Prospero User's Manual
Version 5

Draft of 8 July 1993
Document Revision No. 0.2

B. Clifford Neuman Steven Seger Augart

Information Sciences Institute
University of Southern California

Digital copies of the latest revision of this document may be obtained through Prospero as /papers/subjects/operating-systems/prospero/doc/user-manual.PS.Z, in the #/INET/EDU/ISI/swa virtual system, or through Anonymous FTP from PROSPERO.ISI.EDU as /pub/prospero/doc/prospero-user-manual.PS.Z

This work was supported in part by the National Science Foundation (Grant No. CCR-8619663), the Washington Technology Center, Digital Equipment Corporation, and the Defense Advance Research Projects Agency under NASA Cooperative Agreement NCC-2-539. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any of the funding agencies. The authors may be reached at USC/ISI, 4676 Admiralty Way, Marina del Rey, California 90292-6695, USA. Telephone +1 (310) 822-1511, email info-prospero@isi.edu.

Contents

1 Introduction

The Prospero file system is based on the Virtual System Model. It differs from traditional distributed file systems in several ways. In traditional file systems, the mapping of names to files is the same for all users. Prospero supports user centered naming: users construct customized views of the files that are accessible. A virtual system defines this view and controls the mapping from names to files. Objects may be organized in multiple ways and the same object may appear in different virtual systems, or even with multiple names in the same virtual system.

Prospero directories can contain references to files and directories that are stored on remote nodes. This allows distribution at a much finer level of granularity than is possible in traditional distributed file systems. Prospero also provides several tools to support customization. Among them are the union link and the filter.

2 The Directory Mechanism

In Prospero, the global file system consists of a collection of *virtual file systems*. Virtual file systems usually start as a copy of a prototype. The root contains links to files or directories[1] selected by the user. Directory links can be of several types: conventional, union, and filtered.

A conventional link is similar to a hard link in traditional file systems. It may be made to any type of object, including a directory. It maps a name for an object to the information needed to access the object. As long as an unexpired link to an object exists the object may be accessed by that name. If the object moves, a forwarding pointer will allow continued access using the same name. An object is only deleted when no unexpired links to it remain.

A union link can only be made to a directory. With a union link, the objects included in the linked directory become part of the virtual directory containing the link. Thus, the contents of a virtual directory are the union of the collection of conventional links it contains, and the contents of all directories included through union links.

Filters may be attached to either type of link. A filter alters the set of links that are seen in directories whose paths pass through the filtered link. A filter can specify which links are to appear and which are to be ignored, it can change the features of individual links, or it can synthesize new links that are not in the original directory. Client-side filters are written in C and are dynamically linked during name resolution. A filtered link contains a reference to the filter and any arguments required by the filter.

3 Using the Prospero File System

This section assumes that Prospero has already been installed on your system. Section 10 describes the installation procedure.

[1] To distinguish them from directories and links in traditional systems, and because they are often illusory, directories and links in Prospero are sometimes referred to as virtual directories and virtual links.

To use Prospero you must first determine the name of the directory that contains the executables.[2]

- CSH users That directory contains a file called vfsetup.source, the contents of which must be read by the shell. This can be accomplished by `source`ing it. For example, if the Prospero file system binaries are stored in */usr/pfs/bin* you should either execute the following command or add it to your *.cshrc* file:

```
source /usr/pfs/bin/vfsetup.source
```

- SH users That directory contains a file called `vfsetup.profil`, the contents of which must be read by the shell. This can be accomplished with the `.` command. For example, if the Prospero file system binaries are stored in */usr/pfs/bin* you should either execute the following command or add it to your *.profile* file:

```
. /usr/pfs/bin/vfsetup.profil
```

Before you can begin using the Prospero file system a virtual system must be created for you. Section 10.2.1 explains how the Prospero site administrator can create a new virtual system. Don't be frightened away from experimenting by these comments! Most Prospero sites support a *guest* virtual system which can be used until your own virtual system is created. Prospero, as shipped, is configured so that once you compile the clients you can type "`vfsetup guest`" and start working right out of the box using a guest virtual system at the USC Information Sciences Institute.

To use a virtual system, you must first execute the `vfsetup` command to initialize your environment. For example, if the name of your virtual system is *guest* you should execute the following command:

```
vfsetup guest
```

4 Using the menu browser

The menu browser provides a simple straightforward interface to the Prospero file system.

4.1 Starting it up

In the simplest case, invoke it as "`menu`". If you are vfsetup to a virtual system and there is a directory named `/MENU` in that virtual system, it will use that directory. If you have not run the `vfsetup` command or if there is no link named `/MENU` in the current virtual system, it will bring up the default menu for your site. We have set up a default menu for people who compile Prospero to use the default site, the ISI guest site.

If you are already vfsetup to a virtual system, you can invoke the menu browser as "`menu` *linkname*". The browser will display the Prospero directory *linkname* as its root menu.

[2] By default, the installation directory is */usr/pfs/bin*.

4.2 Using it

The browser will display a numbered list showing the contents of the directory. The items in the list will be followed by one of four characters:

. A file. If it is a data file, you will be able to retrieve it and save it. If it is a text file, you can retrieve it and view it, and then have the option of mailing it, saving it, or printing it.

> A directory (submenu). You can select it to see the contents, and then type `u` to go up to the previous menu.

] A portal. This represents a service you will have to connect to via *telnet* or some similar protocol. Select it in order to receive some instructions and begin a *telnet* session.

: A search. It will display some initiali documentation. It may ask you to select among several possible searches, and will then prompt you for what you are searching for. Later versions of the menu browser will check what you type at the prompts to make sure the format is correct. Type `?` at the beginning of any line in order to receive further documentation about the meaning of that line.[3]

The results of the search will be displayed as a new menu that contains a list of the items that matched the search, possibly including subdirectories.[4]

Type the number of the item you wish to explore in order to explore that item. Type `q` to quit.

4.3 Making your own menus

The main current deficiency of the menu browser interface is that at the moment the browser is read-only. For now, you will have to learn a bit about the command-line client interface in order to set up new directories under Prospero, in order to change the names of menu items, and in order to create new links. We are currently working to put most of the power of the command line interface into the menu browser so that one does not have to use two user interfaces for most purposes.

To make your own menus, you will have to learn the `set_atr` command, to set attributes on objects, the `vln` command, to make new links, and the `vmkdir` command, to make subdirectories.

5 Using the command-line clients

5.1 File Names

The slash, colon, pound sign, open and close parenthesis, and backslash (`/`, `:`, `#`, `(`, `)`, and `\`) are special characters in Prospero. The slash separates components of file names and the colon separates information identifying a virtual system from the name of a file within the virtual system. These characters should not appear in any component of a file name unless they have been quoted.

[3] If you want to search for an item beginning with a literal ?, then precede it with a backslash to escape it. If you want to search for something beginning with a backslash, then precede it with another backslash to excape it.

[4] N.B.: If you make a search that is relayed through the Prospero/Gopher gateway, you may be immediately presented with a submenu containing a single item, and will have to open that item to see the results of your search.

All special characters, including the backslash, can be quoted by preceding them with a backslash. This is important to know, especially when you're working with names returned by the Prospero GOPHER-GW gateway; those names often contain colons, slashes, and other special symbols.

The slash is used in user-level names to indicate moving into a subdirectory, just as it does in UNIX-like operating systems. This is the only important special character to know about upon the first reading of this document; the others are more specialized.

By default, names are resolved relative to the active virtual system. If a colon (`:`) appears in a name, the name is resolved relative to the name space identified preceding the colon. If the character preceding the colon is a pound sign (`#`), then the name preceding the pound sign will be treated as an alias for a previously specified virtual system. For this reason, virtual systems should not have names ending in a pound sign. A double colon (`::`) is the closure operator. When encountered, the name space identified by the CLOSURE attribute[5] of the object named before the double colon is used to resolve the name that follows.

The pound sign (`#`) is also used to resolve name conflicts when the same component of a name is used by more than one object. In this case the pound sign is followed by the magic number of the desired object. For this reason, unless quoted, the pound sign should not be used in a component of a file name if followed by a number (including sign), and if there are no intervening non-numeric characters between the pound sign and the end of the component.

The pound sign (`#`) is additionally used to indicate that a particular named union link is to be followed when resolving a name, or to indicate that a filter is to be applied. In these cases, the pound sign is the first character of a component in a name and it is followed by the name of the union link to be followed or the name of the filter to be applied. For this reason, unless quoted, the pound sign should not be used as the first character in a component of a file name (unless the full name of the component is `#`).

The open and close parenthesis (`(` and `)`) are used to delimit the arguments to a filter specified as part of a file name. The arguments immediately follow the name of the filter (which itself follows a single or double pound sign). If the filter name follows a single pound sign, it is a client-side filter. If the filter name follows a double pound sign, it is a server-side filter.

Therefore, it follows that if the name of a union link ends in a `)`, and it is explicitly specified as part of path name, the `)` must be quoted.

We have not yet specified a syntax for differentiating between client-side pre-expansion and post-expansion filters, nor for differentiating among directory, hierarchy, object, update, and link filters.

If no arguments are required, the null argument list `()` must be included. These uses of the # convention to specify a filter is not currently supported.

Loadable Filters The above filter syntax works for predefined filters. Loadable filters are not currently used. The working definition of their syntax, which is being redefined due to it being difficult to parse, is: If the filter name contains an unquoted colon or slash, then it is interpreted as a loadable filter. If the filter name does not, it is interpreted as a predefined filter. Note that you can specify a loadable filter in the current virtual directory by prefixing its name with the sequence `./`.

[5] See the Attributes appendix to the Prospero Protocol specification for the definition of this attribute

5.2 Finding Things

The Prospero file system provides tools that make it easier to keep track of and organize information in large systems. When first created, your virtual file system is likely to contain links to directories that organize information in different ways. As the master copy of each of these directories is updated, you will see the changes. You may customize these directories. The changes you make to a customized directory are only seen from within your own virtual system, but changes made to the master copy will also be visible to you. See section 5.5 for instructions on customizing a directory.

Users are encouraged to organize their own projects and papers in a manner that will allow them to be easily added to the master directory. For example, users should consider creating a virtual directory that contains pointers to copies of each of the papers that they want made available to the outside world. This virtual directory may appear anywhere in the user's virtual system. Once set up, a link may be added to the master author directory. In this manner, others will be able to find this directory. Once added to the master directory, any future changes will be immediately available to other users.

To add a link to the master copy of any of the shared directories, send a message to your site administrator. The address should be *pfs-administrator* on the primary system for the site. If you are using a virtual system stored at the USC Information Sciences Institute, the address would be *pfs-administrator@isi.edu*.

5.3 The Commands

This section assumes that the Prospero file system has been installed on the system being used. Later sections explain how to install the Prospero file system at a new site and on individual systems.

Most of the commands that are specific to the Prospero file system take a debug option. The form is *-D#* where # is an optional integer and specifies the level of detail. The higher the integer, the greater the detail. By itself, *-D* sets the debugging level to 1. Debugging levels of 9 and above display the actual Prospero protocol messages that go across the network.

5.3.1 Initialization and Changing Virtual Systems

```
vfsetup [-n host path , [-r,v] name , -f file]
```

The `vfsetup`[6] command sets up the selected virtual system. It adds the appropriate directories to the search path and sets all necessary environment variables. `vfsetup` can be called in several ways. With no arguments, it reads the file *~/.virt-sys* and uses the information found to access the virtual system description. The *-v* option takes the name of the virtual directory containing the system description. The *-n* option takes the name of a host and the physical name of the virtual directory on that host that contains the virtual system description. The *-f* option takes the name of a Unix file that is to be read in place of *~/.virt-sys*.

It is also possible to set up a virtual system by specifying its name. If the *-r* option is specified, the name is taken to be the default name for the virtual system at the local site. If the name is

[6] Because it changes environment variables, this command only has an affect when its output is read by the shell. If *vfsetup.source* has been `source`ed, then `vfsetup` is an alias which will call the *vfsetup* executable in the appropriate manner.

specified without a modifier, the name is looked up in the /VIRTUAL-SYSTEMS directory of the presently active virtual system (the site default is used if no virtual system is presently active). For example, if the name of your virtual system is *guest* you can set up the virtual system using following command.

```
vfsetup guest
```

5.3.2 Creating Directories

```
vmkdir directory
```

`vmkdir` creates a virtual directory with the selected name and adds a link from its parent directory.

5.3.3 Adding and Deleting Links

vln

```
vln [-i, -u, -s, -m, -a] {-e access-method-info, -n host hsoname, linkname } newname
```

`vln` adds a new link to a directory. *oldname* is an existing name for the object to which the link is to be made. *newname* is the name of the new link. The *-u* option indicates that the new link is to be a union link. The *-s* option is used to specify a symbolic link. The *-i* option is used to specify an invisible link which will not be displayed in a normal directory listing.

The *-n* option (short for *native*) requires the specification of the name of the host containing the target. The *-n* option indicates that the native information for the target has been specified. *host* is the name of the host on which the target resides and *oldname* is the name of the target on that host. If the *-s* option has also been specified, then *host* is the name of the virtual system to which oldname is relative.

The *-e access-method-info* (external) option indicates that the object resides on a host that does not run Prospero. There are are number of possible values for the *access-method-info*:

GOPHER *host*(*port*) *gopher-selector* {**BINARY** *or* **TEXT**} This access method indicates that the object can be retrieved by sending the selector string *gopher-selector* to a server running at port *port* on host *host*. You must specify whether you want the object to be retrieved using the Gopher binary or text retrieval methods.

TELNET *host*[(*optional-port*) *introductory-message*] If you do not provide a port inside parentheses, then the default *telnet* port will be used. The *introductory-message* will be displayed before the user connects to the service. For example:

> Type `LAX` at the prompt in order to get the current Los Angeles weather forecast;
> type `X` to quit.

AFTP *host path* {**BINARY** *or* **TEXT**} This access method specifies that the object can be retrieved via anonymous FTP, using either the *binary* or *text* retrieval methods.

AFS *afs-path* This access method specifies that the object is available through the Andrew File System. Its name via AFS is *afs-path*. You should not precede the *afs-path* with `/nfs/afs`, `/afs`, or whatever the prefix is that your local system prepends to AFS names.

#-of-access-method-args method-namehost-type host hsoname-type hsoname any additional args This type is used to provide an explicit value for the ACCESS-METHOD attribute. See appendix A of the Prospero protocol specification for a discussion of the format of this attribute.

Note that some of the host names specified in these access methods may include a port number inside parentheses. You will probably have to quote the port number so that whatever shell you use does not interpret it in a way you don't expect.

Some examples of making links Here I'm making an external Gopher TEXT link to a recipie:

```
vln -e GOPHER 'ashpool.micro.umn.edu(70)' 0/fun/Recipes/Balls/rum-balls TEXT rum-balls
```

I can now retrieve this document with `vget` or by running "`menu .`".

Here I'm making a link to a telnettable service. In this case, I have decided to make the *introductory-message* a null string, since the service I'm linking to has its own excellent documentation facilities:

```
vln -e TELNET 'DOWNWIND.SPRL.UMICH.EDU(3000)' '' weather
set_atr weather OBJECT-INTERPRETATION PORTAL
```

In this case, I also had to run `set_atr` so that the menu browser would know this was a PORTAL.

Here I'm making a native link to a directory gatewayed through the ISI Gopher gateway (shipped as part of this distribution). Note that we are currently providing a demonstration Gopher gateway on Prospero server on ZEPHYR.ISI.EDU, port 1570.

```
vln -n 'ZEPHYR.ISI.EDU(1570)' 'GOPHER-GW/GOPHER.MICRO.UMN.EDU(70)/1/' minnesota-root-gopher
```

Specialized capabilities for Closure If the standard input to `vln` has been redirected, the input will be searched for a line of the form "Virtual-system-name: vs-name". If found, *oldname* will be relative to the virtual system which has been read from (closed with) the input. If the *-m* option has been specified, the input will be additionally searched for a line for the form "Virtual-file-name: filename". If found, *filename* will be used in place of *oldname*. The reading of the standard input can be suppressed by specifying the *-a* option, in which case the currently active virtual system will be used.

vrm

```
vrm link
```

`vrm` removes the named link from a directory. It is important to note that `vrm` only removes the link. The object will continue to exist if there are any additional links to it. If there are none, then the object will become subject to garbage collection at a future time.

Another important thing to note is that `vrm` will only remove a link if it exists in the directly indicated directory. You may be confused by `vrm` claiming that a link is not present when you can see it quite clearly via `als` or `vls` or `menu`. This problem arises because the link you're seeing is actually included via a union link. Use the *-u* flag to `vls` to see if this is the case. You can use the *-u* flag to `vcd` to put yourself into the directory that actually contains the link and try the `vrm` again. This problem also often arises with the `list_acl`, `set_acl`, and `set_atr` commands, and it has the same solution.

5.3.4 Listing Directories

Virtual directories may be listed using the `als` command or `vls`. `als` produces straightforward output similar to that produced by the standard UNIX `ls` utility. We recommend its use. `vls` produces more complex output and is more useful for maintaining directories than for exploring them. Of course, if one is interested in browsing, in our opinion invoking the menu broswer program on the current directory (to do this, invoke it as `menu .`) gives one the most straightforward user interface.

```
vls [-A, -a,-c,-f,-i,-u,-v] [ -A attribute ] [-a attribute ] [path]
```

`vls` takes the virtual path name for a file or directory. If the path is for a directory, the links within that directory are displayed. If the path is for any other type of object, then the information for the named link is displayed.

By default, `vls` displays for each link the link name (i.e., the local component of the path name) and the target of the link. The target of the link is generally a host and a name relative to that host.[7] Some special characters may precede the link name; their meanings are:

- U This is a union link (always to a DIRECTORY or DIRECTORY+FILE). Usually only shown if *-u* flag was specified, unless expanding the link failed.
- I for an invisible link (only shown if *-i* flag specified) (could be to a FILE, DIRECTORY, or DIRECTORY+FILE).
- blank (' ') if a normal link to a FILE.
- S for SYMBOLIC
- E for EXTERNAL (to an object on a host that does not run Prospero)
- N for NULL (returned if inadequate permissions),
- D for DIRECTORY
- B (Both) for DIRECTORY+FILE
- O for OBJECT (neither a DIRECTORY nor a FILE, just something that can have attributes associated with it.)
- * Indicates that a filter is associated with the link.
- F Expanding this union link failed.

The *-v* option causes the object type, and the type of each field to be displayed, and it lists the filters associated with the link. It also prevents the truncation of fields that are too long to be cleanly displayed without the *-v* option.

The *-u* option indicates that union links are not to be expanded. By default, union links are expanded, and the results of that expansion displayed. To see which union links are included in a directory, the *-u* option must be specified.

[7] We call these names Host-Specific Object Names, or HSONAMEs.

The *-i* option indicates that invisible links should be displayed; they are normally not.

The *-d* flag indicates that even if the *path* argument to `vls` is a directory, we want to look at that link instead of looking at the contents of the directory. It is just like the *-d* flag to the UNIX `ls` command.

There are cases when a directory might include more than one link with the same name. One way this can happen is if the directory contains union links. By default, only the first link with a particular name is displayed. The *-c* option tells `vls` to display all links, including those with conflicting names. The name of conflicting links will be followed by a "#" and a number that allows them to be uniquely identified.

The *-f* option causes union links which could not be expanded to be displayed. This option is presently set by default.

The *-a* option indicates that the attributes associated with each object pointed to by the link are to be displayed. It also forces the verbose option. If only a particular attribute is desired, the attribute can be specified as part of the *-a* option itself (e.g. -aFORWARDING-POINTER). The *-a* option by itself displays all attributes. The *-A* option is similar to the *-a* option, but it only lists the attributes associated with the link itself, not those associated with the object referenced by the link[8].

5.3.5 Moving Around

```
vcd [-u] path
```

The `vcd`[9] command allows one to change the virtual working directory. If no argument is specified, the home directory is assumed. Otherwise, paths starting with a slash (/) are treated as relative to the root of the virtual file system, and other paths are treated as relative to the current working directory. ".." specifies the directory above the current working directory along the active path from the root.

The *-u* option allows one to change one's virtual working directory to a directory included through a named union link.

```
vwd
vwp
```

The `vwd` command prints the name of the current virtual directory relative to the root of the virtual system. The `vwp` command prints the information describing its physical storage location. These are actually aliases defined at the time you run `vfsetup`.

5.4 Retrieving Files

```
vget virtual-file [local-file]
```

The commands described so far allow you to move around the virtual file system, but they do not allow you to access the files that it names. If a program has been linked with the Prospero compatibility library, the program can access files directly.

[8]When using `vls` to list the results of an archie query, the *-c* and the *-A* options should be specified.

[9]Because it changes environment variables, this command only has an affect when its output is read by the shell. `vcd` is an alias which calls the *p_vcd* executable in the appropriate manner.

The `vget` command can be used to explicitly retrieve a file. The *virtual-file* is the name of the file to be retrieved from the Prospero file system. *local-file* is the real name that you want the file to have in your real current working directory. If *local-file* is omitted the last component of the virtual file name will be used.

If the standard input to `vget` has been redirected, the input will be searched for a line of the form "Virtual-system-name: vs-name". If found, *virtual-file* will be relative to the virtual system which has been read from (closed with) the input. If the *-m* option has been specified, the input will be additionally searched for a line for the form "Virtual-file-name: filename". If found, *filename* will be used in place of *virtual-file*. The *-a* option can be used to suppress the searching of the standard input.

`vget` does not currently open *telnet* connections to objects with access methods of type TELNET. Nor does it treat objects with an OBJECT-INTERPRETATION SEARCH in any interesting way; you should use the `menu` program to open such objects.

5.5 Customizing a Directory

When a change is made to a directory, that change is often visible regardless of the path through which the directory is viewed. There are times when it is desirable to make a change that is only visible when the directory is viewed through a particular path, or from a particular virtual system. Such a change creates a customized view of the directory; the change will not affect the view of the directory when reached through other paths, or from other virtual systems.

To create a customized view of a directory, create an empty directory, add a union link from the directory you just created to the target directory, then remove the link to the old directory and replace it with a link to the directory that was just created.

Links that are added to the customized directory will only be visible through the customized directory, but changes to the target directory will also be visible through the customized directory.

Some directories in your virtual system have already been customized. The root of your virtual system is your own. Changes in the root do not appear in the roots of other virtual systems. Each of the links from the root is also a customized directory. For example, if you add a link to the /authors directory, that link will not be visible to others. Directories at the next level, however, are not customized. Thus, if you add a link to the directory /authors/Shakespeare,William, that change will be visible to others unless you first customize that directory.

You can determine whether a directory has been customized by using the `vls -u` command. That command will show the current directory without expanding union links. Admittedly, this is a little confusing. Future releases will support the concept of an owning virtual system. This will clear up some of the confusion by allowing automatic creation of a customized directory when one is needed.

To have a link added to the master copy of a shared directory you should send a message to *pfs-administrator* at your site, or to *pfs-administrator@isi.edu*.

5.6 Attributes

Attributes associated with the object to which a link points may be retrieved using the *-a* option to the `vls` command. Attributes associated with the link itself may be retrieved using the *-A* option to `vls`. Attributes may be set using the `set_atr` command.

Attributes in Prospero have three different value types: SEQUENCE, FILTER, LINK. They can be in one of three namespaces: APPLICATION, FIELD, and INTRINSIC. They have five different precedences: OBJECT, ADDITIONAL, REPLACEMENT, CACHED, and LINK. The FILTER type, in turn, has a number of options. These are all discussed in depth in the Prospero protocol manual. A link or object may have multiple instances of any attribute on it. This array of specialized features makes `set_atr` appear confusing at first.

set_atr

Shortcuts However, there are some useful shortcuts: if you are organizing your information through Prospero, theattributes you are likely to want to set on an object (COLLATION-ORDER, MENU-ITEM-DESCRIPTION, and OBJECT-INTERPRETATION) are in the APPLICATION namespace, which is the default for `set_atr`. You almost certainly want to just keep one instance of the attribute around, so `set_atr` defaults to using its *replace* option. If you want to delete the attribute entirely, use the *-delete-all* option.

If no attribute precedence is explicitly specified, `set_atr` selects what it believes the correct attribute precedences are using an adaptive mode: If a link is *external*, `set_atr` will set a a REPLACEMENT attribute on it. If a link is to an object stored under Prospero, `set_atr` will set an OBJECT attribute on the object and then cache the value of the OBJECT attribute with a CACHED value on the link itself. If `set_atr` can't modify the object itself, it will override the object's value of the attribute by putting a REPLACEMENT value on the link.

Examples Here is an example of how I use `set_atr` to organize the anonymous FTP Area of a host which has just started running a prospero server.

First, make a starting link to the AFTP area on that host:

```
vln -n ZEPHYR.ISI.EDU AFTP starting-link
vcd starting-link
```

I now run an `als` on the directory and verify the contents. Let's say it contains a file named "00README". I want to make this file have a more descriptive name for people using a menu browser. I also want to make it appear first in the directory, above any files which do not have an explicit COLLATION-ORDER specified. Moreover, I want another file, `Incomplete`, to appear last in the directory:

```
als
  (contents scroll by)
set_atr 00README MENU-ITEM-DESCRIPTION 'Information about the files in this directory'
set_atr -linkprec 00README COLLATION-ORDER NUMERIC 1
set_atr Incomplete MENU-ITEM-DESCRIPTION 'This file is still unfinished.'
set_atr -linkprec Incomplete COLLATION-ORDER LAST NUMERIC 1
```

In this case, I specified the additional *-linkprec* option to set_atr, because COLLATION-ORDER is an attribute that applies only to a link in a directory, not to the underlying object.

If I know that a file is of a particular type, I can set the OBJECT-INTERPRETATION attribute on this file to help the menu browser handle the contents effectively:

Table 1: Protection modes in access control lists

Directory		Link	
Character	Meaning	Character	Meaning
A	Administer	a	Administer
V	View	v	View
L	List	l	List
R	Read	r	Read
M	Modify	m	Modify
D	Delete	d	Delete
I	Insert	does not apply to link	
B	Administer	does not override link	
Y	View	does not override link	
>	Add-rights	]	Add-rights
<	Remove-rights	[	Remove-rights
)	Add-rights	does not override link	
(	Remove-rights	does not override link	

```
set_atr group-photo.gif OBJECT-INTERPRETATION IMAGE GIF
```

The OBJECT-INTERPRETATION attribute is described in the document `doc/working-notes/object-interpretation.txt`, which is included in the Prospero distribution, but which has not yet been integrated into this manual. A quick example of turning a file invisible with the -linkprec option:

```
set_atr .cap -linkprec LINK-TYPE I
```

The full gamut of options This has not been fully documented yet. Type `set_atr` by itself to receive a full list of options.

5.7 Forwarding Pointers

This release includes preliminary support for forwarding pointers. If a file or directory has moved and a forwarding pointer exists, the forwarding pointer will be returned and the request retried. At the moment, forwarding pointers must be added by hand; there are not presently any programs to add them. For information on how to add these by hand, send a message to *info-prospero@isi.edu*.

6 Protection

Access control lists (ACLs) may be associated with directories in Prospero, and with individual links within a directory. These access control lists specify how directory information is to be protected. They have nothing to do with the protection of the file to which a link refers. Table 1 lists the protection modes that may appear within an access control list entry.

When an entry appears in both columns, it means that the entry on the directory overrides the entry on a link. For example, a user with R access to the directory can read a link even if denied r access to the link. If a link permission is stored in a directory access control list, then that permission indicates the default protection associated with links in the directory that do not specify their own access control lists.

The administer permission allows the changing of the access control list. View allows it to be viewed. List allows one to see a directory entry using wildcard searches. Read is required to determine the binding of a link (the file it references). If one is allowed read, but not list, then one can only retrieve the link if its exact name is specified. Modify allows one to change the binding of a link, but it does not allow one to add or delete links. Insert allows links to be added, and delete allows them to be deleted. The add and remove rights are a restricted form of administer. They allow an individual to add or remove a restricted set of rights. For example, ">r" allows one to grant read access to someone else without also allowing one to grant modify access.

Negative rights may be specified by prepending a minus sign (-) to the rights field. The order of access control list entries is important. For negative rights to have an effect, they must precede any rights that authorize access by that individual. When new ACL entries are added, they are added at the front of the list, meaning that recent entries take precedence over older ones.

When access is checked for a link, three access control lists are checked. First, the ACL associated with the link itself is checked. If the link does not have its own ACL, then the default ACL associated with the directory is used. Next, the ACL associated with the directory is checked to see if it grants rights that override those in the link. Finally, if access has still not been granted, a special override ACL is checked. This is maintained by the system and should be used only in emergencies. Negative rights in one list does not override access granted in another.

There are several different ACL entry types. DEFAULT, SYSTEM, and DIRECTORY cause other access control lists to be included. DEFAULT is the default ACL specified by the system on which the Prospero server is running. SYSTEM is also specified separately on each Prospero server. It usually grants additional access to system administrators. DIRECTORY is the default access control list associated with the directory containing a link. It allows one to easily specify that the rights on a link are to be in addition to the default rights specified in the directory. If no rights are associated with these ACL entry types, then the rights granted are based on those in the DEFAULT, SYSTEM, or DIRECTORY access control lists themselves. If rights are specified for such entries, then the rights are the minimum of those specified, and those in the included ACL. The ACLs for new files and directories allow access to DEFAULT and SYSTEM as defined in those lists. Users have the option of removing such access[10].

The ANY, AUTHENT, ASRTHOST, and TRSTHOST ACL types grant rights to the specified individuals according to the accompanying permission list. ANY matches any user. AUTHENT specifies an authentication method, and the name of the authorized individual as returned by that method. At present, this entry type is not supported. The ASRTHOST method specifies a list of authorized principals in the form *user@internet-address*. If no Internet address is specified (and no atsign), then the user is matched regardless of the requesting host. Octets of the Internet address can be wildcarded, or replaced with a '`%`'. A wildcard matches any number, and a '`%`' matches the number corresponding to the local host. For example, "`bcn@%.%.%.*`" matches the user "bcn" on the local subnet. The *user* may be specified as `*`, meaning to match any user at that *internet-address*.

The ASRTHOST type accepts the username asserted by the client. It is not possible to verify that

[10]The SYSTEM access control list is separate from the OVERRIDE list which can not be removed.

the user has not modified the software to claim someone else's identity. The Internet address can generally be considered accurate, though it too can be spoofed by a knowledgeable and determined attacker. The TRSTHOST type is identical to the ASRTHOST type, but is accepted only when the request originates from a privileged port on the requesting system. Although this method might be used to provide security similar to that for the Berkeley R commands, it is not recommended that you install Prospero binaries setuid root until the sources have undergone careful scrutiny for possible security holes[11].

The discussion of ACLs in this section is not inaccurate, but it does not yet reflect the new object and container ACLs that are new with Prospero version Alpha.5.2. Preliminary documentation on the new rights we have developed for object ACLs is available in the text file `doc/working-notes/new-acl-types` in the Prospero distribution. This text file will be merged into this manual shortly.

The `ppw` command, which allows one to set and manipulate a password-based authentication mechanism that is stronger than the `ASRTHOST` type also needs to be documented here, as do the Kerberos version 5 authentication facilities which are present in this release.

The system administrator's `pw_edit` command also must be documented here.

Listing ACLs

The `list_acl` command may be used to list the contents of an access control list.

```
list_acl [-d dir] [-i host acl-name] [ -o object ] [link-name]
```

With no arguments, it lists the ACL for the current directory. If the *-d* option is specified, the argument that follows the option specifies the directory whose ACL is to be listed. An optional link-name specifies that the ACL to be listed is that of the named link within the directory. The *-o* option indicates that the ACL should be listed for the underlying object.

Modifying ACLs

The `set_acl` command allows one to change the access control list associated with a link or directory.

```
set_acl [-asirKE,-n,-N] [-t type] [-d dir] [-l link] [-o link-to-object]
rights principals
```

The *-d* option is followed by the name of a directory. By default, the ACL for the directory is modified. The *-l* option allows one the modify the ACL for an individual link. The *-o* option allows one to modify the ACL for the object pointed to by the link *link-to-object*.

The *-a, -s, -i, -r, -K,* and *-E* options indicate the operation to be performed on the ACL. They correspond in order to add rights, subtract rights, insert a new entry, remove an entry, kill the entire ACL (setting it to the default), and replacing the entire ACL.

If the *-t* option is specified, it must be followed by the type of the ACL entry to be added, deleted, etc. If the *-t* option is not specified, ASRTHOST is the default.

[11] By no means should vget or vcache be installed setuid root as these command write files to paths specified by the user.

The first field following the options specifies the rights to be added or deleted. All remaining arguments are the names of the principals to be included in the particular ACL entry.

When an ACL is set to an initial value (using the *-K* or *-E* options), the SYSTEM ACL is automatically included. The SYSTEM entry can be removed by using the *-r* option in a subsequent `set_acl` command. The addition of the SYSTEM entry can be suppressed by using the *-n* option in conjunction with the original *-K* or *-E* option. Whenever rights are removed from an ACL, the system checks to make sure that the user removing the rights will be able to fix any mistakes. If the the change would result in the user being unable to make subsequent changes, the minimal rights allowing the user to make subsequent changes are automatically added back. This safety mechanism may be overridden by specifying the *-N* option.

7 Server Maintenance and Informational Commands

7.1 pstatus

```
pstatus [<server>]
```

Use `pstatus` to see if the server *server* is alive.

7.2 padmin

```
padmin [-D#] [-N<priority>] [-force] { -kill | -restart
        | -motd | {-set <parameter>} | {-get <parameter>}
        | {-command [-headers | -1 ]} } [<server>]
```

Summary

The old `pkl` and `psrvchat` commands have been replaced by the new 'padmin' command. Padmin can be used to administer a Prospero server (kill it, restart it, set the message of the day). It can also be used by a programmer to send raw Prospero protocol messages to the server (with the `-command` option) and by anybody to retrieve the message of the day (with the `-motd` option).

Long Explanation of the options

The `-D` flag, as usual, sets the debugging level.

`-N` ('nice') sets the priority for a query. Setting a priority of 32765 or greater (`-N` by itself does this) means that this command will be processed only after the queue is empty. This is handy for terminating or restarting a server that gets a lot of traffic and often has requests waiting in the queue.

`-kill` kills the server. `-restart` causes a complete restart, including reloading the binary (handy if you've just updated the binary). If these options are specified, padmin will ask for confirmation, unless the `-force` flag is specified.

If the `-force` flag is specified, padmin will not confirm the `-kill` and `-restart` flags, nor will it output the reply received. (This is modeled upon the `-f` flag to rm).

`-set` takes an extra argument for a parameter to set on the server. It will then read from standard input for the text to send. This can only be used to set parameters whose value is a

SEQUENCE consisting of a single ASCII string. `-motd` is equivalent to `-set` MOTD. (Note that, currently, the only parameters defined on most servers are MOTD, RESTART, and TERMINATE, and they all have this form.)

`-command` reads raw Prospero protocol messages from the standard input and sends them to the server, then shows the response to the user.

A `-header` option may follow the `-command` option. The `-header` option will prefix the messages read from the standard input with standard Prospero VERSION and AUTHENTICATE statements for the current Prospero version. (Note for Prospero programmers: the headers from the `-header` option are generated with the standard `p__start_req()` pfs library call). A `-1` option may follow `-command` instead of `-header`. This will generate VERSION and AUTHENTICATE statements for Prospero version 1 protocol format.

`-k`, `-r`, `-f`, `-m`, `-s`, `-g`, and `-c` also work, as synonyms for the spelled out options. They can't be grouped together, though; you must specify them independently. (i.e., "`padmin -kfD9`" won't work; you must say "`padmin -k -f -D9`".)

For those used to the previous `pkl` command, the default installation procedure also installs `pkl` as a link to `padmin`. When invoked with the name `pkl`, `padmin` duplicates `pkl`'s full functionality.

8 The compatability library

This section documents what we call the "compatability library" interface to Prospero. The compatability library interface is not compiled by default; your site maintainer must specify it. We have not focused recent work on it, and it is not as useful an interface as the menu browser interface and the command line interface, both of which are discussed above.

This section does not apply to you unless your maintainer has turned on the compatability library interface

8.1 Using Existing Applications

The Prospero file system is presently implemented as a library. We provide versions of `cat` and `ls` which have been linked with the Prospero compatability library, which is an additional wrapper around the main Prospero library. The relinked versions may be found in the same directory as the other Prospero binaries, and will appear in your search path once the `vfsetup` command has been executed.

If the `venable` command (see section 8.1) has been executed to set the default name resolution mechanisms to the virtual file system, then the `cd` command may be used to change virtual directories.

If the `venable` command (see section 8.1) has been executed to set the default name resolution mechanisms to the virtual file system, then the `ls` command may be used to list virtual directories.

By default, names which are to be resolved using Prospero must contain or be preceded by a colon (`:`). File names that do not contain and are not preceded by a colon are treated as native Unix file names. The default behavior can be modified by setting the PFS_DEFAULT environment variable. This may be done by using the `venable` command. When enabled, names are resolved relative to the active virtual system by default. Names that are to be treated as native Unix file names must be preceded by an atsign (`@`). `vdisable` will return the default to its original state. The meanings of the values for the PFS_DEFAULT environment variable are listed in Table 2.

Table 2: Settings for the PFS_DEFAULT environment variable

Value	Meaning
0	Never resolve names within the virtual system
1	Always resolve names within the virtual system
2	Resolve names within the virtual system if they contain a :
3	Resolve names within the virtual system by default, but treat names beginning with an @ or full path names that don't exist in the virtual system as native file names
4	Resolve names within the virtual system by default, but treat names beginning with an @ as native file names

9 Filters

Client-side filters are written in C, compiled, and dynamically linked during name resolution. Because of portability problems with the dynamic linker, client-side filters are not included in this release, but are available upon request. There are also server-side filters, which are precompiled into the server. These are intended to be used by special applications; the only ones currently in general use are used by version 3 of Archie.

10 Setting up a new system

This section explains how to install the Prospero file system on a new system. It assumes that the local site has already been configured. This section (Section 10) can be skipped by most users.

10.1 Building the Binaries

[See the INSTALLATION file in the distribution]

10.2 Running the Server on Unix (like) Systems

If you are installing the Prospero server, a user and group ID must be established under which the directory server will run. The directory associated with the user ID should be a location in which additional information about virtual files and directories can be stored. New files which are to exist only within the Prospero file system will also be stored under this directory. It is suggested that you chose the user name *pfs* for this pseudo-user, but other names may be used as well.

Once the user ID has been set up, install the binaries. The directory in which they must be installed is selected at compile time. As originally distributed, it is */usr/pfs/bin*. The program `pstart` should be installed setuid and setgid the pseudo-user just described.

```
pstart [hostname]
```

To start the server run `pstart`. `pstart` takes an optional host name. If specified, the host name must be the primary name for the host on which the server is running. In most cases, the server is able to determine the name on its own and there is no need to specify it as an argument.

`pstart` will connect to the directory associated with the pseudo-user, it will check to make sure that the user id is set appropriately, and it will exec the directory server with the appropriate arguments.

Although it is not recommended, the directory server can also be started manually. You must first be logged in as (or be su'ed to) the user under whose ID you want the server to run. You can then execute `dirsrv` passing as arguments the required directory names.

```
dirsrv [-p#portnum] [-m] root shadow data aftpdir afsdir hostname
```

The *-p#* option allows one to specify an alternate port to run the server on. This alternate server can be reached with the hostname "your-hostname(portnum)". For instance, at ISI, we run a publicly accessible GOPHER-GW server at `ZEPHYR.ISI.EDU(1570)`. Common reasons for running additional servers on alternate ports are for testing reasons, to take some of the load off of your primary server, or to run a server dedicated to publishing a special database.[12]

The *-m* (manual) option prevents the directory server from dissociating itself from the terminal. It is only useful for debugging. *root* is the logical root of the system. Only files below this point (and those under aftpdir and afsdir) will be accessible through the Prospero file system. *shadow* is the name of the directory that is to contain additional information about files and directories. It should typically be the *shadow* subdirectory of the pseudo-user described above. *data* is the local directory under in which new virtual directories and their contents will be stored.

aftpdir is the name of the directory hierarchy to which anonymous FTP has access and *afsdir* is the name of the directory through which files from the Andrew File System may be accessed. If these access methods are not supported by your system, these arguments should be the null string.

Users can use the Prospero file system even if the server is not running, but they will be unable to access files or directories stored locally. If `pstart` is installed setuid and setgid to the Prospero user and group IDs, then the directory server can be started by any user. You may also want to start the directory server from the system's */etc/rc* file.

As things stand, users can access files and directories created on the local system, but they can not create new virtual systems stored locally. If you want to allow virtual systems to be stored locally, then you must have the site administrator add a reference to the new system from the *pfs_storage* virtual directory.

Adding References to the New System

The remainder of this section describes the actions that are to be taken by the site administrator. Systems that are running the release as distributed are part of the USC Information Sciences Institute guest site. The site administrator is *pfs-administrator@isi.edu*. This applies even if your system is running a server. If you are not a site administrator, you can skip this section.

If you want to allow virtual systems to be stored locally, several links must be added and a new virtual directory must be created. This will only be possible if the server has **not** been configured read-only. When a new site is established (see Section 11) these links and directories are automatically created on the primary system for the site. The following steps are only required when adding additional systems.

[12]Caveat: The common UNIX shells require you to quote the `#` in the *-p#* option in order to keep it from being interpreted as the start of a comment.

In the following steps, HOST is the fully qualified domain name for the host to be added and PATH is the full path of the subdirectory of the pseudo-user (described above) which will store the new virtual directories. The last component will typically be *pfsdat*. If that directory does not already exist, it should be physically created.

A link must be added from the virtual directory *pfs_storage* to the pfsdat directory on the new site. While still in the master virtual system, the following steps will add this link.

```
vcd /pfs_storage
vln -n HOST PATH HOST
```

You will next have to create the *local_vsystems* virtual directory. You do this by issuing the commands:

```
vcd HOST
vmkdir local_vsystems
```

10.2.1 Creating a Virtual File System

A virtual file system is created using the `newvs` command. The `newvs` command is for use by the site administrator[13]. To have a virtual system created, send a message to *pfs-administrator* at your site.

If you want to run a server but don't want the hassle of setting up your own site, we can arrange to have virtual systems stored on your server but have your server still be part of the ISI guest site. We recommend this option.

```
newvs [-v#] [-e] [host [name [home [owner [desc_file]]]]]
```

The `newvs` command is used to create a new virtual system. If called with no arguments, the user is prompted for the system on which the new virtual system is to reside, the name of the virtual system, the home directory, the owner, and the name of the description file.

The name of the virtual system is the default name with which it will appear in the local site's master list of virtual systems. This name is not automatically exported beyond the local site.

`newvs` will create the virtual system, assign a global name to it, and add the selected name in the master list of virtual systems for the local site. It will also copy the links from the *prototype* virtual system to the newly created one. The *-e* option will suppress this copying, and will leave the new virtual system empty. As a final step, `newvs` will optionally write a description file that may be read by `vfsetup`.

The *-v* option is followed by an integer and sets the verbosity level. The meaning of the verbosity levels are presented in Table 3.

[13] Systems which are running the release as distributed are part of the University of Southern California Information Sciences Institute guest site. The site administrator is *pfs-administrator@isi.edu*. This applies even if your system is running a server.

Table 3: Verbosity levels for newvs

Value	Meaning
0 (default)	Prompt for required input
1 (-v)	Explain what is required when asking for input
2	List each action taken
3	Stop before each step
4	Stop before each step and explain the action

11 Setting up a New Site

This section explains how to set up a new site. Systems that are running the release as distributed are part of the USC Information Sciences Institute guest site. This applies even if your system is running a server. If you would like to set up your own site, send a message to *info-prospero@isi.edu* to obtain the appropriate additional files. Unless you are setting up your own site, you may skip the remainder of this section.

Before you begin, you will have to obtain a global prefix that will uniquely identify the virtual systems registered at your site. A prefix may be obtained by sending a message to *pfs-administrator@isi.edu*.

The global prefix is part of the low level name for each virtual system. It should be thought of as an address. Users employ higher level names to specify virtual systems. Registering a prefix will allow objects created at your site to be named by others. Even if your site will not be reachable by any other sites, it is still important to register a prefix. Doing so guarantees that no other site has the same prefix. This will make it possible to connect with the rest of the global system should a connection ever be established. More information on setting up an isolated site is described in Section 11.2.

11.1 Setting up the Master Directories

To initially set up a new site, run the command `newpsite`. `newpsite` will construct a skeletal site configuration based on the compile time options described in the previous section.

Once the site has been set up, it will be necessary to create additional directories and add them to the prototype on which new virtual systems will be modeled.

11.2 Setting up an Isolated Site

As was already mentioned, even if your site will be isolated from others, you should still try to register a unique global prefix.

If your site will be isolated from other sites you will have to set up a replica of the global root. This directory must be reachable with the name "#" from your site's master list of virtual systems. To create a replica of the global root, create a directory and add nested subdirectories corresponding to each component of your sites global prefix. The last entry should be a link to your site's master list of virtual systems.

11.2.1 If You Cannot Register a Prefix

If it is not possible to contact *pfs-administrator@isi.edu,* then it may still be possible to generate a unique global prefix based on a unique identifier assigned by another authority. Right now, the only names that may be turned into unique global prefixes are officially registered Internet domain names.

Internet Domain Names. If you have an officially registered Internet domain name, it may be turned into a global prefix by reversing the order of the components, replacing the periods with slashes, and prepending "#/INET/".

```
ISI.EDU      =      #/INET/EDU/ISI
```

12 Glossary

conventional link. A conventional link is similar to a hard link in the Unix file system. It maps a name for an object to the information needed to access it.

filter. A filter is a program attached to a link. A filter can modify the results of directory queries where the path from the root of the virtual file system to the queried directory passes through the filtered link.

global file system. The global file system is the collection of links and directories that make up the virtual file systems accessible to the user. The links and directories form a generalized directed-graph.

link. A link is either a conventional link or a union link, with or without an attached filter.

local system. The local system is the physical system to which a user is logged in, on which processes execute, or on which files are stored.

master directory. A master directory is a directory maintained at the site level, and included through union links as part of the corresponding directory in multiple virtual systems.

site. A site is a collection of virtual systems administered by a particular organization. An important characteristic of a site is that its virtual systems contain prominent references to the other virtual systems that are part of the site.

union link. A union link is a link to a directory that causes the links that are part of the linked directory to appear as part of the directory containing the union link. A directory's contents are the union of the set of conventional links it contains and the contents of all directories included through union links.

view. A view is a mapping from names to objects. Name spaces, parts of name spaces, and individual directories all define views. Because it specifies a name space, a virtual system also imposes a view. It is possible for more than one virtual system to impose the same or similar views.

virtual directory. A virtual directory is a directory in a virtual file system. The contents of a virtual directory might be calculated at the time the directory is queried by applying filters or expanding union links.

virtual file system. A virtual file system is the file system part of a virtual system. It consists of a root directory and all the files and directories that can be reached by traversing 0 or more links. The virtual file system is a projection of the global file system as viewed from the selected root.

virtual system. A virtual system is a distributed system that is assembled from the files, processors, services, applications, users and other components available over a global network. The owner of a virtual system identifies the components of interest, and assembles them into a virtual system by assigning names.

13 Quick Reference

Commands affecting the Prospero file system:

This reference sheet is not as up to date as the rest of this manual. Sorry.

```
    vfsetup [-n host path , [-r,v] name , -f file]
    vcd [-u] path
    vwd
    vls [-v] [-u] [-f] [path]
    vln [-u] [-s] [-e] [-n host1] name1 name2
    vmkdir directory
    vrm link
    vget virtual-file [local-file]
    padmin [ -motd | -kill | -restart | -set parameter | -get parameter
| -command ] [ server ]
    newvs [-v#] [-e] [host [name [home [desc_file]]]]
    list_acl [-d dir] [link-name]
    set_acl [-asirKE,-n,-N] [-t type] [-d dir] [-l link] rights principals
    vdisable ( not at all installations)
    venable  ( not at all installations)
```

Meanings for PFS_DEFAULT:

The meanings of the values for the PFS_DEFAULT environment variable are:

Table 4: Settings for the PFS_DEFAULT environment variable

Value	Meaning
0	Never resolve names within the virtual system
1	Always resolve names within the virtual system
2	Resolve names within the virtual system if they contain a :
3	Resolve names within the virtual system by default, but treat names beginning with an @ or full path names that don't exist in the virtual system as native file names
4	Resolve names within the virtual system by default, but treat names beginning with an @ as native file names