Accounting Services

Accounting Services is an application programming interface (API) that allows a server to charge for the use of its services. For example, a database server can charge for the number of records viewed, the number of requests serviced or the amount of connect time. A print server can charge for the number of pages printed.

The file server supervisor determines the charge for each type of service, and the file server bindery stores the list of authorized accounting servers and each
client's accounting information.

Once a server is listed as an authorized accounting server, it can submit a charge to a client's account, place a hold against a client's account or query a client's account status. An audit trail of all charges is accumulated in an audit file. This chapter is divided into the following sections:

- Bindery Properties
- Accounting Audit File
- File Server Charges
- Data Structures

Bindery Properties

Information used by the accounting APIs is stored in three bindery properties:

- ACCOUNT_SERVERS property
- ACCOUNT_BALANCE property
- ACCOUNT_HOLDS property

These properties are discussed in detail in the following sections.

ACCOUNT_SERVERS Property

When accounting is enabled (using the NetWare utility SYSCON), the ACCOUNT_SERVERS property is created and attached to the file server object. This property contains the bindery object ID of every server, including the file server, authorized to submit accounting records to clients. Before a server can charge for its services, the server must be added to the ACCOUNT_SERVERS property.

For example, to enable a server called PRINTSERV to charge clients for services on a file server called SAM, you must complete the following steps.

1) Install accounting on the file server SAM. Log in as supervisor and run SYSCON. Select to install accounting.

2) Use NWCreateObject to create PRINTSERV as a bindery object. Contact Novell if you need a unique object type.

3) Add PRINTSERV to the ACCOUNT_SERVERS property. Use NWAddObjectToSet.

   For *objectName*, use the file server's name (SAM).

   For *objectType*, use the file server's type (NWOT_FILE_SERVER).

   For *propertyName*, use ACCOUNT_SERVERS.

   For *memberName*, use the object name created in Step 2 (PRINTSERV).

   For *memberTyp*e, use the object that you declared in Step 2 (for a print server, NWOT_PRINT_SERVER).

The ACCOUNT_SERVERS property is of type set and has read-supervisor and write-supervisor security access levels. If a file server object has no ACCOUNT_SERVERS property, servers should assume that accounting is disabled, and no charges for services are made.

ACCOUNT_BALANCE Property

When accounting is enabled, each user object is given an ACCOUNT_BALANCE property. Every user object created after accounting is enabled will also be given an ACCOUNT_BALANCE property. An object's account balance and minimum balance are stored in this property. Servers can deny services to an object if the object has no ACCOUNT_BALANCE property.

The ACCOUNT_BALANCE property is of type data and has read-object and write-supervisor security access levels. The account balance places a general limitation on a client's use of all network resources. However, Accounting Services does not provide any method for placing a ceiling on a client's use of a particular resource or service.

The account balance usually represents some monetary unit such as cents. The system administrator must ensure that all servers submit their charges using the same monetary unit. If an object's account balance has no more funds, servers should refuse further service. However, if service has already been rendered, the server can charge for it even though no funds are in the account balance. Setting the minimum balance to 80000000h (the most negative signed long integer) indicates the object has no minimum balance, in which case service should never be refused.

The ACCOUNT_BALANCE property has several property fields which are described below.

Property Fields

The fields in the first data block of the ACCOUNT_BALANCE property are listed and defined below.

TABLE 1. ACCOUNT_BALANCE property

Offset	Field	Type
0	Balance	int32
4	Minimum Balance	int32
8	Reserved	uint8 [120]

Balance. This field contains a signed integer that indicates the account balance.

Minimum Balance. This field contains a signed integer that indicates the lowest permissible balance. Services that cause the balance to drop below this minimum are denied. The minimum can be positive (requiring the user to always maintain some funds) or negative (allowing the user to receive services after the balance has dropped below zero.)

Reserved. This field contains NetWare internal information.

ACCOUNT_HOLDS Property

The server can place a hold on a client's account before a request is serviced to ensure that the client has sufficient funds to pay for service. If the hold succeeds, this indicates the client has sufficient funds. The server can then perform the requested service, submit a charge for the service and release the amount placed on hold.

If there are holds against a client's account, the holds are stored in the ACCOUNT_HOLDS property. If there are no holds on the account, this property is not present. The ACCOUNT_HOLDS property is dynamic, of type data, and has read-object and write-supervisor security access levels.

If a hold, along with all other holds, require more funds than exist in the client's account (client's funds have gone below minimum balance), the hold request fails. The requested service should then be refused. An attempt to hold also fails if 16 other servers have already placed holds on a client's account.

If the server does not place a hold on an account to ensure the client has sufficient funds, the server should query

the account before rendering service. Then, if the client's account is empty, the request for service should be refused. If a holding server is disconnected, the hold it had on a client's account is cleared. The holding server can also explicitly clear a hold. If a server submits multiple holds, they accumulate into one hold.

The ACCOUNT_HOLDS property has several property fields which are described in the next paragraphs.

Property Fields

The fields in the first data block are shown below.

TABLE 2. ACCOUNT_HOLDS Property

Offset	Field	Type
0	Holder ID 1	uint32
4	Amount 1	int32
8	Holder ID 2	uint32
12	Amount 2	uint32
.	.	
.	.	
.	.	
112	Holder ID 15	uint32
116	Amount 15	uint32
120	Holder ID 16	uint32
124	Amount 16	uint32

Holder ID. This field contains the bindery object ID of the server placing the hold. If this field is zero, the hold slot is not in use regardless of the contents of the amount field.

Amount. This field contains the amount of the hold.

Accounting Audit File

In addition to monitoring a network by charging, querying or holding a client's account, Accounting Services offers a method for monitoring network resources by keeping an audit trail of all charges. The two APIs NWSubmitAccountCharge and NWSubmitAccountNote and the Accounting Audit file (NET$ACCT.DAT) are the tools used to keep the Audit trail.

The 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 two APIs are called, a detailed record of all accounting charges and activities is kept in the Accounting Audit file, NET$ACCT.DAT. For example, when a server submits a charge or a note, a Charge Record or a Note Record is added to the audit file. This section describes the structure of these two records and then describes in detail two of the records' fields: the comment type field and the comment field.

Charge and Note Records

The Charge Record and Note Record are distinguished by the contents of their Record Type field. When the server charges an account through the NWSubmitAccountCharge function call, a Charge Record is added to the NET$ACCT.DAT audit file. This Charge Record contains information about the charge such as the object 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 NWSubmitAccountNote function call, a Note Record is added to the NET$ACCT.DAT audit file. It also contains information such as the bindery object ID of the server submitting the

note, when the note is submitted and so on.

Both records also contain a Comment field.

- **Charge Record**. The Comment field of the Charge Record contains a binary record that holds information about the charge. For
example, the file server uses the comment field of the Charge Record to record the number of service units (connect time, disk I/Os, packet I/Os, etc.) from which a charge is computed.
- **Note record**. 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 contains the physical address of the station logging in or out.

A more detailed list and description of each record's fields are provided below.

Charge Record Structure

The Charge Record structure is shown below.

TABLE 3. Charge Record Structure

Offset	Field	Type
0	Length	uint16
2	Server ID	uint32
6	Time Stamp	uint8 [6]
12	Record Type	uint8 [1]
13	Completion Code	uint8 [1]
14	Service Type	uint16
16	Client ID	uint32
20	Amount	int32
24	Comment Type	uint16
26	Comment	char [*]

Length. This field contains the number of bytes in the record excluding Length itself.

Server ID. This field contains the object ID of the server submitting the accounting request.

Time Stamp. This field contains the date and time the server submitted the charge. The format is year (year = current year - 1900), month, day, hour, minute, second, where each is a byte-sized integer.

Record Type. This field contains the record type: A Charge Record is type 1, and a Note Record is type 2.

Completion Code. This field contains the completion code of the Submit Charge request. If the completion code is SUCCESSFUL (00) or
CREDIT_EXCEEDED (C2), the account balance is debited.

Service Type. This field 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 well-known service type. (Contact Novell's API Consulting Group.) 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.

Client ID. This field contains the object ID of the object that is being charged for service.

Amount. This field contains the amount of the charge. Amount can be negative if the server must make a refund.

Comment Type. 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. (See ªComment Type Field Definitionsº for more information on comment fields.)

Comment types are administered by Novell. (Developers should contact Novell's API Consulting Group for unique comment types.) Comment types greater than 8000h are reserved for experimental purposes.

Comment. The Comment field of the Charge Record contains a binary record that holds information about the charge. For example, the file server uses the comment field to record the number of service units (connect time, disk I/Os, packet I/Os, etc.) from which a charge is computed. (See ªComment Field Definitionsº for more information on comment fields.)

Note Record Structure

The Note Record structure is shown below.

TABLE 4. Note Record Structure

Offset	Field	Type
0	Length	uint16
2	Server ID	uint32
6	Time Stamp	uint8 [6]
12	Record Type	uint8 [1]
13	Reserved	uint8 [1]
14	Service Type	uint16
16	Client ID	uint32
20	Comment Type	uint16
22	Comment	char [*]

Length. This field contains the number of bytes in the record excluding Length itself.

Server ID. This field contains the object ID of the server submitting the accounting request. A Server ID of zero indicates a request by an internal server.

Time Stamp. This field contains the date and time the server submitted the note. The format is year (year = current year - 1900), month, day, hour, minute, second, where each is a byte-sized integer.

Record Type. This field contains the record type: Charge Record is type 1, and Note Record is type 2.

Service Type. This field contains the specific type of service to which the note applies. External servers should use their object type. If a server provides several disparate types of services, and no reasonable object type equivalents exist for these services, the vendor can apply to Novell for a well-known service type. (Contact Novell's API Consulting Group.) Usually, however, the server should use its object type and distinguish the type of service for which the note is produced in the comment field of the note record.

Client ID. This field contains the object ID to which the note applies.

Comment Type. 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. (See ªComment Type Field Definitionsº below for more information on comment fields.)

Comment types are administered by Novell. (Developers should contact Novell's API Consulting Group for unique comment types.) Comment types 8000h and higher are reserved for experimental purposes.

Comment. This field contains a binary record that holds information relating to the charge or note. For example, the file server records in this field 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. (See ªComment Type Field

Definitions" for more information on comment fields.)

Comment Type Field Definitions

The NWSubmitAccountCharge function call and the NWSubmitAccountNote function call allow a file server to record information in the audit file about a client's account activities. This information is a binary record in the Comment field of the Note Record or Charge Record. These record types are described below.

Connect Time Charge

The Comment Type is 1.

This record contains the number of minutes a station was connected to the server, the number of packets sent to the server and the number of disk I/Os.

The fields in this comment type are listed in the following chart.

TABLE 5. Connect Time Charge

Offset	Field	Type
0	Connect Time	uint32
4	Request Count	uint32
8	Bytes Read	uint8[6]*
14	Bytes Written	uint8[6]*

* These bytes are in hi-lo order.

The format control string:

> a"Connected %lu minutes; %lu requests; %04x%04x%04xh bytes read;
> %04x%04x%04xh bytes written."

Disk Storage Charge

The Comment Type is 2.

This Comment contains the number of blocks owned by an account at the time of the charge and the number of days in the charge period. The fields in this comment type are listed below.

TABLE 6. Disk Storage Charge

Offset	Field	Type
0	Blocks Owned	uint32
4	Number of Half Hours	uint32

The format control string:

> a"%lu disk blocks stored for %lu half-hours."

Log In Note

The Comment Type is 3.

This Comment is recorded whenever an object successfully logs in. The
Comment field of the Note Record contains the physical address of the station logging in. The fields in this comment type are listed below.

TABLE 7. Log In Note

Offset	Field	Type
0	Network Address	uint8[4]
4	Node Address	uint8[6]

The format control string:

[a]Login from address %lx:%lx%x.[o]

Log Out Note

The Comment Type is 4.

This Comment is recorded whenever an object logs out. The Comment field of the Note Record contains the physical address of the station logging out. The fields in this comment type are listed below.

TABLE 8. Log Out Note

Offset	Field	Type
0	Network Address	uint8[4]
4	Node Address	uint8[6]

The format control string:

[a]Logout from address %lx:%lx%x.[o]

Account Locked Note

The Comment Type is 5.

This Comment is recorded whenever an account is locked because of too many incorrect login attempts. The Client field of the Note Record contains the object ID of the object being locked out. The Comment field of the Note Record contains the physical address of the station being locked out. The fields in this comment type are listed below.

TABLE 9. Account Locked Note

Offset	Field	Type
0	Network Address	uint8[4]
4	Node Address	uint8[6]

The format control string:

[a]Account intruder lockout caused by address %lx:%lx%x.[o]

Server Time Modified Note

The Comment Type is 6.

This Comment is recorded when an operator modifies the server time. The Time Stamp field contains the current time before the change was made. The Client field contains the object ID of the object which changed the time. The comment field contains the new time and date. The fields in this comment type are listed in the following table.

TABLE 10. Server Time Modified Note

Offset	Field	Type
0	Year Since 1900	uint8 [1]
1	Month	uint8 [1]

2	Day	uint8 [1]
3	Hour	uint8 [1]
4	Minute	uint8 [1]
5	Second	uint8 [1]

The format control string:

ªSystem time changed to 19%02d-%02d-%02d %d:%02d:%02d.º

Comment Field Definitions

The NET$REC.DAT file is a companion file to the NET$ACCT.DAT Audit file. It contains the information for formatting and the control strings for displaying the binary records in the Comment fields of the Charge and Note Records. Utility programs use the information in NET$REC.DAT to display the Charge and Note Records from the NET$ACCT.DAT audit file. The information in the
NET$REC.DAT file is listed in the following table.

TABLE 11. NET$REC.DAT File

Offset	Field	Type
0	Length	uint16
2	Comment Type	uint16
4	Data Type Count	uint8 [1]
5	Data Type Value [1]	uint8
6	Data Type Elements [1]	(as data type defines)
	.	
	.	
	.	
	Data Type Value [n]	uint8
	Data Type Elements [n]	(as data type defines)

Length. This field contains the length of the record descriptor, not including the Length field itself. The length is:

(Field Count * 2) + Format Length + 4

Comment Type. This field contains the type of the comment being described. This Comment Type is also a field in the Charge and Note Records, used there as a reference to this record descriptor. Well-known comment types can be obtained from Novell. (Contact Novell's API Consulting Group.) Comment types 8000h and higher are reserved for experimental purposes.

Data Type. This field contains the data type of the record. Valid types are listed below.

TABLE 12. Data Types

Value	Element Description
1	An 8-bit value
2	A 16-bit value with the bytes stored
3	A 32-bit value with the bytes stored
4	An 8-bit length value followed by an array of characters. The size of the array is specified by the length value.

Note: You should read and write data in hi-lo format. Then, for your particular hardware, take the appropriate action.

Data Type Count. This field contains the number of data types to follow. Data types are listed in Table 12.

File Server Charges

File servers can submit charges for connection time, requests made, blocks read, blocks written and disk storage. The following Accounting Services bindery properties describe these charges. The first four properties (CONNECT_TIME, REQUESTS_MADE, BLOCKS_READ, BLOCKS_WRITTEN) share the same data structure. The last property, DISK_STORAGE, has a different data structure. This section describes each charge.

CONNECT_TIME Property

This property designates the amount a file server can charge for connect time. Connect time is measured from the time the user logs in to the time the user logs out. The user's account is checked each half hour. If the account balance falls below the lowest permissible balance, the user is disconnected following 5 minute and 1 minute warnings.

REQUESTS_MADE Property

This property designates the amount a file server can charge for requests made to the file server. After logging out, the user can be charged for the total number of requests submitted to the file server since logging in. While the user is logged in, the user's account is checked each half hour. If the account balance falls below the lowest permissible balance, the user is disconnected following 5-minute and 1-minute warnings.

BLOCKS_READ Property

This property designates the amount the file server can charge for blocks read. After logging out, the user can be charged for the number of blocks (4096 bytes per block) read since logging in. While the user is logged in, the user's account is checked each half hour. If the account balance falls below the lowest permissible balance, the user is disconnected following 5-minute and 1-minute warnings.

BLOCKS_WRITTEN Property

This property designates the amount the file server can charge for blocks written. After logging out, the user can be charged for the number of blocks (4096 bytes per block) written from disk since logging in. While the user is logged in, the user's account is checked each half hour. If the account balance falls below the lowest permissible balance, the user is disconnected following 5-minute and
1-minute warnings.

DISK_STORAGE Property

This property designates the amount the file server can charge for disk storage. The charge is computed at regular intervals as follows:

- The total disk storage for each user is calculated.
- Total disk storage is multiplied by the number of half hours since the last disk storage charge was made.
- The result is multiplied by the current charge rate.

Unlike charges designated in the CONNECT_TIME, REQUESTS_MADE, BLOCKS_WRITTEN, and BLOCKS_READ properties, the list of times and the charge rates designated in the DISK_STORAGE property represent a specific time a charge is made and a specific rate for the charge.

Data Structures

The file server maintains a schedule of charges for its services in bindery property data structures. The data structure for connection time, requests made, blocks read and blocks written is described below.

CONNECT_TIME, REQUESTS_MADE, BLOCKS_READ and BLOCKS_WRITTEN Charges

The CONNECT_TIME, REQUESTS_MADE, BLOCKS_READ and
BLOCKS_WRITTEN properties share the same data structure. This structure is shown in the following table.

TABLE 13. Data Structure for Four Types of Charges

Offset	Field	Type

0	Time of Next Change	uint32
4	Current Charge Rate Multiplier	uint16
6	Current Charge Rate Divisor	uint16
8	Days Change Occurs Mask 1	uint8 [1]
9	Time Change Occurs 1	uint8 [1]
10	Charge Rate Multiplier 1	uint16
12	Charge Rate Divisor 1	uint16
	.	
	.	
122	Days Change Occurs Mask 20	uint8 [1]
123	Time Change Occurs 20	uint8 [1]
124	Charge Rate Multiplier 20	uint16
126	Charge Rate Divisor 20	uint16

Time of Next Change. This field contains the time of the next change. Time is measured in minutes since January 1, 1985.

Current Charge Rate Multiplier and Divisor. These fields contain the Current Charge Rate Multiplier and Divisor. The charge is computed as follows:

- The unit of service (connect time, requests made, blocks read, or blocks written) is multiplied by the Current Charge Rate Multiplier.
- The resulting value is divided by the Current Charge Rate Divisor to give the charge made against the user's account. (If the Current Charge Rate Multiplier or Divisor is zero, no charge is made for the service.)

Days Change Occurs Mask. This field contains a bit mask that specifies the days of the week for which the charge rate applies. If the bit corresponding to the day of the week is set (bit 0 = Sunday, ..., bit 6 = Saturday), the charge is made during that day at the half hour specified in Time Change Occurs. If this field is zero, no changes are made.

Time Change Occurs. This field contains the half hour (12am = 0, ..., 11:30pm = 47) during which the specified charge rate takes effect. These changes in the charge rate are listed in this property structure according to increasing half hours.

Charge Rate Multiplier and Divisor. These fields contain the charge rate that takes effect at the specified time.

Disk Storage Charges

Charges for disk storage are maintained in the DISK_STORAGE property. The structure of this property is described below.

TABLE 14. (Continued)DISK_STORAGE Data Structure

Offset	Field	Type
0	Time of Next Charge	uint32
4	Time of Previous Charge	uint16
8	Days Charge Occurs Mask 1	uint8 [1]
9	Time Charge Occurs 1	uint8 [1]
10	Charge Rate Multiplier 1	uint16
12	Charge Rate Divisor 1	uint16
	.	
	.	
122	Days Charge Occurs Mask 20	uint8 [1]
123	Time Charge Occurs 20	uint8 [1]
124	Charge Rate Multiplier 20	uint16
126	Charge Rate Divisor 20	uint16

Time of Next Charge. This field contains the time of the next charge. Time is measured in minutes since January 1, 1985.

Time of Previous Charge. This field contains the time of the previous charge. Time is measured in minutes since January 1, 1985.

Days Charge Occurs Mask. This field contains a bit mask that specifies the days of the week for which the charge rate applies. If the bit corresponding to the day of the week is set (bit 0 = Sunday, ..., bit 6 = Saturday), the charge is made during that day at the half hour specified in Time Charge Occurs. If this field is zero, no charges are made.

Time Charge Occurs. This field contains the half hour (12am = 0, ..., 11:30pm = 47) during which the specified charge rate takes effect. These changes in the charge rate are listed in this property structure according to increasing half hours.

Charge Rate Multiplier and Divisor. These fields contain the charge rate used to calculate the disk storage charge at the specified half hour on the specified days.

The charge is computed as follows:

- The current disk storage for each user is calculated in 4K blocks.
- Current disk storage is multiplied by the number of half hours since the last disk storage charge was made.
- The result is multiplied by the charge rate multiplier and divided by the charge rate divisor to produce the final disk storage charge.