Appendix D

The Prospero Protocol

This appendix describes the Prospero protocol. Communication with directory servers uses a reliable datagram protocol described in Appendix E. Requests and responses are human readable commands and multiple commands may be sent in a single message.

Prospero is implemented on top of a datagram protocol to reduce the overhead that would otherwise be incurred when establishing connections to multiple directory servers. The decision to use a datagram protocol directly, instead of through a higher level mechanism such as remote procedure call, was made for reasons of portability. I did not want Prospero to depend on another protocol, software package, or other resource, unless that resource was almost universally supported by every computer on the Internet.

The use of humanly readable commands in the protocol has several advantages. It eliminates problems with byte ordering, and it makes future changes or additions to the protocol easier to incorporate while maintaining compatibility across versions; new commands or additional options to existing commands can be easily added.

The ability to request multiple operations in a single message improves performance, and it provides a simple way to request that a collection of operations (on a single server) be performed atomically.

D.1 Commands

Commands are separated by the <LF>. Items sent in response are separated by <LF>s, commas, spaces, or tabs. Characters (or null fields) can be quoted using single quotes ('). While inside quotes, a quote can be included by doubling it. If multiple options are to be specified for a single command, the options are separated by a "+".

D.1.1 VERSION <version-number> <software-id>

This command requests or specifies the protocol version number. If the specified protocol version number is not supported, the response will be:

```
VERSION-NOT-SUPPORTED TRY xx-yy,zz
```

Where xx-yy and zz lists those versions that are supported. If no arguments are supplied (or if the argument is not a number), then the VERSION command will return the current protocol version number being used by the server in the form.

```
VERSION <number>
```

The software identifier is optional, but if specified, it identifies the software and version of the software that is generating the request.

D.1.2 AUTHENTICATOR <type> <authenticator>

This command authenticates the principal making the request. The type is the type of the authenticator. It might be password, a Kerberos authenticator, or data used by an alternative authentication mechanism.

D.1.3 DIRECTORY <id-type> <object-id> [<version> [<magic-no>]]

This command specifies the directory to be read or modified by the commands that follow as part part of the same request. This command does not generate a response unless the selected directory is not a directory. In that case, the response:

```
NOT-A-DIRECTORY (being phased out) or
FAILURE NOT-A-DIRECTORY
```

is returned, and the remainder of the request ignored.

D.1.4 ATOMIC

This command specifies that all commands that follow are to be completed atomically. If one of the commands fails, then none of the commands are to have an affect. This may require undoing the affects of commands which have already completed. Note that this command works only for requests issued in the same message, and all commands must be executable on the single server to which the message was sent.

D.1.5 LIST <options> COMPONENTS <optional-component>

LIST is used to look up information stored in a directory. <optional-component> is the name of the component relative to the directory specified in the DIRECTORY command. The optional component can include wildcards, or can include multiple names separated by a space. If a space is to be included in a name, the name must be quoted.

Multiple component names are allowed. The multiple components within a single name are separated by the slash (/). If the result of the lookup of one component is another directory at the same storage site, then the directory server will repeat the process and look up the next component. When a lookup results in a dead-end, or a directory at another site, the server will return an entry indicating which components remain to be resolved by the client.

<options> contains a list of options to the LIST command. Among the options supported are EXPAND which tells the remote directory server to expand any union entries in the directory. By default, the response will contain the names of union links to be included, and the client must submit a new query. A directory server can ignore the EXPAND option to the list command if it so desires. The LEXPAND option is the same as EXPAND, but indicates that the server should only expand union entries for local

directories. The LEXPAND option is implied if a multiple component name is being resolved.

The VERIFY option tells the remote server that the purpose of the query is to verify whether the remote directory exists and is a directory. No components are to be returned.

The ONECOMP option tells the remote server that the component name should be treated as a single component. This option is required only when a component of a name includes a slash.

The ATTRIBUTES option indicates that the attributes associated with the link are to be returned. ATTRIBUTES must be the last specified option and is followed by the names of the attributes to be returned as additional options (i.e. they are separated by the +). If no attributes are listed, then all attributes are returned. If this option is specified, but the object resides on a different host, then the attributes stored with the link are returned, not those stored with the object referenced by the link.

On failure, a standard error responses will be returned. On success, the response will be a sequence of entries containing information about the requested files.

```
LINK <L/U/R/P> <type> <component> <host-type> <host-name> <id-type>
     <object-id> <version> <magic-no> (opt: DEST-EXP <dest-expiration>
     LINK-EXP <link-expiration>)
```

An UNRESOLVED response lists the components of the name that must be further resolved relative to the returned link.

```
UNRESOLVED <components>
```

If a filter is returned, it applies to the most recent link or union link that was returned.

```
FILTER <filter-type> <filter-host-type> <filter-host-name>
       <filter-id-type> <filter-object-id> <version> <magic-no>
       (opt: ARGS <filter-args>)
```

The types of filters are: D=DIRECTORY, filter is applied to the current directory; H=HIERARCHY, filter is applied to all directories reachable through the filtered link; O=OBJECT filter is applied to an object other than a directory; U=UPDATE, filter is applied when updating the directory. If union links are to be expanded, filters are normally applied after the expansion. A lower case type code (d, or h) indicates that the filter is to be applied before any union links are expanded.

The optional list of arguments to a filter is a single string. If there are multiple arguments, then they are separated within the string by spaces. If there are multiple arguments or if args contain special characters, then the list of arguments must be quoted so that it will be passed as a single string.

If attributes were requested, the link will be followed by lines that specify the values of the requested attributes. The form of the response will depend on whether the attribute is associated with the link, or with the actual object. If the attribute is associated with the link, the applies-to field indicates whether it applies to the LINK or the object, and if the object whether it is a CACHED attribute, a REPLACEMENT attribute, or an ADDITIONAL attribute. In case of conflicts between attributes associated with the link and those associated with the object, a cached attribute is superseded, a replacement attribute takes precedence, and an additional attributes leaves both intact.

```
OBJECT-INFO <attribute> <type> <value> <optional-info>
LINK-INFO <applies-to> <attribute> <type> <value> <optional-info>
```

If the value of an attribute is a link, the link might itself have attributes. When returned the names of such sub-attributes are enclosed in parenthesis to the appropriate nesting level. If the name of an attribute is to begin with an open parenthesis, the opening parenthesis must be quoted.

The order in which attributes are returned is significant. Some attributes might span multiple lines in which case, each line might appear as a separate attribute. This does not preclude the inclusion of a multi-line value in a single attribute if the value is appropriately quoted, but the existing implementation does not presently support such a representation.

If no files are found, the reply will be:

```
NONE-FOUND
```

D.1.6 LIST-ACL <options> <optional-component>

This request is used to list the access control list for the directory specified in the previous DIRECTORY command, or for a link within the directory. The optional component is required only if requesting the ACL for a link within the directory (option = LINK). It is to be left out when requesting the ACL for the directory itself (option=DIRECTORY). The response will be zero or more lines for the form:

```
ACL <entry-type> <authentication-type> <rights> <principals>
```

D.1.7 GET-OBJECT-INFO <requested-attributes> ID <id-type> <file-id> <version> <magic-no>

This command requests information about an object. <requested-attributes> is a list of those attributes that are desired. Valid attributes include ACCESS-METHODS, CLOSURE, DESCRIPTION, FORWARDING-POINTER, KEYWORDS, LAST-REFERENCED, LAST-WRITER, LOCKS, OWNER, REPLICAS, SIZE, STORAGE-LOCATION, TTL, TTL-EXPIRES. VERSION-NUMBER, VIRT-SYS, WELL-KNOWN-NAMES, and WRITE-DATE. ALL indicates that all attributes are to be returned. Multiple attributes may be separated by commas (with no intervening white space). The above are reserved attributes. User defined attributes are also allowed and are accessed the same way.

The response can be multiple lines, each containing a value for an attribute. The form of the response will be.

```
OBJECT-INFO <attribute> <type> <value> <optional-information>
```

The order in which attributes are returned is significant. Some attributes might span multiple lines, in which case each line might appear as a separate attribute. This does not preclude the inclusion of a multi-line value in a single attribute if the value is appropriately quoted, but none of the existing implementations can deal with such a representation.

D.1.8 EDIT-OBJECT-INFO <id-type> <object-id> <version> <magic-no> ATTRIBUTE <attribute> <optional-type-modification> <type> <value>

This command is used to change information about a file. <attribute> is the attribute about the file that is to be changed. Some attributes involve adding a new instance, deleting an instance, or replacing an instance. In these cases, the <optional-type-modification> will be either ADD, DELETE, or REPLACE. The new value is <value>.

D.1.9 CREATE-LINK <L/U> <component> <link-type> <host-type> <host-name> <id-type> <object-id> <version> <magic-no>

This command creates a new link in the current directory. If the link is a union link, then the component name may be null. If filters or other information must be added, the MODIFY-LINK command should be used once the link has been created.

D.1.10 DELETE-LINK <options> <component> (opt: NUMBER <n>)

This command is used to remove an entry from a directory. Depending on the value of the LINK-EXP, this may or may not actually delete the entry. If it does not, the entry is flagged for deletion once the link expires. The possible values for <options> are shown in table D.1.

The optional NUMBER flag can be followed by the number of the link to be removed. This is necessary if there are multiple links with the same name; it allows the desired link to be specified. NUMBER is only valid with the LINK and ULINK options.

Option	Meaning
VLINK	Virtual link (either conventional or union) with given component name will be removed.
LINK	Same as VLINK, but only conventional links are checked.
ULINK	Same as VLINK, but only union links are checked
ANYLINK	Link with given name will be removed, even if it is in the native directory. If so, the contents of the file may be lost, and other links to the file may become dangling references.

Table D.1: Options for the delete command

D.1.11 MODIFY-LINK <component> <attribute> <optional-type-modification> <value>

This command modifies information associated with a link. <attribute> is the attribute to be modified and includes FILTER, DEST-EXP, LINK-EXP, HOST-TYPE, HOST-NAME, ID-TYPE, and FILEID. <optional-type-modification> may be ADD, DELETE, or REPLACE for those attributes taking multiple values. <value> is the new value of the specified attribute.

D.1.12 MODIFY-ACL <options> <component> <entry-type> <auth-type> <rights> <principals>

This command is used to modify an access control list for a directory, or for a link within a directory. The directory is specified in the previous DIRECTORY command. The component is null (but quoted) if modifying the ACL for the directory (option= DIRECTORY). Options indicate whether the ACL to be changed is for a DIRECTORY, or a LINK. Another option indicates the operations to be performed (one of ADD, SUBTRACT, INSERT, DELETE, SET or DEFAULT), and whether to override the automatic inclusion of the system ACL (NOSYSTEM), or administer access for the client (NOSELF).

D.1.13 CREATE-OBJECT <options> <type> <component> ATTRIBUTE <attributes>

This command will create an object with the specified name and will return the identifier that can be used to open it. If a file, it is created empty. <attributes> allows the creator of a file to specify many of the attributes of the object at creation time. The form for each attribute would be the same as in EDIT-OBJECT-INFO.

D.1.14 CREATE-DIRECTORY <options> <component> ATTRIBUTE <attributes>

This command will create a directory with the specified name. The directory is created empty. <attributes> allows the creator of a directory to specify many of the attributes of the directory at creation time. The form for each attribute would be the same as in EDIT-OBJECT-INFO.

<options> tells the directory server what else is to be done when the directory is created. If options is VIRTUAL, then <component> is a name relative to the current directory, and a link to the new directory is added to the current directory. If <options> is PHYSICAL, then component is an absolute pathname relative to the root, and no link to the new directory is created. It is up to the application to add the link.

The access control list for the new directory will be a copy of the access control list for its parent. An entry will be automatically added to the ACL granting its creator all rights. If the LPRIV option is specified, only those rights needed to allow the creator to set up the directory (list, read, insert, and administer) will be added, and only if the creator does not already have such rights for the parent directory.

D.1.15 UPDATE <options> COMPONENTS <optional-component>

This command tells the server to check each of the named components in the current directory for forwarding pointers. If the referenced object has moved, the link will be updated. If <optional-components> is empty, all components in the directory will be checked. On success, the UPDATE command returns the number of links that were modified.

```
UPDATED <number> links
```

D.1.16 STATUS

This command requests the current status of the server. A humanly readable multiple line response is returned. The response may be presented to the user without additional processing. The response must conform to the following requirements so that it may be read by a program if desired.

The first line must include the server's software version identification enclosed in parenthesis, and the host name of the server. The name of the host should be the name that appears on local links generated by the server, it might not be the primary name of the host. The version identifier must be the first string that appears in parenthesis, and the host name must be the string that immediately follows the version identifier.

If a line contains a colon (:), the string preceding the colon identifies the meaning of the text that follows the colon. Reserved identifiers include Contact, Started, Memory, Data, Root, AFTP, AFS, and DB. The identifiers are case insensitive. If present, AFTP identifies the part of the file system accessible by anonymous FTP.

Other than for the first line of the response, implementations are free to add or modify lines that do not contain a colons. A sample status response follows:

```
Prospero server (Beta.4.2B) JUNE.CS.WASHINGTON.EDU
Requests since startup 4096 (3609+377+2 67+29+0 9 0+0+0 0 3) OF
Started: 26-Aug-91 15:14:56 UTC
Contact: bcn@cs.washington.edu
 Memory: 0(118)vl 0(4)at 0(5)acl 0(1)fi 1(1)pr 2(2)pt 0(711)str
   Data: /u1/vfs/pfsdat
   Root: /
   AFTP: /homes/june/ftp
```

D.2 Standard Responses

D.2.1 SUCCESS <identifying-info>

A command that does not generate any output returns this response if successful.

D.2.2 FORWARDED <host-type> <host-name> <id-type> <object-id> <version> <magic-no>

This response is returned when the object or directory that is the target of an operation has moved. The client can retry the response using the corrected information.

D.2.3 ERROR <text>

This response is returned to indicate an error encountered when parsing the request. The error might be a protocol error, or it might be the result of the server's inability to recognize a keyword or data type. <text> describes the error.

D.2.4 FAILURE <identifying-info> <text>

This response is returned when an operation can not be performed. The defined values for <identifying-info> appear below. <text> is optional text that provides additional information about the failure.

```
NOT-FOUND [FILE,ACL]
ALREADY-EXISTS
NAME-CONFLICT
AUTHENTICATION-REQUIRED
NOT-A-DIRECTORY
NOT-AUTHORIZED
SERVER-FAILED
```

D.2.5 WARNING <identifying-info> <text>

This response is returned to indicate a warning condition which does not affect the correctness of the response. This message can be used to indicate that the client is using an old version of the protocol that, while supported, should be phased out. It can also be used to inform the client of future changes on the server or scheduled downtime. The defined values for <identifying-info> appear below. <text> is optional text that provides additional information about the warning.

`OUT-OF-DATE`

`MESSAGE`

Appendix E

Reliable Datagram Protocol

This appendix describes the reliable datagram protocol (RDP) used by Prospero. As used by Prospero, this protocol is layered on top of the Internet User Datagram Protocol [Postel 80].

Prospero implements its own RDP because I was unable to find any that were suitable for general use. Most systems that use an RDP implement their own around the specific needs of their application. Like these other systems, early versions of the Prospero protocol defined the mechanisms needed for retries and packet sequencing. As these mechanisms were refined, the functionality was moved to a separate protocol layer to improve modularity, and in hope that this general and simple RDP can be used for other purposes.

This RDP was designed so that in the common case, the additional overhead of guaranteeing reliability is as small as possible. Unless special processing is required, the header is kept small, and unless a packet is lost, no additional packets are sent. If a field is not specified, the default value is used in its place. All fields up to and including the last field specified must be filled in, but the header may be truncated at any point from which all remaining fields are the default values. The RDP header contains fields described on the next two pages.

Bytes 0 Version and header length: High order two bits are RDP version number mod 4 (this is version 0). Low order six bits are the header length including byte 0.

Bytes 1-2 Connection ID: Defaults to the expected connection ID. A specified value of 0 means use the default.

Bytes 3-4 Packet number: Defaults to 1 if not specified. A specified value of 0 indicates an unsequenced control packet which should not be passed to the application.

Bytes 5-6 Total number of packets: Defaults to 0 if not known, or retains current value if it was provided in any earlier messages. If the packet number was also not specified, then it defaults to 1. A specified value of 0 means use the default.

Bytes 7-8 Received through: Total number of packets successfully received by peer. Defaults to current value if specified in previous message. Defaults to 0 otherwise. If a value of 0 is specified, then it means reset to 0 (i.e. it forgot the earlier messages).

Bytes 9-10 Backoff (expected time to next packet): Defaults to current value. Specified value of 0 means use default. This is a hint. The client can use a different timeout if it has reason to believe messages are available which have been missed (e.g., gaps in the list of received packets). In any case, the client is always free to use a timeout longer than that specified here.

Bytes 11-12	Flags: Byte 11 is a bit vector, the high order bit (7) meaning please acknowledge and the next bit (6) indicating that the packet is a sequenced control packet only and should not be returned to the application by the rdgram library. The low order bit (0) means that a protocol ID for a higher level protocol follows. The next bit (1) means that a 2 octet priority is specified. Byte 12 specifies 1 of up to 256 other flags. The flags may themselves require additional fields specific to the flag. These fields appear at the end of the header in the order they are needed when reading flags from the low order bit to the high order bit, followed by any extra fields needed by the flag specified by the 12th byte. A value of 255 for the 12th byte is reserved for future expansion.
Bytes 13 and above	Fields specific to particular flags
Next 2 bytes	Protocol ID (if protocol id flag specified): These bytes identify the interpretation of the data carried in the packet. The default and a value of 0 mean that it is not specified, but has been agreed upon externally (i.e. the applications know).
Next 2 bytes	Priority (if priority flag specified): These bytes are a signed integer representing the priority of the request. Not all implementations understand this message, and many that do will not honor requests for expedited handling. Negative numbers indicate expedited handling while higher numbers indicate greater delays. A priority of 0 is normal.

Bibliography

[Postel 80] J. B. Postel. User datagram protocol. DARPA Internet RFC 768, August 1980.