Chapter 9

NetWare Directory Services Module (NDS)
Design Specification

Abstract

NDS provides NetWare Directory Services support for Client32, including authentication, name resolution, Unicode support, and VLM compatibility.

Introduction

NDS is an NLM which contains all the Directory Services (DS) functionality for authentication, name resolution, Unicode support, and VLM compatibility.

Client32 NDS offers several significant new features:

- **Multiple tree support**. A user may now authenticate to more than one tree at a time.

- **Unicode support**. NDS will translate between the local code page and Unicode. This allows applications to use Directory Services without knowledge of Unicode (for example, a utility can pass a name to NDS in local-code-page format (USA-ASCII string) and NDS will convert it to Unicode so it can be used over the network).

- **Login and authentication done at the Requester**. All code for login and authentication has been moved to NDS. This is significant mainly to DOS applications, which will realize a savings in conventional memory when the large libraries are no longer statically linked into the login executable. (Statically linked DOS applications will need to be relinked with new SDKs to realize this conventional memory savings.)

- **NDS name resolution done at the client.** Name resolution will be done at the client when it is advantageous. This will take some of the burden off the server as well as allow the best partition to be used when a name is resolved to multiple addresses.

- **Caching of resolved name object ID and address information.** NDS will cache ID and address information for each relative distinguished name (RDN) within the user's distinguished name.

- **Simplification of connection licensing.** Licensing of a connection will no longer be tied to resource counts on a given connection, but will be administered on an NCP-by-NCP basis. See the *NDS Connection Licensing* chapter for a more complete description of licensing.

Design Description

NDS.NLM is designed to be portable to other operating systems. While certain key components within NDS.NLM may be implemented in assembly language to improve performance for the DOS, MS Windows, and Chicago platforms, equivalent C routines will be provided for portability to other operating systems.

In Figure 1 below, NDS.NLM is shown as two separate pieces to illustrate the multiplexing functions of the Authentication Multiplexor (AuthMux) and the Name Service Multiplexor (NSMux).

Directory Services is one of several name services supported by Client32, along with Bindery and PNW. NDS name resolution requests are funneled to NDS.NLM by NSMux and authentication requests are funneled by AuthMux.

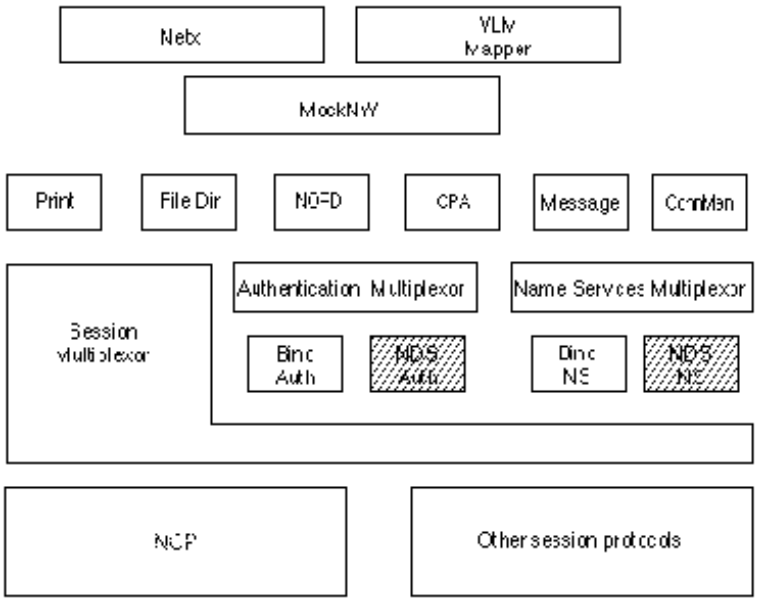

Figure 1. NDS has two major functional areas: authentication and name services.

Besides authentication and name service multiplexing, the NDS module must also handle VLM compatibility and

Unicode support. In addition, NDS.NLM exposes the API that is used by both the NS Mux and AuthMux. (See Figure 2.)

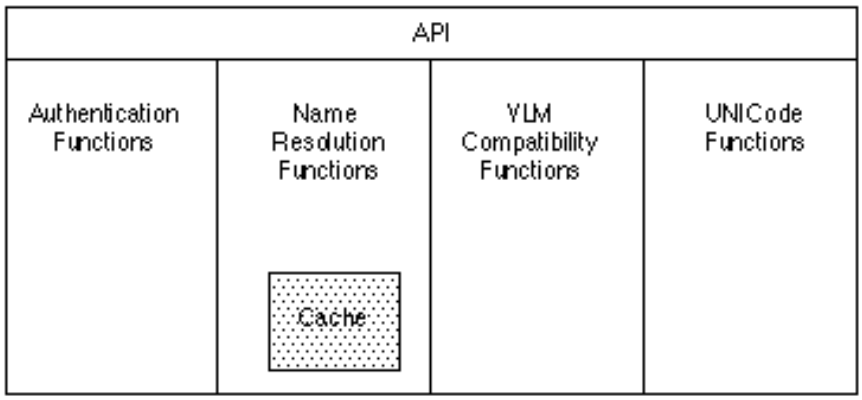

Figure 2. The four functional areas of NDS.NLM.

Authentication

Registration

NDS must register with AuthMux as the provider of authentication services for NetWare Directory Services. NDS registers using **AUTHRegisterSvc** and unregisters using **AUTHUnregisterSvc**.

NDS Connection Licensing

The current method of licensing connections is confusing and difficult to implement, resulting in various licensing problems like connections that never get unlicensed and excessive NCP license traffic. (See *NDS Connection Licensing*.)

Licensing is now done on an NCP-by-NCP basis. For a complete list of NCPs that require licensing, see Appendix 14A.

As NCP packets are sent, a series of checks will be performed on that connection to determine whether the license NCP needs to be sent.

API

Following is the set of APIs NDS must provide as an authentication service module to ConnMan.

API Name	Description
NDSCreateAuthenticationHandle	Allows a user to log in to a tree.
NDSGetAuthenticationInfo	Allows applications to determine who is logged in to which trees.
NDSAuthenticateWithHandle	Used after NDSCreateAuthenticationHandle to authenticate.
NDSAuthenticate	Authenticates a connection if the user has not first used NDSCreateAuthenticationHandle.
NDSUnauthenticate	Unauthenticates a connection.
NDSCloseAuthenticationHandle	Allows a user to log out from a tree.

Authentication handles are used by the libraries (DLLs) to tell the Requester when to authenticate a connection based on the API and resource the application is using. This allows multiple connections in the tree to be authenticated with the authentication materials created with **NDSCreateAuthenticationHandle**.

Name Resolution

NDS will perform Directory Services name resolution (tree walking) at the client when it is beneficial to do so. If NDS determines that it is resolving names over a slow link (modem), then names will be resolved at the server for that portion of the tree.

Algorithms

TBD

Caching

Caching of NDS-resolved name object ID and address information will be limited to the distinguished name of the logged-in user.

Registering with NSMux

NDS uses the Name Services Multiplexor APIs **NSMRegisterNameSvc** and **NSMUnregisterNameSvc** to register and unregister itself as the name service module for NetWare Directory Services.

API

Following is the set of APIs NDS must provide as a name service module to the Name Service Multiplexor.

API Name	Description
NDSGetPreferredTree	Returns the current preferred tree name for the process group and process ID.
NDSSetPreferredTree	Sets the current preferred tree name for the process group and process ID.
NDSResolveNameToAddress	Allows user-readable names to be resolved to a specific address
NDSResolveObjectToId	Resolves a given object name to an object ID and trasport address.

VLM Compatibility

NDS offers a set of compatibility APIs to allow full backward compatibility to VLMs. The following is a list of the data structures that are shared between the VLM interface and the new 32-bit library interface.

Trees
Preferred Tree
Default Name Context
Authentication Information (Tag Data Store)
Monitored Connection

Existing 16-bit applications that use Directory Services will work simultaneously with newer Directory Services applications that use the new 32-bit interface and multiple tree support.

These VLM Compatibility APIs allow any platforms (including 16-bit VLM) who have currently implemented Directory Services to the old library model to move to Client32. OS/2 is an example of such a platform.

NDSVLMWriteTDS
NDSVLMReadTDS
NDSVLMGetSetMonitored
NDSVLMSetUserObjectId
NDSVLMGetPreferredConnId

Unicode Support

TBD

Multiple Tree Support

An unlimited number of tree structures may exist in the system, though any tree structures that aren't being used are purged. To control the over-allocation of unused tree structures, the following LRU (least recently used) algorithm will be used:

A maximum of eight tree structures that aren't being used (that is, have no resources attached to them) will be maintained. These eight will be the eight most recently used structures. Every time a process terminates, NDS

will check to see if there are more than eight tree structures and purge all but the MRU eight.

Tree structures that have been set as *preferred* or that have been logged-in-to cannot be LRUed.

Tree structures that have been accessed by currently running applications cannot be LRUed.

This will allow tree information to remain available between applications and prevent over-allocation of unused memory.

NDS-Specific APIs

The following are the APIs needed to manage Directory Services that are not multiplexed by authentication or name services APIs.

NDSGetDefaultNameContext
NDSSetDefaultNameContext
NDSRequestReply

Configuration File Information

No new configuration parameters will be added for NDS in Client32. The configuration parameters are the same as they were for VLM.

Under the Client32 requester section, "*Netware DOS Requester",* the parameters are:

Preferred Tree The system default preferred tree for Directory Services.

Name Context The system default name context for Directory Services.

Deliverables

All files and information to build NDS.NLM.
All unit test files and information.