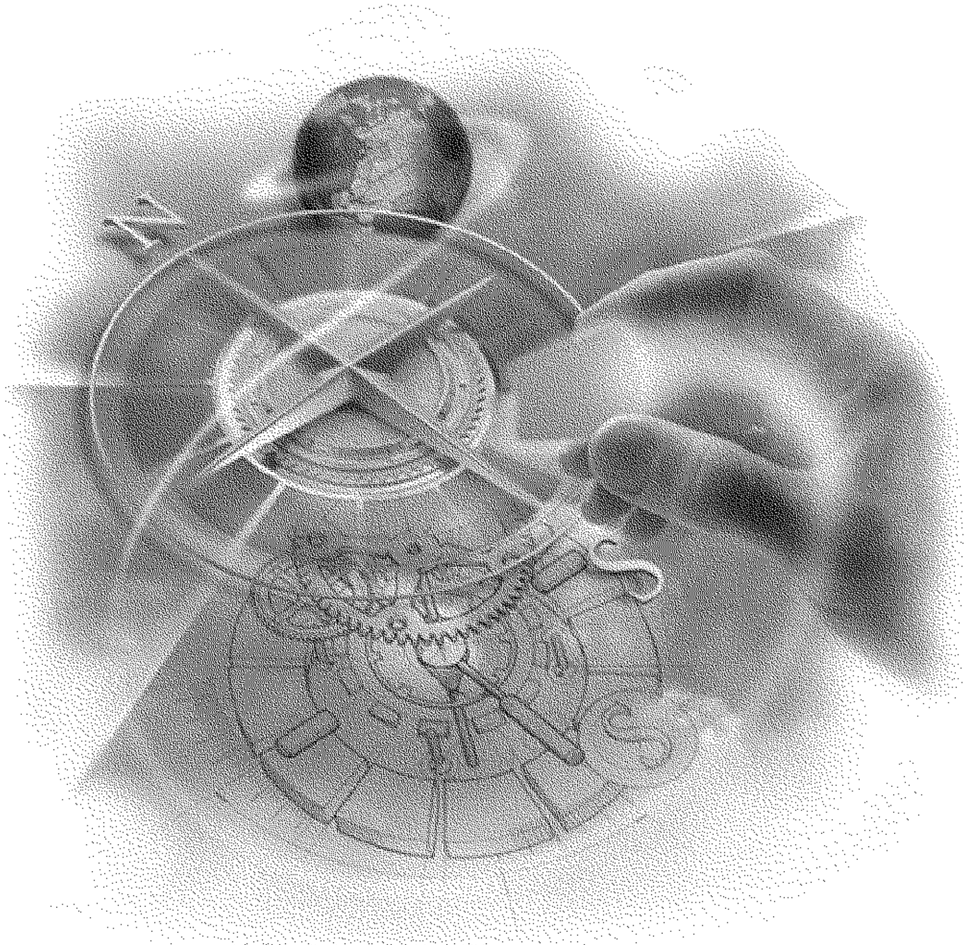
Installation Information File Specification

Novell Developer Kit



Novell®

Legal Notices

U.S. Patent Nos 5,553,139; 5,553,143; 5,677,851; 5,758,069; 5,784,560; 5,818,936; 5,864,865; 5,903,650; 5,905,860; 5,910,803 and other Patents Pending.

Novell, Inc.
122 East 1700 South
Provo, UT 84606
U.S.A.

www.novell.com

Installation Information File Specification
November 1999
104-000190-001

Online Documentation: To access the online documentation for this and other Novell developer products, and to get updates, see developer.novell.com/ndk. To access online documentation for Novell products, see www.novell.com/documentation.

Novell Trademarks

AppNotes is a registered trademark of Novell, Inc.

AppTester is a trademark of Novell, Inc., in the United States.

ArcNet 68 is a trademark of Novell, Inc.

BorderManager is a trademark of Novell, Inc.

C3PO is a trademark of Novell, Inc.

Client 32 is a trademark of Novell, Inc.

ConsoleOne is a trademark of Novell, Inc.

Controlled Access Printer is a trademark of Novell, Inc.

Custom 3rd-Party Object is a trademark of Novell, Inc.

DeveloperNet is a registered trademark of Novell, Inc.

DeveloperNet 2000 is a trademark of Novell, Inc.

GroupWise is a registered trademark of Novell, Inc., in the United States and other countries.

GroupWise 5 is a trademark of Novell, Inc.

Hardware Specific Module is a trademark of Novell, Inc.

HostPublisher is a trademark of Novell, Inc.

Hot Fix is a trademark of Novell, Inc.

HSM is a trademark of Novell, Inc.

InForms is a trademark of Novell, Inc.

Internetwork Packet Exchange is a trademark of Novell, Inc.

IPX is a trademark of Novell, Inc.

IPX/SPX is a trademark of Novell, Inc.

LANalyzer is a registered trademark of Novell, Inc., in the United States and other countries.

Link Support Layer is a trademark of Novell, Inc.

LSL is a trademark of Novell, Inc.

ManageWise is a registered trademark of Novell, Inc., in the United States and other countries.

Mirrored Server Link is a trademark of Novell, Inc.

MLI is a trademark of Novell, Inc.

MLID is a trademark of Novell, Inc.

MSL is a trademark of Novell, Inc.

Multiple Link Interface is a trademark of Novell, Inc.

Multiple Link Interface Driver is a trademark of Novell, Inc.

NControl is a trademark of Novell, Inc.

NCP is a trademark of Novell, Inc.

NDebug is a trademark of Novell, Inc.

NDPS is a registered trademark of Novell, Inc.

NDR is a trademark of Novell, Inc.

NDS is a trademark of Novell, Inc.

NDS Manager is a trademark of Novell, Inc.

NetWare is a registered trademark of Novell, Inc., in the United States and other countries.

NetWare 386 is a trademark of Novell, Inc.

NetWare Aware is a trademark of Novell, Inc.

NetWare Connect is a registered trademark of Novell, Inc, in the United States.

NetWare Core Protocol is a trademark of Novell, Inc.

Third-Party Trademarks

Contents

1 Installation Information File Specification

2 Driver Section-General

3 Driver Section-Parameters

4 Predefined Parameters

5 Driver Installation Information Template

6 Sample DDI/LDI files for Family Drivers with ISA Support

7 16-Bit DOS Client Installation Information (INS) File

1

Installation Information File Specification

This document specifies the contents and development process for the development of installation files for Novell 32-bit drivers.

What is the Installation Information File?

In order for a software utility to install drivers, it must know the parameters associated with each driver, the input required from the user, and how to set up the Novell configuration file(s). The driver installation information file for a driver describes the configurable driver parameters, the input required from the user, and the format of the required output. This information file must be shipped with the driver.

The information file is an ASCII text file containing one or more driver descriptions. Each driver description refers to a primary driver file and can also refer to other auxiliary driver files. Multiple descriptions in multiple files can also refer to a single driver. During installation, the installation information and the referenced driver file(s) are copied to a permanent directory on the machine's hard drive.

IMPORTANT: All driver developers must include an appropriate installation information file with the driver.

Cards and Buses Supported

Previously, ISA, MCA, and EISA were the only bus types supported in installation information files. Following is the expanded list of bus identifiers:

- ISA (standard ISA)
- PNPISA (ISA with a PnP BIOS, or a PnP Configuration Manager)
- EISA
- MCA
- PCI (includes CardBus)
- PCMCIA

In NetWare 4.1 or earlier, INSTALL does not evaluate PCI, PNPISA or PCMCIA but proceeds normally for ISA, EISA and MCA, even if newer bus types are also in the file.

The installation information filename must have one of the following extensions, depending on the type of driver:

Environment	Extension
32 bit LAN Driver (Server and Client32)	.LDI
Server Disk Driver	.DDI
Server Asynchronous Communications Driver	.ADI
Server CAPI (ISDN) Driver	.CDI
Server Platform Support Module (PSM)	.MDI
Generic NLM file	.NDI
Hot Plug Module	.HDI

Information File Format and Syntax

Reserved Keywords

The keywords listed in the following table have special meaning in the installation information file. This specification describes, in the appropriate section, each of the keywords listed in the table.

NOTE: This guide uses the full form of the keywords for clarity; however, Novell strongly suggests you use the abbreviated form when creating the final installation information file. The sample code uses the abbreviated forms.

Keyword	**Abbreviation**
;@# ALIAS	;@# ALIAS
AND	AND
CDESCRIPTION	CD
CHOICE	CH
;@# CLIENT	;@# CLIENT
CPROG	CP
DECIMAL	DEC
DEFAULT	DEF
DESCRIPTION	DES
DLANGUAGE	DLANG
DRIVER	DR
ELSE	ELS
FILE	FILE
FRAME	FR
HELP	HELP
HEX	HEX
HIDDEN	HID
;@# HPROD	;@# HPROD
IF	IF
LANGUAGE	LANG
LIST	LI

Keyword	Abbreviation
NOT	NOT
OCTETBITORDER	OCT
OFILES	OF
OPTIONAL	OPT
OR	OR
OUTPUTFORMAT	OUT
PARAMETERINFORMATION (formerly PARAMETERVERSION)	PAR
PATH	PATH
PRODUCTID	PROD
PROMPT	PR
REQUIRED	REQ
RESERVEDLENGTH	RES
;@# SPROD	;@# SPROD
STRING	STR
SYNTAXVERSION	SYN
;@# SYNTAXVERSION	;@# SYN
TIMEOUT	TIME
TYPE	TYP
UNDEFINED	UND
VALUES	VAL
VERSION	VER

General Format of the Installation File

A driver information file contains one or more driver descriptions, as well as language definitions for the strings within the descriptions. The general format of the description file is shown below:

```
   ; VeRsIoN=<version number and info for support utilities>
   ; CoPyRiGhT=(c)Copyright <years> by <vendor>. All rights reserved.


   Version: <driver description file number>
   SyntaxVersion: 1.0

   Driver <driver description #1 name> <dependency expression>
   {
   <driver #1 description, may include $<string> variables>
   }

   Driver <driver description #2 name> <dependency expression>
   {
   <driver #2 description, may include $<string> variables>
   }
   .
   .
   .

DLanguage: <default language ID>
   $<string #1 variable name> = "<string #1 text in default language>"
   $<string #2 variable name> = "<string #2 text in default language>"
   .
   .
   .

Language: <language #1 ID>
   $<string #1 variable name> = "<string #1 text in language #1>"
   $<string #2 variable name> = "<string #2 text in language #1>"
   .
   .
   .

Language: <language #2 ID>
   $<string #1 variable name> = "<string #1 text in language #2>"
   $<string #2 variable name> = "<string #2 text in language #2>"
   .
   .
   .
```

The following section describes each portion of the above example.

NOTE: String values (help text, label values and so on) are case-insensitive. Don't try to use an LDI, DDI, or other installation information file to pass information to another parser if case sensitivity is important.

VeRsIoN (comment)

This comment is used for support utilities to get additional information about the file and its associated driver. This comment is required. It begins with the semicolon character ; followed by a space and then the key word VeRsIoN= (with that capitalization). The version number and any other pertinent information comes after the equals sign, with no space after the equals sign. For example

```
; VeRsIoN=1.03 Novell LAN Installation Information File for CNE2000
```

CoPyRiGhT

This comment is used for support utilities to get additional information about the driver. This comment is required. It begins with the semicolon character ; followed by a space and then the key word CoPyRiGhT= (with that capitalization). The copyright information must include years copyrighted and current owner. There is no space after the equals sign. For example

```
; CoPyRiGhT=(c)Copyright 1996,1997 by Novell, Inc. All rights reserved.
```

Version

The *Version* label is optional. The *Version* label allows you to manually control your version number of the installation file. While the label is optional, if you use it you MUST make sure the version number in this label matches the number in the required ; VeRsIoN comment string.

SyntaxVersion

The *SyntaxVersion* label is required. *SyntaxVersion* informs the installation/configuration parser of the syntax to expect in the driver information file. Novell controls the syntax version number; the version is currently 1.00.

;@# SyntaxVersion

Same as the *SyntaxVersion* label, but is recognized only during NetWare 5.x installation (See item 9 under General Information File Syntax below).

Driver Section

The driver section includes one or more driver information blocks. Each driver block contains a short description of the driver, help information, and the driver's configurable parameters. Due to the complexity of this section, Chapter 2 of this document is devoted to its syntax.

Language Section

The language section allows *you* to translate text strings (help messages, prompts, etc.) to different languages. Language translation is optional and is not necessary for drivers that will operate only in a single language.

If you implement the language section, the driver section references any translatable text strings with string names. Each language block then contains the string names and the corresponding text in the respective language. See "Language Translation" for details.

General Information File Syntax

This section describes the general syntax of a driver installation information file.

1. You can add comment lines by starting the line with a semicolon (;). The parser ignores everything on the rest of that line, unless you use the three character escape sequence <;@#> (see #9 below). A semicolon can be preceded by white space (tabs or space characters). A comment can exist on the same line as a declaration-the parser will parse the command but ignore everything from the semicolon to the end of the line. For example, the following are both valid (In the second case the *Language: 14* will be parsed correctly and the rest ignored. In the third case, *;@# Sprod: 'Novell~10T Adapter'* will be parsed correctly and the rest ignored):

```
; installation file for driver: NE2000.LAN

Language: 14 ; Spanish for Latin America
```

```
;@# Sprod: 'Novell~10T Adapter' ; See Prod
```

2. Items in angle brackets indicate something that you must supply. The supplied item describes that aspect of the driver. For example:

```
File: <filename>
File: NE2000.LAN
```

3. Items in square brackets ([ ]) are optional.

4. Items separated by the word *or* indicate alternates. For example:

```
THIS or THAT
```

5. Labels consist of one or more words followed by a colon. Labels are not case-sensitive. For example, the label *File:* is the same as *FILE:*.

6. Double quotes (" ") , single quotes (' ') , or whitespace can surround text strings.

   a. Strings without quotes must not contain white space, single or double quote characters, double-byte characters, or the following reserved characters: = { } ( ) , : - ; < > !

   b. Strings surrounded by single quotes can contain white space, single or double quote characters, double-byte characters, or the following reserved characters: = { } ( ) , : - ; < > !

   c. Strings surrounded by double quotes are treated as single-quoted strings. However, these strings are text strings that can be translated to other languages. For example:

   ```
   "The driver defaults to using ..."
   'Novell NE2000'
   ISA
   ```

   A single quote character can appear in a double- or single- quoted string if it is preceded by a backslash (\'). However, a double quote character cannot appear in a double-quoted string even if it is preceded by a backslash.

   A backslash is represented within a single- or double-quoted string as (\\).

The newline and tab characters, \n and \t, can exist within single- or double-quoted strings. The parser changes the \n and \t to the appropriate characters.

7. Keywords and labels must not be enclosed in quotes. Therefore, labels cannot contain white space, single or double quote characters, double-byte characters, or reserved characters.

8. Help text within double quotes can be longer than one line. When a quoted string spans more than one line, all characters between the last non-white space on one line and the first non-white space on the next line will be replaced with a single space character. Help text in the driver section is limited to 1000 bytes; parameter help text is limited to (1011 - <length of translated parameter description text>) bytes.

   You can include a new line character in a quoted string by using \n. The parser will replace the \n by a CR-LF combination. \t indicates a tab.

   Newline characters and tabs may only appear in help text and output format strings. For example:

   ```
   "The ISADISK driver can be loaded twice. When
   loaded more than once, the driver loads
   reentrantly.\n\n

   The default settings are the standard values
   for an internal controller."
   ```

9. The NetWare 5.x installation has been modified to identify potential client-to-server driver match-ups during a network install. If the 16-bit or 32-bit client driver used to connect to the network matches a server driver available in the NetWare OS being installed, then the matching server driver will load automatically as part of the NetWare installation. To implement this functionality, new keywords and syntax have been added to the LDI/DDI specification

   The new syntax provides an escape sequence to hide the new keywords from older parsers that do not correctly implement syntax version checking. The escape sequence makes it possible for parsers to ignore unrecognized keywords in LDI/DDI files with syntax versions newer than their own. Syntax version checking has been fixed in NetWare 5.x so that the NetWare 5.x parsers are backward compatible with previous syntax versions.

A three-character escape sequence <;@#> has been added to the specification. The escape sequence hides new keywords from parsers that don't support syntax version checking correctly. NetWare 5.x parsers recognize the escape sequence and treat anything that follows it on the same line as regular syntax instead of a comment. All new keywords must be placed behind the escape sequence in the LDI/DDI file.

Language Translation

The installation information file’s language section allows the developer to add the translation of text strings to different languages. These text strings can include help messages and prompts. Language translation is optional and is not necessary for drivers that only operate in a single language.

If you implement the language section, any translatable text strings required in the driver descriptions use $ <*string_name*> variables instead of the actual text. Each language block then contains the string name and the corresponding text in the respective language.

The *Language* and *DLanguage* labels identify a block of text string translations for either a particular language or the default language. In all cases, the *DLanguage* (default language) label must be specified. If the *Language* label exists, the *DLanguage* label must also exist and must be the first language label. If only one language label exists in the file, it must be the *DLanguage* label.

When the installation utility encounters a *$<string_name>* variable, it searches for a definition of that string in the language block that corresponds to the installation/configuration utility's current language. If the string is not defined in the utility's current language, the utility searches string definitions in the default language block.

If a quoted string follows a $ <*string_name*> with no intervening white space, and if the string definition is not found, the installation utility uses the quoted string as the string definition. This feature allows you to specify a default string (typically in English) if the string definition is not found in the language sections. If the installation/configuration utility has already found a definition for the string, it will ignore the adjacent quoted string. For example:

```
$DESCRIPTION"Novell NE2000 Driver"
```

Finally, if the utility cannot find a string definition in any of the above mentioned forms, it will use the string name itself as the string text.

Language ID

A number, or language ID, identifies each language block. The language IDs that are currently assigned are listed below:

French (Canada)	0
Chinese-Simplified	1
Danish	2
Dutch	3
English (US)	4
Finnish	5
French (France)	6
German	7
Italian	8
Japanese	9
Korean	10
Norwegian	11
Portuguese (Brazil)	12
Russian	13
Spanish (Latin America)	14
Swedish	15
Chinese-Traditional	16
Polish	17
Portuguese (Portugal)	18
Spanish (Spain)	19

Hungarian	20
Czech	21
FAKE Japanese Wide Roman + Hyou	98
FAKE European Accented Expanded	99

The NetWare operating system and utilities will be translated into Chinese-Simplified, Chinese-Traditional, English (US), French, German, Italian, Japanese, Korean, Portuguese (Brazil), Russian, and Spanish. You can choose to provide text string translations for all, some, or none of the languages available. A sample installation information file with a Spanish language block is illustrated below:

Example with Spanish Language Block

```
; VeRsIoN=1.03 Novell LAN Installation Information File for SAMPLE2
; CoPyRiGhT=(c)Copyright 1997 by Novell, Inc. All rights reserved.

Version: 1.00
SyntaxVersion: 1.00

   Driver SAMPLE2
   {
      Description:$DRIVER_DESCRIPTION
      Help:$DRIVER_HELP
   }

   DLanguage: 4
      ; Default English

         $DRIVER_DESCRIPTION = "Place the driver description here"
         $DRIVER_HELP = "Place the help information here"

   Language: 14
      ; Spanish for Latin America

         $DRIVER_DESCRIPTION = "Poner la descripcion del driver aqui"
         $DRIVER_HELP = "Poner la informacion de ayuda aqui"
```

2

Driver Section-General

This chapter specifies the syntax for the Driver Section of a driver installation information file, except for the parameter section, which is defined in Chapter 3 "Driver Section–Parameters".

Syntax Overview

The format of an information file's *Driver* description is shown below. The driver description contains two sections. The first section contains information the installation program uses to decide (with input from the user) if and how to install this driver. This section includes everything from the beginning of the section through the *HProd* line. The second section contains the parameters that the user can change or select in order to configure the installed driver.

```
Driver <driver description name> <dependency expression>
{

   Description:           "<text description>"
   Help:                  "<multi-line help text>"
   ParameterInformation: <x.xx>
   ProductID:             <list of product ID strings>
   ;@# SProd:             <company name and module name>
   CProg:                 <(server-specific) NLM name>
   File:                  <file name on media>
   OFiles:                <other associated files>
   Timeout:               <decimal timeout value in seconds>
   ;@# Alias:             <old driver name>
   ;@# Client:            <Client driver name>
   ;@# HProd:             <Novell assigned driver number>

   PROMPT or LIST or FRAME
```

```
    {
    <see parameter section>
    }
}
```

All labels in the driver description are optional. You need only include the labels required for the particular driver being described. However we highly recommend that you define the *Description* and *Help* labels, to make installation and configuration easier for inexperienced users.

All labels and keywords can occur only once in the first section of a description. The labels and keywords in the second section of the description, the *PROMPT, LIST, or FRAME* keywords and the *Parameter* definitions, can occur as many times as you desire. The parameters are described later in this document.

The following sections describe the various parts of the driver description.

Driver Description Name

Syntax:

```
Driver <driver description name>
```

Example:

```
Driver SAMPLE1
```

Each driver description has a case-sensitive string (*<driver description name>*) associated with it. This string is a logical name that uniquely identifies the driver description. (Remember that an information file can contain multiple driver descriptions.) This name must not be more than 32 characters, and cannot include white space, quotes (single or double), double-byte characters, or reserved characters (see "General Syntax").

After the user has installed and configured a driver, the *<installation filename>, <driver description name>* pair associates a description with the driver file. Therefore, each description name must be unique from all others within the same file.

Dependency Expression

A dependency expression describes the state of a driver description. A dependency expression can appear in the context of a driver description (for example, before the Driver { }, as shown earlier in the “Syntax Overview”), or in the context of a driver parameter (for example, before the parameter { }). The dependency is either unconditionally or conditionally based on global parameters such as bus type (see "Global Predefined Parameters" in Chapter 4).

Used in the context of a driver description, the dependency specifies the conditions under which the entire driver description is HIDDEN (invisible, inaccessible) or OPTIONAL (visible, selectable) to the user.

Used in the context of a parameter, the dependency specifies the conditions under which the parameter is HIDDEN (invisible, no input, no output), REQUIRED (visible, input required, output required), or OPTIONAL (visible, input optional, output optional).

Example:

```
if (BUS == MCA) OPTIONAL
else HIDDEN
```

A dependency expression allows you to make descriptions invisible to the user if they are not applicable. (For example, a Micro Channel driver description would be hidden if the driver is being installed on an ISA machine).

If no dependency is declared, the driver description state defaults to OPTIONAL.

Dependency Expression Syntax

A dependency has the following syntax:

```
<dependency expression>  <- <state> or
                            if (expression) <state>
                            [else if (expression) <state>...]
                            else <state>

<state>              <-    OPTIONAL
                           HIDDEN
                           REQUIRED (parameter context only)
```

```
<expression>       <-    <log-expr> or
                         <expression><log-op><log-expr>
<log-op>           <-    OR  or  AND

<log-expr>         <-    <rel-expr> or NOT <rel-expr>

<rel-expr>         <-    (<expression>) or
                         <name><rel-op><name> or
                         <name><rel-op><constant>
<rel-op>           <-    ==   !=   <   >   <=   >=
```

<name>. A *name* can be either a global predefined parameter or a local parameter that precedes this parameter in the driver description. If the parameter named is a *PROMPT* parameter, that value depends on the *PROMPT*'s Type label. If the parameter named is a *LIST* or *FRAME* type, its value is a decimal number. String comparisons are *not* case-sensitive.

If the dependency expression is in the context of a driver description, *name* refers *only* to global predefined parameters (see "Global Predefined Parameters").

If the dependency expression is in the context of a parameter, *name* refers to either a global predefined parameter or to a local parameter that precedes this parameter and is in the same driver description.

<constant>. A *constant* may be either numeric or string-valued. Its type is assumed to be the type of the parameter to which it is being compared. All cross-type comparisons between strings and numbers are flagged as syntax errors. The typeless constant value, UNDEFINED, is used for comparing against parameters that do not have a defined value.

The following is an example of a conditional dependency statement:

Example:

```
PROMPT parameter2
if (parameter1 == 5 AND BUS == MCA)
   OPTIONAL
else if (parameter1 != UNDEFINED AND parameter1 != 5)
   REQUIRED
else
   HIDDEN
{
```

```
.
.
.
}
```

General Dependency Expression Syntax Rules

1. With the exception of the global predefined parameters, all names used in dependency expressions must be defined previously in the driver description. No forward references to a name are allowed. Any attempt to forward-reference a name will be flagged as an error and the driver description will be discarded.

2. No circular references to a name are allowed (in other words, a name cannot directly or indirectly depend upon itself).

3. In dependency expressions, any reference to a parameter name whose state is HIDDEN, or whose value is not specified (this could be due to an OPTIONAL parameter whose default value is UNDEFINED, or an OPTIONAL parameter whose default value was defined, but the user deleted it), will return a value of UNDEFINED for that parameter.

4. A REQUIRED parameter must have a valid (defined) value before the user can exit the form. As a result, you can write dependency expressions assuming that a REQUIRED parameter will always have a defined value and will never be UNDEFINED.

Evaluation of Dependency Expressions with UNDEFINED Parameters

Terms with == or != expressions that reference a parameter with an UNDEFINED value yield a valid result. All other relational operators result in an error for the term. Explicitly, the following expressions are valid if *name* has an UNDEFINED value.

```
name == value
name != value
```

The following expressions will result in a term evaluation error if *name* is UNDEFINED.

```
name >= value
name <= value
name > value
name < value
```

This also applies to expressions comparing two parameters (for example, name1 >= name2).

The installation/configuration utility resolves dependency evaluation errors, if they occur, by forcing the state of the parameter or driver description to OPTIONAL and reporting the error to the user before he or she exits the parameter form. (This error is nonfatal, and the driver could still work if the resultant output is reasonable.)

In order to prevent term evaluation errors from resulting in evaluation errors for the entire dependency, you should account for the UNDEFINED value when you write descriptions (either explicitly or implicitly). A dependency expression that explicitly compares an UNDEFINED value to UNDEFINED should appear prior to the term that could result in an error. For example:

```
if (param1 != UNDEFINED AND param1 > 30) REQUIRED
else HIDDEN
```

The above expression will not result in an dependency evaluation error if *param1* has an UNDEFINED value.

The following example illustrates an implicit comparison that takes UNDEFINED into account:

```
if (param2 == 3) REQUIRED
else HIDDEN
```

The above expression will result in a HIDDEN state if *param2* has an UNDEFINED value.

Handling Multiple Buses in One Machine

This specification accommodates multiple buses in the machine for which the driver is intended. This means that some statements for a PCI/ISA machine will be interpreted differently than with previous specifications. Consider the following expression:

```
PR INT if (BUS == ISA) OPTIONAL else HIDDEN { }
```

Previously, the BUS expression evaluated to a single bus type. Now, BUS may indicate a set of buses. The specification now makes the following interpretations:

- In the expression (BUS == <bus type>), if <bus type> is any element of a BUS set, the expression evaluates to TRUE. Otherwise, it is FALSE.
- Similarly, in the expression (BUS != <bus type>), if <bus type> is not an element of a BUS set, the expression will evaluate to TRUE. Otherwise, it will be FALSE.

Therefore,

1. Because an EISA bus is also ISA-compatible, the expression (BUS==ISA) evaluates to TRUE on an EISA machine.
2. Because an ISA machine with PnP support is also ISA-compatible, the expression (BUS==ISA) evaluates to TRUE on an ISA machine with PnP support.
3. On an EISA machine with PnP support, the expressions (BUS==ISA), (BUS==PNPISA), and (BUS==EISA) will all evaluate to TRUE.

NOTE: In previous versions (NetWare 4.1 and before) INSTALL.NLM does not evaluate PCI, PNPISA and PCMCIA in dependency expressions.

Description

Example:

```
Description: "Novell ISADISK (ISA or EISA) Driver"
```

The description label is followed by a case-sensitive string (or symbol reference), which is typically enclosed in double quotes. This string is displayed to the user during installation and configuration. The quoted string, if present, can be a maximum of 60 characters long, and must *not* contain newline characters (either symbolic \n or explicit).

NOTE: You can have multiple *Description* labels in a driver description section. Each description must also have a corresponding *Help* label following it. This feature is particularly useful for multiple language support using a *Description* label and *Help* label for each language-see "Language

Translation" in Chapter 1. The installation utility displays each description and help, but loads the same driver.

Help

Example:

```
Help:"This driver supports up to four NE1000
   network boards installed in ISA servers. Their
   settings must not conflict.\n\n
   You can load the driver for each board and for each
   additional frame type assigned to the board
   (maximum 16 times). The driver loads
   reentrantly, thus conserving memory.\n"
```

The *Help* label is followed by a case-sensitive string, usually enclosed in double quotes, that the user could optionally have decided to display during installation and configuration. This string contains additional information or cautions that a user might need to know about the driver. The help can contain newline characters (either symbolic \n or explicit). Help text in the driver section is limited to 1000 bytes for NetWare 4.x and 1500 bytes for NetWare 5; parameter help text is limited to (1011 - <length of translated parameter description text>) bytes. All explicit newline characters and adjacent whitespace are replaced with a single space. All symbolic new line characters are replaced with a CR-LF combination. \t represents a tab.

See also the note under the *Description* label in the previous section.

Guidelines for Writing Useful Help

Use driver help information correctly. This information is front-line technical information that the user needs to know to get your hardware working properly. Keep in mind the following:

1. Do not tell the user what keys to press or how to navigate the utility. Novell reserves the right to implement new installation utilities with a different user interface.

2. Answer the question, “Why should the user choose this driver?” What hardware (refer to specific numbers) does it work with? What are specific quirks to specific card revisions, etc? Include all hardware card names that could apply.

3. What are critical things to know about the particular driver? What other driver pieces must be loaded to make this driver work properly? Should the user be sure to check a particular parameter depending on a nonstandard hardware jumper, etc.? Does the user need to run EISA configuration and set some switches for optimal performance?

4. Who can the user contact for technical support? Some of the driver descriptions have telephone numbers for the user to call in case of problems. Is there troubleshooting information you could put in the driver help?

5. Format for easy readability. Brevity is important. Generally since not all of the information is visible at once, put the most critical information (why should the user choose this driver?) at the beginning. Information in bullets is easier to read than flowing paragraphs.

6. Watch grammar and spelling errors.

Use parameter help information correctly. This information is intended to guide the user through the specifics of individual parameters. Give them the technical content they need to know how to optimize the performance of your hardware. Typically standard parameters (interrupts, DMA, etc.) need no help information (INSTALL provides some default information). For each custom parameter provide information on the following:

1. What is the purpose of the parameter?

2. Should the user typically modify this parameter? Or is it better for those who don't know to leave it alone?

3. What are critical things to know about the parameter ? Will it cause data corruption if you set it incorrectly? Does it work most reliably with caching off, but is extremely slow?

4. Format for easy readability. Brevity is important. Generally since not all of the information is visible at once, put the most critical information (what does the user need to know about this parameter?) at the beginning. Information in bullets is easier to read than flowing paragraphs.

A description should correspond to a single driver file. There may, however, be multiple descriptions in a single description file. Parameters that are specific to a particular card may be best presented in a separate description for that card.

Write descriptions only for necessary modules. Many simple dependent modules typically do not need a description file (provided they are properly referenced in an OFiles field). Having a separate description file may be necessary if the user is to access it from the installation utility, but if it is always autoloaded, having a description makes it another choice for the user. If possible, make the loading decisions automatically.

Use dependency expressions for drivers and parameters to minimize the complexity and number of choices a user must make. The goal with Hardware Instance Number (HIN) and other recent Novell developments is to eliminate or minimize user interaction at driver installation time. Automate the process as much as possible. In some cases, parameters do not need to be displayed at all, but can be simply set through the driver description file.

Important parameters should occur first in the description. Parameters that are dangerous or infrequently set should occur later in the description, and they will be displayed in the order in which they occur. For example, node address for LAN descriptions should typically be the last parameter (unfortunately, some description examples violate this rule).

ParameterInformation

Example:

```
ParameterInformation: 1.00
```

ParameterInformation refers to information about the version of the driver and allowable command-line parameters (originally it only included version information and was called *ParameterVersion*). The *ParameterInformation* consists of two parts: the information bit-map number (also called information number) and the parameter version number, which are separated by a period character. Novell owns the information bit-map number and will assign specific meaning to its values. The driver developer owns the parameter version number and should change it only when the parameter interface changes, not when the driver description file is modified. The driver description file *Version* number is used to track file modifications.

The *ParameterInformation* information bit-map number precedes the period character. This number is input as decimal but will be interpreted as a bit-map according to the following scheme:

`bit 0:`	`Not used.`
`bit 1:`	`Hardware Instance Number Aware - a value of one means the driver is Hardware Instance Number Aware and can accept slot = <HIN> on its command line.`
`bit 2:`	`Load Once for Multiple Hardware Instances (LOMHI) Capable - a value of one means the disk driver is capable of being loaded once with no command line parameters and using all detectable instances of controller hardware found. (Install.NLM will only load non-HIN aware disk drivers in this manner. DDI files for HIN aware disk drivers should have this bit set to one if LOMHI capable).`
`bit 3:`	`Used for a Generic driver - a value of one gives the generic driver a handicap.`
`bit 4 and above:`	`undefined, must be zero.`

There should be one physical load instance per physical adapter or card. (Multifunction LAN cards already require one load instance per adapter or per function.) This lets the user know what load line corresponds to which card so the desired card can be disabled. There may be multiple logical load instances per physical card.

This means that a driver should not load and register all instances of the adapter. An exception might be an interim release of PCI cards or PnP cards, where the slot number (PCI bus, device, function) or the CSN (PnP card select number) may change after rebooting the machine.

NOTE: For DDI files, HIN awareness only applies to NetWare Peripheral Architecture (NWPA) Host Adapter Modules (HAMs), but not to DDI files for Custom Device Modules (CDMs). For example, PAR: 1.00 is valid for a CDM DDI file, but PAR: 2.00 is not.

The *ParameterInformation* parameter version number follows the period character and always consists of two decimal digits. The driver developer assigns this portion of *ParameterInformation*.

Valid PAR values (see the bit value table above, and the note on DDI files, for details):

1.xx-Not HIN aware.
2.xx-HIN aware.
4.xx-LOMHI-capable.
6.xx-Both HIN aware and LOMHI capable.

NOTE: The PROD string is required for INSTALL.NLM to even attempt to match the driver to the card (the examples in this section all include a PROD string). For more information on the PROD string, see "ProductID".

Non-HIN-Aware Example

```
DR Novell_NE3200
{
   PAR: 1.00 ;This is NOT a HIN Aware Driver
   FILE: NE3200.LAN
   PROD: EISA.NVL.070.1

   PR SLOT REQ
   {
      VAL: 1 - 8
   }
   .
   .
}
```

HIN-Aware Example

The following two examples are equivalent. HIN Aware drivers need to allow a valid slot value that is not restricted in its range.

```
DR Novell_NE3200
{
   PAR: 2.00 ;This IS a HIN Aware Driver
   FILE: NE3200.LAN
   PROD: EISA.NVL.070.1

   PR SLOT REQ
   {
   }
   .
   .


}
```

```
DR Novell_NE3200
{
   PAR: 2.00 ;This IS a HIN Aware Driver
   FILE: NE3200.LAN
   PROD: EISA.NVL.070.1

   PR SLOT REQ
   {
      VAL: 1 - 65535
   }
   .
   .
}
```

Non-HIN Aware, LOMHI Example

```
DR SCSI_HBA
{
   :
   PAR: 4.00;This is a non HIN aware, LOMHI capable disk driver.
   FILE: SCSI.DSK
   PROD: PCI.1000.0004.0000.0000.*

   PR SLOT REQ
   {
      VAL: 1 - 8
   }
   :
}
```

In the example above, when Install.NLM is in autodetection mode, the LOMHI capable flag takes precedence and will override the slot prompt statement and the driver will be loaded with no command line parameters. However, when picked from the driver list in Install.NLM, the driver parameters edit box will prompt for SLOT. (Note that this is for an MCA driver-EISA, of course, would have VAL: 1 - 15 instead of VAL: 1 - 8.)

HIN Aware, LOMHI Example

```
DR SCSI_HBA
{
   :
   PAR: 6.00;This is a HIN aware, LOMHI capable disk driver.
   FILE: SCSI.DSK
   PROD: PCI.1000.0004.0000.0000.*
```

```
    PR SLOT REQ
    {
    }
    :
}
```

Install.NLM will load the disk driver with the above.DDI file contents with a “slot=<HIN>” command line parameter because the driver is HIN aware. Since the disk driver is also LOMHI capable, this functionality is noted by bit 2 being set.

ProductID

The *ProductID* label (hereafter referred to by the shortcut *PROD*) specifies unique identification string(s) assigned to the product. The PROD label enables an install/configuration utility to match a driver with a particular device.

IMPORTANT: As of this specification release, the PROD label is required for every bus type except ISA. However, if an ISA card has an EISA product ID, the PROD label is required, and the EISA product ID should be used.

With version 4.11 of the operating system, Novell introduced new PROD syntax that is compatible with the previous syntax and with the newly included bus types.

Syntax

```
PROD: '<bus>.<INCLUDE|EXCLUDE>.<id>', '...'
```

Enclose each product ID string in single quotes and separate multiple strings with commas. <INCLUDE|EXCLUDE> is optional for all types. If INCLUDE is specified and Install finds a match, the driver will be selected. If EXCLUDE is specified and Install finds a match, the module will not be selected. If neither INCLUDE nor EXCLUDE is used, INCLUDE is assumed. (If both INCLUDE and EXCLUDE are used-which is not generally recommended-EXCLUDE takes precedence.)

Each product ID string must be followed by a single quoted product name string. A product name string is associated with the product ID immediately preceding it. The maximum length of the string is 60 bytes. The product name string is used by installation and configuration utilities. The string must be the marketing product name of the hardware adapter associated with the product ID.

PnP ISA and EISA syntax:

```
'PNPISA.<INCLUDE|EXCLUDE>.VVV.PPP.R'
'EISA.<INCLUDE|EXCLUDE>.VVV.PPP.R'
```

Where:

VVV = Vendor ID, uppercase letters A-Z
PPP = Product ID, hex characters
R = Revision number, hex character

Examples:

```
PROD: 'PNPISA.INCLUDE.CPQ.055.1'
PROD: 'EISA.CPQ.067.0'
```

MCA syntax:

```
'MCA.<INCLUDE|EXCLUDE>.HHHH'
```

Where:

HHHH = Adapter ID (POS registers 1, 0), hex characters

Examples:

```
PROD: 'MCA.8437'
PROD: 'MCA.EXCLUDE.8438'
```

PCI and CardBus syntax:

```
'PCI.<INCLUDE|EXCLUDE>.VVVV.DDDD.NNNN.SSSS.RR'
```

Where:

VVVV = Vendor ID, hex characters
DDDD = Device ID, hex characters

NNNN = Subsystem vendor ID, hex characters (0000 if pre-v2.1 PCI hardware)

SSSS = Subsystem ID, hex characters (0000 if pre-v2.1 PCI hardware)

RR = Revision number, hex characters

Example:

```
PROD: 'PCI.8086.0202.0000.0000.00'
```

PCMCIA syntax:

```
'PCMCIA.<INCLUDE|EXCLUDE>.VVVV.DDDD'
```

The following data is from the card's CISTPL_MANFID tuple:

VVVV = Manufacturer ID (TPLMID_MANF field), hex characters

DDDD = Manufacturer Information (TPLMID_CARD field), hex characters

Example:

```
PROD: 'PCMCIA.1234.0032'
```

Handling Special Characters in a PROD String

The DDI/LDI parser interprets an asterisk (*) or question mark (?) as a wildcard character, a period (.) as a field delimiter, and "\n" and "\t" as newline and tab characters. If you want to keep any of these interpreted characters as literal characters in the PROD string, you must precede (escape) each with a pound sign (#) so the parser will not strip or interpret them. Examples of handling these special characters are shown below.

Literal Character	Example String
* or ?	Escape the character with a pound sign, such as: PROD: ' SCSI.00.venID.devID.V123#* '
\ (backslash)	Escape (precede) a "\n" or "\t" with another backslash, such as: PROD: ' SCSI.00.venID.devID.rev1\\NEW '
. (period)	Escape a period with a pound sign, such as: PROD: ' SCSI.00.venID.devID.V1#.23 '

Literal Character	Example String
# (pound sign)	Escape the pound sign with another pound sign, such as: PROD: ' SCSI.00.venID.devID.V##123 '

An asterisk (*) matches any and all characters from its position in the field to the end of a field. A question mark (?) matches a single character located at its position in the field. Especially for EISA cards, if your driver can control different OEM revisions, you should use a wildcard to match the different revisions (for example, PROD: ' EISA.NVL.070.? '). Wildcarding is allowed for all fields except <INCLUDE/EXCLUDE>. Especially for EISA cards, if your driver can control different OEM revisions, you should use a wildcard to match the different revisions (for example, PROD: ‘ EISA.NVL.070.? ‘). ?

Conditional SCSI and IDE Device Driver Module Support

Conditional device driver module support refers to an extension to the PROD: label that will be added to disk driver descriptions (DDI files) for NWPA CDMs. Used in this manner, the PROD: label indicates which modules to select and load as a result of a SCSI inquiry or an IDE identify command. The above description of escape sequences and wildcards applies to SCSI and IDE PROD: string fields as well.

IMPORTANT: For NWPA CDMs, the PROD: label is required in the DDI file.

Example

The following assumes an adapter driver (HAM) has been loaded. Install will load the HAM and get information from NWPA to create a list of device types associated with the adapter. Next, Install examines all available DDI files for CDMs and tries to match the device type and module. It then loads the module(s) that best match the devices found.

For example, assume a tape device was found after doing a SCSI inquiry. Following is a driver description for TAPEDAI. The inquiry command will find a match in the TAPEDAI.DDI file and then load TAPEDAI.DSK, as illustrated below.

```
Driver TAPEDAI
{
...description and other labels...
PROD: ' SCSI.INCLUDE.01. *. *. * '
...parameters...
}
```

The PROD label with the SCSI designation indicates the device types supported by the TAPEDAI driver.

SCSI Extension Syntax

The complete syntax for the SCSI extension of the PROD: label is shown below. (The PROD: label may exist multiple times for different combinations of type numbers, device product IDs, etc.)

```
PROD:  ’SCSI.<INCLUDE|EXCLUDE>.
   <device type number>.<device vendor ID>.
   <device product ID>.<device product revision>’, ’...’
```

Field	**Description**
INCLUDE	If Install finds a match, the driver will be selected.
EXCLUDE	If Install finds a match, the module will not be selected. If neither INCLUDE nor EXCLUDE is specified, INCLUDE is assumed. (If both INCLUDE and EXCLUDE are used - which is not generally recommended - EXCLUDE takes precedence.)
<device type number>	Device or hardware type listed below; should correspond to the Peripheral Device Type obtained from a SCSI Inquiry: 00 Direct-access device (e.g., magnetic disk) 01 Sequential-access device (e.g., magnetic tape) 02 Printer device 03 Processor device 04 Write-once device (e.g., some optical disks) 05 CD-ROM device 06 Scanner device 07 Optical memory device (e.g., some optical disks) 08 Medium Changer device (e.g., jukeboxes) 09 Communications device * Wildcard, matches any device type
<vendor ID>	Vendor identification string; 1-8 bytes, ASCII. An asterisk (*) matches any vendor ID. <vendor ID> should correspond to the Vendor Identification obtained from a SCSI Inquiry.

Field	Description
<product ID>	Product identification string; 1-16 bytes, ASCII. An asterisk (*) matches any product identifier. <product ID> should correspond to the Product Identification obtained from a SCSI Inquiry.
<product revision>	Product revision string; 1-4 bytes, ASCII. An asterisk (*) matches any product revision. <product revision> should correspond to the Product Revision Level obtained from a SCSI Inquiry.

IDE Extension Syntax

The complete syntax for the IDE extension of the PROD: label is shown below. (The PROD: label may exist multiple times for different combinations of type numbers, model numbers, and firmware revision levels.)

```
PROD: 'IDE.<INCLUDE|EXCLUDE>.
   <device type number>.<model number>.<firmware revision>', '...'
```

Field	Description
INCLUDE	If Install finds a match, the driver will be selected.
EXCLUDE	If Install finds a match, the module will not be selected. If neither INCLUDE nor EXCLUDE is specified, INCLUDE is assumed. (If both INCLUDE and EXCLUDE are used - which is not generally recommended - EXCLUDE takes precedence.)
<device type number>	Device or hardware type listed below; should correspond to the Peripheral Device Type listed. 00 Direct-access device (e.g., magnetic disk) 01 Sequential-access device (e.g., magnetic tape) 02 Printer device 03 Processor device 04 Write-once device (e.g., some optical disks) 05 CD-ROM device 06 Scanner device 07 Optical memory device (e.g., some optical disks) 08 Medium Changer device (e.g., jukeboxes) 09 Communications device * Wildcard, matches any device type Note that these type identifiers are the same as those for SCSI.

Field	Description
<model number>	Model number string; 1-40 bytes, ASCII. An asterisk (*) matches any model number identifier. <model number> should correspond to the Model Number obtained from an IDE Identify.
<firmware revision>	Firmware revision string; 1-8 bytes, ASCII. An asterisk (*) matches any firmware revision. <firmware revision> should correspond to the Firmware Revision obtained from an IDE Identify.

Examining the Driver Description Files

After all adapter drivers have been selected and copied, Install will load the adapter drivers. For each adapter driver, Install will then do a SCSI inquiry or an IDE identify.

For each device found during the SCSI inquiry or IDE identify, Install will search all available driver description files for all the PROD: labels. If the driver matches an EXCLUDE item, that driver description will be skipped. Alternatively, if the driver matches an INCLUDE item, it will be given a fit score.

There may be multiple matches of a device in different driver descriptions. These will be resolved as follows: exact matches get a higher fit score than wildcards, and exact field matches of the first fields (in the PROD: declaration) get a higher score than later ones. The driver description with the highest score will be selected and loaded (if not already loaded).

SProd

The SProd tag is included in the LDI/DDI specification to help end users better identify your company's hardware during NetWare installation. It allows a product description string to be associated with each product ID listed under the Prod tag. The SProd tag consists of a single '[vendor name][~][module name]' syntax. The SPROD field must contain an equal number of strings separated by commas to the number of Prod ID strings in the Prod tag. The Prod ID Description string in the SPROD field is correlated to its corresponding Prod ID string in the PROD field by falling in the same position of their respective lists. If no Prod IDs exist in the PROD field, then a single Prod ID Description must be included in the SPROD field as the default. This string will be used for non-detectable drivers. The total length of a Prod ID Description string in the SPROD field must not exceed 50 characters in length.

Syntax

```
;@#  SPROD:  <'[vendor  name][~][module  name]'><,<'[vendor  name][~][module
name]'>,<'...'>
```

Some rules for using the SProd tag are listed below.

- The SProd tag must be located within the Driver {} structure after the Prod tag.
- Each line of the SProd tag must be preceded by the ;@# delimiter to allow backward compatibility with NetWare 4.11 autodetection parsers.
- Each string must be delimited by single quotes, separated from other strings by commas, and may not exceed 50 characters in length.
- Your company name MUST be the FIRST part of each string.
- At the very least, this string must contain a generic phrase stating the company name and type of product. (i.e. 'Novell~100MB Ethernet LAN card').
- Separate the company name from the rest of the string with a '~' (tilde) character.

Example

Shown below is an example for the Jimmy Dean PORK.LAN driver.

```
Driver PORK
{
Par: 2.00
Desc: $Pork_Description
Help: $Pork_Help
File: PORK.LAN

Prod: 'PCI.9001.3401.*.*.*'
      'PCI.9001.3402.*.*.*'
      'PCI.9001.3403.*.*.*'
      'PCI.9001.3404.*.*.*'
;@# Sprod: 'Jimmy Dean~100MB Ethernet card',
;@#        'Jimmy Dean~10/100 PCI Ethernet',
;@#        'Jimmy Dean~100 Embedded Controller',
;@#        'Jimmy Dean~10T Adapter'

Prompt SLOT REQ
{
}
}
```

CProg (Configuration Program)

Example:

```
CProg: CSL.NLM
```

The *CProg* label specifies a configuration executable and contains the name of an NLM that performs the configuration.

File

Example:

```
File: NE2000.LAN
```

If the *File* label is present, the installation utility on the distribution medium looks for a driver file with the indicated name. If the *File* label is not present, the installation utility uses the description file root name with a default extension according to the type of driver (LAN driver, disk driver, platform support module, etc.).

OFiles (Other Associated Files)

Example:

```
OFiles: FIRMLOAD.COM, MONT400.BIN
```

If the *OFiles* label is present, a comma-separated list of filenames must also appear on the same line. When the primary installation/configuration utility copies the driver file, it will also copy these associated files.

NOTE: If you have *OFiles* dependencies, be sure to include the dependent files with the .LAN, .DDI, .HAM, and .LDI files. If the *Ofiles* dependencies are for server only and not needed or supported for Client32, then the LDI file should be modified to reflect this. This means there may be a need to have two separate LDI files for one driver; one LDI file for the server and one for the client.

Timeout

Example:

```
Timeout: 20
```

If the *Timeout* label is present, it must be followed by a decimal number. This decimal number indicates the maximum time in seconds that the installation utility waits before it determines that a driver failed to load and reports an error to the user. If this label is not specified, the maximum wait time defaults to 5 seconds.

Alias

The Alias tag is used to correlate old drivers to the current driver for upgrade installation purposes and is only used in LDI/DDI files included in the NetWare 5 red box. If the name of a driver for a non-detectable HBA or NIC changes or a new driver with a different name replaces the old driver, INSTALL.NLM will use this information coupled with the load line from the STARTUP.NCF or AUTOEXEC.NCF to determine which driver to load.

NOTE: The Alias tag must be preceded by the ;@# delimiter to allow backward compatibility with NetWare 4.11 autodetection parsers.

Syntax

```
;@# ALIAS: <olddriver>, ...
```

Example 1

```
;@# ALIAS: olddrv1.ham, olddrv2.dsk, olddrv3.dsk, olddrv4.dsk
```

Example 2

```
;@# ALIAS: CNE2100.LAN, CNE1500T.LAN, NE2100.LAN, NE1500T.LAN
```

Example 3

```
;@# ALIAS: CNE2100.LAN
```

Client

The Client tag is used to correlate compatible client drivers to the server driver for auto-detection purposes. When a network install is performed on a system that has a non-detectable LAN card, INSTALL.NLM uses the name of the client LAN driver and a database containing the correlation information to select which LAN driver to load. The correlating information must be maintained in the LDI/DDI file by the driver writer.

NOTE: The Client tag must be preceded by the ;@# delimiter to allow backward compatibility with NetWare 4.11 autodetection parsers.

Syntax

```
;@# CLIENT: <clientdriver>, ...
```

Example

```
;@# CLIENT: CNEAMD, NE1500T, NE2100
```

HProd

Driver ID - The Novell assigned number of the driver associated with the adapter that hosts the device in question. These values are of length "WORD" (16 bits) in HEX format. The final digit may be wild carded with an asterisk ('*').

This keyword and syntax is placed in the .ddi file of a CDM when the CDMwill only support a device if it is hosted by an adapter whose driver has one of the given driver ids.

Syntax

```
;@# HPROD: <driver id><, <driver id>, ...>
```

Example

```
;@# HPROD: 2384, 2542, 245*
```

3

Driver Section-Parameters

Each of the driver's configurable parameters must be defined in the driver description by using one of the parameter types detailed in this section. Parameter specifications define the configurable parameters that the driver needs. A parameter specification includes several components: the parameter values, the presentation to the user, and the output format. The output format controls how the server driver information will be written to the command line.

Parameter Types

Three types of parameters are allowed; two are general parameters, and one is a special purpose parameter.

The two general parameter types are *PROMPT* and *LIST.* They can occur more than once in a driver description. They occur once for every user-configurable driver parameter. Both parameters contain fields for a parameter description and help text, dependency expressions, and output format specification. You can also specify a default value for the parameter, as well as permissible values from which the user can choose.

FRAME, the special purpose parameter type, can be declared only once within a single driver description. This parameter defines the frame types that are supported by the MLID. As with the general parameters, the *FRAME* parameter allows a description, help, and a dependency expression. However, this parameter uses a default method for input and output.

General Parameter Definitions

The general definitions that apply to all three parameters are described in this section. The following sections then provide the specific syntax for each parameter.

The parameter name is a case-sensitive string of from 1 to 16 characters. The parameter name is not displayed on the monitor. It is used only to reference another parameter's value in a dependency expression and to allow the installation/configuration utility to distinguish between driver descriptions. A parameter name can occur only once within a single description.

All configurable parameters can have a default value of UNDEFINED. This value indicates that no initial value is specified. If the parameter value remains UNDEFINED, the driver can determine the appropriate values automatically.

Counting of parameters (e.g. for specifying the DEFAULT) is ALWAYS done relative to 1 (one), NEVER relative to zero. The first item in a LIST, for example, is always value 1, never value 0. There has been confusion on this issue in the past; any utilities that do not count relative to 1 are out-of-specification.

A parameter can exist in one of three states: HIDDEN, REQUIRED, or OPTIONAL. The parameter state affects user input and output as follows:

HIDDEN-Indicates that the parameter is invisible to the user.

REQUIRED-Allows the installation program to determine which parameters are required by the driver at load time. The parameter is displayed, and a valid value must be specified for the parameter (either a default value or a value entered by the user). Output is always generated.

For example, if a driver has a REQUIRED port parameter, the user may not exit the parameter form until a valid value is selected for the parameter. The string, "PORT=xxxx" will always be generated on the command line after the "LOAD <driver>..." string.

PROMPT, LIST, or *FRAME* parameters that are specified as REQUIRED, but have only one valid choice (the default value), have the following unique features: (1) the parameter is not displayed to the user, because the user has no choice to make, and (2) the parameter will generate output. This feature creates an "invisible" parameter that generates output.

OPTIONAL-Signifies the parameters that are allowed but are not required by the driver. The parameter is displayed to the user, but no input is required from the user.

If the parameter has no default value specified, the user may leave it unspecified. If the value of a parameter is not specified, or if a valid default is deleted by the user, no output is generated (in other words, no output for the parameter is displayed on the command line).

If the parameter has a default value specified, and the user accepts the default, no output will be generated for the parameter. Otherwise, if the user changes a parameter to a defined value different from the default, output will be generated. A default value should be specified for an optional parameter if, and only if, the driver will default the parameter to that value.

A dependency expression decides the state of a parameter under various conditions. A parameter state can be specified as unconditionally OPTIONAL or REQUIRED. It can also be specified as conditionally OPTIONAL, REQUIRED, or HIDDEN and depending on the value of other parameters. If a parameter has no dependency, its state defaults to unconditionally OPTIONAL. Refer to "Dependency Expression" for a more detailed description of parameter states with dependency expressions.

The PROMPT Parameter

PROMPT obtains user input for a configurable parameter. The parameter can be a custom parameter or a local predefined parameter (see "Local Predefined Parameters").

The installation/configuration utility uses the specified *Description* string and *Default* value (if any) to prompt the user to enter a value for the parameter. The user can then accept the default value or choose another value from a specified set. The "Syntax" section contains a template for using a *PROMPT* parameter. The *Description* and *Type* fields are required; all other fields are optional.

Syntax

```
PROMPT <parameter_name> <dependency expression>
{
   Description:     "<description text>"
   Help:            "<multi-line help text>"
   Type:           STRING (max_chars) or
                   HEX (max_digits) or
                   DECIMAL (max_digits)

   Values:         <minimum value> - <maximum value> or
                   <value 1>, <value 2>, ... <value n>

   Default:        <default value> or UNDEFINED

   ReservedLength: <hexadecimal length of values reserved or <name>>

   OutputFormat:   '<any string with a %s>'
}
```

The following sections describe the different portions of the above template.

Dependency Expression

PROMPT parameters can be assigned a state of REQUIRED, OPTIONAL, or HIDDEN. You can use a conditional dependency expression to determine the parameter state. You can also use *PROMPT* parameters in the dependency expressions for other parameters. When used in dependency expressions, the *PROMPT* parameter value is the value selected by the user (or UNDEFINED if no value was specified). Refer to "Dependency Expression" for a more detailed description of the dependency usage.

Description

The <*description text*> is the prompt for the user configurable parameter. The description string can be a maximum of 40 bytes.

Help

The *Help* text can be longer than one line and can be a maximum of 1,500 bytes.

Type

Type specifies whether a value is interpreted as string (case-sensitive-install utilities must not UPPERCASE strings), hexadecimal, or decimal. You can optionally specify the maximum number of characters or digits for that value. (Parameter values can have a maximum of 35 characters or digits.) If the maximum length, *<max_chars>* or *<max_digits>,* is not specified, it defaults to the maximum element size in the list of *Values* (described below). If no values are specified, 8 characters are used for parameters having predefined names (see "Local Predefined Parameters" in Chapter 4), and 35 characters are used for parameters without predefined names.

Example 1 below would allow the user to enter up to 35 characters and generate the output as indicated. Example 2 would allow only 10 characters.

Example 1

```
PROMPT param2
{
   Description:  "A string parameter"
   Type:         STRING
   Default:      'my_string'
   OutputFormat: 'String=%s'
}
```

Example 2

```
PROMPT param2
{
   Description:  "A string parameter"
   Type:         STRING (10)
   Default:      'my_string'
   OutputFormat: 'String=%s'
}
```

Values

This field indicates the allowable values for the parameter. The values are displayed on the console as the user highlights the parameter field. The values can be specified by using a range of values or by using a comma-separated list of values. The range or list of values to be displayed must be less than 70 characters long.

Default

The default value is optional and, if used, is displayed along with the description string as part of the parameter prompt. The default value must be of the specified *Type* and must be an element indicated in the *Values* range or list. The default value can also be UNDEFINED. An absent *Default* label is identical in function to a default value of UNDEFINED.

ReservedLength

The *ReservedLength* label is generally ignored, even if it is present. The exception to this is the cases of the *PORTx* and *MEMx* reserved parameters. In these cases, the server environment requires these labels (see "Local Predefined Parameters"). If this label is present, it must contain either a single hexadecimal constant or the name of another parameter whose value (when set) will be used as the reserved length of this parameter. The *ReservedLength* value has the type specified by the *Type* label.

OutputFormat

The *OutputFormat* string describes the way the output is to be generated from the final parameter value. The output is displayed on the server monitor's command line. The format string can contain a maximum of one %s. The output string is created by replacing the %s with the parameter value. Output is generated according to the rules described in the section "General Parameter Definitions."

Example

The example below shows a sample PROMPT parameter block using the local predefined parameter, INT, and the resulting screen displayed in the installation utility.

NOTE: The Description, Help, and Type fields shown below are included to illustrate their use in the PROMPT example. For the local predefined INT parameter, these fields have default values and are not usually required.

```
PROMPT INT REQUIRED
{
   Description:  "Interrupt"
   Help:          "Select the primary interrupt number."
   Type:          HEX(1)
   Values:        2, 3, 5, 7
   Default:       3
   OutputFormat: 'INT=%s'
}
```

Resulting Screens

```
Driver NE2000 parameters

Supported values: 2,3,5,7
Default value:3

Select the primary interrupt number.
```

The LIST Parameter

LIST obtains user input for a configurable parameter. *LIST* is similar to *PROMPT*, with the exception that the user selects an option for the parameter from a menu of valid choices.

The installation utility uses the parameter *Description* and the *Default* choice description (if any) to prompt the user for a selection. The user can then accept the default choice or select another from the menu of choices for the parameter. The "Syntax" section contains a template for using a *LIST* parameter. The *Description* parameter and *Choice* fields are required; all others are optional.

Syntax

```
LIST <parameter_name> <dependency expression>
{
   Description:    "<parameter description text>"
   Help:           "<multi-line help text>"
   CDescription:   "<choice #1 text>"
   Choice:         <choice #1 value> or UNDEFINED
   .
   .
   .
   CDescription:   "<choice #n text>"
   Choice:         <choice #n value> or UNDEFINED
   Default:        <1 to n> or UNDEFINED
   OutputFormat:   '<format string with a %s>'
}
```

The various parts of the above template are described in the following sections.

Dependency Expression

LIST parameters can be assigned a state of REQUIRED, OPTIONAL, or HIDDEN. A conditional dependency expression can be used to determine the parameter state. *LIST* parameters can also be used in the dependency expressions for other parameters. When used in dependency expressions, the *LIST* parameter value is a decimal number indicating the index of the *Choice* selected by the user (or UNDEFINED if no choice was specified). Refer to "Dependency Expression" for a more detailed description of the dependency usage.

Description

The <*parameter description text*> is the prompt for the user configurable parameter. The description string can be a maximum of 40 bytes.

Help

The *Help* text can be longer than one line and can be a maximum of 1,500 bytes.

Choice and CDescription

The installation utility uses *Choice* or *CDescription* to create a menu of valid choices for the parameter. The description text string is typically enclosed in double quotes (if language translation is supported), because the menu choices can usually be translated to different languages.

The installation utility uses the *Choice* field to build the command line entry (see the "OutputFormat" description below). Choice values can be any string, including the null string, or can be UNDEFINED. Ranges are *not* allowed. (For example, Choice: 1-50 is illegal.) Typically choice values will not be enclosed in double quotes, because this results in language specific command line or configuration file parameters.

If the *CDescription* is not provided for a particular *Choice*, the menu text will be the choice string itself. The number of pairs of choice descriptions pairs implies the number of choices. The maximum field width for any given choice description is 35 characters.

Default

The *Default* value is a decimal number indicating the index of the default choice. It must be in the range of 1 to the number of choices. The default value can also be UNDEFINED, which means that none of the choices are initially selected. An absent *Default* label is identical to a default value of UNDEFINED.

The default value is optional and, if used, the corresponding *CDescription* is displayed at the parameter prompt to indicate the default choice. If the default value is specified as UNDEFINED then "(not specified)" will be displayed.

OutputFormat

The *OutputFormat* label describes the way the output is to be generated from the selected parameter choice. The output is displayed on the server monitor's command line. The format string can contain a maximum of one %s. The output string is created by replacing the %s with the selected *Choice* string. The output is generated according to the rules described in the section "General Parameter Definitions."

Example

The example below shows a sample *LIST* parameter block and the resulting screens displayed in the installation utility.

```
LIST Attach_Mode OPTIONAL
{
   Description:   "FDDI Station Attach Mode"

   Help:          "If there is a secondary board in
                  your machine, you may wish to
```

```
                 override the auto sense attachment.
                 Select the correct mode from the list."

   CDescription:  "Single Attach"
   Choice:        '1'

   CDescription:  "Dual Attach"
   Choice:        '2'

   CDescription:  "Auto Sense"
   Choice:        UNDEFINED

   Default:       3

   OutputFormat:  'ATTACH_MODE=%s'
}
```

The FRAME Parameter

FRAME allows the user to select the MLID's default frame types. The installation utility uses the *Description* parameter and the *Default* values to display a list of frame names. The user can add or delete frames from the list. In the server environment, a default logical name can also accompany each frame type. Only the *Choice* fields shown below are required; all other fields shown are optional.

Syntax

```
FRAME <parameter_name> <dependency expression>
{
   Description:    "<parameter description text>"
   Help:           "<multi-line help text>"
   CDescription:   "<frame #1 description text>"
   Choice:         <frame #1 type string>
   .
   .
   .
   CDescription:   "<frame #n description text>"
   Choice:         <frame #n type string>

   Default:        <1,...,n> or UNDEFINED
   OctetBitOrder:  <LSB or MSB>
}
```

The following sections describe the various parts of the above template.

Dependency Expression

If the *FRAME* parameter state is OPTIONAL, the user does not need to indicate any values. If the parameter is REQUIRED, the user must select at least one frame type.

Only one *FRAME* parameter with multiple frame types can be visible (REQUIRED or OPTIONAL) to the user. Multiple *FRAME* parameters can be declared, but only one block can be active; all others must be HIDDEN. (If the parameter is HIDDEN, nothing will be presented to the user, and no output will be generated for that parameter.)

A *FRAME* parameter can also be used in dependency expressions for other parameters. When used in dependency expressions, the parameter's value is a nonzero decimal number indicating the number of frame types selected (or UNDEFINED if no frame types were specified). For a more detailed description of dependency usage see "Dependency Expression".

Description

The <*parameter description text*> is the frame type prompt. The description string can be a maximum of 40 bytes. If the description is not present, the text will default to "Frame Types."

Help

The *Help* text can be more than one line long and can be a maximum of 1,500 bytes. If the help text is not present, the default Frame help text is displayed (see the "Default Help Information" table in the "Local Predefined Parameters" section later in this document).

Choice and CDescription

The *Choice Description (CDescription)* fields create the list of default frame types. The maximum field width for any given frame description is 35 characters. If *CDescription* is not provided for a particular *Choice*, the text displayed to the user will be the frame type string itself.

Choice values can be any string, but should be strings that are understood by the MLID and the protocols that will be used. Each *Choice* can appear only once in the list. Typically frame types are not enclosed in double quotes, because this would result in language specific command line or configuration file parameters.

Default

The *Default* field contains a list of numbers corresponding to default frames, where 1 corresponds to the first frame type, 2 to the second, etc. The value may also be UNDEFINED, indicating that no default frame names are initially selected. An absent *Default* label is identical to a default value of UNDEFINED.

OutputFormat

The output for the FRAME parameter is implied (the OutputFormat label is not used). In the server environment, the load driver command will be reiterated at the command line for each frame type selected.

OctetBitOrder

This label is optional and, if used, should only be present for Token-Ring and PCNII networks. The *OctetBitOrder* field allows the user to specify whether network addresses are in canonical or noncanonical (LSB or MSB) formats (see the *ODI Specification Supplement: Canonical and Noncanonical Addressing*.) The value associated with this label will be the default value for all frame types.

Example (for Assembly Language MLIDs)

The example below shows a sample FRAME parameter block and the resulting screen displayed in the installation utility.

```
FRAME FrameSelect
{
   Help:          "The driver defaults to the 802.2 frame type.
                  You can optionally remove this frame type
                  and/or add the 802.3, 802.2 SNAP, and/or
```

```
                 Ethernet II frame types."

   CDescription: "802.2"
   Choice:       'Ethernet_802.2'

   CDescription: "802.3"
   Choice:       'Ethernet_802.3'

   CDescription: "802.2 SNAP"
   Choice:       'Ethernet_SNAP'

   CDescription: "Ethernet II"
   Choice:       'Ethernet_II'

   Default:      2
}
```

4

Predefined Parameters

Global Predefined Parameters

The following table lists the globally predefined parameters. These parameters are never displayed to the user (although in some cases the user will be prompted by the installation utility for the information needed to create them). They exist only to allow driver description and parameter dependency expressions to reference them. This makes the states of descriptions and parameters conditional upon the value of these global parameters. You can write dependencies assuming that these values will always exist and that they will never be UNDEFINED.

Name	Parameter Type	Possible Values
BUS	PROMPT, STRING	'ISA'
		’PNPISA’
		'MCA'
		'EISA'
		'PCMCIA'
		’PCI’ (includes CardBus)
GT_16	PROMPT, STRING	'TRUE'
		'FALSE'

The BUS Parameter

BUS indicates which bus architecture the installation/configuration utility is working with. (The is the bus architecture of the machine for which the command line information is being created.)

The GT_16 Parameter

If the *GT_16* parameter value is TRUE, the machine has more than 16MB of memory available for the driver to use.

Local Predefined Parameters

Predefined PROMPT Parameter

Some local parameters are standard. Therefore, these local parameters can have more specific meanings than the general parameter definitions mentioned previously. The following table lists the predefined *PROMPT* parameter names:

Name	**Meaning**
INT or INT1	Primary interrupt
INT2	Second interrupt
PORT or PORT1	Primary I/O port
PORT2	Second I/O port
MEM or MEM1	Primary memory address
MEM2	Second memory address
DMA or DMA1	Primary DMA address
DMA2	Second DMA address
SLOT	Machine slot number/ HIN number
NODE	Node address
RETRIES	Number of retries
CHANNEL	Channel number for adapters that use controllers

If a *PROMPT* parameter name is one of the predefined names listed above, all of the labels are optional, and will be defaulted if they are not specified. If a dependency expression is not declared, the parameter state will default to OPTIONAL.

If one or more of the fields are not specified for the local predefined parameters, the installation/configuration utility generates the following defaults:

Parameter Field	**Default Information**
Description:	(INT) "Interrupt number" (INT2) "Secondary interrupt number" (PORT) "Port value" (PORT2) "Secondary port value" (MEM) "Memory address" (MEM2) "Secondary memory address" (DMA) "DMA value" (DMA2) "Secondary DMA value" (SLOT) "Slot Number" (NODE) "Node Address" (RETRIES) "Number of Retries"
Help:	The default help information is listed following this table.
Type:	DECIMAL(8) for SLOT and RETRIES HEX(12) for NODE HEX(1) for INT and INT2 HEX(8) for all others
Values:	1-65535 for SLOT 0-99999999 for RETRIES 0-FFFFFFFFFFFF for NODE 0-F for INT and INT2 0-FFFFFFFF for all others
Default:	UNDEFINED
ReservedLength:	Not defaulted. This field must be specified for a *PORTx* or *MEMx* parameter.

Parameter Field	Default Information
OutputFormat:	(INT)'INT=%s' (INT2)'INT1=%s' (PORT)'PORT=%s' (PORT2)'PORT1=%s' (MEM) 'MEM=%s' (MEM2) 'MEM1=%s' (DMA) 'DMA=%s' (DMA2) 'DMA1=%s' (SLOT) 'SLOT=%s' (NODE)'NODE=%s' (RETRIES)'RETRIES=%s'

Default Help Information

Name	Text
INT INT1	"\nSelect the interrupt level that corresponds to the interrupt setting on the board or device.\n\n The interrupt setting must be unique (one not used by another device in the machine)."
INT2	"\nSelect the interrupt level that corresponds to the second interrupt setting on the board or device.\n\n The interrupt setting must be unique (one not used by another device in the machine)."
PORT PORT1	"\nSelect the port value (base I/O address) that corresponds to the port address setting on the board or device.\n\n Make sure the block of I/O addresses does not overlap the addresses of another device in the machine."
PORT2	"\nSelect the port value (base I/O address) that corresponds to the second port address setting on the board or device.\n\n Make sure the second block of I/O addresses does not overlap the addresses of another device in the machine."
MEM MEM1	"\nSelect the memory address that corresponds to the memory setting on the board or device.\n\n Make sure the block of memory addresses does not overlap the addresses of another device in the machine."

Name	Text
MEM2	"\nSelect the memory address that corresponds to the second memory setting on the board.\n\n Make sure the block of memory addresses does not overlap the addresses of another device in the machine."
DMA DMA1	"\nSelect the DMA channel that corresponds to the DMA setting on the board or device.\n\n Make sure the DMA (Direct Memory Access) channel does not conflict with that of another device in the machine."
DMA2	"\nSelect the DMA channel that corresponds to the second DMA setting on the board.\n\n Make sure the DMA (Direct Memory Access) channel does not conflict with that of another device in the machine."
SLOT	"\nSelect the slot number that corresponds to the expansion slot where the board or device is installed."
NODE	"\nDo not change this address unless you are prepared to administer local addresses according to the IEEE 802.2 specifications.\n\n The driver defaults to the node address on the board."
RETRIES	"\nThis number specifies the maximum number of times the driver will be instructed to retry a failed packet transmission."
FRAME	"\nSelect the frame type used by the protocol your network requires.\n\n If you select a frame type other than the default, configure both client and server to use the same frame type."

The INT Predefined Parameter

The following examples show how to use the predefined parameter *INT*.

Example 1

```
PROMPT INT
{
}
```

In Example 1, the parameter description, output format, type, and type length use default values, as follows:

Description: "Interrupt number"
OutputFormat: 'INT=%s'
Type: HEX (1)
Values: 0-F

Default: UNDEFINED

The help information displayed for this parameter would be:

```
"Select the interrupt level that corresponds to the interrupt setting on the
board or device.
The interrupt setting must be unique (one not used by another device in the
machine).''
```

In Example 1, the installation/configuration utility will not allow the user to enter an interrupt value that is already taken, even though the taken values are not specified in the help text.

Example 2

```
PROMPT INT
{
   Type:     DEC (2)
   Values:   2, 3, 4, 5, 10
   Default:  3
}
```

In Example 2, assume that another MLID is already using interrupt 3. The parameter description and output format default as follows:

Description: "Interrupt number"
OutputFormat: 'INT=%s'

The help information displayed for this parameter would be:

```
Permissible values: 2, 4, 5, 10
Default value: 3 (not-selectable)

Select the interrupt level...
```

Please note in Example 2 that the explicitly declared field “Type: DEC (2)” allows 2 digits to be defined for interrupt 10. Also note that if any parameter field is explicitly declared do not use the default information as defined in Tables 4 and 5.

The PORTx and MEMx Predefined Parameters

In the case of *PORT*x and *MEM*x, the *ReservedLength* is a required label and must be present as part of the *PROMPT* parameter. *ReservedLength* determines whether the specified group of port or memory addresses are available, and prevents the user from entering values that are taken. *ReservedLength* must contain either a single hexadecimal constant or the name of another parameter whose value (when set) will be used as the reserved length of this parameter. If the parameter is *PORTx*, the reserved length represents a range of port values in bytes. If the parameter is *MEMx*, the reserved length represents a range of memory addresses in paragraphs (groups of 16 addresses).

Example MLID

```
; CNE2000.LDI
;
; Novell NE2000 Network Interface Cards.
;
; VeRsIoN=1.03 Novell LAN Installation Information File for CNE2000.
; CoPyRiGhT=(c)Copyright 1996, 1997 by Novell, Inc.  All rights reserved.

VER: 1.03
SYN: 1.00

DR NOVELL_CNE2000
{
   DES:    $CNE2000_1
   HELP:   $CNE2000_2
   PAR:    1.00
   FILE:   CNE2000.LAN
   PROD:   ’EISA.ISA.871.2’, ’EISA.NVL.150.1’
   ;@# SPROD: ’Novell~NE2000 or NE2000 Plus’,
   ;@#        ’Novell/Anthom~NE2000 or compatible’

   LIST Adapter_Bus_Type OPTIONAL
      {
         Description:  $CNE2000_15
         Help:         $CNE2000_16

         CDescription: $CNE2000_17
         Choice:       ''
         CDescription: $CNE2000_18
         Choice:       'ISA'

         Default:      1
         OutputFormat: '%s'
      }
      PR SLOT
      IF ((BUS == ISA) OR (BUS == EISA)) HID
      ELSE REQ
   {
         VAL: 1-65535
   }

   PR INT
      IF ((BUS == ISA) OR (BUS == EISA)) REQ
      ELSE HID
   {
         VAL: 2, 3, 4, 5, A, B, C, F
         TYP: HEX(1)
         DEF: 3
   }

   PR PORT
      IF ((BUS == ISA) OR (BUS == EISA)) REQ
      ELSE HID
   {
         VAL: 300, 320, 340, 360, 240, 280, 2C0
         DEF: 300
         RES: 20
   }

   PR NODE
   {
   }

   FR FrameSelect
   {
         CH:  'Ethernet_802.2'
         CH:  'Ethernet_SNAP'
         CH:  'Ethernet_II'
         CH:  'Ethernet_802.3'
         DEF: 1,4
   }

   PR RETRIES
   {
         VA: 0-255
         DEF: 5
   }

   PR MEM
      IF ((BUS == ISA) OR (BUS == EISA)) OPT
      ELSE HID
   {
         VAL: D0000,D4000,D8000,DC000,C0000,C4000,C8000,CC000,
              E00000,E20000,E40000,E60000,E80000,EA0000,EC0000,
              EE0000,F00000,F20000,F40000,F60000,F80000,FA0000,
              FC0000,FE0000
         DEF: UNDEFINED
         RES: 400
   }

}
```

Note that in the PR MEM section above, the VAL expression is all on one line in the original source file; the line was broken up to fit this document's page size.

5

Driver Installation Information Template

The following is an installation information file template for MLIDs. This template can also be used for server disk drivers by deleting LAN-specific parameters such as node, frame, and protocol.

All translatable strings in the following template are surrounded by double quotes, as in "<translatable string>". If you want to add new strings, follow the convention to surround only strings that should be translated in double quotes. All other strings should either not have quotes or should be surrounded by single quotes. Write and test your information file and check it with (but not concatenated to) the driver.

Replace all strings shown in the <xxxx> format with the appropriate string. You can delete any description line that you do not need. You can also add any additional custom parameters that you need.

```
; VeRsIoN=<version number and extra info for support utilities>
; CoPyRiGhT=(c)Copyright <years> by <vendor>. All rights reserved.

VER: <version number>
SYN: 1.00
; Place introductory comments here.
; Keep comments and white space to a minimum.

DR <Driver Description Name> <Dependency Expression>
{
   DES: "<text description>"
   HELP:"<multi-line help text>"
   PAR:X.xx
   PROD:<product ID string>
   ;@# SPROD:  <product ID description>
   CP:<configuration NLM name>
   PATH:<path on media>
   FILE:<file name on media>
   OF:<other assoc. files>
   TIME:<driver load timeout value>
   ;@# ALIAS:  <alias from old driver to new driver>
   ;@# CLIENT: <client driver>

; You will need only some of the following for any given driver description.
; Delete those not needed and edit those which you do need to correctly
; describe your adapter and driver. You can add custom parameters to
     describe
; driver parameters not covered here. Most likely, one or more of the
; parameters (INT, PORT, MEM, etc.) will be indicated as REQUIRED".

PR INT
{
   VAL: 3,5,7,9
   DEF: 9
}
PR PORT
{
   VAL: 300,310,320
   DEF: 300
   RES: 8
   ; (continued)
   ; ReservedLength for port specifies a range in single value increments
}
PR MEM
{
   VAL: C000,C800,D000,D800
   DEF: C000
   RES: 800
; ReservedLength for memory specifies a range in paragraphs
}
PR DMA
{
   VAL: 1,3,5,7
   DEF: 3
}
PR SLOT
{
   VAL: 1-8
}
PR NODE
{
}
FR FrameSelect
{
   HELP:    "The defaults are set to 802.2 and 802.3 frame types.\n\n
           We strongly recommend that you select at least 802.2.
           For existing networks, select both 802.2 and 802.3"
   CD:      "802.3"
   CH:      'Ethernet_802.3'

   CD:      "802.2"
   CH:      'Ethernet_802.2'
   CD:      "802.2 SNAP"
   CH:      'Ethernet_SNAP'
   CD:      "Ethernet II"
   CH:      'Ethernet_II'
   DEF:     1,2
   ; For Ethernet server drivers, set the default to 802.2 and 802.3.
   ; For all other drivers, set the default to whatever the default
   ; is in the driver, and change the help text accordingly.
}
}
```

6

Sample DDI/LDI files for Family Drivers with ISA Support

First Example

```
; VeRsIoN=<version number and extra info for support utilities>
; CoPyRiGhT=(c)Copyright <years> by <vendor>. All rights reserved.

VER: <version number>
SYN: 1.00
; Place introductory comments here.
; Keep comments and white space to a minimum.

Driver FAMILY
{
   Description:   "Family XYZ Adapter driver."
   Help:          "This driver supports the PCI, PnP ISA and legacy ISA
                  versions of the XYZ adapter."

   Par:2.00

   File:FAMILY.HAM

   Prod: 'PCI.xxxx.xxxx.0000.0000.*', 'PNPISA.xxx.xxx.x'

   ;@# Sprod:   ’Family Corporation~PCI XYZ Adapter’,
   ;@#          ’Family Corporation~PnP ISA XYZ Adapter’

   ;@# ALIAS:  olddrv1.DSK, olddrv2.DSK, olddrv3.DSK
   LIST Adapter_Bus_Type OPTIONAL
   {
      Description:"Is the card a <legacy ISA Card model num>?"
      Help: "A value of 'Yes' is necessary to support legacy ISA cards.
            Plug and Play ISA is not considered legacy ISA."
      CDescription:"No"
      Choice:      ''
      CDescription:"Yes"
      Choice:      'ISA'
      Default:     1
      OutputFormat:'%s'
   }

   PROMPT SLOT if (Adapter_Bus_Type == 2) HID else REQ
   {
   }

   PROMPT INT if (Adapter_Bus_Type == 2) REQ else HID
   {
      Values: 2, 3, 4, 5
      Default:2
   }

   PROMPT PORT if (Adapter_Bus_Type == 2) REQ else HID
   {
      Values:        300, 320, 340, 360, 240, 280, 2C0
      Default:       300
      ReservedLength:8
   }
}
```

Second Example

```
; VeRsIoN=<version number and extra info for support utilities>
; CoPyRiGhT=(c)Copyright <years> by <vendor>. All rights reserved.

VER: <version number>
SYN: 1.00
; Place introductory comments here.
; Keep comments and white space to a minimum.

Driver FAMILY
{
   Description:"Family XYZ Adapter driver."
   Help: "This driver supports the PCI, Plug and Play ISA,
         and legacy ISA versions of the FAMILY adapter."
   Par:  2.00
   File: FAMILY.LAN
   Prod:'PCI.xxxx.xxxx.0000.0000.*', 'PNPISA.xxx.xxx.x'

   ;@# SProd:   ’Family Corporation~PCI FAMILY Adapter’,
   ;@#          ’Family Corporation~PnP ISA FAMILY Adapter’
   ;@# CLIENT:  clientdrv1, clientdrv2, clientdrv3
   ;@# ALIAS:   olddrv1.LAN, olddrv2.LAN, olddrv3.LAN, olddrv4.LAN

   LIST Adapter_Bus_Type OPTIONAL
   {
      Description:"Is the card a <legacy ISA Card model num>?"
      Help: "A value of 'Yes' is necessary to support legacy ISA cards.
            Plug and Play ISA is not considered legacy ISA."
      CDescription:  "No"
      Choice:        ''
      CDescription:  "Yes"
      Choice:        'ISA'
      Default:       1
      OutputFormat:  '%s'
   }

   PROMPT SLOT if (Adapter_Bus_Type == 2) HID else REQ
   {
   }

   PROMPT INT if (Adapter_Bus_Type == 2) REQ else HID
   {
      Values: 2, 3, 4, 5
      Default:2
   }

   PROMPT PORT if (Adapter_Bus_Type == 2) REQ else HID
   {
      Values:        300, 320, 340, 360, 240, 280, 2C0
      Default:       300
      ReservedLength:8
   }

   PROMPT NODE
   {
   }

   FRAME FrameSelect
   {
      Choice:  'Ethernet_802.3'
      Choice:  'Ethernet_802.2'
      Choice:  'Ethernet_SNAP'
      Choice:  'Ethernet_II'
      Default: 2
   }
}
```

7

16-Bit DOS Client Installation Information (INS) File

If your 16-bit DOS ODI LAN driver will be installed with the installation utility, the utility must know how to set up the configuration file (NET.CFG). This means the utility must know each driver's parameter options and which choices the user must make. The driver installation information (.INS) file and the DRIVER.LST file provide installation utilities with this required information.

Each driver has only one .INS file even though the driver usually supports many boards. If the installation utility does not find an INS file for a driver, the utility does not create a NET.CFG file entry for the driver. The installation program uses the .INS file to prompt the user for the parameter options and values necessary to generate a NET.CFG file. Sample INS files are included at the end of this appendix.

Each driver distribution diskette may include one DRIVER.LST file. This file provides the installation program with a quick directory of all ODI drivers found on the distribution diskette.

General Rules

Your INS file must conform to the following rules and also to the conventions used in the examples.

- The INS file must not contain blank lines or tabs. Space characters are not permitted between fields unless otherwise specified.

- Items shown in angle brackets indicate that the user must supply information describing that aspect of the driver. For example:

  ```
  <DriverName>
  NE2000.COM
  ```

- Items shown in square brackets ([ ) and ( ]) are optional.

File Syntax

The format of the INS file for 16-bit DOS ODI drivers is shown below:

```
InS_StArT
<DriverName>
<Version>[,<AssociatedFileList>]
^<Board1 Description>
^<Board2 Description>
   $
   $
   $
[?Help Text>]
<Parameter1 Definition>
<Parameter2 Definition>
   $
   $
   $
InS_EnD
```

Signature Lines

Note the initial and final lines:

```
InS_StArT
$
$
$
InS_EnD
```

These lines illustrate the following crucial points:

- The InS_StArt and InS_EnD signature lines shown above bracket the driver installation information.

- The body of the INS file is preceded by the "InS_StArT" keyword and is terminated by the "InS_EnD" keyword.

Driver Name

The driver filename, including the .COM or .EXE extension, is the first line of the installation information. Installation uses the driver filename to generate the NET.CFG Link Driver <DriverName> command.

Version

The version field indicates the version of the INS specification to which the INS file is written. The version field enables installation utilities to properly handle any future changes to the specification. The version number format is 'X.X.' For this specification, the version is 1.1.

Associated File List

The version number may be optionally followed by a list of associated files separated by commas. The user must copy these files, along with the driver, to the area where the user is installing the client pieces. Examples of associated files include firmware BIN files and special configuration or diagnostic utilities for the adapter or driver.

Board Description

The Board Description(s) provides a list of adapters and related information that the DRIVER.LST file uses. (DRIVER.LST is described later in this appendix.) More than one board can be listed if a single driver will work with a given adapter and its clones. Place multiple board descriptions on separate lines in the DRIVER.LST file and only use them when the hardware options are identical for all the boards. The user generally knows what board he/she has installed, but not necessarily what driver he/she should use with that board.

Each Board Description is preceded by a caret (^) and has the following format:

```
^<BoardName,DriverName,BusCode[,ProductID]>
```

Example

```
^Novell NE2000,NE2000.COM,IEO
```

- BoardName is the full name of the network adapter. This field can contain spaces and can be a maximum of 48 characters long.
- DriverName is the filename (including the extension) of the board's driver. This field can be a maximum of 13 characters long.
- BusCode is a six character code depicting the bus type(s) supported for the adapter. The code is described below:

1st character	I (alpha)	supports ISA adapters
	0 (numeric)	does not support ISA adapters
2nd character	E (alpha)	supports EISA adapters
	0 (numeric)	does not support EISA adapters
3rd character	M (alpha)	supports MCA adapters
	0 (numeric)	does not support MCA adapters
4th character	A (alpha)	supports PCMCIA adapters
	0 (numeric)	does not support PCMCIA adapters
5th character	P (alpha)	supports PCI and CardBus adapters
	0 (numeric)	does not support PCI or CardBus adapters
6th character	V (alpha)	supports VESA Local Bus adapters
	0 (numeric)	does not support VESA Local Bus adapters

NOTE: Because all ISA boards work in EISA machines, 'I' and 'E' should both be used for ISA boards. This enables ISA boards installed in an EISA bus to appear in the installation utility. Keep in mind that ISA boards installed in EISA machines retain ISA functionality and features.

- The ProductID field is optional and only applies to MCA and EISA boards. This field contains the ID string that is stored in the POS registers of MicroChannel and in configuration registers of EISA machines. The installation utility uses this ID string to select the appropriate driver for the board found in the machine. This field is left blank for ISA boards.

Help Text

You can supply one line of optional help text after the board description lines. The help text is preceded with a question mark (?), can be up to 256 bytes in length (not including the question mark), and can contain spaces. The installation program automatically word wraps the help text. The width of the help windows will probably vary because installation programs on different platforms display a different number of characters on a line.

Parameter Definitions

The Parameter Definition section of the INS file specifies the configurable parameters for the driver and defines the valid options for each parameter. The parameter format is described in the next section.

If the network board can be configured by software and does not need to use the NET.CFG file, there should be no parameter definitions.

Parameter Syntax

The NET.CFG file is created with parameter keywords and values. The INS file can define standard and/or custom parameters. The standard driver parameter keywords are: BUS ID, IRQ, PORT, MEM, DMA, SLOT, NODE ADDRESS, and FRAME (see *ODI Specification: DOS Client HSMs, document version 4.0*). The INS file can specify as many parameters as are needed to describe the available hardware and software options.

A parameter definition has the following format:

```
<ParameterCode>(<ParameterKeyword>)<ParameterName>
[<Parameter Help Text>]
<Valid Parameter Options>
$
$
$
```

Parameter Code

The ParameterCode field is eight characters long. The characters are defined as follows:

First character

The code's first character specifies the prompt format the installation utility will use to display the parameter on the screen for the user. This character will also affect how the <*valid parameter options*> are indicated. The possible values for the first character are listed below.

! - Select only one option from the list.
* - Select more than one option from the list.
$ - User input within a range (slightly different from above).
# - Reserved for future use.

Second character

The code's second character indicates whether the user must select this parameter value or whether it is optional. This character also indicates whether a NET.CFG entry will be generated.

R (alpha) - Required. This parameter must be selected. A NET.CFG entry is always generated.

0 (numeric) - Optional. If nothing is selected, no NET.CFG entry will be generated.

Last Six Characters

The last six characters indicate the bus type dependency of the parameter. The installation utility uses this code to determine whether the parameter is to be displayed (used), depending on the bus type of the machine. The bus type characters are defined as follows:

3rd character	I (alpha)	supports ISA adapters
	0 (numeric)	does not support ISA adapters

4th character	E (alpha)	supports EISA adapters
	0 (numeric)	does not support EISA adapters
5th character	M (alpha)	supports MCA adapters
	0 (numeric)	does not support MCA adapters
6th character	A (alpha)	supports PCMCIA adapters
	0 (numeric)	does not support PCMCIA adapters
7th character	P (alpha)	supports PCI and CardBus adapters
	0 (numeric)	does not support PCI or CardBus adapters
8th character	V (alpha)	supports VESA Local Bus adapters
	0 (numeric)	does not support VESA Local Bus adapters

For example, a driver running on an ISA or EISA bus may require the PORT parameter. However, a driver running on an MCA machine may require only the SLOT parameter.

```
!RIE0000(PORT)Base I/O Port
$
$
$
!R00M000(SLOT)Slot Number
$
$
$
```

Parameter Keyword

The parameter's NET.CFG file keyword immediately follows the parameter code characters. The parameter keyword is enclosed in parenthesis and can be up to a maximum of 20 characters. The user-selected value for that parameter is placed in the NET.CFG file if no keyword (empty parenthesis) exists. These keywords can be custom keywords or any of the standard keywords listed in the table below.

Parameter Name

The parameter name follows the parameter keyword. This name is the title the installation utility displays to the user. The length of this field is limited to 30 characters. The parameter names are given in the table below.

Parameter Help Text

A single line of optional help text may follow the parameter line. This line is preceded with a question mark (?) and can be a maximum of 256 characters long. You should only use help text with custom keywords, because the installation utility supplies help text for each standard keyword. Any help text must follow the keyword line and precede the parameter value lines described next in the NET.CFG file. If help is supplied with a standard keyword such as PORT, it will be appended to the default help text available from the installation utility.

The following table lists the standard parameter keywords and the default help text supplied by the installation utility:

Parameter Keywords	Parameter Names	Default Parameter Help Text
IRQ	Hardware Interrupt	"Select the interrupt level that corresponds to the interrupt setting on the board or device. The interrupt setting must be unique (one not used by another device in the machine)."
PORT	Base I/O Port	"Select the port value (base I/O address) that corresponds to the port address setting on the board or device. Make sure the block of I/O addresses does not overlap the addresses of another device in the machine."
MEM	Memory I/O Address	"Select the memory address that corresponds to the memory setting on the board or device. Make sure the block of memory addresses does not overlap the addresses of another device in the machine."
DMA	DMA	"Select the DMA channel that corresponds to the DMA setting on the board or device. Make sure the DMA (Direct Memory Access) channel does not conflict with that of another device in the machine."
SLOT	Slot	"Select the slot number that corresponds to the expansion slot where the board or device is installed."

Parameter Keywords	Parameter Names	Default Parameter Help Text
NODE ADDRESS	Node Address	"Do not change this address unless you are prepared to administer local addresses according to the IEEE 802.2 specifications. The driver defaults to the node address on the board."
FRAME	Frame Type	"Select the frame type(s) used by the protocol(s) that your network requires. Make sure you have both the server and client configured for the same frame type."
BUS ID	Bus ID	"Select the bus type that corresponds to the bus used by the network interface card."

Valid Parameter Options

Each line that follows the parameter description (up to the next parameter description and not including any help) is interpreted as a possible value for that parameter. Note that these are TEXT values and can be no longer than 20 characters. These values must also be in the same format as they would be in the NET.CFG file.

The valid parameter values are specified in either list or range notation, depending on the first character of the ParameterCode. When you specify parameter values in a list form, you must enter each valid parameter value on a separate line.

Value ranges use only three value lines. The first value line is the beginning of the range. The second is the end of the range. The third is optional and is a default value preceded with an @ sign. If a default value is specified, the installation utility will generate the NET.CFG entry. If a default value is not specified, the installation utility will only generate a NET.CFG entry if the user enters a value. If the parameter is required, you must specify a third default value line.

The installation utility treats these values as strings. This allows the user to enter an L or M after the NODE ADDRESS parameter, designating in the NET.CFG file that the node address specified is in canonical (LSB) or noncanonical (MSB) format (see the *ODI Specification Supplement: Canonical and Noncanonical Addressing*, part number 107-000059-001).

Default Parameter Value

You designate a default parameter value in the INS file by preceding the value line with an '@' sign. In the example below, the board setting defaults are INT 3 and PORT 300.

All required parameters must have a default specified. If a parameter is optional, you need not specify the default. The installation utility always displays a blank selection to the user for optional parameters. If no default is specified, then the "no value" option becomes the default. A NET.CFG entry is generated for any option other than the "no value" option. The installation utility does not generate a NET.CFG entry for a "no value" option.

The default settings in the driver INS file should match the default jumper settings of the network board as shipped by the manufacturer, where possible. If the jumper settings cannot be guaranteed, the defaults should match the most likely jumper settings, or the most common user settings.

Example INS File (NE2000)

```
InS_StArT
NE2000.COM
1.0
^Novell/Eagle NE2000,NE2000.COM,IE0000
?Please select the options that match your board's jumper settings.
!RIE0000(INT)Hardware Interrupt
2
@3
5
7
!RIE0000(PORT)Base I/O Port
@300
320
340
360
$OIE0000(NODE ADDRESS)Optional Node Address
0
FFFFFFFFFFF
*RIE0000(FRAME)Media Frame Type(s)
?Note: Ethernet_802.2 is now the default frame type for ODI drivers.
       Existing LANs may be using Ethernet_802.3. The above example
       sets Ethernet_802.2 as the default.
@Ethernet_802.2
Ethernet_802.3
Ethernet_II
```

```
Ethernet_SNAP
InS_EnD
```

Example INS File (using various options)

```
InS_StArT
PCN2.COM
1.0,ROUTE.COM
^IBM PC NetWork BroadBand or BaseBand Adapter II,PCN2.COM,IE0000
^IBM PC NetWork BroadBand or BaseBand Adapter/2,PCN2.COM,00M000,EFEF
?This board may be used with Source Routing.  If your network is using
   Source Routing simply load the ROUTE.COM file after the PCN2.COM
   file has been loaded.
!RIE0000(PORT)Base I/O Port
@620
628
!O00M000(SLOT)Optional Slot Number
1
2
3
4
5
6
7
8
$OIEM000(NODE ADDRESS)Optional Node Address
0
FFFFFFFFFFF
!OIEM000()Primary or Alternate Adapter
?If your adapter is configured for Alternate, please select this option.
ALTERNATE
*RIEM000(FRAME)Media Frame Type
@IBM_PCN2_802.2 MSB
IBM_PCN2_802.2 LSB
IBM_PCN2_SNAP MSB
IBM_PCN2_SNAP LSB
InS_EnD
```

DRIVER.LST File

The DRIVER.LST file is an ASCII text file that lists the network boards the product supports. DRIVER.LST is located at the root of the distribution diskette that contains the ODI driver files. It is specific to the drivers located on the distribution diskette. The installation utility uses DRIVER.LST to avoid the extensive overhead of scanning every .COM or .EXE file on the diskette for specific INS files. You do not need the DRIVER.LST file if the distribution diskette contains less than four .COM or .EXE files. (Note that this is all .COM or .EXE files and not just driver files.) If the installation program does not find a DRIVER.LST file, it will scan all .COM and .EXE files it finds on the distribution diskette to gather this information.

Each network board is listed on a separate line with four columns. The columns are delineated by a comma (,):

- The first column is the full name of the network adapter and can contain a maximum of 48 characters.
- The second column is the filename (including the extension) of the driver used with the board and can contain a maximum of 13 characters.
- The third column is a six character code depicting bus type(s) supported for the adapter. The code is described below:

1st character	I (alpha)	supports ISA adapters
	0 (numeric)	does not support ISA adapters
2nd character	E (alpha)	supports EISA adapters
	0 (numeric)	does not support EISA adapters
3rd character	M (alpha)	supports MCA adapters
	0 (numeric)	does not support MCA adapters
4th character	A (alpha)	supports PCMCIA adapters
	0 (numeric)	does not support PCMCIA adapters
5th character	P (alpha)	supports PCI and CardBus adapters
	0 (numeric)	does not support PCI or CardBus adapters
6th character	V (alpha)	supports VESA Local Bus adapters
	0 (numeric)	does not support VESA Local Bus adapters

NOTE: Because all ISA boards work in EISA machines, 'I' and 'E' should both be used for ISA boards. This enables ISA boards installed in an EISA bus to appear in the installation utility. Keep in mind that

ISA boards installed in EISA machines retain ISA functionality and features.

- The fourth column is the *ProductID* field and applies only to MCA and EISA boards. This field contains the ID string stored in the POS registers in MicroChannel and EISA machines. The installation utility uses this string to automatically select the appropriate driver for the board found in the machine. This field is left blank for ISA boards.

Example DRIVER.LST File

```
Novell NE1000,NE1000.COM,IE0000
Novell NE2000,NE2000.COM,IE0000
Novell NE/2,NE2.COM,00M000,7154
Novell NE3200,NE3200.COM,IE0000,NVL0701
3Com EtherLink II,3C503.COM,IE0000
3Com EtherLink/MC,3C523.COM,00M000,6042
Novell RX-Net & RX-NET II,TRXNET.COM,IE0000
Novell RX-Net/2,TRXNET.COM,00M000,6014
IBM Token-Ring Network Adapter II & 16/4 Adapter,TOKEN.COM,IE0000
IBM Token-Ring Adapter/A,TOKEN.COM,00M000,E000
IBM Token-Ring 16/4 Adapter/A,TOKEN.COM,00M000,E001
IBM PC Network Baseband or Broadband Adapter II,PCN2L.COM,IE0000
IBM PC Network Baseband or Broadband Adapter II/A,PCN2L.COM,00M000,EFEF
Western Digital EtherCard PLUS Elite (all cards),WDPLUS.COM,IE0000
```