Novell Developer Kit

www.novell.com

NETWORK MANAGEMENT

October 5, 2005

Novell®

Legal Notices

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

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

Novell Trademarks

AppNotes is a registered trademark of Novell, Inc.
AppTester is a registered trademark of Novell, Inc., in the United States.
ASM is a trademark of Novell, Inc.
BorderManager is a registered trademark of Novell, Inc.
BrainShare is a registered service mark of Novell, Inc., in the United States and other countries.
C3PO is a trademark of Novell, Inc.
Certified Novell Engineer is a service mark of Novell, Inc.
Client32 is a trademark of Novell, Inc.
CNE is a registered service mark of Novell, Inc.
ConsoleOne is a registered trademark of Novell, Inc.
Controlled Access Printer is a trademark of Novell, Inc.
Custom 3rd-Party Object is a trademark of Novell, Inc.
DeveloperNet is a registered trademark of Novell, Inc., in the United States and other countries.
DirXML is a registered trademark of Novell, Inc.
eDirectory is a trademark of Novell, Inc.
Excelerator is a trademark of Novell, Inc.
exteNd is a trademark of Novell, Inc.
exteNd Director is a trademark of Novell, Inc.
exteNd Workbench is a trademark of Novell, Inc.
FAN-OUT FAILOVER is a trademark of Novell, Inc.
GroupWise is a registered trademark of Novell, Inc., in the United States and other countries.
Hardware Specific Module is a trademark of Novell, Inc.
Hot Fix is a trademark of Novell, Inc.
iChain is a registered trademark of Novell, Inc.
Internetwork Packet Exchange is a trademark of Novell, Inc.
IPX is a trademark of Novell, Inc.
IPX/SPX is a trademark of Novell, Inc.
jBroker is a trademark of Novell, Inc.
Link Support Layer is a trademark of Novell, Inc.
LSL is a trademark of Novell, Inc.
ManageWise is a registered trademark of Novell, Inc., in the United States and other countries.
Mirrored Server Link is a trademark of Novell, Inc.
Mono is a registered trademark of Novell, Inc.
MSL is a trademark of Novell, Inc.
My World is a registered trademark of Novell, Inc., in the United States.
NCP is a trademark of Novell, Inc.
NDPS is a registered trademark of Novell, Inc.
NDS is a registered trademark of Novell, Inc., in the United States and other countries.
NDS Manager is a trademark of Novell, Inc.
NE2000 is a trademark of Novell, Inc.
NetMail is a registered trademark of Novell, Inc.
NetWare is a registered trademark of Novell, Inc., in the United States and other countries.
NetWare/IP is a trademark of Novell, Inc.
NetWare Core Protocol is a trademark of Novell, Inc.
NetWare Loadable Module is a trademark of Novell, Inc.

NetWare Management Portal is a trademark of Novell, Inc.

NetWare Name Service is a trademark of Novell, Inc.

NetWare Peripheral Architecture is a trademark of Novell, Inc.

NetWare Requester is a trademark of Novell, Inc.

NetWare SFT and NetWare SFT III are trademarks of Novell, Inc.

NetWare SQL is a trademark of Novell, Inc.

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

NLM is a trademark of Novell, Inc.

NMAS is a trademark of Novell, Inc.

NMS is a trademark of Novell, Inc.

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

Novell Application Launcher is a trademark of Novell, Inc.

Novell Authorized Service Center is a service mark of Novell, Inc.

Novell Certificate Server is a trademark of Novell, Inc.

Novell Client is a trademark of Novell, Inc.

Novell Cluster Services is a trademark of Novell, Inc.

Novell Directory Services is a registered trademark of Novell, Inc.

Novell Distributed Print Services is a trademark of Novell, Inc.

Novell iFolder is a registered trademark of Novell, Inc.

Novell Labs is a trademark of Novell, Inc.

Novell SecretStore is a registered trademark of Novell, Inc.

Novell Security Attributes is a trademark of Novell, Inc.

Novell Storage Services is a trademark of Novell, Inc.

Novell, Yes, Tested & Approved logo is a trademark of Novell, Inc.

Nsure is a registered trademark of Novell, Inc.

Nterprise is a trademark of Novell, Inc.

Nterprise Branch Office is a trademark of Novell, Inc.

ODI is a trademark of Novell, Inc.

Open Data-Link Interface is a trademark of Novell, Inc.

Packet Burst is a trademark of Novell, Inc.

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

Printer Agent is a trademark of Novell, Inc.

QuickFinder is a trademark of Novell, Inc.

Red Box is a trademark of Novell, Inc.

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

Sequenced Packet Exchange is a trademark of Novell, Inc.

SFT and SFT III are trademarks of Novell, Inc.

SPX is a trademark of Novell, Inc.

Storage Management Services is a trademark of Novell, Inc.

SUSE is a registered trademark of SUSE AG, a Novell business.

System V is a trademark of Novell, Inc.

Topology Specific Module is a trademark of Novell, Inc.

Transaction Tracking System is a trademark of Novell, Inc.

TSM is a trademark of Novell, Inc.

TTS is a trademark of Novell, Inc.

Universal Component System is a registered trademark of Novell, Inc.

Virtual Loadable Module is a trademark of Novell, Inc.

VLM is a trademark of Novell, Inc.

Yes Certified is a trademark of Novell, Inc.

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

Third-Party Materials

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

Contents

About This Guide

This documentation provides information about Accounting, Auditing, and Name Retrieval.

Accounting is a bindery-based service that allows you to track print server and database usage, connection time, and disk storage. Auditing is a directory-based service that allows NetWare 4.x, 5.x, and 6.x servers to generate files that track and record network events. Name Retrieval services allows tree or server names to be successfully retrieved over IP or IPX. This guide describes the following interfaces:

- Chapter 2, “Accounting Functions,” on page 25
- Chapter 6, “Auditing Functions,” on page 61
- Chapter 8, “Name Retrieval Functions,” on page 147
- Chapter 10, “Server-Based Auditing Functions,” on page 157
- Chapter 12, “Bindery-Based Accounting Functions,” on page 169

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation.

Documentation Updates

For the most recent version of this guide, see NLM and NetWare Libraries for C (including CLIB and XPlat) (http://developer.novell.com/ndk/clib.htm)

Additional Information

For information about other CLib and XPlat interfaces, see the following guides:

- NLM Development Concepts, Tools, and Functions (http://developer.novell.com/ndk/doc/clib/ndev_enu/data/hqgkbkjr.html)
- Single and Intra-File Services (http://developer.novell.com/ndk/doc/clib/sngl_enu/data/h68b1qom.html)
- Multiple and Inter-File Services (http://developer.novell.com/ndk/doc/clib/mlti_enu/data/hby40vgi.html)
- NLM Threads Management (http://developer.novell.com/ndk/doc/clib/thmp_enu/data/h7g6q8vc.html)
- Connection, Message, and NCP Extensions (http://developer.novell.com/ndk/doc/clib/cmgnxenu/data/hvxfva0i.html)
- Internationalization (http://developer.novell.com/ndk/doc/clib/intl_enu/data/h70a28iu.html)
- Volume Management (http://developer.novell.com/ndk/doc/clib/vol__enu/data/h6ixme3w.html)
- Client Management (http://developer.novell.com/ndk/doc/clib/clnt_enu/data/he77rked.html)

- Program Management (http://developer.novell.com/ndk/doc/clib/prog_enu/data/h9qu926c.html)
- Server Management (http://developer.novell.com/ndk/doc/clib/srvr_enu/data/hzvjztxz.html)
- Unicode (http://developer.novell.com/ndk/doc/clib/ucod_enu/data/hjg275fp.html)
- Sample Code (http://developer.novell.com/ndk/doc/clib/code_enu/data/hwtnc7wc.html)
- Getting Started with NetWare Cross-Platform Libraries for C (http://developer.novell.com/ndk/doc/clib/startenu/data/hv9aw5v8.html)
- Bindery Management (http://developer.novell.com/ndk/doc/clib/bind_enu/data/h9qzn7u5.html)

For CLib source code projects, visit Forge (http://forge.novell.com).

For help with CLib and XPlat problems or questions, visit the NLM and NetWare Libraries for C (including CLIB and XPlat) Developer Support Forums (http://developer.novell.com/ndk/devforums.htm). There are two for NLM development (XPlat and CLib) and one for Windows XPlat development.

Documentation Conventions

In this documentation, a greater-than symbol (>) is used to separate actions within a step and items within a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

Accounting Concepts

1

This documentation describes Accounting, its functions, and features.

Developers can also take advantage of accounting to charge for services and to audit events. Because Accounting relies heavily on the bindery, a familiarity with bindery operations is necessary before reading about Accounting (see *NDK: Bindery Management*).

1.1 Elements of Accounting

The following elements comprise NetWare® accounting:

1.1.1 Servers of Accounting

Any object permitted to charge for services through the accounting service qualifies as an accounting server. A server receives accounting status by recording its bindery object ID in the ACCOUNT_SERVERS property of the NetWare server. Your application needs supervisor equivalence to add itself to this property. (For more information about adding an application to the ACCOUNTING_SERVERS property, see NWAddObjectToSet (*NDK: Bindery Management*).)

The accounting system defines a special bindery property to manage each element.

1.1.2 Balances of Accounting

When accounting is enabled on a NetWare® server, users are given a global account balance. Expenditures for all accounting-enabled services are charged against this value. The account balance is stored in the user's ACCOUNT_BALANCE property. The following table shows fields within this property.

Offset	Field	Type	Order
0	balance	nint32	high-low
4	credit limit	nint32	high-low
8	reserved	nuint8[120]	

The `balance` field contains a signed long integer indicating the current balance.

The `credit limit` field contains a signed long integer indicating the lowest permissible balance. Requests are denied for services causing the value in `balance` to fall below the value in `credit limit`. `credit limit` can be positive (maintaining a minimum balance) or negative (permitting the balance to fall below 0).

1.1.3 Events of Accounting

Accounting includes a rudimentary auditing service that allows you to record accounting events as they occur. (Accounting audits are separate from the full-scale services provided by Auditing.) When auditing is enabled on a server an accounting audit file, net$acct.dat, is created and put in the

server's sys:system directory. net$acct.dat stores the auditing record. Standard file I/O functions can manipulate this file, which has normal file attributes. The NetWare® PAUDIT utility reads the information from this file to the standard output.

The accounting audit file contains records of two types: charge records and note records. A charge record is added to the file when a user's account balance is charged for services. Note records are added at the discretion of the services using accounting. A note is purely informational and has no effect on any account balances.

The net$acct.dat file grows in size as accounting events occur. This file should be monitored regularly and archived and deleted when it gets too big. Once the net$acct.dat file has been archived and deleted, NetWare will automatically recreate it the next time the user logs in.

Charge Record Auditing

A charge record is created each time a charge is placed against a user balance. The following table shows the format of a charge record:

Offset	Field	Type	Order
0	length	nuint16	high-low
2	server ID	nuint32	high-low
6	time stamp	nuint8[6]	
12	record type	nuint8[1]	
13	completion code	nuint8	
14	service type	nuint16	high-low
16	client ID	nuint32	high-low
20	amount	nuint32	high-low
24	comment type	nuint16	high-low
26	comment	variable	

You can identify the record type by checking the `record type` field. Identify the record type as follows:

charge record: Record type = 1
note record: Record type = 2

The `length` field contains the total length of the remaining fields in the record (that is, record length minus 2 bytes).

The `time stamp` field contains the date and time the charge was submitted. The format is year (year=current year minus 1900), month, day, hour, minute, second, where each is a byte-sized integer.

The `server ID` and `service type` fields identify the object submitting the charge and the type of service. The server ID is a bindery object ID. The service type is the bindery object type of the server. You must provide your service type when you submit the charge. The amount is the amount being charged to the user.

The `client ID` field identifies the user being charged for the service. Although this value identifies the user by bindery object ID, you need to know only the name of the user to submit the charge request. For more information about comment type and the format of the comment field, see Section 1.2, “Event Comments of Accounting,” on page 16.

The `amount` field is a long unsigned integer indicating the amount of the charge.

The `comment` and `comment type` fields supply information specific to the service. Novell defines standard comment types and formats for recording this information. The length of the `comment` field can be determined by the type of the comment. The `completion code` field is set to 0 if the request is successful.

Note Record Auditing

The note record shares many of the same fields found in the change record. The main difference between them is that the note record does not include a charge amount. Notes are used to record events for the system administrator, such as the time a user logged in to the server. The following table shows the format of a note record:

Offset	Field	Type	Order
0	length	nuint16	high-low
2	server ID	nuint32	high-low
6	time stamp	nuint8[6]	
12	record type	nuint8[1]	
13	reserved	nuint8	
14	service type	nuint16	high-low
16	client ID	nuint32	high-low
20	comment type	nuint16	high-low
22	comment	variable	

1.1.4 Holds of Accounting

Account holds on user balances reserve funds for services the server supplies. Holds allow verification that current funds can cover particular requests. Account holds are not required, but your application should at least check `balance` in the ACCOUNT_BALANCE property to verify funding for the specific request.

Holds are recorded in the user property, ACCOUNT_HOLDS. This property is present only if there is a hold on the account. The following table shows the format of the ACCOUNT_HOLDS property.

Offset	Field	Type	Order
0	object ID	nint32	high-low
4	amount	nuint32	high-low

Each hold consists of an `object ID` and an `amount` field. The `object ID` field is the ID of the object placing the hold. The `amount` field is the amount of the hold. Each of the two parameters uses 4 bytes. The maximum size of the property is 128 bytes; therefore, ACCOUNT_HOLDS can have no more than 16 holds placed on the account at once. If the object ID is 0, the `amount` field is not meaningful. HOLDS_STATUS (page 43) and HOLDS_INFO (page 42) are defined to handle this data.

1.2 Event Comments of Accounting

A comment is a field in a charge or note record that stores information specific to the service submitting the charge or note. For example, if you are charging for disk storage, use a disk storage comment. There are six predefined comment types. If one of these doesn't serve your purpose, contact Novell's Developer Relations (see the preface for phone numbers) to request a unique comment type. Standard comment types and their values include the following:

1 = Connect time charge
2 = Disk storage charge
3 = Log in note
4 = Log out note
5 = Account locked note
6 = Server time modified note

Comments are embedded in charge or note records and supplement the standard information that appears there. For example, a client ID of the record identifies the user associated with the comment.

1.2.1 Connect Time Comment

A connect time comment records NetWare® server usage. It includes the number of minutes the workstation was connected to the server, the number of packets sent to the server, and the number of bytes written and read. The following table shows the format of a connect time comment.

Offset	Field	Type	Order
0	connect time	nuint32	high-low
4	request count	nuint32	high-low
8	bytes read (hi)	nuint16	high-low
10	bytes read (middle)	nuint16	high-low
12	bytes read (lo)	nuint16	high-low
14	bytes written (hi)	nuint16	high-low
16	bytes written (middle)	nuint16	high-low
18	bytes written (lo)	nuint16	high-low

1.2.2 Disk Storage Comment

A disk storage comment records the number of blocks owned by the user and the amount of time the blocks have been in the user's possession. The time is expressed in half hour increments. The following table shows the format of a disk storage comment:

Offset	Field	Type
0	number of blocks owned	nuint32
4	number of half hours	nuint32

1.2.3 Login Comment

A login comment records the network node of a workstation that the user logged in from. The NetWare® server uses this comment as an auditing note and not as a charge. The following table shows the format of a login comment:

Offset	Field	Type	Order
0	Net	nuint32	high-low
4	Node (hi)	nuint32	high-low
8	Node (lo)	nuint16	high-low

1.2.4 Logout Comment

A logout comment records the network node of a workstation the user has logged out from. The NetWare® server uses this comment as an auditing note and not as a charge. The following table shows the format of a logout comment:

Offset	Field	Type	Order
0	Net	nuint32	high-low
4	Node (hi)	nuint32	high-low
8	Node (lo)	nuint16	high-low

1.2.5 Account Locked Comment

An `account locked` comment records the network node on which a user's account has been locked because of too many failed login attempts. The following table shows the format of an account locked comment.

Offset	Field	Type	Order
0	Net	nuint32	high-low
4	Node (hi)	nuint32	high-low
8	Node (lo)	nuint16	high-low

1.2.6 Server Time Modified Comment

A server time modified comment records a modification in the system time of the NetWare server. The comment contains the time value before the change was made. The following table shows the format of a server time modified comment:

Offset	Field	Type
0	year since 1900	nuint8
1	month	nuint8
2	day	nuint8
3	hour	nuint8
4	minute	nuint8
5	second	nuint8

1.3 Event Comments of Accounting Format File

A comment format file, net$rec.dat, found in the sys:system directory, serves as a companion to the audit file. This file is for information and ease-of-use only and cannot be written to. (Do not delete this file, NetWare® does not recreate it as it does the net$acct.dat file.) The file allows you to display information from the `comment` field in a charge or note record. For example, a disk storage comment indicates the number of disk blocks owned by a user over a period of half hours. If the user owns 25 blocks for 10 half hours, the `comment` field stores only the numeric values 25 and 10. A record in the net$rec.dat file stores a standard format control string for displaying this data. In this example, the format string is

```
"%lu disk blocks stored for %lu half-hours."
```

This string is intended to fit conveniently into a `printf` -style statement, as in the following code:

```
#include<nwmisc.h>
nuint32   blocks, time;

printf("%lu disk blocks stored for %lu half-hours.",
NWLongSwap(blocks),                NWLongSwap(time);
```

In your code, the format control string would not appear as a literal value but as a pointer to the area into which you copy the formatting string from the comment formatting file. Since values in the comment are stored in high-low order, they are swapped in the example above.

1.3.1 String Formatting Records

The following table shows the format of a record in the net$rec.dat file.:

Offset	Field	Type	Order
0	length	nuint16	high-low
2	comment type	nuint16	high-low

Offset	Field	Type	Order
4	field count	nuint8[1]	
5	data type	nuint8[field count]	
?	buffer	variable	

Use the "comment type" field to associate a formatting record in this file with a comment in the audit file. For example, the formatting record for disk storage comments would have a value of 2 in comment type.

`length` is the total length of the remaining fields in the record (i.e., record length - 2 bytes).

`data type` identifies the data types found sequentially in the associated comment. The following values are defined:

1 = nuint8
2 = nuint16
3 = nuint32
4 = TEXT

Text data is a length-preceded string of printable characters.

The number of data types is stored in the field count field.

Below are two examples of how the `buffer` field is formatted.

The following table is an example of a Connect Time Charge with a Field Count of four.

Offset	Field	Value	Type
0	data type 1	3	nuint8
1	data type 2	3	nuint8
2	data type 3	5	nuint8
3	data type 4	5	nuint8
4	data value 1		nuint32
8	data value 2		nuint32
12	data value 3		nfloat48
18	data value 4		nfloat48
24	length of string	81	nuint8
25	printable string		ASCII

The following table is an example of a Disk Storage Charge with a Field Count of two.

Offset	Field	Value	Type
	data type 1	3	nuint8

Offset	Field	Value	Type
1	data type 2	3	nuint8
2	data value 1		nuint32
6	data value 2		nuint32
10	length of string	55	nuint8
11	printable string		ASCII

1.3.2 Standard Format Control Strings

The following is a list of the format control strings for standard Novell comment types:

Connect Time Format:

```
"Connected %lu minutes: %lu requests; %04x%04x%04xh
bytes read; %04x%04x%04xh bytes written"
```

Disk Storage Format:

```
"%lu disk blocks stored for %lu half-hours."
```

Login Format:

```
"Login from address %lx:%lx%x."
```

Logout Format:

```
"Logout from address %lx:%lx%x."
```

Account Locked Format:

```
"Account intruder lockout caused by address:
%lx:%lx%x."
```

Server Time Modified Format:

```
"System time changed to 19%02d-%02d-%02d%d:%02:%02d."
```

1.4 NetWare Server Charges

NetWare® servers use the accounting system to charge for services and disk storage. The supervisor determines the charge rates, typically by using the SYSCON utility (NetWare 3.x), NETADMIN (NetWare 4.x), and NWADMIN utilities (NetWare 4.x, 5.x, and 6.x). Rates are stored as bindery properties attached to the server object. NetWare servers handle disk storage accounting somewhat differently from other services, so service and storage are considered separately in this section.

1.4.1 Chargeable Services

Services are charged for every half hour from the time the user logs in until the user logs out. If during this period the user account balance is found to have fallen below the credit limit, the user is given a five minute and a one minute warning, and is then disconnected. Chargeable services

include connection time, number of requests, number of blocks read, and number of blocks written. The rates for these services are stored in the following properties:

- CONNECT_TIME
- REQUESTS_MADE
- BLOCKS_READ
- BLOCKS_WRITTEN

Although there is a separate property for each service, the properties share the same format.

The following table shows the format of service rate properties:

Offset	Field	Type	Order
0	time of next charge	nuint32	high-low
4	current charge rate multiplier	nuint16	high-low
6	current charge rate divisor	nuint16	high-low
8	days charge occurs mask	nuint8	
9	time charge occurs	nuint8	
10	charge rate multiplier	nuint16	high-low
12	charge rate divisor	nuint16	high-low

NOTE: Service rate properties can store up to 20 charge rates. Each charge rate includes the following four fields (6 bytes):

```
days charge occurs mask
time charge occurs
charge rate multiplier
charge rate divisor
```

The first charge rate begins at offset 8, the second charge rate begins at offset 14 and the third charge rate begins at offset 20. The following table shows the information for the second and third charge rates, if used. Charge rates 4 through 20 would follow this pattern if used.

Offset	Field	Type	Order
14	days charge occurs mask	nuint8	
15	time charge occurs	nuint8	
16	charge rate multiplier	nuint16	high-low
18	charge rate divisor	nuint16	high-low

Offset	Field	Type	Order
20	days charge occurs mask	nuint8	
21	time charge occurs	nuint8	
22	charge rate multiplier	nuint16	high-low
24	charge rate divisor	nuint16	high-low

The first three fields in the service rate property maintain the *current* charge rate. The time of next charge field contains the time when the current charge is replaced with a different rate. The current charge rate multiplier and current charge rate divisor are used to compute the charges in the following manner:

1. The unit of service (total connect time, number of requests performed, number of blocks read, number of blocks written) is multiplied by the current charge rate multiplier.
2. The product from Step 1 is divided by the current charge rate divisor, and the result is deducted from the user's account balance.

Each group of four fields after the first three fields contain a charge rate. The days charge occurs mask and time charge occurs together indicate when the charge should take effect. The days charge occurs mask indicates the day of the week (bit 0=Sunday,..., bit 6=Saturday) and time charge occurs indicates the time of day on the half hour (0=12am,...,47=11:30pm).

Each rate has its own charge rate multiplier and divisor. The server moves these values into current charge rate multiplier and current charge rate divisor when the charge rate becomes effective.

1.4.2 Disk Storage Charges

Disk storage charges are levied every half hour. The disk storage charge rates are stored in the DISK_STORAGE property attached to the NetWare® server. The format for this property varies slightly from the service properties due to the static nature of storage. Charges are computed for all users during the same period (independent of user login sessions). The following table shows this format:

Offset	Field	Type	Order
0	time of next charge	nuint32	high-low
4	current charge rate multiplier	nuint16	high-low
6	current charge rate divisor	nuint16	high-low
8	days charge occurs mask	nuint8	
9	time charge occurs	nuint8	
10	charge rate multiplier	nuint16	high-low
12	charge rate divisor	nuint16	high-low

Like service rate properties, the disk storage property can store up to 20 charge rates. The first charge rate begins at offset 8 and includes 4 fields (6 bytes). The purpose of each field is exactly the same as for a service rate property (See “Chargeable Services” on page 20). The current rate for disk storage is computed in the following manner for each user:

1. The current disk storage is calculated in 4K blocks.
2. The total blocks occupied by a user are multiplied by the number of half hours since the last disk storage charge was made.
3. The total half-hour blocks are multiplied by the current charge rate multiplier.
4. The product from Step 3 is divided by the current charge rate divisor, and the result is deducted from the user’s account balance.

The time of next charge and time of previous charge are used to calculate when the next charge rate takes effect.

Accounting Functions

2

This documentation alphabetically lists the accounting functions and describes their purpose, syntax, parameters, and return values.

NWGetAccountStatus

Returns the account status of a NetWare® server bindery or NDS™ object including its balance, credit limit, and holds

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Accounting

Syntax

```
#include <nwacct.h>
or
#include <nwcalls.h>

NWCCODE N_API NWGetAccountStatus (
   NWCONN_HANDLE         conn,
   nuint16               objType,
   const nstr8 N_FAR    *objName,
   pnint32               balance,
   pnint32               limit,
   HOLDS_STATUS N_FAR   *holds);
```

Pascal Syntax

```
uses calwin32

Function NWGetAccountStatus
  (conn : NWCONN_HANDLE;
   objType : nuint16;
   const objName : pnstr8;
   balance : pnint32;
   limit : pnint32;
   Var holds : HOLDS_STATUS
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

objType

(IN) Specifies the type of the object for which the status is desired.

objName

(IN) Points to a string containing the name of the object for which the accounting status is desired. (48 characters maximum or will be truncated.)

balance

(OUT) Points to the amount of value units available to the object to buy services on the network (optional, pass in NULL).

limit

(OUT) Points to the lowest level the object's account balance can reach before the object can no longer buy services on the network (optional, pass in NULL).

holds

(OUT) Points to HOLDS_STATUS containing a list of objects placing a hold on the object's account (optional, pass in NULL).

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8996	SERVER_OUT_OF_MEMORY
0x89C0	NO_ACCOUNTING_PRIVILEGES
0x89C1	LOGIN_DENIED_NO_ACCOUNT_BALANCE
0x89C4	ACCOUNTING_DISABLED
0x89E8	NOT_ITEM_PROPERTY
0x89EA	NO_SUCH_MEMBER
0x89EB	NOT_GROUP_PROPERTY
0x89EC	NO_SUCH_SEGMENT
0x89EF	INVALID_NAME
0x89F0	WILD_CARD_NOT_ALLOWED
0x89FC	NO_SUCH_OBJECT
0x89FE	BINDERY_LOCKED
0x89FF	HARDWARE_FAILURE

Remarks

NWGetAccountStatus queries a NetWare server's Bindery or Directory bindery context for the current account status of a specified object by passing the object name and type. NWGetAccountStatus returns the object's balance, limit and holds.

`balance` contains the object's account balance, usually in some established monetary unit such as cents.

`holds` lists servers calling NWSubmitAccountHold against the object and the amount reserved by each value-added server. `holds` also lists the object ID number of a value-added server calling NWSubmitAccountHold against the object. Up to 16 servers can place holds on the account at one time. Multiple holds from the same server are combined. Each server hold is made up of two fields: (1) the object ID of the server placing the hold, and (2) the amount of the server's hold.

The ACCOUNT_SERVERS property on the server must contain the object ID of the requesting server. Otherwise, NWSubmitAccountHold will return NO_ACCOUNTING_PRIVILEGES. The user must have an ACCOUNT_BALANCE property on the NetWare server or NWSubmitAccountHold will return LOGIN_DENIED_NO_ACCOUNT_BALANCE.

NCP Calls

0x2222 23 150 Get Current Account Status

See Also

NWQueryAccountingInstalled (page 29), NWSubmitAccountCharge (page 31), NWSubmitAccountHold (page 34), NWSubmitAccountNote (page 37)

NWQueryAccountingInstalled

Determines whether accounting is installed and/or enabled on a NetWare server

Local Servers: blocking

Remote Servers: blocking

NetWAre Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Accounting

Syntax

```
#include <nwacct.h>
or
#include <nwcalls.h>

NWCCODE N_API NWQueryAccountingInstalled  (
   NWCONN_HANDLE   conn,
   pnuint8         installed);
```

Pascal Syntax

```
uses calwin32

Function NWQueryAccountingInstalled
   (conn : NWCONN_HANDLE;
   installed : pnuint8
) : NWCCODE;
```

Parameters

`conn`

(IN) Specifies the NetWare server connection handle.

`installed`

(OUT) Points to the installed value. It returns 1 if accounting is installed on the NetWare server; otherwise, `installed` returns 0.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL

0x8801	INVALID_CONNECTION
0x8811	INVALID_SHELL_CAL
0x8836	INVALID_PARAMETER
0x890A	NLM_INVALID_CONNECTION

Remarks

Under NETX, if an invalid connection handle is passed to `conn`, NWQueryAccountingInstalled will return 0x0000. NETX will pick a default connection handle if the connection handle cannot be resolved.

When accounting is enabled, the NetWare server object has the property ACCOUNT_SERVERS.

NCP Calls

0x2222 23 17 Get File Server Information
0x2222 23 22 Get Station's Logged Info (old)
0x2222 23 28 Get Station's Logged Info
0x2222 23 60 Scan Property
0x2222 104 1 Ping for NDS NCP

NWSubmitAccountCharge

Charges an amount against an object's balance and either relinquishes a hold or reduces the hold by the charge amount

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Accounting

Syntax

```
#include <nwacct.h>
or
#include <nwcalls.h>

NWCCODE N_API NWSubmitAccountCharge (
   NWCONN_HANDLE       conn,
   nuint16             objType,
   const nstr8 N_FAR  *objName,
   nuint16             serviceType,
   nint32              chargeAmt,
   nint32              holdCancelAmt,
   nuint16             noteType,
   const nstr8 N_FAR  *note);
```

Pascal Syntax

```
uses calwin32

Function NWSubmitAccountCharge
  (conn : NWCONN_HANDLE;
   objType : nuint16;
   objName : pnstr8;
   serviceType : nuint16;
   chargeAmt : nint32;
   holdCancelAmt : nint32;
   noteType : nuint16;
   note : pnstr8
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

objType

(IN) Specifies the type of the object to be charged.

objName

(IN) Points to a string containing the name of the object whose account is to be charged (48 character maximum or will be truncated).

serviceType

(IN) Specifies the type of service for which the charge is being made.

chargeAmt

(IN) Specifies the amount to be charged to the object account balance.

holdCancelAmt

(IN) Specifies the amount the hold will be reduced (set to zero if there are no holds).

noteType

(IN) Specifies the note type to be written to the audit report.

note

(IN) Points to a note associated with the charge and stored as an audit record (255 character maximum or will be truncated).

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8901	ERR_INSUFFICIENT_SPACE
0x890A	NLM_INVALID_CONNECTION
0x8988	INVALID_FILE_HANDLE
0x8994	NO_WRITE_PRIVILEGES_OR_READONLY
0x8996	SERVER_OUT_OF_MEMORY
0x89A2	READ_FILE_WITH_RECORD_LOCKED
0x89C0	NO_ACCOUNTING_PRIVILEGES
0x89C1	LOGIN_DENIED_NO_ACCOUNT_BALANCE
0x89C2	LOGIN_DENIED_NO_CREDIT
0x89C4	ACCOUNTING_DISABLED
0x89E8	WRITE_PROPERTY_TO_GROUP
0x89EA	NO_SUCH_MEMBER

0x89EB	NOT_GROUP_PROPERTY
0x89EC	NO_SUCH_SEGMENT
0x89EF	INVALID_NAME
0x89F0	WILD_CARD_NOT_ALLOWED
0x89FC	NO_SUCH_OBJECT
0x89FE	BINDERY_LOCKED
0x89FF	HARDWARE_FAILURE

Remarks

NWSubmitAccountCharge can write a note about the transaction in an audit record (optional). The charge and hold amounts do not have to be the same.

`objType` and `objName` must uniquely specify the object and cannot contain wildcard characters.

`serviceType` usually contains the object type of the charging account server. The common server object types are listed below:

```
Object           Type               High-Low Format
Job Server       OT_JOB_SERVER      0x0500
Print Server     OT_PRINT_SERVER    0x0700
Archive Server   OT_ARCHIVE_SRVER   0x0900
```

`noteType` contains the number of the note type. Note types are administered by Novell®, Inc. and are listed below:

```
Note      Description Type
1         Connect time charge
2         Disk storage charge
3         Log in note
4         Log out note
5         Account locked note
6         Server time modified note
```

Developers should contact Novell for unique note types. Note types greater than 8000H are reserved.

NOTE: `note` is the entry the value-added server makes in SYS:SYSTEM\NET$ACCT.DAT.

NCP Calls

0x2222 23 151 Submit Account Charge

See Also

NWSubmitAccountHold (page 34), NWSubmitAccountNote (page 37)

NWSubmitAccountHold

Reserves a specified amount of an object's account balance before the object receives and is charged for a service on the network

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Accounting

Syntax

```
#include <nwacct.h>
or
#include <nwcalls.h>

NWCCODE N_API NWSubmitAccountHold (
   NWCONN_HANDLE       conn,
   nuint16             objType,
   const nstr8 N_FAR  *objName,
   nint32              holdAmt);
```

Pascal Syntax

```
uses calwin32

Function NWSubmitAccountHold
  (conn : NWCONN_HANDLE;
   objType : nuint16;
   objName : pnstr8;
   holdAmt : nint32
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

objType

(IN) Specifies the type of the object for which the hold is desired.

objName

(IN) Points to the name of the object for which the hold is desired (48 character maximum).

holdAmt

(IN) Specifies the amount to be held against the object's account balance.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8901	ERR_INSUFFICIENT_SPACE
0x890A	NLM_INVALID_CONNECTION
0x8988	INVALID_FILE_HANDLE
0x8994	NO_WRITE_PRIVILEGES_OR_READONLY
0x8996	SERVER_OUT_OF_MEMORY
0x89A2	READ_FILE_WITH_RECORD_LOCKED
0x89C0	NO_ACCOUNTING_PRIVILEGES
0x89C1	LOGIN_DENIED_NO_ACCOUNT_BALANCE
0x89C2	LOGIN_DENIED_NO_CREDIT
0x89C3	ERR_TOO_MANY_HOLDS
0x89C4	ACCOUNTING_DISABLED
0x89E8	WRITE_PROPERTY_TO_GROUP
0x89EA	NO_SUCH_MEMBER
0x89EB	NOT_GROUP_PROPERTY
0x89EC	NO_SUCH_SEGMENT
0x89EF	INVALID_NAME
0x89F0	WILD_CARD_NOT_ALLOWED
0x89FC	NO_SUCH_OBJECT
0x89FE	BINDERY_LOCKED
0x89FF	HARDWARE_FAILURE

Remarks

NWSubmitAccountHold reserves a specified amount of an object's account balance that object receives and is charged for services on the network.

`objType` and `objName` must uniquely identify the object and cannot contain wildcard characters.

`holdAmt` gets the amount the server expects to charge for the service it is about to provide to the object.

NOTE: No more than 16 servers can reserve amounts of an object's account balance at one time. Multiple holds from the same server are combined.

NCP Calls

0x2222 23 152 Submit Account Hold

See Also

NWQueryAccountingInstalled (page 29), NWGetAccountStatus (page 26), NWSubmitAccountCharge (page 31), NWSubmitAccountNote (page 37)

NWSubmitAccountNote

Adds a note about an accounting transaction to an audit record

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform NetWare Calls (CAL*.*)

Service: Accounting

Syntax

```
#include <nwacct.h>
or
#include <nwcalls.h>

NWCCODE N_API NWSubmitAccountNote (
   NWCONN_HANDLE       conn,
   nuint16             objType,
   const nstr8 N_FAR  *objName,
   nuint16             serviceType,
   nuint16             noteType,
   const nstr8 N_FAR  *note);
```

Pascal Syntax

```
uses calwin32

Function NWSubmitAccountNote
  (conn : NWCONN_HANDLE;
   objType : nuint16;
   objName : pnstr8;
   serviceType : nuint16;
   noteType : nuint16;
   note : pnstr8
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

objType

(IN) Specifies the type of the object for which the note is being submitted.

objName

(IN) Points to the name of the object to receive the note (48 character maximum or will be truncated).

serviceType

(IN) Specifies the object type of the charging account server.

noteType

(IN) Specifies the note type (255 character maximum or will be truncated).

note

(IN) Points to a note to be added to an audit record.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x8901	ERR_INSUFFICIENT_SPACE
0x890A	NLM_INVALID_CONNECTION
0x8996	SERVER_OUT_OF_MEMORY
0x89C0	NO_ACCOUNTING_PRIVILEGES
0x89C1	LOGIN_DENIED_NO_ACCOUNT_BALANCE
0x89C4	ACCOUNTING_DISABLED
0x89E8	WRITE_PROPERTY_TO_GROUP
0x89EA	NO_SUCH_MEMBER
0x89EB	NOT_GROUP_PROPERTY
0x89EC	NO_SUCH_SEGMENT
0x89EF	INVALID_NAME
0x89F0	WILD_CARD_NOT_ALLOWED
0x89FC	NO_SUCH_OBJECT
0x89FF	HARDWARE_FAILURE

Remarks

`objType` and `objName` must uniquely identify the object and cannot contain wildcard characters.

`serviceType` usually contains the object type of the charging account server. The common server object types are listed below:

```
Object          Type                High-Low Format
Job Server      OT_JOB_SERVER       0x0500
```

```
Print Server      OT_PRINT_SERVER      0x0700
Archive Server    OT_ARCHIVE_SRVER     0x0900
```

noteType contains the number of the note type. Note types are administered by Novell and are listed below:

Note	*Description Type*
1	Connect time charge
2	Disk storage charge
3	Log in note
4	Log out note
5	Account locked note
6	Server time modified note

Developers should contact Novell for unique note types. Note types greater than 8000H are reserved.

NOTE: note contains the entry the server adds to the audit record. The audit record is contained in SYS:SYSTEM\NET$ACCT.DAT.

NCP Calls

0x2222 23 153 Submit Account Note

See Also

NWSubmitAccountCharge (page 31), NWSubmitAccountHold (page 34)

Accounting Structures

3

This documentation alphabetically lists the Accounting structures and describes their purpose, syntax, and fields.

HOLDS_INFO

Defines information for an object that places a hold on an account

Service: Accounting

Defined In: nwacct.h

Structure

```
typedef struct
{
   nuint32   objectID ;
   nint32    amount ;
} HOLDS_INFO;
```

Pascal Structure

```
uses calwin32

   HOLDS_INFO = packed Record
      objectID : nuint32;
      amount : nint32
   End;
```

Fields

`objectID`

Specifies the object ID of the server placing the hold. Multiple holds from the same server are combined.

`amount`

Specifies the amount of the associated `objectID` 's hold.

HOLDS_STATUS

Stores a list of holds placed on an account

Service: Accounting

Defined In: nwacct.h

Structure

```
typedef struct
{
   nuint16      holdsCount ;
   HOLDS_INFO   holds [16];
} HOLDS_STATUS;
```

Pascal Structure

```
uses calwin32

HOLDS_STATUS = packed Record
      holdsCount : nuint16;
      holds : Array[0..15] Of HOLDS_INFO
   End;
```

Defined In

nwacct.h

Fields

`holdsCount`

Specifies the number of `NW_HOLDS` in `holds` array.

`holds`

Specifies a list of servers calling NWSubmitAccountHold (page 34) against an object and the amount reserved by each value-added server.

Auditing Concepts

4

This documentation describes Auditing, its functions, and features.

Auditing can be applied to nonauditing activities. For example, an administrator might set up an audit file as a log for analyzing and monitoring network usage, or an NLM application might use an audit file to track special events. Auditing provides the interface to the auditing system, giving your applications secure procedures for setting up audits and reading the results of auditing sessions.

4.1 C2 Auditing Security Requirements

Security is an important aspect of Auditing due to the sensitive nature of auditing procedures and records. NetWare® 4.11 auditing is designed to meet the Class C2 security criteria as specified by the US Department of Defense (DoD) Trusted Computer System Evaluation Criteria (TCSEC) also known as the "orange book."

4.2 Auditing Volumes and Containers

You can set up auditing for either volumes or NDS container objects. Use volume auditing to audit the following:

Accounting events
Extended Attribute events
File events
Message events
QMS events
Server events
User events

Container auditing allows you to audit container NDS events.

When you enable auditing on a container, you audit all objects directly subordinate to the container. You can audit as many objects as you choose by selecting the appropriate containers on which to enable auditing.

When you enable auditing on a volume, you can audit all objects located on the volume

In this document, when there are no differences whether you are auditing a volume or a container, the term object is used to refer to either a volume or container.

4.3 Auditing Security

Audit security is different depending on whether you are using NetWare® versions previous to 4.11. NetWare versions prior to 4.11 use passwords to access and configure the audit files. The network admin enables auditing and assigns an initial password to the auditor. However, the admin isn’t the security equivalent of the auditor and consequently, changing the auditor’s password can prevent even the admin from accessing the audit file.

Access to the audit file with NetWare 4.11 and above is based on an NDS object, the Audit File Object (AFO), and read and write rights to the object. The network admin creates the AFO and assigns rights to the auditor.

4.4 Enabling and Disabling Auditing

Enabling and disabling auditing is different depending on whether you are using NetWare 4.11 and above or previous versions of NetWare 4. However, in both cases, when auditing is enabled an audit file is automatically created. This file is where the audit data is stored.

4.4.1 Auditing Passwords for NetWare 4 Versions Prior to 4.11

Access control to an audit file can be single-tiered or two-tiered. Two-tiered security uses two passwords: one for reading audits and one for configuring audits. As an auditor, you activate two-tiered security for an audited object by calling NWADInitLevelTwoPassword (page 98).

The current level of audit security is indicated by an object's auditing flags. For more information, see "Auditing Flags" on page 51. Two functions verify an auditor's level 1 and level 2 access:

- NWADCheckAccess (page 70) checks level 1 access.
- NWADCheckLevelTwoAccess (page 72) checks level 2 access.

To change either the level 1 or level 2 password, you must be logged in to the object as auditor and receive a valid audit key. Call NWADChangePassword (page 67) to change the passwords.

For related information, see:

- Section 5.3, "Enabling Auditing for NetWare 4 Versions Prior to 4.11," on page 59
- Section 5.4, "Disabling Auditing for NetWare 4 Versions Prior to 4.11," on page 60

AFO Attributes for NetWare 4.11 and above

By creating the AFO using the mandatory NDS attributes you give the user auditor status and access to the audit file. The AFO has two mandatory and several optional NDS attributes. The mandatory attributes are as follows:

- Audit:Contents is where read and write rights are assigned to the auditor.
- Audit:Policy is where you define the audited events, audit archiving options, audit file overflow recovery options, maximum size of the audit file, whether encryption is used, and the number of old audit files.

The optional attributes are as follows:

- Audit:A Encryption Key
- Audit:B Encryption Key
- Audit:Current Encryption Key
- Audit:Path points to the path to the NDS volume where the audit file resides.
- Audit:Link List points to the container, volume, or one or more workstations for which the AFO stores audit
- Audit Type specifies the type of auditing you are doing:

0 = volume auditing
1 = container auditing
2 = external auditing

- Also included are all the standard minimum attributes inherited from Super Class Top.

For related information, see:

- Section 5.1, "Enabling Auditing for NetWare 4.11 and above," on page 59
- Section 5.2, "Disabling Auditing for NetWare 4.11 and above," on page 59

4.4.2 Audit History Files

The audit file is automatically created when auditing is enabled and is accessible only to the auditor.

To close the current audit file and open a new one call NWADResetFile (page 120). The closed file becomes a history file. Up to 15 audit history files can be maintained by Auditing.

Several functions allow you to manipulate the audit files.

- NWADDeleteFile (page 79) deletes an old audit file.
- NWADOpenRecordFile (page 109) opens a record file and allocates a `recordHandle`.
- NWADCloseRecordFile (page 78) closes the record file and frees the `recordHandle` allocated by NWADOpenRecordFile (page 109).
- NWADReadRecord (page 117) reads a specified record.
- NWADGetFileList (page 89) returns the list of files.

A parameter in some of these functions allows you to specify which file you want to use. Set the parameter to -1 to view the current file. To view history files set the parameter to a number from 0 to 15.

The audit file consists of a configuration header and a data segment. The configuration header specifies what type of events are to be audited. The data segment stores event records describing events that have occurred. For more information see Section 4.8, "Audit File Configuration Header," on page 51.

IMPORTANT: Be sure to close and remove all audit files before deleting a container on which auditing has been enabled.

4.5 Auditing Event Records

Event records are generated according to the event bitmap in the configuration header. These records make up the data segment of the audit file. The configuration header includes an *audit record count* that records the current number of event records.

4.5.1 Fields of Auditing Event Records

Although the information in an event record varies for volumes and containers, the key fields in both headers are `process unique ID` and `event type ID`. The process unique ID is the bindery object ID identifying the object that performed the event. The event type ID identifies the type of event audited.

Any event-specific information is appended to the end of the record as an event field. By checking the event type ID, you can determine how to interpret the event. For example, if the event type ID is A_EVENT_BIND_CREATE_PROPERTY is returned, a bindery property was created, and the event field contains the property name and security value.

The event records also record the date and time of the event and its success or failure. Volume events record the connection ID of the station where the event was performed. Container events record the user ID of the object performing the event. Container events also record the replica an event was performed on.

4.5.2 Reading Auditing Event Records

To begin reading event records, call NWADOpenRecordFile (page 109). This function opens the record file and allocates a `recordHandle`. You can then read the event records one at a time by calling NWADReadRecord (page 117). Don't attempt to read the audit file directly. Since the file is compressed it must be read only with Auditing.

Resetting the audit file clears current records and stores them in an old audit file.

For related information, see "Fields of Auditing Event Records" on page 47.

4.6 Audit Status Information

Audit status information indicates whether auditing is enabled on a specified object. If auditing is enabled, the information also includes statistics on the object's audit file. This information is available to any client that has logged in unless you are using NetWare® 4.11, in which case very little information is available to any client.

NWADGetStatus returns audit status information for objects.

Audit status information is returned as an NWAuditStatus structure, indicating the size and version of the audit file.

4.7 Auditing Functions

The following sections group the Auditing functions into functional categories:

4.7.1 Audit Security Functions

These functions control auditor access to volumes and containers (objects) and perform operations on auditor passwords.

Function	Header File	Description
NWADChangePassword	nwaudit.h	Changes the auditor's password for the specified object.
NWADCheckAccess	nwaudit.h	Determines whether the caller has level 1 auditor access to the object.
NWADCheckLevelTwoAccess	nwaudit.h	Determines whether the caller has level 2 auditor access to the object.
NWADInitLevelTwoPassword	nwaudit.h	Initializes an audit key for level 2 access.
NWADLogin	nwaudit.h	Initializes an audit key for level 1 access to the object.
NWADLogout	nwaudit.h	Removes the caller's auditor status from an object.
NWADOpen	nwaudit.h	Allocates `auditHandle` for use in other Auditing functions.
NWADRestartVolumeAuditing	nwaudit.h	Restarts auditing for volumes only.
NWADSetPassword	nwaudit.h	Sets a password on for NetWare® 4.11 only.

4.7.2 Audit Status Functions

These functions enable and disable auditing on volumes and containers (objects) and return audit status information.

Function	Header File	Description
NWADDisable	nwaudit.h	Disables auditing on the specified object.
NWADEnable	nwaudit.h	Enables auditing on the specified object.
NWADGetFlags	nwaudit.h	Returns the auditing flags from the object's audit file configuration header.
NWADGetStatus	nwaudit.h	Returns auditing status information for the specified object.

4.7.3 Audit File Functions

These functions open and close the audit file and the history file and read event records from these files.

Function	Header File	Description
NWADAppendExternalRecords	nwaudit.h	Adds external audit events to an event record.
NWADCloseOldFile	nwaudit.h	Closes the audit file that was opened when auditing was enabled.

Function	Header File	Description
NWADCloseRecordFile	nwaudit.h	Frees `recordHandle` allocated by NWADOpenRecordFile (page 109).
NWADDeleteFile	nwaudit.h	Deletes the old file on the specified object.
NWADDeleteOldFile	nwaudit.h	Deletes an old file in the file list.
NWADGetFileList	nwaudit.h	Returns the file list.
NWADOpenRecordFile	nwaudit.h	Opens the record file and allocates a `recordHandle`.
NWADResetFile	nwaudit.h	Closes the object's current auditing file and opens a new one.
NWADReadRecord	nwaudit.h	Reads a specified record.

4.7.4 Configuration Header Functions

These functions read and write the data stored in the audit file configuration header.

Function	Header File	Description
NWADReadConfigHeader	nwaudit.h	Returns the audit file configuration header for the specified object.
NWADWriteConfigHeader	nwaudit.h	Saves the configuration header data to the audit file on the specified object.
NWADReadBitMap	nwaudit.h	Returns the event bitmap in the audit file configuration header for the specified volume.
NWADWriteBitMap	nwaudit.h	Writes values to the event bitmap in the audit file configuration header for the specified volume.

4.7.5 Audit Property Functions

These functions add, remove, and check for an object's audit property. By adding the property to a user object, you can audit the user's file operations individually.

Function	Header File	Description
NWADChangeObjectProperty	nwaudit.h	Changes the audit property for a user on the specified object.
NWGetNWADVersion	nwaudit.h	Returns version information about the NWAD library.
NWADIsObjectAudited	nwaudit.h	Determines whether the specified object or user is being audited.

4.8 Audit File Configuration Header

This topic discusses the audit file configuration header and how it controls the auditing on the associated volume or container.

4.8.1 Audit File Configuration Header Introduction

The audit file configuration header controls the auditing on the associated volume or container. Although configuration headers differ somewhat between volumes and containers, both contain the following key fields:

- A set of audit flags that indicate the status of auditing for the object.
- An event bitmap that determines which events are audited.
- A count of the current number of events that have been recorded in the audit file.

Additional fields contain information about the state of the audit file. The NWConfigHeader (page 143) structure contains the volume configuration header data.

The configuration header for NetWare® 4.11 is twice the size of that for previous versions of NetWare. This allows for more events to be audited.

Use the following functions to read from or write to the configuration headers:

- NWADReadConfigHeader (page 114) reads an object header.
- NWADWriteConfigHeader (page 130) modifies an object header.

In the case of volumes, you can also read or write just the event bitmap in the header. For more information see “Audit File Configuration Header Event Bitmap” on page 52.

Read or write operations affect the entire configuration header. When modifying a header, be sure to assign values to all the header’s data items.

4.8.2 Auditing Flags

Auditing flags can be read separately from other data in the audit file configuration header. The flags control several important aspects of the auditing process and are stored as a set of bit flags in a single-byte value. There are five flags:

```
Byte 0 = DiscardAuditRcdsOnErrorFlag
Byte 1 = ConcurrentVolAuditorAccess
Byte 2 = DualLevelPasswordsActive (this byte is ignored with NetWare ® 4.11)
Byte 3 = BroadcastWarningsToAllUsers
Byte 4 = LevelTwoPasswordSet (this byte is ignored with NetWare 4.11)
```

- If DiscardAuditRcdsOnErrorFlag is set, the auditing records are discarded if an error occurs.
- If ConcurrentVolAuditorAccess is set more than one auditor can log into the object at a time.
- If DualLevelPasswordsActive is set, dual level password security is in effect.
- If BroadcastWarningsToAllUsers is set, audit warnings are broadcast to all users.

- If LevelTwoPasswordSet is set, the level 2 password has been assigned a value. (This last flag is meaningful only if the dual level passwords flag is set.) Bits 0 through 3 can be set by the auditor.

NWADGetFlags returns the auditing flags for an object.

4.8.3 Audit File Configuration Header Event Bitmap

The event bitmap in the configuration header is the key to configuring the audit file. Each bit in the event bitmap represents an event to be audited. In its initial state, none of the bits in the event bitmap is set. It's up to the auditor to access the audit file's configuration header and set the bits that correspond to events the auditor wants audited.

The event bitmap for containers is a 32-bit value with 32 corresponding events. The event bitmap for volumes is 512 bits (64 bytes).

Events that can be audited fall into several categories:

- NDS Events
- Bindery Events
- NetWare® Server Events
- Queue Management Events
- File System Events
- User Object Events

You must use container auditing to audit NDS events. All other events must be audited on a volume basis. For each category there are many events you can audit. Setting the bit for an event results in an event record being added to the audit file every time that event occurs.

For related information, see:

- "Scope of Auditing Events" on page 52
- "Auditing File Events (NetWare 4.x and above)" on page 53
- "Reading a Volume Event Bitmap" on page 53
- "Event Bits Tables" on page 54

Scope of Auditing Events

The event bitmap doesn't single out specific users to be audited unless you are using NetWare® 4.11 and user restrictions is turned on using the Auditcon utility.

For previous versions of NetWare 4, once an event is set, every occurrence of the event is audited regardless of who performs it. For example, if you set the Delete Bindery Property Event bit, then an event record is generated every time a user deletes a property from an object. File events are an exception. You can configure file auditing to target specific files and users. For more information see "Auditing File Events (NetWare 4.x and above)" on page 53.

If you are interested only in events involving specific items, you can filter the information you read from the audit file by searching specific fields in the event records. For example, you could search for all event records in which a particular user deleted properties from a particular object.

Auditing File Events (NetWare 4.*x* and above)

For NetWare 4 versions prior to 4.11, file events are the only category of events for which you can target specific files and users. To audit file events you must configure the files and the user objects in addition to setting the file event bit in the volume's event bitmap. To target specific files you must set each file's audit attribute. To target specific users you must add an audit property to each user's bindery object. Both procedures are explained below.

User Audit Property

To audit a particular user's file operations, you must assign an audit property to the user's object by calling NWADChangeObjectProperty (page 64). The property doesn't take a value. If the property is present, the specified user will be audited. You must specify the volume or container on which the user is being audited.

For related information, see "File Events for Auditing" on page 53.

Reading a Volume Event Bitmap

The volume event bitmap is a bit stream 512 bits long. Special functions are defined to simplify access to this bitmap.

- NWADReadBitMap (page 112) reads he event bitmap.
- NWADWriteBitMap (page 128) modifies the event bitmap.

File Events for Auditing

Once you set the audit attribute for specific files or add the audit property to specific users, the next step is to set the event bitmap for file operations you want to audit. Some file events are represented by three different bits in the event bitmap: a global bit, a union bit, and an intersection bit. (See the "File Events" on page 54.) Set the one appropriate for the events you are auditing.

Open File audit is an example of the choices presented by the three bits:

- The Global Open File bit generates an event record every time a file is opened (regardless of which file is involved or who opens it).
- The Union Open File bit generates an event record whenever an audited file is opened or an audited user opens a file.
- The Intersection Open File bit generates an event record only when an audited user opens an audited file; other open file operations are ignored.

For related information, see "User Audit Property" on page 53.

Event Bitmap

The volume event bitmap is a bit stream 512 bits long. Special functions are defined to simplify access to this bitmap:

- NWADReadBitMap (page 112) reads the event bitmap.
- NWADWriteBitMap (page 128) modifies the event bitmap.

Event Bits Tables

This is a list of tables that define event bits.

Message Events Table

The following table lists the bits defined in the message event bitmap.

A_EVENT_BROADCAST_TO_CONSOLE	207
A_EVENT_DISABLE_BROADCASTS	204
A_EVENT_ENABLE_BROADCASTS	206
A_EVENT_GET_BROADCAST_MESSAGE	205
A_EVENT_SEND_BROADCAST_MESSAGE	208

Accounting Events Table

The following table lists the bits defined in the accounting events bitmap.

Event Type ID	Event ID
A_EVENT_GET_CURRET_ACNT_STATS	200
A_EVENT_SUBMIT_ACCOUNT_CHARGE	201
A_EVENT_SUBMIT_ACCOUNT_HOLD	202
A_EVENT_SUBMIT_ACCOUNT_NOTE	203

File Events

The following table lists the bits defined in the File event bitmap.

Event Type ID	Event ID
A_EVENT_CREATE_DIRECTORY	75
A_EVENT_DELETE_DIRECTORY	76
A_EVENT_CLOSE_FILE	10
A_EVENT_CREATE_FILE	12

Event Type ID	Event ID
A_EVENT_DELETE_FILE	14
A_EVENT_OPEN_FILE	27
A_EVENT_PURGE_FILE	214
A_EVENT_READ_FILE	42
A_EVENT_RENAME_MOVE_FILE	44
A_EVENT_SALVAGE_FILE	46
A_EVENT_WRITE_FILE	57
A_EVENT_MODIFY_ENTRY	25
A_EVENT_SCAN_DELETED	215
A_EVENT_SCAN_VOL_USER_REST	238
A_EVENT_SET_COMP_FILE_SIZE	242

QMS Events Table

The following table lists the bits defined in the QMS event bitmap.

Event Type ID	Event ID
A_EVENT_Q_JOB_FROM_LIST	231
A_EVENT_Q_JOB_LIST	230
A_EVENT_Q_JOB_SIZE	229
A_EVENT_MOVE_Q_JOB	233
A_EVENT_Q_ATTACH_SERVER	28
A_EVENT_Q_CREATE	29
A_EVENT_Q_CREATE_JOB	30
A_EVENT_Q_DESTROY	31
A_EVENT_Q_DETACH_SERVER	32
A_EVENT_Q_EDIT_JOB	33
A_EVENT_Q_JOB_FINISH	34
A_EVENT_Q_JOB_SERVICE	35
A_EVENT_Q_JOB_SERVICE_ABORT	36
A_EVENT_Q_SWAP_RIGHTS	41
A_EVENT_Q_REMOVE_JOB	37
A_EVENT_Q_SET_JOB_PRIORITY	38
A_EVENT_Q_SET_STATUS	39

Event Type ID	Event ID
A_EVENT_Q_START_JOB	40
A_EVENT_READ_Q_JOB_ENTRY	232
A_EVENT_MOVE_Q_JOB	233
A_EVENT_READ_Q_STATUS	234
A_EVENT_READ_Q_SERVER_STATUS	235
A_EVENT_SET_Q_SERVER_STATUS	261

Server Events Table

The following table lists the bits defined in the server event bitmap.

Event Type ID	Event ID
A_EVENT_CHANGE_DATE_TIME	7
A_EVENT_CONVERT_PATH_TO_ENTRY	259
A_EVENT_DISABLE_LOGIN	243
A_EVENT_DISABLE_ TTS	245
A_EVENT_DOWN_SERVER	18
A_EVENT_ENABLE_LOGIN	244
A_EVENT_ENABLE_TTS	246
A_EVENT_GET_CONN_OPEN_FILES	250
A_EVENT_GET_CONN_SEMS	256
A_EVENT_GET_CONN_TASKS	249
A_EVENT_GET_CONN_USING_FILE	251
A_EVENT_GET_LOG_REC_INFO	255
A_EVENT_GET_LOG_REC_CONN	254
A_EVENT_GET_REMAIN_OBJ_DISK_SPC	248
A_EVENT_GET_PHYS_REC_LOCKS_CONN	252
A_EVENT_GET_PHYS_REC_LOCKS_FILE	253
A_EVENT_GET_SEM_INFO	257
A_EVENT_GET_DISK_UTILIZATION	240
A_EVENT_MAP_DIR_TO_PATH	258
A_EVENT_VOLUME_DISMOUNT	56
A_EVENT_VOLUME_MOUNT	55
A_EVENT_SEND_CONSOLE_BROADCAST	247

Event Type ID	Event ID
A_EVENT_CONSOLE_COMMAND	262
A_EVENT_DESTROY_SERVICE_CONN	260
A_EVENT_VERIFY_SERIAL	239
A_EVENT_VOLUME_DISMOUNT	56
A_EVENT_VOLUME_MOUNT	55

User Events Table

The following table lists the bits defined in the user event bitmap.

Event Type ID	Event ID
A_EVENT_DISABLE_ACCOUNT	17
A_EVENT_GRANT_TRUSTEE	19
A_EVENT_LOGIN_USER	21
A_EVENT_LOGOUT_USER	23
A_EVENT_REMOVE_TRUSTEE	43
A_EVENT_USER_SPACE_RESTRICTIONS	53
A_EVENT_USER_LOCKED	52
A_EVENT_USER_CHANGE_PASSWORD	51
A_EVENT_USER_UNLOCKED	54
A_EVENT_RENAME_USER	45

NDS Events Table

The following table lists the bits defined in the NDS event bitmap.

Event Type ID	Event ID
ADS_ADD_ENTRY	101
ADS_REMOVE_ENTRY	102
ADS_RENAME_OBJECT	103
ADS_MOVE_ENTRY	104
ADS_CHANGE_SECURITY_EQUIV	105
ADS_CHG_SECURITY_ALSO_EQUAL	106
ADS_CHANGE_ACL	107
ADS_CHG_STATION_RESTRICTION	108
ADS_LOGIN	109

Event Type ID	Event ID
ADS_LOGOUT	110
ADS_CHANGE_PASSWORD	111
ADS_USER_LOCKED	112
ADS_USER_UNLOCKED	113
ADS_USER_DISABLE	114
ADS_USER_ENABLE	115
ADS_CHANGE_INTRUDER_DETECT	116
ADS_ADD_PARTITION	117
ADS_REMOVE_PARTITION	118
ADS_ADD_REPLICA	119
ADS_REMOVE_REPLICA	120
ADS_SPLIT_PARTITION	121
ADS_JOIN_PARTITION	122
ADS_CHANGE_REPLICA_TYPE	123
ADS_REPAIR_TIME_STAMPS	124
ADS_MOVE_SUB_TREE	125
ADS_ABORT_PARTITION_OP	126
ADS_SEND_REPLICA_UPDATES	127
ADS_RECEIVE_REPLICA_UPDATES	128

Additional NDS Events (for NetWare 4.11 and above)

NDS has added many more events to support auditing and other functions. For information about these events, see *NDS Event Services*.

For related information, see "Event Bits Tables" on page 54.

Auditing Tasks

5

This documentation describes common tasks associated with Auditing.

5.1 Enabling Auditing for NetWare 4.11 and above

1. Create a DS Audit File Object (AFO) in the container you want to audit using either DS calls or the Auditcon utility.
2. Give a user (the auditor) read and write rights to the Audit:Policy and Audit:Contents attributes of the AFO.
3. Call NWADOpen (page 107) to allocate an audit handle for use in other auditing functions.

NOTE: The creator of the AFO is automatically given auditor status. Remove the creator's DS rights to the AFO if you do not want the creator to have auditor access.

For related information, see "AFO Attributes for NetWare 4.11 and above" on page 46.

5.2 Disabling Auditing for NetWare 4.11 and above

1. Call NWADClose (page 75) to free the audit handle allocated by NWADOpen (page 107).

For related information, see "AFO Attributes for NetWare 4.11 and above" on page 46.

5.3 Enabling Auditing for NetWare 4 Versions Prior to 4.11

1. Call NWADLogin (page 102) to initialize the auditing password.
2. Call NWADEnable (page 86) to enable auditing on the object.

NOTE: For first time auditing, the login function returns ERR_AUDITING_NOT_ENABLED. You can then call NWADEnable (page 86) to enable auditing. The first-time auditor must have supervisor equivalence.

Don't confuse logging in to a volume or container as auditor with logging in to a NetWare® server. Audit login is a unique procedure implemented through a special service request.

If a second-level password is in effect, it must also be initialized. The auditor password is required even when auditing is disabled. For more information see "Auditing Passwords for NetWare 4 Versions Prior to 4.11" on page 46.

When you log in to an object, Auditing returns an audit key. The key permits access to the audit data for the object you have logged in to by providing access to the password calls. Remember to remove the password from memory as soon as you receive the audit key. The audit key is the basis for security throughout an auditing session.

For related information, see “Auditing Passwords for NetWare 4 Versions Prior to 4.11” on page 46.

5.4 Disabling Auditing for NetWare 4 Versions Prior to 4.11

1 Call NWADDisable (page 84) to disable auditing and close the audit file.

For related information, see “Auditing Passwords for NetWare 4 Versions Prior to 4.11” on page 46.

Auditing Functions

6

This documentation alphabetically lists the auditing functions and describes their purpose, syntax, parameters, and return values.

NWADAppendExternalRecords

Adds external audit events to an event record

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERNAL_LIBRARY NWRCODE NWADAppendExternalRecords  (
   NWCONN_HANDLE     conn,
   nuint32           auditFileObjectID,
   nuint32           vendorID,
   nuint32           numberRecords,
   pNWAuditRecord    recordsPtr);
```

Pascal Syntax

```
uses audwin32

Function NWADAppendExternalRecords
  (conn : NWCONN_HANDLE;
   auditFileObjectID : nuint32;
   vendorID : nuint32;
   numberRecords : nuint32;
   recordsPtr : pNWAuditRecord
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare® server connection handle.

auditFileObjectID

(IN) Specifies the NetWare version audit object ID.

vendorID

(IN) Specifies the vendor workstation number assigned by Novell.

numberRecords

(IN) Specifies the number of audit event records in the NWAuditRecord structure.

recordsPtr

(IN) Points to the NWAuditRecord structure containing the records to be stored.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWADAppendExternalRecords will only work with NetWare 4.11. For a list of event bits, see "Event Bits Tables" on page 54.

NCP Calls

104 52 External Audit Append To File

NWADChangeObjectProperty

Adds or removes the audit property for a user on a specified volume or container

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADChangeObjectProperty  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle,
   nuint32         objectID,
   nuint8          auditFlag);
```

Pascal Syntax

```
uses audwin32

Function NWADChangeObjectProperty
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr;
   objectID : nuint32;
   auditFlag : nuint8
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare® server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the `auditHandle` allocated by NWADOpen.

objectID

(IN) Specifies the object ID number on which to change the audit property. If the object to be changed is a volume, `objectID` specifies the user ID; it specifies the NDS object number if an NDS object is to be changed.

auditFlag

(IN) Specifies whether the object will be audited: 1 = the object will be audited; 0 = the object will not be audited.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWADChangeObjectProperty requires a second-level password when enabled.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 02 Add User Audit Property
0x2222 88 06 Delete Audit Property

0x2222 104 221 Audit Change Object Audited

See Also

NWADInitLevelTwoPassword (page 98), NWADIsObjectAudited (page 100), NWADLogin (page 102), NWADOpen (page 107), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADChangePassword

Changes the auditor's password for a specified volume or container

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADChangePassword  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle,
   pnuint8         newPassword,
   nuint8          level);
```

Pascal Syntax

```
uses audwin32

Function NWADChangePassword
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr;
   Var newPassword : nuint8;
   level : nuint8
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the `auditHandle` allocated by NWADOpen.

newPassword

(IN) Points to a NULL-terminated character string containing the new password.

level

(IN) Specifies which password to change; for example, 2 = level two password.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x89D7	PASSWORD_NOT_UNIQUE
0x89D8	PASSWORD_TOO_SHORT
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

If NWADChangePassword fails, the original password is still valid.

NWADChangePassword is only supported for NetWare 4.1 and above. To call NWADChangePassword under NetWare 4.11, a password has to be set and the user who set the password cannot be a password user.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 04 Change Auditor Volume Password
0x2222 88 18 Change Audit Level Two Volume Password
0x2222 88 19 Get Auditing Flags
0x2222 104 203 Directory Services Change Audit Password
0x2222 104 215 Directory Services Change Audit Level Two Password
0x2222 104 216 Get Auditing Flags

See Also

NWADLogin (page 102), NWADOpen (page 107), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADCheckAccess

Checks to see if the auditor has auditor access

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADCheckAccess  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID);
```

Pascal Syntax

```
uses audwin32

Function NWADCheckAccess
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x0001	No Audit Access
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

If zero (0) is returned, the user has auditor access and is currently logged in through NWADLogin.

If one (1) is returned, the user does not have auditor access.

In NetWare 4.11, you can only call NWADLogin once, which will set the auditor access on the server. If subsequent calls are made, an error will be returned.

The second level password is only supported under NetWare 4.1. NetWare 4.11 does not use second level passwords.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 05 Change Auditor Access
0x2222 104 204 Directory Services Check Auditor Access

See Also

NWADCheckLevelTwoAccess (page 72), NWADLogin (page 102), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADCheckLevelTwoAccess

Checks to see if the auditor has level two access

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADCheckLevelTwoAccess  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle);
```

Pascal Syntax

```
uses audwin32

Function NWADCheckLevelTwoAccess
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the auditHandle allocated by NWADOpen.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x0001	No Audit Access
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWADCheckLevelTwoAccess requires a level two password to be initialized.

If zero (0) is returned, the user has level two access.

If one (1) is returned, the user does not have audit level two access.

NWADCheckLevelTwoAccess is only supported on a NetWare 4.1 server. NetWare 4.11, 5.x, and 6.x do not use second level passwords.

If auditIDType is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If auditIDType is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 22 Check Level Two Access
0x2222 104 219 Directory Services Check Audit Level Two Access

See Also

NWADCheckAccess (page 70), NWADInitLevelTwoPassword (page 98), NWADLogin (page 102), NWADOpen (page 107), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADClose

Frees the `auditHandle` allocated by NWADOpen and NULL the pointer to the audit handle

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADClose  (
   pnptr    auditHandle);
```

Pascal Syntax

```
uses audwin32

Function NWADClose
  (Var auditHandle : nptr
) : NWRCODE;
```

Parameters

`auditHandle`

(IN) Points to the `auditHandle` allocated by NWADOpen.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL

Remarks

All `auditHandle` parameters allocated by NWADOpen must be freed by calling NWADClose.

See Also

NWADOpen (page 107)

NWADCloseOldFile

Closes the old auditing file automatically opened by the system when the volume or container is mounted or auditing is enabled

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADCloseOldFile  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle);
```

Pascal Syntax

```
uses audwin32

Function NWADCloseOldFile
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the auditHandle allocated by NWADOpen.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWADCloseOldFile requires a second-level password when enabled.

The old auditing file is kept open by the operating system. If a file needs to be copied or deleted, close it by calling NWADCloseOldFile first.

NWADCloseOldFile is only supported on NetWare 4.1. NWADCloseOldFile cannot be called from NetWare 4.11, 5.x, and 6.x.

If auditIDType is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If auditIDType is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 20 Close Old Audit File
0x2222 104 217 Directory Services Close Old Audit File

See Also

NWADDeleteFile (page 79), NWADInitLevelTwoPassword (page 98), NWADLogin (page 102), NWADOpen (page 107), NWADResetFile (page 120), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADCloseRecordFile

Frees the `recordHandle` allocated by NWADOpenRecordFile

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADCloseRecordFile  (
   pnptr   recordHandle);
```

Pascal Syntax

```
uses audwin32

Function NWADCloseRecordFile
  (Var recordHandle : nptr
) : NWRCODE;
```

Parameters

`recordHandle`

(IN/OUT) Points to the record handle to free.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL

Remarks

All resources used in the `recordHandle` s allocated by NWADOpenRecordFile are freed by calling NWADCloseRecordFile.

See Also

NWADOpenRecordFile (page 109), NWADReadRecord (page 117)

NWADDeleteFile

Deletes the old auditing file on the specified volume or container

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADDeleteFile  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle);
```

Pascal Syntax

```
uses audwin32

Function NWADDeleteFile
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the auditHandle allocated by NWADOpen.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWADDeleteFile requires a second level password when enabled.

NWADDeleteFile is only supported on NetWare 4.1. Call NWADDeleteOldFile for NetWare 4.1 and above for future compatibility.

NWADDeleteFile cannot be called from NetWare 4.11, 5.x, and 6.x.

If auditIDType is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If auditIDType is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 21 Delete Old Audit File

See Also

NWADCloseOldFile (page 76), NWADInitLevelTwoPassword (page 98), NWADLogin (page 102), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADDeleteOldFile

Deletes an old file in the file list

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADDeleteOldFile
  (NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle,
   nuint32         fileCode);
```

Pascal Syntax

```
uses audwin32

Function NWADDeleteOldFile
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr;
   fileCode : nuint32
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing

1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the `auditHandle` allocated by NWADOpen.

fileCode

(IN) Specifies the number of the file on the file list to delete (0-15).

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

If you want to delete the current file, call NWADResetFile to move the current file into the file list. Then call NWADDeleteOldFile to delete that file.

The file number zero (0) indicates the current history file; file numbers 1-15 indicate old files that can be deleted by calling NWADDeleteOldFile.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 26 Delete Old Audit File
0x2222 104 225 Delete Old Audit File

See Also

NWADOpen (page 107), NWADResetFile (page 120), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADDisable

Disables auditing on a specified volume or container

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADDisable  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle);
```

Pascal Syntax

```
uses audwin32

Function NWADDisable
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the auditHandle allocated by NWADOpen.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWADDisable requires a second level password when enabled.

You will lose audit access on the NetWare server after calling NWADDisable because auditing is disabled. To do more auditing, you must log in again by calling NWADLogin.

If a password user disables auditing, that same user cannot enable auditing again in NetWare 4.11. The only way auditing may be enabled again is through the authorized auditing NDS objects.

If auditIDType is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If auditIDType is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 07 Disable Volume Auditing
0x2222 104 206 Directory Services Disable Volume Auditing

See Also

NWADEnable (page 86), NWADInitLevelTwoPassword (page 98), NWADLogin (page 102), NWADOpen (page 107), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADEnable

Enables auditing on the specified NDS container or volume

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADEnable  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle);
```

Pascal Syntax

```
uses audwin32

Function NWADEnable
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the auditHandle allocated by NWADOpen.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled

Remarks

If auditing has never been initialized on the volume, call NWADLogin first; 0x8997 will be returned. Then, call NWADEnable.

If the user is not SUPERVISOR equivalent, NWADEnable will fail the first time it is called.

After NWADEnable has been called successfully, the user must log in again by calling NWADLogin to have access to auditing.

For NetWare 4.11, a different approach must be followed to enable auditing. You must log into NetWare through NDS and have the necessary rights to create objects and add attributes. You may then add auditor access by adding the AFO attributes to a user object and assigning it to a volume or container. This user can then enable auditing on the volume or container.

A password user on NetWare 4.11 cannot enable auditing.

If auditIDType is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If auditIDType is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 104 207 Directory Services Enable Container Volume Auditing

See Also

NWADDisable (page 84), NWADInitLevelTwoPassword (page 98), NWADLogin (page 102), NWADOpen (page 107), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADGetFileList

Returns the file list

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADGetFileList  (
   NWCONN_HANDLE       conn,
   nuint32             auditIDType,
   nuint32             auditID,
   nptr                auditHandle,
   pNWAuditFileList    fileList);
```

Pascal Syntax

```
uses audwin32

Function NWADGetFileList
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr;
   fileList : pNWAuditFileList
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing

1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the `auditHandle` allocated by NWADOpen.

fileList

(OUT) Points to the structure NWAuditFileList containing the size, date, and time for the past 16 elements.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWADGetFileList returns a list of the dates, times, and sizes of old audit files. The list contains a total of up to 16 (0-15) files that can be added to the list by calling NWADResetFile.

If the `fileCreateDateTime` field of NWAuditFileList is not zero, the `fileSize` field is valid.

The file number zero (0) indicates the current history file; file numbers 1-15 indicate old history files. The number of old files to keep is indicated by `bufferSize` in NWADReadConfigHeader and NWADWriteConfigHeader and has a maximum of 15 files.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 23 Return Audit File List
0x2222 104 222 Return Audit File List

See Also

NWADCloseRecordFile (page 78), NWADOpen (page 107), NWADOpenRecordFile (page 109), NWADReadRecord (page 117), NWADResetFile (page 120), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADGetFlags

Returns the auditing flag byte

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADGetFlags  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle,
   pnuint8         flags);
```

Pascal Syntax

```
uses audwin32

Function NWADGetFlags
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr;
   Var flags : nuint8
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing

1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the `auditHandle` allocated by NWADOpen.

flags

(OUT) Points to a byte where the flags can be returned.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8998	VOLUME_DOES_NOT_EXIST

Remarks

NWADGetFlags returns the auditing flag byte, whose contents follow:

```
0x01 = DiscardAuditRcdsOnErrorFlag
0x02 = ConcurrentVolAuditorAccess
0x04 = DualLevelPasswordsActive
0x08 = BroadcastWarningsToAllUsers
0x10 = LevelTwoPasswordSet
0x20 = ArchiveAuditFileOnErrorFlag
```

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 19 Return Audit Flags
0x2222 104 216 Directory Services Return Audit Flags

See Also

NWADLogin (page 102), NWADOpen (page 107), NWADReadConfigHeader (page 114), NWADWriteConfigHeader (page 130), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADGetStatus

Returns the audit information and status of the specified volume or container

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADGetStatus  (
   NWCONN_HANDLE    conn,
   nuint32          auditIDType,
   nuint32          auditID,
   pNWAuditStatus   auditStatus,
   nuint16          bufferSize);
```

Pascal Syntax

```
uses audwin32

Function NWADGetStatus
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditStatus : pNWAuditStatus;
   bufferSize : nuint16
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing

1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditStatus

(OUT) Points to NWAuditStatus containing fields for the information to be returned.

bufferSize

(IN) Specifies the size of the memory space.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8998	VOLUME_DOES_NOT_EXIST
0x89F2	Audit Password Enabled

Remarks

The `historyRecordCount` of the NWAuditStatus structure will remain zero (0) because the history records are kept inside the audit file.

For NetWare 4.11, if NWADGetStatus returns 0x89F2, the user is not allowed auditor access. However, the NWAuditStatus structure will still be filled. You should check the `auditingFlags` field for a value of one (1) which indicates passwords are allowed. If the value is one (1), NWADLogin can then be called with a valid password. Call NWADCheckAccess to set the audit access bit on the server side. Subsequent calls will then be enabled for password users on NetWare 4.11.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 01 Return Volume Audit Status
0x2222 104 200 Directory Services Return Volume Audit Status

See Also

NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADInitLevelTwoPassword

Enables auditor level two access on a specified volume

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADInitLevelTwoPassword  (
   nptr      auditHandle,
   pnuint8   password);
```

Pascal Syntax

```
uses audwin32

Function NWADInitLevelTwoPassword
  (auditHandle : nptr;
   Var password : nuint8
) : NWRCODE;
```

Parameters

auditHandle

(IN) Points to the `auditHandle` allocated by NWADOpen.

password

(IN) Points to a NULL-terminated character string containing the password.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8984	Auditing Not Supported

0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

`auditHandle` is initialized and set up for level two access.

NWADInitLevelTwoPassword does not modify auditing flags and does not verify the password.

Call NWADInitLevelTwoPassword to set up the Directory Service Level Two Password also.

NWADCheckLevelTwoAccess can be called to verify if the password is valid on NetWare 4.1. (NWADCheckLevelTwoAccess is not supported on NetWare 4.11.)

See Also

NWADLogin (page 102), NWADCheckLevelTwoAccess (page 72)

NWADIsObjectAudited

Checks to see if specified object or user is being audited

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADIsObjectAudited  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nuint32         userObjectID);
```

Pascal Syntax

```
uses audwin32

Function NWADIsObjectAudited
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   userObjectID : nuint32
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

userObjectID

(IN) Specifies the object ID to be checked if it is being audited.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL-User is not being audited.
0x0001	SUCCESSFUL-User is being audited.

Remarks

NWADIsObjectAudited returns 0x0000 if the user is not being audited and 0x0001 if the user is being audited.

`userObjectID` must be byte-swapped to the same format in which the server stores it.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 09 Is User Audited
0x2222 104 220 Is Object Audited

See Also

NWADChangeObjectProperty (page 64), NWADLogin (page 102), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADLogin

Enables auditor access on a specified container or volume

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADLogin  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle,
   pnuint8         password);
```

Pascal Syntax

```
uses audwin32

Function NWADLogin
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr;
   Var password : nuint8
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing

1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the `auditHandle` allocated by NWADOpen.

password

(IN) Points to the address of a NULL-terminated character string containing the password.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

Calling NWADLogin is the first step to any auditing function.

`auditHandle` is initialized and setup for future auditing API calls; `auditHandle` must be allocated by the program.

Only a level one password is authenticated with NWADLogin.

If auditing has never been initialized on the Container, call NWADLogin first; AUDITING_NOT_ENABLED will be returned. Then, call NWADEnable.

If the user is not SUPERVISOR equivalent, NWADEnable will fail the first time it is called.

After calling NWADEnable successfully, the user must log in again by calling NWADLogin to have access to auditing.

Once auditing has been enabled, the user does not have to be SUPERVISOR equivalent, but must know the auditor password.

NetWare 4.11 does not use a password unless a password has been set by calling NWADSetPassword. Call NWADGetStatus to determine if a password has been set.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 03 Add Auditor Access
0x2222 88 19 Get Auditing Flags
0x2222 104 202 Directory Services Add Auditor Access
0x2222 104 216 Get Auditing Flags

See Also

NWADCheckAccess (page 70), NWADEnable (page 86), NWADGetStatus (page 95), NWADInitLevelTwoPassword (page 98), NWADLogout (page 105), NWADOpen (page 107), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADLogout

Removes auditor access from a volume or container while `auditHandle` resets to NULL

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADLogout  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle);
```

Pascal Syntax

```
uses audwin32

Function NWADLogout
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the auditHandle allocated by NWADOpen.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWADLogout must be called every time an application logs in as an auditor.

NWADLogout must be called every time an application logs in as an auditor on the NDS container.

NWADLogout cannot be used on NetWare 4.11 unless a password has been set by calling NWADSetPassword. Call NWADGetStatus to determine if a password has been set.

If auditIDType is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If auditIDType is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 13 Remove Auditor Access
0x2222 104 211 Directory Services Remove Auditor Access

See Also

NWADCheckAccess (page 70), NWADGetStatus (page 95), NWADLogin (page 102), NWADOpen (page 107), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADOpen

Allocates `auditHandle` for use in other Auditing functions

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE NWADOpen (
   NWCONN_HANDLE     conn,
   nuint32           auditIDType,
   nuint32           auditID,
   pnptr             auditHandle,
   pNWADOpenStatus   openStatus);
```

Pascal Syntax

```
uses audwin32

Function NWADOpen
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   Var auditHandle : nptr;
   openStatus : pNWADOpenStatus
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the auditID parameter.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the volume number for volume auditing or the object ID of the DS Audit File Object for container auditing.

auditHandle

(OUT) Points to the `auditHandle` allocated by NWADOpen.

openStatus

(OUT) Points to the NWADOpenStatus structure containing the file status.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWGetVolumeNumber can be called to query the volume number.

NWDSAuditGetObjectID can be called to query the object ID.

See Also

NWADClose (page 75), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADOpenRecordFile

Opens the record file

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADOpenRecordFile  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle,
   nint16          fileCode,
   pnptr           recordHandle);
```

Pascal Syntax

```
uses audwin32

Function NWADOpenRecordFile
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr;
   fileCode : nint16;
   Var recordHandle : nptr
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the `auditHandle` allocated by NWADOpen.

fileCode

(IN) Specifies the file number to open. -1 specifies the current file from which to read records while 0-15 specifies old files from which to read records.

recordHandle

(OUT) Points to an allocated record handle.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 24 Init Audit File Read
0x2222 104 223 Init Audit File Read

See Also

NWADCloseRecordFile (page 78), NWADOpen (page 107), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADReadBitMap

Reads the audit bitmap to see what is being audited

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADReadBitMap  (
   NWCONN_HANDLE         conn,
   nuint32               volumeID,
   NWAuditBitMap N_FAR  *buffer,
   nuint16               bufferSize);
```

Pascal Syntax

```
uses audwin32

Function NWADReadBitMap
  (conn : NWCONN_HANDLE;
   auditID : nuint32;
   Var buffer : NWAuditBitMap;
   bufferSize : nuint16
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditID

(IN) Specifies the number of the audited volume.

buffer

(OUT) Points to NWAuditBitMap defining what is being audited.

bufferSize

(IN) Specifies the size of NWAuditBitMap expected. The entire bitmap needs to be received at once.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWAuditMap is a 512-bit stream. If a bit is set to one (1), the corresponding item found in auditBitMapIDs enumeration is being audited.

NWAuditBitMap may also be read by calling NWReadAuditConfigHeader.

When a bit in auditBitMapID is set to one, it generates an event and saves it in the audit file. The following table defines the set bits:

Bit	Function	Description
A_BIT_GOPEN_FILE	Global Open	Specifies all file opens are audited.
A_BIT_IOPEN_FILE	Intersection Open	Specifies the intersection of events by a user or a file open is audited.
A_BIT_UOPEN_FILE	Union Open	Specifies the union of events by a user and a file open is audited.

auditBitMapID lists the bits defined by the event bitmap in the audit file configuration header. The first 32 bits are reserved for NDS for container auditing. For a list of event bits, see “Event Bits Tables” on page 54.

NCP Calls

0x2222 88 10 Read Audit Bit Map

See Also

NWADReadConfigHeader (page 114), NWADWriteBitMap (page 128), NWADLogin (page 102)

NWADReadConfigHeader

Returns the configuration header from the auditing file on a specified volume or container

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADReadConfigHeader  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle,
   nptr            buffer,
   nuint16         bufferSize);
```

Pascal Syntax

```
uses audwin32

Function NWADReadConfigHeader
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr;
   buffer : nptr;
   bufferSize : nuint16
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the `auditHandle` allocated by NWADOpen.

buffer

(OUT) Points to an array in which data is saved.

bufferSize

(IN) Specifies the size of the configuration header for saving data.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

`volumeAuditEventBitMap` in NWConfigHeader may also be read by calling NWReadAuditingBitMap.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

For a list of event bits, see "Event Bits Tables" on page 54.

NCP Calls

0x2222 88 11 Read Audit Config Hdr
0x2222 104 209 Directory Services Read Audit Configuration Header

See Also

NWADLogin (page 102), NWADReadBitMap (page 112), NWADWriteConfigHeader (page 130), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADReadRecord

Reads a specified record

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADReadRecord  (
   nptr       recordHandle,
   nuint16    maxSize,
   nint16     direction,
   pnuint8    buffer,
   pnuint16   bufferSize,
   pnuint8    eofFlag,
   pnuint32   offsetPtr);
```

Pascal Syntax

```
uses audwin32

Function NWADReadRecord
  (recordHandle : nptr;
   maxSize : nuint16;
   direction : nint16;
   Var buffer : nuint8;
   Var bufferSize : nuint16;
   Var eofFlag : nuint8;
   Var offsetPtr : nuint32
) : NWRCODE;
```

Parameters

recordHandle

(IN) Specifies the record handle allocated in NWADOpenRecordFile.

maxSize

(IN) Specifies the size of the buffer passed into the call. NWADReadRecord will write beyond the end of the specified buffer size. Typically, size is 512 bytes.

direction

(IN) Specifies whether to get the previous or the next record:

-1 = Get previous record
1 = Get next record

buffer

(IN/OUT) Points to a buffer to contain the record.

bufferSize

(OUT) Points to the size of data contained in the buffer.

eofFlag

(OUT) Points to a flag indicating whether the end of the file has been reached:

1 = End of file
0 = More file to be read

offsetPtr

(IN) Points to a book marker indicating where the previously read record is located.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

Either an end of file flag or -1 is a valid signal to NWADReadRecord to stop reading from the specified file.

NCP Calls

0x2222 88 25 Read Auditing File

0x2222 104 224 Read Auditing File

See Also

NWADOpen (page 107), NWADOpenRecordFile (page 109), NWADCloseRecordFile (page 78)

NWADResetFile

Resets the audit file on a specified container or volume

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADResetFile  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID,
   nptr            auditHandle);
```

Pascal Syntax

```
uses audwin32

Function NWADResetFile
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the auditHandle allocated by NWADOpen.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

The original file is saved to the OLD file and a new audit file is set up to store audit events.

NWADResetFile requires a level two password when enabled.

The name of the data file is containerID.DAF. When the file is reset, it is renamed to containID.OAF. These files are hidden, and reside in a hidden directory called _NetWare on a NetWare server having a writable replica of the container probably where the object resides.

NWADResetFile is only supported on NetWare 4.1; NetWare 4.11 does not support NWADResetFile.

If auditIDType is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If auditIDType is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 14 Reset Audit File
0x2222 104 212 Directory Services Reset Audit File

See Also

NWADInitLevelTwoPassword (page 98), NWADLogin (page 102), NWADOpen (page 107), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADRestartVolumeAuditing

Restarts auditing for volumes only

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADRestartVolumeAuditing  (
   NWCONN_HANDLE   conn,
   nuint32         auditIDType,
   nuint32         auditID);
```

Pascal Syntax

```
uses audwin32

Function NWADRestartVolumeAuditing
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing
1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL

Remarks

NWADRestartVolumeAuditing is for NetWare 4.11 only and is not supported on NetWare 4.1.

NWADRestartVolumeAuditing should be called when the history file has reached the size limit or a volume is full. The auditor (not a password auditor) should access the volume and correct the situation, and then call NWADRestartVolumeAuditing to restart auditing on the volume. Other users may then access the volume.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 30 Restart Volume Auditing

See Also

NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADSetPassword

Sets a password only on NetWare 4.11

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADSetPassword
  (NWCONN_HANDLE  conn,
   nuint32        auditIDType,
   nuint32        auditID,
   nptr           auditHandle,
   pnuint8        newPassword);
```

Pascal Syntax

```
uses audwin32

Function NWADSetPassword
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr;
   Var newPassword : nuint8
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing

1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the `auditHandle` allocated by NWADOpen.

newPassword

(IN) Points to the new password to be set.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x89D7	PASSWORD_NOT_UNIQUE
0x89D8	PASSWORD_TOO_SHORT
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWADSetPassword is only supported on the next released version after NetWare 4.1.

NWADSetPassword allows auditing to be accessed by a password user who does not have Audit File Object access rights.

Password users have limited access in error conditions and setting up auditing.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 19 Get Audit Flags
0x2222 88 31 Set Audit Password
0x2222 104 216 Get Audit Flags

0x2222 104 229 Set Audit Password

See Also

NWADChangePassword (page 67), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWADWriteBitMap

Writes the audit bitmap definition of what is being audited

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY (NWRCODE) NWADWriteBitMap  (
   NWCONN_HANDLE          conn,
   nuint32                volumeID,
   nptr                   auditHandle,
   NWAuditBitMap  N_FAR   *buffer);
```

Pascal Syntax

```
uses audwin32

Function NWADWriteBitMap
  (conn : NWCONN_HANDLE;
   auditID : nuint32;
   auditHandle : nptr;
   Var buffer : NWAuditBitMap
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditID

(IN) Specifies the number of the audited volume.

auditHandle

(IN) Points to the `auditHandle` allocated by NWADOpen.

buffer

(IN) Points to NWAuditBitMap defining what is being audited.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWADWriteBitMap requires a second level password when enabled.

This constant defines the number of bits in the event bitmap for volumes:

```
NW_AUDIT_NUMBER_EVENT_BITS = 512
```

The buffer is a 512-bit stream. If a bit is set to one (1), the corresponding item found in auditBitMapIDs (nwaudit.h) is being audited.For a list of event bits, see “Event Bits Tables” on page 54.

If auditing is not enabled, NO_DISK_LEFT_FOR_SPOOL_FILE is returned; if auditing is not supported, NO_CREATE_PRIVILEGES is returned.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 16 Write Auditing Bit Map

See Also

NWADLogin (page 102), NWADOpen (page 107), NWADWriteConfigHeader (page 130), NWADReadBitMap (page 112), NWADReadConfigHeader (page 114), NWADInitLevelTwoPassword (page 98)

NWADWriteConfigHeader

Saves the configuration header for the auditing file on a specified volume or container

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

N_EXTERN_LIBRARY NWRCODE  NWADWriteConfigHeader  (
   NWCONN_HANDLE     conn,
   nuint32           auditIDType,
   nuint32           auditID,
   nptr              auditHandle,
   pNWConfigHeader   buffer);
```

Pascal Syntax

```
uses audwin32

Function NWADWriteConfigHeader
  (conn : NWCONN_HANDLE;
   auditIDType : nuint32;
   auditID : nuint32;
   auditHandle : nptr;
   buffer : pNWConfigHeader
) : NWRCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

auditIDType

(IN) Specifies the type of the object to be audited.

0 AUDIT_ID_IS_VOLUME indicates volume auditing

1 AUDIT_ID_IS_CONTAINER indicates container auditing

auditID

(IN) Specifies the identification of the object to be audited.

auditHandle

(IN) Points to the `auditHandle` allocated by NWADOpen.

buffer

(IN) Points to NWConfigHeader containing data.

Return Values

These are common return values; see Return Values (*Return Values for C*) for more information.

0x0000	SUCCESSFUL
0x8801	INVALID_CONNECTION
0x890A	NLM_INVALID_CONNECTION
0x8983	Auditing Hardware Error
0x8984	Auditing Not Supported
0x8997	Auditing Not Enabled
0x8998	VOLUME_DOES_NOT_EXIST
0x89DE	PASSWORD_HAS_EXPIRED_NO_GRACE

Remarks

NWADWriteConfigHeader requires a second level password when enabled.

If auditing is not enabled, NO_DISK_LEFT_FOR_SPOOL_FILE is returned; if auditing is not supported, NO_CREATE_PRIVILEGES is returned.

If `auditIDType` is set to AUDIT_ID_IS_VOLUME to indicate volume auditing, NWGetVolumeNumber can be called to get the volume number of the audit file object.

If `auditIDType` is set to AUDIT_ID_IS_CONTAINER to indicate container auditing, NWDSAuditGetObjectID can be called to get the Directory Service object ID of the audit file object.

NCP Calls

0x2222 88 17 Write Audit Config Hdr
0x2222 88 19 Get Auditing Flags
0x2222 104 214 Directory Services Write Audit CNT Config Hdr
0x2222 104 216 Get Auditing Flags

See Also

NWADInitLevelTwoPassword (page 98), NWADLogin (page 102), NWADReadBitMap (page 112), NWADReadConfigHeader (page 114), NWADWriteBitMap (page 128), NWDSAuditGetObjectID (obsolete 06/03) (NDS Core Services), NWGetVolumeNumber (Volume Management)

NWGetNWADVersion

Returns the version information about the NWAD library

NetWare Server: 4.1 and above

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Auditing (AUD*.*)

Service: Auditing

Syntax

```
#include <nwaudit.h>
or
#include <nwnet.h>

void N_API  NWGetNWADVersion  (
   pnuint8   majorVersion,
   pnuint8   minorVersion,
   pnuint8   revisionLevel,
   pnuint8   betaReleaseLevel);
```

Pascal Syntax

```
uses audwin32

Procedure NWGetNWADVersion
  (Var majorVersion : nuint8;
   Var minorVersion : nuint8;
   Var revisionLevel : nuint8;
   Var betaReleaseLevel : nuint8
);
```

Parameters

majorVersion

(OUT) Points to the major version of the library. For example, 4 would be returned for NetWare 4.1.

minorVersion

(OUT) Points to the minor version of the library. For example, .1 would be returned for NetWare 4.1.

revisionLevel

(OUT) Points to the revision version of the library. For example, c would be returned for NetWare 3.11c.

betalReleaseLevel

(OUT) Points to the beta release version of the library.

Return Values

None

Auditing Structures

7

This documentation alphabetically lists the Auditing structures and describes their purpose syntax, and fields.

NWADOpenStatus

Contains the audit status information for a specified container or volume

Service: Auditing

Defined In: nwaudit.h

Structure

```
typedef struct
{
   nuint32   auditingStatus ;
   nuint32   isTrustedNetWare ;
   nuint32   trustedNetWareStatus ;
   nuint32   reserved1 ;
   nuint32   reserved2 ;
   nuint32   reserved3 ;
   nuint32   reserved4 ;
} NWADOpenStatus;
```

Pascal Structure

```
uses audwin32

NWADOpenStatus  = packed Record
    auditingStatus : nuint32;
    isTrustedNetWare : nuint32;
    trustedNetWareStatus : nuint32;
    reserved1 : nuint32;
    reserved2 : nuint32;
    reserved3 : nuint32;
    reserved4 : nuint32
  End;
```

Fields

`auditingStatus`

Specifies whether auditing has been enabled:

TRUE Auditing is enabled
FALSE Auditing is not enabled

`isTrustedNetWare`

Specifies if NetWare 4.11 or 5.0 are running:

0 FALSE
1 TRUE

trustedNetWareStatus

Specifies the following bit field:

0	Passwords Allowed Field: If the bit is 1, the server has the set parameter Allow Audit Passwords set to ON.
1	**For Containers** Audit File Full Field: If the bit is 1, the audit file is full. Reset the audit file and the auditing system will automatically put audit records into the new file. **For Volumes** Volume Restart Flag Field: If the bit is 1, the volume switch table has been replaced with the audit volume switch table. Reset the audit file and call NWADRestartVolumeAuditing (page 123) to allow the auditing system to automatically put audit records into the new file. All other bits are reserved.

reserved1

Reserved by Novell for future use.

reserved2

Reserved by Novell for future use.

reserved3

Reserved by Novell for future use.

reserved4

Reserved by Novell for future use.

NWAuditBitMap

Defines an array of bytes large enough to contain the number of bits in a configuration header's event bitmap

Service: Auditing

Defined In: nwaudit.h

Structure

```
typedef struct
{
   nuint8   bitMap [NW_AUDIT_NUMBER_EVENT_BITS/8];
} NWAuditBitMap;
```

Pascal Structure

```
uses audwin32

 NWAuditBitMap = packed Record
      bitMap : Array[0..31] Of nuint8
   End;
```

Fields

bitMap

Specifies if the event is audited as defined in the nwadevnt.h file:

0 Event is not audited
1 Event is audited

NWAuditFileList

Service: Auditing

Defined In: nwaudit.h

Structure

```
typedef struct
{
   nuint32   fileCreateDateTime [16];
   nuint32   fileSize [16];
} NWAuditFileList;
```

Pascal Structure

```
uses audwin32

 NWAuditFileList = packed Record
      fileCreateDateTime : Array[0..15] Of nuint32;
      fileSize : Array[0..15] Of nuint32
   End;
```

Fields

`fileCreateDateTime`

Specifies the time the file was saved by calling the NWADResetAuditFile function.

`fileSize`

Specifies the size of the file.

NWAuditRecord

Service: Auditing

Defined In: nwaudit.h

Structure

```
typedef struct
{
   nuint32   recordLength ;
   pnuint8   record ;
} NWAuditRecord;
```

Pascal Structure

```
uses audwin32

 NWAuditRecord = packed Record
    recordLength : nuint32;
    aRecord : pnuint8
  End;
```

Fields

recordLength

Specifies the size of the data to send.

record

Points to the data to send.

NWAuditStatus

Service: Auditing

Defined In: nwaudit.h

Structure

```
typedef struct
{
   nuint16   auditingVersionDate ;
   nuint16   auditFileVersionDate ;
   nuint32   auditingEnabledFlag ;
   nuint32   auditFileSize ;
   nuint32   modifiedCounter ;
   nuint32   auditFileMaxSize ;
   nuint32   auditFileSizeThreshold ;
   nuint32   auditRecordCount ;
   nuint32   auditingFlags ;
} NWAuditStatus;
```

Pascal Structure

```
uses audwin32

NWAuditStatus = packed Record
    auditingVersionDate : nuint16;
    auditFileVersionDate : nuint16;
    auditingEnabledFlag : nuint32;
    auditFileSize : nuint32;
    modifiedCounter : nuint32;
    auditFileMaxSize : nuint32;
    auditFileSizeThreshold : nuint32;
    auditRecordCount : nuint32;
    auditingFlags : nuint32
 End;
```

Fields

auditingVersionDate

Specifies the server version data of auditing.

auditFileVersionDate

auditingEnabledFlag

Specifies whether auditing is enabled:

0 Disabled
1 Enabled

auditFileSize

Specifies the size of the current audit file.

modifiedCounter

Specifies the number of times auditing has been modified.

auditFileMaxSize

Specifies the maximum size for the audit file.

auditFileSizeThreshold

Specifies the size of the file when auditing starts to send warning messages.

auditRecordCount

Specifies the total number of events in the audit file.

auditingFlags

NWConfigHeader

Stores information associated with a volume's audit file configuration header

Service: Auditing

Defined In: nwaudit.h

Structure

```
typedef struct
{
   nuint16         fileVersionDate ;
   nuint8          auditFlags ;
   nuint8          errMsgDelayMinutes ;
   nuint8          reserved1 [16];
   nuint32         auditFileMaxSize ;
   nuint32         auditFileSizeThreshold ;
   nuint32         auditRecordCount ;
   nuint32         historyRecordCount ;
   nuint8          reserved2 [16];
   nuint32         reserved3 [3];
   nuint8          auditEventbitMap [NW_AUDIT_NUMBER_EVENT_BITS/8];
   nuint32         auditFileCreationDateTime ;
   nuint8          reserved4 [8];
   nuint16         auditFlags2 ;
   nuint16         fileVersionDate2 ;
   nuint8          fileArchiveDays ;
   nuint8          fileArchiveHour ;
   nuint8          numOldAuditFilesToKeep ;
   nuint8          reserved5 ;
   nuint32         headerChecksum ;
   nuint32         headerModifiedCounter ;
   nuint32         reserved6 ;
   nuint8          newbitMap [64];
   nuint8          reserved7 [64];
} NWConfigHeader;
```

Pascal Structure

```
uses audwin32

NWConfigHeader = packed Record
      fileVersionDate : nuint16;
      auditFlags : nuint8;
      errMsgDelayMinutes : nuint8;
      reserved1 : Array[0..15] Of nuint8;
      auditFileMaxSize : nuint32;
      auditFileSizeThreshold : nuint32;
      auditRecordCount : nuint32;
      historyRecordCount : nuint32;
      reserved2 : Array[0..15] Of nuint8;
```

```
      reserved3 : Array[0..2] Of nuint32;
      auditEventBitMap : Array[0..31] Of nuint8;
      auditFileCreationDateTime : nuint32;
      reserved4 : Array[0..7] Of nuint8;
      auditFlags2 : nuint16;
      fileVersionDate2 : nuint16;
      fileArchiveDays : nuint8;
      fileArchiveHour : nuint8;
      numOldAuditFilesToKeep : nuint8;
      reserved5 : nuint8;
      headerChecksum : nuint32;
      headerModifiedCounter : nuint32;
      (* Trusted NetWare uses the following two fields *)
      reserved6 : nuint32;
      (*Trusted NetWare uses this bitmap instead of
         volumeAuditEventBitMap above       *)
      newBitMap : Array[0..63] Of nuint8;
      reserved7 : Array[0..63] Of nuint8
   End;
```

Fields

fileVersionDate

Specifies the current version of the audit file.

auditFlags

Specifies the set of bit flags controlling the audit.

errMsgDelayMinutes

Specifies the number of minutes to delay between error messages.

reserved1

Specifies the maximum allowable size of the audit file.

auditFileMaxSize

Specifies the maximum allowable size of the audit file.

auditFileSizeThreshold

Specifies the size at which users are notified that the audit file is approaching its maximum size.

auditRecordCount

Specifies the number of records in the audit file.

historyRecordCount

Specifies the number of records in the audit history file (found only in the volume configuration).

reserved2

Reserved by Novell for future use.

reserved3

Reserved by Novell for future use.

auditEventBitMap

auditFileCreationDateTime

reserved4

Reserved by Novell for future use.

auditFlags2

fileVersionDate2

fileArchiveDays

fileArchiveHour

numOldAuditFilesToKeep

reserved5

Reserved by Novell for future use.

headerChecksum

headerModifiedCounter

reserved6

Reserved by Novell for future use.

newBitMap

Specifies Trusted NetWare uses this bitmap instead of `auditEventBitMap`.

reserved7

Trusted NetWare uses this field.

TIMESTAMP

Stores a NDS™ time stamp and holds the time stamp for the container object

Service: Auditing

Defined In: nwaudit.h

Structure

```
typedef struct
{
   nuint32   seconds ;
   nuint16   replicaNumber ;
   nuint16   event ;
} TIMESTAMP;
```

Pascal Structure

```
uses audwin32

 TIMESTAMP = packed Record
    seconds : nuint32;
    replicaNumber : nuint16;
    event : nuint16
  End;
```

Fields

seconds

replicaNumber

event

Name Retrieval Functions

8

This documentation provides an alphabetical listing of the name retrieval functions and describes their purpose, syntax, parameters, and return values.

These functions can be used to successfully retrieve names of trees or servers over either IP or IPX.

NWGetObjectNamesBeginA

Sets up an environment so that NWGetObjectNamesNextA can retrieve the names of trees or servers

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT*, Windows* 95, Windows 98

Service: Name Retrieval

Syntax

```
#include <nwbindry.h)

NWCCODE  NWGetObjectNamesBeginA (
   nuint32    luObjectType,
   pnuint32   pluHandle);
```

Pascal Syntax

```
uses calwin32

Function NWGetObjectNamesBeginA(
   luObjectType : nuint32;
   Var pluHandle : nuint32
) : NWCCODE; stdcall;
```

Parameters

`luObjectType`

(IN) Specifies which type of object names are to be retrieved.

`pluHandle`

(OUT) Points to a value used by NWGetObjectNamesNextA and NWGetObjectNamesEndA, which should never be changed directly.

Return Values

Returns zero on success. The most common return values on failure include the following:

0x8836	NWE_PARAM_INVALID
0x88FF	NWE_REQUESTER_FAILURE
0x881A	NWE_OUT_OF_HEAP_SPACE

Remarks

NWGetObjectNamesBeginA internally calls WinSock2 name space functions to retrieve tree or server names from SAP and SLP name spaces. It also allocates a buffer to store retrieved names as a sorted and ordered list.

The `luObjectType` parameter can have the following values:

0x0400	OT_FILE_SERVER
0x7802	OT_TREE_NAME

See Also

NWGetObjectNamesNextA (page 150), NWGetObjectNamesEndA (page 152)

NWGetObjectNamesNextA

Retrieves tree or server names as ASCII strings

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT*, Windows* 95, Windows 98

Service: Name Retrieval

Syntax

```
#include <nwbindry.h)

NWCCODE  NWGetObjectNamesNextA (
   nuint32    luHandle,
   pnuint32   pluLenBuffer,
   pstr8      strBuffer);
```

Pascal Syntax

```
uses calwin32

Function NWGetObjectNamesNextA(
   luHandle : nuint32;
   Var pluLenBuffer : nuint32;
   strBuffer : pnstr8
) : NWCCODE; stdcall;
```

Parameters

`luHandle`

(IN) Specifies the handle returned from NWGetObjectNamesBeginA.

`pluLenBuffer`

(IN/OUT) Points to the size of the buffer `strBuffer` passed to the function call. If the function returns NWE_BUFFER_OVERFLOW then this parameter holds the minimum buffer size for this call to be successful.

`strBuffer`

(OUT) Returns the name of a tree or server.

Return Values

Returns zero on success. The most common return values on failure include the following:

0x8866	NO_MORE_ENTRIES

0x88FF	NWE_REQUESTER_FAILURE
0x880E	NWE_BUFFER_OVERFLOW

Remarks

NWGetObjectNamesNextA iteratively returns alphabetically ordered names in ASCII format, one name at a time, until NO_MORE_ENTRIES is returned.

NWGetObjectNamesBeginA must be called before you call NWGetObjectNamesNextA the first time.

The `luObjectType` parameter of NWGetObjectNamesBeginA determines whether NWGetObjectNamesNextA returns tree names or server names.

See Also

NWGetObjectNamesBeginA (page 148), NWGetObjectNamesEndA (page 152)

NWGetObjectNamesEndA

Frees resources allocated by NWGetObjectNamesBeginA

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.11, 3.12, 3.2, 4.x, 5.x, 6.x

Platform: NLM, Windows NT*, Windows* 95, Windows 98

Service: Name Retrieval

Syntax

```
#include <nwbindry.h)

NWCCODE  NWGetObjectNamesEndA (
   nuint32   luHandle);
```

Pascal Syntax

```
uses calwin32

Function NWGetObjectNamesEndA(
   luHandle : nuint32
) : NWCCODE; stdcall;
```

Parameters

luHandle

(IN) Specifies the handle returned from NWGetObjectNamesBeginA.

Return Values

Returns zero on success. The most common return values on failure include the following:

0x88FF	NWE_REQUESTER_FAILURE
0x8836	NWE_PARAM_INVALID

Remarks

You must call NWGetObjectNamesEndA to free resources allocated by successfully calling NWGetObjectNamesBeginA.

See Also

NWGetObjectNamesBeginA (page 148), NWGetObjectNamesNextA (page 150)

Server-Based Auditing Concepts

9

This documentation describes Server-Based Auditing, its functions, and features.

IMPORTANT: The functions described in this documentation were designed for server-centric auditing. For information on auditing that is cross-platform and NDS enabled, see Chapter 4, "Auditing Concepts," on page 45.

The Auditing capabilities built into NetWare® 4.x allow your NLM to meet the Class F2 security criteria specified in the Information Technology (IT) security standards. These standards are written and maintained by the German Information Security Agency (GISA).

In addition to meeting Class F2 security criteria, you can use the Auditing API to provide audit trails for virtually any type of action the NLM performs for an unlimited array of purposes. You can use the audit trail to

- Let system administrators see the pattern of usage of your product
- See how its features are being used
- Debug your product by identifying which of its users a thread is doing work for

The NetWare 4.x OS uses Auditing as a replacement for the system error log. Auditing gives more flexibility since users can select only the audits they are interested in tracking.

In the future, developers can create utilities like Auditcon for various NetWare client platforms. When they do, your NLM will be equipped to work with them.

9.1 Auditcon

From the viewpoint of GISA, an *auditor* is a person who checks the accuracy of network transactions in the same way that a bank auditor verifies financial transactions. Auditing tracks *events*-file, queue, server, and user transactions-and enters a *record* of each event in an *audit file*. To allow the work of a GISA auditor, Novell has written a workstation utility, *Auditcon*, which can extract and display or print audit file information for NetWare events, but only for NetWare events.

NOTE: For Auditing to begin adding records from your NLM, the Auditcon option NLM Add Audit Record has to be enabled.

Some examples of the events NetWare tracks are listed in the following table.

File Events	User Events	Server Events	QMS Events
Opens	Logins	Changing date/time	Queue creations
Closes	Logouts	Opening the Bindery	Job creations
Reads	Password changes	Downing the Server	Job edits
Writes	Modifications of trustee rights	Adding NLM Audit records to audit file	Job service start-up

File Events	User Events	Server Events	QMS Events
Salvages	Grantings of trustee rights	Closing the Bindery	Job service removal
Rename/moves			Setting job priority
Deletes			
Modifications of directory entries			

9.2 Potential Uses

Because any NLM event can be logged, the potential uses of Auditing Services are fairly extensive. In particular, any NLM that provides a single service, like a fax gateway, could use it to keep a record of the services rendered. Or a specialized database NLM, such as one that debits and credits accounts, could use Auditing to track not just events but which users the events originated from.

9.2.1 Fax Gateway

A fax gateway could track at least the following information about each fax it sends:

- Date and time sent
- ID and name of sender
- ID of sending workstation
- Telephone number of recipient
- Receipt: successful or unsuccessful?

Auditcon can display only information about NetWare events. However, you could create a similar utility so users could review and print out the information your application enters into the audit file. For example, your utility might provide the following report:

```
On May 25, 1992, user Kay Billings sent a fax to
(801) 429-3232. It arrived successfully.
```

9.3 Debit/Credit Database

Auditing could track account balances so that those in the red could be extracted with Auditcon. It could also track the specific user that changed a record in the database, for security or efficiency purposes.

9.3.1 Identifying a specific user

If you spawn a thread on behalf of a user, you can make it seem that the thread was a given client. In this way, you can know which client your NLM is actually doing work for.

Thread groups allow you to give one or more threads a single context. At the thread group level, context includes the following three identifiers:

- Current working directory

- Current screen
- Current connection number

The connection number can be set to that of a certain client workstation that has already been authenticated. Then, if the connection number in a request packet does not match the one you set, the request fails.

An extensively commented example, XAUDIT.C, shows how to link the current thread group with a workstation address and user name so that Auditcon displays this information for you. XAUDIT.C is in the EXAMPLES directory.

9.4 Server-Based Auditing Functions

NWAddRecordToAuditingFile	Add records to the audit file
NWSetAuditingIdentity	Set an Auditing Identity for a thread group
NWGetAuditingIdentity	Gets the Auditing Identity of a particular thread group

Server-Based Auditing Functions

10

This documentation provides an alphabetical listing of the server-based auditing functions and describes their purpose, syntax, parameters, and return values.

IMPORTANT: The functions described in this documentation were designed for server-centric auditing. For information on auditing that is cross-platform and NDS enabled, see Chapter 4, "Auditing Concepts," on page 45.

NWAddRecordToAuditingFile

Allows NLM applications to add records to the audit file.

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: Server-Based Auditing

Syntax

```
#include <nlm\nwtypes.h>
#include <nwaudnlm.h>

int  NWAddRecordToAuditingFile (
   LONG    volumeNumber,
   LONG    recordType,
   LONG    stationNumber,
   LONG    statusCode,
   BYTE   *data,
   LONG    dataSize);
```

Parameters

volumeNumber

(IN) Specifies the volume for which to record the related record.

recordType

(IN) Specifies the type of record to be added to the audit file. Types lower than 65,536 are defined by the NLM™ application. Types greater than 65,536 are reserved.

stationNumber

(IN) Specifies the server connection number.

statusCode

(IN) Specifies the code to be written into the record.

data

(IN) Points to the data to be written.

dataSize

(IN) Specifies the size of the data to be written.

Return Values

Decimal	Hex	Constant
0	(0x00)	Successful
151	(0x97)	ERR_AUDITING_NOT_ENABLED
168	(0xA8)	ERR_AUDITING_NO_RIGHTS

Remarks

Before calling the NWAddRecordToAuditingFile function, use the Auditcon utility to set "NLM Add Audit Record" to "Audited" (in "Audit by server events").

Retrieve the record with the Auditcon utility or a client application.

NOTE: To avoid conflicts between applications, developers should contact Developer Support (1-800-REDWORD) when assigning a record type.

See Also

NWGetAuditingIdentity (page 160), NWSetAuditingIdentity (page 162)

NWGetAuditingIdentity

Assigns an auditing identity to a thread group

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: Server-Based Auditing

Syntax

```
#include #include <nlm\nwtypes.h>
#include <nwaudnlm.h>

int  NWGetAuditingIdentity (
   LONG   *addressType,
   BYTE   *networkAddress,
   char   *identityName);
```

Parameters

addressType

(OUT) One of the following currently supported address types:
ASCII_STRING_NET_ADDRESS_TYPE 0x00 (string)
ETHERNET_NET_ADDRESS_TYPE 0x02.

networkAddress

(OUT) Points to a buffer that is to receive a network address of the same type received in the address type parameter.

identityName

(OUT) Points to a buffer that is to receive an ASCIIZ string of the name of the identity being audited for the current thread group (this string can be up to 255 characters long). If this parameter is not set, it receives IDENTITY_HAS_NOT_BEEN_SET (0xFF).

Return Values

Decimal	Hex	Constant
0		Successful
255	(0xFF)	ERR_FAILURE

Remarks

The NWGetAuditingIdentity function returns the auditing identity that has been assigned to a thread group. If no auditing identity has been previously set, ERR_FAILURE is returned and `networkAddress` and `identityName` are NULL-length.

See Also

NWSetAuditingIdentity (page 162)

NWSetAuditingIdentity

Links the current thread group to a workstation address and a user name

Local Servers: blocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: Server-Based Auditing

Syntax

```
#include <nlm\nwtypes.h>
#include <nwaudnlm.h>

int  NWSetAuditingIdentity (
   LONG    addressType,
   BYTE   *networkAddress,
   char   *identityName);
```

Parameters

addressType

(IN) One of the following currently supported address types:
ASCII_STRING_NET_ADDRESS_TYPE 0x00 (string)
ETHERNET_NET_ADDRESS_TYPE 0x00 xxxxxxxx:xxxxxxxxxx

networkAddress

(IN) Points to a buffer that contains the network address of an auditing identity. If this pointer is NULL, the auditing identity for the current thread group is cleared.

identityName

(IN) Points to a buffer that contains the name of an auditing identity (up to 255 characters long). If this pointer is NULL, the auditing identity for the current thread group is cleared.

Return Values

Decimal	Hex	Constant
0	(0x00)	Successful
150	(0x96)	ERR_SERVER_OUT_OF_MEMORY
151	(0x97)	ERR_AUDITING_NOT_ENABLED
168	(0xA8)	ERR_AUDITING_NO_RIGHTS

Remarks

Te NWSetAuditingIdentitys function links the current thread group to a workstation address and a user name. The F2 audit specification for security auditing requires that the audit trail events identify the user name and the workstation address of the event initiator. NLM developers can use this function to ensure F2 auditing compliance. The auditing identity is tracked on a thread group basis; therefore, all threads in a thread group are audited as the single identity set by this function.

See Also

NWGetAuditingIdentity (page 160)

Bindery-Based Accounting Concepts

11

This documentation describes Bindery-Based Accounting, its functions, and features.

IMPORTANT: The Accounting functions in this set were designed for use on a NetWare® 3.x bindery, and work only with bindery emulation set on NetWare 4.x, 5.x, and 6.x servers. For Accounting that is cross-platform and NDS enabled, see Chapter 1, “Accounting Concepts,” on page 13.

This introduction assumes familiar with the bindery-related terms "object," "property," and "value." For definitions of these terms, see Bindery Concepts (Bindery Management).

These Accounting functions enable developers to create servers that can charge for their services. For example, a database server can charge for the number of records viewed, the number of requests serviced, or the amount of connection time. A print server can charge for the number of pages printed.

Each value-added server determines its own rates to charge for each type of service, and the server bindery stores the list of authorized accounting servers and each clients accounting information.

Once a server is listed as an authorized accounting server and has logged in, it can submit a charge to a clients account, place a hold against a clients account, or query a clients account status. An audit trail of all charges is accumulated in an audit file.

11.1 Accounting Audit File

In addition to monitoring a network by charging, querying, or holding a clients account, Accounting Services offers a method for monitoring network resources by keeping an audit trail of all charges. The two NetWare API functions, SubmitAccountCharge and SubmitAccountNote, and the accounting audit file (NET$ACCT.DAT) are tools used to keep the audit trail.

The NET$ACCT.DAT audit file has normal attributes and resides in the SYS:SYSTEM directory. It can be read by utility programs and can be deleted or renamed with standard DOS commands.

When the SubmitAccountCharge and SubmitAccountNote functions are called, a detailed record of all accounting charges and activities is kept in the NET$ACCT.DAT audit file. For example, when a server submits a charge or note, a Charge Record or a Note Record is added to the NET$ACCT.DAT audit file.

11.2 Charge and Note Records

The Charge Record and Note Record are distinguished by the contents of their Record Type fields. When the server charges an account through SubmitAccountCharge, a Charge Record is added to the NET$ACCT.DAT audit file. This Charge Record contains information about the charge such as the ID of the server submitting the charge, when the charge is submitted, the amount of the charge, and so on.

Likewise, when the server calls the SubmitAccountNote function, a Note Record is added to the NET$ACCT.DAT audit file. It also contains information such as the ID of the server submitting the note, when the note is submitted, and so on.

11.2.1 Comment Field

Both records contain a comment field. The comment field of the Charge Record contains a binary record that holds information about the charge. For example, the server uses the comment field of the Charge Record to record the number of service units (connect time, disk I/Os, packet I/Os, and so on) from which a charge is computed.

The comment field of the Note Record contains information such as whether a station is logging in or out and when it is doing so. This field also contains the physical address of the station logging in or out.

11.2.2 Comment Type

Both records also contain a comment type field. This field contains the type of the comment record. The comment type is used to locate the associated record descriptor in the comment record definitions file (NET$REC.DAT). That record descriptor contains the field layout and text descriptions for the comment record.

Comment types are administered by Novell®. (Developers can call Developer Support at 1-800-REDWORD to obtain unique comment types. International customers can call 801-429-5588 or contact their local Novell office.)

11.3 Accounting Record File

The NET$REC.DAT file is a companion file to the NET$ACCT.DAT audit file. It contains the information about formatting and the control strings for displaying the binary records in the comment fields of the Charge Record and Note Record. Utility programs use this information in the NET$REC.DAT file to display the Charge and Note Records from the NET$ACCT.DAT audit file.

11.4 Accounting Services Example

The following scenario illustrates the use of accounting functions.

FS1 is one of many servers on an internetwork. FS1s bindery includes several objects (with various properties), including those listed in the following table (these properties exist only if accounting is enabled).

Object Name	Object ID	Properties
FS1	0x00030011	ACCOUNT_SERVERS
BILL	0x00060025	ACCOUNT_BALANCE
		ACCOUNT_HOLDS
PSERVER	0x5C2701F1	

In this example, BILL is a client, and PSERVER is a print server that advertises its services on the internetwork and appears in FS1s bindery as a bindery object. The supervisor of FS1 has declared PSERVER an accounting server by adding PSERVERs object ID to the 128-byte value segment associated with FS1s ACCOUNT_SERVERS property. This allows PSERVER to charge an object in FS1s bindery for service whenever that object uses PSERVERs printing services.

A user at workstation WS1 logs in to FS1 as BILL. The ACCOUNT_BALANCE property is an Item property with read-supervisor and write-supervisor security access levels. BILLs ACCOUNT_BALANCE property has a 128-byte value segment associated with it, as shown in the following table:

Offset	Content	Type	Order	Value
0	balance	long	signed high-low	00 00 13 88
4	credit limit	long	signed high-low	
8	Reserved	byte [120]		

The balance field specifies the amount of money that BILL can use to purchase services on the internetwork. The internetwork system administrator must choose which monetary unit to use. In this case, each unit equals 1 cent, so BILLs account balance equals $50.00 (0x1388).

The credit limit field specifies the amount of credit available to BILL for purchasing services on the internetwork. This field contains a signed integer that indicates the lowest permissible balance. Services that cause the balance to drop below this limit are denied. This limit can be positive (requiring the user always to maintain some funds) or negative (allowing the user to receive services after the balance has dropped, but above that negative value). The lowest value (0x80000000) actually results in unlimited credit.

BILLs ACCOUNT_HOLDS property has a 128-byte value segment associated with it, as shown in the following table:

Offset	Content		Type	Order
0	Server 1	(object ID)	long	high-low
4	Server 1	(amount)	long	
	Server 2	(objectID)	long	high-low
	Server 2	(amount)	long	signed high-low
120	Server 16	(object ID)	long	high-low
124	Server 16	(amount)	long	signed high-low

Each server object ID field specifies the object ID of a server about to perform a service for BILL. Each server amount field specifies the estimated charge for the service.

A server can charge (SubmitAccountCharge) or hold (SubmitAccountHold) the funds in BILLs ACCOUNT_BALANCE property. However, a placed hold (minus all other holds) fails if the amount in BILLs ACCOUNT_BALANCE is less than the amount specified in BILLs credit limit field. An attempt to hold also fails if 16 other servers have already placed holds on BILLs account. If a server submits multiple holds, they are combined into one hold entry in the users ACCOUNT_HOLDS property value.

If the client makes a request to queue a 10-page file for printing on PSERVER, PSERVER completes the following steps:

1. PSERVER estimates that, at 7 cents per page, the cost of BILLs print job is 70 cents.
2. PSERVER calls SubmitAccountHold to reserve 70 cents of BILLs account balance.
3. PSERVER queues the print job for printing.
4. PSERVER charges BILL 70 cents for the print job. SubmitAccountCharge relinquishes the hold amount associated with that function. A note about the charge is automatically submitted to the audit file.

11.5 Bindery-Based Accounting Functions

AccountingInstalled	Determines whether accounting is installed
GetAccountingStatus	Returns the status of a bindery object
SubmitAccountCharge	Updates the account of a bindery object
SubmitAccountChargeWithLength	Updates the account of a bindery object
SubmitAccountHold	Reserves a specified amount of an objects account balance pending a call to SubmitAccountCharge
SubmitAccountNote	Adds a note about an objects account to an audit record

Bindery-Based Accounting Functions

12

This documentation provides an alphabetical listing of the bindery-based accounting functions and describes their purpose, syntax, parameters, and return values.

IMPORTANT: The accounting functions in this set were designed for use on a NetWare® 3.x bindery, and work only with bindery emulation set on NetWare 4.x, 5.x, and 6.x servers. For accounting that is cross-platform and NDS enabled, see Chapter 1, "Accounting Concepts," on page 13.

AccountingInstalled

Determines if accounting is installed (For cross-platform functionality, see Developing NLMs with Cross-Platform Functions (*NDK: NLM Development Concepts, Tools, and Functions*) and call NWQueryAccountingInstalled (page 29))

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Bindery-Based Accounting

Syntax

```
#include <\nlm\nit\nwaccntg.h>

int AccountingInstalled (
   WORD   fileServerID);
```

Parameters

`fileServerID`

(IN) Specifies the file server ID number of the server to be checked for the presence of accounting.

Return Values

Decimal	Hex	Description
0	(0x00)	Accounting is not installed.
1	(0x01)	Accounting is installed.

Remarks

AccountingInstalled determines whether accounting is installed on the server identified by the `fileServerID` parameter.

GetAccountStatus

Returns the account status of a bindery object (For cross-platform functionality, see Developing NLMs with Cross-Platform Functions (*NDK: NLM Development Concepts, Tools, and Functions*) and call NWGetAccountStatus (page 26))

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Bindery-Based Accounting

Syntax

```
#include <\nlm\nit\nwaccntg.h>

int GetAccountStatus (
   WORD    binderyObjectType,
   char   *binderyObjectName,
   long   *balance,
   long   *limit,
   long   *holds);
```

Parameters

`binderyObjectType`

(IN) Specifies the type of bindery object for which the request is being made (see Object Type for specific object types).

`binderyObjectName`

(IN) Specifies the string containing the name of the bindery object for which the account status request is being made (maximum 48 characters, including the NULL terminator).

`balance`

(OUT) Receives the amount of value units available to the object to buy services on the network. (Optional: You should pass in a NULL value if the `balance` parameter does not need to be returned.)

`limit`

(OUT) Receives the lowest level the object's account balance can reach before the object can no longer buy services on the network. (Optional: You should pass in a NULL value if the `limit` parameter does not need to be returned.)

`holds`

(OUT) Points to a list of objects that have placed a hold on the account (an array of 32 long fields, 16 objects with amounts).

Return Values

Decimal	Hex	Constant
-1		OUT OF MEMORY
0	(0x00)	ESUCCESS
192	(0xC0)	ERR_NO_ACCOUNT_PRIVILEGES
193	(0xC1)	ERR_NO_ACCOUNT_BALANCE
196	(0xC4)	ERR_ACCOUNTING_DISABLED
236	(0xEC)	ERR_NO_SUCH_VALUE_SET
254	(0xFE)	ERR_BINDERY_LOCKED

Remarks

GetAccountStatus queries a server's bindery for the current account status of a specified bindery object by passing the `binderyObjectType` and `binderyObjectName` parameters and returning the `balance`, `limit`, and `holds` parameters.

The `binderyObjectType` and `binderyObjectName` parameters must uniquely identify the bindery object and must not contain wildcard characters. The `binderyObjectName` parameter is a NULL-terminated string that can be from 1 to 48 characters long, including the NULL terminator. Only printable characters can be used. Slashes, backslashes, colons, semicolons, commas, asterisks, spaces, and question marks are prohibited.

The value in the `balance` parameter represents the object's account balance, usually in some monetary unit such as cents.

The `holds` parameter lists servers that have issued SubmitAccountHold calls against the object and the amount reserved by each value-added server. The `holds` parameter is also the object ID number of a value-added server that has issued a SubmitAccountHold call against the object. Up to 16 servers can place holds on the account at any one time. Multiple holds from the same server are combined. Each server hold is made up of two long fields:

- The object ID of the server that placed the hold
- The amount of that server's hold

A list of each server hold is returned in the account `holds` parameter.

The GetAccountStatus function does not record an audit record in the SYS:SYSTEM\NET$ACCT.DAT audit file.

SubmitAccountCharge

Updates the account of a bindery object (For cross-platform functionality, see Developing NLMs with Cross-Platform Functions (*NDK: NLM Development Concepts, Tools, and Functions*) and call NWSubmitAccountCharge (page 31))

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Bindery-Based Accounting

Syntax

```
#include <\nlm\nit\nwaccntg.h>

int SubmitAccountCharge (
   WORD    binderyObjectType,
   char   *binderyObjectName,
   WORD    serviceType,
   long    chargeAmount,
   long    cancelHoldAmount,
   WORD    commentType,
   char   *comment);
```

Parameters

binderyObjectType

(IN) Specifies the type of bindery object whose account is to be updated (see Object Type for specific object types).

binderyObjectName

(IN) Specifies the string containing the name of the bindery object whose account is to be updated (maximum 48 characters, including the NULL terminator).

serviceType

(IN) Identifies the type of service for which the request is being made (usually the object type of the charging account server).

chargeAmount

(IN) Specifies the amount the account server charges for the service it provides.

cancelHoldAmount

(IN) Specifies the amount to be subtracted from the total amount of all holds previously placed by the server. If SubmitAccountHolds was not called before providing the service, this value should be zero.

commentType

(IN) Indicates the type of comment written to the audit report.

comment

(IN) Specifies the comment associated with the object's account charge.

Return Values

Decimal	Hex	Constant
0	(0x00)	ESUCCESS
192	(0xC0)	ERR_NO_ACCOUNT_PRIVILEGES
193	(0xC1)	ERR_NO_ACCOUNT_BALANCE
194	(0xC2)	ERR_CREDIT_LIMIT_EXCEEDED
196	(0xC4)	ERR_ACCOUNTING_DISABLED

Remarks

SubmitAccountCharge charges an object's account balance and relinquishes a hold against the object's account balance. The function can also write a note about the transaction in an audit record (optional). The charge and hold amounts do not have to be the same.

The `binderyObjectType` and `binderyObjectName` parameters must uniquely identify the bindery object and must not contain wildcard characters. The `binderyObjectName` parameter is a NULL-terminated string (maximum 48 characters, including the NULL terminator). Only printable characters can be used. Slashes, backslashes, colons, semicolons, commas, spaces, asterisks, and question marks are prohibited.

The `serviceType` parameter contains the specific type of service for which the charge is made. External servers should use their object type. If a server provides several different services, and no reasonable object type equivalents exist for these services, the vendor should apply to Novell® for a service type. Usually, however, the server should use its object type and distinguish the type of service being charged for in the comment field of the charge record. (See Object Type for specific object types.

The `commentType` parameter contains the type of the comment in the `comment` parameter. Comment types are administered by Novell and are listed below.

- Connect Time Charge
- Disk Storage Charge
- Log In Note
- Log Out Note
- Account Locked Note
- Server Time Modified Note

Developers should contact Novell for unique comment types. Comment types greater than 0x8000 are reserved for experimental purposes. The description is located in the NET$REC.DAT file.

The `comment` is the entry that the value-added server makes in the audit record. This audit record is contained in the SYS:SYSTEM\NET$ACCT.DAT file.

SubmitAccountChargeWithLength

Updates the account of a bindery object (For cross-platform functionality, see Developing NLMs with Cross-Platform Functions (*NDK: NLM Development Concepts, Tools, and Functions*) and call NWSubmitAccountCharge (page 31))

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Bindery-Based Accounting

Syntax

```
#include <\nlm\nit\nwaccntg.h>

int SubmitAccountChargeWithLength (
   WORD    binderyObjectType,
   char   *binderyObjectName,
   WORD    serviceType,
   long    chargeAmount,
   long    cancelHoldAmount,
   WORD    commentType,
   void   *commentData,
   WORD    commentLength);
```

Parameters

binderyObjectType

(IN) Specifies the type of bindery object whose account is to be updated (see Object Type for specific object types).

binderyObjectName

(IN) Specifies the string containing the name of the bindery object whose account is to be updated (maximum 48 characters, including the NULL terminator).

serviceType

(IN) Identifies the type of service for which the request is being made (usually the object type of the charging account server).

chargeAmount

(IN) Specifies the amount the account server charges for the service it provides.

cancelHoldAmount

(IN) Specifies the amount to be subtracted from the total amount of all holds previously placed by the server. If SubmitAccountHolds was not called before providing the service, this value should be zero.

commentType

(IN) Indicates the type of comment written to the audit report.

commentData

(IN) Specifies the comment data associated with the object's account charge.

commentLength

(IN) Specifies the length of the `commentData` parameter.

Return Values

Decimal	Hex	Constant
0	(0x00)	ESUCCESS
192	(0xC0)	ERR_NO_ACCOUNT_PRIVILEGES
193	(0xC1)	ERR_NO_ACCOUNT_BALANCE
194	(0xC2)	ERR_CREDIT_LIMIT_EXCEEDED
196	(0xC4)	ERR_ACCOUNTING_DISABLED
240	(0xF0)	ERR_ILLEGAL_WILDCARD
251	(0xFB)	ERR_INVALID_PARAMETERS

Remarks

SubmitAccountChargeWithLength is identical to SubmitAccountCharge, except that it includes a `commentLength` parameter. This parameter allows the use of binary data for the `commentData` parameter (SubmitAccountCharge expects a NULL-terminated string for `commentData`).

This function charges an object's account balance and relinquishes a hold against the object's account balance. The function can also write a note about the transaction in an audit record (optional). The charge and hold amounts do not have to be the same.

The `binderyObjectType` and `binderyObjectName` parameters must uniquely identify the bindery object and must not contain wildcard characters. The `binderyObjectName` parameter is a NULL-terminated string (maximum 48 characters, including the NULL terminator). Only printable characters can be used. Slashes, backslashes, colons, semicolons, commas, spaces, asterisks, and question marks are prohibited.

The `serviceType` parameter contains the specific type of service for which the charge is made. External servers should use their object type. If a server provides several different services, and no reasonable object type equivalents exist for these services, the vendor should apply to Novell for a service type. Usually, however, the server should use its object type and distinguish the type of service being charged for in the comment field of the charge record. (See Object Type.

The `commentType` parameter contains the type of the comment in the `comment` parameter. Comment types are administered by Novell and are listed below.

- Connect Time Charge

- Disk Storage Charge
- Log In Note
- Log Out Note
- Account Locked Note
- Server Time Modified Note

Developers should contact Novell for unique comment types. Comment types greater than 0x8000 are reserved for experimental purposes. The description is located in the NET$REC.DAT file.

The `comment` is the entry that the value-added server makes in the audit record. This audit record is contained in the SYS:SYSTEM\NET$ACCT.DAT file.

SubmitAccountHold

Reserves a specified amount of an object's account balance pending a call to SubmitAccountCharge (For cross-platform functionality, see Developing NLMs with Cross-Platform Functions (*NDK: NLM Development Concepts, Tools, and Functions*) and call NWSubmitAccountHold (page 34))

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Bindery-Based Accounting

Syntax

```
#include <\nlm\nit\nwaccntg.h>

int SubmitAccountHold (
   WORD    binderyObjectType,
   char   *binderyObjectName,
   long    reserveAmount);
```

Parameters

binderyObjectType

(IN) Specifies the type of bindery object whose account balance is to be partially reserved (see Object Type for specific object types).

binderyObjectName

(IN) Specifies the string containing the name of the bindery object whose account balance is to be partially reserved (maximum 48 characters, including the NULL terminator).

reserveAmount

(IN) Specifies the hold amount to be placed against the client's account pending service.

Return Values

Decimal	Hex	Constant
0	(0x00)	ESUCCESS
192	(0xC0)	ERR_NO_ACCOUNT_PRIVILEGES
193	(0xC1)	ERR_NO_ACCOUNT_BALANCE
194	(0xC2)	ERR_CREDIT_LIMIT_EXCEEDED
196	(0xC4)	ERR_ACCOUNTING_DISABLED

Decimal	Hex	Constant
195	(0xC3)	ERR_TOO_MANY_HOLDS

Remarks

SubmitAccountHold reserves a specified amount of an object's account balance before that object receives and is charged for a service on the network

The `binderyObjectType` and `binderyObjectName` parameters must uniquely identify the bindery object and must not contain wildcard characters. The `binderyObjectName` parameter is a NULL-terminated string (maximum 48 characters, including the NULL terminator). Only printable characters can be used. Slashes, backslashes, colons, semicolons, commas, asterisks, spaces, and question marks are prohibited.

The `reserveAmount` parameter gets the amount that the server expects to charge for the service it is about to provide to the object. No more than 16 servers can reserve amounts of an object's account balance at one time. Multiple holds from the same server are combined.

SubmitAccountNote

Adds a note about an object's account to an audit record (For cross-platform functionality, see Developing NLMs with Cross-Platform Functions (*NDK: NLM Development Concepts, Tools, and Functions*) and call NWSubmitAccountNote (page 37))

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 3.x, 4.x, 5.x, 6.x

Platform: NLM

Service: Bindery-Based Accounting

Syntax

```
#include <\nlm\nit\nwaccntg.h>

int SubmitAccountNote (
   WORD    binderyObjectType,
   char   *binderyObjectName,
   WORD    serviceType,
   WORD    commentType,
   char   *comment)
```

Parameters

binderyObjectType

(IN) Specifies the type of bindery object for which the request is being made (see Object Type for specific object types).

binderyObjectName

(IN) Specifies the string containing the name of the bindery object for which the request is being made (maximum 48 characters, including the NULL terminator).

serviceType

(IN) Identifies the type of service for which the request is being made (usually the object type of the charging account server).

commentType

(IN) Indicates the type of comment in the `comment` parameter.

comment

(IN) Specifies the comment associated with the object's account.

Return Values

Decimal	Hex	Constant
0	(0x00)	ESUCCESS
192	(0xC0)	ERR_NO_ACCOUNT_PRIVILEGES
196	(0xC4)	ERR_ACCOUNTING_DISABLED

Remarks

The `binderyObjectType` and `binderyObjectName` parameters must uniquely identify the bindery object and must not contain wildcard characters. The `binderyObjectName` parameter is a NULL-terminated string (maximum 48 characters, including the NULL terminator). Only printable characters can be used. Slashes, backslashes, colons, semicolons, commas, asterisks, spaces, and question marks are prohibited.

The `serviceType` parameter usually contains the object type of the charging account server.

The `commentType` parameter contains the type of the comment in the `comment` parameter. Comment types are administered by Novell and are listed below.

- Connect Time Charge
- Disk Storage Charge
- Log In Note
- Log Out Note
- Account Locked Note
- Server Time Modified Note

Developers should contact Novell for unique comment types. Comment types greater than 0x8000 are reserved for experimental purposes. The description is located in the NET$REC.DAT file.

The `comment` parameter contains the entry that the server makes in the audit record. The audit record is contained in the SYS:SYSTEM\NET$ACCT.DAT file. Comments are restricted to 256 bytes, including the NULL terminator.

Revision History

A

The following table outlines all the changes that have been made to the Network Management documentation (in reverse chronological order).

October 5, 2005	Transitioned to revised Novell documentation standards.
March 2, 2005	Fixed the preface and legal information.
October 2002	Updated the Pascal syntaxes for the structures.
May 2002	Changed Pascal syntaxes to correctly refer to the calwin32 file.
	Updated Pascal syntax of NWGetObjectNamesBeginA (page 148) and NWGetObjectNamesNextA (page 150).
February 2002	Updated Pascal syntaxes.
	Updated links.
October 2001	Updated Pascal syntax for NWGetObjectNamesBeginA (page 148), NWGetObjectNamesNextA (page 150), and NWGetObjectNamesEndA (page 152).
September 2001	Added support for NetWare 6.x to documentation.
June 2001	Updated tables.
February 2001	Added Delphi (Pascal) syntax to functions where missing.
	Since NWADReadBitMap (page 112) works succesfully on NetWare 4.11, 5.0, and 5.1, fixed the remarks section of this function.
July 2000	Corrected the information for the pluLenBuffer parameter in the NWGetObjectNamesNextA (page 150) function.
March 2000	Moved “Server-Based Auditing Concepts” on page 153, “Server-Based Auditing Functions” on page 157, “Bindery-Based Accounting Concepts” on page 165, and “Bindery-Based Accounting Functions” on page 169 from Server Management.
	Added brief description of Name Retrieval to the preface.
January 2000	Removed NWAuditBitMapTNW and NWDSContainerConfigHdr structures since they are not used by any documented functions.
November 1999	Added “Name Retrieval Functions” on page 147.
	Added library information to each function.