Network Working Group
Request for Comments: 1249

T. Howes
M. Smith
B. Beecher
University of Michigan

August 1991

DIXIE Protocol Specification

Status of this Memo

This RFC defines a mechanism by which TCP/UDP based clients can access OSI Directory Service without the overhead of the ISO transport and presentation protocols required to implement full-blown DAP. This memo provides information for the Internet community. It does not specify any standard. Distribution of this memo is unlimited.

Table of Contents

1. Introduction

OSI Directory Service defines a powerful mechanism for storing and retrieving information about objects, and for arranging those objects in a hierarchical structure. Many types of objects and information can be stored in The Directory, including white pages information, application information, service information, etc. The OSI protocol defined to allow access to this information is the Directory Access Protocol (DAP). The DAP, being an OSI application-layer program, is fairly heavy-weight and requires a substantial amount of computing power and coding investment to implement.

The DIXIE protocol is designed for use by smaller hosts (e.g., Macintoshes and PCs) that do not have the computing power or necessary software to implement a full OSI protocol stack. The DIXIE protocol is also useful for any Internet application that wants a simple interface to X.500 that requires very little coding investment.

The basic idea behind DIXIE is the same as that described in RFC 1202 for the Directory Assistance Protocol. DIXIE offers both UDP and TCP access to The Directory. While the Directory Assistance Protocol exports something of a user interface, DIXIE provides a more direct protocol translation.

1.1 History

The DIXIE protocol has evolved over time, slowly growing into the protocol described by this document. Without an understanding of the circumstances surrounding this evolution, the wisdom of some of the DIXIE design decisions may not be apparent.

2. Protocol

This section describes the DIXIE protocol in detail. DIXIE follows a client-server request and response paradigm. Clients send request packets to a DIXIE server, and the server sends reply packets in return. Communication may be over UDP or TCP, depending upon the needs of the client. All modification operations (ADD, REMOVE, MODIFY, MODIFYRDN) must be performed over a TCP connection, which provides some level of authentication.

Whichever method of communication is used, the general packet format is the same. Each packet consists of a sixteen octet header followed by some data. The format of the header and data for each kind of request is described below.

The representation used for all X.500 data passed between the server and the client is the QUIPU EDB format. So, for example, a Distinguished Name might look something like "c=US@o=University of Michigan". For a complete description of this format, see volume 5 of the ISODE Manual.

The DIXIE server listens on port 96 for both UDP packets and TCP connections.

2.1 Header

The DIXIE packet header is sixteen octets long. For requests, the header is described by the following:

Start	Length	Description
0	1	An opcode specifying one of the operations described below. (see section 2.3 for a summary)
1	2	A request identifier to be included in the reply. This number should be unique to a request.
3	4	The total length of the request packet, excluding the header.
7	2	Unused.
9	1	Options. Currently, there are only three options. If bit 0 is set, "large" attributes will be included in the response. The choice of what constitutes large is up to the implementation. If bit 1 is set, the dereference aliases service control will be set for the X.500 operation. If bit 2 is set, aliases will NOT be dereferenced and searched during a search operation.
10	1	Protocol version. The current version is 1.
11	1	For the search operation, this byte specifies the scope of the search. (see section 2.2.2.1)
12	2	Timelimit in seconds for the operation.
14	2	Sizelimit for the operation (search and list).

For replies, the header is described by the following:

Start	Length	Description
0	1	A return code specifying either success or describing any error that occurred. (see section 2.4 for a description of each code)
1	2	The identifier included in the corresponding request packet.
3	4	The total length of the response packet, excluding the header.
7	3	Unused.
10	1	Protocol version. The current version is 1.
11	5	Unused.

All unused fields should be set to null octets and are reserved for future expansion.

2.2 Operations

This section describes the DIXIE operations, which closely parallel the X.500 DAP operations.

2.2.1 Read

The DIXIE read operation corresponds to an X.500 DAP READ operation.

2.2.1.1 Read Request

The header opcode should be set to 0x01. The data portion of the packet consists of the DN of the entry to read, a null octet, and then a null-octet separated list of attributes whose values are to be returned from the read. If no attributes to return are listed, all attributes are returned. The packet is terminated by two null octets in a row.

2.2.1.2 Read Reply

The reply data for the read operation consists of the entry read, followed by a null octet. An entry consists of the DN of the entry, followed by the octet 0x02, followed by a 0x02-octet separated list of attribute values. An attribute value consists of an attribute type, followed by the octet 0x01, followed by a 0x01-octet separated list of values. Each attribute type, attribute value and distinguished name has the form defined by the QUIPU EDB format.

2.2.2 Search

The DIXIE search operation corresponds to an X.500 DAP SEARCH operation.

2.2.2.1 Search Request

The header opcode should be set to 0x0f. Octet 11 in the header should be set to 0x01, 0x02, or 0x03, for a search scope of base object, one level, or whole subtree, respectively. The data portion of the packet consists of the DN of the entry from which to start the search, a null octet, a string containing the search filter (dish-style), a null-octet, and then a null-octet separated list of attributes whose values are to be returned from the search. If no attributes to return are listed, all attributes are returned. The packet is terminated by two null octets in a row.

2.2.2.2 Search Reply

The reply data to the search operation consists of two octets in network byte order specifying the number of matches returned. Next comes this number of sequences of the form: one 0x03 octet followed by one entry. Each entry is as described above in section 2.2.1.2.

2.2.3 List

The DIXIE list operation corresponds to an X.500 DAP LIST operation.

2.2.3.1 List Request

The header opcode should be set to 0x10. The data portion of the packet consists of the DN of the entry on which to perform the list, followed by a null octet.

2.2.3.2 List Reply

The reply data to the list operation consists of two octets in network byte order specifying the number of subordinates returned, followed by this number of sequences of the form: one 0x03 octet followed by a Relative Distinguished Name of a subordinate.

2.2.4 Modify

The DIXIE modify operation corresponds to an X.500 DAP MODIFY operation.

2.2.4.1 Modify Request

The header opcode should be set to 0x02. The data portion of the packet consists of the DN of the entry to modify, followed by a null octet, followed by a null-separated list of modify operations to perform. Each modify operation is one of the following:

type	remove attribute type
type=value	make value the sole value for attribute type
type+=value	add value to attribute type
type-=value	remove value from attribute type

The second form will see to it that existing values (if any) are deleted before the new ones are added. The third form will add the attribute type if it does not already exist. Note that the QUIPU EDB format, used to specify value, allows multiple values to be specified separated by the "&" character. This operation is only allowed over TCP.

2.2.4.2 Modify Reply

There is no reply data for the modify operation. The only indication of success or failure is the return code in the header.

2.2.5 Modify RDN

The DIXIE modify RDN operation corresponds to an X.500 DAP MODIFYRDN operation.

2.2.5.1 Modify RDN Request

The header opcode should be set to 0x13. The data portion of the packet consists of the DN of the entry to modify, followed by a null octet, followed by the new RDN the entry should have, followed by a final null octet. The old value of the RDN is never kept as an attribute of the entry. This operation is only allowed over TCP.

2.2.5.2 Modify RDN Reply

There is no reply data to the modify RDN operation. The only indication of success or failure is the return code in the header.

2.2.6 Add

The DIXIE add operation corresponds to an X.500 DAP ADD operation.

2.2.6.1 Add Request

The header opcode should be set to 0x11. The data portion of the packet consists of the DN of the entry to add, followed by a null octet, followed by a null-separated list of the entry’s attributes. Each attribute in this list has the form:

```
        type=value
```

where value can consist of a single value, or multiple values separated by the "&" character. The request is terminated by two null octets in a row. This operation is only allowed over TCP.

2.2.6.2 Add Reply

There is no reply data to the add operation. The only indication of success or failure is the return code in the header.

2.2.7 Remove

The DIXIE remove operation corresponds to an X.500 DAP REMOVE operation.

2.2.7.1 Remove Request

The header opcode should be set to 0x12. The data portion of the packet consists of the DN of the entry to remove, followed by a null octet. This operation is only allowed over TCP.

2.2.7.2 Remove Reply

There is no reply data for the remove operation. The only indication of success or failure is the return code in the header.

2.2.8 Bind

The DIXIE bind operation corresponds to an X.500 DAP BIND operation using simple authentication as defined in Recommendation X.509.

2.2.8.1 Bind Request

The header opcode should be set to 0x04. The data portion of the packet consists of the DN of the entry as which to bind, followed by a null octet, followed by the password of the entry as which to bind, followed by a final null octet. A null DN corresponds causes a bind as NULLDN to occur.

2.2.8.2 Bind Reply

The format of the bind reply packet depends on whether the operation was invoked over TCP or UDP. If the operation was invoked over TCP, there is no reply data. Success or failure of the operation is indicated by the return code in the packet header.

If the bind operation was invoked over UDP, the data portion of the reply packet consists of an Internet address in standard dot notation, followed by a 0x01 octet, followed by a decimal number (in text form), followed by a null octet. The address and number should be taken to be the IP address and port number to which the client should connect to obtain an authenticated TCP connection, bound as the entity specified in the request packet.

2.3 Operation Code Summary

This section describes the possible values for the DIXIE header operation code. There are currently 8 possible values:

0x01	Read
0x02	Modify
0x04	Bind
0x0f	Search
0x10	List
0x11	Add
0x12	Remove
0x13	Modify RDN

2.4 Return Code Summary

This section describes the possible values for the the DIXIE header return code. There are currently 17 possible values:

0x01	The request was successful.
0x02	The search did not find any matches.
0x03	Some unknown, generic DIXIE error has occurred.
0x04	The DIXIE opcode was not recognized by the DIXIE server.
0x05	Insufficient access to perform a modification.
0x06	A malformed DN was supplied.
0x07	Some time limit or size limit was reached. Partial results will be returned.
0x08	A modify was attempted before a bind.
0x09	A fragment requested was not found.
0x0a	An attribute type specified is invalid.
0x0b	An attribute specified does not exist in the entry.
0x0c	An attribute value specification is invalid.
0x0d	An attribute value does not exist (as for removal of the

	value).
0x0e	A modification of an entry’s RDN was attempted via a modify operation. This is not allowed (use modrdn instead).
0x0f	A supplied DN references an invalid portion of the tree.
0x10	The DSA has passed back a referral to another DSA (as for a modification to a non-local entry), and the DIXIE server was unable to follow it.
0x11	The DSA is down or unreachable.

3. References

[1] Information Processing - Open Systems Interconnection - The Directory, International Organization for Standardization, International Standard 9594, 1988.

[2] Kille, S., Robbins, C., Roe, M., and A. Turland, "The ISO Development Environment: User’s Manual", Volume 5: QUIPU, Performance Systems International, January 1990.

[3] Rose, M., "Directory Assistance Service", RFC 1202, Performance Systems International, February 1991.

4. Available Implementations

This section is not meant as an endorsement of any implementation, it is provided merely as information for the Internet community. A full Un*x-based implementation of the DIXIE protocol in the form of a DIXIE server and DIXIE application library is freely available for anonymous FTP from the host terminator.cc.umich.edu in the ˜ftp/x500 directory. Un*x and Macintosh clients that use the DIXIE protocol have also been implemented and are available from the same location.

There is also a discussion list for DIXIE-related topics called dixie@terminator.cc.umich.edu. To join, send mail to dixie-request@terminator.cc.umich.edu.

5. Security Considerations

Security issues are not discussed in this memo.

6. Authors' Addresses

Tim Howes
University of Michigan
Information Technology Division
535 West William St.
Ann Arbor, MI 48103-4943

Phone: +1 313 764-2278
EMail: tim@umich.edu

Mark Smith
University of Michigan
Information Technology Division
535 West William St.
Ann Arbor, MI 48103-4943

Phone: +1 313 764-2277
EMail: mcs@umich.edu

Bryan Beecher
University of Michigan
Information Technology Division
535 West William St.
Ann Arbor, MI 48103-4943

Phone: +1 313 764-4050
EMail: bryan@umich.edu