C

Installation Information File

Introduction

To facilitate the ability to programmatically install LAN and disk drivers, installation programs must know the parameters associated with each driver, the interactions that are required from the user, and how to set up the respective configuration file(s). This chapter describes the syntax of the driver information file used to provide installation utilities with the required information.

The install information file for a LAN or disk driver contains driver descriptions that specify configurable driver parameters, user interactions, and output format information. For a preview of several driver information files, refer to the examples at the end of this chapter.

The information file may contain one or more driver descriptions, each referencing a primary driver file and, optionally, other auxiliary driver files. A single driver may be referenced by multiple descriptions in multiple files. During installation, the install information and the referenced driver file(s) are copied to a permanent directory on the user's hard drive.

The driver install information is contained in ASCII text files that are shipped with the driver. The install information filename must have one of the following extensions, depending on the type of driver it references:

Environment	Extension
Server LAN Driver	.LDI
Server Disk Driver	.DDI

Reserved Keywords

The following keywords have special meaning in the installation information file. The keywords are shown in both a full form and an abbreviated form. Each of the indicated keywords will be described in the related section of this appendix.

Note: This appendix uses the full form of the keywords in the examples for clarity, however, the abbreviated form should be used when creating the final install information file. In addition, you should keep whitespace and comments to a minimum.

Keyword	Minimal Abbreviation
AND	AND
CDESCRIPTION	CD
CHOICE	CH
CPROG	CP
DECIMAL	DEC
DEFAULT	DEF
DESCRIPTION	DES
DLANGUAGE	DLANG
DRIVER	DR
ELSE	ELS
FILE	FILE
FRAME	FR
HELP	HELP
HEX	HEX
HIDDEN	HID
IF	IF
LANGUAGE	LANG
LIST	LI
NOT	NOT
OCTETBITORDER	OCT
OFILES	OF
OPTIONAL	OPT
OR	OR
OUTPUTFORMAT	OUT
PARAMETERVERSION	PAR
PATH	PATH
PRODUCTID	PROD
PROMPT	PR
REQUIRED	REQ
RESERVEDLENGTH	RES
STRING	STR
SYNTAXVERSION	SYN
TIMEOUT	TIME
TYPE	TYP
UNDEFINED	UND
VALUES	VAL
VERSION	VER

Information File Overview

A driver information file may contain one or more driver descriptions, as well as language definitions for strings within the descriptions. The general format of the description file appears as follows:

```
;DrIvEr DeScRiPtIoN


    Version: <driver description file number>
    SyntaxVersion: 1.00


    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>"
        .
        .
        .



;DrIvEr DeScRiPtIoN EnD
```

Note the initial and final lines:

```
;DrIvEr DeScRiPtIoN
    .
    .
    .
;DrIvEr DeScRiPtIoN EnD
```

These signature lines are used to bracket the driver install information. The installation program searches for these signatures whenever an information file is appended to the driver module. The lines are required and *must appear exactly as shown*. They should not be translated to another language.

Version:

The *Version* label is optional and is presently ignored by the installation / configuration utility. It is primarily used for manual version control by the user and description writer.

SyntaxVersion:

The *SyntaxVersion* label is mandatory and is controlled by Novell. This label informs the installation / configuration parser of the syntax to expect in the driver information file. The syntax version number is currently 1.00.

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 configurable parameters associated with the driver. The syntax within the driver block will be explained later in this chapter.

Language Section

The language section is used to accommodate the translation of 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 location.

If the language section is implemented, any translatable text strings are referenced using string names. Each language block then contains the string names and the corresponding text in the respective language.

The following page shows an example driver information file.

Example

```
;DrIvEr DeScRiPtIoN


Version: 1.00
SyntaxVersion: 1.00


  ; File  SAMPLE1.INF

    Driver SAMPLE1
    {
        Description:             $DESCRIPTION
        Help:                    $HELP


        PROMPT INT
        {
            Values:              3, 5, 7, 9
            Default:             3
        }


        PROMPT PORT
        {
            Values:              300, 310, 320
            Default:             300
        }


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

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

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

            CDescription:        "802.2 SNAP"
            Choice:              'Ethernet_SNAP'

            CDescription:        "Ethernet II"
            Choice:              'Ethernet_II'

            Default:              1
        }
    }

    DLanguage: 4
    ; Default English
        $DESCRIPTION =           "Sample driver description"
        $HELP =                  "Sample help text information"

    Language: 247
    ; Greek
        $DESCRIPTION =           "@#$&$#@ #@$^@ !)$&#*$&"
        $HELP =                  "~@#$^$#@$@%& #$@#@$& &^&**$#%"


;DrIvEr DeScRiPtIoN EnD
```

General Syntax

This section describes general syntax rules for writing a driver installation information file.

1. Comment lines can be added by starting the line with a semicolon (;). Everything on the rest of that line will be ignored. A semicolon may be preceded by white space (tabs or space characters). A comment may not exist on the same line as a declaration.

   example:
   ```
   ; install file for driver: NE2000.LAN
   ```

2. Items shown in angle brackets indicate something which the description writer must supply describing that aspect of the driver.

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

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

4. Items separated by (*or*) indicate alternate choices that may be used.

   example:
   ```
   THIS or THAT
   ```

5. Labels (words followed by a colon; e.g. *File:*) are **not** case sensitive. For example, the label *File:* is the same as *FILE:*.

6. Text strings may be surrounded by double quotes (" ") , single quotes (’ ’) , or whitespace.

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

   b. Strings surrounded by single quotes may contain whitespace, single or double quote characters, double-byte characters, or reserved characters: `= { } ( ) , : - ; < > !`

   c. Strings surrounded by double quotes are treated as single quoted strings but are used for text strings that can be translated to other languages.

   examples:
   ```
   "The driver defaults to using ..."
   ’Novell NE2000’
    ISA
   ```

A single quote character may appear in a double or single quoted string by preceding the character with a backslash (\'). However, a double quote character *may not* appear in a double quoted string even if preceded by a backslash.

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

The newline and tab characters, \n and \t, may exist within single or double quoted strings. They will be changed to the appropriate characters by the parser.

7. Keywords and labels should not be quoted and therefore cannot contain whitespace, single or double quote characters, double-byte characters, or reserved characters.

8. Help text within double quotes may be multilined. When a quoted string spans more than one line, all characters from the last non-whitespace on one line to the first non-whitespace on the next line will be replaced with a single space character.

   An actual newline character can be included in a quoted string using \n. This will be replaced by a CR-LF combination. A tab is indicated using \t.

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

   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."
```

Language Translation

The language section of the install information file is used to accommodate the translation of 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 location.

If the language section is implemented, 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 a particular language or the default language. 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 install utility encounters a **$<*string_name*>** variable, it will search for a definition of that string in the language block corresponding to the language the install utility is currently using. If the string is not defined in the current language, the string definitions corresponding to the default language block will be searched.

If a quoted string follows a **$<*string_name*>** with no intervening whitespace, and if the string definition is not found, the quoted string will be used as the string definition. This feature allows a default string (typically in English) to be specified if the string definition is not found in the language sections. If a definition for the string has already been found, the adjacent quoted string will be ignored.

example: `$DESCRIPTION"Novell NE2000 Driver"`

Finally, if a string definition is not found in any of the above mentioned forms, the string name itself will be used as the string text.

Language ID

Each language block is identified with a number. The language ID's that have been assigned to date are as follows:

Canadian French	0
Chinese	1
Danish	2
Dutch	3
English	4
Finnish	5
French	6
German	7
Italian	8
Japanese	9
Korean	10
Norwegian	11
Portuguese	12
Russian	13
Spanish	14
Swedish	15

The Novell operating system and utilities will be translated in German, Japanese, French, Spanish, and Italian. You may choose to provide text string translations for all, some, or none of the languages available.

Example

```
;DrIvEr DeScRiPtIoN

Version: 1.00
SyntaxVersion: 1.00

    Driver SAMPLE
    {
        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

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


;DrIvEr DeScRiPtIoN EnD
```

Driver Section

The format of a *Driver* description within an information file is shown below. The driver description contains two sections. The first section contains information used by the installation program to decide (with input from the user) if and how to install this driver. It includes everything from the beginning through the *Timeout* line. The second section contains the parameters. These are the items that the user can change or select to configure the installed driver.

All labels in the driver description are optional. Only the labels required for the driver being described need be included. It is highly recommended, however, that the *Description* and *Help* labels be defined, at a minimum, to help inexperienced users with installation and configuration. All labels and keywords may occur only once in a description, with the exception of the *PROMPT, LIST, or FRAME Parameter* definitions, which may occur as many times as desired. *Parameters* are detailed later in this chapter.

```
Driver <driver description name>  <dependency expression>
{
    Description:               "<text description>"
    Help:                      "<multi-line help text>"
    ParameterVersion:           <x.xx>
    ProductID:                  <list of product ID strings>
    CProg:                      <(server-specific) NLM name>
    Path:                       <path on media>
    File:                       <file name on media>
    OFiles:                     <other associated files>
    Timeout:                    <decimal timeout value in seconds>


    PROMPT or LIST or FRAME
    {
        <see parameter section>
    }
}
```

Driver Description Name

syntax: `Driver <driver description name>`
example: `Driver SAMPLE1`

Each driver description has a case-sensitive string (*<driver description name>*) with which it is associated. This is a logical name that uniquely identifies the driver description for the installation information file (since an information file may contain multiple driver descriptions). This name may not be more than 32 characters, and may not include whitespace, quotes (single or double), double-byte chars, or reserved characters (see "General Syntax").

After a driver has been installed and configured, the method used to associate a description with the driver file is to use the (*<installation filename>, <driver description name>*) pair. Therefore, each description name must be unique from all others within the same file.

Dependency Expression

example:

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

A *Dependency* describes the state of the driver description, either unconditionally or conditionally based on global parameters such as bus type (see "Global Predefined Parameters"). The state of a description has the following effect:

OPTIONAL	Driver description is displayed to the user and he/she may optionally select it.
HIDDEN	Driver description is not displayed nor available to the user.

A dependency expression allows a driver writer to make descriptions invisible to the user if not applicable (e.g. a MCA driver on an ISA machine). If no *Dependency* is declared, the driver description state defaults to optional.

See the "Dependency Expressions" section later in this chapter, for a more detailed description of the grammar and evaluation order allowed 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), 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 (symbolic \n or explicit).

Note: You may have multiple *Description* labels in a driver description section. Each description must also have a corresponding *Help* label following it. The Install utility displays each description and help but will load 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, typically enclosed in double quotes, that the user may optionally have displayed during installation and configuration. It contains additional information, cautions, etc. that a user may need to know about the driver. The help can be a maximum of 1500 characters long, and may contain newline characters (either symbolic \n or explicit). All explicit newline characters and adjacent whitespace will be replaced with a single space. All symbolic newline characters will be replaced with a CR-LF combination. Similarly, a tab may be represented using \t.

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

ParameterVersion:

example: `ParameterVersion:  1.00`

The *ParameterVersion* refers to the version number of the driver parameters (i.e. allowable command-line parameters in the case of a server driver). The *ParameterVersion* number should change **only** when the parameter interface changes, not necessarily when the installation file is modified. The installation file *Version* number is used for that purpose.

ProductID:

examples:

```
; ProductID for Novell NE3200
        ProductID: NVL0701

; ProductIDs for IBM Token Ring and Token Ring 16/4
        ProductID: E000, E001
```

The *ProductID* label specifies unique identification string(s) assigned to the product. One or more comma separated strings are allowed to accommodate the possibility of one driver supporting several cards with different product IDs. For the Micro Channel architecture the string is the file name for the product's .ADF file and whose value is stored in the MCA slot product ID POS registers. For the EISA architecture it is the string used as the name of the product's .CFG file and stored (in encoded form) in the EISA slot manufacturer ID and product ID registers.

Path:

example: `Path: \DRIVERS\LAN`

If the *Path* label is present, it references a directory path within the distribution media in which the driver file is located. If the *Path* is not present, the root directory of the media is implied. See also the *File* label description.

File:

example: `File: NE2000.LAN`

If the *File* label is present, it references the primary driver file on the distribution media. The driver *Path* and *File* name are concatenated (with a '\' between them) to form the full directory specification on the installation media. If the *File* label is not present, the description file root name with a default extension will be used:

Environment	Extension
Server LAN Driver	.LAN
Server Disk Driver	.DSK

OFiles: (Other Associated Files)

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

If the *OFiles* label is present, a comma-separated list of file names must also appear on the same line. When the primary driver file is copied by the installation or configuration utility, these associated files will also be copied. The *Path* string is concatenated to each of the listed files (with a '\' between them) to form the full directory specification for the files.

CProg: (Configuration Program)

example: `CProg: CSL.NLM`

The *CProg* label specifies one or more configuration executables and contains the names of NLM(s) that know how to perform the configuration.

Timeout:

example: `Timeout: 20`

If the *Timeout* label is present, a decimal number must follow. This decimal number indicates the maximum time in seconds the install utility should wait before determining that a driver failed to load and reporting an error to the user. If this label is not specified, the maximum wait time defaults to 5 seconds.

Driver Parameters

Each of the driver's configurable parameters must be defined in the driver description using one of the parameter types detailed in this section.

Parameter specifications define the configurable parameters which are needed by the driver. A parameter specification includes several components: parameter values, presentation to the user, and output formatting. The output format controls how information will be written to the command line for server drivers.

Three types of parameters are allowed, two of which are general, and one is special purpose.

The two general parameter types are PROMPT and LIST. They may occur in multiple instances in a driver description, once for each user configurable driver parameter. Both contain fields for a parameter description and help text, dependency expressions, and output format specification. A default value for the parameter can also be specified as well as permissible values from which the user can choose.

The special purpose parameter type, FRAME, may be declared only once within a single driver description. It exists for defining frame types supported by LAN drivers. As with the general parameters, the FRAME parameter allows a description, help, and a dependency expression. This parameter has a default method for input and output.

The general definitions that apply to all three parameters are described below. The following pages then provide the specific syntax for each parameter.

General Parameter Definitions

The parameter name is a case-sensitive string of 1 to 16 characters. The name is not presented to the user. It is used only in following situations: (1) a parameter may reference another parameter's value in a dependency expression, and, (2) the installation/configuration utility uses the name internally to distinguish between driver descriptions. A parameter name may only occur once within a single description.

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

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

HIDDEN 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 (default or user entered). 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 specified as REQUIRED, but having only one valid choice (the default value), have the following unique features: (1) It will not be displayed to the user (since there is no choice for the user to make), and (2) it will generate output. This feature creates a "invisible" parameter that generates output.
OPTIONAL Signifies parameters that are allowed but not required by the driver. The parameter is displayed to the user, but input from the user is not required. 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 (e.g. no output would be put on the command line for the parameter). 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.

The state of a parameter under various conditions is described by a *Dependency* expression. A parameter state may be specified as unconditionally OPTIONAL or REQUIRED. It may also be OPTIONAL, REQUIRED, or HIDDEN based conditionally on the value of other parameters. If a parameter has no dependency, it's state defaults to be unconditionally OPTIONAL. Refer to the section, "Dependency Expressions", for a more detailed description of how parameter states may be described with dependency expressions.

PROMPT Parameter

The format of the parameter PROMPT is shown below. It is used to obtain user input for a configurable parameter. The parameter can be a custom parameter or a local predefined parameter (see "Local Predefined Parameters").

The install 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 of values. The *Description* and *Type* fields shown below are required; all other fields shown 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>'
}
```

Dependency Expression

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

Description:

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

Help:

The *Help* text can be multi-lined and can be a maximum of 1500 bytes.

Type:

Type specifies whether a value is interpreted as string, 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"), and 35 are used characters 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 may 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 wide.

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 may 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, except in the case of parameters with the reserved names *PORTx* and *MEMx*, in which case this label is required in the server environment (see "Local Predefined Parameters"). If 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 output is to be generated from the final parameter value. The output will be placed on the command-line for the server environment. The format string may contain a maximum of one %s. The output string is created by replacing the %s with the parameter value. Output will be generated according to rules described in the section, "General Parameter Definitions".

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

Note: The *Description, Help,* and *Type* fields shown below are included to illustrate their use in the PROMPT example. However, for the local predefined INT parameter, these fields have default values and are normally not required.

```
Example:

PROMPT  INT  REQUIRED
{
    Description:          "Interrupt"
    Help:                 "Select the primary interrupt number."

    Type:                  HEX(1)
    Values:                2, 3, 5, 7
    Default:               3

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

```
            Driver NE2000 parameters

          Interrupt:  3_
            .
            .
            .
         (other parameters)
```

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

Select the primary interrupt number.
```

LIST Parameter

The format of a LIST parameter is shown below. It is used to obtain user input for a configurable parameter. LIST is similar to PROMPT, except the user selects an option for the parameter from a menu of valid choices.

The install 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 parameter *Description* and *Choice* fields shown below are required; all other fields shown 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>'
}
```

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 may 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 Expressions" for a more detailed description of the dependency usage.

Description:

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

Help:

The *Help* text can be multi-lined and can be a maximum of 1500 bytes.

Choice: and CDescription:

The *Choice Description (CDescription)* is used by the install utility 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) since the menu choices can usually be translated to different languages.

The *Choice* field is used 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 (e.g. Choice: 1-50) are **not** allowed. Typically choice values will not be enclosed in double quotes, since this would result in language specific command-line or configuration file parameters.

If the *CDescription* is not provided for a particular *Choice*, the text presented to the user in the menu will be the choice string itself. The number of pairs of choice descriptions and choices 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 may also be UNDEFINED, which means that none of the choices are initially selected. An absent *Default* label is identical in function 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.

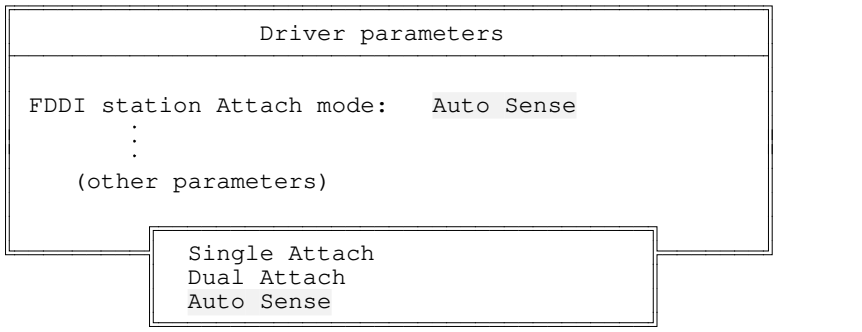
OutputFormat:

The *OutputFormat* label describes the way output is to be generated from the selected parameter choice. The output will be placed on the command-line for the server environment. The format string may contain a maximum of one %s. The output string is created by replacing the %s with the selected *Choice* string. Output will be

generated according to rules described in the section, "General Parameter Definitions".

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

```
Example:

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

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




```
Default value:  Auto Sense

If there is a secondary board in your machine,  you may wish
to overide the auto sense attachment.  Select the correct mode
from the list.
```

FRAME Parameter

The format of the FRAME parameter is shown below. It is used for LAN drivers to allow the user to select the frame types to be loaded by default.

The install utility uses the parameter *Description* and the *Default* values to display a list of frame names. The user will be allowed to add or delete frames from the list. In the server environment, a default logical name may 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>
}
```

Dependency Expression

If the FRAME parameter state is OPTIONAL, no values need be indicated by the user. If the parameter is REQUIRED, at least one frame must be selected by the user.

Only one FRAME parameter with multiple frame types is allowed to be visible (REQUIRED or OPTIONAL) to the user. Multiple FRAME parameters may be declared but only one block may 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 may also be used in dependency expressions for other parameters. When used in dependency expressions its value is a non-zero decimal number indicating the number of frame types selected (or UNDEFINED if no frame types were specified). Refer to "Dependency Expressions" for a more detailed description of dependency usage.

Description:

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

Help:

The *Help* text can be multi-lined and can be a maximum of 1500 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 Parameter" section later in this appendix).

Choice: and CDescription:

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

Choices can be any string, but should be strings understood by the driver and the protocols that will be used. Each *Choice* may appear only once in the list. Typically frame types will not be enclosed in double quotes, since 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 in function 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, a new instance of the load driver command will be entered at the command line for each frame type selected.

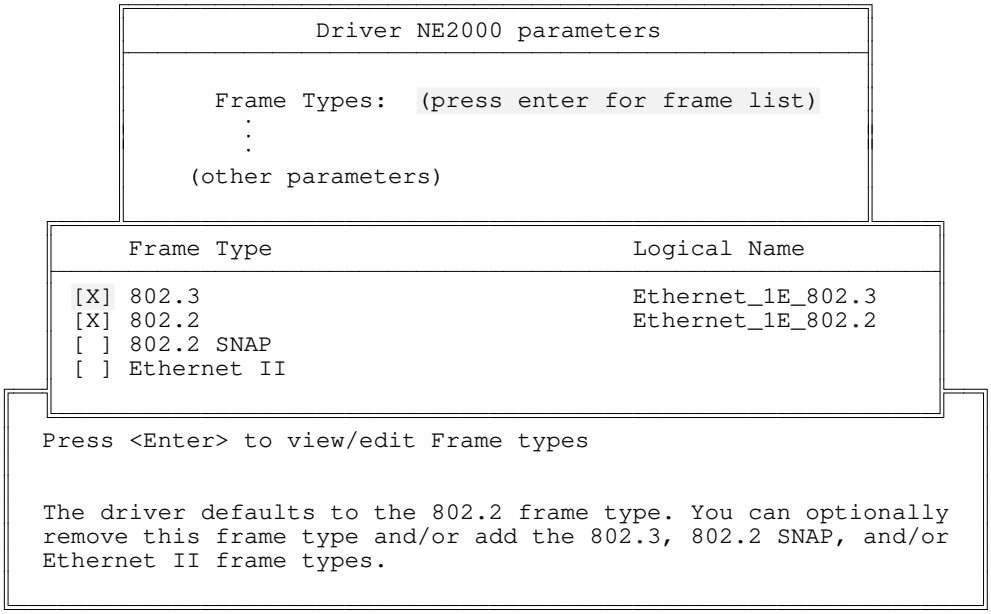
OctetBitOrder

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

non-canonical (LSB or MSB) formats (see Appendix G). The value associated with this label will be the default value for all frame types.

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

```
Example:

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:       "802.3"
    Choice:             'Ethernet_802.3'

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

    CDescription:       "802.2 SNAP"
    Choice:             'Ethernet_SNAP'

    CDescription:       "Ethernet II"
    Choice:             'Ethernet_II'

    Default:             2
}
```



Local Predefined Parameters

Some local parameters are quite standard, and as such, may have more specific meanings than the general parameter definitions mentioned in the previous sections. The following table list the predefined PROMPT parameter names:

Name	Parameter Type	Meaning
INT or INT1	PROMPT	Primary interrupt
INT2	PROMPT	Second interrupt
PORT or PORT1	PROMPT	Primary IO port
PORT2	PROMPT	Second IO port
MEM or MEM1	PROMPT	Primary memory address
MEM2	PROMPT	Second memory address
DMA or DMA1	PROMPT	Primary DMA address
DMA2	PROMPT	Second DMA address
SLOT	PROMPT	Machine slot number
NODE	PROMPT	Node address
RETRIES	PROMPT	Number of retries

If a PROMPT parameter name is one of these, all of the labels are optional, and will be defaulted if not specified. If a dependency expression is not declared, the parameter state will default to OPTIONAL. The range or list of values that the user will be allowed to enter will exclude ranges or values that are already in use by other drivers.

If one or more of the fields are not specified for the local predefined parameters, the following defaults will be generated by the installation / configuration:

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(8) for all others
Values:	0-99999999 for SLOT and RETRIES 0-FFFFFFFFFFFF for NODE 0-FFFFFFFF for all others
Default:	UNDEFINED
ReservedLength:	Not defaulted. This field must be specified for a PORTx or MEMx parameter in the server environment.
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'

Name	Default Help Information
INT INT1	"\nSelect the interrupt level that corresponds to the interrupt setting on the board or other 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 other 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 other 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 other 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 other device.\n\n Make sure the block of memory addresses does not overlap the addresses of another device in the machine."
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 other 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 other device is installed."
NODE	"\nDo not change this address unless you are prepared to administer local addresses acording 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 following examples show the use of the predefined parameter, INT.

```
Example 1:

    PROMPT INT
    {
    }


Example 2:

    PROMPT INT
    {
        Values:     2, 3, 4, 5
        Default:    3
    }
```

In example 1, the parameter description, output format, type, and type length default as follows:

```
Description:    "Interrupt number"
OutputFormat:   'INT=%s'
Type:            HEX (8)
Values:          0-FFFFFFFF
Default:         UNDEFINED
```

The help information displayed for this parameter would be:

```
Permissible values:  any hexadecimal number (digits 0 to F
allowed) of maximum length 8.

Select the interrupt level...
```

If a user tries to enter an interrupt value that is already taken, it will not be allowed by the installation/configuration utility, even though the taken values are not specified in the help text.

However, in example 2, assume that interrupt 3 is already being used for another driver. 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.
Default value:         3 (not-selectable).

Select the interrupt level...
```

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. It will used to determine if the specified group of port or memory addresses are available, and prevent the user from entering values that are taken. 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. If the parameter is PORT*x*, the reserved length represents a range of port values in single increments. If the parameter is MEM*x*, the reserved length represents a range of memory addresses in paragraphs (groups of 16 addresses).

Global Predefined Parameters

The following table lists the globally predefined parameters. These parameters are never displayed directly to the user (although in some cases the user will be prompted by the install utility for the information needed to create them). They exist only for the secondary effect of allowing driver description and parameter dependency expressions to reference them, thus making the state of descriptions and parameters conditional upon the value of these global parameters. Dependencies may be written with the assumption that these values will always exist (they will never be UNDEFINED).

Name	Parameter Type	Possible Values
BUS	PROMPT, STRING	'ISA' 'MCA' 'EISA'
GT_16	PROMPT, STRING	'TRUE' 'FALSE'

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

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

Dependency Expressions

As mentioned previously, a dependency expression can appear in the context of a driver description (i.e. before the Driver { }), or in the context of a parameter (i.e. before the parameter { }).

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).

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 refers to the value of the parameter corresponding to the parameter named. If the parameter named is a PROMPT parameter, the 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 pre-defined parameters (see "Global Predefined Parameters").

If the dependency expression is in the context of a parameter, *Name* may refer to either a global predefined parameter or a local parameter preceding this parameter within the same driver description.

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 which 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
    {
        .
        .
        .
    }
```

With the exception of the global predefined parameters, all names used in dependency expressions must be defined previously in the driver description in which they are used. 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.

No circular references to a name are allowed (a name can not directly or indirectly depend upon itself).

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.

A REQUIRED parameter must have a valid (defined) value before the user may exit the form, so dependency expressions can be written with the assumption that a REQUIRED parameter will always have a defined value (will not have a value that is UNDEFINED).

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 expressions

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

are valid if *name* has an UNDEFINED *value*. The expressions

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

will result in a term evaluation error if *name* is UNDEFINED. This also applies to expressions comparing two parameters (for example, *name1 >= name2*).

Dependency evaluation errors, if they occur, will be resolved in the installation/configuration utility by forcing the state of the parameter or driver description to OPTIONAL and reporting the error to the user before he/she exits the parameter form (this error is non-fatal, and the driver may still work if the resultant output is reasonable).

In order to prevent term evaluation errors from resulting in evaluation errors for the entire dependency, the UNDEFINED value should be taken into account in writing descriptions (either explicitly or implicitly). To write a dependency expression that explicitly handles an UNDEFINED value, a comparison to UNDEFINED should appear **prior** to the term that could result in an error.

```
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.

An example of an implicit comparison that takes UNDEFINED into account might be:

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

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

Examples

Example 1. Novell Server LAN Driver

```
;DrIvEr DeScRiPtIoN


SyntaxVersion: 1.00



    Driver PCN2

    {
        Description:           "Novell PCN2 (ISA or MCA) Driver"

        Help:                  "You can use this driver in an ISA (AT bus),
                                EISA, or a microchannel file server.  You can
                                have a maximum of two PCNII network boards in
                                your file server."

        File:                   PCN2.LAN


        ParameterVersion: 1.00



        PROMPT PORT

            if (BUS != MCA) REQUIRED
            else HIDDEN
        {

            Values:             620, 628
            Default:            620
            ReservedLength:     8
        }



        PROMPT SLOT

            if (BUS == MCA) REQUIRED
            else HIDDEN
        {
            Values:             1-8
        }



        PROMPT NODE

        {
        }



        ;   -(continued)-
```

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

            CDescription:       "PCNII 802.2"
            Choice:             'IBM_PCN2_802.2'

            CDescription:       "PCNII 802.2 SNAP"
            Choice:             'IBM_PCN2_SNAP'

            Default:             1

            OctetBitOrder:       LSB
        }
    }


;DrIvEr DeScRiPtIoN EnD
```

Example 2. Novell Server Disk Drivers

```
;DrIvEr DeScRiPtIoN


SyntaxVersion: 1.00



    Driver DCB
    {
        Description:            "Novell Disk Coprocessor Board (DCB) Driver"

        Help:                   "The Novell Disk Coprocessor Board Driver should
                                 be used with Novell SCSI disk sub-systems.  The
                                 DCB driver can be loaded four times.\n

                                 Although the driver does not prompt you for an
                                 interrupt, the board uses an interrupt; the
                                 driver displays which interrupt it is using.\n

                                 The first time a DCB driver is used, use DISKSET
                                 to edit the EEPROM information and add the current
                                 hard disks that are attached to the board.  You
                                 can run DISKSET by switching to the command-line
                                 (press ALT-ESCAPE>), and type \"load diskset\".
                                 To return to this screen after you are finished
                                 running DISKSET, press <ALT-ESCAPE> again."

        ParameterVersion:       1.00
        File:                   DCB.DSK



        PROMPT PORT
        {
            Values:             320, 328, 340, 348, 380, 388
            Default:            340
        }

    }


;DrIvEr DeScRiPtIoN EnD
```

```
;DrIvEr DeScRiPtIoN


SyntaxVersion: 1.00



    Driver ISADISK_DESCR

        if (BUS != MCA) OPTIONAL
        else HIDDEN
    {
        Description:            "Novell ISADISK (ISA or EISA) Driver"

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

                                 The default settings are the standard values
                                 for an internal controller.  Use the other
                                 supported values only if you are adding a
                                 secondary ISA adaptor."

        ParameterVersion:       1.00
        File:                   ISADISK.DSK



        PROMPT  INT  REQUIRED

        {
            Values:             B, C, E, F
            Default:            E
        }



        PROMPT  PORT  REQUIRED

        {
            Values:             170, 1F0
            Default:            1F0
        }



        LIST InterfaceChoice
        {
            Description:        "Use NetWare-Ready/CCM?"

            Help:               "This switch will cause the driver
                                 to scan for NetWare-Ready and CCM (Novell's
                                 Common Configuration Method).  Typically
                                 this should be set to \"No\"."

            CDescription:       "No"
            Choice:             ''

            CDescription:       "Yes"
            Choice:             '/n'

            Default:             1
            OutputFormat:       '%s'
        }
```

```
        LIST LowDriveChoice
        {
            Description:        "Use the low drive table?"

            Help:               "This switch is used for systems that want to
                                 use a secondary controller that does not have
                                 the BIOS in the system located above 0F000h.
                                 This will allow the driver to look for a drive
                                 table down to 0C800h.  Do not set this to \"Yes\"
                                 unless absolutely necessary, since data on the
                                 disk may be lost if it is set improperly."

            CDescription:       "No"
            Choice:              ''

            CDescription:       "Yes"
            Choice:              '/l'

            Default:              1
            OutputFormat:        '%s'
        }
    }


;DrIvEr DeScRiPtIoN EnD
```

Server Driver Template

The following is an install information file template for NetWare server LAN drivers. 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 "<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 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. Any description line which is not needed can be deleted. Additional custom parameters may be added as needed.

```
;DrIvEr DeScRiPtIoN

VER:    1.00
SYN:    1.00

; Place introductory comments here.
; Keep comments and whitespace to a minimum.

DR  <Driver Description Name>  <Dependency Expression>
{
   DES:    "<text description>"
   HELP:   "<multi-line help text>"
   PAR:      X.xx
   PROD:    <product ID string>
   CP:      <configuration NLM name>
   PATH:    <path on media>
   FILE:    <file name on media>
   OF:      <other assoc. files>
   TIME:    <driver load timeout value>

;  Only some of the following will be needed for any given driver description.
;  Delete those not needed and edit those which are needed to correctly describe
;  your adapter and driver.  You may also need to 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
   ; 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
           It is strongly recommended that at least 802.2
           be selected.  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.

  }
}


;DrIvEr DeScRiPtIoN EnD
```