any password. This is only particularly useful if **ALLOWED.LOGINS** contains ***.**

The initial value of **AUTHENTICATE** is **T**.

FORGET If this is the symbol **FIRST**, the user's password will be erased when idle mode is entered. Otherwise, this property is relevant only when **ALLOWED.LOGINS** is **NIL** (if **ALLOWED.LOGINS** is a list, then some sort of login is required, which will have the effect of erasing any previous login): If **FORGET** is non-**NIL**, the user's password will be erased when idle mode is exited. Initial value is **T** (erase password on exit).

Note: If the password is erased on *entry* to Idle mode (value **FIRST**), any program left running when idle mode is entered may fail if it tries doing anything requiring passwords (such as accessing file servers).

LOGIN.TIMEOUT Value is a number indicating how many seconds Idle's prompt for a login should remain up before timing it out and resuming Idle mode. Initial value is 30. This feature avoids the problem of having an Idle machine "freeze up" indefinitely (stop running the idle pattern) just because someone brushed against the keyboard.

RESETVARS This property is no longer used; rather, the value of the global variable **IDLE.RESETVARS** is used instead.

SUSPEND.PROCESS.NAMES

This property is no longer used; rather, the value of the global variable **IDLE.SUSPEND.PROCESS.NAMES** is used instead.

Section 12.3 Saving Virtual Memory State

AROUNDEXITFNS [Variable]

This variable provides a way to "advise" the system on cleanup and restoration activities to perform around **LOGOUT**, **SYSOUT**, **MAKESYS** and **SAVEVM**; it subsumes the functionality of **BEFORESYSOUTFORMS**, **AFTERLOGOUTFORMS**, etc. Its value is a list of functions (names) to call around every "exit" of the system. Each function is called with one argument, a symbol indicating which particular event is occurring:

BEFORELOGOUT

The system is about to perform a **LOGOUT**. The event function might want to save state, notify a network connection that it is about to go away, etc.

BEFORESYSOUT
BEFOREMAKESYS
BEFORESAVEVM

The system is about to perform a **SYSOUT**, **MAKESYS**, or **SAVEVM**. Often these three events are treated equivalently; however, sometimes the distinction is interesting. For example, a program might want to perform a much more extensive tidying-up before **MAKESYS** than if it is merely doing a routine **SAVEVM.**

AFTERLOGOUT
AFTERSYSOUT
AFTERMAKESYS
AFTERSAVEVM

The system is starting up a virtual memory image that was saved by performing a **LOGOUT**, **SYSOUT**, etc. Ordinarily, the event function should treat all of these the same—in all four cases, some arbitrary amount of time has passed, remote files may have come and gone, a different user may be logged in, or the virtual memory image might even be running on a different workstation.

AFTERDOSYSOUT
AFTERDOMAKESYS
AFTERDOSAVEVM

The system is continuing in the same virtual memory image following a **SYSOUT**, **MAKESYS**, or **SAVEVM** (as opposed to having just booted the same virtual memory image). Ordinarily, these events are uninteresting; they exist solely so that actions taken by the **BEFORExxx** events can be compensated for after the event. For example, if the before event cleared a cache, the after event might initiate refilling it. There is, of course, no event **AFTERDOLOGOUT**, as **LOGOUT** does not "continue".

Section 12.4 System Version Information

(I:12.13)

In the description of the **MACHINETYPE** function, add another machine, the **DOVE** (for the Xerox 1186).

VOLUME II—ENVIRONMENT

Chapter 13 Interlisp Executive

(I:23.37)

(**READLINE** *RDTBL* — —) [Function]

The *Interlisp-D Reference Manual* states:

The description on p 13.37 of **READLINE**'s behavior when one or more spaces precede the carriage return applies only when **LISPXREADFN** is **READ**. **LISPXREADFN** is initially **TTYINREAD**, which ignores spaces before the carriage return, and thus will never prompt you with "..." for an additional line. Also, the new Executive does not use **READLINE** at all, so you will never see this behavior in a new Executive, independent of the setting of **LISPXREADFN**.

Chapter 14 Errors and Breaks

Section 14.5 Break Window Variables

(II:14.15)

Setting the variable **BREAKREGIONSPEC** to NIL no longer creates problems if there is a subsequent break.

Section 14.8 Catching Errors

(II:14.22)

The Nlambda functions **ERSETQ** and **NLSETQ** now allow evaluation of an arbitrary number of forms, rather than only one.

(II:14.26)

For Medley, the Interlisp interpreter's handler for **RESETFORM** has been fixed (in Lyric, it worked only from the Common Lisp interpreter, or compiled) .

Chapter 17 File Package

Note: The File Package is now known as the File Manager.

Section 17.8.1 Functions for Manipulating Typed Definitions

(II:17.26)

(**HASDEF** *NAME TYPE SOURCE SPELLFLG*) [Function]

Clarification: **HASDEF** for type FNS (or NIL) indicates that *NAME* has an editable source definition, not that *NAME* is defined at all. Thus if *NAME* exists on a file for which you have loaded only the compiled version but not the source, **HASDEF** returns **NIL**.

Section 17.8.2 Defining New File Package Types

(II:17.31)

In the **WHENCHANGED** File Package Type Property the *REASON* argument passed to **WHENCHANGED** no longer is **T** or **NIL**. The Note has been revised as follows:

Note: The *REASON* argument passed to **WHENCHANGED** functions is either **CHANGED** or **DEFINED**.

(II:17.32)

The Nospread Function **FILEPKGTYPE** returns a property list rather than an alist.

Section 17.9.2 Variables

(II:17.36)

In the Lyric release, **HORRIBLEVARS** did not preserve common substructures across several variables.

In Lyric, you could not dump an **UGLYVARS** or **HORRIBLEVARS** whose printed representation required more than *ARRAY-TOTAL-SIZE-LIMIT* characters. This is no longer the case with the Medley release.

Section 17.9.8 Defining New File Package Commands

(II:17.47)

The Nospread Function **FILEPKGCOM** returns a property list rather than an alist.

Section 17.11 Symbolic File Format

(**PRETTYDEF** *PRTTYFNS PRTTYFILE PRTTYCOMS REPRINTFNS SOURCEFILE CHANGES*) [Function]

PRETTYDEF accepts only a symbol for its file argument.

In Lyric and Medley, SYSPRETTYFLG is ignored in Interlisp exec and does not pretty-print values in the executive.

(**LISPSOURCEFILEP** *FILE*) [Function]

LISPSOURCEFILEP is more specifically defined to mean that the file is in File Manager format *and* has a file map.

Section 17.11.3 File Maps

File maps are no longer stored on the FILEMAP property. See **GET-ENVIRONMENT-AND-FILEMAP** in the section entitled "Programmer's Interface to Reader Environments" in chapter 3.

Chapter 18 Compiler

CAUTION: Files compiled in Medley cannot be loaded back into Lyric. Medley-compiled .LCOM and .DFASL files will produce an error message when loaded into Lyric. (Lyric-compiled .LCOM and .DFASL files can be loaded and run in Medley.) If you need to run a Medley file in Lyric, load the source file and use the Lyric compiler.

Note that you should not attempt to compile a file containing a function named **STOP**. The format of the .LCOM file produced by **BCOMPL** or **TCOMPL** admits an unfortunate ambiguity in the treatment of the symbol **STOP**—**LOAD** prefers to treat it as the token signifying the end of the file, rather than as starting the definition of the function **STOP**.

There is no such restriction for the .DFASL files produced by **CL:COMPILE-FILE.**

Chapter 21 CLISP

Section 21.8 Miscellaneous Functions and Variables

(**CLEARCLISPARRAY** *NAME — —*) [*Function*]

Macro and CLISP expansions are cached in CLISPARRAY, the system's CLISP hash array. When anything changes that would invalidate an expansion, it needs to be removed from the cache. CLEARCLISPARRAY removes from the CLISP hash array any key whose CAR is *NAME*. The system does this automatically whenever you edit a clisp or macro form, or when you redefine a clisp word or macro definition. However, you may need to call CLEARCLISPARRAY explicitly if you change something in a more subtle way, e.g., you redefine a function used by a macro. If your change invalidates an unknown set of expansions, you might prefer to take the performance penalty of calling (CLRHASH CLISPARRAY) to invalidate the entire cache, just to make sure no incorrect expansions are kept around.

Chapter 22 Performance Issues

Section 22.1 Storage Allocation and Garbage Collection

The following should be appended to the description of garbage collection in Interlisp-D:

Another limitation of the reference-counting garbage collector is that the table in which reference counts are maintained is of fixed size. For typical Lisp objects that are pointed to from exactly one place (e.g., the individual conses in a list), no burden is placed on this table, since objects whose reference count is 1 are not explicitly represented in the table. However, large, "rich" data structures, with many interconnections, backward links, cross references, etc, can contribute many entries to the reference count table. For example, if you created a data structure that functioned as a doubly-linked list, such a structure would contribute an entry (reference count 2) for each element.

When the reference count table fills up, the garbage collector can no longer maintain consistent reference counts, so it stops doing so altogether. At this point, a window appears on the screen with the following message, and the debugger is entered:

```
Internal garbage collector tables have overflowed, due
to too many pointers with reference count greater than 1.
*** The garbage collector is now disabled. ***
Save your work and reload as soon as possible.
```

[This message is slightly misleading, in that it should say "count not equal to 1". In the current implementation, the garbage collection of a large pointer array whose elements are not otherwise pointed to can place a special burden on the table, as each element’s reference count simultaneously drops to zero and is thus added to the reference count table for the short period before the element is itself reclaimed.]

If you exit the debugger window (e.g., with the RETURN command), your computation can proceed; however, the garbage collector is no longer operating. Thus, your virtual memory will become cluttered with objects no longer accessible, and if you continue for long enough in the same virtual memory image you will eventually fill up the virtual memory backing store and grind to a halt.

Section 22.5 Using Data Types Instead of Records

(II:22.13)

The note in this section states that "pages for datatypes are allocated one page at a time." The note should read:

Space for datatypes is allocated two pages at a time. Thus, each datatype for which any instances at all have been allocated has at least two pages assigned to it.

Chapter 23 Processes

Section 23.1 Creating and Destroying Processes

(III:23.2)

ADD.PROCESS no longer coerces the process name to a symbol. Rather, process names are treated as case-insensitive strings. Thus, you can use strings for process names, and when typing process commands to an exec, you need not worry about getting the alphabetic case correct.

Section 23.2 Process Control Constructs

The Medley release fixes the **PROCESS.EVAL** and **PROCESS.APPLY** functions. In **PROCESS.EVAL** and **PROCESS.APPLY**, with argument *WAITFORRESULT* = **T**, if the computation in the other process aborts (or the process is killed), then **PROCESS.EVAL** and **PROCESS.APPLY** return **:ABORTED** instead of hanging.

Section 23.6 Typein and the TTY Process

BACKGROUNDFNS [Variable]

A list of functions to call "in the background". The system runs a process (called "BACKGROUND") whose sole task is to call each of the functions on the list **BACKGROUNDFNS** repeatedly. Each element is the name of a function of no arguments. This is a good place to put cheap background tasks that only do something once in a while and hence do not want to spend their own separate process on it. However, note that it is considered good citizenship for a background function with a time-consuming task to spawn a separate process to do it, so that the other background functions are not delayed.

TTYBACKGROUNDFNS [Variable]

This list is like **BACKGROUNDFNS**, but the functions are only called while in a tty input wait. That is, they always run in the tty process, and only when the user is not actively typing. For example, the flashing caret is implemented by a function on this list. Again, functions on this list should spend very little time (much less than a second), or else spawn a separate process.

Section 23.8 Process Status Window

The Medley release modifies the way in which the Process Status Window can be reshaped and refreshed.

The Process Status Window is now created in such a way that reshaping the window reshapes ONLY the backtrace window, not the main window.

The process status window now refreshes itself automatically following a KILL command.

VOLUME III—INPUT/OUTPUT

Chapter 24 Streams and Files

Section 24.7 File Attributes

(**GETFILEINFO** *FILE ATTRIB*) [Function]

NS file servers implement the following additional attributes for **GETFILEINFO** (neither of these attributes is currently settable with **SETFILEINFO**):

READER The name of the user who last read the file.

PROTECTION A list specifying the access rights to the file. Each element of the list is of the form (*name nametype* . *rights*), where *name* is the name of a user or group or a name pattern, and *rights* is one or more of the symbols ALL READ WRITE DELETE CREATE MODIFY. For servers running Services release 10.0 or later, *nametype* is the symbol "--"; in earlier releases it is either INDIVIDUAL or GROUP, to distinguish the type of name. For example, the value ((Jane Jones: -- ALL) (*: -- READ)) means that user Jane Jones has full access to the file, while all members of the default domain only have read access to the file.

Section 24.9 Local Hard Disk Device

(III:24.22)

In the Medley release, the {DSK} device now accepts a wider range of characters in file names. Almost any character in char set 0 is acceptable. Previously, if you tried to create a file whose name included, for example, an underscore, you would see a "FILE NOT FOUND" error.

Section 24.10 Floppy Disk Device

(III:24.26)

As of the Lyric release, CPM-format floppy disks are no longer supported.

Section 24.12 Temporary Files and CORE Device

(III:24.30)

In Medley, (**GETFILEINFO** xx 'LENGTH) works for both opened and closed **NODIRCORE** streams.

A closed **NODIRCORE** stream can be reopened.

Section 24.18.1 Pup File Server Protocols

UNIXFTPFLG [Variable]

When the Leaf protocol was first implemented for the Vax Unix operating system, its use was inconsistent with the operation of the Pup FTP server on the same host: the Leaf server supported versions, but the Ftp server knew only about the native, versionless file system. Thus, Lisp could not use the two protocols interchangeably. For example, if it used Ftp to write a file FOO, the Ftp server would, in versionless style, overwrite the versionless file FOO, rather than create a new version FOO;6 to supersede the highest version FOO;5 created by the Leaf server.

Lisp thus makes the conservative assumption that the Ftp server is unusable for anything other than directory enumeration on a host of type UNIX. This is unfortunate, since it is often the case that Ftp is more efficiently implemented than Leaf, since one need only tune the performance of sequential access.

More recent versions of the Unix Pup software have a Leaf and Ftp server more in agreement with each other. Setting **UNIXFTPFLG** to true (it is initially NIL) informs Lisp that all the Unix servers accessible on your internetwork that possess Ftp servers are safe to use in parallel with their Leaf servers.

Section 24.18.1 and 24.18.2 Use of BREAKCONNECTION with File Servers

(III:24.37)

In Medley, the function **BREAKCONNECTION** can be used equally well with NS servers and Leaf servers. Formerly, it only worked on Leaf servers, and there was a separate function (BREAK.NSFILING.CONNECTION *HOST*) to handle NS servers.

(BREAKCONNECTION *HOST FAST***)** [Function]

Breaks the file server connection to *HOST*. If *HOST* = T, breaks connections to all file servers that understand the **BREAKCONNECTION** method (currently Leaf and NS). **BREAKCONNECTION** returns the server name, or if *HOST* = T, returns a list of all hosts that responded to the **BREAKCONNECTION** request.

This function may be useful if Lisp and the server disagree about what files are open, or if the Lisp system is caching something that you do not want it to; e.g., if you get a file busy error from another workstation for a file that you may have touched on this workstation.

The behavior of **BREAKCONNECTION** is server-specific. On an NS server, **BREAKCONNECTION** releases any locks that Lisp may have on recently-accessed files, including those for open files, but does not close any files from Lisp's point of view--any subsequent access to an open file will quietly reestablish the connection. Most NS servers have a short timeout on the order of 10 minutes after which an implicit **BREAKCONNECTION** occurs if you have no files open.

On a Leaf server, **BREAKCONNECTION** first closes any files. If the argument *FAST* is true, it marks the files closed without attempting to close them cleanly. Leaf connections ordinarily do not timeout if any files at all are open.

Section 24.18.2 NS File Server Protocols

(III:24.37)

Medley incorporates the random access capability on NS servers provided by the NSRANDOM LispUsers module in Lyric.

The Medley release also supports NS file names containing characters other than character set 0 (e.g., Greek characters).

Section 24.18.3 Operating System Designations

DEFAULT.OSTYPE [Variable]

If a host's name is not found in **NETWORKOSTYPES**, its operating system type is assumed to be the value of **DEFAULT.OSTYPE**. This variable may be of use to sites with many servers all of the same type. Its default value (IFS) is, unfortunately, inappropriate for most sites. It is recommended you set **DEFAULT.OSTYPE** in the initialization file that lives on the local disk (*not* in an init file on a file server, since Lisp needs to know the operating system type before talking to the server).

Chapter 25 Input/Output Functions

Section 25.2 Input Functions

(**LASTC** *FILE*) [Function]

The function **LASTC** can return an incorrect result when called immediately following a **PEEKC** on a file that contains run-coded NS characters.

Section 25.3.2 Printing Numbers

(III:25.15)

In the **PRINTNUM** function, the **FLOAT** format option **(FLOAT 7 2 NIL T)** is illegal; change the option to **(FLOAT 7 2 NIL 0)**.

Section 25.3.4 Printing Unusual Data Structures

(**HPRINT** *EXPR FILE UNCIRCULAR DATATYPESEEN*) [Function]

Using **HPRINT** to save structures that include pointers to raw storage will cause stack overflows. This includes dumping things using the **VARS**, **UGLYVARS**, or **HORRIBLEVARS** filemanager commands.

For example, a font descriptor points to raw storage, and cannot be dumped; for that reason, other system data types (e.g. windows) that point to fonts also cannot be dumped.

Section 25.4 Random Access File Operations

(III:25.20)

The first argument in the **FILEPOS** function should be called *STR* not *PATTERN*.

(III:25.20)

In the Medley release, the function **COPYBYTES** now accepts *START* and *END* arguments even when the input stream is not random access. This caused an error in earlier releases.

Section 25.6 PRINTOUT

(III:25.27)

The PRINTOUT command **.FONT** changes the **DSPFONT** font permanently, that is, even after printout finishes.

Section 25.8.3 READ Macros

(III:25.42-43)

These **READMACROS** appear only in the OLD-INTERLISP-T readtable. (See Section 2 for a description of Lyric readtables.)

Chapter 26 User Input/Output Packages

Section 26.3 ASKUSER

(**ASKUSER** *WAIT DEFAULT MESS KEYLST TYPEAHEAD LISPXPRNTFLG OPTIONSLST FILE*) [Function]

ASKUSER does not accept a string to mean a stream open on the string; you must call **OPENSTRINGSTREAM** if that's what you mean.

Section 26.4 TTYIN Display Typein Editor

(III:26.22)

The following fixes have been made to TTYIN in the Medley release:

- TTYIN now respects the **DSPLEFTMARGIN** of the ttydisplaystream, rather than assuming it is zero.
- You can now assign the keyaction 194 (octal 302--acute accent in the NS character set) to a key and TTYIN will not treat it like the UNDO key (except on the 1132, where this functionality is still on blank-middle).

- TTYIN correctly handles prompts that are wider than the window.
- TTYIN now handles NS characters correctly when you are using a fixed-width font into which you have coerced, say, Classic characters for the non-zero character sets.
- TTYIN now handles Escape completion much more efficiently. If the completion is ambiguous, it completes the unambiguous prefix (as it did in Koto but not Lyric); it also correctly interprets escape characters. For example, in an exec with Common Lisp readtable, it correctly completes symbols that start with \\, or a mixed-case symbol written with vertical bars. Also, Escape completion computes character widths correctly when it lowercases an upper case string, rather than leave some garbage bits on the display.
- The off-by-the-descent bug wherein TTYIN sometimes left stray bits at the bottom of the window has been fixed.

Section 26.4.3 Display Editing Commands

(III:26.25)

?= and Meta-P no longer hang if you had an unbalanced string quote in the input.

?=, Meta-P, and the **FIX** command now work correctly when there are NS characters in the input.

The printout for **?=** is now improved; it respects *print-case*, matches up keywords better, and prints abstract syntax descriptions (such as for cl:do) a bit more clearly.

SMARTARGLIST fetches the argument lists of cl:compiled functions, so **?=** now works in more cases.

The Ctrl-X command, when the caret is already positioned at the end of the input and everything but parentheses are balanced (i.e., no unbalanced string quotes or vertical bars), types as many closing parentheses as necessary to complete the input and then returns, much as if you had typed right bracket ("]") in Interlisp. Thus, if the cursor is somewhere in the middle of the input, typing two Ctrl-X’s is sufficient to complete (assuming all you needed to type were some more parens).

TTYIN can now be used as a substitute for PROMPTFORWORD. The new function TTYINPROMPTFORWORD takes the same set of arguments as PROMPTFORWORD. In the most common cases it then calls TTYIN in "promptforword" mode, so that you can use the mouse and other TTYIN commands on the input. For cases it can’t handle, it calls the old PROMPTFORWORD. These cases are: DONTECHOTYPEIN.FLG or KEYBD.CHANNEL is non-NIL; ECHO.CHANNEL is not a displaystream; or TERMINCHARS.LST contains a character other than cr, space or ^X and you have set the variable TTYIN.USE.EXACT.CHARS (initially NIL) to T. TTYIN saves the old definition of PROMPTFORWORD, so you can either have your program explicitly call TTYINPROMPTFORWORD instead of PROMPTFORWORD, or you can have all calls to

PROMPTFORWORD changed by doing a (MOVD 'TTYINPROMPTFORWORD 'PROMPTFORWORD).

Section 26.4.5 Useful Macros

(III:26.29)

CTRLUFLG is no longer supported by default. To use this feature, turn it on explicitly: **(INTERRUPTCHAR (CHARCODE ^U) 'CTRLUFLG)**.

Chapter 27 Graphic Output Operations

Section 27.1.3 Bitmaps

Note: The printed representation of bitmaps has changed. Please see release notes Chapter 3, Integration of Interlisp-D/ Common Lisp, "Bitmap Syntax."

(III:27.4)

The following function has been added to Bitmap Operations between the functions **EXPANDBITMAP** and **SHRINKBITMAP**:

(**ROTATE-BITMAP** *BITMAP*) [Function]

Given an m-high by n-wide bitmap, this function returns an n-high by m-wide bitmap. The returned bitmap is the image of the original bitmap, rotated 90 degrees clockwise.

(III:27.4)

In the Medley release, the **EDITBM** function is substantially faster with the inclusion of **FASTEDITBM** (a former LispUsers module) in the sysout.

Section 27.3 Accessing Image Stream Fields

The following functions were not documented in the Koto release of the *Interlisp-D Reference Manual*:

(**DSPCLEOL** *XPOS YPOS HEIGHT*) [Function]

"Clear to end of line". Clears a region from (*XPOS,YPOS*) to the right margin of the display, with a height of *HEIGHT*. If *XPOS* and *YPOS* are **NIL**, clears the remainder of the current display line, using the height of the current font.

(**DSPRUBOUTCHAR** *DS CHAR X Y TTBL*) [Function]

Backs up over character code *CHAR* in the display stream *DS*, erasing it. If *X*, *Y* are supplied, the rubbing out starts from the position specified. **DSPRUBOUTCHAR** assumes *CHAR* was printed with the terminal table *TTBL*, so it knows to handle control characters, etc. *TTBL* defaults to the primary terminal table.

Section 27.6 Drawing Lines

(III:27.17)

The non-**NIL** value of the *DASHING* argument of **DRAWLINE** uses LINEWITHBRUSH. LINEWITHBRUSH is a width-by-width brush which draws then lifts.

In the Medley release, when using the color argument, Interpress **DRAWLINE** treats 16x16 bitmaps or negative numbers as shades/textures. Positive numbers continue to refer to color maps, and so cannot be used as textures. To convert an integer shade into a negative number use NEGSHADE (e.g. (NEGSHADE 42495) is -23041).

(III:27.18)

The **RELDRAWTO** function has been corrected so that it no longer draws a spot if the *DX* and *DY* arguments are 0.

Section 27.7 Drawing Curves

(III:27.18)

For the brush width value of **NIL,** the previous default value **(ROUND 1)** has been changed. The default value for the brush width value **NIL** is the **DSPSCALE** of the stream (that is, 1 printer's point wide).

(III:27.19)

A new image stream function, **DRAWARC**, follows **DRAWCIRCLE** in the *InterLisp-D Reference Manual*.

(**DRAWARC** *CENTERX CENTERY RADIUS STARTANGLE NDEGREES BRUSH DASHINGSTREAM*) [Function]

Draws an arc of the circle whose center point is (*CENTERX CENTERY*) and whose radius is *RADIUS* from the position at *STARTANGLE* degrees for *NDEGREES* number of degrees. If *STARTANGLE* is 0, the starting point will be (*CENTERX (CENTERY + RADIUS)*). If *NDEGREES* is positive, the arc will be counterclockwise. If *NDEGREES* is negative, the arc will be clockwise. The other arguments are interpreted as described in **DRAWCIRCLE**.

Section 27.8 Miscellaneous Drawing and Printing Operations

(III:27.20)

To have a filled polygon print correctly, set the global variable **PRINTSERVICE** to floating point value 9.0 for printers running Services 9.0 or later.

When using **FILLPOLYGON** to be sent to Xerox 8044 Interpress printers, the global variable **PRINTSERVICE** must be set to the same value as the Print Service installed on your printer, currently either 8.0, 9.0 or 10.0. Thus, if your printer is running Print Service 9.0, you must set the global variable **PRINTSERVICE** to the floating point value 9.0. This works around an incompatible change in the Xerox 8044 Interpress implementation.

In Medley, Interpress curves are now rendered at a lower accuracy, allowing faster hardcopy. The spline is now rendered at 1/150 inch; in Lyric it was 1/300 inch.

The following function was omitted from previous version of the *Interlisp-D Reference Manual*:

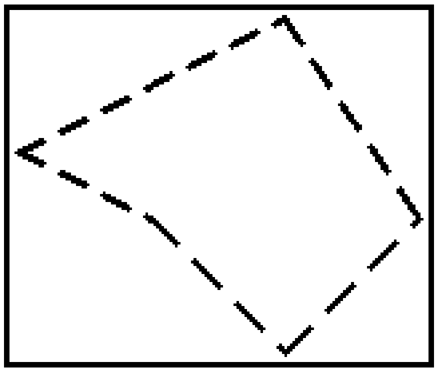
(**DRAWPOLYGON** *POINTS CLOSED BRUSH DASHING STREAM*) [Function]

Draws a polygon on the image stream *STREAM*. *POINTS* is a list of positions to which the figure will be fitted (the vertices of the polygon). If *CLOSED* is non-NIL, then the starting position is specified only once in *POINTS*. If *CLOSED* is NIL, then the starting vertex must be specified twice in *POINTS*. *BRUSH* and *DASHING* are interpreted as described in Chapter 27 of the *Interlisp-D Reference Manual*.

For example,

```
(DRAWPOLYGON '((100 . 100) (50 . 125)
                 (150 . 175) (200 . 100) (150 . 50))
             T '(ROUND 3) '(4 2) XX)
```

would draw a polygon like the following on the display stream XX.



(III:27.20)

The function **FILLPOLYGON** contains two new arguments, *OPERATION* and *WINDNUMBER*. The new form for the function, and definitions for added arguments, follow.

(**FILLPOLYGON** *POINTS TEXTURE OPERATION WINDNUMBER STREAM*) [Function]

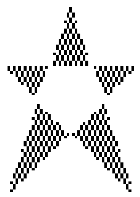
OPERATION is the **BITBLT** operation (see page 27.15 in the *Interlisp-D Reference Manual*) used to fill the polygon. If the *OPERATION* is **NIL**, the *OPERATION* defaults to the *STREAM* default *OPERATION*.

WINDNUMBER is the number for the winding rule convention . This number is either 0 or 1; 0 indicates the "zero" winding rule, 1 indicates the "odd" winding rule.

When filling a polygon, there is more than one way of dealing with the situation where two polygon sides intersect, or one polygon is fully inside the other. Currently, **FILLPOLYGON** to a display stream uses the "odd" winding rule, which means that intersecting polygon sides define areas that are filled or not filled somewhat like a checkerboard. For example,

```
(FILLPOLYGON '((125 . 125) (150 . 200) (175 . 125)
               (125 . 175) (175 . 175))
             GRAYSHADE WINDOW)
```

would produce a display something like this:



This fill convention also takes into account all polygons in *POINTS*, if it specifies multiple polygons.

Section 27.12 Fonts

A revised set of font printing metrics is a part of the Lyric release of Lisp. Note that Koto font files are still available to users who request them.

With the revised font set the interline spacing (line leading) is now consistent across all fonts within a point size. Previously, text with multiple fonts (but with the same point size, i.e., if a word were made bold or italic, or if the family were changed) would have different leading on different lines. The new .WD files clean up document appearance.

Note that these printer metric changes affect only hardcopy, not the display. The contents of the display fonts are essentially unchanged in Lyric.

Generally, line leading in the Lyric font files is tighter than in previous releases of the fonts. The default line leading is now the same as the font's nominal point size. As a consequence of the above, any text file (one not already formatted for Interpress) which is printed after installation of the new fonts will be formatted to a different length. This means that decisions regarding TEdit line leading, widows and orphans, left/right pages, references to page numbers, etc. will need to change. Koto documentation produced by users may need to be reformatted with different line leading, using the new fonts.

All of the font files now have a new naming scheme, which allows **FONTSAVAILABLE** to be able to do more accurate pattern matching. For example, the display font file for modern 8 bold italics used to be named:

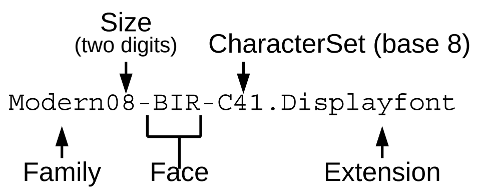
```
Modern8-B-I-C41.Displayfont
```

The file is now named:

```
Modern08-BIR-C41.Displayfont
```

In general font files use the following format:

The family name (e.g., `Modern`); a two digit size (e.g., `08`); a three letter Face (e.g., `BIR`, for Bold Italic Regular); the letter C followed by the font's character set in base 8 (e.g., `C41`); and finally an extension (e.g., `Displayfont`).



The old file naming convention is still supported, however, with the exception of the old Strike file naming convention. In Lyric, **FONTCREATE** will first search for fonts using the new font naming convention, and if the desired font is not found it will search using the Koto convention.

Compatibility considerations You can continue using the old printer metrics (.WD files) in Lyric, thus preserving document looks between Koto and Lyric. If you choose to do so, it is recommended that you rename your old .WD files to the new naming scheme (see above), so that you benefit from the changes to the font searching mechanisms. However, we strongly urge you to use the new .WD files. Otherwise, if you exchange TEdit documents with a site that is using the new files, the documents will print differently at the two sites. The creation date, rather than the naming convention, determines whether a .WD file represents the old or new format.

If, after installing the new .WD files, you wish to print a document using the old Koto formatting, make the font variable **INTERPRESSFONTDIRECTORIES** point to a directory containing the Koto font files. Also any Lyric printer font file information must be uncached from the sysout. To uncache the fonts, perform

```
(for INFO in (FONTSAVAILABLE '* '* '* '*
                    'INTERPRESS)
   do (APPLY 'SETFONTDESCRIPTOR INFO))
```

(III:27.30)

(**STRINGWIDTH** *STR FONT FLG RDTBL*) [Function]

In Lyric **STRINGWIDTH** observes ***PRINT-LEVEL*** and ***PRINT-LENGTH***.

In Medley, STRINGWIDTH with a NIL argument no longer returns the string width of the string with *STANDARD-OUTPUT* font. It now uses DEFAULTFONT.

Some new font manipulation functions have been added to Lisp. They are:

(**WRITESTRIKEFONTFILE** *FONT CHARSET FILENAME*) [Function]

Takes a display font font descriptor and a character set number, and writes that character set into a file suitable for reading in again. Note that the font descriptor's current state is used (which was perhaps modified by INSPECTing the datum), so this provides a mechanism for creating/modifying new fonts.

For example:

```
(WRITESTRIKEFONTFILE (FONTCREATE 'GACHA 10) 0
    '{DSK}Magic10-MRR-C0.DISPLAYFONT)
```

writes a font file which is identical in appearance to the current state of Gacha 10 charset 0.

If your DISPLAYFONTDIRECTORIES includes {DSK}, then a subsequent (FONTCREATE ’MAGIC 10) will create a new font descriptor whose appearance is the same as the old Gacha font descriptor.

However, the new font is identical to the old one in appearance only. The individual datatype fields and bitmap may not be the same as those in the old font descriptor, due to peculiarities of different font file formats.

Section 27.13 Font Files and Font Directories

(III:27.31)

Press fonts are not part of the sysout since PRESS is now a Library module.

Section 27.14 Font Classes

(III:27.32-27.48)

This section has been expunged from the *InterLisp-D Reference Manual*. Renumber the sections which followed the old Section 27.14 as

SECTION 27.15 ⇒ SECTION 27.14 Font Profiles

SECTION 27.16 ⇒ SECTION 27.15 Image Objects

SECTION 27.17 ⇒ SECTION 27.16 Implementation of Image Streams

Section 27.14 Font Profiles

(III:27.34)

The variable **FONTCHANGEFLG** has an additional value, **ALL**. **FONTCHANGEFLG=ALL** indicates that all calls to **CHANGEFONT** are executed.

(III:27.33-34)

The function **FONTNAME** no longer exists. This function was previously used in Interlisp-D to collect the names and values of variables on **FONTDEFSVARS**. The variable **FONTDEFSVARS** is no longer used; it was appropriate when most output devices were fixed-pitch, "line-printer" style devices, but is not suitable for use when most output devices are laser printers.

Chapter 28 Windows and Menus

Section 28.4 Windows

(III:28.13, 28.38)

The **ADDMENU** function will change a window's **RESHAPEFN** and also will change the window's **REPAINTFN**.

Section 28.4.5 Reshaping Windows

(III:28.17)

The Lisp window system allows the following minimum window sizes:

When creating a new window, the width and height specified must be at least 9, or else you will get an error "region too small to use as a window"

When reshaping a window, the smallest shape you can get is width = 26 and height = height of the font to be used in the window. If you specify a smaller region, **SHAPEW** will simply adjust it to fit these limits.

Section 28.4.8 Shrinking Windows Into Icons

(III:28.22)

SHRINKFN [Window property]

In previous releases, there was a bug in the attached window system such that if an attached window had a **SHRINKFN** of the single symbol DON'T, attempting to shrink the window resulted in a break with the message "UNDEFINED FUNCTION DON'T." For this case in Lyric, all windows that can be shrunk will be, while those windows with a **SHRINKFN** of the symbol DON'T will be left open.

To facilitate the management of window regions, the window property **EXPANDREGIONFN** has been added to Lisp. This feature allows applications to arrange for reshaping a window when it is expanded.

EXPANDREGIONFN [Window property]

EXPANDREGIONFN, if non-**NIL**, should be the function to be called (with the window as its argument) before the window is actually expanded.

The **EXPANDREGIONFN** must return **NIL** or a valid region, and must not do any window operations (e.g., redisplaying). If