

Novell[®] DeveloperNet Labs[™]

NetWare[®] Modem Script File Specification

Version 1.5

Novell[®] DeveloperNet Labs[™]

NetWare[®] Modem Script File Specification

October 1, 1999

Version 1.5

107-000044-001

Please ensure you have the most recent version of this document.
You can obtain the current revision from the DeveloperNet Labs web site at:
<http://developer.novell.com>

Disclaimer

Novell, Inc. makes no representation or warranties with respect to the contents or use of this manual, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

© Copyright 1995-1999 by Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express prior written consent of the publisher.

Novell, Incorporated
122 East 1700 South
Provo, Utah 84606

Trademarks

Novell has made every effort to supply trademark information about company names, products, and services mentioned in this document. Trademarks indicated below were derived from various sources.

Hayes and Smartmodem are trademarks of Hayes Microcomputer Products, Inc.

NetWare and Novell are trademarks of Novell, Inc.

Revision History

Revision	Date	Page	Changes
1.0	5/14/93		First Release
1.1	2/18/94		
1.2	6/28/96	3-1/2	Updated Chapter 3 with complete information for NetWare Server and Clients.
		4-1/8	Created Chapter 4 to describe modem script development and certification tools.
1.3	7/21/97		Document title change, minor typographical changes and updated for Novell BorderManager.
1.4	10/29/98		Updated for NetWare 5 and added Appendix A (Novell Modem Auto-Detection).
1.5	10/01/99		Took out testing information from chapter 4.

Table of Contents

Table of Contents.....	v
Introduction	1-1
Modem Script File Format	2-1
Overview	2-1
Description Line Format.....	2-2
Vendor Description.....	2-2
Modem Description	2-3
Modem Name	2-3
Modem Options.....	2-3
Modem Options - Capabilities.....	2-5
Modem Scripts	2-9
Modem Responses	2-13
Script Operations	2-14
Control Break - 'B'	2-14
Control DTR - 'D'	2-14
Set Interface Data Rate - 'R'.....	2-15
Flush Buffers - 'F'.....	2-15
Input String - 'I' or 'i'	2-15
Output String 'O' or 'o'	2-16
Pause 'P'	2-17
Quiet Wait 'Q'.....	2-17
Response Strings.....	2-18
Match Strings.....	2-18
Response Meanings	2-18
Example Descriptions	2-21
Example 1 - Simple Modem.....	2-21
Example 2 - Complex Modem	2-22
Example 3 – Rockwell Compatible 56 Kflex Modem.....	2-24
Environments	3-1
NetWare Server	3-1
NetWare Connect.....	3-1
NetWare MultiProtocol Router	3-1
Novell BorderManager	3-1
Novell NetWare 5 RAS	3-2
NetWare Clients	3-2
DOS.....	3-2
WINDOWS 3.1	3-2
Development & Testing of Modem Scripts	4-1
Development Environment	4-1
Creating Modem Script Files.....	4-1
NetWare Connect.....	4-1
NetWare MultiProtocol Router	4-3

Novell BorderManager / Novell NetWare 5 RAS.....	4-3
Additional Information For Writing Modem Scripts.....	4-4
Appendix A.....	A-1
Novell Modem Auto-Detection.....	A-1

Introduction

This specification defines how to create modem script files to allow modems to work with Novell® products such as NetWare 5, NetWare Connect™, NetWare MultiProtocol Router™ and Novell BorderManager™.

Recent Novell products and those in development are designed to be “modem independent”. This allows new modems to be supported by these products without a new version of the software being released. All that the end user is required to do, is to load the appropriate modem descriptor file onto his system.

These Novell products can interpret modem script files. They execute scripts in the files to perform modem operations as the application requires. Neither the modem control components nor the software products themselves are specific to any one modem or set of modems. Any details specific to modems are contained in the modem script files.

This document describes the format and content of the information present in the modem script files. The method of defining capabilities of a modem is specified and the process of constructing scripts to accomplish modem operations is outlined. Multiple examples illustrate uses of the details presented in this specification.

Modem script files ideally will be created by modem vendors using this specification. Test programs will be provided by DeveloperNet Labs allowing vendors to check that the modem script functions correctly. When the vendor is satisfied that the modem script file is correct, the file and modem can be certified by DeveloperNet Labs. It will then be made available to our mutual customers as a supported modem. It may also be integrated into future releases of Novell products.

When Novell’s products are installed, modem script files are copied along with other product files. As users configure the software they identify the modems to be used from lists of modem names. Any modem that has a modem script will be presented in these lists for the user to select.

Modem Script File Format

Overview

A modem script file includes information describing both a modem vendor and individual modems. The information about the modem vendor is specified first with from one to many descriptions of modems following.

The vendor information begins with the vendor's name, which identifies the company creating the description file. A copyright notice can be included to protect the company's rights. Version information should be added to allow tracking of additions or corrections to description information.

Each file can describe more than one modem. One way to organize modem scripts is to collect information about all modems from one vendor into a single file. This makes it easy to register the single filename with Novell. Another possibility is to group modems by 'family', as might be done with all of the "XYZ Ultra Xxx" models. It is suggested that all modems manufactured by a vendor be located in a small number of files.

Modem description information begins with a line specifying the modem name. This name must be unique within the entire set of modem names known to Novell, and thus should include some form of the vendor's name to avoid conflicting with any other vendor's descriptions. The modem name will be used in lists presented to users and may be up to 39 characters in length. Only one definition of a modem can exist across all modem description files; duplicates will produce unpredictable results.

The description for a modem continues with a line supplying information regarding the features, capabilities, and default values for the modem. This information is needed by the modem control components to determine which logical operations may be performed. Such information would include the highest interface bit rate possible to the modem, whether the modem can be used with leased lines, whether it supports use of hardware flow control, and so forth. These details are supplied explicitly using the `OPTIONS` keyword.

The modem scripts that perform particular operations must be specified next. These scripts are simply strings encoding sub-operations to be executed which together accomplish the desired operation. Multiple lines can be combined if required.

The final section of a modem description will contain the strings used to decode a modem's responses to commands and operations. For instance, the string

returned by a modem when a command is successful must be associated with the meaning 'OK'. Additional response recognition allows modem control components to handle call origination and answering. These strings contain pairs of subsections, the first subsection giving the input to be recognized, and the second specifying the meaning of that input.

The modem script files that are delivered to the customer are in a binary format that is obtained by processing the ASCII text, as described in this document, by a conversion program to be supplied by DeveloperNet Labs to the modem vendor. DeveloperNet Labs is also writing an interactive program that will allow the editing and creation of these modem descriptor files. The concepts described here apply equally well to both methods of creating the script file. Chapter 4 describes these tools and their use.

Description Line Format

Specification lines are given in the form of a keyword followed by an equals sign and then one or more values. For most of the keywords there will be only one string value. Other keywords, such as the OPTIONS keyword, can take multiple values which can be identifiers, numbers, and strings.

Strings are delimited using double quotes and control characters are included by prefixing with a reverse apostrophe, e.g. `M is used to enter a control-M.

ASCII 'nul' characters (\x00) must not be included in strings. Also, script strings cannot contain the ASCII characters '[' and ']', except as noted below.

Specification line length should be limited to less than 160 characters. In any case where more information needs to be specified for a keyword than will fit on one line it is permissible to use multiple lines, where each line repeats the keyword. Each such string will be appended to the previous to form one continuous string in the description. This will be seen in the examples later.

Comments can be included in the description text file. Whole line comments are created by making the first character of the line a semicolon.

Vendor Description

The following keywords are valid in the vendor section:

- MANUFACTURER Descriptive name of modem vendor
- COPYRIGHT Vendor's copyright notice
- MAJOR_VER Version number of modem description(s)
- MINOR_VER Revision number of modem description(s)

The manufacturer and copyright string values may be up to 80 characters in length. The version numbers may have numeric values from zero to 99. Currently none of these values are directly used by modem control components, but are provided for use by modem vendors. Examine the later examples to see typical uses of these keywords. Only one set of these fields is allowed in each modem description file, even if there are multiple modems described.

Modem Description

Modem Name

The modem name string value may be up to 39 characters in length. As noted before, this name must be unique within the entire set of modem names known to Novell. The MODEM keyword begins a new modem description.

There can be multiple descriptions for the same modem with each appropriate for distinct circumstances. For instance, it may be found that most revisions of a particular modem can be initialized quickly but that some ROM levels require delays between output characters. Rather than force all users to wait for a lengthy initialization operation, it is possible to create two descriptions.

```
MODEM = "XYZ Ultra Xxxx"
MODEM = "XYZ Ultra Xxxx (Slow Init)"
```

Modem Options

The OPTIONS keyword permits the definition of the capabilities of the modem. Some options have a value associated with them, others indicate the presence or absence of a modem capability. Options that have parameters are defined as shown here:

```
OPTIONS = DEFAULTRATE=19200, MAXRATE=57600
OPTIONS = FIXEDRATE=19200, DELAY=1
Options = LINKTYPE=ISDN-ASYNC, SCRIPT_Version=1.0
```

The following options require values to be defined:

- DEFAULTRATE Best 'normal' bit rate to modem
- DELAY Delay between command characters
- FIXEDRATE Best bit rate for use with fixed rate usage
- MAXRATE Maximum bit rate to modem
- LINKTYPE Connection method used by modem
- SCRIPT_VERSION Version of this script
- NLSIGNATURE Scripts certified by DeveloperNet Labs

DEFAULTRATE

When a modem operation specifies use of fixed rate mode, the **FIXEDRATE** option supplies the bit rate to be used to communicate to the modem. When that mode is not selected, modem control uses this option to determine the default bit rate for the interface to the modem.

DELAY

Some modems require a greater amount of time to process complex commands. Experience has shown that complex commands that are output to these modems one character at a time will be successful. This option supplies the amount of time to insert between characters on selected commands.

The numeric value is the time in tenths of seconds that modem control should wait between outputting characters. There are two script operations for output: one inserting delays between characters with the other not delaying between characters. If this option is not specified the default delay will be zero (and thus no delay at all).

FIXEDRATE

Modems can be initialized to use one unchanging bit rate between themselves and the DTE. This bit rate is usually set to a value high enough to permit use of compression no matter what line speed is used on a connection. The numeric value is the bit rate to be used when the modem is put into fixed rate mode.

Note: This option also implies that fixed rates are supported by the modem (see **FIXED** script).

MAXRATE

The set of interface bit rates that can be used to communicate from the DTE to a modem usually has an upper bound. This option supplies the maximum interface bit rate to be used with a modem. The numeric value for this option is the maximum rate in bits per second.

LINKTYPE

The link type specifies the connection method used by the modem to establish a connection. If the **LINKTYPE** field is not specified in the **OPTION** string, then "ANALOG" is used as a default value. The possible values of **LINKTYPE** are:

ANALOG	:	For asynchronous modems
X25	:	For X.25 connection type such as "AIOPAD"

ISDN-SYNC	:	ISDN-synchronous for ISDN adapter, such as “ISDN (AT Controlled)”
ISDN-ASYNC	:	ISDN-synchronous for ISDN terminal adapter
TCP	:	For TCP/IP connection type such as “AIO-PPTP”

Note: The **LINKTYPE** is only applicable to Novell BorderManager.

SCRIPT_VERSION

This option defines the version number of this specific modem description in X.X format. This number will be retained when this specific modem description is combined with others into a modem descriptor file. This number is used to identify certified versions of the script. The modem tools used to create the NLSIGNATURE (see below) will add the field if not present.

NLSIGNATURE

This 8 hex character field is used to determine whether the modem description has been modified from the one certified. Tools provided by DeveloperNet Labs to developers and customers will verify whether the signature is correct but only DeveloperNet Labs will be able to create the signature following successful certification.

This signature will be retained when the modem descriptor is combined with others into a single file, but will be detected as invalid if the script is modified from the certified one. Products such as NetWare Connect ignore this field so it is permissible to edit scripts but they will show up as having an invalid signature if checked.

Modem Options - Capabilities

These options inform the modem control components about the capabilities and restrictions of the modem. This is needed so that operation requests for these features can either be accepted, modified or rejected. The presence of a modem option indicates that a particular feature is supported; there is usually a corresponding script to activate that feature.

- ARQ Error control protocol(s)
- AUTOANSWER Automatic call answering
- COMPRESSION Compression protocol(s)
- DIALOUT Call origination
- FIXEDSINGLE Only one rate possible for fixed rate usage
- HAYES Append common responses subset
- HWFC Hardware flow control

- **LEASED** May be connected to leased-lines
- **MANUAL** Manual call origination/answering

OPTIONS = HAYES, FIXED, HWFC
OPTIONS = AUTOANSWER, LEASEDLINE
OPTIONS = ARQ, COMPRESSION

In many cases these options have one or more corresponding scripts that implement the options. As a convenience, it is not necessary to explicitly declare an option - the presence of a script will implicitly infer that the capability exists. For example, the presence of a DIAL script will enable the DIALOUT option. These associations will be noted in the script descriptions.

ARQ

This option specifies that the modem implements one or more error control protocols such as MNP or V.42. While error control may not be active on all connections, a modem initialization operation can request that error control be permitted for the next connection.

AUTOANSWER

This option specifies that the modem may be put into a mode where it can automatically answer incoming calls. When the modem begins answering, modem control will monitor the modem responses for connection parameters.

COMPRESSION

This option specifies that the modem implements one or more data compression methods such as MNP or V.42bis. It is understood that compression may not be active on all connections, and that compression may require simultaneous use of error control protocols and/or other modes such as fixed rate. This option informs modem control that the compression feature is supported by the modem.

DIALOUT

This option specifies that the modem can originate calls when requested by commands. The phone number and other information supplied will be inserted in the dial command as indicated in the dial script.

FIXEDSINGLE

Some modems permit use of the fixed rate feature, but with only one allowable bit rate. Presence of this option specifies that this restriction is true for this modem.

HAYES

Presence of this option informs modem control that this modem's responses include those in a set of common, industry-standard responses. Modem control will append this set of responses to any supplied in the description's response strings. These common responses are summarized here:

- **BUSY** Call origination: remote telephone was busy
- **CONNECT** Connection completed: interface rate is 300bps
- **CONNECT <R>** Connection completed: interface bit rate is 'R'
- **ERROR** Last command failed
- **NO ANSWER** Call origination: remote never answered
- **NO CARRIER** Modem detected absence of carrier signal
- **NO DIALTONE** Call origination: telephone line had no dial tone
- **OK** Last command executed successfully
- **REMOTE RING** Call origination: remote telephone is ringing
- **RING** Telephone line has given a 'ring' indication

The associated meanings accompany the responses and will be shown in later examples.

HWFC

This option specifies that the modem permits the use of hardware flow control between modem and interface. Some type of flow control is often necessary with modems that implement error control and/or compression; hardware flow control is superior to software flow control in that it does not interfere with the data being transmitted.

LEASED

This option specifies that the modem can be connected to leased (non-switched) telephone lines. When specified, modem control will permit the initialization of the modem into leased-line mode. Call origination and answering will employ different scripts than normal dial operations.

MANUAL

This option specifies that the modem can be used in situations where the user or operator will manually originate or answer telephone calls.

Modem Scripts

A modem script is a text string that is sent to the modem to cause a particular activity. They are associated with a particular modem capability and are transmitted to the modem when the application software wishes to invoke that operation.

More information on the content and creation of modem scripts will be given in a later section, "Script Operations". Individual scripts are summarized here:

- ARQ Enable error control protocol(s)
- AUTOANSWER Place modem into auto-answer mode
- COMPRESSION Enable data compression method(s)
- DIAL Originate a call on a switched line
- FIXED Place modem into fixed interface rate mode
- HANGUP Disconnect any call in progress
- HWFC Place modem in hardware flow controlled mode
- INIT Initialize the modem to a known state
- LEASEDLINE Place modem into leased line mode
- LSDANS Accept a leased line connection
- LSDORG Originate a leased line connection
- MANANS Accept manually answered switched connection
- MANORG Originate manually dialed switched connection

ARQ

This script enables the use of any of the error correcting protocols implemented by a modem when the next data connection is begun. Since which protocols may be activated depends on the remote modem, this script only specifies that the best possible for each connection be used. Through monitoring of negotiation progress responses the modem control components can be informed of the characteristics of the protocol activated.

```
ARQ = "O:AT&Q5S36=7S46=0S48=7`M:I:OK:"
```

AUTOANSWER

This script places the modem in the mode of automatically answering incoming telephone calls. A connection can begin without intervention by modem control. Modem control will monitor the progress of connection initiation and detect when the connection is complete and data transfer may begin.

```
AUTOANSWER = "O:ATS0=[R]`M:I:OK:"
```

COMPRESSION

This script enables the use of any of the data compression methods implemented by the modem when the next data connection is begun. Since the particular compression method employed depends partly on the remote modem, this script only specifies the preferred method to be used. Through monitoring of negotiation progress responses the modem control components can be informed of the characteristics of the method activated.

```
COMPRESSION = "O:AT&Q5S36=7S46=2S48=7`M:I:OK:"
```

DIAL

This script is executed when a call origination operation is requested on a switched line. The operation request parameters include whether the dialing should use pulse or touch-tone signaling, and the destination telephone number. These parameters are inserted into the dial script string using the substitution tags, "[T]" and "[P]". These tags are described more fully in a later section.

```
DIAL = "O:ATD[T][P]^M:"
```

FIXED

This script places a modem into fixed interface bit rate mode. This allows the interface to be programmed to one bit rate that can be used for all subsequent connections. The actual rate used will be determined by the associated FIXEDRATE and FIXEDSINGLE options.

```
FIXED = "O:ATS36=3`M:I:OK:"
```

HANGUP

This script causes the modem to disconnect any call that might be in progress. (i.e., place the modem "on hook") This script should specify all needed operations that to ensure the call is disconnected, irrespective of the current modem state.

```
HANGUP = "O:`M:P100:+++P10i25:OK:D10"  
HANGUP = "O:ATE1H0`M:Q"
```

HWFC

This script places a modem into hardware flow controlled mode. In this mode data transfer between modem and interface is controlled through the use of the RS-232 signals Request-to-Send (RTS) and Clear-to-Send (CTS). Each signal controls data transfer in one direction.

```
HWFC = "O:AT&K3`M:I:OK:"
```

INIT

This script causes a modem to be initialized to a known state. This state must have all optional features disabled. That is, the purpose of the INIT script is to put the modem into a state where any of the other features can then be added by individually executing scripts.

The INIT script is usually the first script executed when a modem operation is begun; the only script that could precede it might be the HANGUP script to disconnect a call in progress. The INIT script can make no assumptions about the previous state of the modem. Indeed, the previous user of a modem may not have been Novell's modem control, and so not even modem control will know the state of a modem.

The script must reset everything that can be affected by modem commands. This would include features like echo, call progress, result code, modem signal usage, flow control modes, and so forth. The script must set the correct modes so that modem response strings can be recognized.

```
INIT = "O:`MATZ`M:i25:OK:Q"
INIT = "O:ATE0Q0V1X4F1&C1&D2&L0&M0`M:I:OK:"
INIT = "O:ATS0=0S7=[W]S10=20S12=25`M:I:OK:"
```

LEASED

When a modem initialization operation is requested and the leased line feature is requested, this script is executed to place the modem into leased line mode. In some cases this feature is not under control by commands, but rather some switches must be set. In this case the script may be absent.

```
LEASED = "O:AT&L1`M:I:OK:"
or
;LEASED line is set by hardware switch
```

LSDANS

This script is executed when a call answer operation is requested on a leased line. The modem should attempt to connect to the remote modem using answering frequencies. Once this script is completed modem control will monitor the local modem's responses to detect when a connection has begun.

```
LSDANS = "O:`MATA`M:"
```

LSDORG

This script is executed when call origination is requested on a leased line. The modem should attempt to connect to the remote modem using origination

frequencies. Once this script is completed modem control will monitor the local modem's responses to detect when a connection has begun.

```
LSDORG = "o: `MATD`M:"
```

MANANS

This script is executed when a manual call answer operation is requested. The modem should attempt to connect to the remote modem using answering frequencies. Once this script is completed modem control will monitor the local modem's responses to detect when a connection has begun.

MANORG

This script is executed when manual call origination is requested. The modem should attempt to connect to the remote modem using origination frequencies. Once this script is completed modem control will monitor the local modem's responses to detect when a connection has begun.

Modem Responses

The response strings in a modem description allow recognition and interpretation of data sent from the modem to the DTE. This informs the modem control software of the success or failure of a command. It also lets modem control detect when a call is arriving.

As the responses generated differ between modems, the modem vendor must supply information to allow modem control to recognize responses. Response strings contain from one to many pairs of sub strings, the first giving the input string to be recognized and the second representing the standard 'meaning' of the string

With the ever more complex responses found in newer modems, it is sometimes necessary to perform multi-stage matching of response strings. As an example, when using negotiation progress monitoring to capture added information about connections, the PROTOCOL response can be received. The first stage of recognition would identify the input as the PROTOCOL message. The second stage of recognition would then identify the particular sub strings that may be present in this message.

Modem control will accumulate ASCII characters received from a modem until a carriage return character ('\x0D' or decimal 13) is received; all other control characters are ignored. The accumulated string is then compared to the match strings in the RESPONSES keyword string. When a match is found the meaning is interpreted and the appropriate action is taken.

The following response string keywords are defined:

- RESPONSES
- RESPONSES1
- RESPONSES2
- RESPONSES3
- RESPONSES4

The first keyword provides the first stage matching information. Part of the meanings information can specify that input matching for a message is to continue using information found in one of the other four response string keywords.

```
RESPONSES = ":OK:S6:RING:S7:ERROR:S13"
RESPONSES = ":COMPRESSION:S0M1"
RESPONSES1 = ":NONE:F0:CLASS 5:F2:V.42bis:F3"
```

Further information on response matching is given in the section "Response Strings" below.

Script Operations

A modem script contains a sequence of nano-operations that tell modem control which actions to perform. These actions include output of ASCII characters, controlling interface signals, checking for expected input, and so forth. There is no facility for conditional execution of nano-operations; the entire script is executed unless an error occurs.

Each nano-operation consists of an alphabetic character optionally followed by parameters for that operation. These values may be string and/or time values, or other modifiers for that basic operation. The operations are summarized below:

- 'B' Control asynchronous 'break' signal
- 'D' Control DTR signal
- 'F' Flush Transmit /Receive buffers
- 'I' Check for modem input (must match)
- 'i' Check for conditional modem input (optional match)
- 'O' Output characters to modem
- 'o' Output characters to modem with delay
- 'Q' Wait for end of input
- 'P' Pause script execution
- 'R' Set new interface rate

Control Break - 'B'

This operation is used to turn on the asynchronous 'break' signal momentarily. This might be used to get a modem's attention to switch it into command mode. The format is "B{time}" where {time} is the optional break on duration.

The break operation character may be followed by a decimal number giving the length of time in tenths of seconds for which break is to be turned on. If a time value is not given then the default break on time of one half second will be used.

Control DTR - 'D'

This operation is used to control the DTR signal to the modem. The DTR signal will be turned off momentarily and then turned on again. Turning off this signal might be used to get the attention of a modem when it is in data transfer mode. The format is "D{time}" where {time} is the optional duration in tenths of seconds for the DTR signal to be turned off.

If a time value is not given then the default DTR off time of one half second will be used.

Set Interface Data Rate - 'R'

This operation allows scripts to change the data rate used to communicate with the modem. This is used with modems that do not automatically resynchronize interface data rates after switching back to command mode from data transfer mode.

The format is “Rrate” where rate is the required decimal number specifying a rate in bits per second. The permitted values are:

Rate	Rate	Rate	Rate	Rate
50	75	110	134	150
300	600	1200	1800	2000
2400	3600	4800	7200	9600
19200	38400	57600	115200	

After execution of this operation any further output or input through the interface will use this data rate. Some asynchronous equipment must flush one or both of the input and output streams when changing data rates.

Flush Buffers - 'F'

Characters that have been buffered for output or input but not yet processed may be discarded by this operation. This might be useful when modem responses up to a point may safely be ignored, or if prior output should be discarded when starting new commands. The format is “F{B | I | O}”

The flush operation character must be followed by another character indicating which stream(s) should be flushed. The character is either ‘B’ which flushes both input and output streams, ‘I’ to flush just the input stream, or ‘O’ to flush only the output stream.

Input String - 'I' or 'i'

This operation allows a script to check for a specific string to be received from a modem. For example, after most modem commands a script should check for the returned indication of success, usually ‘OK’. There are two variants : ‘must match’ or ‘optional match’. The format for these is “(I | i){time}:string:” where the nano-operation character may be ‘I’ for the ‘must match’ or ‘i’ for the ‘optional match’ variants, respectively.

The operation character can optionally be followed by a decimal number specifying the maximum time to wait for this response. This value is specified in tenths of a second. If it is not given the default value of five seconds is used.

The next script character is used as a string delimiter and an input match string is collected. This string may contain control characters formatted as for the output string operation.

Modem control continues receiving characters from the modem until one of two occurrences. If a matching string from the modem is completed the nano-operation finishes and the script continues. If a match is not completed and the time-out period has elapsed since the last character was received from the modem, then an input time-out is declared.

If this was a 'must match' input string operation the time-out causes the script to be terminated with a 'bad modem response' error code. Otherwise the time-out simply terminates the optional match operation and continues with the rest of the script.

Output String 'O' or 'o'

This operation allows output of character strings from the script to a modem. The output string may contain any non-nul, non-control ASCII characters. The format of this operation is "(O | o):string:" where the operation may be coded using either 'O' or 'o' followed by the string to be output. If it is necessary to insert a delay between characters the 'o' operation will use the delay time specified using the DELAY option.

The string to be output is bounded by some character chosen by the script creator. The character directly following the operation code will be used to locate the end of the string. The script creator should choose a string delimiter character which will not be used for any interactions with the modem. The character should not be one of the alphanumerics as this would make reading descriptions difficult. A survey of several modems has identified the many punctuation characters that are used within modem commands and responses. This leaves the following set as the characters recommended for use: '<^_{}|:;', by convention a colon is used (":").

As noted in section "Description Line Format", control characters may be inserted into output strings using the accent-grave or reverse apostrophe.

Variable strings may be substituted into output or input strings by use of an substitution marker. A substitution is indicated by a substitution tag name surrounded by [and] characters. For example, substitution of the "tone" or "pulse" modifier and the phone number into a dial-out command might be coded as "O:ATD[T][N]M:" where [T] is replaced with 'T' or 'P' and [N] is replaced with a dial number.

Only a limited number of substitution tags are defined, and the substituted strings are not variable by modem type. The pre-defined tags are:

- T dial tone/pulse modifiers: 'T' or 'P'
- N dial phone number: supplied by application
- R ring count: used on initialization
- W ring count: used on initialization

Care should be taken that the longest command sent to a modem does not exceed what the modem can handle. Many modems are limited to a maximum of 40 command characters excluding the leading 'AT', spaces, hyphens, and final carriage return. The input command can be used to break up long command output sequences.

Pause 'P'

This operation allows a script to pause execution for a period of time. This is useful when modems may require some time to complete complicated modem commands. The format is "P{time}" where {time} is the optional pause time in tenths of seconds. If a time value is not given then the default time value of one second is used (time = 10).

Quiet Wait 'Q'

This is used to skip all the responses from a previous command before issuing a new command. It causes a wait until the modem remains continuously quiet for the specified time.

The format is "Q{time}" where the wait time is optional. When present this time value is in tenths of seconds. If a time value is not given then the default time value of one second is used (time = 10).

This nano-operation will throw away any data received from the modem. Whenever a character is received the elapsed time timer will be reset to zero. When the elapsed time timer reaches the specified wait time value the nano-operation completes successfully. An additional timer records the total time since the nano-operation began. If this timer reaches the sum of the specified wait time plus five seconds then a time-out is declared and the nano-operation completes unsuccessfully causing the script to be terminated with an error.

Response Strings

Match Strings

The first of each pair of strings in the RESPONSES string is known as the match string. As noted above, when modem control is monitoring modem responses, characters received from a modem are collected until a carriage return is received. The input string is then compared (case insensitive) against all the match strings found in RESPONSES keyword strings. This matching operation proceeds in the same order as the RESPONSES string occurred in the description file.

Match strings do not need to be the entire response string to declare a match. Only the initial characters of a response must match the match string. Thus the match string "ERR" will match both the response strings "ERROR" and "ERRONEOUS" but not "ERASE". However, this may make the order that match strings are tried even more important.

Match and meanings strings are delimited by any character that the description creator chooses. Whatever character begins the first RESPONSES string is the delimiter for all the RESPONSES strings. In the interests of clarity a non-alphanumeric character should be used. The colon character (':') is used in our examples.

Here are some example match and meanings strings:

```
:CARRIER <R>:R*S0:CONNECT <R>:S9  
:OK:S6:RING:S7:NO CAR:S12:ERROR:S13  
:PROTOCOL:S0M1  
:NONE:F0:ERROR-CONTROL:F1:LAP-M:F1
```

Response Meanings

The second string of each pair of strings is known as the meanings string. The interpretation of this string defines what the recognized response means to modem control. This includes whether the response is a success, failure, or some intermediate indication. When certain optional connection features are recognized they can be signaled to modem control by this method. Finally, this is the way that bit rates are given to modem control.

There are four types of meanings information: status, feature, rate, and match chaining. The status and feature values are decimal indices into tables used by modem control. The rate decimal value is the actual data rate in bits per second. The match chaining value will be described later.

Character	Type	Meaning
'S'	STATUS	Reports a status; may terminate scanning
'R'	RATE	Reports a data rate
'F'	FEATURE	Reports an enabled feature for this connection
'M'	CHAINING	Continues scanning using another string

Status Meaning

Status information is used to notify modem control when something of significance has been discovered in a response, or to report that scanning should continue. The following status index values are defined:

Value	Meaning	Value	Meaning
0	NONE	11	NO_ANSWER
1-5	RESERVED	12	NO_CARRIER
6	OK	13	ERROR
7	RING	14	NO_DIALTONE
8	RRING	15	VOICE
9	CONNECT	16	UNKNOWN
10	BUSY		

Rate Meaning

The rate meaning is used to tell modem control what the current line data rate is in bits per second. For most modems that implement negotiation progress messages this rate value can be captured from the "CARRIER" response by using the R* construct as in CARRIER <R>:R*S0 or CONNECT <R>:R*S9. This will match any speed response from the modem, and capture that value to return it in the rate definition command.

Rate	Rate	Rate	Rate	Rate
50	75	110	134	150
300	600	1200	1800	2000
2400	3600	4800	7200	9600
12000	14400	16800	19200	21600
24000	26400	28800	38400	57600
115200				

Features Meaning

The feature values indicate to modem control when optional connection features have been enabled on the current connection. Information on which features are enabled or disabled is made available to applications. Applications can use

this to determine whether they must independently perform error control or data compression for a connection. The features values are summarized below:

Value	Feature	Value	Feature
0	NONE	3	COMPR_V42
1	ARQ	4	UNBALANCED
2	COMPR_MNP	5	SYNC

Match Chaining

The match chaining meaning 'M' directs modem control to continue matching using the remainder of the input string (after the initially matched portion) and using a different RESPONSES string. This permits the multi-stage matching that is so useful with complex sets of responses like negotiation progress messages. The following example will illustrate:

```
RESPONSES = " :PROTOCOL:S0M1 "  
RESPONSES1 = " :NONE:F0:ERROR-CONTROL:F1:LAP-B:F1 "
```

```
Input from modem: "PROTOCOL: ERROR-CONTROL/LAP-B"
```

The first string is part of the first stage matching string formed from all the RESPONSES keyword strings. Modem control interprets it to mean that the response beginning with "PROTOCOL" is not a final response, rather that additional matching must be done using the RESPONSES1 string.

Modem control begins repeatedly checking the remainder of the input string against the RESPONSES1 match strings. Each time the match strings are used up modem control advances to the next character in the input string and tries again. This continues until all the characters in the input string have been exhausted. In this manner modem control finds the "ERROR-CONTROL" substring and notes that feature one, ARQ, is enabled for this connection.

Example Descriptions

Example 1 - Simple Modem

```

MODEM      = "Hayes Smartmodem 2400"
OPTIONS    = DEFAULTRATE=2400, MAXRATE=2400
OPTIONS    = HAYES, DELAY=1
INIT       = "o:`MATZ`M:i25:OK:Q"
INIT       = "O:ATE0Q0V1X4F1&C1&D2&L0&M0`M:I:OK:"
INIT       = "O:ATS0=0S7=[W]S10=20S12=25`M:I:OK:"
HANGUP     = "O:`M:P100:+++:P10i25:OK:D100:ATE1H0`M:Q"
DIAL       = "O:ATD[T][N]`M:"
AUTOANSWER = "O:ATS0=[R]`M:I:OK:"
LEASED     = "O:AT&L1`M:I:OK:"
LSDORG     = "o:`MATD`M:"
LSDANS     = "o:`MATA`M:"
MANORG     = "O:ATH1O`M:"
MANANS     = "O:ATH1A`M:"

```

A new modem description is begun with the **MODEM** keyword; in this case we are describing the modem "Hayes Smartmodem 2400". The name includes the vendor name, a reference to the modem family, and the particular designation for this model. This name will appear in lists in Novell's configuration programs.

The first **OPTIONS** keyword line defines the default and maximum interface data rates that can be used with this modem. The second declares that the common set of modem responses can be expected from this modem. It also specifies that, when requested, a delay of one tenth of a second should be inserted between characters output to the modem.

The **INIT** keyword lines define the script to be executed by modem control in order to initialize the modem to the "all features off" state. Additionally this script causes the modem to generate verbose return codes which will match those defined for the **HAYES** option. All of the **INIT** keyword line values are combined to form one **INIT** script string in the modem description.

Note: The first output uses the 'o' form when presenting the modem with the "ATZ" command; this will add a delay between output characters. Further output uses the 'O' form as we expect that the modem will respond to following commands without trouble.

The third **INIT** keyword line illustrates use of the 'W' substitution marker. In this case we are inserting the connection wait time value into the last initialization command output to the modem.

The **HANGUP** keyword script illustrates the extent to which we try to force an ongoing call to be disconnected. We try several techniques as we can not

assume what prior state we may have been in. First a carriage return is output in case we had sent a partial modem command. We then try to escape from data mode by using this modem's established method for doing so: pause for a one second, output "+++", then pause for another second. We then accept and ignore the "OK" response if we escaped data mode. As a final attempt to capture the modem's attention and switch it into command mode we turn off DTR for one second and then turn it on again. We now assume we have the modem in command mode and so output the hang-up command.

The DIAL keyword script illustrates the use of two more substitution markers, "[T]" and "[N]". Depending on the configured setting modem control will either substitute a 'T' or 'P' character in place of "[T]". Modem control will substitute the dial operation telephone number string in place of the "[N]" marker. Once this script is executed modem control will begin monitoring responses from the modem for connection status.

The LEASED keyword script illustrates the situation where a modem can be placed into leased line mode by means of a command. Some modems are put into leased line modem only through the use of switches on the modem. In that case this script would be absent.

The MANORG and MANANS keyword scripts demonstrate one method that enables many modems to be used with manual dial and answer operation.

Note: The presence of the DIAL, AUTOANSWER, LEASED, and MANORG/MANANS scripts implicitly informs modem control that these features are supported by this modem.

When the OPTIONS keyword option HAYES was used, modem control was told to append the RESPONSES strings that would recognize the common set of industry-standard responses. Since no other RESPONSES strings were defined in this description only the appended strings will be present in the description. This will be equivalent to the following:

```
RESPONSES = ":CONNECT <R>:R*S9:CONNECT:R300S9"  
RESPONSES = ":OK:S6:RING:S7:NO CAR:S12:ERR:S13:NO DI:S14"  
RESPONSES = ":BUS:S10:NO AN:S11:REMOTE R:S8"
```

Example 2 - Complex Modem

```
MODEM = "Hayes V-series Ultra 96"  
OPTIONS = DEFAULTRATE=9600,MAXRATE=9600,FIXEDRATE=19200  
OPTIONS = DELAY=1, HWFC  
HANGUP = "O:`M:P100:+++P10i25:OK:D100:ATE1H0`M:Q"  
INIT = "o:`MATZ`M:i25:OK:Q"  
INIT = "O:ATE0N1Q0V1X4&C1&D2&Q5&U0&K0&L0`M:I:OK:"  
INIT = "O:ATW1S95=44`M:I:OK:"
```

```

INIT      = "O:ATS0=0S7=[W]S10=20S12=25`M:I:OK:"
INIT      = "O:ATS36=1S37=0S38=2S46=0S48=128`M:I:OK:"
DIAL      = "O:ATD[T][N]`M:Q"
LSDORG    = "O:`MATX1D`M:"
LSDANS    = "O:`MATA`M:"
MANORG    = "O:ATH10`M:"
MANANS    = "O:ATH1A`M:"
; Capability Scripts - each enables a specific ability
AUTOANSWER = "O:ATS0=[R]`M:I:OK:"
FIXED      = "O:ATS36=3`M:I:OK:"
LEASED     = "O:AT&L1`M:I:OK:"
HWFC       = "O:AT&K3`M:I:OK:"
ARQ        = "O:AT&Q5S36=7S46=0S48=7`M:I:OK:"
COMPRESSION = "O:AT&Q5S36=7S46=2S48=7`M:I:OK:"
; Responses possible with negotiation progress enabled
; CARRIER 300
; CARRIER 1200
; CARRIER 1200/75
; CARRIER 75/1200
; CARRIER 2400
; CARRIER 4800
; CARRIER 9600
RESPONSES = ":CARRIER 1200/75:R1200S0F4"
RESPONSES = ":CARRIER 75/1200:R1200S0F4"
RESPONSES = ":CARRIER <R>:R*S0
; CONNECT
; CONNECT 1200
; CONNECT 1200/75
; CONNECT 75/1200
; CONNECT 2400
; CONNECT 4800
; CONNECT 9600
; CONNECT 19200
; CONNECT 38400
RESPONSES = ":CONNECT 1200/75:S9F4:CONNECT 75/1200:S9F4"
RESPONSES = ":CONNECT <R>:S9:CONNECT:R300S9"
; OK
; RING
; REMOTE RING
; BUSY
; NO ANSWER
; NO CARRIER
; ERROR
; NO DIALTONE
RESPONSES = ":OK:S6:RING:S7:REMOTE R:S8
RESPONSES = ":BUSY:S10:NO ANS:S11"
RESPONSES = ":NO CAR:S12:ERROR:S13:NO DI:S14"
; PROTOCOL: NONE
; PROTOCOL: ERROR-CONTROL/LAP-B
; PROTOCOL: ERROR-CONTROL/LAP-B/HDX
; PROTOCOL: ERROR-CONTROL/AFT
; PROTOCOL: LAP-M/HDX
; PROTOCOL: LAP-M/AFT
; PROTOCOL: ALT

```

```
RESPONSES = ":PROTOCOL:S0M1"  
RESPONSES1 = ":NONE:F0:ERROR-CONTROL:F1:LAP-M:F1:ALT:F1"  
RESPONSES1 = ":X.25:F5F1"  
; COMPRESSION: NONE  
; COMPRESSION: ADC  
; COMPRESSION: CLASS 5  
; COMPRESSION: V.42bis  
RESPONSES = ":COMPRESSION:S0M2"  
RESPONSES2 = ":NONE:F0:ADC:F3:CLASS 5:F2:V.42bis:F3"
```

This is a description of a more complex modem. This modem implements several more features, such as fixed rates, hardware flow control, and negotiation progress responses. Much of this description is similar to the previous description. Only interesting differences will be discussed here.

Note: Lines starting with a semicolon are treated as comments.

The INIT keyword script explicitly enables the negotiation progress responses. In this way modem control can capture the maximum information about the data connection.

This modem implements both error control protocols and data compression; the ARQ and COMPRESSION scripts will be used to enable these features if requested in the modem initialization operation.

This modem uses many more responses than are provided in the common set of modem responses; all responses to be recognized from this modem are explicitly defined by the RESPONSES keywords. Because of the complexity of negotiation progress responses, multi-stage recognition is used.

The RESPONSES strings that match "CARRIER" responses illustrate one situation where the order of match strings is very important. If the "CARRIER <R>" string were not last it would match either "CARRIER 1200/75" or "CARRIER 75/1200" responses, as the "<R>" would match the rate values preceding the '/'. This would cause connection information to be missed, in this case that the connection is using unbalanced data rates.

Note: Using the "CONNECT <R>" match string will match all of the balanced rate responses. If an unbalanced CONNECT occurs the first two match strings will be matched. If the response for connections at 300 bps (which does not have a data rate value) occurs the last match string will match.

Example 3 – Rockwell Compatible 56 Kflex Modem

```
;  
;           Example modem script for Novell BorderManager  
;  
MODEM = "Rockwell Compatible 56K (K56flex)"
```

```
OPTIONS = DEFAULTRATE=115200,DELAY=1,FIXEDRATE=115200
OPTIONS = MAXRATE=115200DIALOUT,MANUAL,AUTOANSWER,HWFC
OPTIONS = ARQ,COMPRESSION,SCRIPT_VERSION=1.0,LINKTYPE=ANALOG
INIT     = "O:`MAT&F`M:i25:OK:"
INIT     = QO:ATS0=0E0Q0V1W1X4&C1&D2&K\\N%C`M:I:OK:"
INIT     = "O:ATS7=[W]S10=20S12=25`M:I:OK:"
HANGUP   = "O:`M:P100:+++P10i25:OK:D100:ATE1H0`M:Q"
DIAL     = "O:ATD[T][N]`M:Q"
AUTOANSWER = "O:ATS0=[R]`M:I:OK:"
HWFC     = "O:AT&K3`M:I:OK:"
FIXED    = "O:ATS36=7`M:I:OK:"
ARQ      = "O:AT\\N3`M:I:OK:"
COMPRESSION = "O:AT%C3`M:I:OK:"
RESPONSES = ":CARRIER 1200/75:R1200S0F4:CARRIER 75/1200"
RESPONSES = ":R1200S0F4:CONNECT 1200/75:S9F4"
RESPONSES = ":CONNECT 75/1200:S9F4:CARRIER <R>:R*S0"
RESPONSES = ":CONNECT <R>:S9:CONNECT:R300S9:OK:S6:RING"
RESPONSES = ":S7:NO CAR:S12:ERR:S13:NO DI:S14:BUS"
RESPONSES = ":S10:NO AN:S11:REMOTE R:S8"
RESPONSES = ":PROTOCOL:S0M1:COMPRESSION:S0M2"
RESPONSES1 = ":NONE:F0:LAPM:F1:ALT:F1:ALT-CELLULAR:F1"
RESPONSES2 = ":NONE:F0:CLASS 5:F2:V.42BIS:F3"
MANANS   = "O:ATH1A`M:"
MANORG   = "O:ATH1O`M:"
```


Environments

Novell's modem control is being implemented in multiple environments. A short description of how modem script files are used in each environment is given here.

NetWare Server

Modem script files on a Novell NetWare Server are placed in a subdirectory accessible to NLMs (NetWare Loadable Modules).

NetWare Connect

The modem control components of NetWare Connect exist in a module called `SYS:SYSTEM\AIO.NLM`. All files containing compiled modem scripts are copied to the `SYS:\SYSTEM\AIO` subdirectory. When `AIO.NLM` is loaded, it searches this subdirectory for files with the extension `MDC` (Modem Definition Compiled). `AIO` then creates a list of all modem names defined in these files and which file contains the description for each. When one of the NetWare Connect services attempts a modem operation on a port, `AIO` determines which modem is attached to that port and ensures that the modem's description has been read into memory. `AIO` then starts the execution of the operation using the service's request parameters and the modem description.

NetWare MultiProtocol Router

The MultiProtocol Router uses compiled modem (`MDC`) files that are placed in the `SYS:\SYSTEM` directory. It interprets these files using the device manager (`DMGMT.NLM`) as needed for modem control. During device manager initialization time, all the modem scripts are read in and a list of available modems is built. This list is used when an application makes a request to get the complete list of modems or modem description for a named modem.

Novell BorderManager

The Novell BorderManager uses compiled modem description (`MDC`) files that are placed in the `SYS:\SYSTEM` directory. The Novell BorderManager Routing interprets `MDC` files using device manager (`DMGMT.NLM`) and Remote Access uses `AIO.NLM` to interpret `MDC` files. When `DMGMT.NLM` and `AIO.NLM` is loaded, all the modem scripts are read in and a list of available modems is built. This list is used when an application makes a request

to get the complete list of modems or modem description for a configured modem.

Novell NetWare 5 RAS

The Novell NetWare 5 RAS uses compiled modem description (MDC) files that are placed in the SYS:\SYSTEM directory. RAS is an integral part of NetWare 5 and uses Device Manager (DMGMT.NLM) and AIO Manager (AIO.NLM) to interpret MDC files. The modem list is built using the Device Manager and AIO Manager and is presented to the application when modem configuration is performed. Modem Auto-Detection (MODEMS.INF) is added to NetWare 5 RAS and this file is stored in the SYS:\SYSTEM\CONNECT directory. A brief description of this is provided in Appendix A: Novell Modem Auto-Detection.

NetWare Clients

Modem script files on NetWare Connect DOS and Windows 3.1 Clients are placed in the directory C:\NET\HSATACC, accessible to remote dial programs. The DIALCON, DOSDIAL and DIALER programs use these modem description files during remote connection establishment to a NetWare Connect Server using modem and telephone line. Novell's LAN Workplace product also uses these scripts to make a PPP or SLIP call to a dial up router.

DOS

NetWare Connect 1.0 (DIALCON)

DIALCON is a connection establishment utility for the NetWare Remote Node (NRN) product in NetWare Connect 1.0. DIALCON is executed by the users to connect or disconnect the telephone line. DIALCON is used to configure the modem parameters for asynchronous connections.

NetWare Connect 2.0 (DOSDIAL)

DOSDIAL is a DOS program that allows remote PC users to dial into a NetWare Connect server directly. DOSDIAL supports the IP and IPX protocols and the PPP and SLIP transport protocols.

WINDOWS 3.1

NetWare Connect 2.0 (DIALER)

The Windows DIALER is the Windows 3.1 version of the NetWare Connect remote client's software. Using DIALER you can dial into the NetWare Connect PPRNS service. The Windows DIALER supports the IP and IPX protocols and the PPP and SLIP transport protocols.

Development & Testing of Modem Scripts

Novell's NetWare Connect and NetWare MultiProtocol Router product uses modem control components. This section describes how to develop modem script files and how to test them.

Development Environment

This section describes how to write and test modem script files for NetWare Connect Server. A PC with COM port, modem, modem cables and DeveloperNet Labs modem development kit (MODEMKIT.ZIP) is required. A phone connection is required to test connectivity to a NetWare Connect Server.

Create a modem script file for the modem under test using the tools available from DeveloperNet Labs in the MODEMKIT.ZIP archive and the description contained in this document.

MDMTEST	: The modem certification program.
MDMTXT	: Convert's MDC file to ASCII file.
MDMCVT	: Convert's ASCII file to MDC file.
MDMMGR	: DOS program to create or edit a modem script file.
MDMWIN	: WINDOWS program to create or edit a modem script file.
WMDMMGR	: WINDOWS program to create or edit a Novell BorderManager or NetWare 5 modem scripts.

Creating Modem Script Files

NetWare Connect

The following procedure is recommended for creating a modem script file for NetWare Connect Server to support your modem.

NetWare Connect allows multiple modem script files to be present simultaneously with each file defining one or more modems. When a user configures the NetWare Connect Server he will be presented with a list of modems contained in all of the MDC files for selection of the appropriate one.

The following steps are recommended for creating or modifying the modem scripts.

Step 1. Obtain the latest version of AIOMDMS1.MDC, AIOMDMS2.MDC and NLABSMDM.MDC modem script files, these are contained in the archive MODEMS.ZIP and are constantly updated to support new modems. This archive is available from Novell's web site at <http://labs.novell.com/wan/modemscr/modem.htm>.

Use the MDMWIN (WINDOWS 3.1) or the MDMMGR (DOS) program to display the list of modems included in the current release of MODEMS.ZIP archive file. Verify whether your modem script is in the released list of modems. Many of these scripts were created by Novell, but may not match the current version of your modem. Alternatively select a modem that is similar to yours as a starting point for creating a new one.

Step 2. Convert the MDC file containing the selected modem into an ASCII file (sample.txt) by using MDMTEXT program. Use a text editor to remove all but the relevant modem script. Make sure you use a unique MODEM name string. Edit the parameters to reflect your modem. You can add multiple entries for the same modem for different conditions of use but make sure they have a unique MODEM name tag. Remember to change the modem name (MODEM), manufacturer's name (MANUFACTURE), and copyright messages (COPYRIGHT) fields. (These fields can be edited with MDMMGR if it is invoked with the /edit switch, they apply to the entire modem descriptor file).

Also add a SCRIPT_VERSION field to the options string. This can currently only be done by editing the text version of the script and converting to binary with the MDMCVT utility. This revision number will appear on the certification bulletin and will aid our customer support groups to resolve problems as the scripts evolve. It is important that you do not modify the script without changing the version number.

DeveloperNet Labs will add the NLSIGNATURE field after certification. This signature can be verified with the VMDMSIG utility.

Step 3. Convert this ASCII file (sample.txt) to a binary file (vendor.MDC) using the MDMCVT program. Use a file name that is unique and easily recognizable as your company's modem description file. The file extension should be MDC to identify it as a modem description file.

Step 4. If necessary edit the binary file (vendor.MDC) directly using the MDMMGR (DOS) and MDMWIN (WINDOWS 3.1) for additional changes. Now you are ready to test the newly developed NetWare modem script.

NetWare MultiProtocol Router

NetWare MultiProtocol Router comes with server based utility MDMCVT.NLM that converts modem control scripts in ASCII format to a compiled binary format (MDC) used by DMGMT. The following procedure is used to compile a modem description file.

1. Copy the modified ASCII modem control script file (SCRIPTNAME.SCR) to the SYS:SYSTEM directory on the server that contains the NetWare MultiProtocol Router.
2. Enter the following command line at the server prompt:

```
Load MDMCVT SYS:\SYSTEM\SCRIPTNAME.SCR  
  
SYS:\SYSTEM\SCRIPTNAME.MDC
```

The file SCRIPTNAME.SCR is ASCII file compiled, and the output is placed in the file SCRIPTNAME.MDC

If DMGMT is loaded and running on the system MDMCVT also does the following tasks:

- Disconnects all current calls.
- Disables further requests for modem connection.
- Reads in all SYS:\SYSTEM*.MDC files and reinitializes DMGMT to use the compiled scripts in these files.
- Reenables modem connection request processing.

Novell BorderManager / Novell NetWare 5 RAS

Novell BorderManager comes with a WINDOWS based utility WMDMMGR.EXE. This utility is used to create new modem scripts or add to the existing list of modems. The MDC files created by the WMDMMGR utility is not compatible with NetWare Connect and NetWare MultiProtocol Router. Consult Novell Border Manager manuals for detailed information about this utility. The MDC files NIASMDM1, NIASMDM2 and NIASCERT contain the modems supported by Novell BorderManager.

Additional Information For Writing Modem Scripts

Using the scripting language defined earlier in this document, you can create a modem script for Novell BorderManager, NetWare Connect and NetWare MultiProtocol Router. Each of the sections of the script should be written with the following recommendations:

MODEM

Use a unique modem description name to identify your modem from the list of other modems during modem selection process. It is recommended to use a combination of vendor name and modem capabilities as a unique name. Don't forget to append the revision number e.g. (2.3).

OPTIONS

This defines the capabilities of the modem that are accessible with the modem scripts. It should normally include DIALOUT, AUTOANSWER, HWFC, ARQ and COMPRESSION. The OPTIONS list is invisible during the use of MDMWIN and MDMMGR script manager and gets built once you define all the features of the modem. An entry will exist for each capability that can be enabled with a text string defined below. If you are creating a modem script using a text editor ensure this list is consistent with the rest of the script.

INIT

It should be able to initialize the modem to a known state, regardless of the existing state. Usually the script should start with "AT&F" to restore the modem to factory defaults. It should then ensure that all options are disabled. Most modems will also require an "&C1&D2" to cause correct behavior of DCD and DTR. Disable auto answer mode (S0=0) and assign dynamic value to wait for carrier after dial (S7=[W]). Disable Error Control and Flow Control is required for proper functionality of the modem scripts. You should enable Error Control and Flow Control in the ARQ and HWFC section of the modem scripts.

HANGUP

Ensure there is an adequate pause before issuing the "ATH0" command.

DIAL

Accept the dial tone and phone number dynamically from the DOSDIAL or DIALER program e.g. ATD[T][N].

AUTOANSWER

Set the configured value for the number of rings prior to answering the call
e.g. ATSO=[R]

HWFC

Enable the Hardware Flow Control Option.

ARQ

Enable the Error Control Option.

COMPRESSION

Enable the Data Compression option.

RESPONSES

This is a very important section. It defines all the responses coming back from your modem. If you have not defined proper responses then the modem script will fail to recognize the responses and drop the connection or fail to proceed to the next stage of establishing the connection. Properly defined RESPONSES help in diagnosing the modem problem during remote connection establishment.

MANUAL

This should give the manual dial and answer options to the user. The MANANS and MANORG should normally contain "ATH1A" and "ATH1O" respectively.

Appendix A

Novell Modem Auto-Detection

Automatic Modem Detection is new feature added to NetWare 5 RAS. This reduces the modem configuration complexity and narrows the modem selection process. This process works on the returned strings of ATIO and ATI3 modem string and modem description available in the modem auto-detection file i.e. MODEMS.INF. The latest release of MODEMS.INF is included for the reference.

```
; Modem Automatic Detection
; -----
; Updated : 10/15/98
;
; To add your modem to Novell RAS auto-detect mode send MDC file
; and ATIO and ATI3 string to "wansupp@novell.com"
;
; Copy this file to sys:\system\connect
;
; Modem Name : The first field is "Modem Name" and
;              must match a name in the modem description file's (MDC)
;
; ATIO        : Wildcard string containing the response to ATIO command
;
; ATI3        : Wildcard string containing the response to ATI3 command
;
; All fields are contained in double quotes ("), and separated by comma and
; white spaces. The defined wildcard characters are :
;
;   ?         : matches any single character
;   !         : matches end-of-line
;   #         : matches any digit
;   @         : matches white spaces (spaces, tabs, cr or lf)
;   *         : matches any substring (0 or more characters)
;
;
; Modem Name                ATIO                ATI3
;
"3COM EtherLink III LAN+Modem PCMCIA 28.8",    "*932*",      "*3COM CORPORATION*"
"3ComImpact IQ ISDN",                          "*3C882*",    "@3ComImpact IQ*"
"AT&T Comsphere 3810 Plus (33.6)",             "*144*",      "@03974896*"
"AT&T Comsphere (3810/3820/3825) Plus",         "*144*",      "@3964792*"
"AT&T DataPort Express(14.4)",                 "*144*",      ""
"AT&T DataPort Express(28.8)",                 "*144*",      ""
"Bocamodem 14.4Kbps V.32bis",                  "*14400*",    "@V*. *--*"
"Bocamodem MV.34E",                            "*28800*",    "@V*. *-V34_*"
"Bocamodem MV.34E",                            "*33600*",    "@V*. *-V34_*
```

```

"Codex 326X FAST",          "*960*",          "@ERROR*"
"Digi RAS Modem",          "**Digi RAS@56000*", "@V#.###-K56_DS"
"Everex Evercom 24E ü",    "**248*",          "@###!"
"General DataComm VF 28.8", "**288*",          "@REV *"
"Hayes Century 2 OPTIMA V.34+FAX", "**33600*",        "@Version #.###"
"Hayes OPTIMA 288 V.34",   "**288*",          "@Version*"
"Hayes ACCURA 288 V.FC",  "**28800*",        "@04-00858-*PASS*"
"Hayes ACCURA 28.8 V.34", "**28800*",        "@04-00971-*"
"Hayes ACCURA 336 Voice", "**33600*",        "@V#.###-V34_DSVD_DS*"
"Hayes Smartmodem OPTIMA 28.8 V.FC", "**28800*",        "@04-00621-*PASS*"
"Hayes Smartmodem OPTIMA 14.4",   "**14400*",        "@04-00504-*PASS*"
"Hayes V-series Ultra 144",      "**14400*",        "@04-00455-*PASS*"
"Hayes V-series Ultra 96",       "**960*",          "@04-00455-*PASS*"
"Hayes V-series Ultra 96",       "**960*",          "@04-00195-*PASS*"
"Hayes V-series 9600",          "**960*",          "@04-00141-*PASS*"
"Hayes V-series 9600",          "**960*",          "@04-00152-*PASS*"
"Hayes V-series 2400",          "**249*",          "@04-00006-*PASS*"
"Hayes Smartmodem 2400",        "**249*",          "@04-00082-*"
"IBM 28.8(V.34) Kbps ISA",      "**28800*",        "@7852-I##-V##*"
"Intel 144/144e",             "**149*",          "@E.C. Version :*"
"Intel 14.4EX",               "**149*",          "@U##,*-*.*"
"Intel SatisFAXtion/400E",      "**149*",          "@U##,*-*.*"
"ISDN(AT Controlled)",         "**64000*",        "@Novell ISDN*"
"Microcom QX/4232bis",         "**1442*",        "@OK*"
"Microcom QX/4232bis",         "**1443*",        "@OK*"
"Microcom DeskPorte FAST",      "**2882*",        "@DeskPorte V.FC*"
"Microcom DeskPorte FAST ES 28.8", "**28800*",        "@V*. *-* VFC*"
"Microcom DeskPorte 28.8S",     "**28800*",        "@V#.###-V34_DP*"
"Microcom DeskPorte 28.8S",     "**28800*",        "@V#.###-V34_DS*"
"Microcom DeskPorte FAST",      "**2882*",        "@DeskPorte 28.8P #.##"
"Microcom DeskPorte FAST",      "**2883*",        "@TravelCard FAST 28.8P
#.##"
"Microcom DeskPorte FAST",      "**2883*",        "@TravelCard FAST 28.8 #.##"
"Microcom DeskPorte FAST",      "**2884*",        "@DeskPorte 28.8 #.##"
"Microcom DeskPorte FAST+",     "**2884*",        "@DeskPorte 28.8 #.##"
"Microcom OfficePorte Voice",   "**33600*",        "@V#.###-V34*"
"Motorola BitSURFR Pro ISDN Modem", "**960*",          "@4574190-1D*PASS*"
"Motorola ModemSURFER 28.8",    "**1634*",        "@Motorola ModemSURFR*28.8*"
"Motorola ModemSURFR 56K",      "**56000*",        "@V#.###D-K56_DLS*"
"Motorola VoiceSURFR 56K",      "**56000*",        "@V#.###D-K56_DLS*"
"Motorola Power 28.8 Data/Fax Modem", "**288*",          "@Motorola Power 28.8*"
"Multi-Tech (slow init)",       "**247*",          "*"
"Multi-Tech 932",               "**247*",          "@119 162 240 001*"
"Multi-Tech 1432BA/BL",         "**247*",          "@119 162 240 001*"
"Multi-Tech 1932BL",            "**247*",          "@119 162 240 001*"
"Multi-Tech 1432ZDX",           "**247*",          "@117 130 240 001*"
"Multi-Tech 1932ZDX",           "**247*",          "@117 130 240 001*"
"Multi-Tech 1432BR",            "**247*",          "@117 162 240 001*"
"Multi-Tech 1432MR",            "**247*",          "*"
"Multi-Tech 2834BA/BL",         "**247*",          "*"
"Multi-Tech 2834BR",            "**247*",          "@102162240*"
"Multi-Tech 2834MR",            "**247*",          "@100162240*"
"Multi-Tech 2834ZDX",           "**247*",          "@101162240*"
"Multi-Tech 28800 & 33600 (slow init)", "**247*",          "*"

```

```

"NetComm SmartModem M7F SM7710",          "*960*",      "@ERROR*"
"NetComm SmartModem M11F",                "**28800*",   "@Series*V.FAST*NetComm Ltd*"
"Practical Periph. 2400SA V.42bis",       "*249*",     "@PM2400SA V.42bis*"
"Practical Peripherals 9600SA",          "*960*",     "@PM9600SA*"
"Practical Peripherals PC288LCD v.34",    "*28800*",   "@PC288*LCD*"
"Practical Peripherals 336 MiniTower",    "*33600*",   "@04-00971-730*"
"Practical Peripherals 336 FLASH MT",     "*33600/56000*", "@33.6K/56K Modem*"
"Practical Peripherals 56K MiniTower",    "*56000*",   "@56K Modem*"
"Rockwell Compatible 56K (K56flex)",      "*56000*",   ""
"SupraExpress 56K",                       "*56000*",   "@V*-K56*SupraExpress 56e*"
"Telebit TrailBlazer Plus",              "*960*",     "@ERROR*"
"USRobotics Courier V.34 28.8K",          "*2886*",   "@###:###:###*"
"USRobotics Courier I-Modem ISDN/V.34",   "**6401*",   "@USRobotics Courier I-Modem
with ISDN/V.34*"
"USRobotics Courier I-Modem ISDN/V.34/x2", "**6401*",   "@USRobotics Courier I-
Modem with ISDN/V.34*"
"USRobotics Courier I-Modem(V.34 & x2)"   "**6401*",   "@USRobotics Courier I-Modem
with ISDN/V.34*"
"USRobotics HST, HST Dual Standard",     "*1444*",   "@###:###:###*"
"USRobotics Courier V.EVERYTHING 56k(x2)",**5607*",   "@USRobotics Courier
V.Everything*"
"USRobotics V.32/V.32bis",               "*1445*",   "@###:###:###*"
"USRobotics Sportster 9600/14400 V42bisü",**965*",   "@OK*"
"USRobotics Sportster 9600/14400 V42bisü",**1444*",   "@OK*"
"USRobotics Sportster 9600/14400 V42bisü",**1444*",   "@Sportster 14,400*"
"USRobotics Sportster V.34 28.8K",       "*2886*",   "@Sportster 28800*"
"USRobotics Sportster 33.6 Faxmodem",    "*3362*",   "@Sportster 33600/Fax*"
"USRobotics Sportster 56k Fax (x2)",     "*5601*",   "@U.S. Robotics Sportster
56000 Fax*"
"Xircom Ethernet+Modem 19.2",            "*19200*",   "@CreditCard Ethernet+Modem
II*"
"Xircom Modem 28.8",                     "*28800*",   "@Xircom CreditCard Modem
28.8*"
"Xircom Ethernet+Modem 28.8",            "*28800*",   "@Xircom CreditCard Modem
28.8*"
"Xircom Ethernet+Modem 33.6",            "*33600*",   "@Xircom CreditCard Modem
33.6*"
"Xircom Token Ring+Modem 33.6",          "*33600*",   "@Xircom CreditCard Token
Ring+Modem 33.6*"
"Xircom Modem 33.6",                     "*33600*",   "@Xircom CreditCard Modem
33.6*"
"Xircom Modem 56",                       "*Modem 56-G*", "@Xircom CreditCard
Modem 56-GlobalACCESS*"
"Xircom Ethernet 10/100 + Modem 56",     "*Modem 56*",  "@Xircom CreditCard
Modem 56*"
"Zoom VFX V.32bis",                      "*14400*",   "**VFX14.4*"
"Zypcom SX/SE/RX-Series",                "*932*",     "@Z34-*"

```