Bindery Services

In a local area network environment there is the need for a name service which provides a way for network resources and clients to be identified. A resource is anything that provides a service such as a file server, print server or database server. A client is the consumer of the services provided by a resource. Each NetWare file server maintains a database of the resources and clients available on the network. This special-purpose database is called the bindery. This chapter is divided into the following sections:

- Bindery Overview
- Bindery Security
- Bindery Objects
- Bindery Properties

Bindery Overview

The fundamental benefit of the bindery is that it allows the design of an organized and secure operating environment. The bindery provides the foundation upon which NetWare's client security and server advertising are built. The bindery contains information on each client and is the basis upon which NetWare's client security mechanisms are built, including client password protection, client accounting and client restrictions.

The bindery also contains information about each resource on the network. For example, resources which advertise their services have their name and internetwork address stored in every file server's bindery in the internetwork. Therefore, the bindery can also be used as a resource directory where clients can extract a listing of all resources available on the network.

The bindery is not designed to be used exclusively by the NetWare operating system, but is a flexible database with an extensive programming interface that can also be used by third-party applications. Applications can access the bindery to extract information about network users and resources. The bindery can also be used to store application specific information. For example, an application's user and configuration information could be stored in the bindery.

Bindery Components

The bindery is comprised primarily of components called objects and properties. An object can be a user, user group, file server, print server, or any other logical or physical entity on the network that has been given a name. Each object has associated with it a set of characteristics called properties, as represented below.

F2.eps ,

Each object can have multiple properties associated with it. Each property may have an attached value. The property's value contains the actual data that is
associated with a property.

A user's set of properties may include a password, an account balance and a list of groups the user is a member of, whereas a server might have just one property which contains its network address.

The actual data associated with a property is stored in the property's value. A user's password, for example, is stored in the property value associated with the password property. An example bindery object and its associated properties and values is shown on the next page.

F1.eps ,

Bindery Constraints

The bindery can contain any combination of up to 65,000 bindery objects and properties. Although the bindery's structure is flexible, it should not be overloaded with seldom-used information. A user's mailing list or login preference, for example, might be better stored in the user's mailbox directory. A large bindery is naturally slower to access than a small bindery, and will affect the overall performance of the network. Also, when updates are made to the bindery, there is no attempt to coordinate requests among multiple workstations. Logical locks may be used to coordinate bindery management among workstations.

Bindery Files

Each file server maintains its bindery as hidden files in the SYS:SYSTEM directory. The bindery files are NET$OBJ.SYS, NET$PROP.SYS and
NET$VAL.SYS. The NET$OBJ.SYS file contains objects; NET$PROP.SYS, properties; and NET$VAL.SYS, property values.

Each file server manages its own bindery files independently of other file servers' binderies. Therefore, the resources and clients available to one file server may not be available to other file servers. Even when similar objects are defined in more than one bindery, each object's unique ID differs from one bindery to another. (An object is given a unique ID when it is created.)

Archiving Bindery Files

It is important to archive the bindery on a regular basis because the bindery files contain information about the file server's clients. However, the bindery files cannot be accessed directly because the file server keeps the bindery files open and locked at all times. To archive the bindery files, the bindery must be closed with the NWCloseBindery function. The NWCloseBindery function allows the supervisor, or an object that is security equivalent to the supervisor, to close and unlock the bindery files, thus allowing the bindery to be archived. After the bindery files have been archived, the NWOpenBindery function is used to give control of the bindery files back to the file server. While the bindery is closed, much of the functionality of the network is disabled. Therefore, the time that the bindery is closed should be kept to a minimum.

Bindery Access

Applications can query the network for information about named network objects through the bindery function calls. In addition to reading the bindery, applications can use bindery function calls to create bindery objects, add properties to bindery objects and give values to bindery objects' properties. The ability of the application to read from or write to the bindery depends upon the security access levels of both the object and the property to be accessed. The security is enforced by the operating system of the file server where the bindery resides.

Bindery Security

Each file server administers the security for its local resources, services and client accounts through the bindery. The bindery has several levels of security which together provide a flexible yet secure operating environment. Each object and property in the bindery has a security access level associated with it. The security access level controls the read and write access to a bindery object and its properties. Each bindery has a SUPERVISOR object that is granted special bindery security privileges. The supervisor is allowed to grant these special administrative privileges to other objects through the security equivalence feature. The security equivalence feature allows a bindery object to be granted the same access rights as another object. The security equivalence feature is also useful in defining user groups. User groups are a means of logically organizing users into workgroups. This allows the system supervisor to simplify the security process. Generally, a user group is assigned specific directory access rights, then users are added to the user group.

Directory Trustees

The bindery's security and the file system's security are independent. The bindery does not store any of the file system's directory trustee information. Directory trustees are stored in directory entries which are an integral part of NetWare's physical directory structure. The only relationship that the bindery and the file system have is that the file system stores each directory's trustee in the form of an object ID. Refer to the File Services discussion for more information on file system security and directory trustees.

Security Levels

The bindery's security level controls the read and write access of others to a bindery object or property. Each bindery object has an object security level associated with it, and each property has a property security level associated with it. The object security and the property security are each two nibbles; the low-order nibble controls the read security and the high-order nibble controls the write security. The following chart defines the values for each nibble.

TABLE 15. Security Levels

Hex	Binary	Access	Description
0	0000	Anyone	Access allowed to all clients, even if the client has not logged in to the file server.
1	0001	Logged	Access allowed only to clients who have logged in to the file server.
2	0010	Object	Access allowed only to clients who have logged in to the file server with the object's name, type and password.
3	0011	Supervisor	Access allowed only to clients logged in to the server as supervisor or as an object with supervisor security equivalence.
4	0100	NetWare	Access allowed only to NetWare.

For example, a bindery object with an object security level of 31h (supervisor write--logged read) can be viewed by any client that has successfully logged in to the server, but can only have properties added to it by a client that is security equivalent to the supervisor. If this object has a property with a property security level of 33h (supervisor write--supervisor read), the property and its value can only be viewed or modified by a client that is security equivalent to the supervisor.

Supervisor Privileges

Each bindery has an object named SUPERVISOR. The supervisor is the network administrator and is given special access to the bindery. The supervisor, for example, is the only object that can create, delete or rename bindery objects. It is also the only object that can close and open the bindery for archiving purposes. The supervisor may grant supervisor privileges to other objects through the security equivalence feature.

Security Equivalence

The security equivalence feature allows a bindery object to be granted the same access rights as another object. For example, using security equivalence, the supervisor can grant supervisor rights to other objects. Once one bindery object is created and given detailed security assignments, another object needing the same security can be given security equivalence to the first object.

SECURITY_EQUALS Property

A list of objects that a client is security equivalent to is stored in the client's SECURITY_EQUALS property. This property is used when determining a client's access rights to the file server. When an object logs in to a file server the object's access rights are logically OR'ed together with the access rights possessed by the objects listed in its SECURITY_EQUALS property. When determining an object's directory access rights, the file server's directory parsing algorithm uses only the first 32 objects listed in this property. However, the bindery does allow the SECURITY_EQUALS set to grow beyond 32 objects.

Security equivalences are not transitive. As a result, an object is security equivalent only to the objects explicitly listed in its SECURITY_EQUALS property. In other words, an object's security equivalence is not extended to the equivalences held by an object to which it is security equivalent. For example, if MARY is security equivalent to JOHN and JOHN is security equivalent to SUPERVISOR, MARY is not automatically security equivalent to SUPERVISOR.

User Groups

The security equivalence feature is also useful in defining user groups. User groups are a means of logically organizing users into workgroups. This allows the system supervisor to simplify the security process. Generally, a user group is assigned specific directory access rights, then users are added to the user group.

When a user is added to a user group with the NetWare utility SYSCON, the user's object ID is added to the group's GROUP_MEMBERS property. The group's object ID is then added to the user's GROUPS_I'M_IN and SECURITY_EQUALS properties.

The group's GROUP_MEMBERS property and the user's GROUPS_I'M_IN property are used together to logically define a group. The
SECURITY_EQUALS property is used to ensure that a group member has the directory rights assigned to the group's object.

Bindery Objects

The bindery is a collection of named objects. An object can be a user, user group, file server, print server, print queue, or any other logical or physical entity on the network that has been given a name. Each object has an object name, an object type, an object ID, a dynamic flag and an object security associated with it. The object name and type uniquely identify the bindery object. The object's ID number, a numeric value that is assigned to the object when it is created, also uniquely identifies the object. The dynamic flag indicates whether the object is dynamic or static. The object security determines whether other objects can access it.

TABLE 16. Object Structure

Field	**Type**
Object Name	char [0-47]
Object ID	uint32
Object Typs	uint16
Properties Flag	uint8
Object State	uint8
Object Security	uint8

Object Name

An object's name is 1 to 47 characters long and must contain only printable characters (21h through 7Dh). Control characters, spaces, slashes (/), backslashes (\), colons (:), semicolons (;), commas (,), asterisks (*), question marks (?) and tildes (~) are invalid characters. Object names are recorded in uppercase in the bindery.

The asterisk (*) and question mark (?) are wildcard characters and can be used to specify a search pattern when scanning for bindery objects. An asterisk (*) matches 0 or more characters. Thus the pattern ***** matches any object name. A question mark (?) matches exactly one character. Thus the pattern **??** only matches two character object names.

An object is uniquely identified by the object's name and the object's type. Thus, two objects with the same name and different types may exist in the same bindery.

Object ID

Each bindery object is given a unique ID when it is created. This ID is a number which is used as a simple method of identifying an object without specifying the object's name and type. The object ID is only guaranteed to be unique within one file server's bindery. Also, the ID number does not identify the file server's bindery in which an object is defined.

Object Type

The object type identifies the object as a specific type of client or resource. NetWare's currently defined object types are listed in Table 17.

TABLE 17. Object Types

Description	Object Type
Unknown	0x0000
User	0x0001
User Group	0x0002
Print Queue	0x0003
File Server	0x0004
Job Server	0x0005
Gateway	0x0006
Print Server	0x0007
Archive Queue	0x0008
Archive Server	0x0009
Job Queue	0x000A
Administration	0x000B
SNA Gateway	0x0021
Remote Bridge Server	0x0024
Synchronization Server	0x002D
Archive Server (Dynamic SAP)	0x002E
Advertising Print Server	0x0047
Btrieve VAP	0x0050
Print Queue User	0x0053
NVT Server	0x009E
Wild	0xFFFF

The WILD object type (FFFFh) is not an actual object type that would be associated with a bindery object. It is used when scanning the bindery if the actual object type is not known or not relevant to the caller. The WILD object type may not be used when adding an object to the bindery.

Object types up to 8000h are reserved by Novell for well-known object types. Other object types may be defined and used by third-party applications as needed. If a general purpose object type is needed, contact Novell's API Consulting Group for assignment of a well-known object type.

Each object type has a set of specifications associated with it. An object created with one of Novell's defined object types must adhere to the specifications that are defined for that particular object type. For example, an object that is declared as a user must have a password property to log in to the file server. A user is also required to have an ACCOUNT_BALANCE property to log in if the file server is charging for its services.

Adhering to object type specifications is particularly important for third-party value added servers. For example, a server that declares itself as a print server of object type 0007h is expected to provide the same services and have the same client/server protocol as the print server developed by Novell.

Some value added servers need to be recorded in the bindery as both a resource and a client. For example, a server that uses the Service Advertising Protocol (SAP) to advertise its existence on the network, and that also accesses other
NetWare APIs such as accounting or queue management, needs to be defined in the bindery as two separate objects, each with different object types.

One object type is used for advertising the server, while the other type allows the server to log in to the file server as a client and use Accounting or Queue
Management.

An advertising server is created as a dynamic object, meaning it is an object that is created and deleted frequently. Also, dynamic objects are deleted from the bindery when the file server is initialized. A client object is not dynamic in nature. Clients are static objects that are not created and deleted frequently, but are created once and deleted by the supervisor only when the client no longer needs access to the file server.

Furthermore, the file server does not perform security checks on servers that are advertising. Any server can advertise its services using the SAP. The lack of security checks means that an advertising server's access to the file server is very limited. Therefore, the server must also be defined as a client object in each file server's bindery that it needs further access to. In order to gain further access, the value added server must log in to the file server as a client.

A number of security checks are performed on the client object before it is allowed access to the file server. Once the client passes the security checks, it has access to other services provided by the file server such as file I/O, queue
management, and accounting.

A print server is an example of a value added server that requires more than one object type. As indicated in the list of NetWare's defined object types there is an advertising print server (0047h), a print queue (0003h) and a static print server (0007h) object type. All three object types are assigned to and used by a Novell print server.

The advertising print server is used to advertise that the physical print server is up and running. The print queue is used as a queuing mechanism between the print server and its clients. The static print server type is used by the physical print server to log in to the file server and access its services such as file I/O, queue management, and accounting.

Properties Flag

The properties flag indicates whether one or more properties are associated with an object.

Object State

The object state indicates the expected life time of an object. An object is either static (00h) or dynamic (01h). A static object is a long term object that must be explicitly deleted from the bindery when it is no longer useful. An object of type user is a good example of a static object. A dynamic object is one that is created and deleted frequently, and therefore, is deleted when the file server is brought down and re-initialized. The dynamic flag is used by advertising servers which need to be dynamically added to and deleted from other file servers' binderies.

Object Security

The object security controls the read and write access of others to the bindery object. The low-order nibble determines who can read (scan for and find) the object. The high-order nibble determines who can write to (add properties to or delete properties from) the object. Table 18 defines the values for each nibble.

TABLE 18. Object Security Levels

Hex	Binary	Access	Description
0	0000	Anyone	Access allowed to all clients, even if the client has not logged in to the file server.
1	0001	Logged	Access allowed only to clients who have logged in to the file server.
2	0010	Object	Access allowed only to clients who have logged in to the file server with the object's name, type and password.
3	0011	Supervisor	Access allowed only to clients logged in to the server as supervisor or as an object with supervisor security equivalence.
4	0100	NetWare	Access allowed only to NetWare.

For example, an object security of 31h (supervisor write--logged read) indicates that any user logged in to the file server can find the object, but only the supervisor can add a property to the object.

Bindery Properties

Properties are named pieces of information that are attached to objects. Each bindery object may have one or more properties associated with it. For example, the user DAN (object type 0001h) might have associated with it the properties GROUPS_I'M_IN, ACCOUNT_BALANCE and PASSWORD.

Note that GROUPS_I'M_IN is not the name of a user group to which the object belongs. It is only the name of one category of information associated with that object. In the same way, ACCOUNT_BALANCE is not an actual numerical balance, and PASSWORD is not an actual password. Properties only identify categories of information associated with the object.

Each property has a value associated with it. For example, a value associated with the property GROUPS_I'M_IN must be the object ID of a user group to which DAN belongs. The value of the property ACCOUNT_BALANCE must be user DAN's current balance. The value of the property PASSWORD must be DAN's login password (perhaps COMPILE). Although a property can only have one value, the value can contain multiple segments (each segment being 128 bytes long).

Property Categories

Properties fall into one of two categories: set property or item property. The category determines the type of values the property can have and the number of values.

Set Property

A set property has associated with it a list or set of object IDs that are contained in the property's value. The property value of a set property can consist of multiple segments where each segment may contain up to 32 object IDs. (Each object ID is 4 bytes. Therefore, the maximum number of object IDs that one 128-byte segment can hold is 32.)

A user's GROUPS_I'M_IN property is a set property. The property value associated with the GROUPS_I'M_IN property contains the object IDs of the groups to which the user, (DAN, in the above example), belongs. The values of set properties are always object IDs grouped into one or more 128-byte segments. It is important that set property values do not contain anything other than object IDs because the operating system interprets each segment of a set property value to be an array of object IDs.

When an object is deleted from a set property, the operating system searches consecutive segments of the property's value for an object ID that matches the object ID of the member to be deleted. When the member is found it is deleted. The remaining IDs in the segment are shifted and the last previously used slot in the segment is filled with zeros. This ensures that IDs within a segment are packed. However, IDs are not packed between segments.

Item Property

An item property has associated with it a property value which can contain any type of data (typically contains a numeric value, a string or a structure). The bindery attaches no significance to the contents of an item property's value. An item property's value is defined and interpreted by APIs and by application programs, not by the bindery process.

A user's PASSWORD and ACCOUNT_BALANCE properties are both examples of item properties. The PASSWORD property is defined to have only one segment and contains an encrypted password. The ACCOUNT_BALANCE property value contains a monetary balance in the first 4 bytes and a credit limit in the next 4 bytes of the 128-byte segment. The rest of the segment is filled with zeros.

Property Fields

Each property has a property name, property state, property type and a property security associated with it. The property name identifies an object's property. The property flags field contains two flags: the static/dynamic flag and the item/set flag. The static/dynamic flag indicates the expected life time of a property. The item/set flag specifies the type of information that is stored in the property's value. The property security determines who has access to the property.

Property Name

A property's name is 1 to 15 characters long and must contain only printable characters (21h through 7Dh). Control characters, spaces, slashes (/), backslashes (\), colons (:), semicolons (;), commas (,), asterisks (*), question marks (?) and tildes (~) are invalid characters. Property names are recorded in uppercase in the bindery.

The asterisk (*) and question mark (?) are wild characters and can be used to specify a search pattern when scanning a bindery object's properties. An asterisk (*) matches 0 or more characters. Thus the pattern ***** matches any property name. A question mark (?) matches exactly one character. Thus the pattern **??** only matches two character property names.

Property State

The property state flag indicates the expected life time of a property. A property is either static or dynamic. A static property is a long term property that must be explicitly deleted from the bindery when it is no longer useful. The ACCOUNT_BALANCE property is an example of a static property. A dynamic property is one that is created and deleted frequently and, therefore, is deleted when the file server is brought down and re-initialized. The ACCOUNT_HOLDS property is an example of a dynamic property.

Bit 0 is the static/dynamic flag and is defined as follows:

F3.eps ,

Property Type

The item/set flag indicates whether the property's value contains an item or a set of object IDs. The contents of item property values are defined and interpreted by applications or by application program interfaces (APIs). The contents of set property values are interpreted by the bindery process as a series of object ID numbers, each 4 bytes long.

Bit 1 is the item/set flag and is defined as follows:

F0.eps ,

Property Security

The property security controls the read and write access of others to the property. The low-order nibble determines who can read (scan for and find) the property. The high-order nibble determines who can write to (modify) the property's value. Table 19 defines the values for each nibble.

TABLE 19. Property Security Levels

Hex	Binary	Access	Description
0	0000	Anyone	Access allowed to all clients, even if the client has not logged in to the file server.
1	0001	Logged	Access allowed only to clients who have logged in to the file server.
2	0010	Object	Access allowed only to clients who have logged in to the file server with the object's name, type and password.
3	0011	Supervisor	Access allowed only to clients logged in to the server as supervisor or as an object with supervisor security equivalence.
4	0100	NetWare	Access allowed only to NetWare.

For example, a property security of 31h (supervisor write--logged read) indicates that any user logged in to the file server can find the property, but only the supervisor can modify the property's value.

NetWare's Properties

NetWare has defined many different properties for many different purposes. Some properties contain further information about an object (e.g., the IDENTIFICATION property) while others are used to administer the network security system (e.g., the PASSWORD and SECURITY_EQUALS property). All properties defined by Novell which contain numeric data (including set properties) are stored in hi-lo order.

Table 6 lists all of the properties defined by NetWare. The bindery properties are described before the table. The other properties are described in the appropriate chapter for the API.

GROUP_MEMBERS Property. The GROUP_MEMBERS property contains a list of users that are members of a user group. This property is attached to user group objects. The GROUP_MEMBERS property is a static/set property and has object read and supervisor write security access levels.

GROUPS_I'M_IN Property. The GROUPS_I'M_IN property contains a list of user groups that a user is a member of. This property is attached to user objects. The GROUPS_I'M_IN property is a static/set property and has logged read and supervisor write security access levels.

IDENTIFICATION Property. The IDENTIFICATION property contains a user or user group's full name. This property is attached to user and user group objects. The IDENTIFICATION property is a static/item property and has logged read and supervisor write security access levels.

OPERATORS Property. The OPERATORS property contains a list of objects that are authorized file server console operators. This property is attached to the file server's object. The OPERATORS property is a static/set property and has supervisor read and supervisor write security access levels.

PASSWORD Property. The PASSWORD property contains an object's encrypted password. This property is attached to user objects. Any object that logs in to the file server is required to have the PASSWORD property. The PASSWORD property can be created with the NWChangeObjectPassword function call. The PASSWORD property is a static/item property and has NetWare read and NetWare write security access levels.

SECURITY_EQUALS Property. The SECURITY_EQUALS property contains a list of objects that an object is security equivalent to. This property is attached to user and user group objects. The SECURITY_EQUALS property is a static/set property and has object read and supervisor write security access levels.

TABLE 20. NetWare-Defined Properties

Property Name	Object Type	Property Flags		Security		Application Interface Programming (API)
		Static/ Dynamic	Item/Set	Write	Read	
ACCOUNT_ BALANCE	User	Static	Item	3	2	Accounting
ACCOUNT_ HOLD	User	Dynamic	Item	3	2	Accounting
ACCOUNT_ SERVERS	File Server	Static	Set	3	1	Accounting
ACCT_ LOCKOUT	File Server	Static	Item	3	3	Security
BLOCKS_ READ	File Server	Static	Item	3	1	Accounting
BLOCKS_ WRITTEN	File Server	Static	Item	3	1	Accounting
CONNECT_	File Server	Static	Item	3	1	Accounting

TIME						
DISK_STORAGE	File Server	Static	Item	3	1	Accounting
GROUP_MEMBERS	User Group	Static	Set	3	1	Bindery
GROUPS_I'M_IN	User	Static	Set	3	1	Bindery
IDENTIFICATION	User	Static	Item	3	1	Bindery
LOGIN_CONTROL	User	Static	Item	3	2	Security
NET_ADDRESS	File Server	Dynamic	Item	4	0	Service Advertising
NODE_CONTROL	User	Static	Item	3	2	Security
OLD_PASSWORDS	User	Static	Item	3	3	Security
OPERATORS	User	Static	Set	3	3	Bindery
PASSWORD	User	Static	Item	4	4	Bindery
Q_DIRECTORY	Print Queue	Static	Item	3	3	Queue Management
Q_OPERATORS	Print Queue	Static	Set	3	1	Queue Management
Q_SERVERS	Print Queue	Static	Set	3	1	Queue Management
Q_USERS	Print Queue	Static	Set	3	1	Queue Management
REQUESTS_MADE	File Server	Static	Item	3	1	Accounting
SECURITY_EQUALS	User	Static	Set	3	2	Bindery
USER_DEFAULTS	Supervisor	Static	Item	3	1	Security