

# **NIAKWA PROGRAMMING LANGUAGE**

## **RELEASE V COMPENDIUM**



1st Edition - August, 1998  
COPYRIGHT © 1998 Niakwa, Inc.

**Niakwa, Inc.**  
23600 N. Milwaukee Avenue  
Vernon Hills, IL 60061  
TEL: (847) 634-8700 FAX: (847) 634-8718 URL: <http://www.niakwa.com>

## **DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES AND PROPRIETARY RIGHTS**

The staff of Niakwa, Inc. (Niakwa) has taken due care in preparing this manual. Nothing contained herein shall be construed to modify or alter in any way the standard terms and conditions of the Niakwa Programming Language (NPL) Support and Distribution License Agreement, the End-User Support Only License Agreement, the Niakwa Software License Agreement and Warranty, or any other Niakwa License Agreement (collectively, the "License Agreements") by which this software package was acquired.

This manual is to serve as a guide for use of the Niakwa software only and not as a source of representations or additional undertakings by Niakwa. The licensee must refer to the License Agreements for Niakwa product and service representations.

No ownership of Niakwa software is transferred by any of the License Agreements. Any use of Niakwa software beyond the terms and conditions of the License Agreements, without the written authorization of Niakwa, is prohibited.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without prior written permission from Niakwa, Inc.

Niakwa, Niakwa Programming Language (NPL), Niakwa Integrated Development Environment (IDE), Niakwa Visual Interface Manager (Vinny), Niakwa Gateway to ODBC, Open NDM (Niakwa Data Manager), and Niakwa Workbench are trademarks of Niakwa, Inc.

All other trademarks are the property of their respective holders.



# PREFACE

## P.1 NPL Release V

NPL Release V is a pivotal new product in the continued evolution of NPL. This release introduces NPL developers to two significantly new products: the Niakwa Integrated Development Environment (IDE) and the new NPL 32-bit MS-Windows RunTime. The Niakwa IDE is designed to provide application developers with a complete set of 32-bit development tools, allowing developers to stay competitive in today's world of evolving 32-bit operating environments.

At the heart of the Niakwa IDE is the new NPL Workbench, a full featured 32-bit editor / debugger which provides developers with a more consistent project development environment than the traditional NPL line editor. By providing the NPL Workbench with appropriate 32-bit extensions to other Niakwa IDE components, the NPL Workbench becomes a robust single source development tool suited for all development projects large and small.

## P.2 The NPL Release V Compendium

Since Niakwa's release of its highly successful Niakwa Programming Language (NPL) Release IV product, many new features and enhancements have been introduced over the past few years, further strengthening the NPL Release IV product line. As with all products, numerous updates and enhancements over time tend to get lost in the multiple documentation updates that accompany an evolving product.

The NPL Release V Compendium is designed to eliminate these scattered updates by accumulating them into a single comprehensive document. In addition, the NPL Release V compendium documents new enhancements and syntax features which have been introduced since the last NPL release.

**NOTE:** Throughout this document, due care has been given to carefully note the NPL revision in which each update/enhancement was introduced. The NPL Release V Compendium is a replacement for the following supplementary document updates issued since the initial introduction of NPL Release IV.

NPL Revision 4.10 Addendum	March 94
NPL 4.10.02 Release Notes	March 94
NPL 4.10.16 Release Notes	July 94
NPL 4.10.23 Release Notes	January 95
NPL Revision 4.20 Addendum	November 95
NPL 4.20.07 Release Notes	November 95
NPL Revision 4.21 Addendum	June 96
NPL 4.21.05 Release Notes	June 96
NPL 4.22 Addendum	April 97
NPL 4.22.13 Release Notes	April 97

## P.3 Historical Time Line of Post NPL Release IV Updates

This section provides a time line of all post NPL Release IV revision updates and briefly describes the key features introduced in each subsequent revision, up to and including NPL Release V.

**NOTE:** Throughout this document, where appropriate, section titles are denoted with the revision numbers (in brackets) to appropriately identify when a specific feature update was introduced. The following sections include an illustration of this convention

### NPL Revision 4.10 [4.10]

- Released March 1994
- Introduced support for peer to peer (NetBIOS) environments such as Windows for Workgroups and LANtastic
- NetBIOS TSR (NIAKNETB.COM) introduced
- SHARE File access logic introduced
- MS-Windows RunTime Debug window introduced
- NetBIOS RunTime introduced
- Niakwa Networks RunTime introduced
- Security updated to handle NetBIOS considerations
- \$MACHINE bytes 32 and 33 introduced. These bytes provide developers with specific RunTime startup information to help verify proper workstation configuration in a NetBIOS or Novell environment.

### NPL Revision 4.10.16 [4.10.16]

- Released July 94
- Improved MS-Windows NetBIOS support
- NIAKNETW.DLL NetBIOS driver introduced

### NPL Revision 4.10.23 [4.10.23]

- Released January 1995
- Introduced MS-Windows 2227 driver support via new MXE=Y device clause
- \$OPTIONS byte 39 extended to provide improved SHARE file locking performance
- VLM extended printer support (more than 3 network printers) introduced
- /EXIT /MAIN subroutine handling improved

**NPL Revision 4.20** [4.20]

- Released November 1995
- Introduced support for Windows 95 and Windows NT
- Introduced new security
- \$DECLARE statement introduced
- Default NPL printer device name changed
- New Portrait and Landscape print enhancements added to MS-Windows RunTime
- \$OBJECT output buffer extended
- New BESDK callback support functions introduced

**NPL Revision 4.21** [4.21]

- Released June 1996
- Improved NPL “No Return Upgrade” procedure introduced
- Registration based user limit bumps introduced
- New RunTime startup search procedure implemented
- \$DECLARE statement enhanced
- New \$RUNDIR and \$ARGS RunTime system variables introduced
- Improved mixed RunTime co-existence on a single network
- MS-Windows RunTime A:\SETUP introduced
- Editor functionality enhanced
- NIAKNETB/NIAKNETW files updated
- NIAKNETB unload option implemented
- MS-Windows Demo RunTime introduced

**NPL Revision 4.22** [4.22]

- Released April 1997
- Universal Upgrade
- Enhanced A:\SETUP procedure introducing Network and Workstation setup options
- Line editor CUT/COPY/PASTE/UNDO introduced
- \$OPTIONS byte 57 introduced to implement CUA key mode support for cut and paste functionality

**NPL Revision 4.30** [4.30]

- Released August 1998
- MS-Windows A:\SETUP standardized for all RunTime platforms
- Universal Upgrade enhancements introduced
- New COPYALL batch file introduced for MS-DOS and 386/DOS-Extender RunTime installation
- LIST PUBLIC and LIST DIM statements extended
- Enhanced Boolean expressions added
- Improved Mixed Network Authentication implemented
- Default NIAKNETW value changed
- New /C command line option implemented for Instant Vinny support
- New RAW.DRV printer driver introduced
- Enhanced HALT key sequence in MS-Windows RunTime implemented
- \$DECLARE enhanced to support /POINTER parameters
- \$OPTIONS byte 38 extended to allow for more selective enforcement of explicit variable declarations
- New \$LDATE statement implemented to fulfill Y2K compliance
- New DATA LOAD/SAVE BU statements implemented
- New REDIM statement implemented
- New LIST PROC statement implemented
- READ DC and LISTDC extended with new LDATE restriction

**NPL Revision 5.0** [5.00]

- Released August 1998
- Windows 98 support introduced
- New NPL MS-Windows 32-bit RunTime introduced
- Niakwa Integrated Development Environment introduced
- Integrated 2227 asynchronous communications support in the new MS-Windows 32-bit RunTime
- Support for variables > 64K introduced in the new MS-Windows 32-bit RunTime
- \$DECLARE updated to support 32-bit calls from the new MS-Windows 32-bit RunTime
- RTIW.INI enhanced to handle both 16-bit and 32-bit specific sections

## P.4 Supported Operating Environments

NPL Release V is currently supported on the following 32-bit operating environments:

### Supported Standalone Systems

Microsoft Windows 95  
Microsoft Windows 98 \*\*  
Microsoft Windows NT 4.0

### Supported NetBIOS/Peer Environments

Microsoft Windows 95  
Microsoft Windows 98 \*\*  
Microsoft Windows NT 4.0  
Microsoft Windows NT Terminal Server \*\*\*

### Support Network Operating Systems

Novell NetWare 3.x and greater \*

### Supported Workstation Systems

Microsoft Windows 95  
Microsoft Windows 98 \*\*  
Microsoft Windows NT 4.0

For end-user sites that have not migrated completely to a 32-bit operating environment (i.e., MS-DOS and MS-Windows 3.x workstations still in use), Niakwa continues to provide its legacy 16-bit suite of RunTime programs with all NPL Release V Gold Keys. This allows you to provide a smooth integration of the 32-bit MS-Windows RunTime into your customers' operating environments.

- \* **All workstations running the NPL Release V RunTimes require Novell NetWare 32-bit Client Software to operate correctly on all Novell Networks. The Microsoft supplied MS-Client for Novell NetWare does not fully support all of the required NetWare API components required by NPL. Consult your Novell Network Administrator to insure that an appropriate Novell supported 32-bit client is installed on each workstation that requires NPL.**
- \*\* **Microsoft Windows 98 is tested and supported on the NPL Release V 32-Bit MS-Windows RunTime only. While Niakwa's limited testing has shown that our 16-bit RunTime products are operational and stable, any reported problems and subsequent corrections resulting from Windows 98 will only be applied to our 32-bit MS-Windows RunTime.**
- \*\*\* **Microsoft Windows Terminal Server is tested and supported on the NPL Release V 32-Bit MS-Windows RunTime only. While Niakwa's limited testing has shown that our 16-bit RunTime products are operational and stable, any reported problems and subsequent corrections resulting from Windows Terminal Server will only be applied to our 32-bit MS-Windows RunTime.**



## P.5 New and Upgrade RunTime Distribution

Release V of the Niakwa RunTime Package is distributed with either a new NPL MS-Windows Gold Key diskette or an NPL Universal Upgrade Package (indicated by a label on the installation guide cover). Instructions for installing a new NPL Gold Key are presented in Chapter 2 of the compendium. Instructions for installing an NPL Universal Upgrade are presented in Chapter 4.

**NOTE:**           **Upon completion of an NPL Universal Upgrade, refer to Chapter 2 for instructions on recalling and installing NPL security in the event you should ever need to move the NPL RunTime Replacement Gold Key Diskette to a new location.**

### P.5.1 What is the NPL Universal Upgrade?

The NPL Universal Upgrade is designed to allow existing NPL RunTimes to be upgraded to the most current NPL Revision in an easy manner. Upon completion of the NPL Universal Upgrade, the NPL RunTime Replacement Gold Key Diskette is fully licensed and replaces the old NPL Gold Key.

**NOTE:**           **Each NPL Universal Upgrade Package contains a new NPL RunTime Replacement Gold Key Diskette. The old NPL Gold Key being upgraded will be disabled during the final process of this procedure and should be disposed of to avoid future confusion with the newer NPL RunTime Replacement Gold Key Diskette. Refer to Chapter 4 for a complete discussion of the NPL Universal Upgrade.**

### P.5.2 Benefits of the NPL Universal Upgrade

The benefits of the NPL Universal Upgrade are as follows:

- The NPL RunTime Replacement Gold Key Diskette eliminates the need to maintain or return the old NPL Gold Key.
- The NPL RunTime Replacement Gold Key Diskette inherits all characteristics of the original NPL Gold Key (i.e., keeps the same serial and Gold Key numbers), and in many instances (depending on the upgrade purchased) provides additional functionality.

- The NPL RunTime Replacement Gold Key Diskette replaces security used by the older (pre 4.2x) NPL Gold Keys with a more stable form of security, allowing operation under Windows 95, Windows 98, Windows NT, and future 32-bit operating environments.
- Once the new NPL RunTime Replacement Gold Key Diskette is in place, future enhancements such as increasing the user license and adding additional NPL platforms can be done in place without requiring a new Gold Key.
- The NPL Universal Upgrade is registration-based and simplifies the order process.

## P.6 NPL Release V Compendium Overview

The NPL Release V Compendium is designed as a single source reference for all post NPL Release IV features and enhancement in addition to introduction of all NPL Release V specific features.

Chapter 1 of the Compendium introduces the NPL Release V products and discusses the contents of each. In addition, a general discussion of NPL security is covered.

Chapter 2 of the compendium covers installation of the NPL IDE and RunTime products.

Chapter 3 discusses Numerous configuration issues as they relate to the various operating environments that NPL supports.

Chapter 4 discusses preparing for and running the NPL Universal Upgrade.

Chapter 5 discusses changes to the operation of the NPL RunTime.

Chapter 6 addresses platform specific RunTime updates.

Chapter 7 discusses generic additions and enhancements specific to all NPL platforms.

Chapter 8 discusses additions and updates to the NPL Statements Guide.

# Table of Contents

PREFACE .....	P-1
P.1 NPL Release V .....	P-1
P.2 The NPL Release V Compendium .....	P-2
P.3 Historical Time Line of Post NPL Release IV Updates .....	P-3
NPL Revision 4.10 <sup>[4.10]</sup> .....	P-3
NPL Revision 4.10.16 <sup>[4.10.16]</sup> .....	P-3
NPL Revision 4.10.23 <sup>[4.10.23]</sup> .....	P-3
NPL Revision 4.20 <sup>[4.20]</sup> .....	P-4
NPL Revision 4.21 <sup>[4.21]</sup> .....	P-4
NPL Revision 4.22 <sup>[4.22]</sup> .....	P-4
NPL Revision 4.30 <sup>[4.30]</sup> .....	P-5
NPL Revision 5.0 <sup>[5.00]</sup> .....	P-5
P.4 Supported Operating Environments .....	P-6
P.5 New and Upgrade RunTime Distribution .....	P-7
P.5.1 What is the NPL Universal Upgrade? .....	P-7
P.5.2 Benefits of the NPL Universal Upgrade .....	P-7
P.6 NPL Release V Compendium Overview .....	P-8
TABLE OF CONTENTS .....	TOC-1
INTRODUCTION .....	1-1
1.1 Overview .....	1-1
1.2 The Niakwa IDE Package .....	1-2
1.2.1 The Niakwa Workbench .....	1-3
1.2.2 Niakwa Visual Interface Manager .....	1-3
1.2.3 Niakwa Open Data Manager .....	1-4
Phase 1 .....	1-4
Phase 2 .....	1-4
1.2.4 Niakwa Gateway to ODBC .....	1-5
1.2.5 Online Documentation .....	1-6
1.2.6 Additional Files .....	1-7
Niakwa Utilities .....	1-7
Niakwa Compiler .....	1-7
MS-Windows 32-bit BESDK .....	1-7
1.3 NPL Release V RunTime Package .....	1-8
1.3.1 NPL Release V RunTime Specific Enhancements .....	1-8

1.3.2	Differences between the 16 and 32-bit Windows RunTime Products	1-9
1.3.3	MS-Windows RunTime Files (32-bit)	1-11
1.3.4	MS-Windows RunTime Files (16-bit)	1-11
1.3.5	MS-DOS RunTime Files	1-12
1.3.6	386/DOS-Extender RunTime Files	1-14
1.4	NPL Universal Upgrade Package	1-14
1.4.1	NPL Release V Upgrade Diskette	1-15
1.4.2	NPL RunTime Replacement Gold Key	1-15
1.5	NPL Security	1-16
1.5.1	Differences between New and Old version NPL Security <sup>[4.20]</sup>	1-16
1.5.2	Mixed Network Security Considerations <sup>[4.30/5.00]</sup>	1-17
1.5.3	Mixed RunTime Coexistence <sup>[4.20]</sup>	1-18
1.5.4	Upgrade Considerations	1-18
1.5.5	Windows NT Network Considerations <sup>[4.30/5.00]</sup>	1-19
1.5.6	Windows Terminal Server Considerations <sup>[5.00]</sup>	1-19
1.6	General Operational Enhancements from Past Revisions	1-20
1.6.1	Unique Terminal Identification	1-20
1.6.2	New RunTime Startup Search Procedure <sup>[4.21]</sup>	1-20
INSTALLATION		
2.1	Overview	2-1
2.2	Installation of the NPL RunTime	2-2
2.2.1	Running SETUP	2-2
2.2.2	Choosing Type of Setup	2-3
2.2.3	Selecting the Software Installation Directory	2-5
2.2.4	Selecting RunTime Options	2-6
2.2.5	Selecting Miscellaneous Options	2-7
2.2.6	Additional SETUP.INI Parameters	2-8
	RTP Files Defaults	2-8
	Miscellaneous Options Defaults	2-8
2.2.7	32-Bit Windows RunTime Installation Notes	2-9
2.2.8	Installing on Networks	2-10
	Novell	2-10
	Windows NT	2-10
	NetBIOS Host	2-10
2.2.9	Running Workstation SETUP	2-11
2.2.10	Optional Application Setup	2-11
2.3	Gold Key Security Installation	2-14
2.4	RunTime Installation to MS-DOS platforms	2-16
2.4.1	Running COPYALL.BAT	2-16
	Install the RunTime files	2-16

	Install Gold Key Security	2-17
	New Security Install Parameter <sup>[4.21]</sup>	2-17
	Recall Gold Key Security	2-18
	Files installed by COPYALL.BAT	2-19
2.4.2	Manual Installation of Non-Windows Files	2-20
2.5	Installation of the IDE	2-21
2.5.1	Installing the Niakwa Workbench	2-21
2.5.2	Installing Niakwa Visual Interface Manager	2-24
2.5.3	Installing the Niakwa Open Data Manager	2-25
2.5.4	Installing the Niakwa Gateway to ODBC	2-26
2.5.5	Installing MS-Windows 32-bit Supplementary Files	2-26
2.5.6	Installing the Standard NPL Development Support Files	2-27
2.6	Acrobat Reader	2-27
2.6.1	Installing Acrobat Reader	2-27
2.6.2	Operating Acrobat Reader	2-28
	CONFIGURATION	3-1
3.1	Overview	3-1
3.2	Hardware Requirements	3-2
3.3	NetBIOS Configurations	3-3
3.3.1	Establishing NPL Communication with NetBIOS	3-4
	Current NIAKNETW.DLL Considerations <sup>[4.30 / 5.00]</sup>	3-4
	Past NIAKNETW.DLL Considerations <sup>[4.10.16]</sup>	3-5
	Current NIAKNETB.COM Considerations <sup>[4.30 / 5.00]</sup>	3-6
	Past NIAKNETB.COM Considerations <sup>[4.10]</sup>	3-7
3.3.2	Microsoft Windows NT Networks <sup>[4.20]</sup>	3-8
	Configuring Windows NT Servers and Workstations <sup>[4.20]</sup>	3-8
	Protocol Stack Considerations	3-9
	Windows 95/98 Workstations <sup>[4.20]</sup>	3-10
3.3.3	Peer to Peer Environments <sup>[4.10]</sup>	3-11
	Windows 95/98 Peer to Peer <sup>[4.20]</sup>	3-11
	Windows for Workgroups <sup>[4.10.16]</sup>	3-11
	MS-DOS Peer to Peer Networks <sup>[4.10]</sup>	3-12
3.4	Novell Network Considerations <sup>[4.30 / 5.00]</sup>	3-14
3.5	Mixed Network Considerations	3-15
3.5.1	Running in a Peer Network attached to a Novell Network	3-15
3.5.2	Running in a Network with both Novell and NT Servers	3-15
3.5.3	Running a Stand-Alone Single User RunTime on a Network	3-16
	Novell NetWare	3-16
	Windows NT	3-16
3.6	Wide Area Network Considerations	3-16

3.7	MS-Windows Terminal Server Considerations	3-17
RUNTIME UPGRADES		
4.1	Overview	4-1
4.2	The NPL Universal Upgrade <sup>[4.21]</sup>	4-2
4.2.1	Upgrade Considerations	4-3
4.2.2	General Requirements	4-4
4.2.3	Current Upgrade Considerations <sup>[4.30/5.00]</sup>	4-5
4.2.4	Upgrading Old Security RunTimes <sup>[4.21]</sup>	4-5
	Upgrading in MS-DOS	4-6
4.2.5	Upgrading New Security RunTimes <sup>[4.30/5.00]</sup>	4-7
	Upgrading in Windows 3.1	4-7
	Upgrading in MS-Windows 95/98 or Windows NT	4-7
4.2.6	How to obtain an Upgrade Registration Code <sup>[4.22]</sup>	4-8
	SHOWNPL	4-9
4.2.7	Overview of the NPL Universal Upgrade	4-10
4.2.8	Installing the NPL Universal Upgrade Program	4-11
4.3	Limit Upgrades <sup>[4.21]</sup>	4-14
4.3.1	Limit Upgrade Requirements	4-14
4.3.2	Requesting the Limit Upgrade Code from Niakwa	4-14
4.3.3	The Certificate of License and Authenticity	4-16
4.3.4	Running the NPL Limit Upgrade Program	4-17
RUNTIME OPERATION		
5.1	Overview	5-1
5.2	The RunTime Environment	5-2
5.2.1	Shared vs. Exclusive Access <sup>[4.10]</sup>	5-2
5.2.2	Implied \$BREAK after Disk I/O <sup>[4.10]</sup>	5-2
5.2.3	Two Types of RunTimes <sup>[4.10]</sup>	5-2
5.2.4	Three File Opening Modes <sup>[4.10]</sup>	5-3
5.2.5	User Counts <sup>[4.10]</sup>	5-4
	Description of Count Method	5-6
5.2.6	Mixed RunTime User Counts <sup>[4.21]</sup>	5-8
5.2.7	New RunTime Startup Search Procedure <sup>[4.22]</sup>	5-8
5.3	Compiler Enhancements	5-9
5.3.1	New 32-bit Windows Compilers <sup>[5.00]</sup>	5-9
	Statement Separators Replaced by Spaces in Source.	5-9
5.3.2	General Compiler Enhancements	5-9
	New -ERRFORMAT Option <sup>[4.10.23]</sup>	5-10
	\$OBJECT Output Buffer Extended <sup>[4.20]</sup>	5-11
	B2C Extended to Permit Compiling of Boot Programs <sup>[4.30/5.00]</sup>	5-11

	Enhanced KEEPREMS ON parameter	[4.30/5.00]	5-12
	Update to 2200S Option	[4.30/5.00]	5-13
5.3.3	B2CWIN32		5-14
5.4	RunTime Error Codes		5-15
5.4.1	Error code Updates		5-15
5.4.2	Warning Messages Updates		5-15
PLATFORM SPECIFIC UPDATES			
6.1	Overview		6-1
6.2	General MS-Windows Considerations		6-2
6.2.1	New MS-Windows Setup Options		6-2
	Remote Install Warning Option	[4.22]	6-2
	NIAKNETW Install Option	[4.22]	6-2
6.2.2	RTIWIN.INI Updates		6-3
	Establishing Unique 16-bit and 32-bit Sections	[4.30/5.00]	6-3
	New MS-Windows Debug Parameters	[4.10]	6-4
	New Mixed Network Startup Options	[4.30/5.00]	6-7
	New True Type Font Parameters	[4.30/5.00]	6-7
6.2.3	MS-Windows Debug Mode	[4.10]	6-8
	Options of the Debug Window.		6-9
	Hard Mode		6-10
6.2.4	New /C (Control) Command Line Option	[4.22]	6-10
6.2.5	\$DECLARE Statement Implemented	[4.20]	6-11
6.2.6	New MS-Windows Demo RunTime	[4.30/5.00]	6-11
6.2.7	New MS-Windows 2227 Communications Support	[4.10.23]	6-12
6.2.8	Default Printer Device Name Change	[4.20]	6-12
6.2.9	\$OPEN Considerations for Print Class Devices	[4.10]	6-13
6.2.10	Landscape and Portrait printing under MS-Windows	[4.20]	6-14
6.2.11	New \$DEVICE Clause "XPA=Y"	[4.21]	6-14
6.3	Windows NT Considerations		6-15
6.3.1	Windows NT Workstations on a Novell Server	[4.20]	6-15
6.3.2	Using Windows NT Clients with Windows NT Server	[4.20]	6-16
6.3.3	Using DOS NetBIOS Sessions Under Windows NT	[4.20]	6-16
6.3.4	Diskette Access under Windows NT	[4.20]	6-16
6.3.5	Printer Device Names	[4.20]	6-16
6.3.6	WIN2227 Considerations	[4.20]	6-17
6.4	Windows 95 Considerations		6-17
6.4.1	Diskette Access Under Windows 95	[4.20]	6-17
6.4.2	Windows 95 Logout Considerations	[4.20]	6-17
6.5	General Network Considerations		6-18
6.5.1	Improved Mixed-Network Authentication	[4.30/5.00]	6-18

6.5.2	New Mixed Network Support	[4.21]	6-18
6.5.3	\$DEVICE UNC Notation	[4.20]	6-19
6.6	General MS-DOS Considerations		6-20
6.6.1	Enhanced PC Keyboard Support	[4.20]	6-20
PROGRAMMER'S GUIDE UPDATES			7-1
7.1	Overview		7-1
7.2	Device Support		7-2
7.2.1	Landscape and Portrait printing under MS-Windows	[4.20]	7-2
7.2.2	New \$DEVICE Clause "XPA=Y"	[4.21]	7-2
7.2.3	Enhanced Sector Address Checking	[4.22]	7-3
7.2.4	Automatic Diskimage Extension	[4.30]	7-3
7.2.5	Universal Naming Convention (UNC) Support	[4.20]	7-4
7.2.7	Extended Printer Port Support	[4.10.23]	7-4
7.2.6	Enhanced Printer Control	[4.30]	7-5
7.2.7	Passthrough Printing (XPA=Y)	[4.21]	7-7
7.2.8	Read Only Status (RDO=Y)	[4.20]	7-7
7.2.9	New Device Clause Option (MXE=Y) for 2227 Support	[4.10.23]	7-8
7.3	HELP Subsystem	[4.10]	7-8
7.4	Procedure Handling	[4.10.23]	7-9
7.5	Immediate Mode Subroutine Handling	[4.10.23]	7-10
7.6	Memory and Program Resolution	[4.10.23]	7-10
7.7	Atomic File Locking	[4.10.23]	7-11
7.8	BESDK Support	[4.20]	7-13
7.8.1	Callback Support	[4.20]	7-13
7.8.2	32-Bit BESDK Support	[5.00]	7-14
	Declarations		7-14
	Sharing DLL's		7-15
	C Library used with DLL's		7-15
7.9	File I/O Using Streams	[5.00]	7-16
7.10	New Century Date Module	[4.21]	7-19
7.11	Accented Characters	[4.22]	7-19
7.12	Enhanced Boolean Expressions	[4.30]	7-21
7.13	HALT Key Sequence	[4.30]	7-22
7.14	NPL Interactive Line Editor		7-23
7.14.1	In-Line Comments Extended	[4.21]	7-23
7.14.2	New Line Editor Functions (CUT/COPY/PASTE)	[4.22]	7-24
	Default NPL Keyboard Equivalence Extensions		7-24
	Using the Mouse to Select Text		7-26
	Using the Keyboard to Select Text		7-27
	Delete		7-29



	Copy .....	7-29
	Paste .....	7-30
	Undo .....	7-30
	Block Indent and Deindent .....	7-30
	Paste Control Enhanced <sup>[4.30/5.00]</sup> .....	7-30
	LANGUAGE ENHANCEMENTS .....	8-1
8.1	Overview .....	8-1
8.2	Corrections .....	8-2
8.2.1	\$SOURCE Correction <sup>[4.10]</sup> .....	8-2
8.2.2	Library Function Corrections <sup>[4.10]</sup> .....	8-2
8.3	Syntax Enhancements .....	8-3
8.3.1	\$DEMO .....	8-3
	Enhanced Keywords <sup>[4.10]</sup> .....	8-3
	Improved Debugging of \$DEMO Scripts <sup>[4.10.23]</sup> .....	8-3
	\$DEMO Extended <sup>[4.20]</sup> .....	8-3
	\$DEMO Mouse Support <sup>[4.22]</sup> .....	8-3
8.3.2	\$DEVICE .....	8-4
	\$DEVICE() Extended <sup>[4.20]</sup> .....	8-4
	\$DEVICE Default Number Increased <sup>[5.00]</sup> .....	8-4
	Passthrough Printing “XPA=Y” <sup>[4.21]</sup> .....	8-5
	Read Only Status (RDO=Y) <sup>[4.20]</sup> .....	8-5
8.3.3	INCLUDE Command Restrictions <sup>[4.22]</sup> .....	8-6
8.3.4	KEYIN <sup>[4.30]</sup> .....	8-7
8.3.5	LIST Command <sup>[4.10]</sup> .....	8-7
8.3.6	LIST DC with LDATE <sup>[4.30]</sup> .....	8-7
8.3.7	LIST DT Command <sup>[4.10]</sup> .....	8-8
8.3.8	LIST PUBLIC and LIST DIM Commands <sup>[4.30]</sup> .....	8-9
8.3.9	MAT SORT <sup>[4.10]</sup> .....	8-9
8.3.10	MODULE Command <sup>[4.10]</sup> .....	8-9
8.3.11	\$OSERR <sup>[4.10]</sup> .....	8-10
8.3.13	ON ERROR <sup>[4.30]</sup> .....	8-10
8.3.13	READ DC with LDATE <sup>[4.30]</sup> .....	8-11
8.4	New NPL Statements and Syntax .....	8-12
8.4.1	\$ARGS <sup>[4.21]</sup> .....	8-12
8.4.2	CLOSE # <sup>[5.00]</sup> .....	8-13
8.4.3	\$DECLARE <sup>[4.20]</sup> .....	8-14
	\$DECLARE Warnings and Cautions <sup>[4.20]</sup> .....	8-16
	Extended \$DECLARE Parameter Types <sup>[4.21]</sup> .....	8-18
	Dialog Support <sup>[4.21]</sup> .....	8-18
	\$DECLARE Controls <sup>[4.21]</sup> .....	8-19

	\$DECLARE Functions [4.21]	8-19
	\$DECLARE /POINTER parameters [4.30]	8-20
	32-bit \$DECLARE Support [5.00]	8-23
8.4.4	DATA LOAD BU [4.30]	8-25
8.4.5	DATA SAVE BU [4.30]	8-27
8.4.6	#HDC System Variable [4.30]	8-29
8.4.7	\$IMAGEF and \$IMAGEL [5.00]	8-31
8.4.8	\$LDATE [4.30]	8-32
8.4.9	LIST PROC [4.30]	8-33
8.4.10	OPEN # [5.00]	8-36
	OPEN # file modes	8-37
8.4.11	\$OPTIONS# [5.00]	8-38
8.4.12	READ # [5.00]	8-41
8.4.13	REDIM [4.30]	8-42
8.4.14	\$RUNDIR [4.21]	8-43
8.4.15	SEEK # [5.00]	8-44
8.4.16	SELECT DISK/ # [5.00]	8-45
8.4.17	\$USER [4.22]	8-46
8.4.18	WRITE # [5.00]	8-48
8.5	System Variable Additions and Changes	8-49
8.5.1	\$MACHINE Changes	8-49
	Byte 2 - Hardware Manufacturer Code [4.30]	8-49
	Byte 30- Reserved [4.10]	8-49
	Byte 31 -Reserved [4.10]	8-49
	Byte 32 - RunTime Environment Detected [4.10]	8-50
	Byte 33 - File Locking and User Count [4.10]	8-51
	Byte 34 - Reserved	8-51
	Bytes 35 through 37 - Expiration Date [4.20]	8-51
	Byte 38 through Byte 40 - Reserved	8-51
	Bytes 41 through 44 - Mouse Position [4.21]	8-52
8.5.2	\$OPTIONS Additions and Enhancements	8-52
	Byte 14 - \$BREAK Performance [4.10]	8-52
	Byte 20 - Improved Performance of Remote Terminals [4.30]	8-53
	Byte 33 - Suppress HELP Processor Options [4.20]	8-54
	Byte 38 - Enhanced Required Variable Declaration [4.30]	8-55
	Byte 39 - File Locking and Sharing [4.21]	8-56
	Byte 42 - Restrict RETURN ERROR [4.10]	8-60
	Byte 45 - Enhanced Line Recall Operation [4.30]	8-61
	Byte 46 - Enhanced HELP Processing [4.10]	8-63
	Byte 47 - MS-Windows Options [4.10]	8-63
	Byte 48 - Enhanced Print Device Handling [4.10.23]	8-67

---

Byte 49 - COM Variable Resolution	[4.10.23]	8-69
Byte 50 - CGA Emulation	[4.10.23]	8-70
INPUT SCREEN / PRINT SCREEN Considerations	[4.10.23]	8-70
Limitations and Warnings for \$OPTIONS Byte 50	[4.10.23]	8-71
Byte 51 - Multitasking Timing Control	[4.20]	8-72
Byte 52 - HALT/Function Control	[4.20]	8-73
Byte 53 - Box Graphics Display	[4.20]	8-74
Byte 54 - Mouse Support for Terminal Emulators		8-76
Byte 55 - Mouse Pixel Position Support for Terminal Emulators		8-76
Byte 56 - PRINT SCREEN Ignore	[4.21]	8-77
Byte 57 - CUA Key Mode and Availability of the F10 Key	[4.22]	8-78
Byte 58 - Enable Streaming I/O	[5.00]	8-79
8.5.3 \$BOXTABLE	[4.20]	8-80
Byte 1 - Disable / Enable Character Boxes		8-80
INDEX OF TERMS		Index-1

This page intentionally left blank



# CHAPTER 1

## INTRODUCTION

### 1.1 Overview

The NPL Release V Compendium is intended as an aid in the correct installation and use of Niakwa's NPL Release V RunTime products and Integrated Development Environment (IDE) development tools. In addition, the compendium documents all post NPL Release IV additions and updates, up to and including the NPL Release V products.

Section 1.2 discusses the contents of the Niakwa Release V IDE CD-ROM.

Section 1.3 discusses the contents of the NPL Release V RunTime Package.

Section 1.4 discusses the contents of the NPL Release V Universal Upgrade Package.

Section 1.5 discusses NPL Security.

Section 1.6 discusses general operational enhancements.

## 1.2 The Niakwa IDE Package

This section introduces all development components distributed as part of the Niakwa Integrated Development Environment (IDE) Package. The Niakwa IDE includes four new core components in addition to the standard NPL Compiler, Utility, Terminal Support and BESDK files provided in past releases. All components of the Niakwa IDE are distributed on a single CD-ROM in the following directory structure:

```
\NPLV_IDE
  WB
  VIM
  OPEN_NDM
  ODBC
  W32BESDK

NPL430
  COMPILER
  UTILITY
  TERMINAL
  DOSBESDK
  W16BESDK
  386BESDK

MANUALS
  ACROREAD
  PDF_FILES
  MISC.
```

The following sections provide a brief description of each component included on the Niakwa IDE CD-ROM. Refer to Section 2.5 for specific instructions on installing each of the Niakwa IDE components.

### 1.2.1 The Niakwa Workbench

The Niakwa Workbench is a new full featured 32-bit Editor/Debugger. This initial release of the NPL Workbench includes the editor portion of the product which offers developers a rich GUI based development environment to replace the traditional NPL line editor. The NPL Workbench includes online documentation via the Microsoft Winhelp facility.

**NOTE: The NPL Workbench is a true 32-bit product, and must be installed in either a Windows 95/98 or Windows NT environment.**

### 1.2.2 Niakwa Visual Interface Manager

The Niakwa Visual Interface Manager “Vinny”, (formerly Visual NPL and informally referred to a “Vinny”) allows NPL developers to leverage the rich GUI based user interface development environment of Visual Basic without having to give up the core tried and true NPL routines at the heart of all NPL applications.

The Release V version of Vinny introduces 32-bit support to the product and contains the following file updates:.

VnDev32.bas	32-bit developer modifiable VB code
VnUtil32.bas	32-bit main VB routines and constants
VnLink32.frm	32-bit VB form for communication with 32-bit NPL
VnChar32.frm	32-bit VB form used for creating controls on the fly

**NOTE: 32-bit Vinny is tested and supported on Microsoft Visual Basic 5.0 or greater.**

In addition to the above file changes, the Vinny install program has been updated so that:

- The current product is 32-bit only.
- Installs Instant Vinny as part of the standard setup of the product (no longer a separate install routine.)

Refer to the Niakwa Visual Interface Manager Release Notes for a complete discussion of all updates and enhancements introduced in Release V.

### 1.2.3 Niakwa Open Data Manager

The Niakwa Open Data Manager (formerly NDM and now called “Open NDM”) offers developers an alternative to NPL data management by providing NPL with a single interface to third party industry standard ISAM and SQL databases.

Open NDM is a phased implementation as follows:

#### **Phase 1**

This phase introduces 32-bit Btrieve support for the NPL 32-bit MS-Windows RunTime. In addition, two new API calls have been implemented to allow for improved handling of Year 2000 date conversions.

#### **Phase 2**

NDM Utilities will be integrated into the NPL Workbench in this phase. In addition NDM for ODBC/SQL and NDM TCP/IP Gateway will be released.

NPL Release V introduces Phase 1 of Open NDM and includes the following file updates:

NDMUTIL.BS2	NDM Utility Programs	Revision 1.20.14
NDMUTIL.OBJ	NDM Utility start program	Revision 1.20.14
NDMBTRV.QLB	MS-DOS Btrieve Support Library	Revision 1.20.13
NDMBTRV.DLL	Win16 Btrieve Support Library	Revision 1.20.13
NDMBTR32.DLL	Win32 Btrieve Support Library	Revision 1.20.13
NDMPCK32.DLL	Win32 Pack/Unpack Support Library	Revision 1.20.13

Refer to the Niakwa Open Data Manager Release Notes for a complete discussion of all updates and enhancements introduced in Release V.



#### 1.2.4 Niakwa Gateway to ODBC

The Niakwa Gateway to ODBC allows developers direct access to Microsoft's industry standard ODBC API interface. The Niakwa Gateway to ODBC is compliant with Microsoft's ODBC 3.0 specification and provides support for standard SQL core, level 1 and level 2 API calls.

ODBCDEMO.BS2	Diskimage containing ODBC Gateway demo and library routines
SQL.BS2	ODBC Gateway Routines
BOOT.OBJ	Boot program for demo program
PREBOOT.OBJ	Pre-Boot program for demo program
TEST.BAT	Batch file to execute the demo program

**NOTE:** The 32-bit NDM Gateway to ODBC API routines are linked as standard components of the 32-bit MS-Windows RunTime. No external library is required to be loaded at startup.

The Niakwa Gateway to ODBC product assumes a developer is experienced in the use of ODBC and the SQL language. Refer to the NPL Release V Niakwa Gateway to ODBC Developer Guide for a complete discussion of this new product.

**NOTE:** Developers who are not familiar with ODBC or SQL may want to consider using the NDM for ODBC/SQL interface which will be introduced in Phase 2 of Open NDM. The NDM for ODBC/SQL interface provides developers with the same capabilities of the Gateway to ODBC product, but minimizes the required expertise level of dealing with ODBC and SQL routines directly.

### 1.2.5 Online Documentation

The Niakwa IDE contains the following online documents.

- NPL Release IV Statements Guide
- NPL Release IV Programmer's Guide
- NPL Release IV MS-DOS Supplement
- NPL Release V Compendium
- Visual NPL Developers Guide
- Niakwa Data Manager
- Niakwa Gateway to ODBC

The above documents are stored in PDF format and may be referenced using the Adobe Acrobat viewer which is distributed as part of the Niakwa IDE. Refer to Chapter 2 for details on installing the Adobe Acrobat viewer.

**NOTE:** Some of the above files were unable to be converted to PDF format in time for the final release. Files not converted have been distributed in standard rich text format (RTF) files which are accessible by most word processors.

### 1.2.6 Additional Files

In addition to the new Niakwa IDE components discussed above, the Niakwa IDE CD-ROM incorporates all of the standard NPL development and support files disks which were included as part of all past NPL releases. These include:

- Niakwa Utilities
- Niakwa Terminal Support Files
- MS-DOS BESDK
- 386/DOS-Extender BESDK
- MS-Windows 16-bit Supplementary Files
- Niakwa Compiler

In addition to the above files, the following new support files are provided:

- MS-Windows 32-bit BESDK

The following discusses additions and updates contained on the above file sets where applicable.

#### **Niakwa Utilities**

The Niakwa "Edit Options" Utility program has been updated to include all \$OPTIONS byte extensions and Help descriptions.

#### **Niakwa Compiler**

The Niakwa compiler now contains the following new 32-bit program files:

B2CWIN32.EXE	Windows 32-bit compiler used by the Niakwa Workbench
B2CCON32.EXE	Windows 32-bit compiler for use in batch files

#### **MS-Windows 32-bit BESDK**

This is a new file set introduced to facilitate the development of external subroutine calls from the new 32-bit MS Windows RunTime.

## 1.3 NPL Release V RunTime Package

The NPL Release V RunTime Package contains an Installation Guide and an NPL Release V Gold Key containing a set of Windows Setup diskettes. All NPL Release V products are licensed for the new NPL 32-bit MS-Windows RunTime. In addition, NPL Release V RunTimes continue to include all NPL Revision 4.30 16-bit RunTime products (i.e., MS-DOS, 16-bit MS-Windows and 386/DOS-Extender) to provide continued support for end-user sites which have not yet migrated to 32-bit operating environments.

### 1.3.1 NPL Release V RunTime Specific Enhancements

The following provides an overview of enhancements introduced in the NPL 32-bit MS-Windows RunTime. For a complete discussion on the operation of these new features, refer to the Release V Documentation.

- NetBIOS communication support is embedded and no longer requires NIAKNETW.DLL.
- Support for 2227 asynchronous communications is built-in and no longer requires the WIN2227.DLL.
- \$DECLARE has been updated to provide support for calls to 32-bit DLL's.
- New 32-bit Windows BESDK is introduced.
- RTIW.INI now supports optional sections specific to 16-bit or 32-bit Windows versions.
- New 32-bit Open NDM and Vinny libraries have been added.
- New language statements have been added featuring a Y2K compliant \$LDATE.

### 1.3.2 Differences between the 16 and 32-bit Windows RunTime Products

The following list describes known differences between the NPL 16-bit and 32-bit versions of the Windows RunTime. It is recommended that developers thoroughly test their applications to insure full upward compatibility with the new 32-bit MS-Windows RunTime.

- Output to COM and LPT devices is synchronous, so the 32-bit version it may be slower than the 16-bit version. During large writes to devices, the NPL window will not respond to keyboard/ mouse.
- Access to the Windows spooler is supported, however the options dialogs work slightly differently. If a printer name uses select (?) or SET=Y options, the standard Page Setup dialog is displayed to allow access to the most common printer setup options (form size, paper feed options, portrait/landscape, margins).
- Printers other than the default printer may be selected via the Printer button on that dialog if the select option is specified. Specific printer options can be accessed from the Properties button of that dialog.
- If the select option is not specified but SET=Y is, the detailed printer options dialog is not accessible. This means some options that would previously have been available when only the SET=Y option was used may be unavailable.
- Access to MXE=Y ports is via embedded support (not an external WIN2227.DLL). Any updates or changes to this feature will require an update of NPL.
- Access to raw diskettes is currently only supported when running under Windows NT. Windows 95/98 does not support raw diskettes at this time.
- This version of the 32-bit Windows RunTime does not support UNICODE filenames. Long file names may be used for \$DEVICE specifications, however file names with embedded blanks are not accessible. On some file systems supported by Windows NT, file names are case sensitive.
- The NIAKNETW NetBIOS communications support is embedded. Multiple windows on a single network node counts as 1 user. The NIAKNETW logic does not remain resident after the 32-bit Windows RunTime exits. As a result, response times to user count queries will be longer than the 16-bit version for the first 32-bit version user to start on the network.

- Communications between 16-bit Windows/DOS RunTimes and 32-bit Windows RunTime versions running on the same machine is not supported. If RTIWIN.INI options to allocate specific #PART values are used, NPL does not enforce uniqueness between 16 and 32-bit versions. Use of \$PSTAT to monitor other tasks on the same machine will only be able to see other tasks using the same (16-bit, DOS or WIN32) version. Under NetBIOS, 16-bit and 32-bit versions running on the same node count as 1 user each.
- File locking targeted to Novell servers no longer uses the 'efficient' (timeout) Novell lock, since this adversely impacts other tasks running on the same node (in particular, network access by other tasks is suspended while the lock is pending).
- The HELP processor displays the shell entry as "Command", not "MS-DOS".
- On the B2CWIN32 version only, if source code is extracted from pcode, then statement separators (":") following a new line are replaced with spaces, and character set translation is enabled..
- The RTIWIN32 version contains revised logic to avoid spooling problems encountered in Windows 95. The RAW.DRV driver need not be installed for this correction to take effect.
- The use of \$OPTIONS byte 16 to enable the use of a math coprocessor causes NPL to use C library functions to evaluate transcendentals after converting to an internal double floating point form. In general this results in faster execution of these functions but the results may not be identical to the value returned when using the internal NPL numerics, or when the coprocessor is enabled for other platforms (DOS, 16-bit Windows, 386/DOS Extender, SCO). Specifically, function values where the argument or result exceed the range of an IEEE double precision number (approx  $IE+ / - 308$ ), the results are in general unreliable.

The following sections discuss all files on the NPL Release V Gold Key RunTime package. All files discussed in the following sections are distributed (in no specific order) across the NPL Release V Gold Keys in compressed format. Refer to Chapter 2 for details on installing the NPL Release V RunTime Package. Refer to Chapter 4 for details on applying the NPL Universal Upgrade.

**1.3.3 MS-Windows RunTime Files (32-bit)**

RTIWIN32.EXE	Interpretive 32-bit Windows RunTime
RTPWIN32.EXE	Non-interpretive 32-bit Windows RunTime
MSVCRT.DLL	Microsoft Visual C RunTime Library required by all 32-bit executables
NPLKDL32.DLL	32-bit \$DECLARE dialog function support
NPLKCT32.DLL	32-bit \$DECLARE picture button and other custom controls

**1.3.4 MS-Windows RunTime Files (16-bit)**

NIKNETW.DLL	Required for MS-Windows based NetBEUI networks
NPLKCTL.DLL	Used for \$DECLARE support by the MS-Windows RunTime
NPLKDLG.DLL	Used for \$DECLARE support by the MS-Windows RunTime
SHAREDLL.DLL	Used internally by the MS-Windows RunTime
WININSTX.DLL	Used by the MS-Windows based installation procedure
RAW.DRV	Driver used by the Windows 95 print spooler
LAUNCH.EXE *	Used by the MS-Windows RunTime for MS-DOS commands
NIKINSW.EXE	A MS-Windows based security installation program. Refer to Section 2.3 for more information.
RTISLAVE.EXE	A program used by RTIWIN for the RunTime Debug option
RTIWIN.EXE	The Niakwa Interpretive MS-Windows RunTime Program
RTPWIN.EXE	The Niakwa Non-interpretive MS-Windows RunTime Program
BASFONT.S FON *	The Niakwa screen fonts provided for use with MS-Windows

---

IBMFONTS.FON *	The IBM screen fonts provided for use with MS-Windows
NPLFONT.TTF *	Niakwa True Type font provided for use with MS-Windows
RTINERR.HLP *	Text file containing MS-Windows error messages that are optionally displayed by the Interpretive RunTime for MS-Windows (RTIWIN)
RTINERR.IDX *	Contains the index used when the RTINERR.HLP file is accessed
RTIINI.TXT *	File describing parameters used in the RTIWIN.INI configuration file
*	<b>These files are also used by the 32-bit Windows RunTimes.</b>

### 1.3.5 MS-DOS RunTime Files

COPYALL.BAT	Batch file used to install RunTime files on a Non-Windows MS-DOS based computer
EXPAND.EXE	File used by the COPYALL.BAT batch file
LIMIT.BAT	Allows the current RunTime license to be updated
NIAKINSR.BAT	Used by the security install/recall procedures
NIAKINST.BAT	Installs security on the hard drive
NIAKMOVR.BAT	Used by the security install/recall procedures
NIAKRCLL.BAT	Recalls security from the hard drive
RESET.BAT	Allows the security install count to be updated remotely
DOSINST.BS2 *	Used by the security install/recall procedures
NIAKNETB.COM	A TSR required by NPL to run in a DOS based NetBIOS networks
NIAKAREG.DAT	Used by the RunTime security check
NIAKSER.DAT *	Used by the RunTime security check



---

IBMFONT0.EGA	Screen character set for the IBM Enhanced Graphics Adapter
INSTALL.ERR	Error message file used by the install/recall procedures
NTFSMARK.EXE *	Used by MS-Windows based security install/recall procedure
RTI.EXE	The Interpretive RunTime Program
RTP.EXE	The Non-interpretive RunTime Program
IBMFONT0.HGA	Screen character set for the Hercules Monochrome Graphics Adapter
ERRORMSG.HLP **	Text file containing error messages that are displayed when using the Interpretive RunTime Program (RTI)
RTIERR.HLP	Text file containing MS-DOS error messages that are displayed optionally when using the Interpretive RunTime Program (RTI)
ERRORMSG.IDX **	Index listings used to access the ERRORMSG.HLP file
RTIERR.IDX	Index used to access the RTIERR.HLP file
DOSINST0.OBJ *	Used by the install/recall procedures
DOSLIMI0.OBJ *	Used by the limit/upgrade procedure
DOSRESE0.OBJ *	Used by the reset procedure
DOSINSTX.QLB	Used by the install/recall procedures
LIMITUPG.TXT	Text messages used by the limit/upgrade procedure

\* **These files are used by both Windows and DOS security.**

\*\* **These files are used by all versions of RTI.**

### 1.3.6 386/DOS-Extender RunTime Files

CFIG386.EXE	A utility that allows customization of startup switches that set up the 386/DOS-Extender environment used by the Niakwa 386/DOS-Extender RunTime. Refer to Section 3.4 of the NPL 386/DOS-Extender Addendum in the NPL Release IV MS-DOS Supplement for more information.
RTI386.EXE	The Niakwa 386 Interpretive RunTime Program
RTP386.EXE	The Niakwa 386 Non-interpretive RunTime Program
RTIPERR.HLP	Text file containing the Niakwa 386 RunTime error messages that are displayed when using the Niakwa 386 Interpretive RunTime Program (RTI386)
RTIPERR.IDX	Contains the index used when the RTIPERR.HLP file is accessed

## 1.4 NPL Universal Upgrade Package

The NPL Universal Upgrade Package is a special registration-based RunTime used to upgrade an old NPL Gold Key to the most current NPL Revision. The NPL Universal Upgrade Package consists of an NPL RunTime Package Installation Guide and four diskettes, labeled as follows:

- NPL Upgrade Diskette
- NPL RunTime Replacement Gold Key Diskette - Disk 1 of 3
- NPL RunTime Replacement Package Diskette - Disk 2 of 3
- NPL RunTime Replacement Package Diskette - Disk 3 of 3

The following sections detail the files supplied on the NPL Upgrade Diskette and the NPL RunTime Replacement Diskettes. Refer to Chapter 4 for a complete discussion of the NPL Universal Upgrade procedure.

### 1.4.1 NPL Release V Upgrade Diskette

The NPL Release V Upgrade Diskette is used to initiate and perform the upgrade procedure. The following lists all files contained on the NPL Upgrade Diskette.

#### **In the Root Directory:**

UPGRADE.EXE            Upgrade installation program. This program is used to perform the upgrade procedure.

#### **In the \BASIC2C directory:**

DOSUPGR.BS2            Diskimage containing programs used by the upgrade procedures

NIAKUPG.DAT            Data file used by the upgrade procedures

RTPUPGR.EXE            Program used by the upgrade procedures

DOSUPGR0.OBJ           Startup file used by the upgrade procedures

DOSUPGRX.QLB           External library used by the upgrade procedures

DOSUPGR.TXT            Contains text messages used by registration-based installation

NIAKUPGx.COM           Temporary files used by the upgrade procedure

### 1.4.2 NPL RunTime Replacement Gold Key

The NPL RunTime Replacement Gold Key Diskettes are identical to the standard NPL Release V Gold Key diskettes discussed in Section 1.3 above. The NPL RunTime Replacement Gold Key is non-operable until it is enabled by the NPL Universal Upgrade procedure, using the registration code provided on the "Certificate of License and Authenticity" supplied with all NPL Universal Upgrades.

## 1.5 NPL Security

NPL security has gone through a number of changes since the introduction of NPL Release IV. The most significant of these changes was introduced in NPL Revision 4.20 when Niakwa changed the underlying security procedure it used from a third party security tool to a Niakwa developed and supported security mechanism. This change was made in response to a growing number of 32-bit operating environments that were incompatible with the previous third-party security solution employed by Niakwa. The new security scheme simplifies the installation process and eliminates the many known problems related to the old NPL security mechanism.

Since its implementation, a number of changes have been applied to the new NPL security scheme to adapt it to the growing number of mixed network configurations that confront developers today.

The following sections discuss a variety of general topics related to NPL Security. Refer to Chapter 2 for platform specific considerations for installing NPL security.

### 1.5.1 Differences between New and Old version NPL Security [4.20]

While many changes were implemented in the new NPL security scheme, the most notable non-change was that while the type of security used by NPL had changed, the actual installation procedure had not changed. The following items describe the changes made to the new security scheme which have been present since the introduction of NPL Revision 4.20.

1. The “NIAKSECx.COM” TSR files were eliminated, they are no longer required.

**NOTE:** **Legacy versions of these files may be removed from your system if you upgrade from a Revision 4.10 Gold Key or older (assuming you are not maintaining an older version of NPL on your system).**

2. NPL Security no longer uses the \LOGIN directory to search for security on any drive. Upon installation, the security files will be stored in a directory based upon the default NPL startup search procedure as follows:
  - A. If the NIAKWA\_RUNTIME environment variable is not set, NPL attempts to install security first in the directory where the current NPL RunTime was executed from. Typically, this will be the currently selected drive/directory designation.

- B: If the NIAKWA\_RUNTIME environment variable is set, then security is installed in the drive designation specified, provided that it starts with "<drive-letter>:". If the environment variable does not exist or does not start with this sequence, the default "x:\BASIC2C" is used instead.

Niakwa strongly recommends the specified directory be a simple subdirectory of the root. For example:

Recommended:

```
C:\NPL or F:\APPS\BASIC2C
```

Not Recommended:

```
C:\MISC\KEEP\STUFF\WAY\DOWN\IN_THE\TREE
```

When security is installed to a drive, the specified directory will be created if it does not exist, given that any prerequisite parent directory already exists. For example, if C:\MISC\KEEP\STUFF\WAY\DOWN\IN\_THE does not exist prior to the install, the install procedure would fail.

3. Installation on Novell networks no longer requires the user to be logged in as supervisor to run the NIAKINST install procedure. However the user must have sufficient "Create" and "Write" rights to effectively install the fingerprint.

**NOTE:** Although it is not required, logging on as supervisor to run the NIAKINST install procedure is still acceptable.

### 1.5.2 Mixed Network Security Considerations [4.30/5.00]

Upon startup, NPL queries the environment and performs a series of logical procedures to perform its security check. Since NPL uses different operations to validate security in different operating environments, NPL Release V has added improved mixed network security authentication to better determine which security checks NPL should use, when it encounters a mixed network environment. Refer to Section 6.5 for a discussion of the improved Mixed Network Security.

### 1.5.3 Mixed RunTime Coexistence <sup>[4.20]</sup>

In the past, occasionally there was a need to operate more than one Gold Key on a Novell Network. A side effect of this was that the user counting mechanism was not a combination of the two Gold Keys. This typically resulted in a premature user limit exceeded error on the lower Gold Key user license.

New support has since been introduced allowing NPL to track unique Gold Key user counts when running on a Novell Network. Refer to Section 6.5 for further details.

### 1.5.4 Upgrade Considerations

While the NPL Universal Upgrade is a simple and convenient upgrade process, there are several security considerations which should always be checked prior to performing the NPL Universal Upgrade.

1. When upgrading older NPL Gold Keys (revisions prior to 4.20) to NPL Release V, you must first verify that the Gold Key Security is operational on the system you intend to upgrade and you must be able to successfully install/recall the Gold Key from that system as well.

If a system has been upgraded to Windows 95/98/NT, take steps to insure that the old Gold Key can be put on a system that allows the old Gold Key security to be installed/recalled correctly, thus assuring that the Universal Upgrade will not encounter problems.

The reason for this is that security prior to revision 4.20 would fail on many systems, especially if they had been recently upgraded.

2. When ordering an upgrade, take time to run the SHOWNPL batch file on the system with the Gold Key you intend to upgrade. This guarantees the accuracy of the information you provide to Niakwa, thus eliminating the small (but potential) possibility of receiving an incorrect upgrade registration code.

3. Never attempt to run the Universal Upgrade from a MS-DOS box under Windows when upgrading an old security Gold Key. Older NPL security will fail in this circumstance, thus insuring that the Universal Upgrade will fail. Note that on revision 4.20 and greater Gold Keys, running the universal upgrade in a MS-DOS box is acceptable and recommended.

**NOTE:** By taking the above considerations into account prior to performing a Universal Upgrade, you eliminate the most common problems associated with failed Universal Upgrades, thus reducing your overall time spent performing this relatively short and convenient process.

### 1.5.5 Windows NT Network Considerations [4.30/5.00]

When installing NPL on a Windows NT Network, you must install security from the NT Server, NOT across the network from a workstation. The NPL SETUP procedure will warn you if you attempt to do this.

In addition, execution of a single user Gold Key installed on a Windows NT Server is allowed from the NT server only. Attempting to execute from a workstation a single user Gold Key installed on a Windows NT server will fail. Workstations must install the single user Gold Key locally in order to access the RunTime appropriately.

**NOTE:** The above limitation applies to Windows NT networks only and does not affect single user Gold Keys installed on Novell Networks. This limitation is required to insure proper startup operation of NPL in a mixed network environment.

### 1.5.6 Windows Terminal Server Considerations [5.00]

Since all sessions running in a terminal server environment appear as local sessions to NPL, terminal sever sessions should be started with the /T command line parameter to assure appropriate terminal identification across the network.

**NOTE:** All Gold Keys installed on a Windows Terminal Server system require that a functional network protocol is installed. If no suitable network protocol is detected, the RunTime will exit with the following message:

**“On a Windows Terminal Server system a network protocol must be available”**

## 1.6 General Operational Enhancements from Past Revisions

### 1.6.1 Unique Terminal Identification.

If the NIAKNETB TSR is detected at the startup of the RunTime, \$NETID is set to the Network Interface Card (NIC) adapter address (if one is present), enabling developers to establish unique terminal and partition values. Refer to Section 5.3 of the NPL Novell NetWare Addendum for details.

**NOTE:** In cases where NIAKNETB is not detected, developers can still establish unique terminal and partition values using the BASIC2C\_ID environment variable. Refer to Section 5.3 of the NPL Novell Netware Addendum for details.

Previous revisions of NPL defaulted all non-Novell NetWare systems to a #ID value of 0. Refer to Section 5.3 of the NPL Release IV Novell NetWare Addendum for a complete discussion on establishing unique terminal identification.

### 1.6.2 New RunTime Startup Search Procedure <sup>[4.21]</sup>

If NIAKWA\_RUNTIME is not set, NPL will check the directory containing the RunTime being executed first, before looking in the current directory or other locations. In most circumstances, this eliminates the need for setting the NIAKWA\_RUNTIME variable.





## CHAPTER 2

# INSTALLATION

### 2.1 Overview

This chapter discusses installation of the various Niakwa components.

Section 2.2 discusses installation of the NPL RunTime.

Section 2.3 discusses installation of the Gold Key security.

Section 2.4 discusses installing the NPL RunTime in a DOS environment.

Section 2.5 discusses installation of the various components of the Niakwa IDE.

## 2.2 Installation of the NPL RunTime

NPL Release V RunTimes ship as a set of Windows Setup diskettes. All NPL Release V RunTime products are licensed for the new NPL 32-bit Windows RunTime. In addition, NPL Release V RunTimes continue to include all NPL Revision 4.30 16-bit RunTime products (i.e., MS-DOS, 16-bit MS-Windows and 386/DOS-Extender) to provide continued support for end-user sites which have not yet migrated to 32-bit operating environments.

**NOTE:** In addition, a batch file is included to automate the process of extracting DOS only components of the RunTime in a Non-Windows environment. Refer to Section 2.4 for details on installing NPL into a non Windows environment.

### 2.2.1 Running SETUP

The NPL Release V RunTime contains a complete set of Windows (A:SETUP) diskettes. To install the RunTime to a Windows platform:

1. Insert the NPL Release V Gold Key (Disk 1 of 3) into drive A: or B:.
2. Select "Run" from either Program Manager (Windows 3.x) or the Start menu bar (Windows 95/98/NT).
3. Run the Windows setup routine (A:\SETUP or B:\SETUP).

**NOTE:** The NPL 32-bit Windows RunTimes (RTIWIN32 and RTPWIN32) are operable under Windows 95, Windows 98, and Windows NT only.

**WARNING:** Upon completion of the setup procedure, you may be prompted to restart Windows. It is advisable to close other applications before running SETUP.

The installation routine installs all licensed RunTime products and security to the specified RunTime directory. The following sections provide a detailed discussion of each phase of the NPL setup program.

**NOTE:** The Setup program uses context-sensitive help. Click the Help button at any time for more information on a particular topic.

### 2.2.2 Choosing Type of Setup

The opening Setup screen is displayed as shown in Figure 2-1. Three setup options are available.

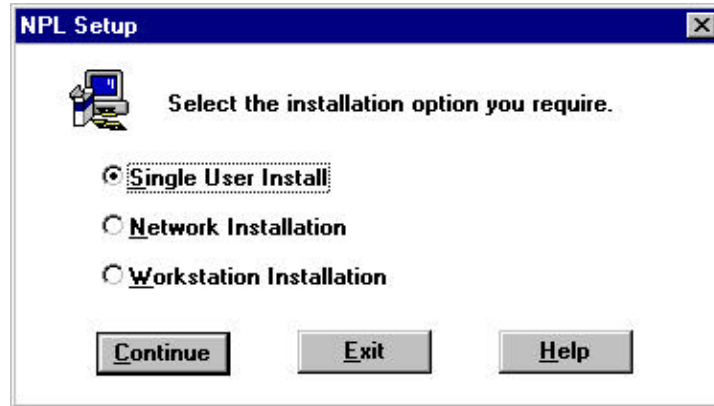


Figure 2-1 NPL Setup Screen

- **Single User Install** This option is intended for standalone workstations, when the NPL RunTime is to be installed and executed locally on a system. This option installs all NPL files and NPL security.
- **Network Installation** This option is intended for installation of NPL on a network file server. This option copies all NPL files and installs the NPL security to the network server. After the installation is complete, all client workstations must run SETUP (from the installation directory on the server) and choose the Workstation Installation option to continue configuration of remaining components.
- **Workstation Installation** This option is only available on a client workstation after a Network Installation has been completed and SETUP is executed from the installation directory. This option is intended to simplify workstation configuration by adding program groups and icons, modifying the client's AUTOEXEC.BAT, install fonts, and registering the network address for the client workstation.

**NOTE:** When performing a Network install to a Windows NT Server using NTFS, Setup must be run on the NT server, with full access privileges to the install directory.

Once the type of installation is selected, the “NPL Custom Installation” window appears (Figure 2-2), allowing the user to modify values for the destination path, RunTime Program files, and miscellaneous options.

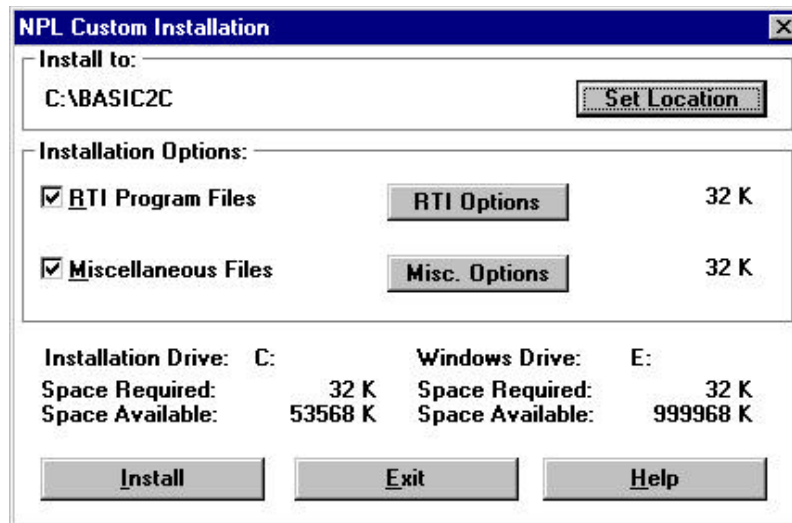


Figure 2-2 NPL Custom Installation Screen

The following section discusses each available option in detail.

### 2.2.3 Selecting the Software Installation Directory

The default destination directory is set to C:\BASIC2C. If a different directory is required, click the Set Location button on the “NPL Custom Installation” window to specify an alternate location.

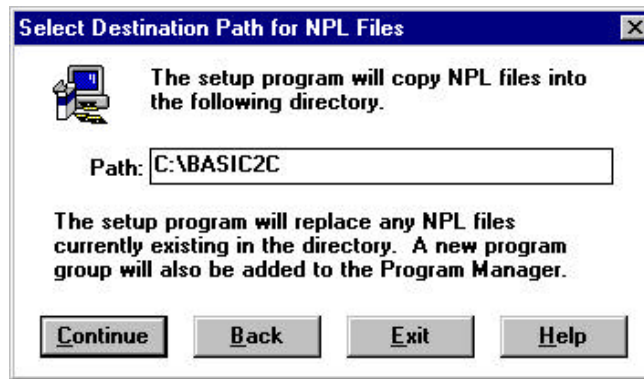


Figure 2-3 Set Location Screen

Click Continue to accept the change, or Back to return to the main setup screen.

**NOTE:** If this is a network installation, sufficient Create/Write privileges are necessary when specifying a network directory path.

If you are installing to an existing NPL system, SETUP will detect if the NIAKWA\_RUNTIME variable is set in AUTOEXEC.BAT and adjust it accordingly based upon the installation path specified. If NIAKWA\_RUNTIME is not set on the system, no modification of AUTOEXEC.BAT is performed.

## 2.2.4 Selecting RunTime Options

The setup program allows selection (or suppression) of RunTime program files that are copied to the system. Clicking on the RTI Options button brings up the window shown in Figure 2-4.

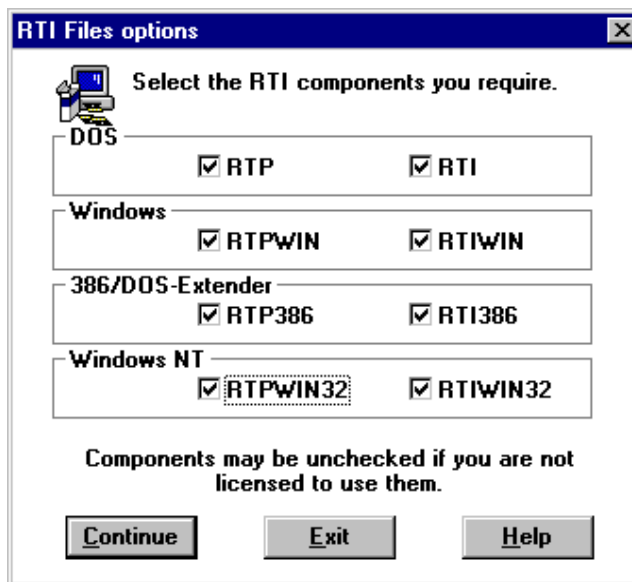


Figure 2-4 RTI files options

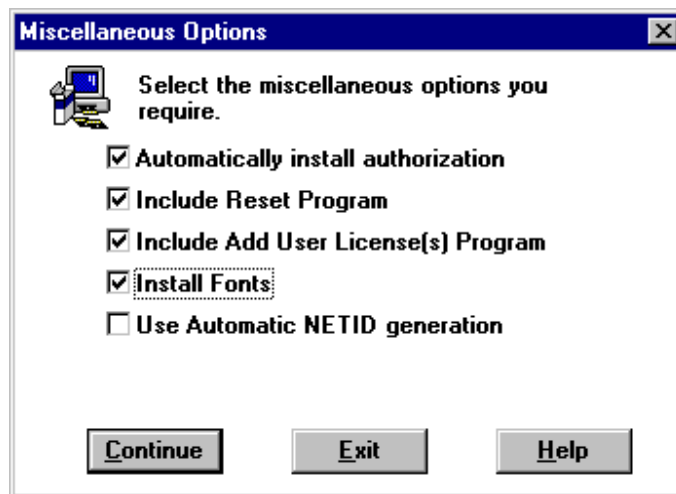
By default, all RunTime programs licensed for use on the current Gold Key are selected and may vary from those displayed in Figure 2-4. Clicking on a check box toggles on/off the corresponding RunTime program to determine whether it is to be copied.

**NOTE:** Unlicensed RunTime programs can be checked, and subsequently copied to the system, however the license prohibits execution of the RunTime. The license can be upgraded at a later date to allow execution. See Section 4.4 regarding License Upgrades.

### 2.2.5 Selecting Miscellaneous Options

Some options are user-selectable prior to installation. Certain program icons can be added or suppressed, MS-Windows fonts automatically installed, and Gold Key fingerprint automatically installed.

Figure 2-5 displays a list of available options. For more information on a particular option, click on the option and then click on the Help button. Clicking on the check box toggles on/off the option. When all desired options are selected, click the Continue button to proceed with installation.



*Figure 2-5 Miscellaneous Options Screen*

**NOTE:** Refer to Section 2.2.6 for a description of each Miscellaneous Option and its default.

### 2.2.6 Additional SETUP.INI Parameters

This section describes items in the Gold Key's SETUP.INI file that may be modified to change the defaults for the setup program installation:

#### **RTP Files Defaults**

InstallDOS=ON	Installs RTP and RTI by default, otherwise set to OFF
InstallWindows=ON	Install RTPWIN and RTIWIN by default, otherwise set to OFF
InstallD3X=ON	Install RTP386 and RTI386 if licensed, otherwise set to OFF
InstallTypeNotAvailable	<b>Do not change this value.</b> This assists the install procedure in determining whether a Network/Workstation is available.

#### **Miscellaneous Options Defaults**

InstallAuthorization=ON	Installs the NPL Security automatically by default. Set to OFF if you do not want security installed by the setup program.
InstallReset=OFF	Off by default. When set to ON, setup installs a program icon for running the RESET program under MS-Windows.
InstallLimit=ON	Setup installs a program icon for running the LIMIT program under MS-Windows by default. Set to OFF if you do not want this option installed.
InstallFont=ON	Automatically installs the NPL screen fonts under MS-Windows by default. Set to OFF if you do not want these fonts installed.



---

InstallNetid=ON	Automatically creates a NETID.TBL file if necessary and adds the \$NETID value of the workstation to the NETID.TBL file. The existence of this file will restrict the use of NPL to listed network stations and enforce assignment of specific terminal and partition numbers to each station.
InstallGroup=NPL	This option sets the title of program group created by setup.

**WARNING:** Windows 95/98 requires this to be a legal long file name, so avoid using punctuation (spaces are okay).

### 2.2.7 32-Bit Windows RunTime Installation Notes

This section describes installation notes specific to the new NPL Release V 32-bit windows executables (RTIWIN32.EXE and RTPWIN32.EXE).

The NPL Windows 32-bit executables require the following files to be installed in the Windows system directory (usually WINNT\SYSTEM32 for Windows NT and WINDOWS\SYSTEM for Windows 95). These files are automatically installed to the appropriate directory by the SETUP program.

MSVCRT.DLL  
NPLKDL32.DLL  
NPLKCT32.DLL

Insure the above files and all other application DLL's are present in the WINDOWS\SYSTEM directory.

The following components of the 16-bit windows version are either obsolete or embedded in the 32-bit Windows version:

RTISLAVE.EXE  
SHAREDLL.DLL  
NIAKNETW.DLL  
WIN2227.DLL

16-bit external libraries designed for 16-bit NPL Windows executables will not function with the new 32-bit Windows executables.

\$MACHINE byte 20 will be HEX(01), indicating a 32-bit environment. Variables larger than 64K can be declared.

### 2.2.8 Installing on Networks

This section describes the differences between installing the NPL RunTime on a Novell Server and a Windows NT dedicated server.

#### **Novell**

When installing the RunTime to a Novell network server, A:\SETUP must be run from a workstation. The workstation must be logged in as "SUPERVISOR" or "ADMINISTRATOR" depending on the revision of Novell installed on the server. See section 2.2.9 for a description of running Workstation Setup.

#### **Windows NT**

When installing the RunTime to a Windows NT network server, A:\SETUP must be run from the server. The workstation must be logged on with administrator rights. See section 2.2.9 for a description on running Workstation Setup.

#### **NetBIOS Host**

When installing the RunTime to a NetBIOS Host machine, A:\SETUP must be run from the Host machine. See section 2.2.9 for a description on running Workstation Setup.

### 2.2.9 Running Workstation SETUP

The "Workstation Installation" option is available only after a Network Installation has been completed and when SETUP is executed from the installation directory. This option will add groups and icons, modify the client's AUTOEXEC.BAT, install fonts, and register the network address for the client workstation.

### 2.2.10 Optional Application Setup

In addition to the standard NPL install options, the NPL setup program also provides an option to allow SETUP.EXE to automatically hook into your application's setup program.

To implement this setup option, the developer must supply a fourth diskette containing the following vendor-written files:

<u>File name</u>	<u>Uncompressed Size</u>	<u>Compressed File Name</u>
APPSETUP.OBJ	256	APPSETUP.OB_
APPSETUP.BS2	1024000 (END=3999)	APPSETUP.BS_

**NOTE:** The files must be the exact size as indicated above before being compressed using the MS-compress program (refer to the MS-Windows SDK documentation for information regarding the use of this program). The files may contain anything, but both the files must be on the diskette.

The intention here is to allow this fixed set of program names to contain an NPL boot program and diskimage which will be sufficient to complete the application installation (i.e., copying from multiple diskettes.)

Because of the minimum size requirement, typically the boot program will contain only the most elementary link to an appropriate program on the diskimage. For example:

```
0010 ;APPSETUP boot
$DEVICE(#0)="APPSETUP.BS2"
LOAD T"APPSTART"
```

To enable the NPL setup program to detect the inclusion of any vendor supplied files, the developer must modify the SETUP.INI file on the Gold Key diskette. The following example shows the default values of the relevant entries:

```
InstallAppSetup=OFF  
AppSetupCommand=rtpwin appsetup  
InstallAppRun=OFF  
AppRunCommand=rtpwin boot  
AppRunDescription=NPL boot
```

To enable the above options, make the following changes:

```
InstallAppSetup=ON  
AppSetupCommand=xxxxxxxxxxxxxxxxxxxxxx  
InstallAppRun=ON  
AppRunCommand=yyyyyyyyyyyyyyyyyyyyyy  
AppRunDescription=zzzzzzzzzzzzzzzzzz
```

Here xxxxxxxxxxxx is the specific command required to start the application setup phase. This may be an NPL application which assumes:

- The availability of RTPWIN
- The two APPSETUP files have been copied to the current drive and directory.
- The NPL installation directory is the current drive and directory.

The string yyyyyyyyyyy is the specific command required to run the application after setup is complete. This may be an NPL application which can assume:

- The availability of RTPWIN
- The two APPSETUP files have been copied to the current drive and directory.
- The NPL installation directory is the current drive and directory.

In addition, files installed by the application setup procedure, if specified, should be available. The string *zzzzzzzz* is the name given to the program manager icon in the installed group.

**Known Problem:**        **If NIAKWA\_RUNTIME is already set when setup is run, and NPL is installed to a different directory, the newly installed RTI may not be functional when the application setup command is run. In this case, the application setup will only work if the previously installed license can run the new RTI.**

## 2.3 Gold Key Security Installation

By default, the NPL MS-Windows Setup program automatically installs the Gold Key Security authorization using a new Windows-based installation routine. Figure 2-6 shows the Authorization installation window.

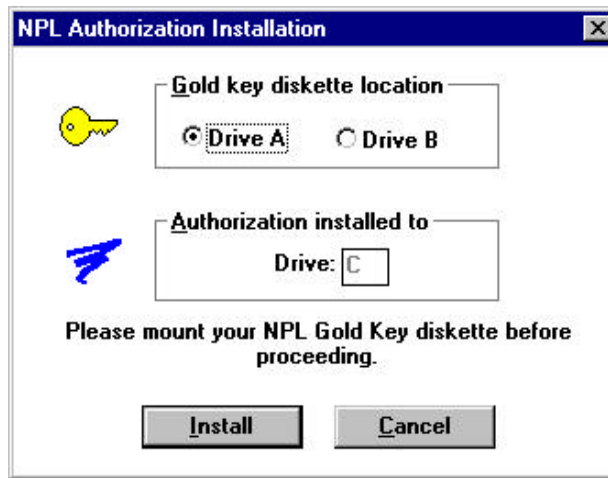


Figure 2-6 NPL Authorization Installation Screen

This window is shown only if necessary to prompt for the Gold Key diskette. By default, the program attempts to read the floppy in the source drive (where installation files were copied from) and installs to the drive specified in the destination path selected earlier in the setup procedure.

If the window is displayed, select the appropriate floppy drive letter and destination drive letter. Be certain the Gold Key diskette is inserted into the drive, and click the Install button to proceed.

If the “Automatic Gold Key Installation” option is disabled (in the Miscellaneous Options window), the authorization may optionally be installed manually, in one of three ways:

- Installation using the provided “Install or Recall Authorization” icons
- MS-Windows command-line installation using NIAKINSW.EXE

- MS-DOS command-line installation using NIAKINST.BAT/NIAKRCLL.BAT (refer to Section 2.4 for details.)

By default, the automated installation procedure places an icon in the selected program group labeled “Install or Recall Authorization”.

To install or recall authorization for the system, simply insert the disk labeled “NPL Gold Key Disk 1 of X” and double-click on the “Install or Recall Authorization” icon. Select the appropriate install or recall operation from the displayed dialog box. Check that the indicated floppy drive is correct, mount the disk labeled “NPL Gold Key Disk 1 of X” and click the OK button.

If no icons were installed or are available (i.e., on a network workstation), the authorization can be recalled using NIAKINSW.EXE directly. The syntax for this program is as follows:

```
NIAKINSW.EXE I | U <drive>:
```

Where I = Install, U = Uninstall (recall), and <drive> is the diskette drive letter where the Gold Key is inserted. Using Program Manager or File Manager, select the Run... option from the File menu. Windows 95/98 or Windows NT 4.x users use the Run... option from the Start menu.

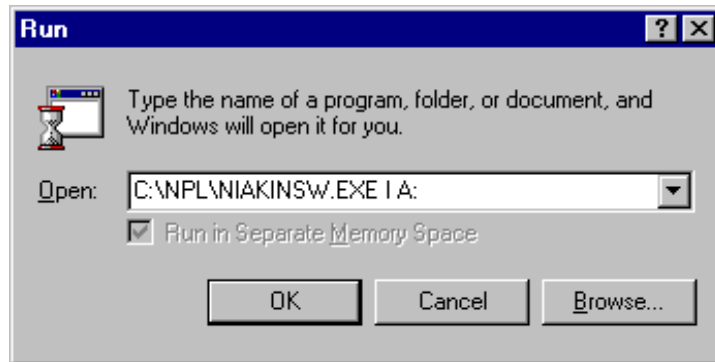


Figure 2-7 Run Command Line Window

In Figure 2-7, the command line installs the authorization from the Gold Key in floppy drive A: to the hard drive (assuming NPL files were installed to C:\NPL.)

## 2.4 RunTime Installation to MS-DOS platforms

The NPL Release V RunTime is a complete set of Windows (A:SETUP) format diskettes. In order to install the RunTime in a Non-Windows environment, a new batch program, "COPYALL.BAT" has been included with all NPL RunTimes.

### 2.4.1 Running COPYALL.BAT

To install the NPL RunTime on a Non-Windows systems, perform the following steps.

#### Install the RunTime files

1. Create the NPL Directory and select it as the current drive/directory.

```
md C:\NPL
cd C:\NPL
```

2. Insert disk 1 in the drive, and copy the file "COPYALL.BAT" from this diskette to the current directory.

```
COPY A:COPYALL.BAT
```

**NOTE:** Because of diskette swapping, do not attempt to run COPYALL.BAT while selected directly to the floppy drive.

3. Run the batch file:

```
COPYALL A:
```

Optionally you can specify the target directory for the expanded files:

```
COPYALL A: C:\NPL
```

**NOTE:** If the above parameters are not specified, the program defaults to drive A: and either the NIAKWA\_RUNTIME value (if specified), or the current directory.

This installation method does not copy files related to the Windows version of the RunTime. If you require these files, you must use the Windows A:\SETUP procedure to install them.



**Install Gold Key Security**

COPYALL.BAT cannot automatically install the Gold Key Security. It must be installed manually.

To install Gold Key Security follow these steps:

1. Insert the original Gold Key Diskette (Disk 1 of 3) into drive A: or Drive B:
2. You are probably already selected to the RunTime directory on the hard drive. If not, select it:

```
CD \NPL
```

3. Enter:

```
NIAKINST A: C:
```

Upon completion of the NIAKINST program the following message will be displayed.

```
RunTime Program Successfully installed
```

**New Security Install Parameter** <sup>[4.21]</sup>

To simplify running fingerprint installation from MS-Windows, the NIAKINST.BAT file now accepts an optional third parameter used to determine whether security is being installed (I) or recalled (D). Specification of this parameter will suppress the Gold Key prompt when installing or recalling security. For example:

```
C:\NPL421>NIAKINST A: C: I
```

The above command line would install NPL security without user intervention (assuming the Gold Key is initially in drive A:.)

**NOTE:** By default, the NIAKINST.BAT file attempts to determine the value of the NIAKWA\_RUNTIME environment variable and uses this directory to install the NPL Security if it is set.

**Recall Gold Key Security**

To manually recall Gold Key security from the hard drive, place the original Gold Key diskette in the diskette drive and follow these steps.

1. If the host system is a Novell NetWare network installation, make sure the system being used for this recall is a workstation logged in as SUPERVISOR (Novell NetWare 3.X), ADMIN (Novell NetWare 4.X) or Administrator (MS-Windows NT) or equivalent.
2. If the NIAKWA RunTime files have been copied to any directory other than \BASIC2C or the security files are placed on a different drive, make sure either the directory is set in the PATH variable, or optionally the NIAKWA\_RUNTIME environment variable is set.
3. Select the drive/directory where the NPL files have been previously copied. For example:

```
C:  
CD \BASIC2C
```

4. Enter:  
**NIAKRCLL C: A:**

NIAKRCLL generates the following message upon successful recall of the RunTime Program:

```
RunTime Program successfully recalled to original  
diskette.
```

**Files installed by COPYALL.BAT**

The following files are expanded and copied into the RunTime directory by the COPYALL.BAT program.

EXPAND.EXE	LIMIT.BAT	UPGRTEST.OBJ
RTP386.EXE	LIMITUPG.TXT	DOSINSTX.QLB
NTFSMARK.PIF	NIAKINSR.BAT	NIAKINSW.EXE
NTFSMARK.EXE	NIAKRCLR.BAT	AUTOID.EXE
NIAKAREG.DAT	REGISTER.TXT	DOSINST.BS2
NIAKSER.DAT	SHOWNPL.BAT	RTIPERR.IDX
CUSTOM.SUS	NIAKINST.BAT	RTIPERR.HLP
RTI386.EXE	NIAKMOVR.BAT	CFG386.EXE
RTI.EXE	NIAKRCLL.BAT	RTIERR.IDX
RTP.EXE	RESET.BAT	RTIERR.HLP
DOSINST0.OBJ	UPGRTEST.TXT	NIAKNETB.COM
DOSLIMI0.OBJ	DOSINST.ERR	ERRORMSG.IDX
DOSRESE0.OBJ	WININSTX.DLL	ERRORMSG.HLP
	NPLNETID.DLL	

### 2.4.2 Manual Installation of Non-Windows Files

The COPYALL.BAT program uses the MS-DOS EXPAND.EXE utility to extract MS-DOS RunTime files from the NPL RunTime diskettes. If it is necessary to extract files other than those required by the MS-DOS and 386/DOS-Extender RunTime programs, the EXPAND utility can be run manually to extract the files required.

The EXPAND utility is shipped on disk 1 of the Gold Key set. If it is ever necessary to extract individual files from the setup diskettes, you can do this using the DOS command line:

```
EXPAND uncompressed-file-name target-directory
```

If the uncompressed file name is not known, use the following command line:

```
EXPAND -r compressed-file-name target-directory
```

For example:

```
EXPAND A:\rtiwin32.exe C:\BASIC2C
```

```
EXPAND -r A:\rtiwin32.ex_ C:\BASIC2C
```

The -r option also allows wildcards to be specified. For example:

```
EXPAND -r A:*.ex_ C:\BASIC2C
```

## 2.5 Installation of the IDE

As noted above, NPL Release V includes the introduction of the Niakwa Integrated Development Environment, which is a functional replacement of the past NPL Release IV Development Package. The Niakwa IDE is the foundation upon which NPL applications will be built into the future. This section discusses the installation of each component of the new Niakwa IDE.

### 2.5.1 Installing the Niakwa Workbench

The Niakwa Workbench is a full featured 32-bit editor/debugger and is the core of the new Niakwa IDE. The Workbench offers developers a more versatile and convenient approach to development than using the traditional NPL line editor. Since the Niakwa Workbench is a 32-bit product, it must be installed in either a Windows 95/98 or Windows NT operating environment.

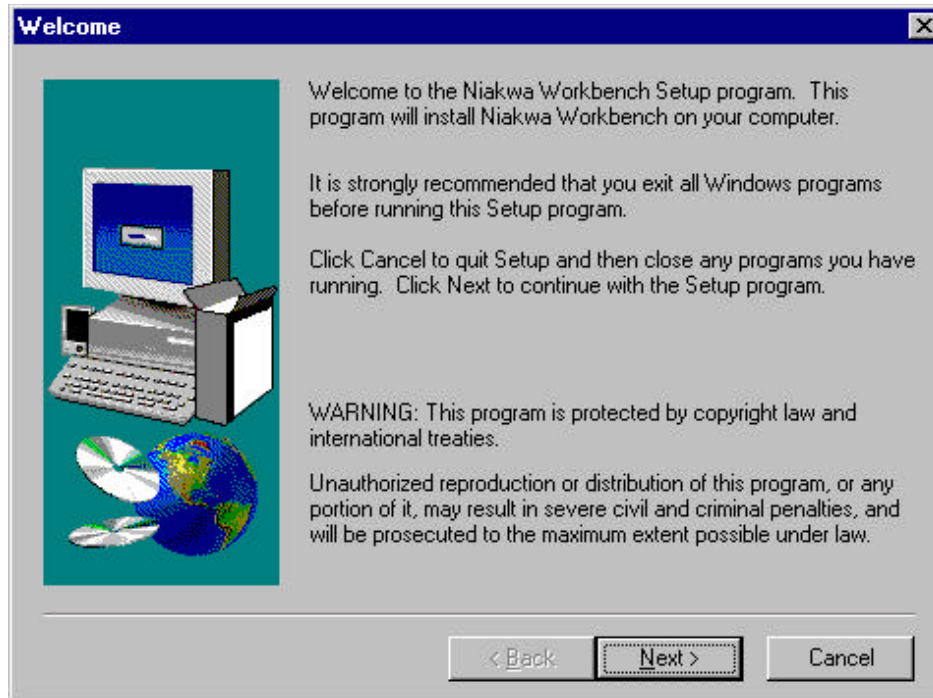
To install the Niakwa Workbench, insert the Niakwa IDE CD-ROM in the CD-ROM drive and perform the following:

1. Select Start/Run
2. Enter the following command line in the Run Dialog box:

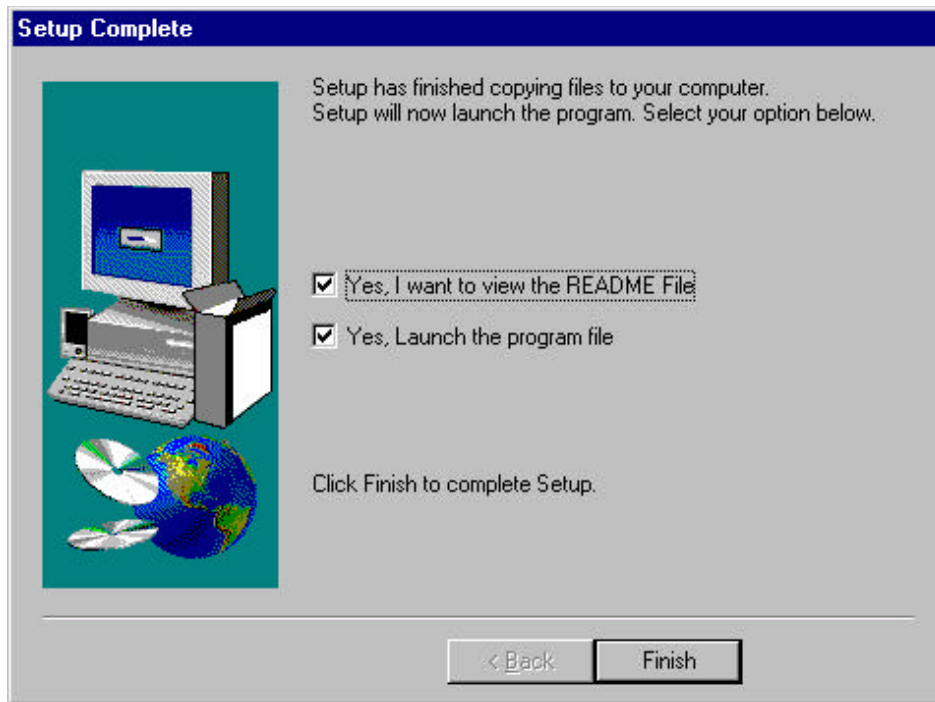
**G:\NPLV\_IDE\WB\SETUP**

The above assumes G: is the CD-ROM spec. Adjust accordingly for your system.

The Setup program will guide you through the rest of the installation procedure.



*Figure 2-8 Workbench Welcome Screen*



*Figure 2-9 Workbench Final Setup Screen*

**NOTE:** Review the README file upon completing setup for an overview of the Niakwa Workbench.

The Niakwa Workbench documentation is provided as an online HELP file within the product. To access the documentation within the Niakwa Workbench, select the “HELP” menu from the task bar and then select “Contents”.

The Niakwa Workbench is a work in progress. The initial release of the Niakwa Workbench includes the full screen editor portion of the product, complete project management tools, syntax checking and highlighting.

## 2.5.2 Installing Niakwa Visual Interface Manager

The Niakwa Visual Interface Manager (“Vinny”, formerly Visual NPL) is the graphical user interface builder of the Niakwa IDE. The Release V version of Vinny is a 32-bit product requiring both NPL Release V and Visual Basic 5.0. In addition, the Instant Vinny development tool has also updated to 32-bit on this release and is now bundled as part of Vinny.

To install Vinny, insert the Niakwa IDE CD-ROM in the CD-ROM drive and perform the following:

1. Select Start/Run.
2. Enter the following command line in the Run Dialog box:

**G:\NPLV\_IDE\VIM\SETUP**

The above assumes G: is the CD-ROM spec. Adjust accordingly for your system.

The Setup program will guide you through the rest of the installation procedure.

The following are additional product notes to consider:

- Refer to the Niakwa Visual Interface Manager Release Notes for a complete discussion of all updates and enhancements introduced in Release V.
- Refer to the existing Visual NPL Developer’s Guide for complete information on the use of the Visual Interface Manager and Instant Vinny.
- To run the 32-bit Visual Interface Manager and Instant Vinny applications, you need to be running all 32-bit components (RTIW32 or RTPWIN32, Vinny 5.0, VB5, 32-bit OCX’s, and 32-bit DLL’s).
- Refer to the Niakwa Visual Interface Manager Release Notes for a complete discussion on migrating your 16-bit projects to 32-bit Vinny 5.0.



### 2.5.3 Installing the Niakwa Open Data Manager

The Niakwa Open Data Manager (“Open NDM”, formerly NDM) is the data management engine of the Niakwa IDE. This new release of Open NDM represents an evolving product. In its final form Open NDM will provide developers with a fully graphical data management facility, integrated within the Niakwa Workbench.

The Open NDM files are located on the Niakwa IDE CD-ROM in the following directory:

```
\NPLV_IDE\OPEN_NDM
```

This directory contains a complete set of Open NDM files including new NDM Utilities and API libraries for Btrieve.

The following are additional product notes to consider:

- Refer to the Niakwa Data Manager Programmer’s Guide for details on the correct installation and use of Open NDM.
- Refer to the Niakwa Open NDM Release Notes for a discussion of what has been updated in this release.
- This release of Open NDM introduces 32-bit Btrieve support to the product. Refer to the Niakwa Open NDM Release Notes for details on using the new 32-bit API with NPL Release V.
- Pervasive SQL has been reviewed and tested. It is upward compatible with Btrieve 6.15. Refer to the Niakwa Open NDM Release Notes for details.
- Open NDM is a work in progress. The next release of this product will introduce ODBC support, TCP/IP support for Unix hosts, tighter integration with the Niakwa Workbench and the new graphical Visual NDM (Open NDM Utilities).

## 2.5.4 Installing the Niakwa Gateway to ODBC

The Niakwa Gateway to ODBC API library offers developers with a SQL background a direct approach to implementing ODBC support into their applications. The ODBC API library contains a series of SQL Core, Level 1 and Level 2 function calls based upon the ODBC 3.0 specification.

The Niakwa Gateway to ODBC files are located on the Niakwa IDE CD-ROM in the following directory:

```
\NPLV_IDE\ODBC
```

This directory contains a complete set of Niakwa Gateway to ODBC files including both 16 and 32-bit API libraries.

The following is an additional product note to consider:

- Refer to the Niakwa Gateway to ODBC Programmer's Guide for details on the correct installation and use of the Gateway to ODBC and its demo programs.

## 2.5.5 Installing MS-Windows 32-bit Supplementary Files

A new BESDK has been introduced in NPL Release V to add support for 32-bit Windows external library development.

To install the new Win32 BESDK, insert the Niakwa IDE CD-ROM in the CD-ROM drive and run the following batch file:

Enter the following command line from a DOS prompt:

```
C:\NPLV>G:\NPLV_IDE\W32BESDK\INSTALLW C:\NPLV
```

The above assumes G: is the CD-ROM spec and C:\NPLV is the directory you want the Win32 BESDK installed. Adjust accordingly for your system. The Win32 BESDK files are installed into appropriate subdirectories in the target drive/directory location.

Refer to section 7.8.2 for additional details on using the new Win32 BESDK.

### 2.5.6 Installing the Standard NPL Development Support Files

The NPL Compiler, Utility, Terminal Support and BESDK files are standard NPL development support files shipped with all new versions of NPL. These files have been updated with this release and are compatible with both NPL Revision Release V and Revision 4.30 products.

These files are contained on the Niakwa IDE CD-ROM in the following directories:

```
\NPL430\COMPILER  
\NPL430\UTILITY  
\NPL430\TERMINAL  
\NPL430\DOSBESDK  
\NPL430\W16BESDK  
\NPL430\386BESDK
```

Installation of the above files is unchanged from previous releases. Refer to your NPL Release IV platform specific supplement for detail on installing these standard development files.

## 2.6 Acrobat Reader

The Niakwa IDE contains several online documents which are stored in PDF format and may be referenced using the Adobe Acrobat Reader which is distributed as part of the Niakwa IDE.

### 2.6.1 Installing Acrobat Reader

To install the Adobe Acrobat Viewer included in the IDE:

1. Select to the CD-ROM
2. Select the \MANUALS\ACROREAD directory and run either:  
  
AR32E301.EXE for Windows95/98/NT systems  
or  
AR16E301.EXE for Windows 3.1/WfW systems.

The Adobe setup routine will guide you through the installation process.

## 2.6.2 Operating Acrobat Reader

Start the Adobe Acrobat Reader using the Icon created during the installation process. Select the document you wish to view from the IDE CD-ROM in the \MANUALS\PDF\_FILES directory. You can page through the document or perform a text search for specific information. The Adobe Acrobat Reader has the ability to print the entire document, or just a selected portion.

**NOTE:**           **Additional information on using the Adobe Acrobat Reader is available on the Adobe web site, [www.adobe.com](http://www.adobe.com). Additional Adobe Readers are also available there.**



## CHAPTER 3

# CONFIGURATION

### 3.1 Overview

Since the introduction of NPL Release IV, many changes have been made to NPL, allowing it to adapt to a number of new operating environments. This chapter discusses specific configuration issues related to supported NPL operating environments that Niakwa has either previously or currently tested.

Section 3.2 discusses hardware requirements and considerations.

Section 3.3 discusses specific NetBIOS operating system configuration issues.

Section 3.4 discusses Novell network considerations.

Section 3.5 discusses Mixed network considerations.

Section 3.6 discusses issues related operation in a WAN (Wide Area networks).

Section 3.7 discusses Microsoft Windows Terminal Server Considerations.

## 3.2 Hardware Requirements

The following table illustrates the minimum system requirements for each operating environment supported by NPL.

System (version) Requirements	MS-Windows 32-bit RunTime	MS-Windows 16-bit RunTime	386/DOS-Extender RunTime	MS-DOS RunTime
MS-DOS	Not Available	Not Available	3.1 or greater	3.1 or greater
MS-Windows	Not Available	3.1 or greater	3.1 or greater	3.1 or greater
MS-Windows for Workgroups	Not Available	3.11 or greater	3.1 or greater*	3.1 or greater*
MS-Windows 95	4.00 or greater	4.00 or greater	4.00 or greater*	4.00 or greater*
MS-Windows 98	4.10 or greater	Not Supported	Not Supported	Not Supported
MS-Windows NT	3.51 or greater	3.51 or greater	3.51 or greater*	3.51 or greater*
Novell NetWare	2.10 or greater	2.10 or greater	2.10 or greater	2.10 or greater
Personal NetWare	Not Available	1.0 or greater	1.0 or greater	1.0 or greater
MS-Windows Terminal Server	32-bit 4.00 or greater	Not Supported	Not Supported	Not Supported
LANtastic	Not Available	5.00 or greater	4.10 or greater	4.10 or greater
Processor	As required by MS-Windows	As required by MS-Windows	80386 or better	8088 or better
Total System Memory	As required by MS-Windows	As required by MS-Windows	4 MB	640KB

\* Supported as standalone or NetWare workstation. NetBIOS operation is supported only with the MS-Windows RunTime.

**NOTE:** The guidelines provided here represent the minimum requirements to successfully execute the RunTime component. Applications written in NPL may have additional requirements beyond those provided.

### 3.3 NetBIOS Configurations

NPL supports a variety of NetBIOS operating environments. While the configuration of each NPL supported operating environment may vary, the operation of NPL within all NetBIOS environments is very straightforward and consistent. The following outlines the general methodology used to install NPL in a NetBIOS configuration.

1. The network is configured and operational using the NetBIOS protocol.

**NOTE:** While many environments support the configuration of multiple types of network protocols, the NetBIOS protocol is required by NPL to operate correctly.

2. NPL is installed on either a peer workstation or workgroup server.
3. Appropriate drive mappings are created on all workstation required to run NPL
4. The NPL NetBIOS TSR program is configured to run on all workstations requiring access to NPL. MS-DOS based NPL RunTimes require the NIAKNETB.COM program to be running, while MS-Windows based RunTimes utilize the NIAKNETW.DLL program.

**NOTE:** As of NPL Release 4.30 and greater, the MS-Windows RunTime (16 and 32-bit) no longer requires the "NiaknetwEnabled=1" parameter to be included in the RTIW.INI configuration file. MS-Windows RunTimes now default to launching the NIAKNETW.DLL when it is required.

5. Batch files or icons are setup to access the NPL application on all workstations required to run the application. In a Windows environment, this is accomplished by running SETUP from the system NPL was installed on and selecting the Workstation install option.

The following sections discuss NPL NetBIOS considerations and configuration issues for a variety of supported NPL environments. Many of the environments defined below are untested in NPL Release V. NPL Release V testing focuses on Windows95/98 and Windows NT NetBIOS configurations.

### 3.3.1 Establishing NPL Communication with NetBIOS

When NetBIOS support for NPL was initially introduced, NIAKNETB.COM was intended for use by MS-DOS based NPL RunTime programs to establish communications with MS-DOS based NetBIOS networks such as Artisoft's LANtastic or Novell's Personal NetWare.

With the Release of Microsoft's Windows for Workgroups 3.11, an alternative for establishing NetBIOS communications was required since NIAKNETB did not work and was not supported in the WfW environment. This led to the development of the Windows based NIAKNETW.DLL

The NIAKNETW.DLL and NIAKNETB.COM programs are the primary components used by NPL to establish user counts and authenticate with a NetBIOS network. NIAKNETB is used by NPL MS-DOS RunTime programs (i.e., RTx.EXE and RTx386.EXE) to authenticate in an MS-DOS environment, while NIAKNETW is used by the 16-bit MS-Windows RunTime program (RTxWIN.EXE). The 32-bit windows support for NetBIOS is embedded, and does not require a separate program.

Niakwa has tested subsequent MS-Windows releases running NIAKNETB in a DOS box and has found that Windows NT workstations provide adequate NetBIOS support, allowing NIAKNETB.COM to load. Thus, enabling MS-DOS based NPL RunTime programs to operate in a MS-Windows NT peer or server network. However, Niakwa highly recommends using the NPL MS-Windows RunTime programs whenever operating in a Windows environment.

The process of how to properly invoke either the NIAKNETB.COM or NIAKNETW.DLL programs are discussed below.

#### **Current NIAKNETW.DLL Considerations** [4.30 / 5.00]

The following changes have been implemented in NPL Release V which may have an impact on applications running in NetBIOS environments.

1. The Windows RunTime no longer attempts to locate or use a NIAKNETB.COM TSR that may have been installed before MS-Windows was loaded. Developers still using this configuration are advised to switch to the new NIAKNETW.DLL instead.

**WARNING:** NIAKNETW has been updated to version 1.15 on this release. Developers should make sure all older versions of NIAKNETW.DLL are removed from both the server and workstations to avoid possible conflicts with older revisions.



2. If no value is specified in RTIWIN.INI for "NiaknetwEnabled=", the default value used is 1. This is a change from previous releases where the default was 0. This setting may be explicitly disabled by setting "NiaknetwEnabled=0".

**NOTE:** All single user RunTimes ignore this option with one exception. If a MS-Windows Terminal Server is detected, NIAKNETW.DLL is required by all RunTimes.

**NOTE:** This change is made in connection with the improved mixed network authentication logic and allows NPL to authenticate with a NetBIOS environment, in the event that NPL first detects a Novell network and fails Novell authentication.

3. NIAKNETW.DLL response time has been reduced. Maximum response time when checking for NIAKNETW user counts has been reduced to 5 seconds from 13.

In general, NPL Release V does its best to automate the process of invoking NIAKNETW.DLL without having to perform the special configuration procedure required by past revisions of NPL.

#### **Past NIAKNETW.DLL Considerations** <sup>[4.10.16]</sup>

This section present historical commentary on NIAKNETW.DLL in consideration of those sites which may not be using a current revision of NPL.

For previous NPL revisions, the NPL RunTime configuration file, (RTIWIN.INI) must contain the following setting to enable NPL to launch the NIAKNETW.DLL file at startup:

**NiaknetwEnabled=1**

**NOTE:** The above setting must be added to the [GENERAL] section of either the local RTIWIN.INI file or the configured NetworkIniFile. This setting is case sensitive, so it must be exact.

Once the above setting is in place, the NPL MS-Windows RunTime dynamically loads NIAKNETW.DLL at start up (if it is not already present). Therefore, NIAKNETW.DLL must be located using the standard MS-Windows DLL search procedure. For the simplest operation, ensure that NIAKNETW.DLL is in the same directory as the NPL Windows executables.

When NIAKNETW.DLL is launched, a small status window appears at the top of the display with the caption "NiakNetW Monitor - Establishing NETBIOS interface" when the Windows RunTimes are checking for NetBIOS authorization (whether using NIAKNETW.DLL or NIAKNETB.COM). This feature was added because some NetBIOS operations effectively stop MS-Windows for a number of seconds. The status windows is displayed to indicate that the

behavior is temporary, and the workstation is not locked. If NIAKNETB.COM is used, the caption begins with "NiakNetB".

If for any reason NIAKNETW.DLL cannot be initialized, a message box displaying the caption:

**"NiakNetW Initialization Error"**

will appear, with diagnostic information about the last NetBIOS operation that did not succeed. Selecting "OK" will proceed as if NIAKNETW.DLL was not loaded. If the fingerprint is not installed locally, a single user RunTime will not share network files. Other RunTimes may refuse to run if NetBIOS is available.

The following are additional product notes regarding the NIAKNETW.DLL program.

1. NIAKNETW.DLL is required for NetBIOS-type user count checking and authorization sharing in WFW 3.11, which only provides NetBIOS services after MS-Windows starts up.
2. NIAKNETW.DLL is never loaded if security is authenticated from a Novell NetWare network.
3. NIAKNETW.DLL is never loaded for locally installed single user RunTimes, or if NIAKNETB.COM was loaded before MS-Windows is started.
4. Prior to NPL Revision 4.21, when NIAKNETW.DLL was in use, DOS sessions of the RunTime under MS-Windows would not have access to NIAKNETW's user count checking and authorization sharing.

**NOTE:** As of NPL Revision 4.21, NIAKNETB and NIAKNETW were revised to allow each to be run concurrently. Testing has shown that when multiple copies of NIAKNETB.COM are running, they may count as one user or multiple users. This is strictly dependent on the implementation of NetBIOS in use.

5. NIAKNETB.COM will not install in a DOS box if NIAKNETW.DLL has been previously loaded.
6. NIAKNETW.DLL will not operate if NIAKNETB.COM has already been started in a DOS box.

**Current NIAKNETB.COM Considerations** [4.30 / 5.00]

The following change has been implemented in NPL Release V which may have an impact on applications running in NetBIOS environments.

The Windows RunTime no longer attempts to locate or use a NIAKNETB.COM TSR that may have been installed before MS-Windows was loaded. Developers still using this configuration are advised to switch to the new NIAKNETW.DLL instead.

**WARNING:** NIAKNETB has been updated on this release. Developers should make sure all older versions of NIAKNETB.COM are removed from both the server and workstations to avoid possible conflicts with older revisions.

**Past NIAKNETB.COM Considerations** <sup>[4.10]</sup>

This section presents historical commentary on NIAKNETB.COM in consideration of those sites which may not be using a current revision of NPL.

NIAKNETB.COM must be used to run the MS-DOS based NPL RunTime programs in a DOS based NetBIOS environment.

To use NIAKNETB.COM, set up a batch file. The first line of the batch file should load NIAKNETB.COM and the last line should unload it.

For example, the batch file may look like this.

```
G:\BASIC2C\NIAKNETB.COM
G:\BASIC2C\RTI \D30 START.OBJ
G:\BASIC2C\NIAKNETB /U
```

The last line in the batch file above unloads NIAKNETB.COM. If that is not done, an error message will be displayed when the DOS window closes.

**NOTE:** The /U (unload) parameter is supported on version 1.7 or greater of NIAKNETB.COM. This version of NIAKNETB was introduced as part of NPL Revision 4.21.

**NOTE:** As of NPL Revision 4.21, NIAKNETB and NIAKNETW have been revised to allow each to be run concurrently. Testing has shown that when multiple copies of NIAKNETB.COM are running, they may count as one user or multiple users. This is strictly dependent on the implementation of NetBIOS in use.

The NIAKNETB batch file must be run on each workstation after the system has successfully loaded NetBIOS communications drivers, but before the NPL application is started.

**NOTE:** The above batch file may also be run in a MS-Windows NT DOS box, assuming appropriate NetBIOS protocols have been established.

### 3.3.2 Microsoft Windows NT Networks <sup>[4.20]</sup>

NPL Release V is tested and supported on Microsoft Windows NT 3.51 and greater. This section describes the networking components and property settings in Windows NT required to establish NetBIOS communications with the Niakwa RunTime.

#### Configuring Windows NT Servers and Workstations <sup>[4.20]</sup>

This section discusses the NetBIOS configuration requirements for NPL to operate correctly in a MS-Windows NT environment. The following process applies to both Windows NT servers and workstations.

The network configuration screen can be displayed by choosing Network from the Control Panel. Select the protocols tab. The protocol list displays the installed networking protocols. The following items are NetBIOS drivers required in the protocol list for NPL to operate correctly.

- NetBEUI Protocol
- NWLink IPX/SPX Compatible Transport
- NWLink NetBIOS

When NWLink, IPX/SPX Compatible Transport, and the NetBEUI protocols are loaded, Windows NT automatically loads NWLink NetBIOS. If the above protocols are not displayed, contact your system administrator to install the appropriate NetBIOS protocols.

**NOTE:** Other protocols may be listed in the protocols screen. The protocols referred to above are the required minimum for NPL to operate properly on an NT network. Other protocols such as TCP/IP are transparent to NPL and will not affect the operation of the RunTime.

This is a NetBIOS type network so NIAKNETW.DLL must be enabled for NPL to lock files and track terminal numbers correctly. Refer to section 3.3.1 above for details.

### **Protocol Stack Considerations**

Occasionally, even if properly configured, the RunTime may experience difficulty authenticating with the NT server from some NT workstations, resulting in an erroneous “Number of Authorized users exceeded” error message. This problem is typically related to the Lana Number configuration of NetBIOS on the NT workstation, if the first Lana number is bound to a wide-area network. To correct this problem on an NT 4.0 system:

1. Log in as administrator.
2. Open Properties of Network Neighborhood (or open the Control Panel/ Network applet).
3. Click the Services Tab.
4. Click on NetBIOS Interface.
5. Click the Properties... Button.
6. The network route associated with Lana number 000 should be:

NwlnkNb-> NwlnkIpx  
or  
Nbf->(your network Card Name).

If Lana number 000 is set to “NetBT->(something), it needs to be adjusted.

Niakwa strongly recommends that the network route associated with Lana number 000 be “NwlnkNb-> NwlnkIpx”. This is consistent with past recommendations for Windows for Workgroups. However, before adjusting Lana 000, you should check a working NT system to make sure you are in sync with other NT systems on the network.

**WARNING: Under no circumstances use a network route containing NetBT or Wan in the name.**

To correct the Lana Number:

1. Highlight the 000, and click the Edit Button.
2. Change the number to a high value (say, 9).

3. Highlight the number next to NwlnkNb-> NwlnkIpx, and edit this to say 0. To keep things tidy, you could change the 9 to keep all the numbers sequential. After you close the Networks dialog, you will need to restart the machine for the change to take effect.

### **Windows 95/98 Workstations** [4.20]

This section discusses the NetBIOS configuration requirements for NPL to operate correctly from a Windows 95/98 workstation attached to a Windows NT environment.

The network configuration screen can be displayed by choosing Network from the Control Panel. The list of networking components installed should appear similar to:

Client for Microsoft Networks  
(Your Network Adapter is shown here)  
IPX/SPX Compatible Protocol  
NetBIOS support for IPX/SPX Compatible Protocol  
NetBEUI  
File and Printer Sharing for Microsoft Networks

If the above protocols are not displayed, contact your system administrator to install the appropriate NetBIOS protocols.

**NOTE:** **Windows 95/98 does not automatically select a default protocol for you. In the Properties section for the IPX/SPX Compatible Protocol verify that the following 3 options are selected:**

- 1) **I want to Enable NetBIOS over IPX/SPX.**
- 2) **Set this protocol to be the default protocol.**
- 3) **In the Bindings setup, choose all the components to communicate using the NetBIOS protocol.**

**You must then restart the computer for the new settings to take effect.**

This is a NetBIOS type network so NIAKNETW.DLL must be enabled for NPL to lock files and track terminal numbers correctly. Refer to section 3.3.1 above for details.

### 3.3.3 Peer to Peer Environments [4.10]

The following sections describe various configuration considerations for the correct operation of NPL in a peer to peer NetBIOS environment.

#### Windows 95/98 Peer to Peer [4.20]

Configuring a Windows 95/98 workstation to run NPL in a peer to peer environment is no different than configuring the same workstation to run in an NT server environment. Refer to Section 3.3.2 above for details.

#### Windows for Workgroups [4.10.16]

When configuring WFW 3.11, at installation time most drivers will install two protocols that support NetBIOS calls:

1. IPX/SPX Compatible Transport with NetBIOS
2. Microsoft NetBEUI

Even though there is only one network card on the workstation, internally, these two protocols appear as if they are related to separate network cards. The NIAKNETB/NIAKNETW security sharing (and user count) always uses the primary NetBIOS LAN adapter number. For this reason it is very important that all WFW 3.11 users select the same "Default Protocol" (the first protocol listed in the network setup). If the "Drivers ..." button is clicked on, the default protocol is displayed.

**NOTE: If this guideline is not followed, security sharing across the network will not operate correctly.**

Niakwa has also been made aware of an incompatibility with revision 5.00 of the Microsoft Smart Drive program (currently shipping with MS-DOS 6.20 and WFW 3.11) that can, under some circumstances, cause MS-Windows to crash and return to a DOS prompt. If this situation is encountered, replace the SMARTDRV.EXE program with any version other than 5.00.

**NOTE: The version of SMARTDRV currently executing can be checked by executing the command "SMARTDRV /S".**

#### MS-DOS Peer to Peer Networks [4.10]

Niakwa has tested and supports Novell's NetWare Lite, Personal NetWare and Artisoft's LANtastic NetBIOS compatible networks. Many other NetBIOS compatible networks will likely operate as well, but are not officially tested or supported by Niakwa.

The performance of NetBIOS networks is typically much slower than a comparable configured Novell NetWare installation. As such, expect performance of the NetBIOS based RunTimes and the applications to follow suit.

**NOTE:** **Disk I/O performance can be improved (especially in Windows environment) by implementing atomic file locking. Refer to Chapter 8, \$OPTIONS Byte 39 for details.** [4.10.23]

Installation of the NPL security fingerprint from a workstation to a NetBIOS file server can cause the NetBIOS RunTime to behave inappropriately. The NPL security fingerprint must always be installed "locally" (i.e., install using the server as the workstation). Refer to Section 2.6 of the NetBIOS Addendum for details.

The NPL Once-a-Day security check can behave incorrectly (i.e., Once-a-Day security appears not to work), if all workstations on the NetBIOS network are not set to the same date.

NetWare Lite's Cache, NLCACHE, can cause the system to lock when executing the NIAKINST or NIAKRCLL security programs. The RunTime can also lock the system under certain situations when using NLCACHE. Because of this situation, Niakwa recommends not using the NLCACHE program.

**NOTE:** **Niakwa has not tested the above described NLCACHE problem with Niakwa's NPL security.**



The write cache of the MS-DOS SMARTDRV command should be turned off to allow NDM to work correctly under some NetBIOS operating environments. If an NDM application is unable to write new files to the hard drive, turn off the write cache option of the SMARTDRV command.

Novell has recently discontinued the NetWare Lite product and replaced it with Personal NetWare. Niakwa's testing has shown the Personal NetWare performs identically to NetWare Lite in regards to the use of the NPL NetBIOS RunTime and is fully supported by Niakwa.

Personal NetWare uses Novell's Universal Client (VLM) software. By default the NETX.VLM module is loaded by Personal NetWare. When the NETX.VLM module is loaded, the RunTime incorrectly determines that an Novell NetWare network is present. When this occurs a NetBIOS Runtime (non-Niakwa Network RunTime) will not run. To resolve this problem it is necessary to add the following statement to the "Netware DOS Requester" section of the NET.CFG file.

**EXCLUDE VLM = NETX.VLM**

### 3.4 Novell Network Considerations [4.30 / 5.00]

The following table lists the workstation client software required by NPL to operate correctly in a Novell NetWare network.

Workstation Operating System	MS-Windows 32-bit RunTime	MS-Windows 16-bit RunTime	386/DOS-Extender RunTime	MS-DOS RunTime
MS-DOS	Not Available	Not Available	Novell NetWare VLM or Novell's Client 32 for MS-DOS	Novell NetWare VLM or Novell's Client 32 for MS-DOS
MS-Windows	Not Available	Novell NetWare Client or MS-Windows	Not Available	Not Available
MS-Windows 95	Novell NetWare Client 32	Novell NetWare Client 32	Not Available	Not Available
MS-Windows 98	Novell IntranetWare Client 32	Not Supported	Not Available	Not Available
MS-Windows NT 3.51 or greater	Novell Client 32 for Windows NT	Novell Client 32 for Windows NT	Not Available	Not Available

**NOTE:** Prior to NPL Release V, all MS-Windows NT systems have been required to to run using the Novell NetWare Client 32 for MS-Windows NT, while Windows 95 systems managed to operate correctly using Microsoft's "Client Services for NetWare".

Developers are cautioned that Microsoft Client Services for NetWare drivers are no longer adequate for NPL Release V to correctly operate under MS-Windows 95/98, and are advised to migrate to the readily available Novell Client software.

## 3.5 Mixed Network Considerations

This section discusses considerations to take into account when installing NPL into a mixed network environment. A mixed network is defined as a Network with both Novell and NetBIOS protocols running on it

### 3.5.1 Running in a Peer Network attached to a Novell Network

When running NPL on a peer to peer network attached to a Novell network, it is necessary to use a “Supported Networks” instead of a “NetBIOS” RunTime. Even though NPL is only running in the “NetBIOS environment”, NPL will “see” the Novell network. A “NetBIOS” RunTime will return an error “Not authorized for use with NetWare” when trying to run in this environment.

In a mixed environment if the workstation is logged on to the Novell network the RunTime will first attempt to authenticate with the Novell server. If that fails, the RunTime then attempts to authenticate using the NetBIOS layer.

### 3.5.2 Running in a Network with both Novell and NT Servers

When running NPL on a network with both Novell and NT servers, it is necessary to use a “Supported Networks” RunTime instead of a “NetBIOS” run time. Even though NPL may only be running from the NT server, NPL will “see” the Novell network and attempt to use Novell for file locking and to track user counts. A “NetBIOS” only RunTime will return an error “Not authorized for use with NetWare” when trying to run in this environment.

In a mixed environment if the workstation is logged on to both the Novell server and the NT server, the RunTime will attempt to use Novell for file locking and to track user counts. If the workstation is only logged on to the NT server, the RunTime will attempt to establish file locking and user counts using the NetBIOS layer.

### 3.5.3 Running a Stand-Alone Single User RunTime on a Network

The following discusses the implications of installing a single user RunTime on a network server.

#### Novell NetWare

All single-user RunTimes are “NetWare enabled”. They can be installed on a NetWare server and accessed by any single workstation.

#### Windows NT

A single-user RunTime may not be accessed remotely when installed to a Windows NT server. If a RunTime is installed on a MS-Windows NT server, it may only be accessed from the server. The RunTime must be installed locally on the machine that will access NPL. Security has to be installed on the local drive in order to function.

## 3.6 Wide Area Network Considerations

Since NetBIOS is not supported across Wide Area Networks (WAN's), it is not possible to remotely access NPL across a WAN when the Gold Key is installed at the central host site.

The proper method to configure NPL to operate in a WAN environment is to install single or multiuser RunTimes at the remote sites, thus authenticating RunTime security locally using an appropriate NetBIOS configuration. Once security has been passed, the remote sites can access systems across the WAN, given appropriate remote drive mappings and privileges have been established.

**NOTE:** NetBIOS is required by NPL to count and determine unique terminal identification in a peer environment. While NPL does not provide direct support for TCP/IP, disk I/O requests are passed to the native operating system, and it decides which transport is used to route the disk request. Whether that request is routed via TCP/IP or NetBIOS is transparent to NPL.

## 3.7 MS-Windows Terminal Server Considerations

Since all sessions running in a terminal server environment appear as local sessions to NPL, terminal sever sessions should be started with the /T command line parameter to assure appropriate terminal identification across the network.

**NOTE:** All Gold Keys installed on a Windows Terminal Server system require that a functional network protocol is installed. If no suitable network protocol is detected, the RunTime will exit with the following message:

**“On a Windows Terminal Server system a network protocol must be available”**

This page intentionally left blank



## CHAPTER 4

# RUNTIME UPGRADES

### 4.1 Overview

The NPL Universal Upgrade is designed to allow existing NPL RunTimes to be upgraded to the current NPL version in an easy manner. The upgrade procedure is fully automated, replaces all previous Gold Key diskettes and may be used to increase the capabilities of the previous Gold Key. Upon completion of the NPL Universal Upgrade, the NPL RunTime Replacement Gold Key Diskette is a fully licensed Gold Key and replaces the old NPL Gold Key.

Section 4.2 discusses the NPL Universal Upgrade procedure.

Section 4.3 discusses the NPL Limit Upgrade.

The following lists some of the benefits of the NPL Universal Upgrade.

- The NPL Universal Upgrade is registration based. New capabilities (i.e.: platform, user counts, etc.) may be selected when the registration code is requested from Niakwa. This allows the NPL Universal Upgrade Packages to be stocked by vendors and customized for each installation as required, using a unique registration code.
- The NPL Replacement Gold Key Diskette inherits all characteristics of the old NPL Gold Key (i.e., keeping the same serial and Gold Key number), and allows additional functionality to be added, depending on the upgrade requested.
- Once the NPL Replacement Gold Key Diskette is in place, future enhancements such as enabling additional NPL options can be done in place without requiring a new Gold Key.

**NOTE:**        **The old NPL Gold Key being upgraded is disabled as part of the final process of this procedure and should be disposed of to avoid future confusion with the new NPL RunTime Replacement Gold Key Diskette.**

## 4.2 The NPL Universal Upgrade <sup>[4.21]</sup>

NPL Release V improves on the flexibility of the Niakwa Universal Upgrade procedure. To insure the best results when running a Universal Upgrade, always verify that the NIAKWA\_RUNTIME environment variable is set to your target RunTime upgrade directory. If NIAKWA\_RUNTIME is not set, then the Universal Upgrade must always be run from an MS-DOS Window or MS-DOS, and the current directory must be the target NPL directory.

**NOTE:**        **This section discusses key environmental issues related to the NPL Release V Universal Upgrade procedure. Specifically, consideration is given to the operating system the Universal Upgrade is performed on and the revision of the old NPL Gold Key being upgraded.**

**Based upon developer input and a lack of upward compatibility issues, the Trial Installation period option of the Niakwa Universal Upgrade has been removed from NPL Release V.**



### 4.2.1 Upgrade Considerations

While the NPL Universal Upgrade is a simple and convenient upgrade process, there are several considerations which should always be checked prior to performing the NPL Universal Upgrade.

- When upgrading older NPL Gold Keys to NPL Release V (revisions prior to 4.20), you must first verify that the Gold Key Security is operational on the system you intend to upgrade and you must be able to successfully install/recall the Gold Key from that system as well.

If a system has been upgraded to Windows 95/98/NT, take steps to assure that the old Gold Key can be put on a system that allows the old Gold Key security to be installed/recalled correctly, thus assuring the Universal Upgrade will not encounter problems. The reason for this is that security prior to revision 4.20 would fail on many systems, especially if they had been recently upgraded.

- When ordering an upgrade, take time to run the SHOWNPL batch file on the system with the Gold Key you intend to upgrade. This guarantees the accuracy of the information you provide to Niakwa, thus eliminating the small (but potential) possibility of receiving an incorrect upgrade registration code.
- Never attempt to run the Universal Upgrade from a MS-DOS box under Windows when upgrading an old security Gold Key. Older NPL security will fail in this circumstance, thus assuring the Universal upgrade will fail. Note that on revision 4.20 Gold Keys and greater, running the universal upgrade in a MS-DOS box is acceptable and recommended.

**NOTE:** By taking the above considerations into account prior to performing a Universal Upgrade, you eliminate the most common problems associated with failed Universal Upgrades, thus reducing your overall time spent performing this relatively short and convenient process.

#### 4.2.2 General Requirements

For the upgrade procedure to be successful, the following is required:

- The NPL Gold Key being upgraded must be Revision 3.xx or greater.
- The Gold Key security must be recalled to the original Gold Key diskette.
- The original RunTime must be functional on the system where the UPGRADE is being performed.
- The Upgrade registration code must be obtained from Niakwa.

**NOTE:** The RunTime revision appears on the NPL start-up screen when the RunTime is initially executed (Revision 3.xx only.). In addition, the RunTime revision number and RunTime type appear on the original Gold Key diskette label.

**HINT:** It is necessary to have the Certificate of License and Authenticity present before beginning the upgrade procedure. Under certain circumstances, it may be possible to obtain the registration code at the time the upgrade is being performed. However, it is highly recommended that the upgrade registration code be obtained prior to performing the upgrade in order to avoid potential delays.

**NOTE:** If NPL security is unable to be recalled for any reason, contact your Authorized Niakwa Distributor for information on obtaining a RESET prior to the upgrade procedure.

### 4.2.3 Current Upgrade Considerations [4.30/5.00]

This section discusses known issues with the Universal Upgrade procedure.

- When the Universal Upgrade is run within a DOS box while in Windows 95/98, the final “Congratulations” screen is displayed, but hidden in the background before the SETUP procedure is completed. When SETUP is complete, select back to the DOS task and complete the final screen as instructed.
- If the Universal Upgrade is directed to a remote network directory, the program will warn you that all upgrades directed to a Windows NT server **MUST** be performed locally on that system. If you proceed, no error occurs but the installed authorization is vulnerable to programs such as virus scans that can damage the installed security.
- After installing via the Universal Upgrade, the SETUP.INI file on disk 1 contains new default installation options which instruct SETUP:
  1. Do not install security (since it has previously been installed.)
  2. Use the default installation directory (if NIAKWA\_RUNTIME is not set in the Windows environment), used by the Universal Upgrade.

These options are reset to standard values after the next time SETUP is run from the Gold Key diskette.

### 4.2.4 Upgrading Old Security RunTimes [4.21]

All NPL RunTime Gold Keys prior to NPL Revision 4.20 require the Universal Upgrade to be performed from a MS-DOS prompt (not a DOS-Window), and the old NPL RunTime must be functional on the target system (i.e., security can be installed and recalled).

**Upgrading in MS-DOS**

When the Universal Upgrade is run in this environment, the upgrade procedure upgrades the Gold Key diskette, then expands all non-Windows files from the new Gold Key diskette to the target directory. Upon completion, if the RunTime was licensed for MS-Windows, the Universal Upgrade instructs you to execute MS-Windows and run SETUP from the new NPL Gold Key to complete the install.

**NOTE:** Upon completion of the Universal Upgrade, NPL security is installed. If at this time you need to move the new NPL Gold Key to a new system, perform the following:

1. **Recall NPL security to the new Gold Key diskette by using NIAKRCLL.BAT.**
2. **Run the NPL Setup program on the new target system.**
3. **In the Miscellaneous Options section of the SETUP program, enable the Install Security Option. This is required since this option is disabled as part of the upgrade procedure (since security has already been installed.)**

Revision 3.xx, 4.00, and 4.10 RunTimes all use older security. Some of these older RunTimes were already installed on 16-bit operating systems which were later upgraded to 32-bit O/S's like Windows 95. These RunTimes continued to function but could not be recalled/installed in a 32-bit environment. There are some special considerations for upgrading these older RunTimes.

1. The original RunTime must be operational on the system where the Upgrade is being performed.
2. The Upgrade must be performed from a true DOS prompt, not a DOS box under Windows.
3. If this is a Windows RunTime, once the DOS portion of the installation is complete, Windows must be started and A:\SETUP must be run to complete the Upgrade.

#### 4.2.5 Upgrading New Security RunTimes <sup>[4.30/5.00]</sup>

For NPL Gold Keys at or above NPL revision 4.20, the NPL Universal Upgrade can be run in a MS-Windows environment. This section discusses issues specific to the MS-Windows environment the upgrade is executed from.

##### Upgrading in Windows 3.1

When the Universal Upgrade is run from MS-Windows 3.1, the upgrade procedure upgrades the Gold Key diskette. Upon completion, the Universal Upgrade instructs you to execute MS-Windows and complete the upgrade procedure by running SETUP from the new NPL Gold Key to complete the install (this operation cannot be launched by a DOS program in this environment.)

##### Upgrading in MS-Windows 95/98 or Windows NT

When the Universal Upgrade is run from Windows 95/98 or Windows NT, the upgrade procedure first upgrades the Gold Key diskette. Upon completion, the Universal Upgrade launches SETUP automatically to complete the upgrade process.

RunTimes with Revision 4.20 and greater operate with new security which is capable of operating in a 32-bit environment. The new security allows some functional differences over the old style security used with Release 4.10 and earlier.

1. The Upgrade Process can be started in a DOS Window.
2. Under Windows 95/ 98 and Windows NT if the upgrade process is started in a DOS Window, the Upgrade process will automatically launch A:\SETUP to complete the Upgrade process.

**NOTE:** Under Windows 3.1 and 3.11 (Windows for Workgroups) A:\SETUP must be run manually to complete the upgrade.

#### 4.2.6 How to obtain an Upgrade Registration Code [4.22]

In most instances, the Universal Upgrade will ship with a "Certificate of License and Authenticity" which contains the Upgrade Registration Code required to activate the Universal Upgrade.

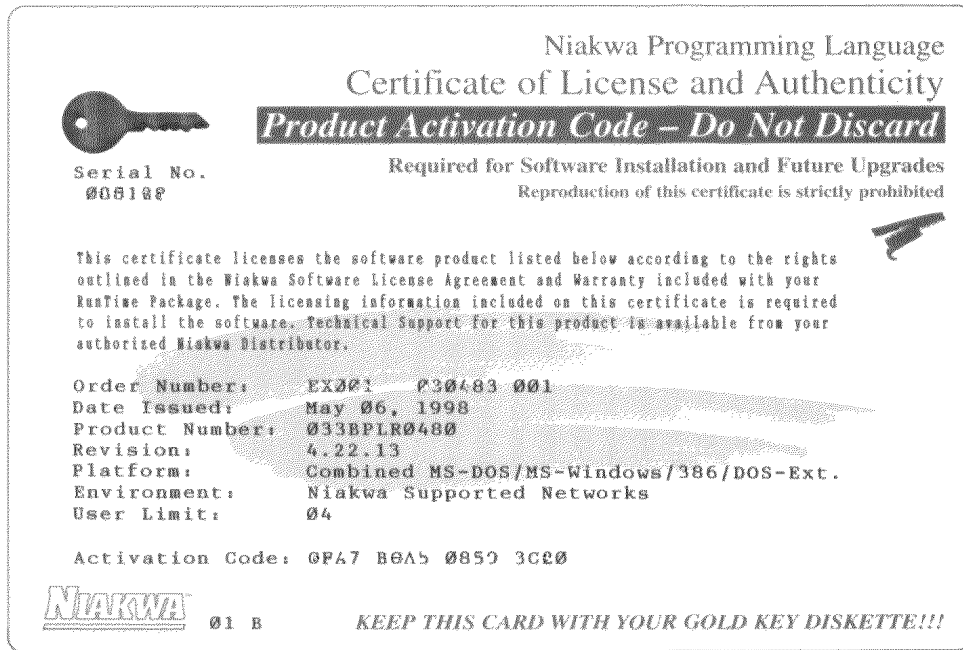


Figure 4-1 Certificate of License and Authenticity

The Upgrade Registration Code is generated based on information you provide about the current RunTime, when placing your order for the Universal Upgrade. The best way to be sure that you provide the most accurate information about the current Gold Key is to run SHOWNPL on the current system before placing your order.

**SHOWNPL**

The SHOWNPL utility (shown in Figure 4-2) is used to display information about the currently installed RunTime, which is required to obtain an Upgrade Registration Code.

```

Command Prompt - shownpl
===== Niakwa License Configuration =====
Checking the current RunTime License

  NPL Revision:          4.2x
  Number of Authorized Users: 16
  Licensed Products:    MS-Windows
                       386/DOS-Extender
                       Novell

  RunTime Serial Number: 204717
  Product Number:       033BPLR1680

The above information is required to obtain an upgrade registration code.
Please write it down and provide your developer with this information.

Press any key to exit...
  
```

*Figure 4-2 SHOWNPL Screen*

The SHOWNPL utility can be found on:

- All Revision 4.20 and greater Gold Keys
- All NPL Release IV and V Upgrade diskettes (white disk) in the \BASIC2C directory
- On Niakwa's Web Site ([www.niakwa.com](http://www.niakwa.com))

The SHOWNPL utility consists of the following three files:

```

SHOWNPL.BAT
UPGRTEST.OBJ
UPGRTEST.TXT
  
```

Copy these files into the installed RunTime directory and run SHOWNPL. Record all information and provide it to Niakwa at the time of ordering the upgrade. This process assure that you are provided with a correct Upgrade Registration Code

#### 4.2.7 Overview of the NPL Universal Upgrade

The NPL Universal Upgrade program performs the following tasks depending on the options selected.

- Verifies that NPL security has been recalled to the old NPL Gold Key. Refer to Chapter 2 for details on recalling NPL security.
- Prompts for an upgrade registration code supplied by Niakwa
- Optionally displays information required to obtain a registration code
- Copies all upgrade software to the NPL drive:\directory
- Copies the old NPL Gold Key serial number to the new NPL RunTime Replacement Gold Key Diskette
- Installs the new NPL security from the NPL RunTime Replacement Gold Key diskette to the NPL drive:\directory containing the RunTime files
- Prompts the user to record appropriate information on the NPL RunTime Replacement Gold Key diskette label for future reference
- Instructs the user to dispose of the old NPL Gold Key diskettes upon completion of the upgrade
- When applicable, automatically launches Windows SETUP or prompts the user to load Windows and run SETUP depending on the version of MS-Windows in use

To insure the best results when running a Universal Upgrade, always verify that the NIAKWA\_RUNTIME environment variable is set to your target RunTime upgrade directory. If the NIAKWA\_RUNTIME environment variable is not set, then the Universal Upgrade must always be run from a MS-DOS-Window or MS-DOS, and the current directory must be the target NPL directory.

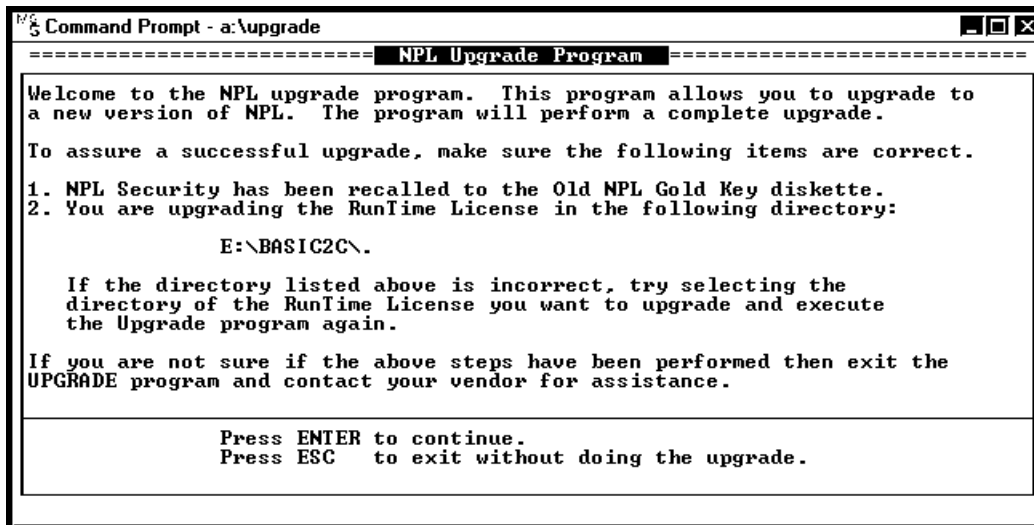


#### 4.2.8 Installing the NPL Universal Upgrade Program

To install the UPGRADE program, place the “NPL Upgrade Diskette” in the diskette drive and enter the following (presuming the current directory is E:\BASIC2C):

```
E:\BASIC2C> A:\UPGRADE
```

The screen in Figure 4-3 is displayed:

The image shows a screenshot of a Windows Command Prompt window titled "Command Prompt - a:\upgrade". The window displays the "NPL Upgrade Program" screen. The text on the screen is as follows:

```
===== NPL Upgrade Program =====  
Welcome to the NPL upgrade program. This program allows you to upgrade to  
a new version of NPL. The program will perform a complete upgrade.  
To assure a successful upgrade, make sure the following items are correct.  
1. NPL Security has been recalled to the Old NPL Gold Key diskette.  
2. You are upgrading the RunTime License in the following directory:  
    E:\BASIC2C\  
If the directory listed above is incorrect, try selecting the  
directory of the RunTime License you want to upgrade and execute  
the Upgrade program again.  
If you are not sure if the above steps have been performed then exit the  
UPGRADE program and contact your vendor for assistance.  
-----  
Press ENTER to continue.  
Press ESC  to exit without doing the upgrade.
```

Figure 4-3 Initial Upgrade Screen

Pressing <ESC> at this screen exits the Upgrade program.

Pressing <ENTER> displays the Registration Entry screen shown in Figure 4-4.

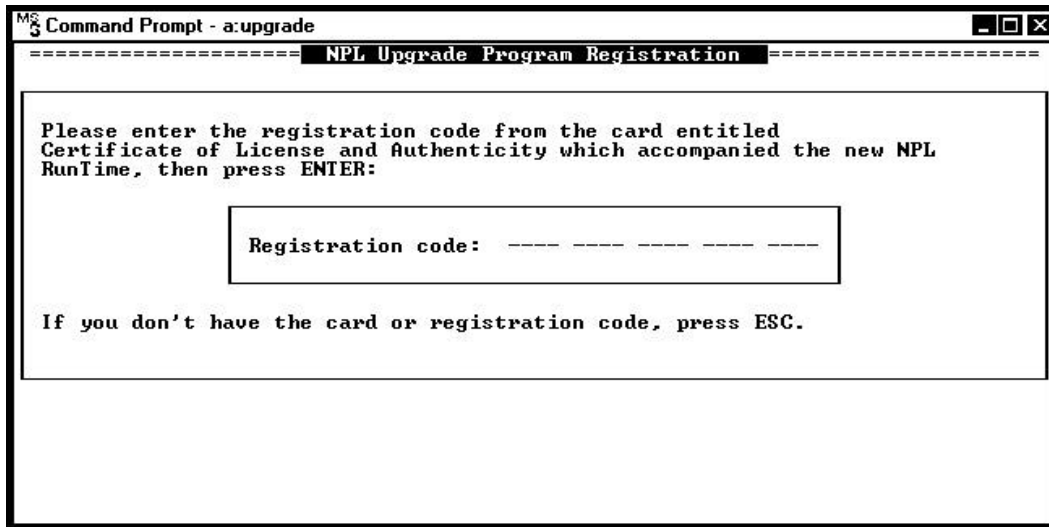


Figure 4-4 Registration Entry Screen

Enter the registration code obtained from Niakwa or from the “Certificate of License and Authenticity” shipped with the upgrade. If the upgrade registration code is valid the upgrade procedure will begin. If you do not have a registration code, pressing <ESC> displays the “RunTime Information Screen” shown in Figure 4-5.

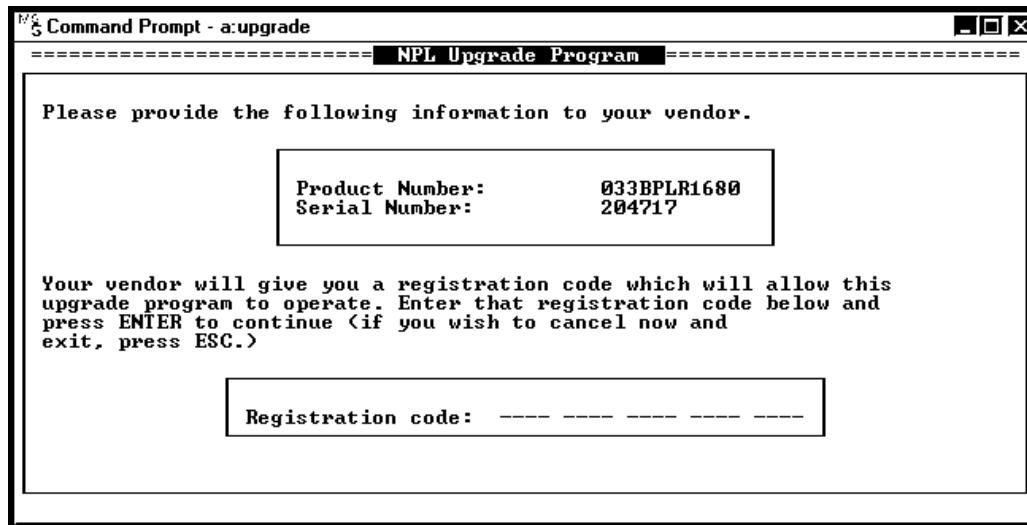


Figure 4-5 RunTime Information Screen

The information in the “RunTime Information Screen” is required by Niakwa to generate a valid Upgrade Code, in the event that this information was not provided at the time of the upgrade order.

**NOTE:** **SHOWNPL supplies information about user counts and various RunTimes authorized on this Gold Key. For that reason it is recommended the SHOWNPL be used to obtain the information needed to generate the Upgrade Registration Code.**

Pressing <ESC> exits the program.

Run the NPL Universal Upgrade again after you have received the registration code from Niakwa or you may enter the registration code in this screen. Once a valid registration code is entered, the upgrade will proceed.

Upon completion, the NPL RunTime Replacement Gold Key Diskette is upgraded to the current version of NPL, and the old NPL Gold Key diskette is disabled. The NPL RunTime Replacement Gold Key Diskette is a fully functional Gold Key and may be installed and recalled just like any standard NPL Gold Key. Refer to Chapter 2 for information on installing and recalling the upgraded RunTime.

**IMPORTANT:** Make sure the serial number is recorded on the NPL RunTime Replacement Gold Key Diskette label. Store the new Gold Key, the supplementary diskettes, and Certificate of License and Authenticity together in a safe location.

## 4.3 Limit Upgrades <sup>[4.21]</sup>

The NPL Limit Upgrade program allows NPL developers to add additional NPL platforms and increase the user license of NPL Gold Keys in the field. The LIMIT program, (included on all new and upgrade NPL RunTime packages) is used to extend the current user limit or functionality of a Gold Key in the field by inputting a license code obtained from Niakwa.

### 4.3.1 Limit Upgrade Requirements

For the Limit Upgrade procedure to be successful, the following is required:

- The NPL Gold Key being upgraded must be Revision 4.21 or greater.
- The original Gold Key diskette must be available and on hand.
- The Limit Upgrade code must be obtained from Niakwa.

**NOTE:** Revision 4.21 RunTimes allowed only the user limit to be increased on the current license. Support for adding additional NPL platforms is available on Revision 4.22 RunTimes and greater.

### 4.3.2 Requesting the Limit Upgrade Code from Niakwa

To obtain the "Limit Upgrade Code", you must first run the LIMIT upgrade program and provide the resulting information to Niakwa.

**NOTE:** The LIMIT Upgrade procedure may optionally be run from MS-Windows. The MS-Windows RunTime installation automatically includes an icon to perform this process. Refer to Chapter 2 for details.

For example, assuming you are selected to the current NPL drive/directory and the Gold Key is inserted in floppy drive A, enter the following command line:

```
C:\NPL>LIMIT A:
```

The NPL Limit Upgrade screen is displayed in Figure 4-6.

You must enter an Limit Upgrade Code to upgrade the RunTime User Limit.

To get your Limit Upgrade Code, please contact NIAKWA, INC. and provide the following information:

Current User Limit: 4  
Current Upgrade Level: 7  
RunTime Serial Number: 123456  
License Code: 2

Required User Limit: [you must decide this]  
NetBIOS or Novell: [you must decide this]

If you do not have a Limit Upgrade Code, you may press TAB to quit.

Enter your Limit Upgrade Code: AAAA BBBB CCCC DDDD

*Figure 4-6 Limit Upgrade Entry Screen*

The information provided in Figure 4-6 must be provided completely to Niakwa to obtain a license code for the requested new functionality. Once all information has been submitted, you will be provided with a 16 digit code to be entered in the final field noted above. Upon entering the license code, the procedure will confirm the success or failure of the upgrade process.

### 4.3.3 The Certificate of License and Authenticity

Typically a Limit Upgrade is performed on an installed RunTime. In most cases the Limit Upgrade code can be requested by phone during normal business hours. While the actual Upgrade Limit Code may be returned to you by phone or fax, the original Certificate of License and Authenticity will be included with the invoice. The Certificate of License and Authenticity should be stored in a safe place with the Gold Key Diskette.

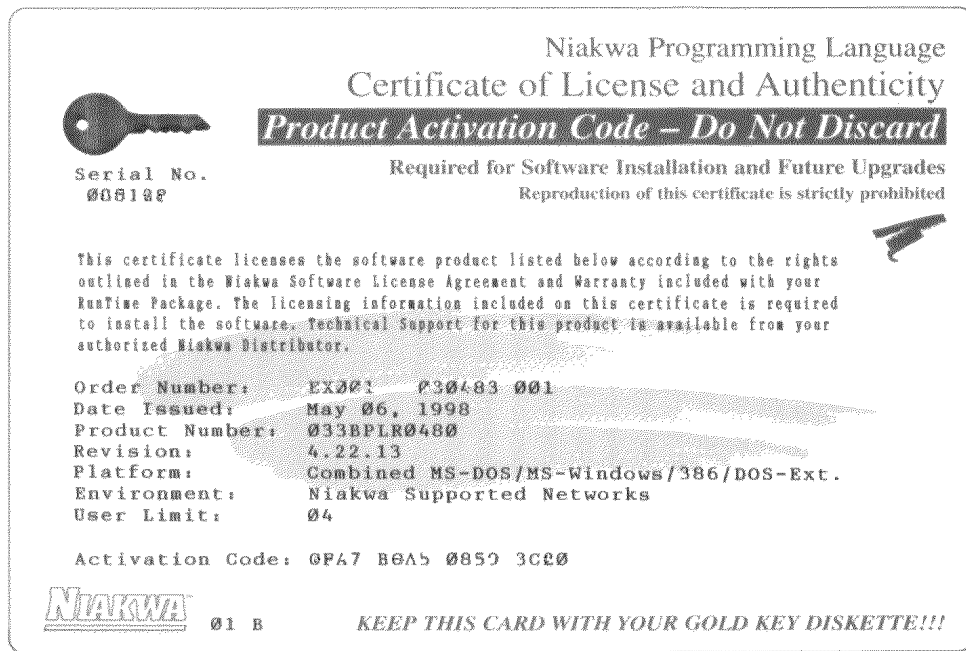


Figure 4-7 Certificate of License and Authenticity

#### 4.3.4 Running the NPL Limit Upgrade Program

To run the LIMIT Upgrade program:

1. From a DOS prompt select the RunTime directory
2. Insert the Gold Key in floppy drive A.
3. Enter:

```
C:\NPL>Limit A:
```

4. The LIMIT UPGRADE screen will be displayed again.
5. Enter the Limit Upgrade Code obtained from Niakwa.

**WARNING:** This procedure attempts to update both the installed copy of the RunTime and the Gold Key diskette. Developers should insure that the NPL Gold Key is available in the diskette drive prior to running the LIMIT procedure. The Limit program will display a warning if it is unable to update the Gold Key on the floppy drive. In this instance, the developer is advised to update the Gold Key diskette with the updated NIAKSER.DAT file on the system.

This page intentionally left blank





## CHAPTER 5

# RUNTIME OPERATION

### 5.1 Overview

This chapter discusses operational differences and enhancements in the NPL RunTime which have changed since the introduction of NPL Release IV.

Section 5.2 discusses general operational enhancements to the operation of the NPL RunTime.

Section 5.3 discusses updates and enhancements specific to the NPL Line Editor.

Section 5.4 cover updates and changes to the NPL Compiler.

Section 5.5 documents new RunTime error codes.

## 5.2 The RunTime Environment

This section discusses several operational differences introduced since the introduction of NPL Release IV. Many of these changes have been implemented to enhance NPL's support of multi-tasking and mixed network environments.

### 5.2.1 Shared vs. Exclusive Access <sup>[4.10]</sup>

NPL opens files using the SHARE category of file locks. This is a significant change from past revisions which would always open files using the Exclusive category on single-user systems. This means that multiple workstations with single-user RunTimes installed locally may now share files with other RunTimes in mixed network environments.

### 5.2.2 Implied \$BREAK after Disk I/O <sup>[4.10]</sup>

\$OPTIONS byte 14 (implied \$BREAK after disk I/O) is no longer ignored under MS-DOS. In a multi-session environments such as DOS tasks under MS-Windows or mixed network environments, setting this byte to HEX(01) will improve interleaved access to files. The default remains HEX(00).

### 5.2.3 Two Types of RunTimes <sup>[4.10]</sup>

The following sections discuss how the RunTime determines the mode in which it opens a file and whether or not the user count is incremented. To understand this, it is first necessary to understand the components that make up the decision tree logic NPL uses at startup to determine file locking and user counts.

Past revisions of NPL were distributed as single user MS-DOS RunTimes or multi-user Novell NetWare RunTimes. As of NPL revision 4.10, NPL for MS-DOS environments is distributed as a NetBIOS RunTime (available in 1 to 8-user increments) or a Niakwa Networks RunTime. The two types of RunTimes are described below:

NetBIOS RunTime	Operates in a standalone single user or multi-user NetBIOS environment.
Niakwa Networks RunTime	Has all the capabilities of the NetBIOS RunTime plus the ability to operate on a Novell NetWare network.

**NOTE:** All single-user RunTimes are treated as Niakwa Networks RunTimes. This conforms to the previous functionality of NPL MS-DOS RunTimes. The multi-user NetBIOS RunTimes (3, 4, and 8 user) will not operate when Novell NetWare drivers are detected and the NPL security is not passed locally (i.e., from the local hard drive or from the Gold Key diskette).

#### 5.2.4 Three File Opening Modes <sup>[4.10]</sup>

The RunTime is able to open files in three different modes. These include:

Exclusive	This is the mode in which all single-user NPL RunTimes would open files prior to Revision 4.10 of NPL. In this mode, only one workstation can access the file (i.e., other tasks will get an error if they attempt to access open files.)
-----------	---

**NOTE:** Exclusive mode is no longer used by current NPL revisions.

Share	NPL accesses files using SHARE-compatible lock calls.
Novell	NPL opens files in shared mode and accesses files using Novell NetWare-compatible lock calls.

### 5.2.5 User Counts [4.10]

NPL provides additional methods for tracking the number of active NPL users on the system. These methods are described below:

Fingerprint	The user count is not tracked. This mode is invoked when the security is passed locally (i.e., from a local drive or Gold Key diskette) and the NIAKNETB TSR is not present.
-------------	--

For example, a laptop with single-user NPL RunTime installed on it could be connected to a network with a 4 user NPL RunTime installed on it. Execution of the single user RunTime would not increment the user count on the network.

This method can also be used with NetBIOS environments that may encounter conflicts with the NIAKNETB TSR. In this scenario, a single-user NPL RunTime could be installed at each workstation and security would be passed locally. In this case, file sharing and other multi-user features are enabled.

**NOTE: This method is not available for installations under Windows Terminal Server or other multi-user extensions to MS-Windows. On such sites the fingerprint is always treated as non-local except if the fingerprint on the Gold Key diskette is used.**

NIAKNETB	The user count is incremented by NIAKNETB/NIAKNETW. This mode is invoked when either the DOS TSR or Windows DLL is present and Novell NetWare drivers are not detected.
----------	---

SEMAPHORE	The user count is incremented by a NetWare Semaphore. This mode is invoked upon detection of a Niakwa Networks RunTime, when Novell NetWare drivers are present and security is not passed locally (i.e., from a local drive or Gold Key diskette).
-----------	---

UNCOUNTED	The user count is not tracked.
-----------	--------------------------------

As the above methods illustrate, NPL conforms to many different configurations. To simplify system planning, the following tables have been designed to provide developers with a clear layout of the startup functionality of NPL.

Figure 5-1 illustrates the startup behavior of a NetBIOS RunTime. The first three columns represent the environment considerations NPL takes into account at startup, while the last two columns illustrate the user count and file sharing behavior (for non-local files) used by the RunTime. Figure 5-2 illustrates the startup behavior of a Niakwa Networks RunTime.

RunTime Environment			Startup Behavior	
NetWare Detected	NIKNETB Detected*	Fingerprint Local	Count Method	Share Method
N	N	N	WON'T_RUN-0	N/A
N	N	Y	FINGERPRINT (Local)	SHARE
N	Y	N	NIKNETB	SHARE
N	Y	Y	NIKNETB	SHARE
Y	N	N	WON'T_RUN-1	N/A
Y	N	Y	FINGERPRINT (Local)	NOVELL
Y	Y	N	WON'T_RUN-1	N/A
Y	Y	Y	FINGERPRINT (Local)	NOVELL

*Figure 5-1 Startup Behavior of a NetBIOS RunTime*

- \* The presence of NIKNETB.COM is ignored if the user limit is 1, and a Windows Terminal Server environment is not detected.

RunTime Environment			Startup Behavior	
NetWare Detected	NIKNETB Detected*	Fingerprint Local	Count Method	Share Method
N	N	N	UNCOUNTED	EXCLUSIVE
N	N	Y	FINGERPRINT (Local)	SHARE
N	Y	N	NIKNETB	SHARE
N	Y	Y	NIKNETB	SHARE
Y	N	N	SEMAPHORE	Novell
Y	N	Y	FINGERPRINT (Local)	Novell
Y	Y	N	SEMAPHORE	Novell
Y	Y	Y	FINGERPRINT (Local)	Novell

*Figure 5-2 Startup Behavior of a Niakwa Networks RunTime*

\* The presence of NIKNETB.COM is ignored if the user limit is 1, and a Windows Terminal Server environment is not detected.

#### **Description of Count Method**

The following briefly describes the count methods indicated in Figure 5-1 and Figure 5-2.

FINGERPRINT  
(LOCAL)

Refer to the earlier description under User Counts.

NIKNETB

Refer to the earlier description under User Counts.

SEMAPHORE

Refer to the earlier description under User Counts.

WON'T\_RUN-0

NPL displays one of three messages depending on which of the following conditions is encountered.

If the installed fingerprint is on a Windows Terminal Server, NPL exits with the message:

**On a Windows Terminal Server system, a network protocol must be available.**

If the license is for a single user or no NetBIOS protocol is detected, NPL exits with the message:

**Authorization must be installed on a local drive.**

If user limit is not 1 and the NetBIOS RunTime is detected (but the NetBIOS TSR is not loaded), NPL exits with the message:

**NetBIOS is running but NIAKNETB.COM is not. Authorization must be installed on a local drive.**

To correct the above problem, install the NetBIOS TSR prior to running NPL.

WON'T\_RUN-1

Exits with the message:

**This NPL RunTime is not authorized for use with Netware. Authorization must be installed on a local drive.**

In addition to using the above tables, developers may determine the status of the NPL RunTime environment and startup behavior under program control by using Byte 32 and 33 of \$MACHINE. Byte 32 is used to examine the specific RunTime environment encountered at startup, while Byte 33 is used to determine the specific startup behavior applied to the RunTime at startup. Refer to Chapter 8 for a complete description of possible values contained in these new \$MACHINE bytes.

### 5.2.6 Mixed RunTime User Counts <sup>[4.21]</sup>

NPL has been enhanced to maintain user counts on a Novell NetWare server or using NIAKNETB/NIAKNETW on a peer-to-peer network using the specific serial number of the license in use. As a result, multiple RunTime licenses may now operate independently of one another (provided they do not have the same serial number).

**WARNING:** In the event a Revision 4.21 version of NPL is used on a system running an older version of NPL, a count is charged to both the old and new user count when a Revision 4.21 RunTime or greater is installed. However, only the count of the current serial number is checked against the license user limit. Thus, versions of NPL prior to Revision 4.21 will continue to see user counts reflecting the total number of users on the system. Sites running multiple licenses on the same network or over mixed networks are encouraged to upgrade to a current NPL revision to take advantage of this change.

**NOTE:** The updated version of NIAKNETB.COM and NIAKNETW.DLL (v 1.7) must be used with Revision 4.21 of NPL. Older versions of NPL may also use these newer versions, but will not benefit from the user count changes.

### 5.2.7 New RunTime Startup Search Procedure <sup>[4.22]</sup>

The environment variable NIAKWA\_RUNTIME can be used to specifically set the location of various system files, including licensing and security information. If the NIAKWA\_RUNTIME environment variable was not set, previous versions of NPL would typically look in the current directory and ultimately in a root level \BASIC2C directory of the current drive for these system files.

As of this release, if NIAKWA\_RUNTIME is not set, before looking in the current directory or other locations, NPL will now check the directory containing the RunTime being executed first. In most circumstances, this eliminates the need for the NIAKWA\_RUNTIME environment variable.



In some instances, it may be desirable to set NIAKWA\_RUNTIME.

For example:

- When multiple directories contain different versions of the RunTime.
- If copies of the RunTime are stored locally rather than on a server to maximize load time.

There may be some sites where the NIAKWA\_RUNTIME environment variable is currently not set and which rely on the exact search behavior of the RunTime to locate certain system files. For example, some sites may expect files such as KEYBOARD.xxx and SCREEN.xxx to be located in the current directory or in the \BASIC2C directory. If this directory is not where the RunTime is located, and the RunTime directory also contains these files, the expected version of the file will not be used. In such cases, it is recommended that all conflicting files should be removed from the RunTime directory and the expected files placed in the current directory.

## 5.3 Compiler Enhancements

This section discusses extensions and updates which have been implemented in the NPL compiler programs since the introduction of NPL Release IV.

### 5.3.1 New 32-bit Windows Compilers <sup>[5.00]</sup>

NPL Release V introduces two new versions of the Niakwa compiler.

B2CWIN32.EXE is a 32-bit windows based version of Niakwa's compiler intended for interactive use. B2CCON32.EXE performs all of the same functions as B2CWIN32 but is intended for use in batch files.

#### **Statement Separators Replaced by Spaces in Source.**

If source code is extracted from pcode, statement separators (":") at the start of a source line are replaced by spaces.

### 5.3.2 General Compiler Enhancements

This section discusses general enhancements to both the DOS and Windows compilers.

**New -ERRFORMAT Option** [4.10.23]

B2C supports a new option "-ERRFORMAT" to control the format of error diagnostic lines. Supported values for this option include:

- NPL Default. Generates the same error output as previous releases.
- MS Generates error output conforming to Microsoft C compilers and other tools.

For example:

```
filename line-number column : Text of error message...
```

On the current implementation, -ERRFORMAT MS requires that the source files must be .SRC files (not diskimages) and is not allowed when using -LSTFORMAT 2200[S].

With the -ERRFORMAT MS option, B2C can be used as a syntax checker for those full-screen editors (such as M) that capture the output of a compiler and scan it for errors. By displaying the error information using this format of line, the editor can use the information preceding the ":" to locate the source file, display it, and position the cursor at the error.

For example, using the editor M.EXE, add the following line to tools.ini in the [m] section:

```
extmake:src B2C.EXE -errformat MS -objloc NUL -srcloc %|pF. %|fF
```

replacing B2C.EXE with the full path specification of the B2C program.

This allows a rapid syntax check of any .SRC file using the "argcompile" key (default F5). The program is compiled (output discarded) and the cursor is positioned to the first syntax error, with the diagnostic on the communication line. Additional errors can be positioned to by the "compile" key (default SHIFT-F3).

**NOTE:** The M editor referenced above is no longer available. For full screen editing and syntax checking capabilities, use the Niakwa Workbench.

**NOTE:** Only local syntax errors are detected with this technique (those that would be diagnosed when enter is pressed or if the program is entered line-by-line under RTI). Other non-local syntax (e.g., unmatched structured IF) and semantic errors (i.e., incorrect parameters to FUNCTION calls) will only be detected at run time.

**\$OBJECT Output Buffer Extended** [4.20]

The output of \$OBJECT is no longer restricted to a maximum of 1K. If the output buffer used in the assignment containing the \$OBJECT statement exceeds 1K (and does not overlap the source buffer), it will be used as a temporary buffer and will define the maximum size of the output of the \$OBJECT statement. For example:

```
DIM Obj$256,Src$512
Obj$=$OBJECT(Src$)      ;;limit is 1K as on previous
                        releases
```

whereas:

```
DIM Larger_Obj$32000,Src$512
Larger_Obj$=$OBJECT(Src$) ;;limit is size of Larger_Obj$
                        (32000 bytes)
```

### **B2C Extended to Permit Compiling of Boot Programs** [4.30/5.00]

The B2C compiler has been extended to permit compiling from source consisting of one or more boot programs. Specify the name of the boot program where the equivalent .SRC file name would go, including the extension.

**NOTE:**        **On previous releases any specified extension in the list of source file names generated the error:**

#### **Extensions not permitted**

As of this release, any file extension may be used for input files when SRCLOC is a directory. If the file is pcode (based on the label), the source will be decompiled as it would for a program in a diskimage. If the file is not pcode, it is compiled as ASCII text.

For example:

```
COPY TEST.SRC TEST.TXT
b2c TEST.TXT
```

=> produced TEST.OBJ

```
b2c -lstloc . -listformat .src -objloc nul TEST.OBJ
```

=> produces TEST.SRC from the TEST.OBJ boot program.

**WARNING:** Caution should be exercised when compiling from .OBJ files in the current directory. The default OBJLOC value (".") will produce a .OBJ file with the same name in the current directory, overwriting the source file. Redirect compiled code (OBJLOC) to the nul device or to a different directory or diskimage to avoid this problem.

**NOTE:** Because of the label checking mechanism, boot programs smaller than 256 bytes are not recognized as valid boot programs. Programs saved by RTI's SAVE BOOT program always are padded to 256 bytes or more. Programs compiled with earlier versions of B2C may produce a .OBJ program smaller than 256 bytes. All boot programs produced by this version of B2C are padded to at least 256 Bytes.

**Enhanced KEEPREMS ON parameter** [4.30/5.00]

If compiled with KEEPREMS ON, the compiler no longer ignores blank lines. They are treated as a comment line and count as a statement (just as a comment would).

For example ("<" indicates a return graphic):

```

:0010 A=1<
      <
      B=2<
      <
      C=3

:LIST
:0010 A=1

      B=2

      C=3

```

**NOTE:** The pcode generated for a blank line is downward compatible to releases that support a line comment (";"). On those releases, the ";" indicator will be displayed instead of a blank line. So for the above example, previous releases would LIST as:

```

:0010 A=1
      ;
      B=2
      ;
      C=3

```

A null statement immediately before a return graphic also becomes legal, and equivalent to a line comment. So if you prefer the \$OPTIONS setting that retains the colon at the start of multi-statement lines, this is permitted also.

For example:

```

:0010 A=1<
      <
      B=2 :<
      :<
      : C=3

:LIST
:0010  A=1
      :
      : B=2 :
      :
      : C=3

```

**NOTE:** A line number with no text following it still means "delete this line" in command mode. B2C will compile a line with no text, but recalling the same line in RTI and pressing RETURN will delete the line.

#### Update to 2200S Option [4.30/5.00]

2200S atomized source code generated on previous releases of B2C (using the -LSTFORMAT 2200S option), may have produced syntax errors or compiled incorrectly if extracted into .SRC files and recompiled using B2C. The current version will leave spaces in the 2200S output where these are required syntactically.

For example:

```

0010 DELETE T A$
      : PRINT SCREEN V$
      : DIM /PUBLIC K
      : END RECORD X

```

Old B2C versions would store the above code in 2200S format as:

```

0010 DELETETA$
      : PRINT SCREENV$
      : DIM /PUBLICK
      : END RECORDX

```

As a result, the code does not recompile correctly, since syntactically significant spaces are missing.

Current versions of B2C now store 2200S format as:

```
0010 DELETE T A$  
      : PRINT SCREEN V$  
      : DIM /PUBLIC K  
      : END RECORD X
```

which will recompile correctly.

### 5.3.3 B2CWIN32

B2CWIN32 is a variant compiler used by the Niakwa Workbench. For batch compile operations use B2CCON32, a 32-bit compiler that is upward compatible with older B2C versions.

## 5.4 RunTime Error Codes

This section discusses extensions to the NPL Release IV Error and Warning messages.

### 5.4.1 Error code Updates

The following error codes have been added since the introduction of NPL Release IV. Refer to Appendix B of the NPL Release IV Programmer's Guide for a complete description of all NPL error codes.

- ERR 264 - Statement would do forced RETURN ERROR to external, which is disabled.
- ERR 265 - Statement would do forced RETURN ERROR to RUN, which is disabled.
- ERR 266 - Automatic /EXIT procedure did not complete
- ERR 267 - Statement would do forced RETURN ERROR in /EXIT procedure, which is disabled.
- ERR 268 - Immediate INCLUDE is not legal. Current module is not resolved.
- ERR 305 - Cannot locate \$DECLARE function.
- ERR 306 - \$USER opcode out of range.
- ERR 307 - \$USER operation failed.

### 5.4.2 Warning Messages Updates

The following RunTime Warning Messages have been added since the introduction of NPL Release IV. Refer to Appendix B of the NPL Release IV Programmer's Guide for a complete description of all NPL warnings codes.

- WARN 04 - STOP during InSendMessage. Automatic execute option used.
- WARN 05 - Halted during InSendMessage. ReplyMessage(0) issued.

This page intentionally left blank





## CHAPTER 6

# PLATFORM SPECIFIC UPDATES

### 6.1 Overview

This chapter discusses updates and enhancements which are specific to one or more NPL platforms.

Section 6.2 discusses general operational changes and enhancements that apply to all MS-Windows RunTime.

Section 6.3 discusses general updates and enhancements specific to Windows NT.

Section 6.4 discusses general updates and enhancements specific to Windows NT.

Section 6.5 discusses general updates and enhancements in NPL's network support logic.

Section 6.6 discusses general updates and enhancements specific to MS-DOS RunTimes.

## 6.2 General MS-Windows Considerations

This sections discusses general operational changes and enhancements which are applicable to all NPL MS-Windows RunTimes.

### 6.2.1 New MS-Windows Setup Options

Several new Windows setup options have been implemented since the introduction of the NPL MS-Windows Setup routine. This section covers all new setup options available in SETUP.INI.

#### **Remote Install Warning Option** <sup>[4.22]</sup>

This NetBIOS option displays a warning to the user that installation to a MS-Windows NT Server must be performed from the server and not from the remote workstation.

Option defaults are:

WarnIfRemote=ON	for NetBIOS-only RunTimes
WarnIfRemote=OFF	for Novell enabled RunTimes

This option is only checked in response to a network install.

#### **NIAKNETW Install Option** <sup>[4.22]</sup>

This option simplifies the process of configuring workstations to operate in a NetBIOS environment.

Option defaults are:

InstallNiaknetw=ON	for NetBIOS-only RunTimes
InstallNiaknetw=OFF	for Novell enabled RunTimes

If this option is ON at setup time, and there is no NiaknetwEnabled option configured in the [GENERAL] section of RTIWIN.INI, the following entry is added to the [GENERAL] section:

```
[GENERAL]
NiaknetwEnabled=1
```

**NOTE:** Current revisions of all MS-Windows RunTimes have been updated to automatically launch NIAKNETW.DLL when it is required. This process only occurs when no Novell Network is detected and the RunTime is not installed locally.

### 6.2.2 RTIWIN.INI Updates

This section discusses RTIWIN.INI parameter and section updates introduced since the rollout of NPL Release IV. Refer to Section 3.4 of the NPL MS-Windows Addendum for a complete discussion of all RTIWIN.INI parameters.

#### Establishing Unique 16-bit and 32-bit Sections [4.30/5.00]

NPL now supports optional 16-bit or 32-bit sections in the RTIWIN.INI file for values specific to the Windows version of the RunTime in use. The name of the platform specific section is derived from the non-platform specific key followed by ',16' or ',32'.

For example:

```
[general]
# options apply to all applications
```

```
[general,16]
# options apply to all applications
# when running rtiwin.exe
ExternalLibrary1=C:\NPL\NDMBTRV.DLL
```

```
[general,32]
# options apply to all applications
# when running rtiwin32.exe
ExternalLibrary1=C:\NPL\NDMBTR32.DLL
```

```
[C:\NPL\UTILITY.OBJ]
# options apply if boot program is C:\NPL\UTILITY.OBJ
```

```
[C:\NPL\UTILITY.OBJ,16]
# options apply if boot program is C:\NPL\UTILITY.OBJ
# when running rtiwin.exe
```

```
[C:\NPL\UTILITY.OBJ,32]
# options apply if boot program is C:\NPL\UTILITY.OBJ
# when running rtiwin32.exe
```

Similar sections may be specified in the network.ini file.

The most common use of these sections will be where external libraries are specified using the ExternalLibraryx options, and the formats of DLL's used for 16-bit and 32-bit applications are incompatible.

**NOTE:**        **If a 32-bit version of a DLL is to be implemented, in addition to adding the name to a [,32] section, any previous 16-bit DLL's should be moved to a [,16] section, in order to avoid possible conflicts of 32-bit applications attempting to load 16-bit DLL's. This allows both 16 and 32-bit versions of the NPL Windows RunTime to co-exist on the same network. When only a 16-bit or 32-bit application is installed, the [,16] and [,32] sections are not required.**

Window positioning information that is maintained by NPL is updated in the non-platform specific section of the RTIWIN.INI file.

#### **New MS-Windows Debug Parameters** <sup>[4.10]</sup>

As of Revision 4.10 the following three parameters have been added to the list of parameters available for the RTIWIN.INI file. The three new parameters are for use with the Debug Windows option of the interpretive version (RTIWIN.EXE and RTIWIN32.EXE) of the NPL MS-Windows RunTime.

#### **DebugAutoSize**

Purpose:        For the Debug Window, allows specification of the status of the autosize flag (on or off) and the font sizes to be used if off. In addition, the maximum width of the screen used by the application is specified (usually 80 columns, but may also be 132 for applications that select a wide display).

The DebugAutoSize option and font size may be changed by the user by resizing the windows. The values are saved in the application's option section when the window is closed as defaults for the next session.

Available values are:

1. 0 for off or non-zero for on
2. Font width in pixels when debugging autosizing is off
3. Font height in pixels when debugging autosizing is off
4. Application display size in columns (usually 80 or 132)

Value: Four numeric values separated by blanks.

Default: 1 0 0 80

Example: DebugAutoSize= 0 7 12 80

### **HardMode**

Purpose: Specify whether the Hard Mode option (from the Debug Window's Options menu) is active. Refer to section 6.2.3 below for a complete discussion of this parameter.

Value: Numeric, 0 = not active, 1 = active

Default: 0

Example: HardMode=0

### **DebugWindow**

Purpose: A display preference option that specifies where and how large the Debug Window should appear. Five numbers are required, specifying the x and y coordinates (in pixels) from the top left of the screen, the x and y size of the screen (in pixels) and a fifth number showing whether the window is initially visible.

**NOTE:** If the number of pixels is out of range for the display being used (i.e., set to 790 when the monitor can only display 748 pixels horizontally), the Debug Window is automatically moved to make it at least partially visible.

The location, size and visibility of the Debug Window may be changed by the user. Current values are saved in the program's RTIWIN.INI option section when the window is closed as the defaults for the next session.

Available values are:

1. Specifies x pixels from the top left of the screen
2. Specifies y pixels from the top left of the screen
3. Specifies x horizontal size in pixels from 1
4. Specifies y vertical size in pixels from 2
5. 1, Debug Window is visible, 0 it is not visible.

Default: Defaults provided by system

Example: DebugWindow= 1 3 640 271 0

**New Mixed Network Startup Options** [4.30/5.00]

Two new options are available to turn off warning dialog boxes which may be generated when operating in a mixed network environment. Refer to Section 3.5 for a complete discussion of new mixed network considerations.

**WarnNovellNoLogin**

If a workstation is attached to a Novell server but not logged in, the NPL RunTime is unable to authenticate with the Novell server. If this condition is detected, a warning is displayed. To disable this warning, set the "WarnNovellNoLogin" option to 0. For example:

```
[ GENERAL ]  
WarnNovellNoLogin=0
```

**NOTE:** If this option is not set, the default value for this option is 1 (ENABLED).

**WarnNovellDriver**

Some Novell client software (for example, Microsoft "Client Service for Novell") provides subsets of the Novell client capability which are inadequate for the NPL RunTime to operate. If this condition is detected, a warning is displayed. To disable this warning, set the "WarnNovellDriver" option to 0. For example:

```
[ GENERAL ]  
WarnNovellDriver=0
```

**NOTE:** If this option is not set, the default value for this option is 1 (ENABLED.)

**New True Type Font Parameters** [4.30/5.00]

The FontCharSet and FontFaceName options have been updated to support the following new parameters:

```
FontCharSet=2  
FontFaceName=Niakwa Console
```

When the above parameters are set, NPL will attempt to use the new NPLFONT.TTF true type font.

**NOTE:** This font is included on all current NPL revisions and is installed as part of the NPL MS-Windows RunTime Setup routine. Resolution of this font will vary widely depending on the screen resolution and video controller capabilities.

### 6.2.3 MS-Windows Debug Mode <sup>[4.10]</sup>

The MS-Windows RunTime (RTIWIN.EXE and RTIWIN32.EXE) supports a dual window debug mode which may be selected from the RunTime Options Menu. When Debug is active, all immediate mode operations update a separate text-only window (24 lines of 80 characters).

**HINT: The Debug Window can be located and sized separately from the application display.**

The Debug Window permits STEPIing through programs which update the application display window without adversely affecting the contents of this window. Specifically, the following output will always appear in the Debug Window when this option is selected:

- Output from LIST and TRACE statements directed to the screen
- Output from STOP and END statements
- The Immediate mode command prompt and command line entries
- Unrecoverable error statement displays
- Output from PRINT statements executed from Immediate Mode.

In addition, the Debug Window and the application have separate windows event queues. This means that the keyboard and mouse input for the two windows is effectively independent (i.e., the window with the input focus will receive keys entered at that time).

Whenever an active program HALTs or executes a STOP or END statement, the Debug Window becomes the active window. Immediate mode commands must be entered into this window and output from LIST or PRINT statements, which would normally appear on the application window, is displayed on the Debug Window instead. TRACE output normally directed to the screen is always directed to the Debug Window.

**NOTE: INPUT SCREEN statements executed by programs (not in Immediate Mode) always get information from the application window, not the Debug Window.**

**NOTE: The Debug Window option is convenient for debugging many conventional NPL applications, especially when debugging routines update the application window.**

The Debug Window option is essential for debugging NPL applications which are event-driven. As noted above, the Debug Window uses a separate application message queue aside from the



NPL application. This allows the programmer to STOP within message handling routines, without disrupting the sequence of events in the application. The Debug Window remains visible even if the NPL application window has been hidden.

When using 16-bit RTIWIn, the Debug Window options requires that the RTISLAVE.EXE module be located in the same directory as RTIWIn. This program is not required for the 32 bit version.

**NOTE:** Byte 47 of \$OPTIONS has been updated to allow for disabling the "Debug" option from the applications options menu. Refer to Chapter 8 for details.

**Options of the Debug Window.**

The following options are available from the Debug Window.

HALT                    Equivalent to pressing CTRL-BREAK. This option tells NPL to halt execution at the first opportunity-usually the start of the next statement.

STEP                    Available while HALTEd. This command does the equivalent of a STEP# command, if STEP mode is not yet active. It then does the equivalent of an EXECUTE (single step) key.

**NOTE:** Any partially entered command is discarded.

CONTINUE                Available while HALTEd. This does the equivalent of a CONTINUE command.

**NOTE:** Any partially entered command is discarded.

**Hard Mode**

When this option is selected (from the Debug Window's Options menu), any time RTIWIN waits for a key in the Debug window, all other activity in MS-Windows is effectively halted.

Hard Mode is ignored by the 32-bit version.

**NOTE:**        **This is not usually required for conventional NPL applications.**

For NPL applications which are event-driven, if the 'DisablePoll' function has been called to disable implied polling of the message queue, Hard Mode takes effect in the Debug Window, even if it is not selected from the menu.

While an application is halted in Hard mode, messages cannot be sent to other applications. This means that if a Debug Window is moved or resized, exposed areas of the screen may not be repainted until execution is resumed and the message queue is read. In addition, if Hard mode is active and the mouse cursor is moved outside the Debug Window, the mouse pointer's display form remains "stuck" in the shape typically used to indicate a resize operation.

**6.2.4 New /C (Control) Command Line Option** <sup>[4.22]</sup>

This release introduces the /C (control) command line option for the NPL Windows RunTime (RTIWIN and RTPWIN.)

The /C startup flag is implemented for support of "Instant Vinny" applications. The intended use of this option is to simplify the process of pasting an NPL application window as a control onto a form defined elsewhere. The flag has several effects, including:

- The NPL application window no longer displays a caption, menu bar border or scroll bars.
- The NPL application window starts hidden, until instructed to show itself.
- The default colors used for button background and text are determined as follows:

If the RTIWIN.INI settings for "StandardRGBColor0" and "StandardRGBColor7" are assigned values, these color values are used for NPL text where the program specifies black and white color, respectively. In the absence of RTIWIN.INI settings, the default NPL

Window colors of black and white are replaced by the current desktop color values for button background and text respectively.

**NOTE:**        **The automatic font sizing option is enabled, regardless of any value in RTIWIN.INI.**

A registered message "NPL\_IV\_SETHOST" may be used by a host program (usually an Instant Vinny application) to instruct RTIWIN to position the application window so that it is located permanently over a window belonging to the host program. The parameters to this message are:

IParam=0 :    wParam is the handle of the host window.  
IParam=1 :    wParam ignored: synchronize position.  
Other Values : Reserved

Normally, if the keyboard focus switches to any other window belonging to the host application (by use of the mouse), NPL resets keyboard focus back to NPL application window.

**NOTE:**        **It is unlikely that these options will be of use to applications except Instant Vinny. Niakwa reserves the right to change or extend the effect of this option in future releases.**

#### 6.2.5    \$DECLARE Statement Implemented <sup>[4.20]</sup>

A new statement, "\$DECLARE", provides developers direct access to external routines in MS-Windows DLL files. Refer to Chapter 8 for a complete description of the "\$DECLARE" statement.

**NOTE:**        **While the \$DECLARE statement gives developers access to the MS-Windows API at a higher level than the Niakwa Gateway to Windows API library product, a formal understanding of "C" and the Windows API calls is still required. Developers interested in a simplified interface to the Windows API should explore the capabilities of Niakwa's Visual Interface Manager (Vinny) product.**

#### 6.2.6    New MS-Windows Demo RunTime <sup>[4.30/5.00]</sup>

New NPL MS-Windows Demonstrator RunTimes are available. Each demo disk includes non-interpretive versions of the 16-bit and 32-bit MS-Windows RunTimes. Demo packs ship 10 disks to a pack with each disk licensed for either 10 or 40 uses.

### 6.2.7 New MS-Windows 2227 Communications Support <sup>[4.10.23]</sup>

A new driver, "WIN2227.DLL" has been implemented to provide 2227 Asynchronous Communications support to the MS-Windows RunTime. The WIN2227.DLL driver relies upon the Windows communication drivers directly, resulting in improved performance and reliability over the past MS-DOS version of this driver.

This WIN2227 driver is invoked using the new \$DEVICE clause "MXE=Y". Refer to section 6.3.7 for a complete discussion on invoking this driver.

**NOTE:**        **The NPL MS-Windows 32-bit RunTime provides embedded support for access to "MXE=Y" devices (no WIN2227.DLL required.) For NPL applications requiring this support on the NPL MS-Windows 16-bit RunTime, the WIN2227.DLL driver is available as part of the Niakwa Scientific and Communications Driver product.**

### 6.2.8 Default Printer Device Name Change <sup>[4.20]</sup>

The decorated \$DEVICE names for printers (/dev/prn or /dev/lpt1 for LPT1) are no longer recognized as devices under Windows NT. In addition, these device names are becoming intermittently unrecognized under Windows environments from one system to the next. To address this, the DOS, Windows and Pharlap 386/DOS-Extender versions of NPL have been revised to use "LPT1" as defaults in the device table to avoid this problem. As an alternative, UNC notation may be used for access to shared printers or network print queues.

### 6.2.9 \$OPEN Considerations for Print Class Devices <sup>[4.10]</sup>

\$OPEN directed to print devices in MS-DOS based operating environments is only useful under MS-Windows when multiple tasks on the same PC are attempting to print to the same device concurrently.

**NOTE:** This refers only to the use of actual print devices such as LPT1 or /dev/prn.

Device locking exhibits a number of peculiarities under 16-bit Windows or using 16-bit RTIWIN. The following section's notes apply only to 16-bit environments:

\$OPEN is not required under any of the following conditions:

- Not operating under MS-Windows
- Operating under MS-Windows, but directing output to the Windows Print Manager
- Operating under MS-Windows, but directing output to different device names (i.e., LPT3 versus LPT1 for example)
- Operating under MS-Windows, but never printing from more than one task at a time

The use of SHARE calls performed by \$OPEN when directed to print devices is not well supported on many newer environments (e.g., Technical Note #68 on Novell VLM drivers). Revision 4.10 has been modified to avoid the problems with SHARE calls to print devices by only issuing SHARE calls to printers that are reported by DOS as "local". Printers reported as "remote" are not locked.

Spooled printers are reported as "local" under Novell NetWare with NETX drivers loaded. Therefore spooled printers are locked by \$OPEN.

Novell NetWare with VLM drivers and all NetBIOS networks report spooled printers as "remote" and therefore no locking is performed. In these environments, it is possible to get intermingled output if printing is conducted from multiple tasks under MS-Windows to the same device.

### 6.2.10 Landscape and Portrait printing under MS-Windows <sup>[4.20]</sup>

Output directed to the MS-Windows Print Manager under the windows RunTime may now request landscape or portrait orientation using \$DEVICE. To obtain landscape mode, replace the document name field (if any) with the keyword LANDSCAPE followed by a command and the document name (if any). Use the keyword PORTRAIT to request portrait mode.

For example, to print to the default \$DEVICE() in landscape mode use:

```
$DEVICE ( / 2xx ) = " > ( ) LANDSCAPE "
```

If the CFG=Y clause is also used, the selected landscape/portrait orientation is the default in the printer configuration dialog, but may be overridden by the operator.

### 6.2.11 New \$DEVICE Clause "XPA=Y" <sup>[4.21]</sup>

NPL now permits the MS-Windows RunTime transparent access to MS-Windows printer drivers that support the "Pass Through" printer escape controls, if the XPA=Y clause is added to the device specification. Refer to Chapter 7 for a complete discussion on passthrough printing.

## 6.3 Windows NT Considerations

This section discusses general updates and enhancements which are specific to Windows NT.

### 6.3.1 Windows NT Workstations on a Novell Server <sup>[4.20]</sup>

By default, Windows NT workstations connected to a Novell NetWare server are configured with the Microsoft client software for NT Workstation when Windows NT is installed.

To date, the version of the Microsoft client software for NT Workstations (Client Service for NetWare) or NT Server (Gateway to NetWare) does not support the 16-bit functions required by the Windows RunTime.

To work around this problem, these sites must be configured with Novell's client software (NetWare Client32 for Windows NT or Novell Intranetware Client for Windows NT) to operate the Windows RunTime in this environment correctly.

When Intranetware Client is installed, the standard DOS and 386/DOS Extender RunTimes may be run in a DOS box.

The NetWare Client for Windows NT software is an older product that provides a "DOS support" box in which the standard DOS and 386/DOS Extender RunTime may be run if necessary.

The "DOS support" box adds IPX and NETX support layers to the default NT command prompt. During this process it loses any globally mapped network drives, so you must log in again to access any server drives.

**NOTE: Mapped drives in the DOS support box are isolated from any changes made to the globally mapped drives.**

This software is available from Novell, and may be downloaded either from Novell's internet site or Netware.site on Compuserve.

### 6.3.2 Using Windows NT Clients with Windows NT Server <sup>[4.20]</sup>

If the Windows NT system you are using is not connected to a Novell server, the system will operate as a peer network using NIAKNETW.DLL. Make sure that the [general] section in RTIWIN.INI has NiaknetwEnabled=1.

If neither of the above is loaded, the fingerprint must be installed on a local drive.

**NOTE:**           **In any instance, setting the NIAKWA\_RUNTIME environment variable must be done through the Control Panel/ System application.**

### 6.3.3 Using DOS NetBIOS Sessions Under Windows NT <sup>[4.20]</sup>

In DOS sessions under Windows NT, NIAKNETB may be run to connect on a peer-to-peer network, but either one DOS box or a number of Windows sessions may be active at any time (not both). Closing the NIAKNETB DOS box is required to allow running Windows sessions.

After running the Windows RunTime with NIAKNETW enabled, logging off is required before running a NIAKNETB DOS box.

**NOTE:**           **Niakwa strongly recommends using the NPL Windows RunTime under Windows NT instead of RTI in a DOS session under Windows NT.**

### 6.3.4 Diskette Access under Windows NT <sup>[4.20]</sup>

Raw diskettes under Windows NT appear to have some incompatibilities, particularly if media types are changed in drives that support both low and high density media. This problem is currently under review.

### 6.3.5 Printer Device Names <sup>[4.20]</sup>

The decorated \$DEVICE names for printers (/dev/prn or /dev/lpt1 for LPT1) no longer are recognized as devices under Windows NT. In addition, these device names are becoming intermittently unrecognized under Windows environments from one system to the next. To address this, the DOS, Windows and Pharlap 386/DOS Extender versions of NPL have been revised to use "LPT1" as defaults in the device table to avoid this problem.



### 6.3.6 WIN2227 Considerations <sup>[4.20]</sup>

For the NPL WIN2227 communications driver to work under Windows NT, the name of the COM port must be specified in \$DEVICE without extra decorations (i.e., as COM1 or COM2, not /dev/com1 or /dev/com2). Otherwise, attempts to use MXE=Y type devices in the Windows RunTime will hang all 16-bit applications.

**NOTE:**        **The number of the COM port may be different from DOS if some COM ports are not configured on the system. For example, if COM1 is not configured, what was DOS COM2 becomes NT COM1.**

## 6.4 Windows 95 Considerations

### 6.4.1 Diskette Access Under Windows 95 <sup>[4.20]</sup>

Due to an apparent problem in the Windows 95 support for direct access to non-DOS formatted media, you may be unable to access 720K diskettes using \$DEVICE()="x: 720=Y" using either the Windows, DOS or 386/DOS Extender versions when running Windows 95.

To verify media types in drives that support both low density and high density media, the media check routine has been modified to read the last sector of a newly configured diskette (not previously accessed since \$DEVICE assigned or \$CLOSE operation), and for low-density media, to attempt to read the sector after this (expecting to fail) to ensure that high-density media is not mounted.

The media check can result in a slow first diskette access, and for some media combinations (e.g., accessing 360KB media using \$DEVICE()="x: 1.2=Y") some drive noise may result.

### 6.4.2 Windows 95 Logout Considerations <sup>[4.20]</sup>

On Windows 95 workstations in a NetBIOS environment, when attempting to logoff the current session, you may encounter a message indicating that NIAKNETW.DLL is still loaded and Windows 95 then proceeds to restart. To avoid this condition, the developer should insure that the NIAKNETW.DLL file is located in the Windows system directory and deleted from the current RunTime directory.

The standard setup procedure installs NIAKNETW.DLL in the Windows system directory.

## 6.5 General Network Considerations

This section discusses general updates and enhancements in NPL's network support logic.

### 6.5.1 Improved Mixed-Network Authentication <sup>[4.30/5.00]</sup>

NPL Revision 4.30 implements improved security authentication in mixed network environments. On previous releases, if the NPL RunTime was installed on a Windows NT server, the NPL security check would fail in the following circumstances:

If the workstation running NPL was connected to a Novell server, but is not logged into the server, authentication would fail, (typically displaying "Number of Authorized Users Exceeded") and refuse to run.

Microsoft Windows NT Workstations with client software other than Novell's Client 32, (for example, the Microsoft "Client Service for Novell" which ships with Windows NT Workstation) would fail upon detection of a Novell server and refuse to run.

As of this release, NPL no longer fails in both of the above situations, but behaves as if the Novell server is not attached. In either case, the user is presented with a warning box indicating the problem. By clicking OK, NPL automatically attempts to authenticate with the NIAKNETW user counting mechanism if available.

Refer to Section 6.2.2 above for a description of two new RTIWIN.INI options that may be set in conjunction with this new startup functionality.

### 6.5.2 New Mixed Network Support <sup>[4.21]</sup>

On networks that contain both Novell NetWare and non-NetWare file servers, Revision 4.21 of NPL attempts to determine whether each file is on a NetWare server so that the (more efficient) NetWare file locks are only directed to files located on NetWare servers. Files on other servers will be locked using the more transportable (but less efficient) SHARE locks. The 32-bit version of RTI never uses NetWare file locks.

**NOTE:** This new functionality has been added to NPL as a convenience to those who have encountered mixed network environments and wished to access files on both types of servers. In general, Niakwa recommends that developers isolate their data files on a single server and avoid mixed server access to avoid unnecessary performance degradation.

**If mixed file locking cannot be avoided, developers are highly encouraged to implement the atomic file locking capabilities of NPL using Byte 39 of \$OPTIONS to minimize performance degradation. Refer to Section 7.7 for a complete discussion of \$OPTIONS Byte 39.**

Developers may also take advantage of Novell-style Universal Naming Convention (i.e., server\volume:path) and the newer Microsoft-style UNC (\\server\share\path) network file names in environments that support these conventions. Developers are cautioned to be aware that all operating environments do not necessarily report correctly that a file is on a NetWare server. Refer to the following section for a discussion on known issues under MS-Windows NT and mixed network handling.

### 6.5.3 \$DEVICE UNC Notation <sup>[4.20]</sup>

Niakwa has done preliminary testing with the "Universal Naming Convention" or UNC for the \$DEVICE specification. Our testing has been successful for referencing diskimages, boot programs, and network printers with UNC notation. For example:

```
$DEVICE(/D19)="\\hal\sys\project\new\boot.obj"  
$DEVICE(/D20)="\\hal\sys\basic2c\utility.bs2".  
$DEVICE(/215)="\\hal\printq0"
```

**HINT:** Anyone that uses network printing may find UNC notation beneficial because it eliminates the need for the traditional DOS Capture and helps those who are constrained by the DOS limit of only 3 LPT ports. Furthermore, mapped drive letters are no longer needed.

**NOTE:** Although Niakwa has found UNC notation to work in many instances, Niakwa has not exhaustively tested UNC notation in every networking environment. Be aware that support for UNC depends on the level of client software that is in use (i.e., NETX). Niakwa recommends that the developer thoroughly test any application which uses UNC notation prior to releasing it into the field.

## 6.6 General MS-DOS Considerations

This section discusses general updates and enhancements specific to running NPL MS-DOS RunTimes.

### 6.6.1 Enhanced PC Keyboard Support <sup>[4.20]</sup>

Entries have been added to the built-in IBM keyboard translation tables to support new keyboard codes which are available as of DOS 5.0. Specifically, the keys affected are:

PC Keyboard	NPL Virtual Keyboard
CTRL+INSERT (complex 92)	(-SHIFT-INSERT-) (SF '5A)
CTRL+DELETE (complex 93)	(-SHIFT-DELETE-) (SF '59)
CTRL+UP (complex 8D)	(-SHIFT-NORTH-) (SF '56)
CTRL+DOWN (complex 91)	(-SHIFT-SOUTH-) (SF '55)
F11 (complex 85)	('10) (SF '0A)
F12 (complex 86)	('11) (SF '0B)
SHIFT+F11 (complex 87)	('26) (SF '1A)
SHIFT+F12 (complex 88)	('27) (SF '1B)

**NOTE:** Previous versions of DOS did not report any keys for these combinations.



## CHAPTER 7

# PROGRAMMER'S GUIDE UPDATES

### 7.1 Overview

Chapter 7 describes enhancements and improvements to the Niakwa Programming Language environment that would otherwise be covered in the NPL Release IV Programmer's Guide. Specific changes to the language are covered in Chapter 8.

Section 7.2 covers enhancements to device support for NPL with respect to disk-class devices (diskimages) and print-class devices (printers).

Sections 7.3 through 7.14 discuss specific non-syntax updates and additions to the programming language since the introduction of Release IV. Chapter 8 covers language syntax enhancements and additions.

## 7.2 Device Support

This section covers enhancements to device support within NPL. Covered are improvements to diskimage devices and special printing considerations.

### 7.2.1 Landscape and Portrait printing under MS-Windows <sup>[4.20]</sup>

Output directed to the MS-Windows Print Manager under the windows RunTime may now request landscape or portrait orientation using \$DEVICE. To obtain landscape mode, replace the document name field (if any) with the keyword LANDSCAPE followed by a command and the document name (if any). Use the keyword PORTRAIT to request portrait mode.

For example, to print to the default print device in landscape mode use:

```
$DEVICE ( / 2xx ) = "> ( ) LANDSCAPE"
```

If the CFG=Y clause is also used, the selected landscape/portrait orientation is the default in the printer configuration dialog, but may be overridden by the operator.

### 7.2.2 New \$DEVICE Clause "XPA=Y" <sup>[4.21]</sup>

Revision 4.21 of NPL now permits the MS-Windows RunTime transparent access to MS-Windows printer drivers that support the "Pass Through" printer escape controls, if the XPA=Y clause is added to the device specification.

### 7.2.3 Enhanced Sector Address Checking <sup>[4.22]</sup>

Additional checking is done for sector addresses out of range when accessing diskimages without the EXT=Y option. On previous releases sector numbers larger than 65535 but less than 16777216 could be used on direct access disk statements even if the disk image was not the EXT type. For read accesses, this would result in an error but for write access, the diskimage size could be extended past 16MB (depending on the platform and on available disk space). Checking is done on the start sector addresses for:

DATA LOAD/SAVE BA/BM/BU/DA

COPY

and for the input and output range end addresses for:

COPY, MOVE

### 7.2.4 Automatic Diskimage Extension <sup>[4.30]</sup>

The B2C program has been updated to automatically extend the END=value of a diskimage file if the compiled programs exceed the current capacity.

**NOTE:**           **This functionality is not available on all platforms. Specifically SuperDOS and VMS do not support it.**

### 7.2.5 Universal Naming Convention (UNC) Support <sup>[4.20]</sup>

Niakwa has done preliminary testing with the "Universal Naming Convention" or UNC for the \$DEVICE specification. Our testing has been successful for referencing diskimages, boot programs, and network printers with UNC notation. For example:

```
$DEVICE(/D19)="\\hal\sys\project\new\boot.obj"  
$DEVICE(/D20)="\\hal\sys\basic2c\utility.bs2".  
$DEVICE(/215)="\\hal\printq0"
```

**HINT:** Anyone that uses network printing may find UNC notation beneficial because it eliminates the need for the traditional DOS Capture and helps those who are constrained by the DOS limit of only 3 LPT ports. Furthermore, mapped drive letters are no longer needed.

**NOTE:** Although Niakwa has found UNC notation to work in many instances, Niakwa has not exhaustively tested UNC notation in every networking environment. Be aware that support for UNC depends on the level of client software that is in use (i.e., NETX). Niakwa recommends that the developer thoroughly test any application which uses UNC notation prior to releasing it into the field.

### 7.2.7 Extended Printer Port Support <sup>[4.10.23]</sup>

Support has been added for logical parallel port addresses up to LPT9. This extension has been implemented to allow developers to take full advantage of Novell's extended CAPTURE command in Netware 4. To take advantage of this extension, workstations must be using Novell's VLM or Client 32 drivers. By default, the drivers support only three ports. The available ports can be increased by setting the "Network Printers" parameter to a value of 9. For VLMs this is handled in the NET.CFG file. For Client 32, it is in the Network Properties within Control Panel.



In addition, workstations running under MS-Windows 3.x and using the Windows Print Manager require a modification to the [ports] section of the WIN.INI file to enable Windows to address the additional logical LPTx ports. For example:

```
[ PORTS ]
LPT1 :
LPT2 :
LPT3 :
LPT4 :
.
.      ( Etc. )
```

**NOTE:** Add a LPTx: line for each new captured printer port to be addressed.

Once extended printer support has been enabled, NPL will recognize logical parallel device names up to LPT9.

### 7.2.6 Enhanced Printer Control <sup>[4.30]</sup>

For non-transparent Windows output redirected to the print manager, (i.e., \$DEVICE=">(…)") previous versions removed all control codes (less than HEX(20)) discarding all except carriage return HEX(0D), linefeed HEX(0A) and form feed HEX(0C). The default font for the device was used. On this release, the following control sequences are also interpreted by all MS-Windows RunTime platforms for this type of print device and can affect the underline and bold styles of the printed font.

```
HEX(020400000E) - text following this is not bold, not underlined.
HEX(020402000E) - text following this is bold, not underlined.
HEX(020400040E) - text following this is not bold, is underlined.
HEX(020402040E) - text following this is bold, and underlined.
```

The following shorter sequences may also be used:

```
HEX(0E)          - text following this is bold, not underlined.
HEX(0F)          - text following this is not bold, not underlined.
```

All other sequences of the form:

```
HEX(02...0E) and HEX(02...0F)
```

are reserved for future use, and are deleted from the output.

These sequences are similar to the control sequences accepted by the screen for bright and underline video modes, but do not exactly follow the conventions. In particular, the changed modes are not turned off by a HEX(0F), and HEX(0E) always turns on bold (not the last video mode).

The control sequence should always be PRINTed as a single string. A control code printed in pieces, such as:

```
PRINT HEX(0204);  
IF PrintBold=1  
    PRINT HEX(0200);  
ELSE  
    PRINT HEX(0000);  
END IF  
PRINT HEX(0E);
```

may fail to operate as expected.

**NOTE:** Caution should be exercised when using these control codes, since the exact results will depend on the current Windows printer driver. In particular, the following are known potential problems:

**On some printers, the default font is already bold. Turning bold off on these printers will give you either no change, or a lighter font than the default.**

**On some printers, the bold version of the font has wider characters than the non-bold version. Consequently, columns may not line up when mixing bold and non-bold characters .**

### 7.2.7 Passthrough Printing (XPA=Y) <sup>[4.21]</sup>

As of this release, when the Windows version prints to XPA=Y type printers it will use the printer driver RAW.DRV by preference. NPL Revision 4.30 installs this driver to the windows/system directory during setup.

The RAW.DRV is known to support the PASSTHROUGH escape required to do XPA=Y printing. In addition, a workaround in the Windows 95 environment requires sending a NEWFRAME escape to the driver, otherwise PASSTHROUGH output is discarded. When using the RAW driver, a NEWFRAME escape does not issue any data to the configured port.

### 7.2.8 Read Only Status (RDO=Y) <sup>[4.20]</sup>

A new clause to the \$DEVICE specification allows accessing printer class files which are read-only using the \$GIO read microcommands. Normally, any attempt to access a printer class file requires write access to the file, and if the file does not exist an attempt is made to create the file automatically. On previous releases, if the file was tagged as read-only, a device not available error occurred when attempting to access the file.

The clause 'RDO=Y' added to a \$DEVICE specification indicates that the program will only read from the file (using \$GIO) and so write access is not needed. In this case, the device is opened by NPL in read only mode and will not be created (if it does not exist, an error occurs). For example:

```
0010 $DEVICE(/01C)="G:\READONLY.DAT RDO=Y"  
      DIM L$10,Length,Buffer$512  
      $GIO/01C(HEX(8700 C620),L$)Buffer$  
      Length=VAL(STR(Buffer$,9),2)
```

**NOTE:** If the file G:\READONLY.DAT is tagged as a read-only file, the \$GIO statement would fail on previous releases, but will successfully work with this release.

### 7.2.9 New Device Clause Option (MXE=Y) for 2227 Support <sup>[4.10.23]</sup>

The NPL Windows RunTime now supports a new \$DEVICE option, "MXE=Y", for support of Niakwa's new 2227 asynchronous communications driver under MS-Windows. This option is supported on devices supported by the MS-Windows COMM.DRV driver (i.e., COM1, COM2, COMx) if the auxiliary dynamic link library WIN2227.DLL is available on the system.

**NOTE:**        **The WIN2227.DLL driver is now available as part of the latest Niakwa Scientific and Communications Driver product.**

Devices configured with this option can perform telecommunications options via \$GIO micro commands, similar to the capabilities provided by the DOS IBM2227.COM, IBM2227A.COM or IBM2227V.COM drivers (addressed as /dev/rtp2227, /dev/rtp2227a, or /dev/rtp2227x under DOS). However, the capabilities are handled by the Windows drivers (no DOS driver required), which result in improved performance and reliability under MS-Windows.

## 7.3 HELP Subsystem <sup>[4.10]</sup>

The HELP display now expands any ASCII tab characters it encounters in source text to an equivalent number of spaces in a source ASCII file (tabs at 8-character tab stops). This means that the warning generated by the Indexed Help file processing utility:

**Warning: TAB character(s) in source file.**

can be ignored, if the source editor used to create the file uses 8-character tab stops.

**NOTE:**        **The resulting indexed HELP file can only be used with Revision 4.10 of NPL or greater.**

## 7.4 Procedure Handling [4.10.23]

NPL's handling of attempts to perform unstructured exits from automatically called /EXIT procedures has changed. Early releases of NPL did not attempt to prevent unstructured exits from these procedures (i.e., using \$END, LOAD T, LOAD RUN, etc.) and could fail as a result. Problems also existed with RETURN ERROR(x) and CONTINUE RETURN when executed in an automatic /EXIT.

All attempts to perform an unstructured exit from an automatically called /EXIT procedure are trapped. The procedure exits as if a RETURN ERROR(266), "Automatic /EXIT procedure did not complete," occurred.

**NOTE:** For automatic calls to /EXIT procedures, RETURN ERROR(x) status is ignored and consequently all attempts to make unstructured exits during automatic calls to /EXIT procedures are ignored.

Alternatively, if \$OPTIONS byte 42 bit HEX(20) is set, an error 267, "Statement would do forced RETURN ERROR in /EXIT procedure, which is disabled," occurs on the offending statement. In this case, the programmer must explicitly exit the procedure (typically via a RETURN or RETURN ERROR).

As with automatically called /MAIN procedures, a warning is issued if a program STOPS or HALTs while executing an automatic /EXIT procedure.

**NOTE:** It is strongly advised that /EXIT procedures avoid statements that attempt to do unstructured exits. These will not operate as expected. If there is more than one module loaded, and each has a /EXIT procedure, then NPL can only guarantee that both /EXIT procedures will be called if each /EXIT procedure can be relied on to complete.

## 7.5 Immediate Mode Subroutine Handling [4.10.23]

The availability of DEFFN' subroutines using SF keys while programming has been extended. On previous releases of the RunTime, DEFFN' subroutines would become unavailable if a program modification was made that declared new variables (the new variable declarations were flagged with a P55 error). This restriction has been removed.

**NOTE:**        **The module must still be full resolvable (i.e., no syntax errors, unresolved references, unmatched structured statements, etc.) to execute the DEFFN' subroutines.**

## 7.6 Memory and Program Resolution [4.10.23]

Temporary memory requirements necessary to resolve FUNCTION and PROCEDURE bodies with large recursive variables have been significantly improved. Early NPL versions required a temporary allocation of approximately twice the total recursive variable requirement. For recursive values that are not assigned an initial value in the DIM statement, this maximum temporary allocation is now typically equal to the total recursive requirement.

**NOTE:**        **Recursive variables which are assigned an initial value still require a temporary allocation of approximately double the variable size.**

For example, if SPACEF is 250000, the following program will now execute:

```
1000 DIM A$60000, B$60000, C$60000
      PROCEDURE `X
      DIM D$60000
      END PROCEDURE
```

whereas previous releases of NPL would fail with an A01 error; "Memory Overflow (Text <--> Variable Table)," at the END PROCEDURE statement. If the above example changed DIM D\$60000 to DIM D\$60000="Z", then a larger temporary allocation is still required.

## 7.7 Atomic File Locking [4.10.23]

The NPL MS-DOS, MS-Windows and 386/DOS-Extender versions now support the HEX(01) bit in \$OPTIONS byte 39 that was previously supported only under UNIX. This option affects both local drives (with SHARE type locks) and network drives, on Novell NetWare or NetBIOS based networks.

When the HEX(01) bit of \$OPTIONS byte 39 is set to 0 (the default value), NPL issues an implied \$OPEN for the duration of each disk statement, if an explicit \$OPEN has not already been issued. The purpose of this implied \$OPEN is to ensure that if any other users issue an explicit \$OPEN against the disk image, this lock is respected (the platter is not accessed until the lock is released).

When the HEX(01) bit of \$OPTIONS byte 39 is set to 1, NPL will allow atomic disk statements (i.e., those completed with a single read or write operation) to access a diskimage file without issuing an implied \$OPEN. If other users have issued an explicit \$OPEN against the diskimage, the platter is not accessed until the lock is released.

The following disk statements are always atomic:

```
DATA LOAD BA
DATA SAVE BA
DATA LOAD BM      (if buffer size is a multiple of 256 bytes)
DATA SAVE BM      (if buffer size is a multiple of 256 bytes)
LIMITS INDEX
```

**NOTE:**        **The net effect of this option is to improve overall concurrent access to shared diskimage files (especially in networked environments where file locking operations are typically slow), if the majority of access to these diskimage files uses the above atomic operations.**

Under MS-DOS, MS-Windows and 386/DOS Extender, requests to read or write data on a file that is currently locked returns an error. If this occurs in any situation where the current task has suppressed an implied lock, NPL assumes the error could be due to a lock on the data being held by another user or task. NPL then issues the suppressed implied lock and retries the operation. If an error still occurs, the error is reported (as on previous releases), otherwise no error is reported and the program continues.

**NOTE:** The size of the implied lock issued in the event of a retry is affected by the HEX(02) bit of \$OPTIONS bit 39. If the HEX(02) bit is 0, a full file lock is issued. When the HEX(02) bit is 1, only the area of the file being read by the statement is locked.

For maximum concurrence both the HEX(01) and HEX(02) bits should be set.

Under UNIX, if an explicit \$OPEN is active on a file, any requests to read or write data in the file automatically wait for the lock to be released.

The following describes all possible values of \$OPTIONS byte 39. Values in this byte are treated as bit flags controlling special optimizations for implied \$OPEN operations.

Bit Position	Bit Value	Description
HEX(01)	0	No special avoidance of implied \$OPENS.
	1	RTI suppresses implied \$OPENS on atomic DATALOAD/DATASAVE BA/BM operations. If the affected area is currently locked by another task:  Under UNIX, NPL waits for the lock to be released.  Under MS-DOS, MS-Windows and 386/DOS Extender, NPL receives an error, suppresses it and issues a retry. If an error is still encountered, it is reported as in previous releases.
HEX(02)	0	No special handling of implied \$OPENS.
	1	The RunTime issues limited file locks instead of implied \$OPENS for atomic DATALOAD/DATASAVE BA/BM operations.  On UNIX systems with both the HEX(01) and HEX(02) bit set, the HEX(01) bit takes precedence.
HEX(04)	0	No special handling of implied \$OPENS.
	1	The RunTime issues limited file locks instead of implied \$OPENS for atomic DATALOAD/DATASAVE DC/DA operations.



Bit Position	Bit Value	Description
Other Bits	0	Reserved. Default 0 and should not be used.

**NOTE:** Previous releases of NPL will treat all values the same as HEX(00) on all MS-DOS based operating environments. The HEX(01) bit of \$OPTIONS byte 39 was previously implemented on all UNIX based operating environments. Refer to Section 8.5 for specific information on changes to \$OPTIONS.

Use of the HEX(04) option can substantially increase the ability to access a diskimage concurrently by many users, provided the accessed areas do not overlap. This option should only be set if DATALOAD DC/DA statements are always used by the application to access only single data records (save with one DATASAVE statement), starting at the first sector of the record.

**WARNING:** Reading multiple consecutive records with a single DATALOAD DC/DA statement with the HEX(04) bit set, without first locking the entire diskimage (with \$OPEN), can cause access failures on some operating systems, notably MS-DOS based systems. The errors may be reported as errors on the statement or as a "Device not ready" screen. Because the failure may only occur when records other than the first are concurrently accessed (and locked) by another user, this problem may not appear when testing the application in a single user environment.

## 7.8 BESDK Support <sup>[4.20]</sup>

### 7.8.1 Callback Support <sup>[4.20]</sup>

Release 4.20 introduced two new BESDK callback support functions which may be of use in the Windows environment. These are:

```
WORD rtp_postkey(HWND nplWindow,WORD keyvalue);
```

This function allows the external library to do the equivalent of sending a keystroke to the main NPL window. The handle of the NPL window must be supplied, and the key value is the NPL virtual key value reported to KEYIN plus 0x100 if the key should be reported as a special function key.

```
WORD rtp_modeless_notify(HWND hDlgModeless, WPARAM wparam);
```

This function should be called by external libraries that create modeless dialog windows whenever the modeless dialog receives a WM\_ACTIVATE message (with wparam containing the wParam value for the message). Doing this will ensure that NPL's main message dispatch loop properly calls IsDialogMessage for the window, which may be required to get correct handling of TAB keys and mnemonic keys while the dialog is active.

Both functions return a boolean value, which indicates whether the requested action completed successfully (TRUE) or not (FALSE).

A FALSE return value may mean either that an older version of RTIWIN is running or in the case of the keyboard posting function, it may indicate that the keyboard buffer was full.

### 7.8.2 32-Bit BESDK Support <sup>[5.00]</sup>

The 32-bit Windows BESDK assumes external routines are developed using the Microsoft Visual C/C++ compiler version 5.0. Other versions are untested.

Programmers are of course free to set up projects using the IDE as well. NPL uses the multithread external (DLL) version of the C RunTime libraries, and DLL's built for use as externals should use this as well.

#### Declarations

Declaring exported FUNCTIONS from a 32-bit DLL requires new keywords which are not compatible with non-Windows and 16-bit Windows environments. The "rtpall.h" file declares these as macros which will be null for other platforms and versions.

The recommended prototypes for the RTPEXT FUNCTIONS are:

```
RTP_WIN32DLLEXPORT int RTP_EXPORT CALLBACK  
RTPEXT(struct rtpreq * req_base);
```

It is also important for all external DEFFN' type functions to be declared with the "rtpdeffn\_ext" macro to ensure the correct calling conventions. For example:

```
RTP_WIN32DLLEXPORT int rtpdeffn_ext mysub(...);
```

The RTP\_WIN32DLLEXPORT macro is optional here, and is only required if the function must be visible externally.

External FUNCTION and PROCEDURE type functions (which always have one parameter) should be defined using the RTPFN\_DEFINE\_EXTERNAL() macro before the function body. For example:

```
RTPFN_DEFINE_EXTERNAL(myprocedure,pparam)
{
    ...
}
```

If the function requires a prototype (i.e., in an include file), use the RTPFN\_DECLARE\_EXTERNAL() macro in the include file. For example:

```
RTPFN_DECLARE_EXTERNAL(myprocedure,pparam);
/* fwd declaration */
```

### **Sharing DLL's**

Unlike 16-bit DLL's, tasks that use a 32-bit DLL do not implicitly share static variables. This makes it much simpler to write DLL code that is shareable.

Since you can still declare an RTPEXT\_SHAREABLE function, and NPL calls it when the library is loaded (and unloaded), if the current number of users is stored in a static variable it should never have a value other than 0 or 1, and so the return code from RTPEXT\_SHAREABLE should always be TRUE.

### **C Library used with DLL's**

The 32-bit Windows RunTime uses the Multithreaded DLL version of the Microsoft C library. For compatibility purposes your DLL should use this too. The /MD option to CL.EXE tells the C compiler to use this type of C RunTime library.

## 7.9 File I/O Using Streams [5.00]

Support for "stream" I/O has been added as an alternative access method to native files. Rather than using \$GIO command sequences or DATALOAD/DATASAVE BA statements, files can be accessed using more intuitive traditional "BASIC-style" OPEN, CLOSE, READ, WRITE statements. Access to native files using stream I/O is supported by enabling \$OPTIONS Byte 58 as follows:

Bit HEX(01) - Stream I/O with device specifications allowed.

**NOTE:**        **The file I/O functions described here are not supported syntactically or otherwise by NPL Revision 4.30. The use of Stream I/O requires NPL Release V.**

When this \$OPTIONS byte is enabled, alpha-literals or variables used in device specifications defined by a SELECT #n statement may be either:

- A 3-character (usually hex digit) value as on previous releases
- A directory name of any length

With \$OPTIONS byte 58 set to 0, behavior is as it previously was, requiring such a literal or alpha-variable to be at least 3 characters, of which the first 3 must be hex digits and all characters after the 3rd are ignored.

When the entered SELECT # string names a directory, access to native files in that specified directory is supported using the following new statements and functions:

```
$OPTIONS # alpha-function
OPEN # statement
SEEK # function
READ # function
WRITE # function
CLOSE # statement
```

If the entered value for SELECT # is blank, or if the device number has been cleared (i.e., with DATA SAVE DC CLOSE) access to files using OPEN # references the current directory.

Syntax details of these new statements can be found in Section 8.4. An example program follows.

```
0000 DIM O$64,buffer$64,line_len
      O$=$OPTIONS
      STR(O$,58,1)=OR HEX(01)      ;enable stream I/O
      $OPTIONS=O$

      WindowsDir$="C:\WINDOWS"
      SELECT #1<WindowsDir$>

      OPEN #1,"rtiwin.ini"
      O$=$OPTIONS#1
      STR(O$,2,1)=OR HEX(01) OR HEX(02) ;read mode
      $OPTIONS#1=O$
      WHILE TRUE
        line_len=READ #1,buffer$
      IF END THEN BREAK
        PRINT STR(buffer$,,line_len)
      WEND
      CLOSE #1
```

The behavior of these statements and functions may be operating system dependent; the legality and length of file names and directory names varies from platform to platform. Filenames may or may not be case sensitive, allow embedded spaces or forbid or restrict reserved characters like '/' '>' '<' '!' or other non-alphanumeric values. On some platforms, files may have specific types which exclude access via these calls. Portability across platforms is the responsibility of the programmer.

Use of other NPL statements with the above stream I/O statements is not implemented on this release, but may be implemented on future releases.

The status information for SELECT # that is associated with a directory appears as:

```
#n"directory-name"
```

instead of #n/ddd (where ddd is the three-digit device address.)

The status information for an open stream displayed by LIST DT appears in the open file table display, and is in the form:

```
#n"filename" current-seek-value
```

Refer to Section 8.4 and the appropriate statement reference section for more information on using Stream I/O statements.

## 7.10 New Century Date Module <sup>[4.21]</sup>

A new module (CENTDATE) has been added to the NPLSYS.BS2 system library. This new module contains the following two functions:

```
FUNCTION 'CenturyToday$           ;Returns todays DATE function as  
                                     "YYYYMMDD"
```

and:

```
FUNCTION 'CenturyDate$(GivenDate$,EraCutoff$2) ;Returns todays GivenDate  
with a century as  
"YYYYMMDD"
```

```
;The EraCutoff$ is the  
earliest year that the date  
could be valid and still be  
in the 1900's'
```

**NOTE:** As of NPL Revision 4.30, the system variable \$LDATE has been added which uses a four-digit year. Refer to Section 8.4.8 for more information on using \$LDATE.

## 7.11 Accented Characters <sup>[4.22]</sup>

The following key sequences can be used to enter accented characters from the standard NPL character set range HEX(10)-HEX(1F) under MS-DOS and MS-Windows.

You must use the numeric keypad to enter the number while ALT is depressed. Only recent (4.20) and higher versions of RTI/RTI386 will accept these key sequences under MS-DOS and MS-Windows.

NPL Code	Character Name	Key Sequence
HEX(10)	a circumflex	ALT-131 â
HEX(11)	e circumflex	ALT-136 ê
HEX(12)	I circumflex	ALT-140 î
HEX(13)	o circumflex	ALT-147 ô
HEX(14)	u circumflex	ALT-150 û
HEX(15)	a dieresis	ALT-132 ä
HEX(16)	e dieresis	ALT-137 ë
HEX(17)	I dieresis	ALT-139 ï
HEX(18)	o dieresis	ALT-148 ö
HEX(19)	u dieresis	ALT-129 ü
HEX(1A)	a grave	ALT-133 à
HEX(1B)	e grave	ALT-138 è
HEX(1C)	u grave	ALT-151 ù
HEX(1D)	A dieresis	ALT-142 Ä
HEX(1E)	O dieresis	ALT-153 Ö
HEX(1F)	U dieresis	ALT-154 Ü
HEX(7D)	e acute	ALT-130 é
HEX(7E)	c cedilla	ALT-135 ç
HEX(7F)	cent sign	ALT-155 ¢



## 7.12 Enhanced Boolean Expressions [4.30]

In logical expressions, a numeric expression may be used as a boolean, where a 0 value in the numeric expression means FALSE, and a non-0 value means TRUE.

For example, the following statements are equivalent:

```

IF X<>0 AND Y<>0 OR Z<>0
IF X<>0 AND Y<>0 OR Z
IF X<>0 AND Y OR Z<>0
IF X<>0 AND Y OR Z
IF X AND Y<>0 OR Z<>0
IF X AND Y<>0 OR Z
IF X AND Y OR Z<>0
IF X AND Y OR Z

```

The new syntax may be used in IF, WHILE, UNTIL and most logical CASE statements.

For example:

```

WHILE There_May_Be_More
UNTIL User_Pressed_Exit OR Valid_Data_Entered
CASE UserName$="Admin" OR Override

```

For CASE statements a simple numeric expression is treated as a numeric CASE, rather than a logical CASE. To use a numeric expression as a boolean in a CASE statement, you must use the <>0 form.

For example:

```

SWITCH                                ;;logical switch
CASE City$="Toronto"                  ;;typical logical case
CASE File_Is_Open<>0                  ;;valid use of boolean
CASE File_Is_Open                      ;;invalid use of boolean
END SWITCH

```

**NOTE:** Pcode generated for this syntax is new to Revision 4.30, but is downward compatible to older releases. On older releases, the expression is displayed as the equivalent '<>0' form. If the statement is changed using the older version, the distinction between the two syntaxes is of course lost.

When using this form, care must be taken to use spaces to separate the boolean expression from any logical operators.

The following are NOT equivalent:

```
IF S AND Y
IF SANDY
```

## 7.13 HALT Key Sequence <sup>[4.30]</sup>

The interpretive MS-Windows, DOS and Pharlap versions of NPL have been enhanced to support either of the following key sequences as the HALT key:

CTRL+ALT            Default under MS-DOS and Pharlap

CTRL-BREAK         Default under MS-DOS

In addition, developers may choose to disable one or both of the HALT key sequences.

**NOTE:**         **Refer to Section 8.5 for \$OPTIONS Updates (Byte 52) below for details on enabling this new feature.**

## 7.14 NPL Interactive Line Editor

This section discusses general enhancements and updates implemented in the NPL line editor.

### 7.14.1 In-Line Comments Extended <sup>[4.21]</sup>

In-line comments are now accepted in function calls within numeric expressions at the same locations where they are permitted in a procedure call (i.e., after the open parentheses or a comma after a parameter.) Programs saved with this syntax require Revision 4.21 of NPL or later to load and are only understood by Revision 4.21 of B2C or later. For example:

```
1000 hDlg='CreateDialogIndirectParam(      ;now build it...
      'LocalDS,                          ;use this instance
      'CastPointer$( 'GlobalLock(hDlgTemplate),1),
      NSM_parentWindow,                  ;owner window
      lpDlgProc,                          ;dialog procedure
      lpInfo)                             ;initialization info
```

### 7.14.2 New Line Editor Functions (CUT/COPY/PASTE) <sup>[4.22]</sup>

This section discusses the new Cut/Copy/Paste functions of the NPL line editor and how to implement these new functions.

**NOTE:** All Special Function Key values in this section are stated as HEX() values. For example, SF'40 means a KEYIN returns HEX(40), which corresponds to a decimal value of 64.

#### Default NPL Keyboard Equivalence Extensions

Support for the following operations using both the mouse and keyboard have been added to the NPL line editor.

**CUT      COPY      DELETE      PASTE      UNDO**

The default NPL keyboard equivalences were extended in Release 4.22 to support these new operations. In addition, a number of previously undefined CTRL sequences were added to the default keyboard equivalences.

The following table lists these new default keyboard equivalences.

NPL Code Key	NPL Virtual Key	MS-DOS Key	Operation
'49	(-DELETE-) Text Block	DELETE *	Delete selected text
'59	(-SHIFT-DELETE-) Text Block  (-SHIFT-DELETE-)	CTRL-DELETE *	Cut selected text  Line Delete If no text is selected.
'40	(-COPY- ) Text Block	CTRL-Y CTRL-+	Copy selected text
'41	(-PASTE-)	CTRL-V CTRL-(-)	Paste selected text
'5F	(-SHIFT-RECALL-)	CTRL-T	Undo last operation
'7E	(-TAB-)	TAB *	Block Indent selected text
'7F	(-SHIFT-TAB-)	SHIFT-TAB *	Block Deindent selected text
'55	(-SHIFT-SOUTH-)	CTRL-SOUTH	Move end of line
'56	(-SHIFT-NORTH-)	CTRL-NORTH	Move start of line
'5A	(-SHIFT-INSERT-)	CTRL-INSERT	Line Insert

\* If no text is selected these keys operate as on previous releases. Selected text appears highlighted in inverse video.

**NOTE:** The '59 virtual key does either a line delete if no text is selected (same as previous releases) or cuts the selected text to the clipboard.

**NOTE:** When a "-" (dash) is used in a key sequence it indicates that the key sequences should be pressed simultaneously.

On the NPL Windows versions, the following alternatives are available for some NPL virtual keys:

NPL Virtual Key	MS-Windows Key
(-SHIFT-DELETE-)	SHIFT-DELETE
(-SHIFT-INSERT-)	SHIFT-INSERT
(-SHIFT-NORTH-)	SHIFT-NORTH
(-SHIFT-SOUTH-)	SHIFT-SOUTH
(-SHIFT-EAST-)	SHIFT-EAST
(-SHIFT-WEST-)	SHIFT-WEST

**WARNING:** If an application uses a modified **KEYBOARD.TBL** or **KEYBOARD.WIN** file from a previous release, the file will need to be updated to use these new default keyboard mappings.

#### **Using the Mouse to Select Text**

Text selection with the mouse is supported by default in MS-Windows when using RTIWIN, RTPWIN, RTIWIN32 and RTPWIN32. Text selection with the mouse is also supported under MS-DOS by RTI, RTP, RTI386 and RTP386 when the RunTime is launched using the "/K" command line startup option (assuming an appropriate mouse driver is configured.)

Text is most easily selected by clicking and dragging using the left mouse button. A left mouse click on the screen moves the cursor to the nearest editable location. For multi-screen program lines, clicking on the --PREV-- or --NEXT-- indicator does the equivalent of a PREV-SCRN and NEXT-SCRN key.

**Using the Keyboard to Select Text**

In order to select text from the keyboard, the CUA keyboard mode must first be enabled using a new \$OPTIONS byte. Bits HEX(01), (02), and (04) of byte 57 of \$OPTIONS have been implemented and support the following bit values:

Byte 57	HEX(00)	Default
	HEX(01)	When set to 1, enables CUA key mode in command line editing.
	HEX(02)	When set to 1, enables CUA key mode in LINPUT
	HEX(04)	When set to 1, enables CUA key mode in INPUT

For example, to enable CUA key mode in command line editing only, set \$OPTIONS byte 57 to HEX(01). To enable CUA key mode in command line editing and under program control in LINPUT entries, set \$OPTIONS byte 57 to HEX(03).

Byte 57 of \$OPTIONS makes editing keys follow CUA conventions more closely. In particular, sections of the line can be selected using shifted movement keys. The selected section appears highlighted in inverse video. In this mode, the shifted movement keys move only as far as an unshifted key, so for example (-SHIFT-EAST-) moves 1 space to the right, not 5.

**NOTE:** Refer to Chapter 8 for a complete description of all \$OPTIONS Byte 57 values.

The following keys are effected:

NPL Virtual Key	Operation in non-CUA Mode	Operation in CUA Mode
(-SHIFT-NORTH-)	Move to start of line	Select, move up 1
(-SHIFT-SOUTH-)	Move to end of line	Select, move down 1
(-SHIFT-EAST-)	Move 5 to right	Select, move right 1
(-SHIFT-WEST-)	Move 5 to left	Select, move left 1

Niakwa reserves the right to extend this CUA-compliant behavior on future releases.

**NOTE:** In DOS mode, due to MS-DOS keyboard driver limitations which report the same code for shifted and unshifted keys, the (-SHIFT-x-) virtual keys are by default mapped to the CTRL+keypad combinations.

**NOTE:** On some terminals, underline and inverse video appear the same. For this reason, LINPUT statements using the "-" option to underline the affected variable may be unsuitable for using the text selection CUT/PASTE.

When CUA key mode is enabled, the previous functions provided by the shifted keys (to move 5 spaces in either direction) are no longer available using the arrow keys. If you require these functions, the following new NPL virtual keys will perform the equivalent of the move-by-5 functions, and their shifted equivalents.

NPL Virtual Key	NPL Code Key
(-WEST-5-)	SF'4E
(-SHIFT-WEST-5-)	SF'5E
(-EAST-5-)	SF'4B
(-SHIFT-EAST-5-)	SF'5B



In addition the following new numeric keypad function virtual keys may be defined by developers to provide functionality equivalent to edit mode SF keys (whether you are in edit mode or not):

NPL Virtual Key	NPL Code Key
(-BEGIN-)	SF'47
(-SHIFT-BEGIN-)	SF'57
(-ENDLINE-)	SF'44
(-SHIFT-ENDLINE-)	SF'54

When CUA keyboard mode is enabled, the shifted versions select as well as move. No keys are currently assigned by default to these functions for the IBM DOS or MS-Windows keyboards.

The NPL utility "Keyboard Translation Table Editor" has been updated to include these new virtual keys in the reminder display.

**WARNING:** If an application uses a modified **KEYBOARD.TBL** or **KEYBOARD.WIN** file from a previous release, the file will need to be updated to use these new default keyboard mappings.

#### Delete

Once selected, program text can be deleted using the (-DELETE-) key, or cut (deleted and copied to the clipboard area) by using the (-SHIFT-DELETE-) key. Typing non-movement characters implicitly deletes the selected text first.

**NOTE:** If no text is selected, (-DELETE-) and (-SHIFT-DELETE-) operate as before.

#### Copy

Text selected may be copied to the clipboard area without deleting it by using the (-COPY-) key. When using the default keyboard mappings, this new NPL virtual key (SF '40) is mapped to the value CTRL-Y on IBM compatible (and most other) keyboards. In addition CTRL-+ on an IBM numeric keypad is mapped to this key. When in edit mode, SF'0 will also perform the COPY function.

**Paste**

Text that has been copied to the clipboard area may be pasted into the current position by using the (-PASTE-) key. This new NPL virtual key (SF'41) is mapped to the value CTRL-V on IBM (and most other) keyboards. In addition CTRL+'-' on the IBM PC numeric keypad is mapped to this key. When in edit mode, SF'1 will also perform the PASTE function.

**NOTE:**           **Text selection is canceled by any cursor movement (except as specified below.)**

**Undo**

A single level undo key is now supported. The most recent change to the text buffer can be reversed by pressing the (SHIFT-RECALL) key. This NPL virtual key (SF'5F) previously operated the same as (-RECALL-). The key is mapped as CTRL-T on the IBM (and most other) keyboards.

When the undo key is pressed, either the most recently entered text string is deleted, or the most recently deleted text string is reinserted (or both). Any text insert by the undo command is selected. An accidental undo key can be reversed by pressing the key again.

**Block Indent and Deindent**

When text is selected on a command line, the TAB and SHIFT-TAB keys perform block indent and deindent functions at the start(s) of the selected line(s). Block indenting and deindenting can be useful for adding and maintaining structured indenting to programs.

The amount of the indent is determined by the size of the gap between the 5th and subsequent non-blank position in the \$TABS system variable (by default, 4 spaces). When deindenting, non-blank characters are never removed. Statement separators (colons) at the start of the selection area or the start of a subline are not moved by either operation.

**Paste Control Enhanced** [4.30/5.00]

With Release 4.30, when text is pasted from the Windows Clipboard, ASCII TAB characters (HEX(09)) are replaced by single spaces (HEX(20)).



## CHAPTER 8

# LANGUAGE ENHANCEMENTS

### 8.1 Overview

This chapter covers syntactical additions and enhancements implemented since the introduction of NPL Release IV, which would otherwise be covered in the NPL Release IV Statements Guide.

Section 8.2 discusses corrections to the NPL Release IV Statements Guide.

Section 8.3 describes extensions to existing language statements and syntax.

Section 8.4 describes new language statements and syntax.

Section 8.5 describes enhancements or additions to special system variables \$MACHINE and \$OPTIONS.

## 8.2 Corrections

The following sections outline known corrections or typographical errors that exist in the NPL Release IV Statements Guide. Pertinent sections and pages are referenced.

### 8.2.1 \$SOURCE Correction <sup>[4.10]</sup>

On page 2-673, the description for alpha-variable-2 of the \$SOURCE function is incorrect. The correct description is as follows:

receiver-variable = variable to receive ASCII text source equivalent of P-code.

alpha-variable-2 = P-code that is to be converted to ASCII text.

### 8.2.2 Library Function Corrections <sup>[4.10]</sup>

In Section 3.2, the second file name in the file listings section is incorrect. Replace the second file name "NPLSYS" with "NPLDEV". The description of the file is correct.

In Section 3.3.4, the example code at the beginning of the section is incorrect. Replace the example code with the following:

```
Pcodeline$=$SOURCE(Pcodebuffer$[ ,Table$ ])
```

In Section 3.4.1 replace the existing section with the following:

3.4.1 'ObjectIoCreateFile

```
FileName$8, DeviceNumber,Size, /POINTER Stream$
```

This library routine initializes the specified program name on the indicated Device to be an empty program. Stream\$ is a string field variable used for buffered output to the file. A non-zero NPL error code is returned if the operation cannot complete (e.g., file already exists as data file, cannot create file, etc.).

In Section 3.4.2, the section heading is incorrect. Replace the section heading with the following:

'ObjectIoClearIdentifierTable

## 8.3 Syntax Enhancements

This section provides detailed descriptions of enhancements to existing syntax. These sections are provided as supplements to the existing reference in Chapter 2 of the NPL Release IV Statements Guide.

### 8.3.1 \$DEMO

#### **Enhanced Keywords** <sup>[4.10]</sup>

Two new keywords for \$DEMO scripts allow a demo to change the default background and foreground colors for boxes displayed by \$DEMO. The new keywords are:

(BOXFGn)                      Set foreground color to n (0 <= n <= 7)

(BOXBGn)                      Set background color to n (0 <= n <= 7)

These codes must appear before any boxes they affect and remain in effect until the RunTime is exited or until overridden by a new value. No spaces are permitted in the codes.

Before these color codes can be used, color support must be enabled. Refer to Section 7.4 of the Programmer's Guide for details.

**NOTE:**                      **These codes are ignored by older versions of the RunTime (revisions prior to 4.10).**

#### **Improved Debugging of \$DEMO Scripts** <sup>[4.10.23]</sup>

A new \$OPTIONS bit simplifies the process of debugging demonstrations that use \$DEMO scripts. When the HEX(10) bit of byte 42 of \$OPTIONS is enabled, any keyboard feed from a \$DEMO script is suspended to allow for immediate mode commands if the program HALTs or executes a STOP statement. If the bit is turned off (the default) the \$DEMO script is used to enter an immediate mode command, as on previous releases.

#### **\$DEMO Extended** <sup>[4.20]</sup>

On 32-bit platforms, the system variables for \$DEVICE(), \$HELPINDEX, and \$DEMO have been extended to 128 bytes to allow for longer filenames.

#### **\$DEMO Mouse Support** <sup>[4.22]</sup>

\$DEMO scripts now support commands to position the mouse cursor at a specific row and column (and update \$MACHINE bytes 23 and 24) so that subsequent mouse events reported as keystrokes are correctly handled by application programs.

The syntax in a demo script is:

(RCrr,cc)

Where: RC is the keyword that indicates a mouse row-column item, rr is the numeric value of the row and cc is the numeric value of the column to which the mouse cursor is moved.

For example:

(RC5,10)

Indicates that the next mouse event occurs at row 5, column 10.

The keyboard logging facility (SELECT LOG) will add an item of this type to the log file before each mouse event keystroke. Older versions of RTI ignore these items if they are found in a \$DEMO script. In addition to updating \$MACHINE, the mouse cursor is moved to the indicated row and column, which must be within the legal range.

**NOTE:**        **In a DOS box under Windows, the mouse cursor will only move if the full screen option is in effect.**

### 8.3.2 \$DEVICE

#### **\$DEVICE() Extended** <sup>[4.20]</sup>

On 32-bit platforms, the \$DEVICE entries have been extended to allow up to 128 bytes for longer filenames.

#### **\$DEVICE Default Number Increased** <sup>[5.00]</sup>

On 32-bit platforms, the default maximum number of supported \$DEVICE entries has been increased from 16 to 255. This effectively makes the /D startup option no longer necessary on these platforms.

**Passthrough Printing "XPA=Y"** [4.21]

\$DEVICE permits the MS-Windows RunTime transparent access to MS-Windows printer drivers that support the "PassThrough" printer escape controls, if the XPA=Y clause is added to the device specification.

**NOTE:** Use this special device clause if your application requires passing printer escape sequences (i.e., font changes, compress print) to a printer through Windows' Print Manager. If this device clause is not used, some printer escape sequences may be stripped out by the MS-Windows Print Manager.

For example:

```
$DEVICE(/215)=">(HP LaserJet Series II,HPPCL,LPT3:) ERR=Y XPA=Y"
```

**NOTE:** The XPA=Y clause only has an effect on device access made through MS-Windows printer drivers (i.e., uses the ">(xxxxx)" syntax), and only if the driver supports the "PassThrough" escape functionality.

The output from the document is handled by the Print Manager. If the printer driver does not support the PassThrough functions, an error P48 occurs on the first access to the printer device.

**NOTE:** Since many standard printer drivers do not support passthrough mode, when the XPA=Y option is used NPL will bypass any specified printer driver and use RAW.DRV (which supports "PassThrough" escapes) instead. You must have complete knowledge of the control codes used by a printer to use the transparent print option. Any documents sent to the printer using transparent mode must be sure to set all relevant spacing and other options, since these can be left in any state by the preceding document. This functionality has been implemented in NPL Revision 4.30 and greater, and is specific to Windows 95/98 environments.

**Read Only Status (RDO=Y)** [4.20]

A clause to the \$DEVICE specification allows accessing printer class files which are read-only using the \$GIO read microcommands. Normally, any attempt to access a printer class file requires write access to the file, and if the file does not exist an attempt is made to create the file automatically. On previous releases, if the file was tagged as read-only, a device not available error occurred when attempting to access the file.

The clause 'RDO=Y' added to a \$DEVICE specification indicates that the program will only read from the file (using \$GIO) and so write access is not needed. In this case, the device is opened by NPL in read only mode and will not be created (if it does not exist, an error occurs).

For example:

```
0010 $DEVICE(/01C)="G:\READONLY.DAT RDO=Y"
      DIM L$10,Length,Buffer$512
      $GIO/01C(HEX(8700 C620),L$)Buffer$
      Length=VAL(STR(Buffer$,9),2)
```

**NOTE:** If the file G:\READONLY.DAT is tagged as a read-only file, the \$GIO statement would fail on previous releases, but will successfully work with this release.

### 8.3.3 INCLUDE Command Restrictions <sup>[4.22]</sup>

Immediate Mode command lines containing INCLUDE statements are no longer legal unless the current context module is resolved. This restriction is required since the resolution of the INCLUDE statement can effectively invalidate references to static variables preceding the INCLUDE statement.

For example:

```
:0 ;SUB
      DIM Z$32
:SAVE T"SUB"
:0 ;ROOT
      INCLUDE T"SUB"
:RUN
:MODULE "SUB"
:LIST
:0 ;SUB
      DIM Y$32
:Z$="X":INCLUDE T"SUB"
                        ^ERR 268
Immediate INCLUDE is not legal.
Current module is not resolved.
```

The INCLUDE statement implicitly deletes the variable Z\$ in module SUB at resolve time. Executing the Z\$= assignment would then cause module corruption on revisions prior to 4.10. With subsequent revisions the INCLUDE statement is tagged as illegal in immediate mode with a new error code 268.



**8.3.4 KEYIN** <sup>[4.30]</sup>

The KEYIN statement now permits labels as branch targets wherever line numbers are permitted. Line number and labels may be mixed if necessary. For example:

```
KEYIN Key$, ,SpecialFunctionKey
KEYIN Key$,NormalKey,SpecialFunctionKey
KEYIN Key$,30,SpecialFunctionKey
KEYIN Key$,NormalKey,40
```

**8.3.5 LIST Command** <sup>[4.10]</sup>

\$DECLARE and FUNCTION/ PROCEDURE declarations optionally accept empty parentheses '()' to indicate no parameters. The decompiler removes the empty parentheses when LISTing. For example:

```
:0001 $DECLARE 'GetDesktopWindow( )
:LIST
0001 $DECLARE 'GetDesktopWindow
```

**8.3.6 LIST DC with LDATE** <sup>[4.30]</sup>

The LDATE keyword has been added to the list of restriction keywords of the LIST DC statement for displaying a listing of a diskimage or library.

In the General Form description, the restrict parameter is changed as follows:

*restrict= { LDATE rel-op alpha-mask }  
{ any previously valid restrict }*

The LDATE keyword specifies a mask on the extended file date in the format yyyy/mm/dd.

The LDATE keyword should be used in preference to the DATE keyword when libraries or diskimages contain files dated in more than one century value.

Examples:

```
0010 LIST DC T LDATE>="1997/01/12"
0020 LIST DC T LDATE="2001/01/31"
0030 LIST DC T LDATE<=CutoffDate$
0040 T$=$LDATE
      M$=STR(T$,,4)&"/"&STR(T$,5,2)&"/"&STR(T$,7,2)
      LIST DC T LDATE=M$      ;;show files saved today
```

### 8.3.7 LIST DT Command <sup>[4.10]</sup>

The LIST DT command has been revised to indicate special situations related to file locking. Files that are determined to be globally read-only (and so do not require file locking on a \$OPEN), appear with the designation:

```
$OPEN/xxx ;; Read only file
```

Files that are determined to be local, where there is no SHARE loaded (so file locking is not available), appear with the designation:

```
$OPEN/xxx ;; local file, no SHARE.EXE
```

**NOTE:** These two cases represent the only legitimate cases where two RunTimes may have the same diskimage file \$OPEN at the same time.

**WARNING:** Applications that share local files must have SHARE (or equivalent file-sharing support) loaded to avoid possible data corruption. If SHARE is not loaded, file locking of local files is effectively ignored. MS-Windows can issue a warning message box before permitting a second Windows RunTime to access local files when SHARE is not loaded, but does not detect multiple MS-DOS tasks accessing the file in similar circumstances.

Raw diskettes are an exception when considering diskimage locks. In general, diskimage locks on raw diskettes cannot be enforced under MS-DOS. The MS-Windows version of the RunTime contains logic to prevent two MS-Windows tasks from accessing the same raw diskette at the same time, but this mechanism is not respected by MS-DOS tasks running at the same time.

### 8.3.8 LIST PUBLIC and LIST DIM Commands <sup>[4.30]</sup>

LIST PUBLIC and LIST DIM have been revised to suppress displaying the names of public entities defined in named public sections, if the declaring module is scramble protected.

Programmers who ship scramble protected libraries containing named public sections are advised to give all entities in these sections prefixed names, and to document that all public names with this prefix are reserved.

This convention is intended to avoid possible conflict with application programs that might attempt to declare the same entity name in a public declaration. Such a conflict will now be somewhat more difficult for the application programmer to diagnose, since a LIST PUBLIC or LIST DIM would not show the location of the library's prior declaration.

### 8.3.9 MAT SORT <sup>[4.10]</sup>

Performance of MAT SORT under 32-bit (386/DOS-Extender and Windows 95/NT) versions of NPL for arrays less than 64K bytes and with 2-byte locators has been improved to the equivalent of Release III.

**NOTE:**           **Arrays larger than 64K bytes are still permitted with 2-byte locators, but are slightly (estimated 15%) slower than would be predicted based on performance for smaller array sizes.**

### 8.3.10 MODULE Command <sup>[4.10]</sup>

Programmers should be aware of a subtle change in the MODULE statement. This change can affect Immediate Mode commands.

Currently, when a program is STOPped or HALTed (i.e. can be CONTINUED), commands entered in Immediate Mode act in the context of the currently HALTed location. This allows you to inspect and modify variables with the effect being as if the entered lines were part of the program at the indicated location.

The situation where the program cannot be CONTINUED is a little more complicated. On early 4.00 revisions, the context used initially would be set to the current module, but could be changed by any program modification to point to the changed module. Variables local to a FUNCTION or PROCEDURE would remain in context until an operation exits or cancels the function.

Immediate Mode MODULE statements would have no effect on the context, except for specific statements that referred to program modifications, (i.e. program LIST statements, LOAD, SAVE, RENUMBER).

On subsequent revisions, an Immediate Mode MODULE statement entered when a program cannot be CONTINUED affects the context of subsequent Immediate Mode commands. This allows easier inspection and modification of static variables in various modules during a debug session.

Any operation (such as a legal RETURN or RETURN(x) statement) which re-establishes the CONTINUE-able status of the program will return the context to the HALTED location.

### 8.3.11 \$OSERR [4.10]

File sharing has been extended to better support multi-tasking environments when SHARE is installed (e.g., DOS tasks under MS-Windows, DESQview, Novell DOS 7, etc.) and mixed networks.

To better diagnose any sharing conflicts that may occur under NetBIOS networks and multi-tasking environments, the messages reported for MS-DOS errors (\$OSERR) have been extended to include the extended MS-DOS error codes 13-58 (this was previously done only on the MS-Windows implementation of NPL).

**NOTE:**            **Developers with translated RTIERR.HLP or RTIPERR.HLP files should update their files to include these new error messages.**

### 8.3.13 ON ERROR [4.30]

The ON ERROR statement has been enhanced to allow support for both line numbers and statement labels in reference to the branching statements.

The new General Form is as follows:

```
ON ERROR alpha-variable1, alpha-variable2 GOTO {line-number}
                                             {statement-label}
```

The statement to be executed when the unhandled error occurs may be specified using either a line number or statement-label.

Examples:

```
0010 ON ERROR ErrCode$,ErrLine$ GOTO 1000
0020 ON ERROR ErrCode$,ErrLine$ GOTO ErrorHandlerCode
```

### 8.3.13 READ DC with LDATE <sup>[4.30]</sup>

The LDATE keyword has been added to the list of restriction keywords of the READ DC statement for reading the index of a diskimage or library.

In the General Form description, the restrict parameter is changed as follows:

```
restrict= { LDATE rel-op alpha-mask }
          { any previously valid restrict }
```

The LDATE keyword specifies a mask on the extended file date in the format yyyy/mm/dd.

The LDATE keyword should be used in preference to the DATE keyword when diskimages contain files dated in more than one century value (i.e., 19xx and 20xx).

For Example:

```
0010 READ DC T FileName$,NextRead,LDATE>="1997/01/12"
0020 READ DC T FileName$,NextRead,LDATE="2001/01/31"
0030 READ DC T FileName$,NextRead,LDATE<=CutoffDate$
0040 T$=$LDATE
      M$=STR(T$,4)&"/"&STR(T$,5,2)&"/"&STR(T$,7,2)
      NextRead=0
      REPEAT
        READ DC T FileName$,NextRead,LDATE=M$ :;find
        files saved today
      UNTIL NextRead=0
```

## 8.4 New NPL Statements and Syntax

This section describes new statements and syntax that have been introduced since Release IV. Where applicable, the revision is shown in superscript showing when the particular statement was introduced.

### 8.4.1 \$ARGS <sup>[4.21]</sup>

Any arguments on the command line following the boot program name are normally ignored by the RunTime. The value of these arguments can now be determined by NPL programs using the new system variable, "\$ARGS". This variable is only legal on the right hand side of a string assignment (LET) statement.

For example:

```
0010 ;P_ARGS
      A$=$ARGS
      PRINT A$
```

The following examples illustrate the output of the above program:

RTI /D20 /Xmylib p_args	displays:	(blank)
RTI p_args receivables	displays:	receivables
RTI /D20 /Xmylib p_args payables	displays:	payables

**NOTE:** On the MS-DOS version of NPL, the entire command line including information past the boot program name is scanned for the /M and /U command line options. In general, developers are encouraged to avoid passing argument values starting with '/' or '-' using \$ARGS.

**8.4.2 CLOSE #** <sup>[5.00]</sup>

The **CLOSE #** statement closes and frees the file currently open on the specified stream number.

The syntax is as follows:

**CLOSE #***stream\_number*

For example:

**CLOSE #30**

The **CLOSE #** statement allows explicit closing of a stream, however a subsequent **OPEN #** of the same stream with a different file or path performs an implicit **CLOSE #** on that stream.

**8.4.3 \$DECLARE** <sup>[4.20]</sup>

The \$DECLARE statement allows developers to easily access external routines in MS-Windows DLL files.

**NOTE:** While the \$DECLARE statement gives developers access to the MS-Windows API at a higher level than the Niakwa Gateway to Windows API library product, a formal understanding of “C” and the Windows API calls is still required. Developers interested in simplified high level routines at a very high level should explore the capabilities of Niakwa’s Visual NPL products.

\$DECLARE functions may only be called on the 16-bit or 32-bit MS-Windows versions of NPL (RTPWIN/RTPWIN and RTPWIN32/RTIWIN32).

The syntax for \$DECLARE is as follows:

First, define C-data-type as one of the following keywords denoting a C language data-type:

<u>keyword</u>	<u>C equivalent data type</u>	<u>Windows typedefs</u>
INT()	unsigned int	UINT
INT(-)	int	
INT(1)	unsigned char	BYTE
INT(-1)	char	
INT(2)	unsigned short int	WORD,HANDLE,Hxxx
INT(-2)	short int	
INT(4)	unsigned long	DWORD
INT(-4)	long	
NUM()	double	
STR()	char _far * (null-terminated)	LPSTR,LPVOID
DIM()	char []	LPxx
DIM(n)	char [n]	
STR(DIM)	char *[]	

The \$DECLARE statement syntax is:



```
$DECLARE 'function-name' (param-1 [, param2 ...]) [[TO ret-type]] [= dllspec]
```

where:

*param-n* is *[/POINTER] [ [TO] RETURN]* C-data-type [*param-identifier*] and specifies the C data type required for each function parameter. *TO RETURN* or *RETURN* indicates that the parameter is passed by reference, and receives a return value. */POINTER* indicates that the parameter must not use intermediate storage when passing by reference.

The */POINTER* options is used either for performance reasons, or because the address is stored by the called procedure for later use. If the address is stored, be sure that the NPL memory reorganization is disabled, and that the variable remains declared for as long as the called procedure may reference it. Incorrect use of */POINTER* (passing a string of insufficient length, for example) may result in unpredictable errors and should only be used by advanced developers.

*ret-type* is a C-data-type specifying the function return type. The default is *INT()* (unsigned integer).

*dllspec* is an alpha-variable or literal specifying the library and entry point name of the function, for example:

```
"[library-name.]procedure-name"
```

The default is to look for the function using the same name as the declared function-name.

A *\$DECLARE* statement is a special version of an external *FUNCTION* declaration, with the following differences:

1. The external function is located using the GetProcAddress Windows API, directed to one of the default external libraries or to a specific library. The default API libraries for 16-bit windows are:

NPLKDLG.DLL	KERNEL.EXE
USER.EXE	GDI.EXE
MMSYSTEM.DLL	DDEML.DLL
GSWDDL.DLL	TIFF.DLL

**NOTE:** The NPLKDLG.DLL provides a number of functions also found in KCML and is intended to syntactically support these functions upon migration to NPL. This file is not required by NPL.

2. For 16-bit windows, parameters are passed in a method compatible with the `_far` and `_pascal` C keyword, (i.e., parameters are pushed in order of declaration and all pointers are 32-bits). For 32-bit windows, parameters are passed in a method compatible with the `_stdcall` C keyword, (i.e., parameters are pushed in reverse order of declaration and all pointers are 32-bits). Numeric types are normally passed by value, while string types are typically passed by reference.
3. All parameters passed by reference are passed using copies of the original data, unless the `/POINTER` option is used on the parameter. For `STR()` parameters, data conversion on NPL strings to C strings occurs before the call, and from C strings to NPL strings after the call. No data conversion is done on parameters passed with the `/POINTER` option.

#### **\$DECLARE Warnings and Cautions** <sup>[4.20]</sup>

Developers who have used the KCML implementation of \$DECLARE should observe the following cautions and restrictions to ensure compatibility:

- The location of the \$DECLARE API is established by NPL at resolve time. If the API cannot be located at resolve time, it cannot be called (an error occurs at execution time). KCML defers locating the API until execution time. If the DLL environment is changing (i.e, if DLL's are copied into or deleted from the current directory between resolve time and executing a call to a \$DECLARE), NPL may not find the same API as KCML.
- KCML permits calls to \$DECLARED functions which ignore any return values using 2 types of syntax - with the `GOSUB` keyword or without. NPL does not support calls to \$DECLARE functions using the `GOSUB` keyword.

- KCML permits multiple \$DECLARES for the same function name and DEFFN declarations with the same name. The first declaration in a program is used. NPL does not permit multiple \$DECLAREs for the same function name.
- KCML permits references to \$DECLAREd functions to appear before the declaration. NPL requires the \$DECLAREs statement to appear before any use of the function name.
- KCML permits incorrect parameter counts and types to execute. NPL requires the number and types of parameters to \$DECLAREs functions to be correct, with the following exception:

A numeric argument can be passed to a STR() or DIM() parameter. The value passed to the API is the integer value of the argument cast to a pointer. This may not be used if the RETURN option is specified on the parameter. The main use of this is to allow passing a NULL pointer to API calls using the value 0. A number of other windows API's may also require this kind of argument for 'special cases'.

The following parameter type is only supported syntactically. Attempts to actually call \$DECLAREd functions with this type will generate a run-time error:

STR(DIM)

**NOTE:** Only INT() return value types are supported. Other return types are only supported syntactically. Attempts to call \$DECLAREd functions with other return types will generate a run-time error.

There are no known API's in the MS-Windows environment that require this type of argument or return code.

Versions of NPL running on platforms other than MS-Windows support the syntax and semantics of \$DECLARE. However, attempting to call any \$DECLAREd function will result in the same runtime error as would occur if the named function did not exist under Windows.

An error code (305, recoverable) occurs if the function named in the \$DECLARE statement cannot be located when the function is called. The default error description is:

'Cannot locate \$DECLARE function.'

**NOTE:** This new error description has been added to the **ERRORMSG.HLP** file included with all Release 4.20 RunTimes.

**Extended \$DECLARE Parameter Types** [4.21]

\$DECLARE now supports the following parameter types, which previously generated errors at RunTime:

```
[TO] RETURN INT()
[TO] RETURN INT(-)
[TO] RETURN INT(1)
[TO] RETURN INT(-1)
[TO] RETURN INT(2)
[TO] RETURN INT(-2)
[TO] RETURN INT(4)
[TO] RETURN INT(-4)
```

**NOTE:** Parameters passed to the function must be in the range of a **BIN(,-6)** integer or a range error (**X71**) will occur. This is a requirement even for just **RETURN** parameters. Otherwise there is no checking for input parameters being in a specific input range and the low order part of the value only is passed.

**Dialog Support** [4.21]

The first dialog made visible does not have to be the last hidden or destroyed for the hidden main window to reappear. In addition, display of multiple dialog boxes is permitted, but only the most recently shown dialog is enabled. This is consistent with the KCML implementation and is the simplest approach.

The following API's have been added to support modeless dialogues:

```
$DECLARE 'NplShowDialogModeless( INT( ) )
```

The parameter is the dialog ID as for **NplShowDialog**. Any dialog that was previously displayed, (or the NPL main window) remains active and able to accept input focus.

Since modeless dialogues are permitted, a program is required to resolve ambiguities in reported events by calling:

```
$DECLARE 'NplGetDialog
```

This function returns the dialog ID for the last event reported from 'NplKeyin or a KEYIN statement. 'NplKeyin is not affected by keystrokes typed into the NPL main window.

Knowing the dialog ID is necessary if there are duplicated control ID's on the active dialogs.

There is no implied visibility relation between a modal dialog and subsequently displayed modeless dialogs. Modeless dialogs start on top, and whichever dialog has focus will be 'on top' and potentially may obscure all or part of the other windows. If it is required that a dialog always be on top of another dialog, instead of using the 'NplCreateDialog(dialogID) call, the following function should be used.

```
$DECLARE 'NplCreateOwnedDialog(dialogID,ownerID)
```

where ownerID is the dialog ID of a previously created dialog, or 0 to make the dialog owned by the NPL main window. The new dialog will be created so that it is always on top of its owner, and minimizes if the owner does.

#### **\$DECLARE Controls** <sup>[4.21]</sup>

NPL Revision 4.21 introduces support for a new type of custom control (Class "NPLIPIC") which has been defined. This new control is like a "NplPicture" control, but displays an icon file. The name of the icon file can be specified as the "name" field when creating the control. If there is no specific path, the usual windows places are checked if the file is not in the current directory and has no explicit path name. The name can also be passed after the control is created using the same message format as for a NplPicture control (including the 'stretch' option). If there is more than one icon in the file (i.e., for monochrome and color versions, large and small size), the first icon defined in the file is used.

**NOTE:**           **To easily load a bitmap file and display it, use the NplPicture or NplPicButton controls.**

#### **\$DECLARE Functions** <sup>[4.21]</sup>

The following new function returns the window handle of the main NPL window if required.

```
$DECLARE 'NplMainWindow
```

**\$DECLARE /POINTER parameters** [4.30]

\$DECLARE /POINTER parameters (extended)

---

General Form:

```
$DECLARE 'decl-name'[(parm1[,parm2...])][TO basic-type][=dll-spec]
```

Where:

```
decl-name= The name of an external function or procedure
parm=      [/POINTER][TO[RETURN]] basic-type [parameter-name]
basic-type= {INT([-] ) }
              {INT([-]1) }
              {INT([-]2) }
              {INT([-]4) }
              {STR( ) }
              {DIM([size]) }
              {NUM( ) }
dll-spec=  literal-string or alpha-variable containing an alias
```

Line breaks are permitted after each comma in a multiple parameter list.

---

The \$DECLARE statement is used to define the interface to an external subroutine (usually written in C) located in an external .DLL. External .DLLs are supported in the Windows environment, and may not be available on other platforms.

The decl-name identifier specifies the name of the function when accessed from within NPL. It serves as the equivalent of both a PROCEDURE /EXTERNAL declaration and a FUNCTION /EXTERNAL declaration. In order to call the external subroutine, the same syntax is used as for calling a PROCEDURE or evaluating a FUNCTION.

The parameter types information specifies the native data types expected by the subroutine. An INT() type is the equivalent of a C 'unsigned int'. The precision of this type of parameter is platform dependent, but is usually 16 or 32 bits. The INT(1) type is the equivalent of a 8-bit unsigned integer (C 'unsigned char'). The INT(2) type is the equivalent of a 16-bit unsigned integer (C 'unsigned short int') and INT(4) is the equivalent of a 32-bit unsigned integer (C 'unsigned long int'). For each INT() type, the '-' indicator changes the native C type to the 'signed' equivalent.

The STR() type is a null-terminated ASCII string. The DIM() type represents an unformatted data area or structure. If a size is specified in the DIM() type, the string passed on the call must be at least that many characters. The NUM() type represents a floating point number (C 'double').

In addition to this basic type information, each parameter may indicate that information is returned in the parameter using the RETURN or TO RETURN option.

The identifier in each parameter specification is optional, and other than providing documentation for the call, is ignored. The identifier's type should be numeric for INT() and NUM() parameters, and a string for STR() and DIM() parameters.

By default, values are passed using the standard C conventions (INT() values are passed by value, STR() values are passed as pointers). NPL data types are converted or copied to intermediate storage areas before the call. For STR() data types, trailing spaces are removed and a null appended in the copy. When converting to internal data types, range errors are ignored. If RETURN or TO RETURN is specified, after the subroutine returns the value in the intermediate storage area is converted back to an NPL data type. For STR() data types, any characters at or after the first null character are replaced with spaces.

It is also possible to specify a /POINTER option on a parameter, which indicates that the value is passed by reference but without allocating any intermediate storage area. This is typically only useful with DIM() type parameters, which never require any data conversion.

The TO basic-type phrase after the parameter list indicates the native data type returned by the subroutine. If not specified, the default basic-type is INT() (note: C ' unsigned int').

The "=dll-spec" indicates where the external subroutine is located. If no dll-spec is provided, the system searches a standard set of system API libraries using the decl-name identifier. The exact specifics of this search are platform-dependent. If a dll-spec is provided, the string must contain the name of the API call referenced, optionally preceded by the file-name of the DLL in which the API is contained and a '.'.

The exact search procedure used to locate the external subroutine is platform-dependent. On most systems, the drive/directory containing the file need not be specified, and the filename extension is often optional.

When the subroutine is called, NPL parameters are converted to the equivalent C data types. An INT() or NUM() parameter requires a numeric-expression argument, and a DIM() or STR() parameter requires an alpha-variable or literal-string. If the parameter is by reference, the argument must be a non-constant variable of the appropriate type.

There is one exception to this type-checking rule: If a numeric constant appears as an argument to a by-reference STR() or DIM() parameter, the value passed to the external is a reserved pointer (in C, (void\*)(number)). This allows calling some APIs that will accept NULL (0) as a special value for pointer values.

Note that (unlike the BESDK implementation of PROCEDURE/EXTERNAL and FUNCTION/EXTERNAL) there is no way for NPL to check the correctness of the specified \$DECLARE interface. If a subroutine is called with an incorrect number or types of parameters, the most likely result will be an address fault but other unpredictable (and undesirable) behavior is also possible.

In addition, it is the programmer's responsibility to ensure that variables returned via by-reference parameters are allocated sufficient space to store the result. Failure to meet this requirement can also result in address faults or other unpredictable behavior.

The called subroutines should not rely on addresses passed from NPL remaining valid after the call returns.

Functions declared with \$DECLARE are normally searched when the program containing them is resolved. If the function does not exist, an error occurs only if the external is subsequently called when the program executes. A \$OPTIONS bit (byte 47, bit HEX(80)) may be set which causes all \$DECLARE references to nonexistent subroutines to generate an error at resolve time.

\$DECLARE definitions that appear within a PUBLIC/ END PUBLIC section are available to other modules that INCLUDE the defining module.

Defining a \$DECLARE function is equivalent to declaring both a numeric FUNCTION and a PROCEDURE with the same name. Consequently a named PROCEDURE or FUNCTION in the same scope is not permitted.

**Examples:**

```
0010 $DECLARE 'GetDesktopWindow
0020 $DECLARE 'FindWindow(STR() lpszClassName$,STR() lpszWindow$)
      $DECLARE 'SetWindowText(INT() hWnd, STR() lpsz$)
0030 $DECLARE 'NWSendBroadcastMessage(INT() conn,STR() message$,
      INT() connCount, DIM() connList$, RETURN DIM() resultList$)
      ="CALWIN32.NWSendBroadcastMessage"
```



```

0040 $DECLARE 'SQLAllocConnect(INT(4) henv, TO RETURN INT(4) hdbc)
      TO INT(2)='sqlAPIName$("SQLAllocConnect")

0100 $DECLARE 'GetWindowText(INT() hWnd, RETURN STR() Text$, INT()
      TextLength)

0110 $DECLARE 'GetPrivateProfileString(STR() Section$, STR()
      Entry$, STR() Default$, /POINTER DIM() ReturnBuffer$, INT()
      cbReturnBuffer, STR() FileName$) TO INT()

1000 DIM hWnd, _NULL=0
      hWnd='FindWindow(_NULL, "Untitled - Notepad")
      'SetWindowText(hWnd, "Poke Windows To Death")
      nChars='GetPrivateProfileString("general", _NULL, " ",
      Buffer$, LEN(STR(Buffer$)), "rtiwin.ini")

```

### **32-bit \$DECLARE Support** <sup>[5.00]</sup>

\$DECLARE has been updated for NPL Release V RunTimes to support 32-bit calls, however some caution is required:

1. Only 32-bit DLL's can be called via \$DECLARE. In many cases the name of the external library will be different for equivalent 32-bit FUNCTIONS. The standard library list searched if no library name is specified is:

```

NPLKDL32.DLL           ;NPL dialog FUNCTIONS
KERNEL32.DLL
USER32.DLL
GDI32.DLL

```

For other libraries, check with your vendor for the appropriate equivalent name.

It is often possible to write NPL programs that use \$DECLARE that will run in both 16 and 32-bit Windows environments, but extreme care is required.

2. The names of API calls referenced by \$DECLARE are case sensitive under WIN32. So while both the following \$DECLARE calls are legal and work under WIN16, only the first will work under WIN32:

```

$DECLARE 'GetDesktopWindow  ;;correct case, legal WIN16 and WIN32
$DECLARE 'GetDeskTopWindow  ;;incorrect case, works WIN16 only

```

The HEX(80) bit in \$OPTIONS byte 47 can be set to help catch this kind of error, which can otherwise easily be missed during testing. Refer to Section 8.5 for a description of this updated \$OPTIONS byte.

3. The actual names of API calls in the standard DLL's are not the same as documented if any of the function arguments are text (null-terminated) strings. For these API calls, WIN32 defines two entry points, with an 'A' suffix for the ANSI version and a 'W' suffix for the UNICODE version. If an API cannot be located without the suffix, NPL will look for it after adding the 'A' suffix.

If neither form is found, NPL will look for the function with `__stdcall` name decorations, (i.e., `_FunctionName@4` for a function with 4 bytes of parameters). This would be the exported name if no DEF file was used to change the name of an exported function when the DLL was built.

Only FUNCTIONs declared using the `__stdcall` calling conventions can be correctly accessed using \$DECLARE.

4. Many messages related to the Windows API have been revised for WIN32. Specifically, messages that attempted to pack two values (such as a handle and a control ID) into the LPARAM message component have changed. Message cracking and packaging FUNCTIONs for C programs that smooth the transition from 16 to 32-bit code are quite often implemented as C macros rather than as API's and so can't be called via \$DECLARE.
5. The size of an INT() parameter is 16 bits under 16-bit Windows, but 32 bits under WIN32. Caution should be used when declaring FUNCTIONs to use INT(2) for WORD parameters, INT(4) for DWORD parameters and INT() for HANDLE, UINT or WPARAM parameters, if the resulting program must run in both environments.

#### 8.4.4 DATA LOAD BU <sup>[4.30]</sup>

The DATA LOAD BU statement is used to load raw, unformatted contents of the specified file at the byte-address (expr1) into the specified alpha-variable.

The syntax is as follows:

```
DATA LOAD BU T [file-number, ] (expr1[,return-value])alpha-variable  
                [disk-address, ]  
                [<address-var>,]
```

Where:

expr1               = a numeric-expression or alpha-variable.

return-value       = a numeric-receiver or alpha-variable.

alpha-variable     = alpha-variable into which data is to be loaded.

Enough data is read to fill the specified alpha-variable. A buffer of zero bytes performs no access. If the operation requires that data beyond the physical end of the diskimage to be read, an I98 error (Illegal Sector Address or Platter Not Mounted) results and the data in the specified alpha-variable is undefined.

DATA LOAD BU statements are typically used to access data in a diskimage file where sector boundaries are to be ignored. This may be useful to access files with fixed record-lengths, or with variable lengths where the length of the record is known.

Expr1 contains the starting byte number to be loaded. The first byte of the diskimage is number 0. If expr1 is an alpha-variable, the binary value of the first 2 bytes is used.

**NOTE:**            **Use of an alpha-variable for the starting byte number is not advised since this limits access to files with 65535 bytes or less.**

After execution of the statement, the return-value contains the byte number immediately following the last byte accessed by the operation. Provided the device and other parameters are legal, this occurs even if an error occurs on the read operation (for example, if end of file is reached). If return-value is an alpha-variable, the value is contained in the first two bytes in binary.

**NOTE:** An error P51 (Variable or Value Too Short) results if an alpha-variable is specified as the return-variable and the byte number exceeds 65535.

This is an atomic access, and will always use the fast locking mechanism of DATA LOAD BA (whether enabled by \$OPTIONS or not) when available.

If DATA LOAD BU reads less than the specified size (usually due to an EOF), then the 'next byte' receiver variable is set to the next readable byte before the error condition is raised.

DATA LOAD BU is a direct access instruction as opposed to a catalog instruction; therefore the internal device table is not modified by a DATALOAD BU instruction.

Use of FUNCTIONs as arguments in a DATALOAD or DATASAVE statement which does not have the platter \$OPEN may result in a disk operation which is not integral. Any implied lock that RTI would issue against the disk is released for the duration of the function call. If the application depends on this implied lock to maintain data integrity, arguments should be evaluated separately from the DATALOAD/DATASAVE statement.

```

Dim Array1$(22)12
DATA LOAD BU T#1,(X,X)Array1$('FnRslt(K));better would be...
index='FnRslt(K)
DATA LOAD BU T#1,(X,X)Array1$(index)

0010 DATA LOAD BU T#n,(Strtfl+Rnum*Rsize)STR(Rec$, ,Recsize)
      ERROR DO
          EndFile=1
      END DO
0020 DATA LOAD BU T#n, (SeqAccessPtr,NextSeqAccessPtr)Rec$
      ERROR DO
          BytesRead=NextSeqAccessPtr-SeqAccessPtr
      END DO

```

#### Compatibility Issues:

This statement is not supported on a Wang 2200.

DATA LOAD BU may not be supported on all diskimage types. In particular, access to raw diskettes may not be supported. Some operating systems (i.e., VMS) do not support access to data files in this manner.

#### 8.4.5 DATA SAVE BU <sup>[4.30]</sup>

The DATA SAVE BU statement is used to save the contents of the specified alpha-variable or literal-string starting at a specified byte number (expr1).

The syntax is as follows:

```
DATA SAVE BU T [${file-number, } (expr1[,return-value]) data-value  
               [disk-address,]  
               [<address-var>,]
```

Where:

expr1 = a numeric-expression or alpha-variable.

return-value = a numeric-receiver or alpha-variable.

data-value = alpha-variable or literal-string containing the data to be saved.

All data (the defined length) of the alpha-variable is written. A buffer of zero bytes does no access. If the operation would require that data beyond the physical end of the diskimage be written, the file is extended if possible. An error results if the write of the specified size cannot be performed.

DATA SAVE BU statements are typically used to access data in a diskimage file where sector boundaries are to be ignored. This may be useful to access files with fixed record-lengths, or with variable lengths where the length of the record is known.

Expr1 contains the starting byte number to be saved. The first byte of the diskimage is number 0. If expr1 is an alpha-variable, the binary value of the first 2 bytes is used.

**NOTE:** Use of an alpha-variable for the starting byte number is not advised since this limits access to files with 65535 bytes or less.

After execution of the statement, the return-value contains the byte number immediately following the last byte accessed by the operation. Provided the device and other parameters are legal, this occurs even if an error occurs on the save operation (for example, if a volume full condition occurs). If return-value is an alpha-variable, the value is contained in the first two bytes in binary.

**NOTE:** An error P51 (Variable or Value Too Short) results if an alpha-variable is specified as the return-variable and the byte number exceeds 65535.

DATASAVE BU is a direct access instruction as opposed to a catalog instruction; therefore the internal device table is not modified by a DATA SAVE BU instruction.

If DATA SAVE BU writes less than the specified size (due to a volume full) the 'next byte' receiver variable is set to the next writeable byte before the error condition is raised. In many cases any such write generates the error before any bytes are written, but this may depend on the environment.

Use of FUNCTIONs as arguments in a DATALOAD or DATASAVE statement which does not have the platter \$OPEN may result in a disk operation which is not integral. Any implied lock that RTI would issue against the disk is released for the duration of the function call. If the application depends on this implied lock to maintain data integrity, arguments should be evaluated separately from the DATALOAD/DATASAVE statement.

For example:

```
DATA SAVE BU T#1,(X,X)'FunctionResult$(K);better would be...
rec$='FunctionResult$(K)
DATA SAVE BU T#1,(X,X)Rec$

0010 DATA SAVE BU T#n,(Strtfl+Rnum*Rsize)STR(Rec$,,Rsize)
      ERROR DO
      WriteFailed=1
      END DO
0020 DATA SAVE BU T#n,(SeqAccessPtr,SeqAccessPtr)_Rec$
```

#### Compatibility Issues:

This statement is not supported on a Wang 2200.

DATA SAVE BU may not be supported on all diskimage types. In particular, access to raw diskettes may not be supported. Some operating systems (eg VMS) do not support access to data files in this manner.

### 8.4.6 #HDC System Variable [4.30]

A built-in numeric function #HDC returns the handle of the device context (HDC) last opened in a Windows environment.

Knowing this handle can allow applications to use the MS-Windows GDI functions (accessed via \$DECLARE) to draw on and otherwise modify the appearance of printed pages.

PRINT and PRINTUSING statements directed to the printer will print at the current position, using the currently selected font. NPL advances the current horizontal GDI position each time text is printed. At the end of line, the vertical GDI position is advanced by a value appropriate to the currently selected font's text metrics. If this exceeds the page size, a new page is started.

The #HDC value is available only when printing using the windows GDI type \$DEVICE specifications, such as:

```
$DEVICE(/215)=">(?)"
```

The following is a small sample program demonstrating the new #HDC function.

```
0000;HDCTEST
;
;Basic Windows API functions and structures
;
$DECLARE'GetDeviceCaps(INT(),INT(-))="GetDeviceCaps"
DIM _VERTRES=10,_HORZRES=8
RECORD POINT
FIELD point_x=HEX(D002)
FIELD point_y=HEX(D002)
END RECORD
$DECLARE'CreatPen(INT(),INT(),INT())="CreatPen"
$DECLARE'SelctObject(INT(),INT())="SelctObject"
$DECLARE'DeleteObject(INT())="DeleteObject"
DIM _PS_SOLID=0
$DECLARE'MoveToEx(INT(),INT(),INT(), TO RETURN
DIM(#RECORDLENGTH(POINT))="MoveToEx"
$DECLARE'LineTo(INT(),INT(),INT())="LineTo"
FUNCTION'Rgb(R,G,B)
RETURN(R+G*256+B*65536)
END FUNCTION
```

```
0010 ;
      ; A utility function to draw a solid rectangle around
      the page border
      ;
      PROCEDURE 'DrawBoxAroundPage(PenWidth)
      DIM StartPos$#RECORDLENGTH(POINT)
      DIM black,hDc,hPen,hOldPen
      DIM PageHeight,PageWidth
      black='Rgb(0,0,0)
      hDc=#HDC
      IF hDc=0 THEN RETURN
      PageHeight='GetDeviceCaps(hDc,_VERTRES)
      PageWidth='GetDeviceCaps(hDc,_HORZRES)
      hPen='CreatePen(_PS_SOLID,PenWidth,black)
      hOldPen='SelectObject(hDc,hPen)
      'MoveToEx(hDc,0,0,StartPos$)
      'LineTo(hDc,PageWidth-PenWidth,0)
      'LineTo(hDc,PageWidth-PenWidth,PageHht-PenWidth)
      'LineTo(hDc,0,PageHeight-PenWidth)
      'LineTo(hDc,0,0)
      'MoveToEx(hDc,StartPos$.point_x,StartPos$.point_y,Star
          tPos$)
      'SelectObject(hDc,hOldPen)
      'DeleteObject(hPen)
      END PROCEDURE

0020 ;
      ; Example use of the utility function to draw a solid
      rectangle around the user output
      ;
      $DEVICE(/215)=">(?)"
      $OPEN /215
      SELECT PRINT 215
      PRINT
      PRINT TAB(30);"Document Heading"
      ;before closing the document, draw the frame.
      'DrawBoxAroundPage(2)
      $CLOSE /215
```



**NOTE:** The #HDC value should be retrieved immediately after starting the document (with a \$OPEN directed to the printer device address). If no GDI printer is currently open, #HDC returns 0.

Access to pages using GDI functions is not supported for printers accessed with \$DEVICE specifications that use the XLA=Y option. #HDC may return 0 for such printers in some environments, and where it is not 0 the results are unpredictable.

#### 8.4.7 \$IMAGEF and \$IMAGEL <sup>[5.00]</sup>

The \$IMAGEF and \$IMAGEL string functions allow the retrieval of image statements (preceded by a “%” sign) within a program.

The \$IMAGEF and \$IMAGEL functions must be used on the right hand side of a string assignment statement. The syntax is as follows:

```
receiver_var$=$IMAGEF T[platter,] filename$
```

```
receiver_var$=$IMAGEL T[platter,] filename$
```

Where:

platter = any legal device specification (#n, /xxx, /<a\$>)

filename = the name of a program file on the diskimage specified by the device.

The value returned for \$IMAGEF is the text of the first image (%) statement in the named program.

The value returned for \$IMAGEL is the text of the last image (%) statement in the named program.

The text does not include the “%” at the start of the image statement.

This function is specifically implemented to support SPEED/FOURD derived applications. Other applications are advised not to depend on this function.

A number of recoverable errors can occur when evaluating the function:

- If the specified device is invalid, an error P48 can occur on the statement.

- If the named file does not exist or is not a program an error D82 can occur on the statement.
- If the file is not an NPL program based on the label, or cannot be loaded by the current version, an error D88 can occur.
- If the program is protected, an error I96 occurs.
- If the program can be loaded but contains no image statements, an error D88 occurs.
- If the image statement exceeds 256 bytes it may be truncated.

#### 8.4.8 \$LDATE <sup>[4.30]</sup>

The \$LDATE is a special system variable which can be used as a receiver (Form 1) to set the system date or as a function (Form 2) which allows an alpha-variable to be set to the system's date.

Form 1:

*\$LDATE=alpha-expression*

Form 2:

*alpha-receiver=\$LDATE*

When used as a receiver (Form 1) to set the date including century to the specified alpha-expression. The alpha-expression must be in the format "CCYYMMDD" (where CC are the century digits of the year.)

When used as a function (Form 2), the date is returned as an alpha-string, eight characters in length. The first 4 characters are the year including the century, the next two are the month and the last two are the day of the month.

**NOTE:**           **Unlike DATE the "\$" is required.**

The \$LDATE function/ system variable should be used in preference to the DATE function for applications that will be doing date arithmetic where the century value needs to be taken into consideration (eg near year 2000).

Changing either \$LDATE or DATE affects both system variables.

For Example:

```

0010 Today$=$LDATE
0020 M$( )=$LDATE
0030 $LDATE="19971021" ;;October 21, 1997
0040 $LDATE=A$
0050 $LDATE="20010131" ;;Jan 31, 2001

```

**Compatibility Issues:**

Operation of this statement may vary on different hardware versions of NPL. Access privileges may be needed to set the system date under certain operating systems.

**8.4.9 LIST PROC** [4.30]

The LIST PROC command produces a listing of all FUNCTIONS and PROCEDURES referenced by the program in the current LIST module, and on which program lines they are referenced. The syntax is as follows:

```
LIST [title][S] PROC [*] [[low-line],[high-line]][proc-identifier1],[proc-identifier2]
```

Where:

*proc-identifier1*=low range of procedure/function to be displayed  
*proc-identifier2*=high range of procedure/function to be displayed

The listing includes any FUNCTIONS of both numeric and string return-value types, and PROCEDURES (which do not return a value) in the specified range.

The LIST procedure refers to program text in the current list module. This is set to the currently executing module whenever a program HALTs or continues, or when changed using the MODULE command, and can be referenced using LIST DT.

The default for the LIST PROC command lists line-numbers where the specified PROCEDURE(s) and FUNCTION(s) appear. Specifying the "\*" parameter causes the program statement containing the specified PROCEDURE(s) or FUNCTION(s) to be listed in addition to the line number.

Refer to LIST general parameters for details on general parameters for all LIST statements.

The optional name range parameters operate as follows:

- If only proc-identifier1 is specified, LIST PROC output for only that specific procedure is generated.

- If proc-identifier1, (comma) is specified, LIST PROC output for procedures starting with proc-identifier2 in ascending ASCII sequence is generated.
 

```

:LIST PROC (100,200)
:LIST PROC (2000,) 'Get_Status$, 'Current_Status$
:LIST PROC *(,4000), 'Count_Hats
      
```
- If, (comma) proc-identifier2 is specified, LIST PROC output for procedures starting at the lowest ASCII sequence up to and including proc-identifier2 inclusive is generated.
- If proc-identifier1, proc-identifier2 is specified, LIST PROC output for procedures within the range of proc-identifier1 to proc-identifier2 inclusive is generated
- If no range parameters are specified, LIST PROC output is generated for all procedures referenced by the program in the current LIST module.

The optional low/high range parameters are used to specify the range of lines accessed from which references are displayed. These operate as follows:

- If only low-line is specified, only that specific program line is accessed.
- If low-line,(comma) is specified, all program lines starting at low-line are accessed in ascending sequence.
- If ,(comma)high-line is specified, all program lines starting at the lowest ASCII sequence up to and including high-line are accessed.
- If low-line , high-line is specified, all program lines within the range of low-line to high-line inclusive are accessed.
- If no line-numbers are specified, the entire program in the current list module is accessed.

```

:LIST PROC
:LIST PROC 'Did_It, 'Do_It
:LIST PROC * 'Did_It, Do_It$
:LIST PROC 'Update_Activity,
:LIST PROC *,'Rain_Event
      
```

```

0010 ; Mainline
      :PROCEDURE `Set_Position(A$16) /PUBLIC /FORWARD
      :PROCEDURE `Do_It (n) /FORWARD
      :DIM n, Z$16
      : `Do_It(n)
      : `Set_Position (Z$)
0020 END
0030 PROCEDURE `Set_Position (A$16) /BEGINS
      :RETURN
      :END PROCEDURE
      :PROCEDURE `Do_It(n) /BEGINS
      :RETURN
      :END PROCEDURE

:list proc
`Do_It- 0010 0010 0030
`Set_Position
      - 0010 0010 0030

:list proc *
`Do_It -----
0010 ;; PROCEDURE `Do_It(n) /FORWARD
0010 :::: `Do_It(n)
0030 ::: PROCEDURE `Do_It(n) /BEGINS

`Set_Position -----
0010 : PROCEDURE `Set_Position(A$) /PUBLIC /FORWARD
0010 :::: `Set_Position(Z$)
0030 PROCEDURE `Set_Position(A$16) /BEGINS
LIST PROC (cont.)

```

If a range is specified for the LIST PROC statement, the return-type (if any) of the identifier is ignored when determining if the FUNCTION or PROCEDURE is eligible for the listing.

Functions declared via \$DECLARE (which can be called either as a FUNCTION or PROCEDURE) will appear in two separate items in the listing - once for references to the named external as a FUNCTION, and once for references to the named external as a PROCEDURE.

For Example:

```

:LIST PROC 'GetStatus
0010 LIST PROC *

```

```
0010 LIST PROC (8000,)  
0010 LIST PROC 'vn, 'vnZ
```

LIST PUBLIC PROC statement displays PUBLIC FUNCTIONs and PROCEDUREs, ignoring the return type of the named FUNCTIONs.

#### 8.4.10 OPEN # [5.00]

The OPEN #statement is used to open and/or optionally create the specified file on the specified stream. The syntax is as follows:

```
OPEN #stream, filename $ [, [mode $] [, [perms $] [, [buffer size ]]]
```

The stream must have previously been selected to a native operating system directory with the SELECT # statement. Any file previously open on the stream is first closed.

The mode parameter determines how the file is to be treated once opened. The valid settings for the mode parameter are discussed in the "OPEN # file modes" section.

If the mode parameter is not specified then "r+b" is assumed, i.e. for open for both read and write operations in binary mode, error if the file does not exist.

The perms parameter specifies the default permissions to be set on the file when it is created. It is supported only under UNIX operating systems and is specified in standard octal format. The three trailing octal digits represent the user, group and public permissions for the file. Each digit can be set as follows:

```
Read    4  
Write   2  
Execute 1
```

Therefore, to make the file readable and writeable by the owner and the members of the owners group, but only readable to all other users you would set the permissions to "0664". Although the execution bit can be set it is not relevant here.

The buffersize, if specified, is used to fix the size the buffer used for READ #. If not specified then a default size is chosen by the RunTime. Setting an explicit size of zero disables buffering. Byte 3 of \$OPTIONS # can be inspected to determine whether a buffer has been allocated for a particular stream.

For example:

```
OPEN #1, "./DATAFILE", "a+b", "0666"  
ERROR msg$ = "Error opening file"
```

In this example the file "DATAFILE" is opened in the directory named by SELECT #1 (or current directory if #1 is not defined) and will be created if it does not already exist. The permissions specify that the file is readable and writable by all users.

#### OPEN # file modes

<u>Mode</u>	<u>Description</u>
<b>r</b>	Open for read operations only. Writing to the file will give an error. Open will fail if the file does not exist.
<b>w</b>	Open for write operations only. If the file exists then its contents are destroyed.
<b>a</b>	Open file for appending. A new file is created if the file does not exist.
<b>a+</b>	Open for both read and write operations. Error if the file does not exist.
<b>w+</b>	Open for both read and write operations. If the file exists then its contents are destroyed.
<b>a+</b>	Open for reading and appending. Create a new file if the file does not exist.



Optional Translation Flags

- b** Binary mode, no translation is performed.
- t** Text mode, translate NL character HEX(0A) into DOS CR+LF HEX(0D0A)

An optional translation flag of "b" or "t" may be appended to the mode to specify how new-line translation is to be performed under MS-DOS platforms. These flags are ignored under UNIX. The default is "b", or no translation.

Examples:

```
OPEN #30,filename$, mode$
OPEN #stream, "DATA3",,permissions$
OPEN #9, "CUSTFILE", "r+b", "0600", buffer_len
```

**8.4.11 \$OPTIONS#** <sup>[5.00]</sup>

\$OPTIONS # is used to set the port options for streams.

**NOTE:** Do not confuse the "streams" \$OPTIONS# with the system variable \$OPTIONS. Accidentally writing to the wrong variable can produce unpredictable results.

The syntax is as follows:

Form 1. *alpha-variable\$=\$OPTIONS#stream*

Form 2. *\$OPTIONS#stream= alpha-expression*

Port options are set using bits, very similar to the system \$OPTIONS variable. The individual bits and their meaning are shown in the table below:

Byte	Bit	Description
<b>1</b>	HEX(01)	Set to 1: All writes will append automatically to the end of the file.
	HEX(02)	Set to 1: All writes are synchronous and flush the buffer cache.
	HEX(04)	Set to 1: Reads will not block if there are no characters available on a telecommunications device or a pipe.
<b>2</b>	HEX(01)	Set to 1: Enables line oriented read mode. If set READ # will then only read up to the "end-of-line" character specified by byte 5. Setting this bit automatically allocates a buffer.
	HEX(02)	Set to 1: Instructs READ # to ignore the character stored in byte 6.
	HEX(04)	Set to 1: Instructs READ # to truncate the rest of the line if the receiver variable is not large enough to hold the current line.
<b>3</b>	HEX(01)	(Read Only) Set to 1 if a buffer has been allocated.
	HEX(02)	(Read Only) Set to 1 if the ignore character (see byte 6) has been detected.
	HEX(04)	(Read Only) Set to 1 if the "end-of-line" character (see byte 5) has been detected.
	HEX(08)	(Read Only) Set to 1 if the truncation as specified by the HEX(04) bit of byte 2 has taken place.
<b>4</b>		Reserved for future use.
<b>5</b>		Specifies End-of-Line character, used when setting byte 2. Default value is HEX(0A) or LF.
<b>6</b>		Specifies ignore character, used when setting the HEX(02) bit of byte 2. Default value is HEX(0D) or CR.

When modifying values in \$OPTIONS # the recommended procedure is to obtain the current value in a string variable at least 16 bytes long (to allow for future expansion), modify the required bits and then set the value.

For example:

```
DIM O$16
O$=$OPTIONS#3
STR(O$,1,1) = OR HEX(02)
$OPTIONS#3=O$
```

would turn on synchronous writes and

```
DIM O$16
O$=$OPTIONS#3
STR(O$,1,1) = BOOL4 HEX(02)
$OPTIONS#3=O$
```

would turn off synchronous writes.

### 8.4.12 READ # <sup>[5.00]</sup>

The READ # statement is used to read data from the file currently open on the specified stream. The syntax is as follows:

```
receiver_var = READ #stream, alpha_variable
```

The number of bytes copied to the buffer is returned to the receiver\_var. After the operation the current file pointer is advanced by the number of bytes read. If there are not enough bytes to fill the alpha variable then the numeric receiver will be less than the size of the buffer variable and the end of file flag will be set. This can be tested with the END condition in an IF, WHILE or UNTIL statement.

The SEEK # function can be used to change the position of the file pointer prior to a read.

READ # may generate recoverable errors. The reason for the error can be found using the ERR function. Possible errors are:

```
D80 file not open
I92 Timeout error
I96 file has write access only
```

Example:

```
0010 DIM buffer$128
0020 OPEN #1, "datafile", "r+"
0030 WHILE TRUE
        pos = READ #1, buffer$
        IF END THEN BREAK
        GOSUB 'ProcessData
    WEND
```

Other syntax examples:

```
pointer = READ #1,buffer$
WHILE READ #1,buffer$ = LEN(buffer$)
```

**8.4.13 REDIM** <sup>[4.30]</sup>

The REDIM statement is used to redimension the specified array or static variable according to the specified array dimension and string length parameters.

```
REDIM {numeric-array-id(dim1[,dim2])    }
      {alpha-array-id$(dim1[,dim2])length}
      {alpha-id$length                  }
```

Where:

dim1,dim2 = a valid array dimension  
length = a valid string length

REDIM may be used to contract or expand the allocated size of statically allocated arrays and string variables.

There may not be any pending /POINTER or stack references to the referenced variable.

If the new defined size of the variable is reduced from the previous allocated size, excess memory is released for use by other NPL variables or code.

If the new defined size of the variable is expanded, previously unallocated storage is initialized to 0 for numeric arrays, and blanks for string arrays and scalars.

Variable expansion will succeed provided there is sufficient memory to allocate both the old variable and the new larger allocation.

REDIM may be used to replace MAT REDIM where either:

1. The excess space used by the old variable is large, or
2. The variable is an alpha scalar.

REDIM is unlike MAT REDIM in that:

1. This statement can only redimension one variable at a time.
2. For the case of string scalar variables, the expression used to specify the new string length may not start with a left parenthesis “(”. This may

mean the expression will need to start with 0+ to be syntactically correct.

3. There is no implied variable declaration. The variable must be previously declared.

```
0010 REDIM A$ActualSize
0020 REDIM Array$(NewSize)NewLength
0030 REDIM Array(NewSize)
0040 REDIM Matrix(NewRows,NewCols)
0050 REDIM BigString$0+(Table1Sz+Table2Sz)*Elementsz
```

#### 8.4.14 \$RUNDIR <sup>[4.21]</sup>

The directory containing the copy of RTI.EXE being executed can now be determined by NPL programs using the new system variable, \$RUNDIR. This variable is only legal on the right side of a string assignment (LET) statement. For example:

```
0010 ;P_RUNDIR
      A$=$RUNDIR
      PRINT A$
```

For example:

The current directory is F:\BASIC2C and contains RTI.EXE:

```
RTI p_rundir          displays:      F:\BASIC2C\
```

The current directory is C:\APP and the path locates RTI.EXE in C:\NPL:

```
RTI p_rundir          displays:      C:\NPL\
```

The current directory is C:\APP and rti is specified explicitly:

```
D:\TOOLS\RTI p_rundir displays:      D:\TOOLS\
```

**8.4.15 SEEK #** <sup>[5.00]</sup>

The SEEK # function repositions the current file pointer for file open on the specified stream relative to the beginning, end or for an increment from the current position.

The syntax is as follows:

```
SEEK #stream [ [,{BEG}][,numexp] ]
              [ [ {FOR} ] ]
              [ [ {END} ] ]
```

Where:

*BEG* = Parameter to seek from beginning of file.

*FOR* = Parameter to seek for an increment from the current position.

*END* = Parameter to seek from the end of file.

Specifying just an offset is shorthand for BEG. The new offset is returned by the function. By not specifying any offset the current position in the file may be determined.

An error will cause SEEK # to return the value of -1.

A negative seek expression is valid with FOR and END, but will give an error P34 for BEG.

The following program shows the result of several SEEK # functions assuming that the functions were executed in the order of appearance. The return values are shown on the right hand side.

```
0000 DIM file$1000
      OPEN #1, "file1", "w+"
      ret = WRITE #1, file$
      offset = SEEK #1, BEG           : ; 0
      offset = SEEK #1, BEG, 100     : ; 100
      offset = SEEK #1, END, -100    : ; 900
      offset = SEEK #1, 500          : ; 500
      offset = SEEK #1, FOR, 100     : ; 600
      offset = SEEK #1              : ; 600
      CLOSE #1
```

**8.4.16 SELECT DISK/ #** <sup>[5.00]</sup>

The SELECT DISK/# command is used to set or change a stream entry in the device table.

The syntax is as follows:

Form 1.            **SELECT** {*DISK* }{/devaddr        }  
                               {#stream}{"literal\_string" }  
                               {<alpha-value>    }

Form 2.            *alpha-receiver* = \$SELECT( {#stream} )  
   {*DISK* }

SELECT DISK can only change the value of #stream 0, which is used as the default LIST DC, LOAD, SAVE device. Specifying a #stream allocates the specified device to the specified #stream. Using a #stream of zero is the same as using the DISK parameter.

A native operating system directory path may be selected by specifying a literal string in place of the device address. Device addresses or path names can be assigned to alpha variables, but must be surrounded by angle brackets '<>'. For example:

```
SELECT #22 <file_path$>
```

The \$SELECT function can also be used to examine the current filename or device name currently assigned, for example:

```
device_entry$(1) = $SELECT(#1)
```

This would place the currently selected device address to stream '#1' into the variable device\_entry\$(1). If the stream had previously been selected to a native directory, the directory name would then be placed into device\_entry\$(1).

Other syntax examples:

```
SELECT DISK /D10  
SELECT #1<"D:\NPL\LIB">, #2 /D99  
address$ = $SELECT(#27)
```

For literals, the syntax with and without <> are equivalent. LIST displays the <> form.



```
SELECT #1 "C:\TEMP"
```

is the same as:

```
SELECT #1<"C:\TEMP">
```

On 32-bit platforms, the default maximum of open file entries has been increased to 256. This effectively means that a `SELECT #stream` (where `stream > 15`) to increase the number of open files is no longer required on these platforms.

#### 8.4.17 \$USER <sup>[4.22]</sup>

The `$USER` statement allows developers easy access to user application counting logic.

Syntax is:

```
$USER result-num-var, serial-var$, app-name$, opcode-num-expr
```

Where:

Result-num-var is a numeric variable or array element.

Serial-var\$ must be a modifiable alpha variable.

App-name\$ is an alpha literal or string variable.

Opcode-num-expr is a numeric expression containing an operation code.

Values for the operation code must be in the range 0-65535. The value of the opcode determines the uses of the other parameters.

Currently supported operations are:

- 0 - Count other registered network users of application app-name\$, with serial number serial\$ and return the number in result-num-var. Use this call only if you are not currently logged in.

**NOTE:** This is a relatively long operation (up to 30 seconds) on a peer network and should be used sparingly.

- 1 - Register (log in) as user of application app-name\$, with serial number serial\$ and return the registration handle in result-num-var.
- 2 - Deregister (log out) as user on application app-name\$, with serial number serial\$. Result-num-var must contain the registration handle returned by opcode 2.

All other values are reserved for future use and generate a recoverable error 306.

For all the above calls, Serial\$ must be non-blank and contain a serial number of up to 9 digits. Leading zeros are ignored. Serial numbers that do not meet these requirements will generate an X75 error. App-name\$ should not contain a HEX(00) character and should be fairly short (less than 128 characters is recommended). Trailing spaces on both app-name\$ and serial\$ are ignored. It is the application's responsibility to log out once it has logged in, no matter how the program exits. To ensure compliance with this requirement, it is recommended to place any required logout code in the /EXIT routine of a module which otherwise remains resident for the duration of the application.

As currently implemented, the user counting and login/logout logic will only provide meaningful results in either:

1. a Novell network environment
2. a peer network environment where NIAKNETB is loaded or NIAKNETW is enabled.

Due to memory constraints the calls are only supported in the 386/DOS-Extender and all MS-Windows versions (not by the MS-DOS real mode version).

In multitasking environments such as Windows, the user count is reported as it is for the RunTime, (i.e. 1 count for multiple RTI windows and 1 for each application running in a DOS box).

In peer environments a very limited number (2 under DOS, 10 under Windows) of combinations of application names and serial numbers may be logged in at any time on a specific machine. Developers are requested to use this facility sparingly.

**8.4.18 WRITE #** <sup>[5.00]</sup>

The WRITE # function writes data to the file currently open on the specified stream. The syntax is as follows:

```
receiver_var = WRITE #stream, alpha_value$
```

The number of bytes written to the file is returned to the receiver\_var. After the write the current file pointer is advanced by the number of bytes written. The entire contents of the buffer variable are written including any trailing blanks.

The SEEK # statement can be used to change the position of the file pointer prior to a write. WRITE # will automatically extend the file to the new position if the pointer is set to a position beyond the end.

WRITE # may generate recoverable errors. The reason for the error can be found using the ERR function. Possible errors are:

```
D80 file not open  
I96 file has read access only
```

Some syntax examples:

```
0010 bytes_out = WRITE #1,RecordBuffer$  
0020 ERROR DO  
        bytes_out = -1  
      END DO  
      IF bytes_out <> -1  
        GOSUB `ProcessNextRecord  
      END IF
```

## 8.5 System Variable Additions and Changes

This section outlines changes made to specific \$MACHINE and \$OPTIONS bytes within NPL. Where applicable, some information may replace previously documented bytes. For all other bytes previously defined, refer to the NPL Release IV Statements Guide.

### 8.5.1 \$MACHINE Changes

#### **Byte 2 - Hardware Manufacturer Code** [4.30]

On the MS-Windows RunTimes, \$MACHINE byte 2 now contains a value that can be used to distinguish between Windows 95 and Windows NT environments. On previous releases this value (usually) contained HEX(FF). On the new release the value is:

HEX(01) for Win95 platforms  
HEX(02) for WinNT platforms  
HEX(00) for other Windows platforms.

#### **Byte 30- Reserved** [4.10]

This byte is used by NPL internally and should not be used by NPL developers.

#### **Byte 31 -Reserved** [4.10]

This byte is used by NPL internally and should not be used by NPL developers.

**Byte 32 - RunTime Environment Detected** [4.10]

Byte 32 of \$MACHINE is used to record the networking operating environment detected by NPL at startup of the RunTime. Bits in byte 32 of \$MACHINE have the following meanings:

Byte 32	HEX(01) - bit = 0	Security was passed from a non-local drive or from a network-wide once-per-day security mechanism.
	HEX(01) - bit = 1	Security was passed from a local drive (i.e., local drive or Gold Key diskette). When no NIAKNETB or Novell NetWare network is detected, it enables full file sharing capabilities that would otherwise not be available. In addition, this would indicate that the current user has not been charged against the network-wide user limit (only if a Novell NetWare network is detected).
	HEX(02) - bit = 0	The NIAKNETB TSR was not detected. Security passed from a local drive (i.e., local drive or Gold Key diskette). This does not charge against the network-wide user limit.
	HEX(02) - bit = 1	The NIAKNETB TSR was detected.
	HEX(04) - bit = 0	A Novell NetWare network environment was not detected.
	HEX(04) - bit = 1	A Novell NetWare network environment was detected.
	HEX(08) - bit = 0	This is a NetBIOS RunTime.
	HEX(08) - bit = 1	This is a Niakwa Networks RunTime.

All other bits are reserved for future use. The default value is 0.

**Byte 33 - File Locking and User Count** <sup>[4.10]</sup>

The high order nibble of \$MACHINE byte 33 is used to record the user count method used by NPL for the current user. The low order nibble of byte 33 is used to record the locking mechanism used by NPL for the current user.

Byte 33	HEX(0F) - bit = 00	Network files are not shareable.
	HEX(0F) - bit = 01	Network files are locked by SHARE.
	HEX(0F) - bit = 02	Network files are locked by Novell NetWare calls.
	HEX(F0) - bit = 00	Uncounted RunTime
	HEX(F0) - bit = 10	Counted from local fingerprint
	HEX(F0) - bit = 20	Counted using NIAKNETB
	HEX(F0) - bit = 30	Counted using a Novell NetWare semaphore
	HEX(F0) - bit = 40	Counted using IBM AS/400 security

**Byte 34 - Reserved**

This byte is used by NPL internally and should not be used by NPL developers.

**Bytes 35 through 37 - Expiration Date** <sup>[4.20]</sup>

\$MACHINE bytes 35, 36 and 37 can be used to determine the expiration date of a RunTime. If no expiration date exists (by default) then all bytes contain zero.

Byte 35	Indicates the expiration Year
Byte 36	Indicates the expiration Month
Byte 37	Indicates the expiration Day

**Byte 38 through Byte 40 - Reserved**

These bytes are used by NPL internally and should not be used by NPL developers.

**Bytes 41 through 44 - Mouse Position** <sup>[4.21]</sup>

Mouse coordinate position information is now available in bytes 41-44 of \$MACHINE, when running in /G mode using the 386/DOS-Extender or MS-Windows version of the NPL RunTime. This option is not implemented in real-mode DOS due to memory constraints.

Byte 41	Y (vertical) coordinate of mouse (high order)
Byte 42	Y (vertical) coordinate of mouse (low order)
Byte 43	X (horizontal) coordinate of mouse (high order)
Byte 44	X (horizontal) coordinate of mouse (low order)

**NOTE:** A value of HEX(FF) indicates either the NPL version or platform does not support reporting mouse coordinates.

After receiving a mouse click or drag key, the following example will allow an NPL program to determine the mouse coordinates at the time of the event:

```
DIM M$64
M$=$MACHINE & ALL(FF)    ;;& ALL(FF) handles older versions

Y=VAL(STR(M$,41),2)      ;;vertical coordinate
X=VAL(STR(M$,43),2)      ;;horizontal coordinate
```

A value of 65535 would indicate that mouse pixel support is not available.

### 8.5.2 \$OPTIONS Additions and Enhancements

**Byte 14 - \$BREAK Performance** <sup>[4.10]</sup>

\$OPTIONS byte 14 is no longer ignored under MS-DOS. In a multi-session environment (e.g., MS-DOS tasks under MS- Windows) setting this byte to HEX(01) improves interleaved access to files. The default remains HEX(00).

**Byte 20 - Improved Performance of Remote Terminals** [4.30]

Performance of remote terminals has been improved. Also available on this release is a “full screen refresh” option which may be applied using \$OPTIONS Byte 20 by using 2 new bits HEX(02) and HEX(04). Bits in byte 20 have the following meaning.

Bit Position	Bit Value	Description
HEX(01)	0	(DOS) Use text output to console to allow pausing. (XON/XOFF enabled) (Other) Default. Ignored
	1	(DOS) Default. Use binary output to console for increased speed. (XON/XOFF disabled) (Other) Ignored
HEX(02)	0	No effect
	1	Delay screen output for all PRINT statements until next KEYIN (any form), then clear this option.
HEX(04)	0	No effect
	1	Delay screen output for all PRINT statements until next KEYIN (any form). Do not clear this option.

**NOTE:** All values in \$OPTIONS byte 20 are ignored when using direct video (text modes under DOS without /R, and Windows).

The HEX(02) and HEX(04) options are intended for screen-painting applications operating in remote terminal environments. Often, such applications can benefit from holding back actual output to the display until the screen painting operation is complete.

Completion of the paint operation is assumed to be indicated by the next statement (INPUT/ STOP/ KEYIN) which waits for operator entry. At this point the net effect of the screen changes is sent to the terminal.

In the case of the HEX(02) option bit, the effect is temporary. This option might be set by a routine before it starts a major update operation. The HEX(02) bit is turned off automatically once the screen changes are displayed.



In the case of the HEX(04) option bit, the effect is permanent. This option might be set by a browser application that typically does large amounts of screen updating with relatively little operator entry.

On the windows version, operations other than INPUT/ STOP/ KEYIN may refresh the screen also, if the environment requires it.

**Byte 33 - Suppress HELP Processor Options** [4.20]

The following new bits in \$OPTIONS byte 33 (HEX(10) and HEX(20)) are defined to suppress all HELP display options when help is invoked from an INPUT, LINPUT or KEYIN command.

The intention here is to restrict operators from going off into 'Display' or other options that might be confusing when the HELP key is pressed accidentally.

**NOTE:**        **HEX(01), HEX(02), HEX(04) and HEX(08) bits have been previously defined. Refer to the NPL Release IV Statements Guide, \$OPTIONS, for a complete description of these bits.**

Byte 33	HEX(10) bit = 0	No special suppression of >Close= option in system menu (Windows version only).
	HEX(10) bit = 1	Disables >Close= option in system menu (Windows version only) when set. Note that if the >Kill Basic2C= option is disabled, this will also disable the Windows >Close= option.
	HEX(20) bit = 0	No special suppression of HELP Display options occur when HELP is invoked under program control from a INPUT, LINPUT or KEYIN command.
	HEX(20) bit = 1	All HELP Display options except "Leave Help" are disabled when HELP is invoked under program control from a INPUT, LINPUT or KEYIN command.

**Byte 38 - Enhanced Required Variable Declaration** [4.30]

On previous releases the following values were supported:

- HEX(00) - Undeclared numeric and string scalars are declared by default.
- HEX(01) - Undeclared numeric and string scalars are flagged with an error P55 at resolve time.

The above values are still supported. In addition, a non-zero value the following bits allow for exceptions:

- |         |         |   |
|---------|---------|---|
| Byte 38 | HEX(02) | undeclared numeric scalars with short (pre-Rev4) variable names are declared by default |
|         | HEX(04) | undeclared string scalars with short (pre-Rev4) variable names are declared by default  |

Either or both of the HEX(02) or HEX(04) bits may be set. Any non-zero value is treated by older releases as if it were HEX(01), and setting either bit always checks all long variable names.

Example:

```
0010 ;OLDCODE
0020 A=1
0030 B$="A"
0040 ;NEW CODE
0050 DIM Apple,Banana$3
0060 Appel=3           ;;note spelling error
```

Value of \$OPTIONS Byte 38	Undeclared variables in the code	Error Generated
HEX(00)	A,B\$16,Appel	None
HEX(01)	None	20 (A)
HEX(03)	A	30 (B\$)
HEX(07)	A,B\$16	60 (Appel)

**NOTE:** In all cases, default declarations are never permitted inside the body of a FUNCTION or PROCEDURE.

**Byte 39 - File Locking and Sharing** [4.21]

The NPL MS-DOS, MS-Windows and 386/DOS-Extender versions now support the HEX(01) bit in \$OPTIONS byte 39 that was previously supported only under UNIX. This option affects both local drives (with SHARE type locks) and network drives, on Novell NetWare or NetBIOS based networks.

**NOTE: Developers can expect a performance benefit in all environments by enabling this option.**

When the HEX(01) bit of \$OPTIONS byte 39 is set to 0 (the default value), NPL issues an implied \$OPEN for the duration of each disk statement, if an explicit \$OPEN has not already been issued. The purpose of this implied \$OPEN is to ensure that if any other users have issued an explicit \$OPEN against the diskimage, the lock is respected (the platter is not accessed until the lock is released).

When the HEX(01) bit of \$OPTIONS byte 39 is set to 1, NPL will allow atomic disk statements (i.e., those completed with a single read or write operation) to access a diskimage file without issuing an implied \$OPEN. If other users have issued an explicit \$OPEN against the diskimage, the platter is not accessed until the lock is released.

The following disk statements are always atomic:

DATA LOAD BA	
DATA SAVE BA	
DATA LOAD BM	(if buffer size is a multiple of 256 bytes)
DATA SAVE BM	(if buffer size is a multiple of 256 bytes)
LIMITS INDEX	

**NOTE: The net effect of this option is to improve overall concurrent access to shared diskimage files (especially in networked environments where file locking operations are typically slow), if the majority of access to these diskimage files use the above atomic operations.**

Under MS-DOS, MS-Windows and 386/DOS Extender, requests to read or write data on a file that is currently locked returns an error. If this occurs in any situation where the current task has suppressed an implied lock, NPL assumes the error could be due to a lock on the data being held by another user or task. NPL then issues the suppressed implied lock and retries the operation. If an error still occurs, the error is reported (as on previous releases), otherwise no error is reported and the program continues.

**NOTE:**           **The size of the implied lock issued in the event of a retry is affected by the HEX(02) bit of \$OPTIONS bit 39. If the HEX(02) bit is 0, a full file lock is issued. When the HEX(02) bit is 1, only the area of the file being read by the statement is locked.**

For maximum concurrence both the HEX(01) and HEX(02) bits should be set.

Under UNIX, if an explicit \$OPEN is active on a file, any requests to read or write data in the file automatically wait for the lock to be released.

The following describes all possible values of \$OPTIONS byte 39. Values in this byte are treated as bit flags controlling special optimizations for implied \$OPEN operations.

Bit Position	Bit Value	Description
HEX(01)	0	No special avoidance of implied \$OPENs.
	1	RTI suppresses implied \$OPENs on atomic DATALOAD/DATASAVE BA/BM operations. If the affected area is currently locked by another task:  Under UNIX, NPL waits for the lock to be released.  Under MS-DOS, MS-Windows and 386/DOS Extender, NPL receives an error, suppresses it and issues a retry. If an error is still encountered, it is reported as in previous releases.
HEX(02)	0	No special handling of implied \$OPENs.
	1	The RunTime issues limited file locks instead of implied \$OPENs for atomic DATALOAD/DATASAVE BA/BM operations.  On UNIX systems with both the HEX(01) and HEX(02) bit set, the HEX(01) bit takes precedence.
HEX(04)	0	No special handling of implied \$OPENs.
	1	The RunTime issues limited file locks instead of implied \$OPENs for atomic DATALOAD/DATASAVE DC/DA operations.
Other Bits	0	Reserved. Default 0 and should not be used.

**NOTE:** Previous releases of NPL will treat all values the same as HEX(00) on all MS-DOS based operating environments. The HEX(01) bit of \$OPTIONS byte 39 was previously implemented on all UNIX based operating environments.

Use of the HEX(04) option can substantially increase the ability to access a diskimage concurrently by many users, provided the accessed areas do not overlap. This option should only

be set if DATALOAD DC/DA are always used by the application to access only single data records (save with one DATASAVE statement), starting at the first sector of the record.

**WARNING:** Reading multiple consecutive records with a single DATALOAD DC/DA statement with the HEX(04) bit set, without first locking the entire diskimage (with \$OPEN), can cause access failures on some operating systems, notably MS-DOS based systems. The errors may be reported as errors on the statement or as a “Device not ready” screen. Because the failure may only occur when records other than the first are concurrently accessed (and locked) by another user, this problem may not appear when testing the application in a single user environment.

The following new bit in \$OPTIONS byte 39 is intended to suppress the RunTime’s default logic of mixed network support as discussed in Section 3.5.

Byte 39	HEX(08) bit = 0	Default. NPL attempts to automatically handle mixed network support by attempting to determine if a file is on a NetWare server or local.
	HEX(08) bit = 1	NPL suppresses mixed network support logic.
	Other bits = 0	All other bits reserved.

**Byte 42 - Restrict RETURN ERROR** <sup>[4.10]</sup>

The following new bits in \$OPTIONS byte 42 are defined for restricting the use of statements which would cause forced RETURN ERROR(x). These new bits are only applicable for operations due to attempts to clear the return stack during a callback from an external routine or functions called while program resolution is in progress.

**NOTE:**        **HEX(01) and HEX(02) bits have been previously assigned. Refer to the NPL Release IV Statements Guide, \$OPTIONS, for a complete description of these bits.**

Byte 42	HEX(04) bit = 0	Statements which attempt to clear the return stack frame during a callback from an external library generate a forced RETURN ERROR(251) to the external caller
	HEX(04) bit = 1	Statements which attempt to clear the return stack frame from a callback from an external library generate an error 264. The return stack is not affected
	HEX(08) bit = 0	Statements which attempt to clear the return stack frame during a function call used in a declaration (while a RUN statement is in progress) generate a forced RETURN ERROR(262) to the declaration.
	HEX(08) bit = 1	Statements which attempt to clear the return stack frame from a callback from an external library generate an error 265. The return stack is not affected.

**HINT:** These values may be set by developers to ensure that unstructured exits are handled differently (when the forced RETURN ERROR action is inappropriate).

**NOTE:**        **If either of these options is set and the function call encounters an error, it is necessary to inspect the current call stack (using LIST STACK) and provide appropriate return values for functions using immediate mode RETURN(x) or RETURN ERROR(x) commands. This is true until eventual program execution returns to the external callback or declaration expression.**

Statements which would clear the return stack (e.g., CLEAR P, program modifications, LOAD statements or RUN statements) are effectively disabled by these options until the special call conditions are completed.

In addition to manually setting these \$OPTION bits, several function calls have been bundled with the NPL Gateway to MS-Windows API Library, which can be used to set or clear these bits whenever necessary.

**Byte 45 - Enhanced Line Recall Operation** [4.30]

This new \$OPTIONS bit changes the behavior of the line recall operation to simplify statement cutting and pasting on long lines.

HEX(20)	0 - Default	No effect
	1	Line recall operation adds a return graphic to the end of any recalled line.
HEX(40)	0 - Default	Display of abbreviated long lines with colons between each statement.
	1	Display of abbreviated long lines with comma and statement number (no colons.)

This simplifies selecting the last statement on the line with a mouse, and when cut out, the selected text will include a return graphic.

In addition, new statement additions to the end of the line are easier to perform, since only cursor down operations are required to position to the end of the line (or the cursor can be positioned directly with the mouse). It is no longer necessary to insert the return graphic at the end.

If \$OPTIONS byte 45 bit HEX(40) is set, when NPL displays abbreviated forms of long program lines (showing only one statement on the line), a comma and a statement number are displayed before the statement instead of the long list of colons displayed by previous releases.

The first statement on the line is 1 (to conform with the RUN line[,statement] command). The first and second statement still display in the old format, which is briefer.



This new display format affects:

- LIST statements with the '\*' option which display program statements.
- STEP displays of the next executable statement.
- LIST STACK displays of FOR and GOSUB statements.

For example:

```
:1000 ;Program
      ;This is the second statement
      FOR X=1 TO 2: NEXT X
      ;This is the fifth statement
:STEP
:RUN
1000 ;Program:...
;(-EXECUTE-)
1000,2: ;This is the second statement
;(-EXECUTE-)
1000,3: FOR X=1 TO 2: ...
;(-EXECUTE-)
1000,4: NEXT X: ...
;(-EXECUTE-)
1000,4: NEXT X: ...
;(-EXECUTE-)
1000,5: ;This is the fifth statement
```

The default \$OPTIONS value leaves the listing in the old format.

**Byte 46 - Enhanced HELP Processing** [4.10]

This byte is documented in the NPL MS-DOS Addendum but not in the NPL Release IV Statement's Guide. The description is repeated here as a convenience to developers.

This \$OPTIONS byte allows customization of the HELP processor's use of keys. The allowed values of byte 46 are:

Byte 46	HEX(00)	RETURN keys advances to next HELP field (default)
	HEX(01)	RETURN key operates like EXECUTE on the current HELP field.

Other values are reserved and should not be changed.

**Byte 47 - MS-Windows Options** [4.10]

The following bits in \$OPTIONS Byte 47 are defined for controlling the functionality of the NPL MS-Windows RunTime.

Byte 47	HEX(01) - bit = 0	RTIWIN may poll message queue to check for HALT key and allow other tasks to run during extended operations.
	HEX(01) - bit = 1	RTIWIN never polls message queue to check for HALT key or allow other tasks to run during extended operations.
	HEX(02) - bit = 0	The RTIWIN output for Immediate Mode operations can appear in the same window as the application, if the current DebugWindow option is disabled.
	HEX(02) - bit = 1	The RTIWIN output for Immediate Mode operations should always appear in a separate DebugWindow. If the Debug Window option is off, the Debug Window may not appear until the first Immediate Mode operation occurs.
	HEX(04) - bit = 0	If a STOP statement occurs while processing an intertask message dispatched by SendMessage, a warning indicating a ReplyMessage() / Yield() has been issued to avoid a deadlock condition is displayed. MS-Windows may hang at this point.
	HEX(04) - bit = 1	If a STOP statement occurs while processing an intertask message dispatched by SendMessage, NPL responds as if an EXECUTE key had been pressed to continue execution. Warning message #4, "STOP during InSend Message. Automatic execute option used." is displayed.
	HEX(08) - bit = 0	The "Debug" option of the main window is enabled.
	HEX(08) - bit = 1	The "Debug" option of the main window is disabled
	HEX(10) - bit = 0	NPL's internal memory reorganization is enabled
	HEX(10) - bit = 1	NPL's internal memory reorganization is disabled.

**NOTE:** When the HEX(01)-bit option is set, the NPL application runs exclusively between operations which wait for input (KEYIN, INPUT, LINPUT, STOP) or elapsed time (delays due to SELECT P). It normally cannot be HALTed unless a separate debug window is active. Menu options, including windows resizing, etc., are disabled between these operations.

**NOTE:** The HEX(04) HEX(08) and HEX(10) bits may be set by event-driven applications which use the NPL Gateway to MS-Windows API library and have their own

fetch/dispatch loop. The task switcher mechanism may use intertask SendMessage calls to activate applications. Caution must be exercised if STOP statements are embedded in NPL code that might be called during processing of the following message values.

Message ID	Message Number
	WM_ACTIVATE
6	WM_SETFOCUS
7	WM_KILLFOCUS
8	WM_GETTEXT
13	WM_ERASEBKGND
20	WM_ACTIVATEAPP
28	WM_ICONERASEBKGND
39	WM_?? /*undocumented*/
51	WM_WINDOWPOSCHANGING
70	WM_WINDOWPOSCHANGED
71	WM_NCPAINT
133	WM_NCACTIVATE
134	WM_?? /*undocumented*/
136	WM_?? /*undocumented*/
783	

All other bits are reserved. The default value is 0.

**NOTE:** NPL's internal memory reorganization, which can cause addresses returned by the NPL Gateway to MS- Windows API calls 'CastLPSTR() and 'CastLPCSTR() to become invalid before they are used, may be explicitly disabled by using the HEX(10) option bit.

In addition to manually setting these \$OPTION bits, several function calls have been bundled with the NPL Gateway to MS-Windows API Library, which can be used to set or clear these bits

whenever necessary. Refer to the NPL Gateway to MS-Windows API Programers Guide for details.

The following new bits HEX(20) in \$OPTIONS byte 47 permits bypassing the standard Windows keyboard driver translation logic for the numeric keypad. This allows programs to remap the decimal or other keys on the keypad to other values, without affecting the standard part of the keyboard.

Also, if the HEX(80) bit is set, references to \$DECLARE functions which cannot be located at resolve time are trapped as errors, and generate an error 243:

(No EXTERNAL function is defined with this name and these parameters.)

Byte 47	HEX(20) = 0	The Windows keyboard driver is allowed to translate the numeric keypad keys as on all previous releases.
	HEX(20) = 1	The 16 keypad keys ( 0-9, *, +, comma (if any), -, decimal, and slash) will be reported as complex key codes HEX ( 35-44) respectively. The ENTER and NUM-LOCK keys are not affected.
	HEX(80) =0 - (Default)	No effect
	HEX(80) =1	\$DECLARE functions not located at resolve time are trapped and generate an error 243.
	Other bits = 0	Other bits are reserved

Setting this option during testing can avoid shipping programs that use invalid \$DECLARE statements in infrequently used code.

This option should only be set for operation in a MS-Windows environment since all \$DECLARE functions are non-existent in other environments.

**Byte 48 - Enhanced Print Device Handling** [4.10.23]

The following defines new \$OPTIONS byte 48 settings.

Bit Position	Bit Value	Description
HEX(01)	0	Default. NPL automatically closes all print class \$DEVICES when the HELP key is pressed or \$SHELL commands are executed.
	1	NPL does not close print class \$DEVICES that refer to on-line devices, such as LPT1, when the HELP key is pressed or \$SHELL commands are executed.
HEX(02)	0	Default. NPL does not explicitly flush Novell print CAPTUREs when the related print class device is logically closed. If the CAPTURE is configured with NoAutoEndCap, (NA) output is not flushed by a logical close.
	1	NPL flushes Novell print CAPTUREs configured with NoAutoEndCap (NA) when the related print class device is logically closed.
HEX(04)	0	Default. NPL interprets all SHARE return codes.
	1	NPL ignores non-recoverable SHARE return codes when SHARE locks are directed to print class devices in environments that do not support SHARE locks directed at device files (i.e., an OS/2 DOS box). In such a situation, NPL behaves as if the lock were granted, although the printer is not locked.
Other Bits	0	Reserved. Default 0 and should not be used.

**NOTE:** Windows GDI spooler-pipes, UNIX-style pipes and standard-handle files are exceptions.

A logical close occurs on print class devices any time it has been accessed and one of the following conditions occur:

- A \$CLOSE is issued to the device.
- Option bit HEX(01) is not set and, the HELP key is pressed or a \$SHELL statement is executed (resulting in an implied \$CLOSE).
- A \$DEVICE() clause is reassigned (resulting in an implied \$CLOSE).
- A program exits using \$END (resulting in an implied \$CLOSE).

Developers may wish to set the HEX(01) bit when device support is directed through an otherwise transparent spooling mechanism, (i.e., Novell NetWare's CAPTURE command) to prevent the implied \$CLOSE on HELP and \$SHELL commands from interrupting a spool entry.

If more explicit control of spooling under NetWare is required, applications may use the NoAutoEndCap (NA) option of CAPTURE, and set the HEX(02) bit only when logical close operations should explicitly flush the current CAPTURE into the print queue.

Previous releases of NPL would by default issue a file lock request to SHARE subsystems if a file was a device file (i.e., LPT1, COM1, etc.). While this logic was fairly reliable running from a DOS box under MS-Windows for Workgroups, Windows 95 no longer provides reliable file lock error codes that would allow NPL to distinguish between multiple users. As a result of this functionality, a new bit in \$OPTIONS byte 48 has been enabled with the following options:

Bit Position	Bit Value	Description
HEX(08)	0	NPL uses SHARE to lock device files.
	1	Default. NPL does not ask SHARE to lock device files. Instead, NPL acts as if the file lock has been granted.

**NOTE:** To avoid the growing number of device false lock problems reported under NetBIOS environments, the HEX(08) bit is enabled by default.

**Byte 49 - COM Variable Resolution** [4.10.23]

Intermediate memory requirements necessary for INCLUDED modules during program overlays can be minimized using the new \$OPTIONS Byte 49. Bits in byte 49 have the following meaning.

Bit Position	Bit Value	Description
HEX(01)	0	Default. Modules that have no COM variables and are referenced by an INCLUDE are discarded at the end of the resolution of the root module.
	1	Modules that have no COM variables and are not referenced by INCLUDE statements in places other than the root module are discarded after root module text is cleared and before loading new program text into the root module
Other Bits	0	Reserved. Default 0 and should not be used.

When the set of required library modules is very different before and after a program overlay (i.e., a LOAD T occurs), in some environments the intermediate memory required to hold both sets of library modules may not be available. This can occur although there is sufficient memory to hold the required modules after unreferenced modules are discarded. Previous versions of NPL required that such a situation be handled specially by overlays using an intermediate module. This module contained no INCLUDE statements (or, optimally, only INCLUDE statements common to the before and after overlay programs), to allow unreferenced modules to be discarded.

Now, the HEX(01) option provides the equivalent of doing an overlay using an intermediate module that has no INCLUDE statements, without doing the overlay.

While the HEX(01) option requires the minimum available memory to perform an overlay, it will also require the maximum execution time since any modules that would otherwise be retained across the overlay were deresolved, discarded, reloaded and resolved.

**NOTE:**        **To obtain the best performance, this option should be set only in environments where memory limitations are severe (i.e., MS-DOS real mode).**

Obtaining better overlay performance (to avoid discarding and reloading modules used before and after the overlay) when the option is set requires additional programming. The simplest method is to declare a COM variable in modules that are typically retained throughout an overlay and provide a PUBLIC function that changes (using COM CLEAR) the status of this variable to a non-COM. This function must be called before an overlay when the module is no longer needed.



**NOTE:** It is possible that use of this option may affect the operation of some programs, since public and static variables in library modules, that would otherwise remain resident, are reinitialized by the module unload/reload operation. This is a result that would not occur if the module was retained throughout the overlay.

**Byte 50 - CGA Emulation** [4.10.23]

This byte has been implemented to provide an option for emulating the color effects used on CGA monitors when run in other environments.

NPL applications that specifically use the underlined colors options on a CGA monitor, may dynamically change byte 1 of \$OPTIONS (underline replacement color on a CGA display) to obtain a variety of colors. Applications of this type typically suffer when run under MS-Windows or in graphics mode (/G option under MS-DOS) since these environments display the actual underlines and the color replacements that worked on a CGA monitor no longer display.

Here, even on a CGA monitor, attempts to preserve the screen contents with INPUT SCREEN/PRINT SCREEN will not work if the value in byte 1 of \$OPTIONS is changed when displaying different areas of the screen. This is because all PRINT SCREEN operations use the current value in \$OPTIONS byte 1 for all the displayed underlined characters.

When \$OPTIONS byte 50 is set to HEX(00) (the default), no changes from previous releases occur.

When \$OPTIONS byte 50 is set to HEX(01) (CGA emulation), all underlined and underline/inverse video characters display using the same color combination as would occur if the monitor was a CGA, instead of with underlines.

Other bit values are reserved and should not be used.

**INPUT SCREEN / PRINT SCREEN Considerations** [4.10.23]

Besides standard display affects, when the CGA emulation option is set, the values returned by INPUT SCREEN in the colors section reflect those colors and the underline and inverse video attributes for underlined characters will be "off". Also, characters in the normal (not alternate) character set in the range HEX(90)-HEX(FF) (excluding extensions to the normal character set defined by \$OPTIONS bytes 18 and 19) will appear in the characters section as their equivalent values with the HEX(80) bit stripped off.

**NOTE:** This means applications may not locate underlined character strings in an INPUT SCREEN buffer by examining the character or attribute areas, if the display was generated with the CGA emulation option set.

When the CGA emulation option is set, values passed to PRINT SCREEN in the colors section will display as colors on the terminal, even when the color control sequences are disabled (i.e., if byte 22 of \$OPTIONS is HEX(00)).

**Limitations and Warnings for \$OPTIONS Byte 50** [4.10.23]

When the CGA emulation option is set, PRINT operations translate underlined video options to a color combination. Consequently, values in the attributes section of the PRINT SCREEN buffer, (under normal circumstances) will never have the underlined video flags set. Characters in the normal character set (i.e., those without the alternate character set attribute) will not be in the range of values that would cause them to be displayed as underlined. If buffer values, which are generated by any method other than an INPUT SCREEN operation, are supplied to PRINT SCREEN and the buffer has these video attributes set, the appearance of these characters will be system-dependent.

**NOTE:**        **Use of this option on monochrome monitors is not recommended. It will result in the removal of the underline video attribute on all displays. Underlined characters will appear bright or blinking (or the closest available equivalent) if a CGA monitor would display a bright foreground color or as a blinking character.**

Dynamically changing this option when displaying different parts of the screen is not recommended.

The following new bit in \$OPTIONS byte 50 instructs the RunTime to maintain the color of the perimeter line (25th line of screen) when available (i.e., MS-Windows) in the same way as direct video modes would on a CGA or EGA terminal under DOS. This option will also affect the values stored in an INPUT SCREEN buffer's 25th line color area.

Byte 50	HEX(02) bit = 0	Color of 25th line maintained as in previous NPL versions.
	HEX(02) bit = 1	Maintains color of 25th line in MS-Windows in same manner as direct video modes (CGA, EGA) under MS-DOS.
	Other bits = 0	All other bits reserved.

**Byte 51 - Multitasking Timing Control** [4.20]

This byte has been implemented to handle timing issues related to multitasking environments.

Byte 51	HEX(01) bit = 0	Default - An implied \$BREAK operation occurs after a polling KEYIN if no keystroke was detected.
	HEX(01) bit = 1	No implied \$BREAK operation occurs after a polling KEYIN if no keystroke was detected.
	Other bits	All other bits reserved.

NPL 4.10.xx versions and greater will typically perform the DOS equivalent of a timeslice release operation when an explicit or implied \$BREAK operation is performed. This was not done by prior revisions of NPL.

The timeslice release operation is used by multitasking support layers (i.e., Windows virtual machines, DESQview, etc.) to better support multiple task scheduling. Unfortunately, this is a relatively inefficient operation and the effect on applications that use polling KEYIN in low level loops can be severe when compared to previous releases. Such applications may wish to enable this bit, with the understanding that if the NPL program runs in background, foreground response of other applications may be adversely affected.

**Byte 52 - HALT/Function Control** [4.20]

This byte permits supporting a closer equivalent to the old 2200 RESET key under certain conditions.

Byte 52	HEX(01) bit = 0	Default -Previous NPL functionality of HALT key sequences apply.
	HEX(01) bit = 1	When the options bit is set, pressing the HALT key sequence (DOS CTRL-ALT, Windows CTRL-BREAK) a second time if the system does not immediately respond to the first will invoke the HELP display with the message 'Reset Key Sequence Pressed' under certain conditions. At this point the 'Leave Help' or 'Cancel' keys will continue normal operation and then HALT when the statement completes. The 'Reset' option will immediately terminate processing. As always with the Reset option, currently open files are \$CLOSEd.
	HEX(02) 0 (WinDefault)	CTRL-BREAK acts as a HALT key
	HEX(02) 1 (DOS Default)	CTRL-BREAK is disabled as a HALT key
	HEX(04) 0 (DOS Default)	CTRL-ALT acts as a HALT key
	HEX(04) 1 (WinDefault)	CTRL-ALT is disabled as a HALT key
	Other bits	All other bits reserved.

**NOTE:** A limited number of other Help options may also be available on this help display, such as 'Display' and 'Kill Runtime'.

The specific conditions that permit RESET operation include:

- While waiting for a busy device (hung \$OPEN) under DOS/ Windows.
- During a buffered I/O operation (COPY, MOVE).
- During a long I/O operation (SCRATCH DISK, MOVE END on network file).

This option may be useful in an environment where unexpected device interlocks frequently occur.

This option is supported for all interpretive MS-DOS, MS-Windows and 386/DOS Extender versions of NPL (RTI, RTIWIN, RTIWIN32 and RTI386) and non-interpretive MS-Windows and 386/DOS Extender versions of NPL (RTPWIN, RTPWIN32, and RTP386). This option is not supported by the non-interpretive MS-DOS version of NPL (RTP). The \$OPTIONS byte that disables the HALT key will also disable this option. If the \$OPTIONS byte that disables the RESET option in HELP is set, the Reset option will be unavailable.

**NOTE:** For technical reasons, the use of CTRL-BREAK on the DOS and Pharlap versions is only available if Mouse driver support is selected on the command line (/K option). CTRL-ALT continues to be supported with or without /K.

#### **Byte 53 - Box Graphics Display** <sup>[4.20]</sup>

This byte supports different preferences in the way box graphics are displayed on the screen.

On previous releases, a PRINT BOX() is designed to be as non-invasive as possible, affecting already displayed text as little as possible, and preserving all video and color modes. In addition, character boxes are always displayed without the bright or blinking attributes; the intention here was to make vertical box lines as uniform as possible.

In some cases though, users have requested the ability to attach colors to a displayed box, and possibly make the foreground color of the box bright as well. The new \$OPTIONS byte 53 allows this as well as a number of other options.

Bits in \$OPTIONS byte 53 have the following effect on existing displayed text during a PRINT BOX() when set:

Byte 53	HEX(01) bit = 1	Foreground color changes to currently selected
	HEX(02) bit = 1	Background color changes to currently selected
	HEX(04) bit = 1	Existing box fragments are erased
	HEX(08) bit = 1	Underline video changes to currently selected
	HEX(10) bit = 1	Bright video changes to currently selected
	HEX(20) bit = 1	Reverse video changes to currently selected
	HEX(40) bit = 1	Background color changes to currently selected
	HEX(80) bit = 1	Existing text is erased.

The default value is HEX(00), which is compatible with previous releases. Common values will use a combination of the above bits, for example:

HEX(53)	Foreground, background, bright and reverse changes to current selection. Text is not erased but colors (including effects of bright and reverse video) will be changed to be the same as the box.
HEX(FF)	All pre-existing information is overwritten, including text.

**NOTE:** These changes affect the contents of the values returned by INPUT SCREEN. The effects occur whether drawing a PRINT BOX() or erasing a PRINT BOX().

Text attributes and colors are affected in either true graphics or character graphics mode.

**NOTE:** The color of true boxes is not affected by the foreground or background colors.

It is recommended that the cursor should be off when the value of this \$OPTIONS byte is changed, and a consistent value in the option should be used.

**NOTE:** The option also affects the result of printing blank characters on top of existing boxes to a limited extent: If the HEX(10) or HEX(20) bits are set, the bright and blinking attributes will not be suppressed.

**Byte 54 Mouse Support for Terminal Emulators**

Byte 54 provides support for terminal emulators that have the ability to provide mouse support. \$OPTIONS byte 54 must be set to the complex key code that indicates a mouse position report is about to be received.

Byte 54	HEX(00) bit =0	Default. This feature is disabled.
	HEX(00) bit =1	When set, indicates mouse position for terminal emulators.
	Other bits = 0	All other bits reserved.

The default value for most terminal types is HEX(00), which means that this feature is disabled.

**NOTE:** This byte applies only to the UNIX operating system and has no effect in the MS-Windows environment.

**Byte 55 Mouse Pixel Position Support for Terminal Emulators**

Byte 55 provides support for terminal emulators that have the ability to provide mouse support. \$OPTIONS byte 55 must be set to the complex key code that indicates a mouse pixel position report is about to be received.

Byte 55	HEX(00) bit =0	Default. This feature is disabled.
	HEX(00) bit =1	When set, indicates mouse pixel position for terminal emulators.
	Other bits = 0	All other bits reserved.

The default value for most terminal types is HEX(00), which means that this feature is disabled.

**NOTE:** This byte is specific only to UNIX operating environments.

**NOTE:** This byte applies only to the UNIX operating system and has no effect in the MS-Windows environment.

**Byte 56 - PRINT SCREEN Ignore** <sup>[4.21]</sup>

This byte is available that causes PRINT SCREEN to ignore the last line of information in the buffer (which is normally used to redisplay box horizontals that would be in this area).

Byte 56	HEX(01) bit =0	Default - True boxes are handled as in past releases.
	HEX(01) bit =1	When using true boxes, BOX items displayed by \$DEMO scripts do not blank the last line of the area. Documented in the Release 4.21 Addendum.
	HEX(02) bit =0	Default - PRINT SCREEN commands are handled as in past releases.
	HEX(02) bit =1	NPL ignores the last line of information in the buffer on all PRINT SCREEN commands, on all platforms and configurations.
	Other bits = 0	All other bits reserved.

Applications that use this setting and also use pop up displays must be careful not to draw boxes of the full size specified by INPUT SCREEN, since the horizontal lines of the bottom of the box will affect the screen outside the area that is subsequently restored by PRINT SCREEN.



**Byte 57 - CUA Key Mode and Availability of the F10 Key** <sup>[4.22]</sup>

Byte 57 of \$OPTIONS has been implemented in NPL Revision 4.22 and currently provides developers with two new capabilities.

Bits 1, 2, and 4 of Byte 57 are specific to enabling CUA key mode in order to allow keyboard text selection.

Bit 8 of Byte 57 allows developers to suppress the standard MS-Windows functionality of the F10 key and remap it for application use.

The values are:

Byte 57	HEX(00)	Default
	HEX(01)	When set to 1, enables CUA key mode in command line editing.
	HEX(02)	When set to 1, enables CUA key mode in LINPUT
	HEX(04)	When set to 1, enables CUA key mode in INPUT
	HEX(08) = 0	Default - same as previous releases - F10 selects the main window menu bar, and may not be remapped.
	HEX(08) = 1	F10 key does not select the menu bar, and may be remapped.

For a complete discussion of \$OPTIONS Byte 57 Bits 1, 2, and 4, refer to Section 5.3.2, "Using the Keyboard to Select Text." For historical reasons, most non-NPL Windows applications reserve F10 as an alternate system key to access the main window menu bar. In order to follow this convention, and to allow operation using older keyboards that do not have F11 or F12 keys, previous NPL versions do not make use of any of the F9-F12 keys, and in fact the F10 key cannot be remapped.

For applications that make heavy use of function keys, the ability to remap the F10 key has been reinstated as an option.

**Byte 58 Enable Streaming I/O** <sup>[5.00]</sup>

Enabling \$OPTIONS byte 58 allows access to native files using stream I/O.

Byte 58	HEX(01) Bit = 0	Default
	HEX(01) bit = 1	Extended device specifications allowed.
	Other bits = 0	All other bits reserved.

When this options byte is enabled, alpha-literals or variables used in device specifications defined by SELECT #n may be either:

A 3-character (usually hex digit) value as on previous releases.

A directory name of any length.

**NOTE:** On previous releases, and if the \$OPTIONS bit is not set, the length of such a literal or alpha-variable has to be at least 3 characters, of which the first 3 must be hex digits and all characters after the third are ignored.

**8.5.3 \$BOXTABLE** <sup>[4.20]</sup>**Byte 1 - Disable / Enable Character Boxes**

A new value in \$BOXTABLE Byte 1 (HEX(02)) can be used to obtain character boxes under Windows even when the font selected does not contain character box pieces as part of the font's character set.

Byte 1	HEX(00)	Default - "Character" boxes are disabled
	HEX(01)	"Character" boxes are enabled
	HEX(02)	"Character" boxes are enabled. The values in the remaining part of \$BOXTABLE are ignored.
	Other bits	All other bits reserved.

On other platforms and on previous releases of RTI, setting the first byte of \$BOXTABLE to HEX(02) is treated the same as setting it to HEX(01).

**NOTE:**        **On future versions of RTI and RTI386, setting the first byte of \$BOXTABLE to HEX(02) may have a similar effect if /G mode is in use.**

## Index of Terms

\$ARGS	1-4, 8-12
\$BOXTABLE	8-80
\$CLOSE	6-17, 8-30, 8-68
\$DECLARE	1-4, 1-5, 1-8, 1-11, 5-15, 6-11, 8-7, 8-14-8-20, 8-22-8-24, 8-29, 8-35, 8-66
\$DEMO	8-3, 8-4, 8-77
\$DEVICE	1-9, 2-11, 6-12, 6-14, 6-16, 6-17, 6-19, 7-2, 7-4, 7-5, 7-7, 7-8, 8-3-8-6, 8-29-8-31, 8-68
\$HELPINDEX	8-3
\$LDATE	1-5, 1-8, 7-19, 8-8, 8-11, 8-32
\$MACHINE	1-3, 2-10, 5-7, 8-1, 8-3, 8-4, 8-49-8-52
\$NETID	1-20, 2-9
\$OBJECT	1-4, 5-11
\$OPEN	6-13, 7-11-7-13, 8-8, 8-26, 8-28, 8-30, 8-56-8-59, 8-73
\$OPTIONS	1-3-1-5, 1-7, 1-10, 3-12, 5-2, 5-13, 6-9, 6-19, 7-9, 7-11-7-13, 7-16, 7-17, 7-22, 7-27, 8-1, 8-3, 8-22, 8-24, 8-26, 8-37-8-40, 8-49, 8-52-8-63, 8-66-8-71, 8-74-8-76, 8-78, 8-79
\$RUNDIR	1-4, 8-43
\$SHELL	8-67, 8-68
#ID	1-20
2200S	5-13, 5-14
AUTOSIZE	6-4
B2C	5-10-5-14, 7-3, 7-23
BESDK	1-4, 1-2, 1-7, 1-8, 2-26, 2-27, 7-13, 7-14, 8-22
BLOCK DEINDENT	7-25
BLOCK INDENT	7-25, 7-30
Byte 1	8-70, 8-80
Byte 14	5-2, 8-52
Byte 16	1-10
Byte 2	8-39, 8-49
Byte 20	2-10, 8-53
Byte 22	8-71
Byte 3	8-37
Byte 30	8-49
Byte 41	8-52
Byte 42	7-9, 8-3, 8-52, 8-60
Byte 43	8-52
Byte 44	8-52

Byte 45	8-61
Byte 46	8-63
Byte 47	6-9, 8-22, 8-24, 8-63, 8-66
Byte 48	8-67, 8-68
Byte 49	8-69
Byte 5	8-39
Byte 50	8-70, 8-71
Byte 51	8-72
Byte 52	7-22, 8-73
Byte 53	8-74, 8-75
Byte 54	8-76
Byte 55	8-76
Byte 56	8-77
Byte 57	1-4, 7-27, 8-78
Byte 58	7-16, 8-79
Byte 6	8-39
Certificate of License And Authenticity	4-4, 4-8, 4-14, 4-16
Copy	1-4, 2-16, 4-9, 4-17, 5-11, 7-3, 7-24, 7-25, 7-29, 8-21, 8-43, 8-73
COPYALL.BAT	1-12, 2-16, 2-17, 2-19, 2-20
CUA	1-4, 7-27-7-29, 8-78
Cut	1-4, 7-24, 7-25, 7-28, 7-29, 8-61
DATALOAD DC	7-13, 8-59
DATASAVE BA	7-12, 7-16, 8-58
DATASAVE DC	7-13, 8-58
Debug	1-3, 1-11, 6-4-6-6, 6-8-6-10, 8-10, 8-63, 8-64
Delete	5-13, 5-14, 6-20, 7-24-7-26, 7-29
Error Codes	5-1, 5-15, 8-10, 8-68
Fingerprint	1-17, 2-7, 2-17, 3-6, 3-12, 5-4-5-7, 6-16, 8-51
Gateway to ODBC	1-2, 1-5, 1-6, 2-26
Gold Key Security	1-18, 2-1, 2-14, 2-17, 2-18, 4-3, 4-4
Hard Mode	6-5, 6-10
IDE	1-2, 1-1, 1-8, 1-1, 1-2, 1-6, 1-7, 2-1, 2-21, 2-24-2-28, 7-14
Install Security	1-16, 1-19, 4-5, 4-6
Instant Vinny	1-5, 1-3, 2-24, 6-11
KEEPREMS	5-12
LIMIT	1-4, 1-12, 1-13, 1-18, 2-8, 2-19, 4-1, 4-14-4-17, 5-5-5-8, 5-11, 6-19, 7-4, 8-50
Limit Code	4-16
Mixed Network	1-5, 1-16, 1-17, 1-19, 3-1, 3-5, 3-15, 5-2, 6-7, 6-18, 6-19, 8-59
MXE=Y	1-3, 1-9, 6-17, 7-8
NDM	1-2, 1-2, 1-4, 1-5, 1-8, 2-25, 3-13
NIAKINST	1-12, 1-17, 2-14, 2-17, 2-19, 3-12

NIAKINSW	1-11, 2-14, 2-15, 2-19
NIAKNETB	1-3, 1-4, 1-12, 1-20, 2-19, 3-3-3-8, 3-11, 5-4-5-8, 6-16, 8-47, 8-50, 8-51
NIAKNETW	1-3-1-5, 1-8, 1-9, 1-11, 2-10, 3-3-3-11, 5-4, 5-8, 6-2, 6-3, 6-16-6-18, 8-47
NT	1-4, 1-6, 1-8, 1-3, 1-9, 1-18, 1-19, 2-2, 2-3, 2-9, 2-10, 2-15, 2-18, 2-21, 2-27, 3-2-3-4, 3-8-3-11, 3-14-3-16, 4-3, 4-5, 4-7, 6-1, 6-2, 6-12, 6-15-6-19, 8-9, 8-49
ODBC	1-2, 1-2, 1-4-1-6, 2-25, 2-26
OPEN NDM	1-2, 1-4, 1-5, 1-8, 2-25
Paste	1-4, 7-24, 7-25, 7-28, 7-30
Recalling Security	2-17
RTIWIN.INI	1-5, 1-8, 1-10, 1-12, 3-3, 3-5, 6-2-6-4, 6-6, 6-10, 6-11, 6-16, 6-18, 7-17, 8-23
Select Text	7-26, 7-27, 8-78
SETUP.INI	2-8, 2-12, 4-5, 6-2
SHOWNPL	1-18, 2-19, 4-3, 4-8, 4-9, 4-13
UNC Notation	6-12, 6-19, 7-4
Undo	1-4, 7-24, 7-25, 7-30
Universal Upgrade	1-4, 1-5, 1-7, 1-8, 1-1, 1-10, 1-14, 1-15, 1-18, 1-19, 4-1-4-3, 4-5-4-8, 4-10, 4-11, 4-13
Visual Interface Manager	1-2-1-4, 2-24, 6-11
VLM	1-3, 3-13, 3-14, 6-13, 7-4
WAN	3-1, 3-9, 3-16
WIN2227	1-8, 1-9, 2-10, 6-12, 6-17, 7-8
Workbench	1-2, 1-1, 1-3, 1-4, 1-7, 2-21-2-23, 2-25, 5-10, 5-14