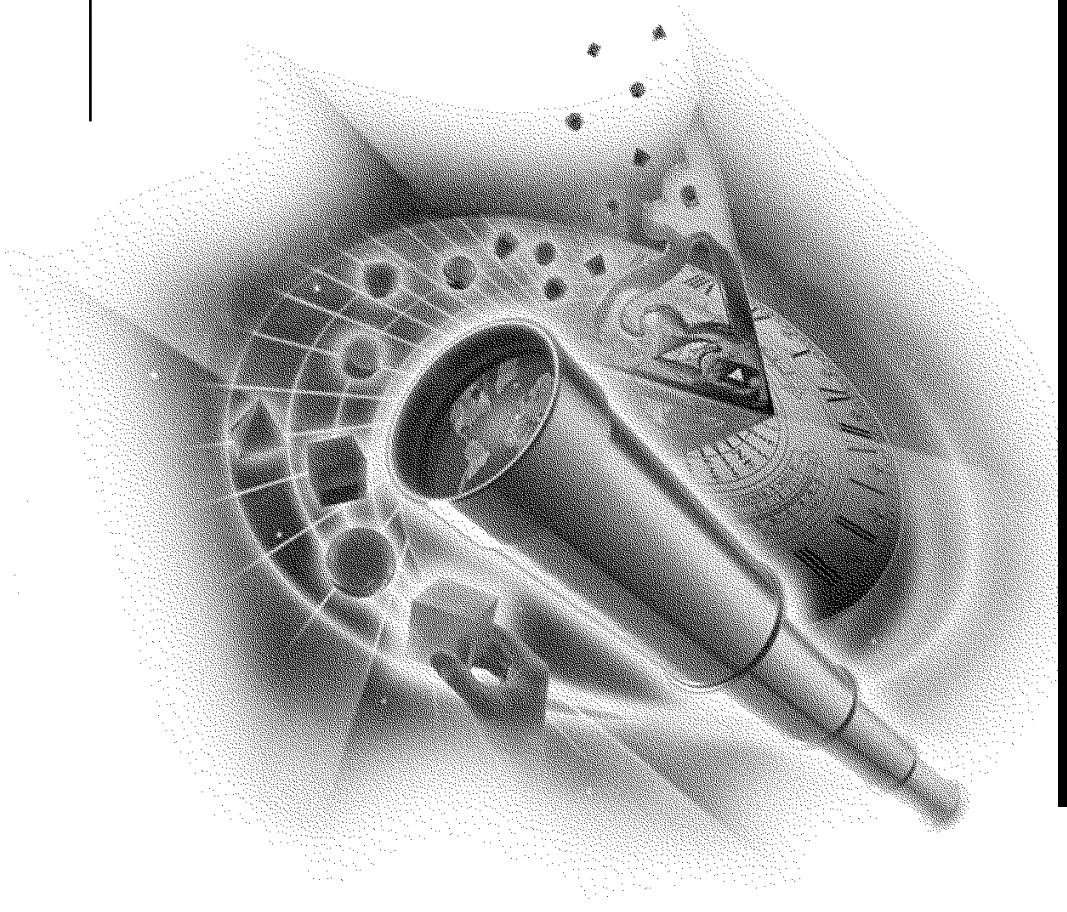
NDS WAN Traffic

Management



NDS™ 7

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, 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.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

This product may require export authorization from the U.S. Department of Commerce prior to exporting from the U.S. or Canada.

U.S. Patent Nos. 4,555,775; 5,157,663; 5,349,642; 5,455,932; 5,553,139; 5,553,143; 5,594,863; 5,608,903; 5,633,931; 5,652,854; 5,671,414; 5,677,851; 5,692,129; 5,758,069; 5,758,344; 5,761,499; 5,781,724; 5,781,733; 5,784,560; 5,787,439; 5,818,936; 5,828,882; 5,832,275; 5,832,483; 5,832,487; 5,859,978; 5,870,739; 5,873,079; 5,878,415; 5,884,304; 5,893,118; 5,903,650; 5,905,860; 5,913,025; 5,915,253; 5,925,108; 5,933,503; 5,933,826; 5,946,467; 5,956,718; 5,974,474. U.S. and Foreign Patents Pending.

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

www.novell.com

NDS WAN Traffic Management
January 2000
104-001273-001

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

Novell Trademarks

For a list of Novell trademarks, see the final appendix of this book.

Third-Party Trademarks

All third-party trademarks are the property of their respective owners.

Contents

NDS WAN Traffic Management

WAN Traffic Manager is a snapin to NWAdmin that allows you to manage server-to-server traffic across WAN links, reducing network costs. You can set policies that govern when different types of traffic are sent and apply the policies to servers and groups of servers.

1 Understanding

Before you can use WAN Traffic Manager effectively, you will need to know some basic information. This section will help you understand key WAN Traffic Manager terms and concepts.

What Is WAN Traffic Manager?

WAN Traffic Manager allows you to manage traffic across WAN links, reducing network costs. It consists of three elements.

- Wtm.nlm

  Wtm.nlm resides on each server in the tree. Before NDS™ sends server-to-server traffic, wtm.nlm reads a WAN Traffic Policy and determines whether that traffic will be sent.

- WAN Traffic Policies

  These are rules that control the generation of NDS traffic. WAN Traffic Policies are text stored as an NDS property value on the NetWare Server object, the LAN Area object, or both.

- An NWAdmin Snapin

  The snapin is the interface to WAN Traffic Manager. It allows you to create or modify policies, to create LAN Area Objects, and to apply policies to LAN Areas or to servers. When WAN Traffic Manager is installed, the schema will include a LAN Area Object and three new detail pages on the NetWare Server object. These are LAN Area Membership, WAN Policies, and Cost.

What Does WAN Traffic Manager Control?

WAN Traffic Manager controls server-to-server traffic generated by NDS. It can restrict traffic based on cost of traffic, time of day, type of traffic, or any combination of these.

Although it was designed to control traffic across WAN links, WAN Traffic Manager can control NDS traffic between any servers in an NDS tree.

WAN Traffic Manager controls only periodic events initiated by NDS, such as replica synchronization, the Janitor process, and the Limber process. It does not control events initiated by administrators or users, nor does it control non-NDS server-to-server traffic such as time synchronization.

Why Use WAN Traffic Manager?

Network directories, such as NDS, create server-to-server traffic. If this traffic crosses wide area network (WAN) links unmanaged, it can needlessly increase costs and overload slow WAN links during high-usage periods.

WAN Traffic Manager lets you control NDS traffic over WAN links by applying policies to NDS.

For example, you might restrict NDS traffic over a WAN link during high-usage times. This shifts high-bandwidth activities to off-hours. You might also limit replica synchronization traffic to times when rates are low to reduce costs.

What Is a WAN Traffic Policy?

A WAN Traffic Policy is a set of rules that control the generation of NDS traffic.

A policy is text stored as an NDS property value on the Server object, the LAN Area object, or both. The policy is interpreted according to a simple processing language.

The property in which this text is stored is Wanman: Wan Policy. Multiple policies are handled as multiple values of this property.

Predefined policy groups are distributed with WAN Traffic Manager. You can use these policies as they are, you can modify these policies, or you can write new policies.

You create or apply policies from the WAN Policies page of the Server object or of the LAN Area object.

You can apply policies to individual servers or you can create LAN Area objects and assign several servers to one of these objects. Any policy that is then applied to the LAN Area object is automatically applied to all servers that are assigned to the object.

2 Planning

You will need to decide which servers need WAN Traffic Manager. This section contains information to help you plan how and where to apply WAN policies.

Which Servers Need WAN Traffic Manager?

WAN Traffic Manager (wtm.nlm) must reside on each server whose traffic you want to control.

If a partition's replica ring includes servers on both sides of a wide area link, you should install WAN Traffic Manager on all servers in that replica ring.

Do You Need a LAN Area Object?

A LAN Area object allows you to easily administer WAN traffic policies for a group of servers.

Once you create a LAN Area object, you can add servers to or remove servers from the LAN Area object. When you apply a policy to the LAN Area, that policy will apply to all the servers in the LAN Area.

Creating a LAN Area object is recommended if you have multiple servers in a LAN that is connected to other LANs by wide area links.

If you do not create a LAN Area object, you must manage each server's WAN traffic individually.

Pre-defined Policy Groups

The following are groups of predefined policies with similar functions:

1-3AM.WGM Group
7AM-6PM.WGM Group
COSTLT20.WGM Group
IPX.WMG Group
TCPIP.WMG Group
SAMEAREA.WMG Group
ONOSPOOF.WMG Group
OPNSPOOF.WMG Group
NDSTTYPS.WMG Group

3 Setting Up

WAN Traffic Manager allows a great deal of flexibility in designing WAN Policies. This section will show you how to create and apply WAN policies that will meet your needs.

Applying WAN Policies

You can apply WAN Policies to an individual server or to a LAN Area object. Policies applied to an individual server manage NDS traffic for that server only. Policies applied to a LAN Area object manage traffic for all servers that belong to the object.

1. Launch NWAdmin.
2. Choose the Network Server object or LAN Area object to which you want to apply a policy.
3. Right-click and choose Details > WAN Policies.
4. Click on the Predefined Policy Groups text box to display a drop-down list of policy groups.
5. Choose the policy group you want to apply.
6. Click Load Group.

   The Policy Load Results box displays the policies in the group and confirms that no errors were found in the policies.
7. Click Advanced > OK.

   The Policies dialog box displays a list of the policies loaded from the policy group.
8. Review the policies by highlighting a policy, clicking Edit, and reading what the policy does.

9. Select any policies that you do not want applied.
10. Click Delete > OK.

    Continue until only the policies you want applied are displayed.
11. Click Close.
12. Click OK to confirm the policies you have applied

    or

    Click Cancel to discard the policies you have applied.

    WAN Traffic Manager will look in wanman.ini for a WAN policy groups section, which contains a 'key = values' statement. 'Key' is the policy name displayed in the snap-in and 'value' is the path to the text files containing delimited policies.

Modifying WAN Policies

WAN Traffic Manager comes with predefined WAN Policies. You can apply these policies to a server or LAN Area object and use them as they are or modify them for your own situation.

1. Launch NWADMIN.
2. Highlight the server or LAN Area object whose WAN policy you want to modify.
3. Right click and select Details > WAN Policies
4. Click Advanced and then OK.
5. In the Policies dialog, highlight the policy and click Edit.

   The policy is displayed in a simple text editor, which allows you to make changes.
6. Edit the policy to conform to your needs.

   To understand the structure of a WAN policy, see "WAN Policy Structure" on page 67.

   To understand the syntax of a WAN policy, see "Constructions Used within Policy Sections" on page 30.
7. Choose Policy > Check Policy.

   This will identify errors in syntax or structure. WAN Traffic Manager will not run policies with errors.

8. Choose Save or Save As under the Policy drop-down menu and save the edited policy, either under its old name or under a new name; then click Policy > Close.
9. Choose and delete any policies you do not want applied; then click Policies > Close.
10. Click OK on the WAN Policies page.

Writing New WAN Policies

WAN Traffic Manager uses policies to control WAN traffic. It can use a predefined policy, a predefined policy that you have modified, or a new WAN policy that you write yourself.

1. Launch NWAdmin
2. Highlight the server or LAN Area object for which you want to write a WAN Policy.
3. Right click and select Details > WAN Policies.
4. Click Advanced and then OK.
5. In the Policies dialog, type the name of the policy and click New.

   A simple text editor appears, which allows you to write the new policy.
6. Enter text for the policy using the prescribed WAN Traffic Manager syntax and structure.

   To understand the structure of a WAN policy, see "WAN Policy Structure" on page 67.

   To understand the syntax of a WAN policy, see "Constructions Used within Policy Sections" on page 30.

   You might also look at one or more predefined policies as examples. In many cases it will be easier to modify one of these existing policies than to write an entirely new one.
7. Choose Policy > Check Policy.

   This will identify errors in syntax or structure. WAN Traffic Manager will not run policies with errors.
8. Choose Policy > Save; then choose Policy > Close.

9. Choose and delete any policies you do not want applied; then choose Policies > Close.
10. Click OK on the WAN Policies page.

Creating a LAN Area Object

A LAN Area object allows you to manage policies for a group of servers. For more information about its purpose, see "Do You Need a LAN Area Object?" on page 15.

1. Launch NWAdmin.
2. Choose the container you want the LAN Area object created in.
3. Right-click and choose Create.
4. Choose the LAN Area object icon and click OK.

   If the LAN Area object class is missing from the list, the schema has not been updated. Make sure you have installed the WAN Traffic Manager snap-in and run Tools > Add Wanman Schema.
5. Click OK.
6. Specify the name of the LAN Area object and, optionally, check one of the two check boxes.

   Check Define Additional Properties if you want to set properties for the LAN Area object you are creating (you can change these properties after you have created the object).

   Check Create Another LAN Area if you want to create another group.
7. Click Create.
8. Continue with one of the topics below:

   "Adding Servers to a LAN Area Object" on page 21

   "Applying WAN Policies" on page 17

Adding Servers to a LAN Area Object

A LAN Area object lets you manage WAN traffic for servers as a group rather than individually. Each WAN policy you apply to the LAN Area object is automatically applied to the servers belonging to it. A server can belong to only one LAN Area object. If the server you are adding already belongs to a LAN Area object, the server will be removed from that object and added to the new object.

1. Launch NWAdmin.
2. Chooose the server you want to add to the LAN Area object.
3. Right-click and choose Details > LAN Area Membership.
4. Using the Browse button, find and click on the LAN Area object you want the server added to.
5. Click OK in the Select Object dialog.
6. Click OK in the LAN Area Membership page.
7. Repeat Steps 1 through 6 for each server you want to add.

   To apply a WAN policy to the LAN Area object, thereby applying the policy to all the servers in the group, see "Applying WAN Policies" on page 17.

Limiting WAN Traffic to Hours You Select

WAN Traffic Manager comes with two predefined WAN Policy groups that limit traffic to specific hours. You can modify these policies to limit traffic to any span of hours you select. The instructions below are for modify the 1-3am group, but you can use a parallel method to accomplish the same thing with the 7am-6pm group.

1. Highlight the server or LAN Area object whose WAN traffic you want to restrict to a range of hours.
2. Right click and select Details > WAN Policies.
3. Click the Predefined Policy Groups text box to display a drop-down list of policy groups.
4. Choose the policy group 1-3am and click Load Group.

   The Policy Load Results box displays the policies in the group and confirms that no errors were found in the policies. Two policies will load:

1-3am and 1-3am, NA. If you plan to manage backlink traffic you will need to follow steps below for 1-3am, NA as well as 1-3am

5. Click Advanced and then OK.

6. In the Policies dialog, highlight the 1-3am policy and click Edit.

   The policy is displayed in a simple text editor, which allows you to make changes. For example, if you want to limit traffic to 2 am to 5 pm rather than from 1 am to 3 am, change the top line in each of the following pairs to the bottom line:

```
/* This policy limits all traffic to between 1 and 3 am*/

/* This policy limits all traffic to between 2 am and 5
 pm*/

Selected := Now.hour > 1 AND Now.hour < 3;

Selected := Now.hour > 2 AND Now.hour < 17;

/* Between 1am and 3am this policy has a high priority */

/* Between 2am and 5pm this policy has a high priority */

/* Between 1am and 3am, send*/

/* Between 2am and 5pm, send*/
```

   Notice that in the comment lines (set off with /* and */), the hour can be designated using a.m and p.m. In the active code, however, it must be designated using 24-hour format. In that case, 5 p.m. becomes 17.

   To better understand the structure of a WAN policy, see "WAN Policy Structure" on page 67.

   To better understand the syntax of a WAN policy, see "Constructions Used within Policy Sections" on page 30.

7. Choose Policy > Check Policy.

   This will identify errors in syntax or structure. WAN Traffic Manager will not run policies with errors.

8. Choose Save As under the drop down menu and save the edited policy under the name 2am-5pm or under a similar name that will indicate the function of the policy; then click Policy > Close.

9. Choose and delete the original 1-3am policy; then click Close in the Policies dialog.

   If you want to keep the original 1-3am policy, you may add the new policy under a different name.

10. Click OK on the WAN Policies page.

Assigning Cost Factors

You can assign destination cost factors to network address ranges or default cost factors to servers or LAN Area objects.

These cost factors let WAN Traffic Manager compare the cost of traffic with certain destinations and manage it using WAN policies.

1. Launch NWAdmin.
2. Highlight the server or the LAN Area object that will be the source of traffic.
3. Choose Object > Details.
4. Choose the Cost page.
5. If you want to assign a default cost to the server or LAN Area object, enter it in the Default Cost field.

   The cost must be a non-negative integer. If supplied, the default cost will be assigned to all destinations on the server or in the LAN Area object that do not fall within a destination address range with an assigned cost. For example, you might specify the cost in monetary units, such as dollars, or packets per second.
6. If you want to assign a cost to a destination address range, click Add.
7. Choose TCP/IP or IPX/SPX.
8. Specify the start address and end address of the range, in appropriate format for TCP/IP or IPX/SPX.
9. Specify the cost as a non-negative integer.
10. Click OK.
11. Click OK again at the LAN Area Object Cost page or at the Server Object Cost page.

Before new cost factors become effective, you must either enter the WANMAN REFRESH IMMEDIATE command at the server console or reload wtm.nlm.

For information about a sample policy that restricts traffic based on cost factor, see “COSTLT20.WGM Group” on page 30.

For information about how to modify a policy, see “Modifying WAN Policies” on page 18.

4 Managing

After you have loaded WAN Traffic Manager, you can use several console commands to manage how policies are applied and reported. This section describes these commands.

WAN Traffic Manager Console Commands

You can use the following commands from the server console once you have loaded wtm.nlm.

- **WANMAN = ON [OFF]**

  Enables [disables] WAN traffic management at this server.

- **WANMAN POLICY DISABLE =** *policyname*

  Unloads a policy. You can use wildcards at the end of *policyname* to disable multiple policies (example: WANMAN POLICY DISABLE = POL*).

- **WANMAN POLICY ENABLE = *policyname***

  Loads a policy. You can use wildcards at the end of *policyname* to enable multiple policies (example: WANMAN POLICY ENABLE = POL*).

- **WANMAN LOGFILE = ON [OFF]**

  Enables [disables] the logging of activities to sys:system\wanman.log.

- **WANMAN LOGFILE MAX SIZE = *filesize***

  Sets the maximum size of the log file in kilobytes. The default size of the log file is 512 KB, and can range from 10 KB to 10 .24 MB. When the file reaches the maximum size, it is renamed wanman.old and a new log file is created.

- **WANMAN REFRESH IMMEDIATE**

  Initializes WAN Traffic Manager.

5 Troubleshooting

WAN Traffic Manager generates a log file that shows various activities. This section explains how to set up and view this file for troubleshooting.

Troubleshooting WAN Traffic Manager

Wtm.nlm generates a log file for troubleshooting WAN Traffic Manager. This log file is named wanman.log and is found in the sys:system directory of each server running wtm.nlm. You can use a text editor to view the log file.

Various activities are displayed on the server screen and copied to the log file, such as:

- Refreshing of policies
- Requests from NetWare Administrator
- Selection of policies
- Provider results

The default size of the log file is 512 KB, and can range from 10KB to 10 .24 MB. You can change the size of the log file with a WAN Traffic Manager console command. Once the log file reaches its size limit, it is renamed wanman.old and a new one is created.

Viewing WAN Traffic Messages

You can view WAN traffic messages related to specific WAN traffic types.

1. Execute the following command at the console:

   ```
   SET DSTRACE=+WANMAN
   ```

   To disable this feature, you can execute the following command at the console:

   ```
   SET DSTRACE=-WANMAN
   ```

2. Press Alt+Esc to switch to the Directory Services screen and view the messages.

3. To copy NDS* messages to a log file, use the following commands at the server console:

   ```
   SET NDS TRACE FILENAME =  volume:\path\filename
   SET NDS TRACE TO FILE = ON
   ```

   If the logifle does not already exist, this command will create one.

   You can use a text utility such as Microsoft* Notepad to view the log file.

6 Policies

1-3AM.WGM Group

The policies in this group limit the time traffic can be sent to between 1 and 3 a.m. There are two policies.

- 1 - 3 am, NA

  This policy limits the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization to these hours.

- 1 - 3 am

  This policy limits all other traffic to these hours.

To restrict all traffic to these hours, both policies must be applied.

7AM-6PM.WGM Group

The policies in this group limit the time traffic can be sent to between 7 a.m. and 6 p.m. There are two policies.

- 7 am - 6 pm, NA

  This policy limits the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization to these hours.

- 7 am - 6 pm

  This policy limits all other traffic to these hours.

To restrict all traffic to these hours, both policies must be applied.

COSTLT20.WGM Group

The policies in this group allow only traffic to be sent that has a cost factor below 20. There are two policies.

- Cost <20, NA

  This policy prevents the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization unless the cost factor is less than 20.

- Cost < 20

  This policy prevents all other traffic unless the cost factor is less than 20.

To prevent all traffic with a cost factor of 20 or more, both policies must be applied.

Connection Management

Servers in a replica ring require a highly-secure connection for transferring NetWare Core Protocol* (NCP*) packets. These secure connections, called *virtual client* connections, are established by the connection management process.

The connection management process might also need to establish a virtual client connection for schema synchronization or Backlink processes. Time synchronization might also require such a connection, depending on the configuration of time services.

Constructions Used within Policy Sections

A WAN Policy contains three sections: Declaration, Selector, and Provider. Contructions used within the Declarations section are not described here. The following statements and constructions can be used, except as noted, in the Selector and Provider sections.

Comments

Comments are not executed, but are used to document the policy.

Comments can be set off using /* at the beginning of the line and */ at the end. For example:

/* This is a comment. */

Comments can also be distinguished by // at the end of the line before a comment. For example:

```
IF L2 > L3 THEN //This is a comment.
```

IF-THEN Statement

IF-THEN statements are used to run a block of declarations conditionally.

Examples:

```
IF <boolean expression> THEN <declarations>
END

IF <boolean expression> THEN <declarations>
ELSE  <declarations>
END

IF <boolean expression> THEN <declarations>
ELSIF <boolean expression> THEN <declarations>
END
```

IF <boolean expression>THEN

This is the first clause in an IF-THEN statement. The boolean expression is evaluated for a TRUE or FALSE result. If TRUE, the declarations that immediately follow are run. If FALSE, execution jumps to the next corresponding ELSE, ELSIF, or END declaration.

ELSE

This declaration marks the beginning of declarations that run if all corresponding preceeding IF-THEN and ELSIF statements results in FALSE. For example:

```
IF <boolean expression> THEN <statements>
ELSIF <boolean expression> THEN <statements>
ELSIF <boolean expression> THEN <statements>
ELSE <statements>
END
```

ELSIF<boolean expression> THEN

The boolean expression is evaluated if the preceding IF-THEN declaration returns a FALSE. The ELSIF declaration is evaluated for a TRUE or FALSE result. If TRUE, the declarations that follow are run. If FALSE, execution jumps to the next corresponding ELSE, ELSIF, or END declaration. For example:

```
IF <boolean expression> THEN <statements>
ELSIF <boolean expression> THEN <statements>
ELSIF <boolean expression> THEN <statements>
END
```

END

The END declaration terminates an IF-THEN construction.

RETURN

The RETURN command gives the results of the Selector and Provider sections.

Selector

In a Selector section, the RETURN declaration provides the integer result used as a weight for the policy. RETURN assigns a policy weight between 0-100, where 0 means do not use this policy, 1-99 means use this policy if no other policy returns a higher value, and 100 means use this policy. If no RETURN declaration is made in a Selector section, a default value of 0 will be returned.

A semicolon is required to terminate the declaration. For example:

```
RETURN 49;
RETURN L2;
RETURN 39+7;
```

Provider

In a Provider section, the RETURN declaration provides the SEND or DONT_SEND result. If no RETURN declaration is made, a default value of SEND will be returned.

A semicolon is required to terminate the declaration. Examples:

```
RETURN SEND;
RETURN DONT_SEND;
RETURN L1;
```

Assignment

The assignment declaration changes the value of a symbol using the := characters. The defined variable or system variable is stated first, then the := with a value, variable, or operation following. For example:

```
<variable>.<field>:=<expression>; <variable>:=<expression>;
```

t1 and t2 are of type TIME, i1 and i2 are type INTEGER, b1, b2 are BOOLEAN valid assignments:

```
t1 := t2;
b1 := t1 < t2;
i1 := t1.mday - 15;
b2 := t2.year < 2000
```

Invalid assignments:

```
b1 := 10 < i2 < 12;
```

(10 < i2) is boolean, and a BOOLEAN cannot be compared to an INTEGER.

You might use b1 := (10 < i2) AND (i2 < 12); instead:

```
  b2 := i1;
```

b2 is BOOLEAN, and i1 is INTEGER, therefore, they are incompatible types.

You might use b2 := i1 > 0; instead.

The assignment declaration must be terminated with a semicolon.

Strict type checking is performed. You are not allowed to assign an INT to a TIME variable.

Arimetic Operators

You can include arithmetic operators in assignment declarations, RETURN declarations, or IF constructions. The valid operators are:

- Addition (+)
- Subtraction (-)
- Division (/)
- Multiplication (*)
- Modula (MOD)

Use only INT variable types with arithmetic operators. Do not use TIME, NETADDRESS, and BOOLEAN variable types in arithmetic expressions.

Avoid operations that result in values outside of the range -2147483648 to +2147483648 or division by zero.

Relational Operators

You can use relational operators in IF constructions. The valid operators are:

- Equal to (=)
- Not equal to (<>)
- Greater than (>)
- Greater than or equal to (>=)
- Less than (<)
- Less than or equal to (<=)

You can use any relational operators with TIME and INT variable types. You can also use <> and = with NET ADDRESS and BOOLEAN variable types.

Logical Operators

The valid operators are:

- AND
- OR
- NOT
- Less than (<)

- Greater than (>)
- Equal to (=)

Bitwise Operators

You can use bitwise operators on INT variable types to return an integer value. The valid operators are:

- BITAND
- BITOR
- BITNOT

Complex Operations

The following precedence rules are enforced when processing complex expressions. Operators with the same precedence order are processed left-to-right. The order is as follows:

- Parenthesis
- Unary (+/-)
- BITNOT
- BITAND
- BITOR
- Multiplication, division, MOD
- Addition, subtraction
- Relational (>, >=, <, <=, =)
- NOT
- AND
- OR

If you are not certain of precedence, use parentheses. For example, if A, B, and C are integers or variables, A<B<C is not allowed. A<B would return a boolean value, not an integer value, which cannot be compared to an integer C. However, (A<B) AND (B<C) would be syntactically correct.

PRINT

You can use PRINT declarations to send text and symbol values to the server's WAN Traffic Manager display screen and to the log file.

PRINT statements can have any number of arguments that may be literal strings, symbol names or members, integer values, or boolean values, separated by commas.

You must enclose literal strings in double quotes (”). PRINT declarations must end in a semicolon (;). For example:

PRINT ”INT=”,10,”BOOL=”,TRUE,”SYM=”,R1;

TIME and NETADDRESS variables use formatted print declarations. TIME symbols are printed as follows:

```
m:d:y h:m
```

NETADDRESS variables are printed as follows:

Type length data

is either IP or IPX, *length* is the number of bytes, and *data* is the hexadecimal address string.

Cost/Cost Factor

Cost or Cost Factor is used by WAN policies to determine the relative expense of WAN traffic. In policies, you can use DestCost to determine whether to send traffic or not.

A cost factor is expressed as expense per unit of time. It can be in any units as long as the same units are used consistently on each Cost Detail page and in each WAN traffic policy. Thus you can use dollars per hour, cents per minute, yen per second, or any other ratio of expense to time, as long as you use that ratio exclusively.

Cost or Cost Factor must be given as a non-negative integer.

You can assign destination cost factors representing the relative expense of traffic to particular address ranges. Therefore, you can assign cost for an entire group of servers in one declaration. You can also assign a default cost factor, to be used when no cost is specified for a destination.

The cost factors are stored in an address cost list on the server, in the NetWare Server object, or in the LAN Area object.

When wtm.nlm receives a get WAN policy request, it looks up the destination address for the proposed traffic. If that address falls within the address range of the server, it assigns the associated cost to the system variable DestCost. If not, it checks the LAN Area object's address range. If the destination address falls within the LAN Area object's range, it assigns that associated cost to DestCost.

If no cost is assigned for the destination, the default cost is used. If you have specified no default cost for the server or LAN Area object, a value of -1 is assigned.

Declaration Section

The Declaration section of a policy contains definitions of local variables and variables coming in through a client request. These definitions are used within the Selector and Provider sections. These variables are stored along with system-defined variables.

Variable declarations are separated by a semicolon. Multiple declarations for the same type can be combined in one line or wrapped to the next line: they are not line-sensitive. A sample Declaration section is shown below:

```
REQUIRED INT R1;
REQUIRED TIME R2;
REQUIRED BOOLEAN R3,R4;
REQUIRED NETADDRESS R5,R6;
OPTIONAL INT P1 := 10;
OPTIONAL BOOLEAN := FALSE;
LOCAL INT L1 :=10;
LOCAL INT L2;
LOCAL TIME L3;
LOCAL BOOLEAN L4 :=TRUE, L5 :=FALSE;
LOCAL NETADDRESS L6;
```

The required and optional declarations are specific to a particular traffic type. Policies that do not contain the required variables will not run. The optional declarations must have a value to provide a default if none is passed in. WAN Traffic Manager provides system symbols (predefined variables) for use with all traffic types.

Each declaration consists of three parts: scope, type, and a list of name/optional value pairs.

Valid scopes are REQUIRED, OPTIONAL, LOCAL, and SYSTEM.

Valid types are INT, BOOLEAN, TIME, or NETADDRESS. You cannot assign values to TIME and NETADDRESS types in the Declaration section. If these types do not already have a value, they receive their values in the Selector or Provider sections. Only single types are initialized in the Declaration section.

Values in a declaration must be constants rather than variables or expressions. Thus, the declaration 'LOCAL INT L2 := L3;' is not allowed. A value initializing a variable in the declaration section may be changed in the Selector and Provider sections of the policy.

Heartbeat

Heartbeat ensures that Directory objects are consistent among all replicas of a partition. This means that any server with a copy of a partition must communicate with the other servers to check the consistency.

This process runs by default once every 30 minutes on every server that contains a replica of a partition.

INT Variable Type

The INT variable type is used for integer values from -2147483648 to +2147483647. The value will be indeterminate if it is not set in a declaration or a WAN policy request. WAN Traffic Manager uses several predefined variables based on the INT variable type. These predefined variables are:

- INT DayOfWeek
- INT DayOfWeekUTC
- INT DestCost
- INT TrafficType
- INT<NDS_____>

INT DayOfWeek

The day of week associated with Now. The wday portion of time is kept in the DayOfWeek system symbol, which is Sunday . . . Saturday with values of 0 through 6.

INT DayOfWeekUTC

The day of week associated with NowUTC. The wday portion of time is kept in the DayOfWeek system symbol, which is Sunday . . . Saturday with values of 0 through 6.

INT DestCost

An administrator-supplied cost for the address. It is generated by checking the DestAddress against the WANMAN:Cost property of the server and then the LAN Area object. If the DestAddress falls within one of the ranges of addresses entered by the administrator, then the cost associated with the range of the DestAddress is the value of DestCost. If the DestAddress does not fall within one of the ranges, WAN Traffic Manager checks whether the administrator has supplied a

WANMAN:Default Cost

attribute. If so, DestCost will be equal to this default. DestCost is always positive. Thus, -1 is recognized as not falling in any range and means that no WANMAN:Default Cost attribute was supplied. This allows the policy writer to tell whether or not to rely on the cost.

If a conflict between the WANMAN:Cost or WANMAN:Default Cost attributes of the server and the LAN Area object, that of the server will be used.

This variable is not available for NO_ADDRESSES policies.

INT TrafficType

Reflects the traffic type of the GetWanPolicy request for which the policy is being run. For example, the folllowing policy specifies a TrafficType of NDS_SYNC:

IF TrafficType=NDS_SYNC THEN <action> END.

IPX.WMG Group

The policies in this group allow only IPX traffic. There are two policies.

IPX, NA

This policy prevents the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization unless the traffic that would be generated is IPX.

IPX

This policy prevents all other traffic unless the traffic is IPX.

To prevent all non-IPX traffic, both policies must be applied.

IPX/SPX Address

An IPX address looks like 11223344:5566778899AA:BBCC (network, node, socket) where the numbers are in hexadecimal. X.type is IPX, X.a is 0x11 (0x indicates hex), X.b is 0x22, X.c is 0x33, X.d is 0x44, X.e is 0x55, X.f is 0x66, X.g is 0x77, X.h is 0x88, X.i is 0x99, X.j is 0xAA, X.k is 0xBB, and X.l is 0xCC.

X is a variable of type NETADDRESS.

LOCAL Scope

Variables defined as LOCAL in scope can be used in multiple sections, but only once within the Declaration section.

LOCAL scope variables exist only for a particular policy; that is, their values are not returned to the calling client.

All parameter types can be defined. However, since TIME and NETADDRESS types cannot be initialized in the Declaration section, do not assign values to these types.

Limber

Limber ensures that a server's replica pointer table is updated when that server's server name or address is changed. Such changes occur when:

- The server is rebooted with a new server name or IPX internal address in the autoexec.ncf file.
- When an address is added for an additional protocol.

When a server is booted, the limber process compares the server's name and IPX address with those stored in the replica pointer table. If either are different, NDS automatically updates all replica pointer tables that contain a listing of that server.

The limber process also checks that the tree name is correct for each server in a replica ring.

Limber runs five minutes after the server boots up and then every 3 hours.

NA

NA indicates that this is a policy that contains a NO_ADDRESSES statement. Five of the NDS Traffic Types require NO_ADDRESSES statements. These are NDS_Backlinks, NDS_Janitor, NDS_Limber, NDS_Schema_Sync, and NDS_Check_Login_Restrictions.

NDS Traffic

The following NDS processes generate server-to-server traffic:

- Replica synchronization
- Schema synchronization
- Heartbeat
- Limber
- Backlink
- Connection management
- Server status check

NDSTTYPS.WMG

The policies in this group are sample policies for various NDS* traffic types. They contain the variables NDS passes in a request of this TrafficType.

- Sample NDS_SYNC

  Sample policy for NDS_SYNC traffic type

- Sample NDS_BACKLINKS

  Sample policy for NDS_BACKLINKS traffic type

- Sample NDS_BACKLINK_OPEN

  Sample policy for NDS_BACKLINK_OPEN traffic type

- Sample NDS_SCHEMA_SYNC

  Sample policy for NDS_SCHEMA_SYNC

- Sample NDS_SCHEMA_SYNC_OPEN

  Sample policy for NDS_SCHEMA_SYNC_OPEN traffic type

- Sample NDS_LIMBER

  Sample policy for NDS_LIMBER traffic type

- Sample NDS_LIMBER_OPEN

  Sample policy for NDS_LIMBER_OPEN traffic type

- Sample NDS_CHECK_LOGIN_R

  Sample policy for NDS_CHECK_LOGIN_RESTRICTIONS traffic type

- Sample NDS_CHECK_LOGIN_R_OPEN

  Sample policy for NDS_CHECK_LOGIN_RESTRICTION_OPEN traffic type

- Sample NDS_JANITOR

  Sample policy for NDS_JANITOR traffic type

- Sample NDS_JANITOR_OPEN

  Sample policy for NDS_JANITOR_OPEN traffic type

- Sample Catch All without Addresses

  Sample catch all policy for traffic types without addresses

- Sample Catch All with Addresses

NDS_BACKLINKS

Before NDS checks any backlinks or external references, it queries WAN Traffic Manager to see if this is an acceptable time for this activity. The NDS_BACKLINKS does not have a destination address; it requires a NO_ADDRESSES policy. If WAN Traffic Manager returns DONT_SEND, backlink checking will be put off and rescheduled. The following variables are supplied.

- Last (Input Only, Type TIME)

  Time of the last round of backlink checking since DS.NLM was started. When NDS starts, Last is initialized to zero. If NDS_BACKLINKS returns SEND, Last is set to the current time after NDS finishes backlinking.

- Version (Input Only, Type INTEGER)

  The version of NDS.

- ExpirationInterval (Output Only, Type INTEGER)

  The expiration interval for all connections created while backlinking.

  <0, 0 Use the default expiration interval (default)

  >0 Expiration interval to be assigned to this connection

- Next (Output Only, Type TIME)

  This variable indicates when NDS should schedule the next round of backlink checking.

  In past, 0 Use the default scheduling

  In future Time when backlinking should be scheduled

- CheckEachNewOpenConnection (Output Only, Type INTEGER)

  This variable tells NDS what to do if it needs to create a new connection while doing backlinking. CheckEachNewOpenConnection is initialized to zero. Zero (0) indicates that NDS should not query WAN Traffic Manager, but should proceed with making the connection as in normal operations. One (1) indicates that NDS should query WAN Traffic Manager using the NDS_BACKLINK_OPEN traffic type. If WAN Traffic Manager returns SEND, NDS proceeds to make the connection; otherwise it returns an error and continues with other backlinking tasks.

Two (2) indicates that NDS should not query WAN Traffic Manager, but should act as though the connection had failed. This variable enables policies that can backlink normally, decide on a connection-by-connection basis, or that only reuse existing connections.

0 Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default)

1 Call WAN Traffic Manager and let the policies decide whether to allow the connection

2 Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail

- CheckEachAlreadyOpenConnection (Output Only, Type INTEGER)

  This variable tells NDS what to do if it needs to reuse a connection it believes is already open while doing backlinking. CheckEachAlreadyOpenConnection is initialized to zero. Zero (0) indicates that NDS should not query WAN Traffic Manager, but should proceed with reusing the connection as in normal operations. One (1) indicates that NDS should query WAN Traffic Manager using the NDS_BACKLINK_OPEN traffic type. If WAN Traffic Manager returns SEND, NDS proceeds to reuse the connection; otherwise it returns an error and continues with other backlinking tasks. Two (2) indicates that NDS should not query WAN Traffic Manager, but should act as though the connection has failed and cannot be reopened. This variable is used in environments that spoof connections to enable policies to decide if the connection is really up or if it is being spoofed.

  0 Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default)

  1 Call WAN Traffic Manager and let the policies decide whether to allow the connection or not

  2 Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail.

NDS_BACKLINK_OPEN

NDS_BACKLINK_OPEN is a traffic type that is only used if either CheckEachNewOpenConnection or CheckEachAlreadyOpenConnection was set to 1 during the corresponding NDS_BACKLINKS query.

This query is generated whenever CheckEachNewOpenConnection is 1 and NDS* needs to open a new connection for backlinking or CheckEachAlreadyOpenConnection is 1 and NDS needs to reuse an already existing connection.

- Version (Input Only, Type INTEGER)

  The version of NDS.

- ExpirationInterval (Input and Output, Type INTEGER)

  If ConnectionIsAlreadyOpen is TRUE, ExpirationInterval will be set to the expiration interval already set on the existing connection. Otherwise it will be set to the ExpirationInterval assigned in the NDS_BACKLINKS query. A zero value indicates that the default (2 hours) should be used. On exit, the value of this variable is assigned as the expiration interval for the connection

  <0,0 Use the default expiration interval (default)

  >0 Expiration Interval to be assigned to this connection

- ConnectionIsAlreadyOpen (Input Only, Type BOOLEAN)

  This variable is TRUE if NDS wishes to reuse an existing connection and FALSE if it needs to create a new connection.

  - TRUE

    NDS believes it already has a connection to this address and can reuse that connection.

  - FALSE

    NDS does not have a connection to this address and must create one.

- ConnectionLastUsed (Input Only, Type TIME)

  If ConnectionIsAlreadyOpen is TRUE, then ConnectionLastUsed is the last time that a packet was sent from NDS using this connection. Otherwise, it will be zero.

  - TRUE

    ConnectionLastUsed is the time that NDS last sent a packet on this connection.

  - FALSE

    ConnectionLastUsed will be zero.

NDS_CHECK_LOGIN_RESTRICTIONS

Before NDS checks a login restriction, it queries WAN Traffic Manager to see if this is an acceptable time for this activity. The traffic type NDS_CHECK_LOGIN_RESTRICTIONS does not have a destination address; it requires a NO_ADDRESSES policy. If WAN Traffic Manager returns DONT_SEND, the check will error out. The following variables are supplied:

- Version (Input Only, Type INTEGER)

  The Version of NDS.

- Result (Output Only, Type INTEGER)

  If the result of NDS_CHECK_LOGIN_RESTRICTIONS is DONT_SEND, then the following values will be returned to the operating system.

  0 Login is allowed

  1 Login is not allowed during the current time block

  2 Account is disabled or expired

  3 Account has been deleted

- ExpirationInterval (Output Only, Type INTEGER)

  The expiration interval that should be assigned to this connection.

  <0, 0 Use the default expiration interval (default)

  >0 Expiration Interval to be assigned to this connection

- CheckEachNewOpenConnection (Output Only, Type INTEGER)

  0 Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default)

  1 Call WAN Traffic Manager and let the policies decide whether to allow the connection or not

  2 Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail

- CheckEachAlreadyOpenConnection (Output Only, Type INTEGER)

  0 Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default)

  1 Call WAN Traffic Manager and let the policies decide whether to allow the connection or not

  2 Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail

NDS_CHECK_LOGIN_RESTRICTIONS_OPEN

NDS_CHECK_LOGIN_RESTRICTION_OPEN is only used if either CheckEachNewOpenConnection or CheckEachAlreadyOpenConnection was set to one during the corresponding NDS_CHECK_LOGIN_RESTRICTIONS query. This query is generated whenever CheckEachNewOpenConnection is one and NDS* needs to open a new connection pursuant to running limber. This query is generated whenever CheckEachNewOpenConnection is one and NDS needs to open a new connection pursuant to doing checking the login restriction or CheckEachAlreadyOpenConnection is one and NDS needs to reuse an already existing connection. The following variables are provided:

- Version (Input Only, Type INTEGER)

  The version of NDS.

- ExpirationInterval (Input and Output, Type INTEGER)

  <0, 0 Use the default expiration interval (default)

  >0 Expiration Interval to be assigned to this connection

- ConnectionIsAlreadyOpen (Input Only, Type BOOLEAN)
  - TRUE

    NDS believes it already has a connection to this address and can reuse that connection
  - FALSE

    NDS does not have a connection to this address and must create one
- ConnectionLastUsed (Input Only, Type TIME)

  If ConnectionIsAlreadyOpen is TRUE, then ConnectionLastUsed is the last time that a packet was sent from NDS using this connection. Otherwise, it will be zero.
  - TRUE

    ConnectionLastUsed is the time that NDS last sent a packet on this connection
  - FALSE

    ConnectionLastUsed will be zero

NDS_JANITOR

Before NDS runs the janitor, it queries WAN Traffic Manager to see if this is an acceptable time for this activity. The NDS_JANITOR does not have a destination address; it requires a NO_ADDRESSES policy. If WAN Traffic Manager returns DONT_SEND, janitor work will be put off and rescheduled. The following variables are supplied.

- Last (Input Only, Type TIME)

  Time of the last round of janitor work since DS.NLM was started. When NDS starts, Last is initialized to zero. If NDS_JANITOR returns SEND, Last is set to the current time after NDS finishes the janitor.
- Version (Input Only, Type INTEGER)

  The version of NDS.

- ExpirationInterval (Output Only, Type INTEGER)

  The expiration interval for all connections created while running the janitor.

  <0, 0 Use the default expiration interval (default)

  >0 Expiration Interval to be assigned to this connection

- Next (Output Only, Type TIME)

  This variable indicates when NDS should schedule the next round of janitor work.

  In the past, 0, use the default scheduling

  In the future, time when janitor should be scheduled

- CheckEachNewOpenConnection (Output Only, Type INTEGER)

  This variable tells NDS what to do if it needs to create a new connection while running the janitor. CheckEachNewOpenConnection is initialized to zero. Zero (0) indicates that NDS should not query WAN Traffic Manager, but should proceed with making the connection as in normal operations. One (1) indicates that NDS should query WAN Traffic Manager using the NDS_JANITOR_OPEN traffic type. If WAN Traffic Manager returns SEND, NDS proceeds to make the connection; otherwise it returns an error and continues with other janitor tasks. Two (2) indicates that NDS should not query WAN Traffic Manager, but should act as though the connection had failed. This variable enables policies that can run the janitor normally, decide on a connection-by-connection basis, or that only reuse existing connections.

  0 Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default)

  1 Call WAN Traffic Manager and let the policies decide whether to allow the connection

  2 Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail

- CheckEachAlreadyOpenConnection (Output Only, Type INTEGER)

  This variable tells NDS what to do if it needs to reuse a connection it believes is already open while running the janitor. CheckEachAlreadyOpenConnection is initialized to zero. Zero (0) indicates that NDS should not query WAN Traffic Manager, but should proceed with reusing the connection as in normal operations. One (1)

indicates that NDS should query WAN Traffic Manager using the NDS_JANITOR_OPEN traffic type. If WAN Traffic Manager returns SEND, NDS proceeds to reuse the connection; otherwise it returns an error and continues with other janitor tasks. Two (2) indicates that NDS should not query WAN Traffic Manager, but should act as though the connection has failed and cannot be reopened. This variable is used in environments that spoof connections to enable policies to decide if the connection is really up or if it is being spoofed.

0 Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default)

1 Call WAN Traffic Manager and let the policies decide whether to allow the connection or not

2 Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail

NDS_JANITOR_OPEN

NDS_JANITOR_OPEN is only used if either CheckEachNewOpenConnection or CheckEachAlreadyOpenConnection was set to 1 during the corresponding NDS_JANITOR query. This query is generated whenever CheckEachNewOpenConnection is 1 and NDS* needs to open a new connection pursuant to doing backlinking, or CheckEachAlreadyOpenConnection is one and NDS needs to reuse an already existing connection.

- Version (Input Only, Type INTEGER)

  The version of NDS.

- ExpirationInterval (Input and Output, INTEGER)

  If ConnectionIsAlreadyOpen is TRUE, ExpirationInterval will be set to the expiration interval already set on the existing connection. Otherwise, it will be set to the ExpirationInterval assigned in the NDS_JANITOR query. A zero value indicates that the default (2 hours, 10 seconds) should be used. On exit, the value of this variable is assigned as the expiration interval for the connection.

  <0, 0 Use the default expiration interval (default)

  >0 Expiration Interval to be assigned to this connection

- ConnectionIsAlreadyOpen (Input Only, Type BOOLEAN)

  This variable is TRUE if NDS wishes to reuse an existing connection and FALSE if it needs to create a new connection.

  - TRUE

    NDS believes it already has a connection to this address and can reuse that connection

  - FALSE

    NDS does not have a connection to this address and must create one

- ConnectionLastUsed (Input Only, Type TIME)

  If ConnectionIsAlreadyOpen is TRUE, then ConnectionLastUsed is the last time that a packet was sent from NDS* using this connection. Otherwise, it will be zero.

  - TRUE

    ConnectionLastUsed is the time that NDS last sent a packet on this connection

  - FALSE

    ConnectionLastUsed will be zero

NDS_LIMBER

Before NDS runs limber, it queries WAN Traffic Manager to see if this is an acceptable time for this activity. The traffic type NDS_LIMBER does not have a destination address; it requires a NO_ADDRESSES policy. If WAN Traffic Manager returns DONT_SEND, limber will be put off and rescheduled. The following variables are supplied.

Last(Input Only, Type TIME)

Time of last limber since NDS started.

- Version(Input Only, Type INTEGER)

  The version of NDS.

- ExpirationInterval (Output Only, Type INTEGER)

  The expiration interval for all connections created while running limber checks.

  <0, 0 Use the default expiration interval (default)

  >0 Use the default expiration interval (default)

- CheckEachNewOpenConnection (Output Only, Type INTEGER)

  0 Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default)

  1 Call WAN Traffic Manager and let the policies decide whether to allow the connection or not

  2 Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail

- CheckEachAlreadyOpenConnection(Output Only, Type INTEGER)

  0 Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default)

  1 Call WAN Traffic Manager and let the policies decide whether to allow the connection or not

  2 Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail

- Next (Output Only, Type TIME)

  Time for the next round of limber checking. If this is not set, NDS_LIMBER will use the default.

NDS_LIMBER_OPEN

NDS_LIMBER_OPEN is only used if either CheckEachNewOpenConnection or CheckEachAlreadyOpenConnection was set to 1 during the corresponding NDS_LIMBER query. This query is generated whenever CheckEachNewOpenConnection is 1 and NDS* needs to open a new connection pursuant to running limber. This query is generated whenever CheckEachNewOpenConnection is 1 and NDS needs to open a new connection pursuant to doing schema synchronization or CheckEachAlreadyOpenConnection is 1 and NDS needs to reuse an already existing connection.

- Version (Input Only, Type INTEGER)

  The version of NDS.

- ExpirationInterval (Input and Output, Type INTEGER)

  The expiration interval that should be assigned to this connection.

  <0, 0 Use the default expiration interval (default)

  >0 Expiration Interval to be assigned to this connection

- ConnectionIsAlreadyOpen (Input Only, BOOLEAN)

  TRUE NDS believes it already has a connection to this address and can reuse that connection

  FALSE NDS does not have a connection to this address and must create one

- ConnectionLastUsed (Input Only, Type TIME)

  If ConnectionIsAlreadyOpen is TRUE, then ConnectionLastUsed is the last time that a packet was sent from DS using this connection. Otherwise, it will be zero.

  TRUE ConnectionLastUsed is the time that NDS last sent a packet on this connection

  FALSE ConnectionLastUsed will be zero

NDS_SCHEMA_SYNC

Before NDS synchronizes schema, it queries WAN Traffic Manager to see if this is an acceptable time for this activity. The traffic type NDS_SCHEMA_SYNC does not have a destination address; it requires a NO_ADDRESSES policy. If WAN Traffic Manager returns DONT_SEND, schema syncronization will be put off and rescheduled. The following variables are supplied:

- Last (Input Only, Type TIME)

  Time of the last successful schema synchronization to all servers.

- Version(Input Only, Type INTEGER)

  The version of NDS.

- ExpirationInterval(Output Only, Type INTEGER)

  The expiration interval for all connections created while synchronizing the schema.

  <0, 0 Use the default expiration interval (default)

  \>0 Expiration Interval to be assigned to this connection

- CheckEachNewOpenConnection(Output Only, Type INTEGER)

  0 Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default)

  1 Call WAN Traffic Manager and let the policies decide whether to allow the connection

  2 Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail

- CheckEachAlreadyOpenConnection (Output Only, Type INTEGER)

  0 Return Success without calling WAN Traffic Manager, allowing the connection to proceed normally (default)

  1 Call WAN Traffic Manager and let the policies decide whether to allow the connection

  2 Return ERR_CONNECTION_DENIED without calling WAN Traffic Manager, causing the connection to fail

NDS_SCHEMA_SYNC_OPEN

NDS_SCHEMA_SYNC_OPEN is only used if either CheckEachNewOpenConnection or CheckEachAlreadyOpenConnection was set to 1 during the corresponding NDS_SCHEMA_SYNC query. This query is generated whenever CheckEachNewOpenConnection is 1 and NDS* needs to open a new connection pursuant to doing schema synchronization or CheckEachAlreadyOpenConnection is 1 and NDS needs to reuse an already existing connection.

- Version (Input Only, Type INTEGER)

  The version of NDS.

- ExpirationInterval (Input and Output, INTEGER)

  The expiration interval that should be assigned to this connection.

  <0,0 Use the default expiration interval (default)

  >0 Expiration Interval to be assigned to this connection

- ConnectionIsAlreadyOpen (Input Only, BOOLEAN)

  TRUE NDS believes it already has a connection to this address and wishes to reuse that connection

  FALSE NDS does not have a connection to this address and wishes to create one

- ConnectionLastUsed (Input Only, Type TIME)

  If ConnectionIsAlreadyOpen is TRUE, then ConnectionLastUsed is the last time that a packet was sent from DS using this connection. Otherwise, it will be zero.

  TRUE ConnectionLastUsed is the time that NDS* last sent a packet on this connection

  FALSE ConnectionLastUsed will be zero

NDS_SYNC

Whenever NDS* needs to synchronize a replica, it makes a query to WAN Traffic Manager using the traffic type NDS_SYNC. The following variables are provided by NDS for use in WAN policies.

- Last (Input Only, Type TIME)

  Time of the last successful synchronization to this replica.

- Version (Input Only, Type INTEGER)

  The version of NDS.

- ExpirationInterval (Output Only, Type INTEGER)

  The expiration interval for the connection to the server holding the updated replica.

  <0, 0 Use default expiration interval (default)

  >0 Expiration Interval to be assigned to this connection

NETADDRESS Variable Type

NETADDRESS variables must receive their values in the Selector or Provider sections. Therefore, do not assign values to NETADDRESS variables in the declaration.

WAN Traffic Manager uses two predefined variables based on the NETADDRESS variable type. These predefined variables are:

- NETADDRESS DestAddress
- NETADDRESS SrcAddress

NETADDRESS is declared as a structure with the following elements:

Type	IPX or IP
a	(IP or IPX)
b	0-255 (IP or IPX)
c	0-255 (IP or IPX)
d	0-255 (IP or IPX)

Type	IPX or IP
e	0-255 (IPX only)
f	0-255 (IPX only)
g	0-255 (IPX only)
h	0-255 (IPX only)
i	0-255 (IPX only)
j	0-255 (IPX only)
k	0-255 (IPX only)
l	0-255 (IPX only)

You can access NETADDRESS variable elements by appending a period and the element name to the variable name. For example, 'R1.type' accesses the 'type' element of the R1 variable.

NETADDRESS variables with the REQUIRED scope receive their type element from the get WAN policy call.

Examples

X is a variable of type NETADDRESS.

A TCP/IP address looks like 11.22.33.44 where the numbers are 0 to 255 decimal. X.type is IP, X.a is 11, X.b is 22, X.c is 33, and X.d is 44.

An IPX address looks like 11223344:5566778899AA:BBCC (network, node, socket) where the numbers are in hex. X.type is IPX, X.a is 0x11 (0x indicates hex), X.b is 0x22, X.c is 0x33, X.d is 0x44, X.e is 0x55, X.f is 0x66, X.g is 0x77, X.h is 0x88, X.i is 0x99, X.j is 0xAA, X.k is 0xBB, and X.l is 0xCC.

NETADDRESS DestAddress

Destination address of the server to which the traffic is to be sent. This address can be either TCP-IP or IPX. Not available if NO_ADDRESSES flag is present.

NETADDRESS SrcAddress

Source address of the server that is attempting to send traffic. Not available if NO_ADDRESSES flag is present.

NO_ADDRESSES Statement

A NO_ADDRESSES statement in a declaration indicates that the request is not being sent to a particular address. Rather, the request checks that the process can be initiated. If so, a request can be made to a specific address. For example, an NDS_Janitor request, which contains a NO_ADDRESSES statement, can be sent to check that a Janitor operation can be initiated. If so, a Janitor_Open request can be made to a specific address.

Five of the NDS Traffic Types require NO_ADDRESSES statements. These are NDS_Backlinks, NDS_Janitor, NDS_Limber, NDS_Schema_Sync, and NDS_Check_Login_Restrictions.

The NO_ADDRESSES statement is the first statement in a declaration (it can be preceded by comments). A sample declaration showing a NO_ADDRESSES statement is shown below:

```
/* This policy limits all traffic to between 1 and 3 am */
NO_ADDRESSES
LOCAL BOOLEAN Selected;
SELECTOR
  Selected := Now.hour > 1 AND Now.hour < 3;
  IF Selected THEN
    RETURN 50; /* between 1am and 3am this policy has a high priority */
  ELSE
    RETURN 1;  /* return 1 instead of 0 in case there are no other policies */
               /* if no policies return > 0, WanMan assumes SEND */
  END
END
PROVIDER
  IF Selected THEN
    RETURN SEND; /* between 1am and 3am, SEND */
  ELSE
    RETURN DONT_SEND; /* other times, don't */
  END
END
```

Name

A mandatory, multivalued property that specifies one or more names for the object.

Each value can be up to 64 characters (bytes) long.

NOTE: This field allows only one value, which is the object's official name in the Directory. It should follow standard object naming conventions. After creating the object, you can enter additional values for the Name property in the Other Name field.

In NDS terminology, Name is abbreviated as CN (Common Name) for leaf objects and C (Country), L (Locality), S (State or Province), O (Organization), or OU (Organizational Unit) for container objects.

ONOSPOOF.WMG

The policies in this group allow only existing WAN connections to be used. There are two policies.

- Already Open, No Spoofing, NA

  This policy prevents the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization except on existing WAN connections. Compare Already Open, Spoofing.

- Already Open, No Spoofing

  This policy prevents all other traffic to existing WAN connections. Compare Already Open, Spoofing NA.

To prevent all traffic to existing connections, both policies must be applied.

OPNSPOOF.WMG

The policies in this group allow only existing WAN connections to be used but assumes that a connection that hasn't been used for 15 minutes is being spoofed and should not be used. There are two policies.

- Already Open, Spoofing, NA

  This policy prevents the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema

synchronization except on existing WAN connections that have been open less than 15 minutes. Compare Already Open, No Spoofing.

- Already Open, Spoofing

  This policy prevents other traffic to existing WAN connections that have been open less than 15 minutes. Compare Already Open, No Spoofing, NA.

To prevent all traffic to existing connections open less than 15 minutes, both policies must be applied.

OPTIONAL Scope

Variables defined as OPTIONAL in scope can be used in multiple sections of a policy, but only once within the Declaration section.

OPTIONAL variables are assigned to a default value. These values are not initialized; they are set only if a value is not passed. If a WAN policy request does not pass a new value to the parameter that matches in both name and type, the value defined in the Declaration is used when processing the policy.

You must assign a value to variables defined as OPTIONAL in scope. Therefore, since TIME and NETADDRESS types cannot be initialized in the Declaration section, do not use OPTIONAL scope with these variable types.

Provider Section

The provider section begins with the keyword PROVIDER and concludes with the keyword END. The body of the provider section consists of a list of Declarations.

The result of this declarations list should be a value representing the policys suggestion to SEND or DONT_SEND.

The result of a Provider section is given in a RETURN declaration. If no RETURN declaration is made, a default value of SEND will be returned. The following is a sample provider section:

```
PROVIDER
RETURN SEND;
END
```

For more information on writing declarations, see "Constructions Used within Policy Sections" on page 30.

REQUIRED Scope

Variables defined as REQUIRED in scope can be used in multiple sections, but only once within the Declaration section.

No values may be defined for a REQUIRED variable; its value must come from the GetWanPolicy request.

Replica Synchronization

Replica synchronization ensures that changes to NDS* objects are synchronized among all replicas of the partition.

This means that any server that holds a copy of a given partition must communicate with the other servers to synchronize a change.

Two types of replica synchronization can occur:

- 'Immediate sync' occurs after any change to an NDS object, or any addition or deletion of an object in the Directory tree.
- 'Slow sync' occurs for specific changes to an NDS object that are repetitive and common to multiple objects such as changes to login properties. An example of this would be an update to Login Time, Last Login Time, Network Address, and Revision properties when a user logs in or out.

The slow sync process runs only in the absence of an immediate sync process. By default, the immediate sync process runs ten seconds after any change is saved and the slow sync runs 22 minutes after other changes are made.

SAMEAREA.WMG

The policies in this group allow only traffic in the same network area. A network area is determined by the network section of an address. In a TCP-IP address, Wan Traffic Manager assumes a class C address; addresses whose first three sections are the same are in the same network area. In an IPX address, all addresses with the same network portion are considered to be in the same network area. There are three policies.

- Same Network Area, NA

  This policy prevents the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization

  unless the traffic that would be generated is in the same network area.

- Same Network Area, TCP/IP

  This policy restricts TCP/IP traffic unless that traffic that would be generated is in the same TCP/IP network area.

- Same Network Area, IXP

  This policy restricts IPX traffic unless that traffic that would be generated is in the same IPX network area.

Selector Section

The Selector section of a policy begins with the keyword SELECTOR and concludes with the keyword END. Selector sections are evaluated to determine which loaded policy will be used.

The Selector sections of all the currently-loaded policies are run to determine which policy has the greatest weight. When evaluated, the section returns a weight between 0-100, where 0 means do not use this policy, 1-99 means use this policy if no other policy returns a higher value, and 100 means use this policy.

The result of a Selector section is given in a RETURN declaration. If no RETURN declaration is made, a default value of 0 will be returned. The following is a sample Selector section:

```
SELECTOR
RETURN 49;
END
```

When the Selector sections of multiple policies are evaluated, more than one policy might return the same value. In this case, it is indeterminate which policy will be selected. All else being equal, a server policy will override a WAN policy.

For more information on writing declarations, see "Constructions Used within Policy Sections" on page 30. See also "Provider Section" on page 60.

System Symbols

Symbols that represent predefined variables used by WAN Traffic Manager:

NETADDRESS DestAddress
NETADDRESS SrcAddress
TIME Now
TIME NowUTC
INT DayOfWeek
INT DayOfWeekUTC
INT TrafficType
INT DestCost
INT<NDS_____>

TCP/IP Address

A TCP/IP address looks like 11.22.33.44 where the numbers are 0 to 255 decimal. X.type is IP, X.a is 11, X.b is 22, X.c is 33, and X.d is 44.

X is a variable of type NETADDRESS.

TCPIP.WMG

The policies in this group allow only TCP/IP traffic. There are two policies.

- TCPIP, NA

  This policy prevents the checking of backlinks, external references, and login restrictions, the running of janitor or limber, and schema synchronization unless the traffic that would be generated is TCP/IP.

- TCPIP

  This policy prevents all other traffic unless the traffic is TCP/IP.

To prevent all non-TCP/IP traffic, both policies must be applied.

TIME Variable Type

TIME variables must receive their values in the Selector or Provider sections or from the WAN policy request. Do not assign values to TIME variables in the Declaration.

WAN Traffic Manager uses two predefined variables based on the TIME variable type. These predefined variables are:

- TIME Now
- TIME NowUTC

Time is declared as a structure with the following fields:

Field	Definitions	Valid Values
min	minutes	integer
hour	hours	integer
mday	day of month	1-31
mon	month	1-12 or JANUARY, FEBRUARY, etc.
year	year	Four-digit numeric: 1997, etc.

You can access TIME variables by appending a period and the element name to the variable name. For example, R1.min accesses the minutes element of the R1 variable.

When using mon elements, either the numerical representation or the actual name in uppercase (examples: JANUARY, FEBRUARY) can be used. These are constants stored by the system.

Values for the year element must be entered using four digits.

Values outside the range of a time element do not adjust the time parameter. For example, if R1.min is 65, there is no adjustment of R1.min to 5 nor R1.hour to R1.hour+1.

If a TIME variable is used in a relational expression (=, >=, <=, or <>), normalization will be performed for the comparison, but the actual symbol value is not changed. Required variables and optional variables with values assigned are passed by the client or a call. They are used in a policy and copied back out to the client.

The value will be indeterminate if it is not set in a Declaration or a WAN policy request.

Negative numbers are not allowed for time elements.

TIMECOST.WMG Group

The Policies in this group restrict all traffic to between 1 and 1:30a.m. but allows servers in the same location to talk continuously. This group uses the following policies, all of which must be applied:

- COSTLT20

  The first policy is for COSTLT20 which has a priority of 40 for NA and address traffic.

- Disallow everything

  The second policy allows no traffic to be sent. If WAN Traffic Manager finds no(zero) policies where the selector returned greater than zero, it defaults to SEND. This policy prevents this case.

- NDS Synchronization

  This policy restricts NDS_SYNC traffic to between 1AM and 1:30AM.

- Start rest. Procs, NA

  This policy allows all processes to start at any time, but WAN Traffic Manager must be consulted for each *_OPEN call. It schedules the process to run four times a day 1:00, 7:00, 13:00, and 19:00.

- Start unrest. Procs 1-1:30 NA

  This policy allows all processes to start between 1:00 and 1:30 a.m. and run to completion without further queries to WAN Traffic Manager. The processes run four times a day, every six hours. The 1:00 process will be handled by this policy, the others by the Start rest. Procs, NA.

Traffic Types

The following traffic types have been defined as system parameters.

“NDS_BACKLINK_OPEN” on page 45

“NDS_BACKLINKS” on page 43

“NDS_CHECK_LOGIN_RESTRICTIONS” on page 46

“NDS_CHECK_LOGIN_RESTRICTIONS_OPEN” on page 47

“NDS_JANITOR” on page 48

“NDS_JANITOR_OPEN” on page 50

“NDS_LIMBER_OPEN” on page 53

“NDS_LIMBER” on page 51

“NDS_SCHEMA_SYNC_OPEN” on page 55

“NDS_SCHEMA_SYNC” on page 54

“NDS_SYNC” on page 56

WAN Traffic Manager also provides system symbols (predefined variables) for use with all traffic types.

Variable Name

A combination of alphanumeric characters in a string of any length. Since only the first 31 characters are used, a variable must begin with a unique 31-character string. A variable name must start with an alphabetic character, or the symbol is interpreted as a numeric constant.

Variable names are case sensitive: variable R1 is not the same as variable r1. The underscore character (_) is allowed in variable names.

Variables

WAN Traffic Manager uses several variables in policy statements. They are:

- BOOLEAN
- NETADDRESS
- TIME
- INT

WAN Policy Structure

The WAN policy consists of three sections:

“Declaration Section” on page 37

“Selector Section” on page 62

“Provider Section” on page 60

7 Glossary

Add

Lets you specify a new cost factor for a network address or range of addresses. You will need to specify an address type of either TCP/IP or IPX/SPX and then provide the actual address or provide the start and end addresses for a range of addresses.

Advanced

Displays the Policies Dialog Box. From there you can create, edit, rename, and delete WAN policies.

Backlink

A process that verifies external references, which are pointers to NDS objects that are not stored in the replicas on a server. The backlink process normally runs two hours after the local database is opened and then every thirteen hours.

BOOLEAN Variable Type

Used for values of only TRUE or FALSE. The value will be indeterminate if it is not set in a declaration or a WAN policy request.

Browser Button

Allows you to browse the current tree and select a LAN Area object to add the NetWare Server to. You cannot add the server to more than one LAN Area object.

Clear

Removes the server from the LAN Area object displayed in the text box.

Default Cost

The cost entered here will be used by WAN policies when no specific cost factor has been assigned above.

Delete

This button is active only if a line is selected in the Cost box. Choose Delete to remove the selected cost from the list, then OK to make that deletion effective, or choose Cancel to discard that deletion.

Department

An optional, multivalued property that specifies one or more departments or divisions for the object. For example: Marketing.

Each value can be up to 64 characters (bytes) long.

In NDS* terminology, Department is called OU (Organizational Unit).

Description

An optional, single-valued property that provides a description for the object. The description can be up to 1,024 characters (bytes) long.

INT<NDS_____>

Any of the constants (for example, NDS_SYNC) used to specify the traffic types that NDS is using as a client. Thus the policy can contain something like:

IF TrafficType=NDS_SYNC THEN <action> END.

LAN Area Object

An NDS object that a specified collection of servers belongs to. All servers belonging to the same LAN Area object are governed by any WAN policy applied to that object.

A LAN Area object is a good way to manage traffic generated by a group of servers without applying a WAN policy individually to each server.

Locality

Specifies optional, additional values for the Name property.

Each value can be up to 128 characters (bytes) long.

Organization (O)

An optional, multivalued property that specifies one or more organizations for the object. For example: ACME Corporation.

Each value can be up to 64 characters (bytes) long.

Owner

The User or other object that created and administers the LAN Area object.

Policy Load Results

Displays the results of loading the WAN policy group.

Schema Synchronization

Schema synchronization ensures that the schema is consistent across the partitions in the Directory tree, and that all schema changes are updated across the network.

This process runs once every 4 hours by default.

Server Status Check

Each server without a replica initiates a server status check. It establishes a connection to the nearest server that holds a writable replica of the partition containing the NetWare Server object.

The server status check runs every six minutes.

Start

Network and node numbers that begin the address range.

Stop

Network and node numbers that end the address range.

TIME Now

Local time that the current policy processing began.

TIME NowUTC

UTC time that the current policy processing began.

A Novell Trademarks

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

Advanced NetWare is a trademark of Novell, Inc.

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

AppNotes is a registered service mark of Novell, Inc. in the United States and other countries.

AppNotes is a registered service mark of Novell, Inc. in the United States and other countries.

AppTester is a registered service mark of Novell, Inc. in the United States and other countries.

BrainShare is a registered service mark of Novell, Inc. in the United States and other countries.

C-Worthy is a trademark of Novell, Inc.

C3PO is a trademark of Novell, Inc.

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

Certified NetWare Administrator in Japanese and CNA-J are service marks of Novell, Inc.

Certified NetWare Engineer in Japanese and CNE-J are service marks of Novell, Inc.

Certified NetWare Instructor in Japanese and CNI-J are service marks of Novell, Inc.

Certified Novell Administrator and CNA are service marks of Novell, Inc.

Certified Novell Engineer is a trademark and CNE is a registered service mark of Novell, Inc. in the United States and other countries.

Certified Novell Salesperson is a trademark of Novell, Inc.

Client 32 is a trademark of Novell, Inc.

LAN WorkGroup is a trademark of Novell, Inc.

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

LAN WorkShop is a trademark of Novell, Inc.

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

LANalyzer Agent is a trademark of Novell, Inc.

Link Support Layer and LSL are trademarks of Novell, Inc.

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

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

Media Support Module and MSM are trademarks of Novell, Inc.

Mirrored Server Link and MSL are trademarks of Novell, Inc.

Mobile IPX is a trademark of Novell, Inc.

Multiple Link Interface and MLI are trademarks of Novell, Inc.

Multiple Link Interface Driver and MLID are trademarks of Novell, Inc.

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

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

Natural Language Interface for Help is a trademark of Novell, Inc.

NDS Manager is a trademark of Novell, Inc.

NE/2 is a trademark of Novell, Inc.

NE/2-32 is a trademark of Novell, Inc.

NE/2T is a trademark of Novell, Inc.

NE1000 is a trademark of Novell, Inc.

NE1500T is a trademark of Novell, Inc.

NE2000 is a trademark of Novell, Inc.

NE2000T is a trademark of Novell, Inc.

NE2100 is a trademark of Novell, Inc.

NE3200 is a trademark of Novell, Inc.

NE32HUB is a trademark of Novell, Inc.

NEST Autoroute is a trademark of Novell, Inc.

NetExplorer is a trademark of Novell, Inc.

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

NetWare Link/PPP is a trademark of Novell, Inc.

NetWare Link/X.25 is a trademark of Novell, Inc.

NetWare Loadable Module and NLM are trademarks of Novell, Inc.

NetWare LU6.2 is trademark of Novell, Inc.

NetWare Management Agent is a trademark of Novell, Inc.

NetWare Management System and NMS are trademarks of Novell, Inc.

NetWare Message Handling Service and NetWare MHS are trademarks of Novell, Inc.

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

NetWare Mirrored Server Link and NMSL are trademarks of Novell, Inc.

NetWare Mobile is a trademark of Novell, Inc.

NetWare Mobile IPX is a trademark of Novell, Inc.

NetWare MultiProtocol Router and NetWare MPR are trademarks of Novell, Inc.

NetWare MultiProtocol Router Plus is a trademark of Novell, Inc.

NetWare Name Service is trademark of Novell, Inc.

NetWare Navigator is a trademark of Novell, Inc.

NetWare Peripheral Architecture is a trademark of Novell, Inc.

NetWare Print Server is a trademark of Novell, Inc.

NetWare Ready is a trademark of Novell, Inc.

NetWare Requester is a trademark of Novell, Inc.

NetWare Runtime is a trademark of Novell, Inc.

NetWare RX-Net is a trademark of Novell, Inc.

NetWare SFT is a trademark of Novell, Inc.

NetWare SFT III is a trademark of Novell, Inc.

NetWare SNA Gateway is a trademark of Novell, Inc.

NetWare SNA Links is a trademark of Novell, Inc.

NetWare SQL is a trademark of Novell, Inc.

NetWare Storage Management Services and NetWare SMS are trademarks of Novell, Inc.

NetWare Telephony Services is a trademark of Novell, Inc.

NetWare Tools is a trademark of Novell, Inc.

NetWare UAM is a trademark of Novell, Inc.

NetWare WAN Links is a trademark of Novell, Inc.

NetWare/IP is a trademark of Novell, Inc.

Novell Gold Partner is a service mark of Novell, Inc.

Novell Labs is a trademark of Novell, Inc.

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

Novell NE/2 is a trademark of Novell, Inc.

Novell NE/2-32 is a trademark of Novell, Inc.

Novell NE3200 is a trademark of Novell, Inc.

Novell Network Registry is a service mark of Novell, Inc.

Novell Platinum Partner is a service mark of Novell, Inc.

Novell Press is a trademark of Novell, Inc.

Novell Press Logo (teeth logo) is a registered trademark of Novell, Inc. in the United States and other countries.

Novell Replication Services is a trademark of Novell, Inc.

Novell Research Reports is a trademark of Novell, Inc.

Novell RX-Net/2 is a trademark of Novell, Inc.

Novell Service Partner is a trademark of Novell, Inc.

Novell Storage Services is a trademark of Novell, Inc.

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

Novell Technical Services and NTS are service marks of Novell, Inc.

Novell Technology Institute and NTI are registered service marks of Novell, Inc. in the United States and other countries.

Novell Virtual Terminal and NVT are trademarks of Novell, Inc.

Novell Web Server is a trademark of Novell, Inc.

Novell World Wide is a trademark of Novell, Inc.

NSE Online is a service mark of Novell, Inc.

NTR2000 is a trademark of Novell, Inc.

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

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

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

Open Data-Link Interface and ODI are trademarks of Novell, Inc.

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

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

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

Storage Management Services and SMS are trademarks of Novell, Inc.

Technical Support Alliance and TSA are collective marks of Novell, Inc.

The Fastest Way to Find the Right Word is a registered trademark of Novell, Inc. in the United States and other countries.

The Novell Network Symbol is a trademark of Novell, Inc.

Topology Specific Module and TSM are trademarks of Novell, Inc.

Transaction Tracking System and TTS are trademarks of Novell, Inc.

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

Virtual Loadable Module and VLM are trademarks of Novell, Inc.

Writer's Workbench is a registered trademark of Novell, Inc. in the United States and other countries.

Yes, It Runs with NetWare (logo) is a trademark of Novell, Inc.

Yes, NetWare Tested and Approved (logo) is a trademark of Novell, Inc.

ZENworks is a trademark of Novell, Inc.