F.G.H.
4/9/85

LispCourse #20 : Living in the Network World ž Part 2

Completions and Corrections

FTP Server package

There is a Lisp Library package that allows a Lisp machine to act as a server for PUP FTP.

Load {eris}<lisp>harmony>library>ftpserver.dcom and run eval the function (FTPSERVER). This will start up a FTP server process that runs in the background.

Once the server is running you can copy files back and forth to your machine's local disk from another machine running Interlisp (or running Alto FTP).

For example, if I start up FTPSERVER on my machine, I can go over to ISLPool DLion and do a *(COPYFILE '{DSK}<LISPFILES>INIT '{HALASZ}INIT)* to copy an INIT file from the DLion local disk to the local disk on my machine (i.e.., the Dorado named Halasz). I could also have refered to my machine by number as in *(COPYFILE '{DSK}<LISPFILES>INIT '{6#60#}INIT)*.

Copying works both ways: *(COPYFILE '{HALASZ}INIT '{DSK}<LISPFILES>INIT)* would work just as well.

You can also do a DIR and run the FILEBROWSER to a machine running the Interlisp FTP sever. For example, if my Dorado is running FTPSERVER then I can do a (FB {HALASZ}) or (FB {6#60#}) from the ISL Dlion to get a listing of the files on my Dorado.

Documentation is on {eris}<lisp>harmony>library>ftpserver.tedit (& .press).

Note on the use of the word *host.*

In the Interlisp documentation, the word *"host"* is often used to refer to a machine. A *host name* is what we refer to here as a *machine name*. For example, Phylum is refered to as a host and "Phylum" as a host name. The distinction between machines and hosts is non-existent.

The NS World

Machine Addresses

Machine addressess seem to be less important (from the user's point of view) in the NS world than in the PUP world.

Each machine on the NS network has an address: a 32-bit (i.e., between 0 and a very big number) network number and a 48-bit machine number. As in the PUP world, both the network number and the machine number are followed by a *"#"*. When printed out, the machine number prints as three 16-bit quantities separated by periods.

To find the address of your machine, evaluate the variable **\MY.NSADDRESS**. To find the address of some other machine from its name (see below for NS name conventions) use the function **LOOKUP.NS.SERVER** as in *(LOOKUP.NS.SERVER 'Phylex:),* which returns *204#0.125000.20217#.*

Examples:

Phylex:'s address is

StarFile:'s address is 131#0.125000.24314#

PaperMate:'s address is 142#0.125000.12122#

ISLPoolDlion's address is 142#0.125000.32462#

Halasz's Dorado address is 6#0.52612.100312#

Note: The NS network number is, by convention, the same as the PUP network number for the same physical network. For example: the ISLPoolDlion is known as **204#**36# in the PUP world and **204#**0.125000.32462# in the NS world.

Moreover, Phlyex: is on the same physical network (i.e., cable) as the ISLPoolDLion. Phylex: is known as **204#**0.125000.20217# in the NS world. Phylex: has no PUP address because it does not speak PUP.

NS Names: Machines and people are part of a common name space

Basic Concepts: Objects, Domains, and Organizations

In the PUP world, the naming of machines is separate from the naming of people (or groups of people). Machines are named using a litatom

processed by a Name server. People are named using a Grapevine name and password processed by a Grapevine authetication server.

For example:

PHYLUM and *ERIS* refer to machines

Halasz.pa and *Feuerman.pasa* refer to people

LispUsers^.pa and *NoteCardsInfo^.pasa* refer to groups of people.

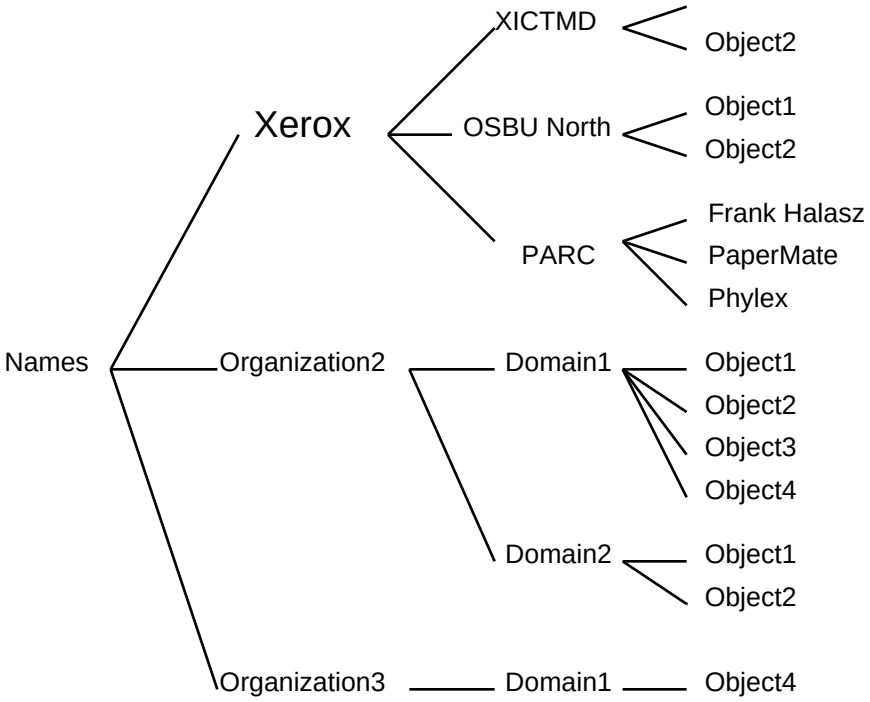
In the NS world all objects, including machines, people, and groups, are named in the same way, i.e., according to the NS naming standard.

The NS naming standard is hierarchical and works as follows:

Every object has an *object name*. An object is a machine, a person, a group, or any other "thing" in the NS world.

Every object also belongs to a particular *domain*, which has a *domain name.* There are always many objects within a given domain.

Every domain belongs to an *organization*, which has an *organization name*. There are generally many domains within an organization.



An *organization* is intended to refer to a single company or major division within a company. For example: all of Xerox is considered a single organization.

A *domain* is intended to be a small division of an organization that shares common resources such as file servers and printers. A domain might be anywhere from large facilties like PARC (with 160 people) to small projects like WBST129UL (with 25 people).

Unlike Grapevine registries (e.g., *.pa*), a domain is not intended to specifically refer to geographic location. Often, however, domains are used to divide an organization into organizational/geographic groups.

Examples from within Xerox:

PARC is a domain

OSBU North is one domain.

OSBU South is another domain.

OSBU Bayshore is a third domain.

The Webster, N.Y. facility has 20 or so different domains corresponding to different organizational groups located in

Webster. Examples: *WBST128, WBST129, WBST129UL,* and *WRC*.

NS Names and Aliases; Properties of Objects

An object's NS name consists of an object name, a domain name, and an organization name separated by *":"*'s.

Examples:

Frank Halasz:PARC:Xerox

Phylex:PARC:Xerox

Pluto:OSBU North:Xerox

Bill Liles:XSIS:Xerox

Within a domain, an object may have several alternative object names known as ***aliases***. An object's alias can always be used in place of its real object name to refer to the object.

Examples:

Within PARC:Xerox

Halasz: is an alias for *Frank Halasz:*

aifile: is an alias for *Phylex:*

print: is an alias of *LispPrint:*

Thus:

Frank Halasz:PARC:Xerox and *Halasz:PARC:Xerox* refer to the same object. So do *Phylex:PARC:Xerox* and *aifile:PARC:Xerox*

Every named object in the NS world has a bunch of properties that describe the object. For example, every object name has an address, either a NS network address or a NS mail server address. For the most part these properties are not accessible to the ordinary user.

However, every object has a property that tells what type of object it is. For example: people have a property called USER. FileServers have a property called FILE.SERVICE.

The possible properties that define types of objects are:

USER ž a person

PRINT.SERVICE ž a print server

FILE.SERVICE ž a file server

MEMBERS ž a distribution list

These properties come in handy when you want to know all the file servers in PARC or all the people in the OSBU North domain. (See below for how to do these querires).

Default Domain and Organization

In Lisp, you can leave the domain and organization parts off of any NS name. If you do so, the values of the atoms **CH.DEFAULT.DOMAIN** and **CH.DEFAULT.ORGANIZATION** will be used for the domain and organization parts of a name whereever necessary.

Usually, CH.DEFAULT.DOMAIN and CH.DEFAULT.ORGANIZATION are set in your system INIT file. You can change them in your personal INIT file or by SETQing them in the Lisp Exec if you refer to domains and/or organizations different than the ones set in the system INIT file.

At Parc, CH.DEFAULT.DOMAIN is set to *PARC* and CH.DEFAULT.ORGANIZATION is set to *Xerox.*

Examples: Given the standard settings for PARC,

Phylex: refers to *Phylex:PARC:Xerox*

StarFile: refers to *StarFile:PARC:Xerox*

Tundra:OSBU North: refers to *Tundra:OSBU North:Xerox*

Clearinghouses: Name and Authentication Servers for NS World

Introduction

In the NS world, a server called a ***Clearinghouse*** carries out the functions of both the PUP Name server (machine name to machine address translation) and the Grapevine authentication service (checking peoples names/passwords and distribution list maintenance).

Every domain has one or more Clearinghouses.

Every Clearinghouse knows the names and addresses of the Clearinghouses for every domain and organization on the NS internet.

Every so often (each night or every other night), all the Clearinginghouses on the network exchange information about who they are and what domains they cover, etc. Thus, every Clearinghouse has reasonably up-to-date information about all other Clearinghouses.

Clearinghouse Services

Name Service

Every time you type in an NS name, your D-machine interrogates its Clearinghouse server for the translation for the name. For example, if you type in *DIR {Phylex:Parc:Xerox}<halasz>**, your machine will interrogate its Clearinghouse for the address of Phylex:Parc:Xerox.

If the named object is not in the same domain as your Clearinghouse, then your Clearinghouse tells your machine the appropriate Clearinghouse to contact for the domain of the object. Your machine then queries that Clearinghouse to get the address of the named object.

Authentication Service

Similarly, if you attempt to login to a NS file server (which is done every time you access it, even for a DIR), the file server will ask its Clearinghouse about your NS name and password. If you are registered on that Clearinghouse, you will be okayed. Otherwise, that Clearinghouse will give the file server the location of the Clearinghouse for your domain and the file server will check with your Clearinghouse.

For example, if I log into *Tundra:Osbu North:Xerox*, Tundra gets the location of the PARC Clearinghouse (since my name is Halasz:*PARC*:Xerox) from the Osbu North Clearinghouse. Tundra then checks the name and password with the PARC Clearinghouse . If I am registered on the PARC Clearinghouse, then I am allowed to login on Tundra.

Note: The NS world and the PUP/Grapevine world are totally separate. You must be registered on a Clearinghouse somewhere to access NS servers of any type. A Grapevine user name and password is not enough!!

Distribution Lists

Clearinghouses also maintain distribution lists for the NS mail system. Unfortunately, there is no interface like MAINTAIN whereby normal users can make changes to these distribution lists. All changes to the lists on a given Clearinghouse must be done by the administrator of that Clearinghouse. (At PARC, Carol Lehner is the Clearinghouse administrator).

Your Clearinghouse and the set of known Clearinghouses

Your Clearinghouse

The previous descriptions refer to "your Clearinghouse". In the NS world, each D-machine has to discover for itself the location of its Clearinghouse (i.e., the nearest Clearinghouse that is up). This works as follows:

The first time you refer to an NS object (e.g., an NS file server or an NS printer) your D-machine goes out and tries to find the nearest Clearinghouse server to it. It does this by looking around in turn on each of the various networks it knows about until it finds a Clearinghouse.

While its looking, you will see messages of the form *Looking for Clearinghouse servers on net 132* appearing in your Prompt window. Each such message represents one net that your machine is looking for Clearinghouses on. When a Clearinghouse has been found, it prints out something like *Noting Clearinghouse 131#0.125000.77654* in the Prompt window.

The parameter ***CH.NET.HINT*** determines the network on which your machine begins its search for a Clearinghouse. It should be set in your system or personal INIT files. If CH.NET.HINT is set to correct network number, the search for a Clearinghouse will be efficient (providing your Clearinghouse is up). If CH.NET.HINT is incorrectly set, your machine may look on many, many nets before it finds a Clearinghouse. At PARC, CH.NET.HINT should be set to 89.

The function call **(START.CLEARINGHOUSE T)** will carry out a search for the nearest Clearinghouse and set the appropriate variables to make it "your" Clearinghouse. this function is done automatically for you whenever you make your first NS reference. However, if your Clearinghouse goes down you may have to manually reinitiate the search using START.CLEARINGHOUSE.

The set of known Clearinghouses

Every time you refer to an NS domain other than your own, your machine asks your Clearinghouse for the location of the Clearinghouse for that domain. It then tries to make contact with that Clearinghouse.

While contact is being made a message like *Finding Clearinghouse server for WBST129* is displayed in your Prompt window. When it finds the Clearinghouse the

message *Noting Clearinghouse 204#0.125000.23452* will be printed.

Your machine then remembers the location of that domain/Clearinghouse pair so that the next time you access the domain, it will know what Clearinghouse to contact.

Example: If I type *DIR {Tundra:OSBU% North},* my machine firsts makes contact with the OSBU North Clearinghouse printing the appropriate messages in the Prompt window. It then queries that Clearinghouse for the address of Tundra.

The function call **(SHOW.CLEARINGHOUSE)** will print a graph of all of the Clearinghouses that your machine knows about. It prompts for a region for the graph window.

Accessing the Clearinghouse Information from Interlisp

Interlisp allows provides several functions that allow you to query the information on the Clearinghouse network. These function allow you, for example, to find the names of all the NS print servers in PARC.

The functions are the following:

(CH.LIST.ORGANIZATIONS *ObjectPattern*) ž returns a list of all of the organizations on the NS Internet whose organization name matches the *OrganizationPattern*. *OrganizationPattern* can include the * wildcard character, standing for 0 or more of any characters. If *OrganizationPattern* is NIL or just *, then the list returned will contain all of the organizations known.

Examples:

10_(CH.LIST.ORGANIZATIONS)

("..." "a long organization." "aflc" "AIL" "airgate" "Allen-Bradley Co." "Boeing" "Comm Test" "Conserve" "C˙Servers" "CTMD" "Demo" "DPI" "Earth" "Fuji Xerox" "G/C" "GDP" "Leesburg" "Lou OPD" "MANKKAA"

"MATRA" "NASA/JSC" "Nash" "Ois-l" "org" "R&D" "Rx" "RX IT" "RX Sweden" "RX-N" "RX/Denmark" "Rxa" "RXCH" "RXDK" "RXEG" "RXF" "RXFINLAND" "Rxg" "rxhh" "rxhol" "rxihq" "Rxihq Tsd" "rxn" "RXS" "RXSF" "RXSweden" "rxtsd" "RXUK" "Sanyo Kiko" "SBD" "SIEMENS" "SiemensAG" "SMD" "TA" "veroh" "Versatec" "Xci" "XCSS" "Xerox" "Xerox OSD" "Xerox Visitors" "Xopd" "XOS" "xosm") 11_(CH.LIST.ORGANIZATIONS ’Xerox*]

("Xerox" "Xerox OSD" "Xerox Visitors")

(CH.LIST.DOMAINS *DomainPattern*) ž returns a list of all of the domains whose domain name matches the *DomainPattern*. *DomainPattern* consists of the domain and organization parts of an NS name. The domain part (but not the organization part) can include the * wildcard character, standing for 0 or more of any characters. If the organization part is left off, the default organization (i.e., the value of CH.DEFAULT.ORGANIZATION) is used. If *DomainPattern* is NIL or just *, then the list returned will be a list of all domains in the default organization.

Examples:

```
20_(CH.LIST.DOMAINS "*:")
("A&E1" "A&E2" "A&E3" "AAAI" "Albany"
"Albuquerque" "AlphaMesa" "Alphaservices-Es"
"Alphaservices-Pa" "AlphaServices-PTL"
"Alphaservices-Rx" "alphaservoices-rx" "ATL-
ECE" "Atlanta" "Austin" "BIRMINGHAM" "Bones"
"Brookriver" "CIN OPS" "CrashTest" "ctmd"
"dagobah" "DallasInfomart" "DEMO ROOM"
"detroit" "DlosCE" "DlosEtron" "DLOSL200"
"DlosL300" "DlosLC" "DlosLV" "DlosLV-Comm"
"DlosMBT" "DlosNSC" "DlosSkeff" "Ece-Hq" "ECE-
WASH" "ecewest" "EDGE-Net" "EDNPS Test" "El
Segundo" "´el`e¸c`at`uˆo¨u" "Es A&E1" "ES A&E1 CHS"
"ES A&E2" "ES A&E3" "ES EDNPS Test" "ES GSD"
"ES GSD/WCO" "eS M1" "ES M4" "ES PSI Test" "ES
XC OST" "ES XC Pri" "ES XC XDMS" "ES XC16"
"Evaluation" "FAX-TEST" "GGW-Test" "Grnch"
"Harb" "HENR801A" "HENR801B" "HENR801C"
"HENR801E" "Henr801F" "HENR868" "Houston" "HUB
TEST" "HWDENGDlos" "ISD-NAM" "lac" "LAS VEGAS"
"loop ECE" "LSBG-ECE" "M1 ES" "Minneapolis
Demo" "MWEAOC" "mwwaoc" "NCRHQ" "NEFSO2" "NER-
```

```
OSM" "NetA"  "NetB" "ns/csc" "NSC" "NSC-5.0"
"OGC" "OPD-ENG" "OPD-HQ" "OPD-IS" "OPD-LE"
"OPD-MFG" "OPD-MPO" "OPD-MS" "orange" "OS
Service" "osba" "Osbu Bayshore"  "OSBU North"
"OSBU South" "OSD Associates" "OSM-OKC" "OSM-
TRG" "OSMDB" "Parc"  "Parc Place" "Phoenix"
"PQAnet1" "PQAnet2" "PQAnet3" "PR" "ProdTest"
"ProductTest" "Psd-Executive" "PSD5700"
"Rainbow" "Raisin" "roch" "ROCH041"  "Roch805"
"Roch805t" "ROCH853" "Roch888" "ROCH892"
"RochECE" "SalmonDomain"  "San Antonio" "Santa
Fe" "South" "STHQ" "Test Base" "Test Net"
"TestServices"  "TOMF" "TransientRoutes" "TSC"
"TYSON" "Vista" "Walnut Creek" "waoc"  "WBST MFG
HUB" "WBST102A" "wbst102t" "wbst102tchs"
"WBST105" "Wbst105B"  "WBST114" "Wbst128"
"WBST129" "WBST129UL" "WBST200" "WBST200LL"
"WBST200UL"  "WBST205LL" "wbst205t" "WBST205UL"
"WBST207" "Wbst207ul" "WBST208" "WBST212"
"WBST218LL" "WBST218UL" "WBST223" "WBST223CHS2"
"WBST300" "WBST304" "WBST311"  "WBST311C"
"wbst311J" "WBST311X" "WBST845" "WRC" "xais-
demo" "xc es"  "XNS-DLBK" "XNS-MAR" "XOS WR"
"XOS-MAR" "xos-mwr" "XOS-OPS" "XOS-SCBC"  "XOS-
SR" "XRCC" "Xsis" "XSIS North" "Xsis-Ai" "XSIS-
HQ")

21_(CH.LIST.DOMAINS "P*C:Xerox"]
("Parc")

22_(CH.LIST.DOMAINS "WBST*:")
("WBST MFG HUB" "WBST102A" "wbst102t"
"wbst102tchs" "WBST105" "Wbst105B"  "WBST114"
"Wbst128" "WBST129" "WBST129UL" "WBST200"
"WBST200LL" "WBST200UL"  "WBST205LL" "wbst205t"
"WBST205UL" "WBST207" "Wbst207ul" "WBST208"
"WBST212"  "WBST218LL" "WBST218UL" "WBST223"
"WBST223CHS2" "WBST300" "WBST304" "WBST311"
"WBST311C" "wbst311J" "WBST311X" "WBST845")
```

(CH.LIST.OBJECTS *ObjectPattern Property*) ž returns a list of all of the objects whose object name matches the *ObjectPattern* AND who have property *Property*. *ObjectPattern* consists of an NS name. The object part (but not the domain and organization parts) can include the * wildcard character, standing for 0 or more of any characters. If the domain and/or organization parts are left off, the default domain and/or organization is used. If

ObjectPattern is NIL or just *, then the list returned will be a list of all objects in the default domain and organization.

The *Property* argument indicates the type of object to look for and can be one of: USER, PRINT.SERVICE, FILE.SERVICE, MAIL.SERVICE, MEMBERS, or ALL as described above. If the *Property* argument is NIL, then ALL will be used meaning all types of objects.

Examples:

```
28_(CH.LIST.OBJECTS (QUOTE *:Parc:Xerox)
             (QUOTE FILE.SERVICE]
("phylex" "Starfile")

29_(CH.LIST.OBJECTS (QUOTE *:Parc:Xerox)
             (QUOTE PRINT.SERVICE]
("Kukai" "PaperMate" "print")

30_(CH.LIST.OBJECTS '*:Parc:Xerox 'USER)
("Akis Doganis" "Ann Derrick" "Arnold J Blum"
"Art Farley" "Barbara Hemstad"  "Bart Kinne"
"Beaumont Sheil" "Benny Pugh" "Bev Manes" "Bill
Hunter"  "Bill Jackson" "Bill van Melle" "Bill
Winfield" "Bob Allen" "Bob Ritchie"  "Brad
Burns" "CA Lehner" "Cari Sullivan" "Carlin
DeCato" "Carol Lehner"  "Cathy Turner" "Charles
Orgish" "Cheryl James" "Courtney Tagupa"  "Cyndi
Vanderhorst" "D. Austin Henderson" "Dale Mann"
"Dan Jordan"  "Dan Russell" "Dave Pirogowicz"
"David Myron Levy" "David Porter"  "David
Vinayak Wallace" "David Weckler" "Diane
Hutchins" "Don Charnley"  "Dorene Allen" "Doug
Walters" "Ed Fiala" "Eric Rawson" "Eric Schoen"
"Eric Steffensen" "Facilities Monitoring"
"Fernando Ponce" "Frances Grimble"  "Frank
Halasz" "Frank Shih" "Frank Vest" "Fumiko
Mannes" "Gary Chang"  "Gary Emanuel" "Gary
Rhoades" "Gary Toyama" "Gene Hall" "Giuliana
Lavendel"  "Gloria Warner" "Greg Nuyens" "Gregor
Kiczales" "Guest" "Hal Murray"  "Harriet Weeks"
"Henry S. Thompson" "Herb Jellinek" "Hugh Vander
Plas"  "I-Wei Wu" "Irene Lile" "Jacqeline M.
Guibert" "Jay Trow" "Jean Gascon"  "Jeanette
Figueroa" "Jeannie Lewandowski" "Jim Cooper"
"Jim D'Alfonso"  "John Brown" "John D. Sybalsky"
"John L. White" "John Larson" "John S. Brown"
"John Shaw" "John White" "Jon Bokelman" "Joseph
Kaminski" "Julian Orr"  "Karen Martelli" "Kathy
Jarvis" "Kelly Roach" "Kenneth Beckman" "Kerry
```

```
Brown"  "Larry Masinter" "Librarian" "Lillian
Barth" "Lorraine Watanabe"  "LouAnne Johnson"
"Lynne Seymour" "Maia Pindar" "Mariela Esser"
"Mark Chow"  "Mary Hausladen" "Meg Withgott"
"Melissa Monty" "Michael Dawson"  "Michael
Fisher" "Michael Herring" "Michael Plass"
"Michael Sannella"  "Michael Young" "Michalene
Casey" "Michel Desmarais" "Mike Dixon"  "Mimi
Gardner" "Mitchell Lichtenberg" "Nancy Freige"
"Neil Gunther"  "Nicholas Briggs" "PARCPublic"
"Patricia Sheehan" "Paul Ricci" "Paul Turner"
"PC Demo" "Per-Kristian Halvorsen" "Peter
Struss" "Richard Burton"  "Richard E. Sweet"
"Richard Martin" "Robert Allen" "Robert
Bachrach"  "Robert Spinrad" "Robert Tremain"
"Ronald Kaplan" "Ronald Schmidt"  "Salina
Snipes" "Sharon Johnson" "Sharon Penner" "Star1"
"Star2" "Star3"  "Star4" "Stephen Jackson"
"Stephen Quarterman" "Steve Martino"  "Steve
Wallgren" "Steven Purcell" "Susan Newman" "Susi
Lilly" "Susie Mulhern"  "Sweetsun Chen" "Tak
Oki" "Tami DeMerritt" "Tayloe Stansbury" "Terry
Haney"  "Thomas Hartmann" "Tiao-Yuah Huang" "Tim
Brunner" "Tim Diebert" "Toby Morrill"  "Tom
Moran" "Victor Bojorquez" "Victoria Carlson"
"Wes Dorman" "Yoko Nonaka"  "Zoran Popovic")
```

(CH.LIST.ALIASES.OF *ObjectPattern*) ž returns a list of all of the aliases of *Object*, where *Object* is an NS name.

The object part (but not the domain and organization parts) can include the * wildcard character. However, the function will return the aliases of only the first object found that matches the pattern. Thus, it makes little sense to use wildcards in this function.

If the domain and/or organization parts are left off, the default domain and/or organization is used. If *Object* is NIL, * is assumed and the aliases of the first object in the default domain will be returned.

Examples:

36_(CH.LIST.ALIASES.OF (QUOTE Frank% Halasz))
(Halasz:)
37_(CH.LIST.ALIASES.OF (QUOTE StarFile:))
(Help Server: Help Service:)
38_(CH.LIST.ALIASES.OF (QUOTE Phylex:))
(aifile:)

39_(CH.LIST.ALIASES.OF (QUOTE aifile:))
(aifile:)
40_(CH.LIST.ALIASES.OF (QUOTE *:PARC:))
(ht: HThompson:)

(CH.RETRIEVE.MEMBERS *Object* 'MEMBERS) ž retrieves the list of members of a distribution list. *Object* is the NS name of the distribution list. A distribution list is an NS object with a MEMBERS property, i.e., an object returned by *(CH.LIST.OBJECTS "*:PARC:XEROX" 'MEMBERS)*.

Example:

48_(CH.LIST.OBJECTS '* 'MEMBERS]
("AllParc" "AllXerox" "Alpha BWS" "BWS Users" "HelpGroup" "ICL Star Users" "LispAccess" "LISPCORE" "NewXAISEmployees" "PTS")
49_(CH.RETRIEVE.MEMBERS (QUOTE PTS:) 'MEMBERS]
(Carol Lehner: Toby Morrill:)

Mail Service

The NS world provides a mail delivery service similar to, but separate from, the Grapevine mail system. To use this mail system, you will need get registered in a Clearinghouse and have a mail folder set up for you on some mail server. At PARC, see Carol Lehner to have this done.

If you load {eris}<lisp>harmony>library>nsmail.dcom then evaluate the function call **(LAFITEMODE 'NS)**, you can use Lafite to gain access to your NS mail. The Lafite *Get Mail* command will retrieve mail from your NS mail folder and the *Send Mail* command will send the mail out using the NS mail system.

You can send and recieve mail in only one world at a time. To read your Grapevine mail again, evaluate **(LAFITEMODE 'GV)**. To return to NS mail use **(LAFITEMODE 'NS)** again.

You can intermix Grapevine and NS mail messages in a Lafite mail folder. That is, you needn't use different mail folders as you switch between the two mail

systems. However, the *Answer* and *Forward* will not work correctly for NS mail when the LAFITEMODE is GV, and vice versa.

When sending NS mail, you need to include the full NS name of the recipient (i.e., name/alias, domain, and organization). If the recipient belongs to the same domain and/or organization, you can omit these parts of the NS name. For example, to send mail to me the receipient list should be *Halasz:PARC:Xerox* or *Frank Halasz:PARC:Xerox*.

Your NS mail server and the Clearinghouse take care of delivering the mail to the appropriate mail server for each recipient.

I think there are distribution lists allowed in the NS mail system. These are the NS objects with the MEMBERS property discussed above. Mailing to these distributiuon lists is like mailing to each member of the list. For example. mailing to *PTS:PARC:Xerox* is like mailing to *Lehner:PARC:Xerox* and *Morril:PARC:Xerox*, since these are the only two MEMBERS of *PTS:PARC:Xerox*.

> Unlike in Grapevine, these distribution lists can be changed only by the Clearinghouse administrator. Although they can be examined using the CH.LIST.OBJECTS and CH.RETRIEVE.MEMBERS functions described above.

File Service, Print Service, Press versus Interpress printers

File service in the NS world is provided by NS file servers. These are similar to IFSs, but do not have some of the IFS features. The salient points of NS file server have been discussed in the LispCourse sections on filing.

Print service in the NS world is provided by NS printers (the Xerox 8044 printer). The 8044 NS printers are abysmally slow but in general have much higher print quality than the Dover and full-press printers in the PUP world. All NS printers print files only in the Interpress document format. Press files cannot be printed in the NS world.

Most files in Interlisp can printed on either the Press (Pup) or Interpress (NS) printers. But note that the fonts available on the two printers are different.

Interlisp does most of its screen displays in PUP-world fonts like Gacha, TimesRoman, and Helvetica. Whenprinted on an Press printer, these fonts appear as they do on the screen (except at higher resolution). When printed on an Interpress printer, these fonts are translated into their NS analogs: Terminal, Modern, and Classic. For most applications, this is okay. But files which are, for example, carefully adjusted to fit on a Press page will not fit in the same way (or not at all) on an Interpress page.

You can use the NS fonts on the screen (e.g., in a TEdit window) by setting the correct font variables. When printed on a Press printer, these fonts will be printed in their Pup-world analogs. When printed on an Interpress printer, these fonts will appear as they do on the screen (modulo the resolution).

Interfaces Between the PUP and NS Worlds ž Interlisp-D and Mail Gateways

The PUP and NS world are generally separate worlds. But since Interlisp-D speaks to both worlds, you can often mix and match operations in the PUP world with operations in the NS world. For example, you can COPYFILE a file between an IFS and an NS file server as in *(COPYFILE '{phylex:}<halasz>init '{phylum}<halasz>init).*

In all of the interactions where you are dealing with a mix of NS and PUP, your Interlisp machine is the intermediary between the two worlds. For example, COPYFILE copies the file from the NS server to the Lisp virtual memory using the NS filing protocol and then copies it from the Lisp virtual memory to the IFS using the FTP protocol.

There is one interface between the PUP and NS worlds that exists apart from your machine: the mail gateway between the Xerox PUP Internet and the Xerox NS Internet. The mail gateway is a machine running somewhere in OSD that takes Grapevine mail addressed to NS mail recipients, translates it into NS mail and gives it the NS mail system to deliver. Similarly, it takes NS mail intended for Grapevine recipients, translates it to Grapevine mail, and then gives it to the Grapevine to deliver.

To send mail through the gateway from Grapevine to an NS mail recipient, you should address the mail to ***"NSName"*.ns**, where *NSName* is the NS Name of the recipient. For example, to send me NS mail from Lafite running in GV mode,

you should address the mail to *"Halasz:PARC:Xerox".ns* (don't forget the " before and after the NS Name).

To send mail through the gateway, from the NS world to the Grapevine world, you have to carry out a similar, but presently (9-Apr-85 00:16:00) unknown trick in specifying the recipients address.

Documentation on Networks

There is relatively little documentation on the Interlisp/NS world.

The Interlisp/PUP world is scattered throughout the IRM and package documentation, since almost all parts of Interlisp take advantage of the network at times.

The Interlisp interface to both NS and PUP networks is covered in Chapter 21 of the IRM. All of the Clearinghouse functions described above are covered here. Most of the chapter, however, is aimed at programmer's interface to the PUP and NS networks.

The Lafite interface to NS mail is documented in {eris}<lisp>harmony>library>nsmail.tedit (&.press).

There are several PARC Blue&White reports covering various apsects of both the PUP and NS network designs. You can probably get a list of these from the TIC.