Chapter 3
NIOS Client NLMs

Client NLM Introduction 20

NIOS Client Model 20

OS-Specific NIOS Components 22
    OS-Specific NIOS API 22
    Mapper API 23
    Portable NIOS API 23
    Subset of NetWare OS API 24

Portable NIOS Components 24
    Portable Client Components 24
    LAN Drivers and Communications Modules 25
        Topology Support Modules 25
        Link Support Layer 27
        Internetwork/Sequenced Packet Exchange
            Protocols 27
        Transport Service Interface 28

NIOS Logging 29

Client NLM Introduction

A driving concept behind the design of the NIOS Client was creating modular, reusable, and platform-independent software. To that end, all NIOS Client modules are in an NLM executable format and can be loaded and unloaded as necessary.

NIOS Client system modules:

- Are dynamically loadable and unloadable.
- Use NLM executable format.
- Run exclusively in a 32-bit flat memory model.
- Allocate memory that is guaranteed not to move or be discarded.
- Are fully language enabled.
- Are configured by utilities that use a configuration file.
- Require little or no static configuration information from the user. For example, it is no longer necessary to supply the number of open IPX sockets. As more sockets are opened, IPX dynamically allocates more memory to handle the additional sockets.
- Are written in C and Assembly.
- Run on single-processor Intel 386/486/Pentium systems.

Also included in this chapter is a functional description of the new NIOS logging feature.

NIOS Client Model

This section describes the cross-platform NIOS client model as illustrated in Figure 3.1; the sections that follow discuss each component of the model.

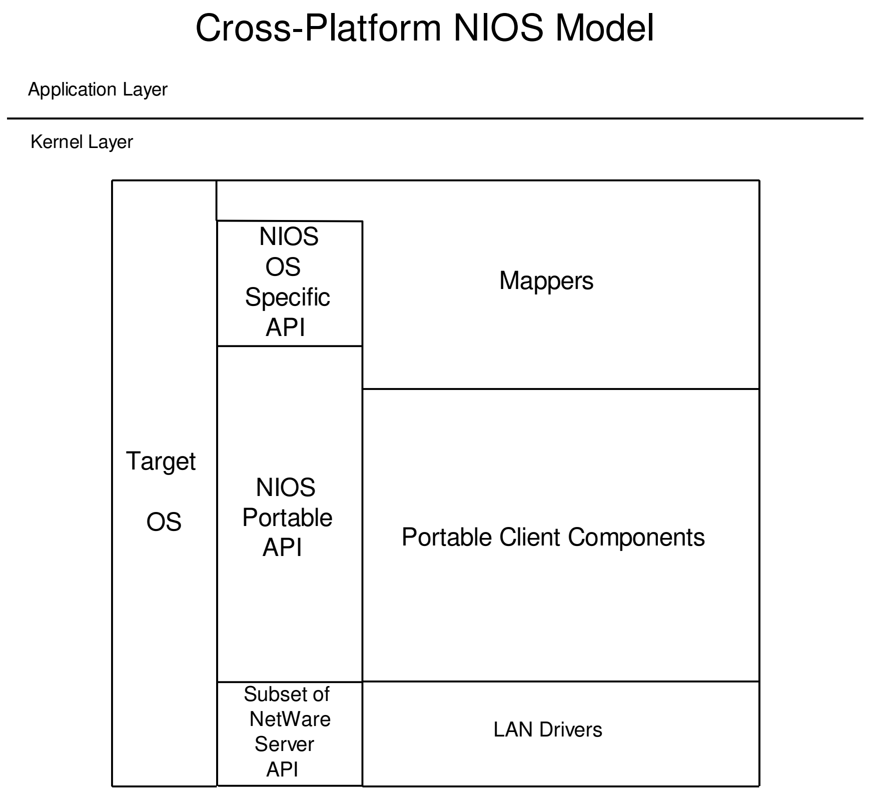

Figure 3.1: *The NIOS Client Model for cross-platform implementation. Common boundaries indicate modules that may communicate with each other.*

The Application Layer, also called the User Layer or Ring-3, provides services to network users at the application-based level. Relying on services performed at lower levels, the application layer is least involved with the underlying network hardware.

The kernel layer, also called Ring-0, is the general name for the core layer or more privileged level of the operating system.

The NetWare I/O Subsystem (NIOS) is the isolating layer between NetWare system-level components (core modules and various API mappers) and the host OS. As such, NIOS contains a full set of platform-independent (portable) APIs. These APIs provide a base from which powerful system-level functionality can be built.

Additionally, NIOS contains the Client NLM loader/unloader functions. These APIs are responsible for loading and unloading NLMs and LAN drivers.

NIOS implements a subset of NetWare OS APIs (called the NetWare OS Emulation module) that are necessary in order to use NetWare OS-compatible LAN drivers (LAN files) in the target OS environment.

Since NIOS components are divided into OS-specific and portable modules, we will next discuss briefly the OS-specific components and then move our attention to the portable NIOS components.

OS-Specific NIOS Components

Figure 3.2 identifies the OS-specific components of the NIOS client by shading them. The descriptions in this section are, of necessity, quite brief; see the design specifications written for each operating system to get more details.

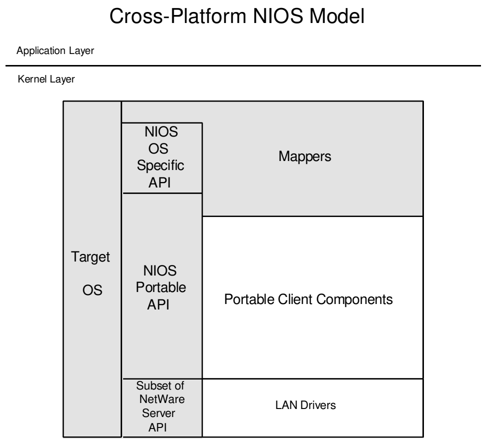

Figure 3.2: The shaded areas indicate the OS-specific components of the NIOS client.

OS-Specific NIOS API

The OS-specific NIOS API component provides a layer between NIOS Mapper modules and the Target OS, which enhances and adds additional functionality to the base OS. For example, the

DOS/MS Windows-specific NIOS API adds a protected-mode environment to DOS.

Mapper API

Mappers expose kernel level functionality to applicatiions. An example of a mapper API is the DOS 16-bit IPX/SPX far call interface.

Portable NIOS API

The portable NIOS API is the heart of the NIOS client providing many different types of services to portable components. These services fall into the following categories:

- Configuration Services
- Debug Services
- Event Services
- Handle Management Services
- Hardware Interrupt Services
- Information Services
- Linked List Services
- Module Management Services
- Memory Management Services
- Popup Video Services
- Process Management Services
- Returnable Memory Management Services
- Statistics Services
- Time/Date Services
- User Interface Services
- Utility Services

The NIOS API provides an abstraction of kernel OS services that are needed by NetWare's core client modules. Chapter 2 lists the service calls in each category; a detailed description of each API call is located in Chapter 6.

Subset of NetWare OS API

A subset of the NetWare 3.1x and 4.x servers' APIs is implemented on top of NIOS to allow the server's ODI LAN drivers to be loaded on the client.

The existence of the server API subset on the client does not suggest that the client will support server NLMs. Although the API supports the operation of NetWare LAN drivers, it does not duplicate all functionality or side-effects of the server's full API.

Portable NIOS Components

The portable components of the NIOS client, including LAN Drivers and other communications modules, make use of the portable NIOS API and the NetWare OS API subset to ensure their own portability. (See Figure 3.3).

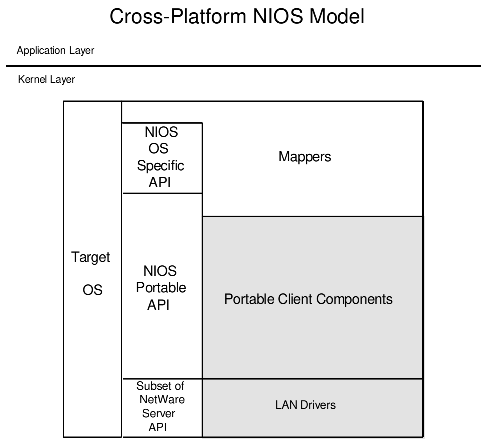

Figure 3.3: Shaded areas indicate portable components.

Portable Client Components

In order for the Requester to be as portable as posssible, Novell designed the following Requester components to be independent of any particular operating system, name service, authentication provider, session protocol, or transport protocol:

- Authentication Multiplexor
- Bindery
- ConnMan
- File and Directory Manager (FileDir)
- Message
- Name Services Multiplexor (NSMux)
- NCP
- NetWare Directory Services (NDS)
- NetWare Object Resource Database (NORD/CPA)
- Print
- Private and Global Resource Tracking
- Session Multiplexor

These portable components are described in detail in the *Portable Client32 Design Specification*.

LAN Drivers and Communications Modules

The client uses unmodified NetWare OS-compatible (3.x and 4.x) LAN drivers. This provides a huge pool of proven, certified LAN drivers for the client environment.

Plus, the burden on third-party LAN adapter manufacturers is greatly reduced since the NIOS Client does not introduce another LAN driver interface to which developers have to write.
Other portable modules that relate closely to the LAN Drivers are the following:

- Topology Support Modules
- Link Support Layer
- Internetwork/Sequenced Packet Exchange Protocols
- Transport Service Interface

Topology Support Modules

Topology Support Modules (TSMs) are components of the NetWare OS LAN driver architecture which provide an intermediate layer between the LAN driver and the Multiple Link Interface (MLI) of the LSL module. There is a separate TSM for each supported topology type (for example, ETHERTSM.NLM, TOKENTSM.NLM). (See Figure 3.3.)

Note: The MLI is an API that allows the LSL to remain independent of physical media.

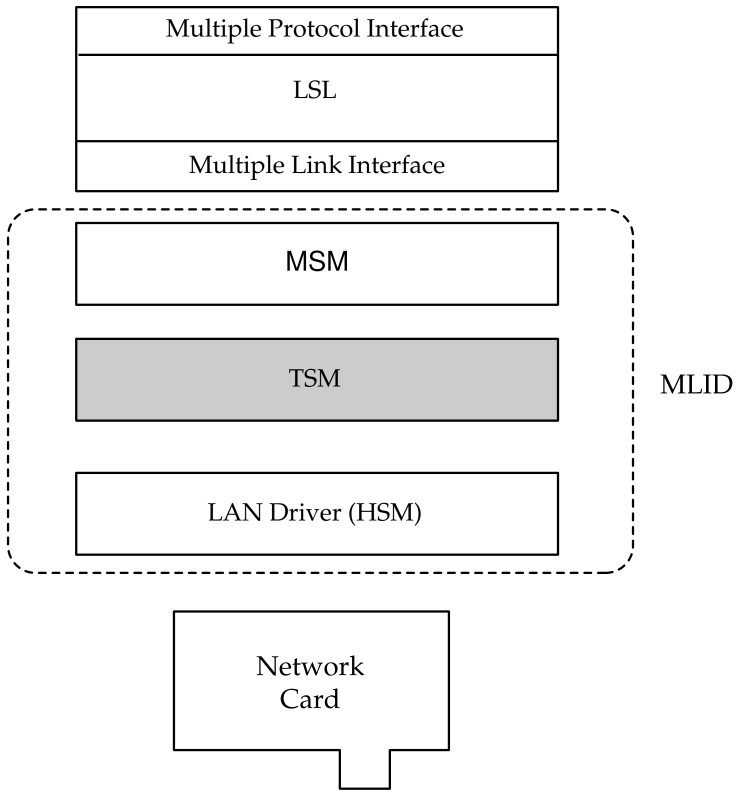

Figure 3.3: Topology Support Module

The client uses customized TSM modules which use a different packet-receive mechanism than the TSMs written for the NetWare OS. Instead of using the LSL buffer pool approach of the NetWare OS TSMs, the NIOS Client uses a Receive Look-Ahead approach, allowing protocols to preview packet header information and provide buffers directly for incoming packets. The Receive Look-Ahead method is much more efficient in a client environment than the traditional buffer pool method.

The client provides backward compatibility to the NetWare OS TSMs. A TSM written for the NetWare OS can be used on the NIOS Client.

Link Support Layer

The Link Support Layer (LSL) is the central component of any Open Data-Link Interface (ODI) implementation. (See Figure 3.4.)

The core of the NIOS Client's LSL is actually a ported version of the LSL used by the NetWare OS compliant with the ANSI "C" LSL interface. Because it does not use any OS-specific APIs, the unmodified core LSL can be used on other platforms supported by NIOS.

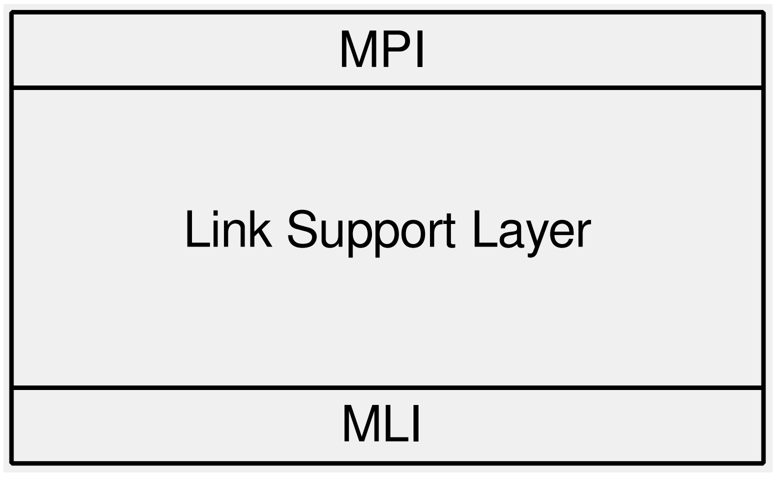

Figure 3.4: Link Support Layer

Internetwork/Sequenced Packet Exchange Protocols

The NLM (IPX.NLM) that implements the IPX and SPX protocols for the NIOS Client comprises a number of functional components. (See Figure 3.5.) The core IPX and SPX protocols are OS-independent and need not be modified to work on other NIOS-supported platforms.

The 16-bit DOS/ MS Windows IPX (IPXODI.COM) could bind to only one LAN driver at a time. The NIOS Client's IPX, however, is multiple-board aware; it can bind to more than one LAN driver at a time. While servers have always offered this capability by using internal routers, no such technology has been generally available on client workstations.

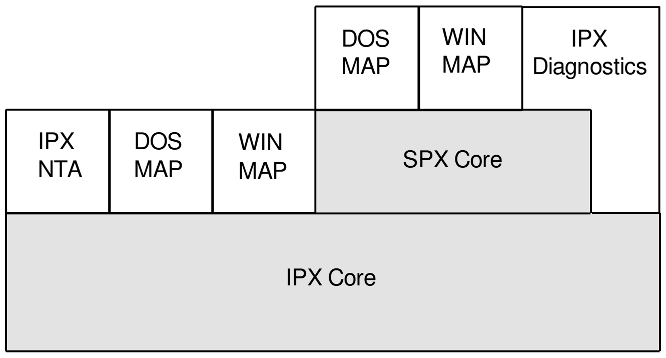

Figure 3.5: IPX.NLM architecture

The 32-bit IPX bypasses the need for cumbersome internal routers by demanding that client applications be multiple board-aware. An application queries IPX to find which boards are registered, and uses a new **GetLocalTarget** to select the best board on which to send and receive packets.

Built on top of the IPX API are four API mappers: two that emulate the 16-bit IPX/SPX APIs, and two that emulate the 16-bit MS-Windows IPX/SPX interface. (The MS-Windows 16-bit interface was previously provided by VIPX.386.)

Additionally, an IPX diagnostics module emulates the Diagnostic/GNMA functionality currently available with NetWare clients.

IPX.NLM also includes the IPX Transport Service Agent (TSA) that interfaces with the TRAN.NLM.

Transport Service Interface

In an attempt to evolve to an entirely transport-independent Requestor/Shell, the NIOS Client contains a transport independent layer called the Transport Service Interface (TSI). This layer is made up of a transport manager called TRAN.NLM and individual Transport Service Agents (TSAs). NTAs offer a consistent API to the Transport Service Users (TSUs) to each relevant transport protocol. (See Figure 3.6.)

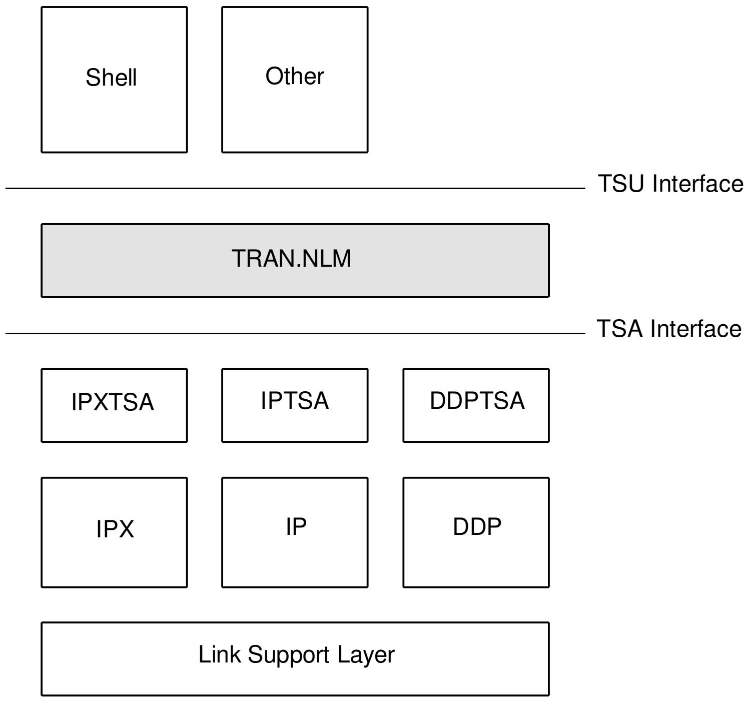

Figure 3.6: Transport Service Interface (TSI)

An example of a TSU is the NIOS Client Shell. Before the NIOS Client, NetWare Shells spoke only IPX. But the NIOS Client Shell runs on any transport. This is possible because the TSI masks protocol differences from the Shell. Instead of issuing an IpxSendPacket request, the NIOS Client Shell would send a generic SendPacket. The TSI would translate it into a protocol-specific request.

Each supported protocol will have its own TSA. A TSA is a mapper to a protocol. For example, the IPX TSA is linked with IPX.NLM to provide an indepentent transport interface for the IPX protocol.

NIOS Logging

The NIOS logging feature is a tool that enables developers and users to store or view diagnostic messages in a well known central location. NIOS logs messages to a file called NIOS.LOG in the NIOS system directory. The system directory is either the place where NIOS.EXE is loaded from; or, in the case of Windows 95, the *NwHomeDir* specified in system.ini [386Enh].

By default, logging is disabled. You can enable logging in the following ways:

1. Load NIOS.EXE with the /L option (DOS only).
2. Specifiy NwEnableLogging=TRUE in the system.ini [386Enh] (Windows 95).
3. Type *ENABLE LOGGING* from the DOS prompt after you load NIOS.
4. Call the **NiosEnableLogging** API function with the NIOS_LOG_ENABLE parameter.

You can disable logging in the following ways:

1. Logging is off by default.
2. Type *DISABLE LOGGING* at the DOS prompt after you load NIOS.
3. Call the **NiosEnableLogging** API with the NIOS_LOG_DISABLE parameter.

The maximum size of the logfile can be controlled with the *NIOS LOG MAX SIZE* field in the configuration database (the default is 64K).

If logging is enabled, the following types of messages will be time-stamped and logged:

- **NiosPrintf** with the MT_LOG_STATUS message type.
- **NiosPrintf** with the MT_DEBUG_OUT message type (if no debugger is present).
- **NiosDprintf**
- Items parsed from the configuration database (typically the NET.CFG).

You can suppress the time-stamp by using the MTF_NO_TIMESTAMP flag (not available with **NiosDprintf**).

Logging is available to all modules, including internal NIOS components. This service is not available at interrupt time; if call at interrupt time the message will be discarded.