the mysteries of binding and propagating

by Dave Cottle

Have you ever wondered how a NetInfo™ domain gets linked into a domain hierarchy? Or how clone servers really work? To find the answers to these questions, you need to delve into the inner workings of NetInfo. Take a deep breath and get ready to explore the wondrous world of binding and propagating.

a look at the basics

First, a quick look at some basic information about NetInfo domains. For many, this will be a refresher. Here are the major points to keep in mind as you unravel these mysteries:

- The information in a NetInfo domain is stored in files contained in a subdirectory of /etc/netinfo. Each domain has its own directory named tag.nidb. All NeXT™ computers have a directory for the local domain called local.nidb. Computers that serve additional domains have additional directories.

- The daemon process netinfod (for NetInfo daemon) serves information from a specific domain. When you use ps to examine system processes, you see processes of the form netinfod tag, where tag matches the first part of the name of a directory under /etc/netinfo. The process netinfod local serves information from the database tagged local for the local domain. Computers that serve multiple domains run multiple netinfod processes.

- NetInfo domains are organized into a hierarchy, with the root domain ("/") at the top level and local domains at the bottom level. Requests for information always begin in the local domain and are passed up the hierarchy until the information is found or the root domain is reached.

- Information within a domain is organized into a hierarchy of NetInfo directories. The top-level NetInfo directory is also called root ("/").

- A clone NetInfo server provides access to a read-only copy of a domain.

Figure 1 shows a three-level hierarchy of NetInfo domains. Each domain is identified by name (within each box) and database tag (to the left of each box). This illustration shows no indication of which computer serves which domain, other than the local domains. If you trace the hierarchy from bottom to top, you can see that the host everest has access to the information in its local domain, the acctng domain, and the root domain ("/").

figure 1:   three-level NetInfo domain hierarchy

A48_3-level.tiff ¬

the serves property

The relationship between NetInfo domains in the domain hierarchy is determined by the serves property. Host entries are stored in the /machines directory of a domain, and it's here you'll find the serves property. The serves property identifies the domains served by the associated host. Figure 2 shows the partial contents of the domains in a two-level hierarchy made up of three computers: etna, olympus, and everest. The master server for the root domain is olympus, and everest is a clone server.

As you can see from the illustration, a serves property has values in the form domain/tag, which indicates that this host serves the domain domain from a database tagged tag. Domains can be designated with a relative name, where . means the current domain and .. means the parent domain.

For example, the host entry for olympus in the root domain indicates that it serves the domain olympus from a database tagged local, and also serves the current domain (.) from a database tagged network.

Any given domain will have at least one host entry with a serves property for the current domain (./tag) and might have entries for the parent domain (../tag) or child domains (domain/tag).

binding

Equipped with this information, we can take a look at how the domain hierarchy is built at boot time. Here is exactly what happens:

1. As a NeXT computer boots, the daemon process nibindd (NetInfo binding daemon) is started. The nibindd process searches /etc/netinfo looking for subdirectories named tag.nidb. For each directory it finds, nibindd starts a netinfod process.

   In the example shown in figure 2, the nibindd daemon on etna finds only one subdirectory in /etc/netinfo-local.nidb. As a result, the single process netinfod local is started.

figure 2: the serves property in a two-level domain hierarchy

A49_2-level.tiff ¬

2. As a netinfod process starts up, it searches the /machines directory in its NetInfo database looking for entries with a serves property that has a value in the form ../tag. A serves property with such a value indicates that the associated host serves the parent domain (..) from a database tagged tag.

On etna, the host entry /machines/broadcasthost in the local domain has a serves property with the value ../network. This indicates that broadcasthost serves the parent domain from a database tagged network.

3. For every host entry found with a serves property value ../tag, a bind request is sent to the associated Internet address. The bind request includes the tag of the parent domain, the tag of the current domain, and the Internet address of the host making the request.

The host etna sends a bind request to the Internet address for broadcasthost-255.255.255.255. This address is the special broadcast address, which means that the request will be sent to every computer on the local network (including etna).

4. The recipients of the bind request pass the message on to their nibindd daemon. The nibindd process checks to see if there is a netinfod process running that serves NetInfo data from a database that has a tag matching the parent domain tag in the bind request. If a match is found, the request is passed on to the appropriate netinfod process.

The nibindd process on olympus, as well as the one on everest, finds that there is indeed a netinfod process running for a database tagged network.

5. The receiving netinfod process searches its own /machines NetInfo directory looking for entries that include the Internet address of the host making the request. The entry is also checked to see if it has a serves property with a value in the form domain/tag, where tag matches the tag of the domain that sent the bind request. If a host entry is found that meets these requirements, a message is sent back to the host that made the request, indicating that the receiving domain can serve as the parent of the requesting domain.

The netinfod process on olympus finds a host entry for etna with an Internet address that matches the one in the bind request. The serves property for this host entry has the value etna/local, which matches the tag in the bind request. All the requirements are met, so olympus sends a message to etna indicating that it can serve the parent domain. Because everest is serving an exact copy of the domain on olympus, the netinfod process finds the same information and also sends a message to etna.

6. The domain that initiated the request binds to the first server that responds. From then on, whenever information is needed from theparent domain, the request is sent directly to the server that responded first to the bind request. The first request is for the Internet addresses of all servers of the parent domain. The /machines directory of the parent domain is searched again, this time for serves properties with the value ./tag, where tag is the tag of the parent domain. The host serving the child process stores the Internet addresses of all parent domain servers (including the one it's bound to) for possible future use.

   In this case, etna binds to either olympus or everest, depending on which server responds first. Once bound, a search of the parent domain finds that the host entry for olympus has a serves property with the value ./network, as does the host entry for everest. The Internet address of each is returned to etna.

Here's the whole thing in a nutshell: When the server for a NetInfo domain is started, it searches its database for hosts that are potential servers of its parent domain. A request is sent to each potential parent server, asking if that server can act as the parent. The child domain binds to the first server that sends a positive response.

rebinding

As you might have suspected, that isn't the end of the story. Unless there's only a single server for the parent domain, a child domain rarely stays bound to its parent forever. Sometimes, a parent server gets bogged down with other activities and can't respond quickly enough to information requests. A parent server might also become unavailable if it's turned off or disconnected from the network. If a child sends a request and doesn't get a response within a set period of time, it sends out another bind request. This time, the request is sent to the Internet addresses that the child stored in the last step of the binding process. Again, whichever server responds first becomes the parent domain server (this might even be the same server as before, if it's able to respond quickly enough).

An interesting side effect of the binding process is that a new clone server won't get used until the client computers are rebooted. Because the Internet addresses for all the parent domain servers are determined at boot time, any servers for the parent domain added after initial binding won't be recognized.

One other situation can cause a domain to rebind to its parent: a write request. As long as the child requests only read access to information, it doesn't matter if it's bound to the master server or a clone. However, if it makes a write request (such as when a network user changes passwords), it must be bound to the master server. Remember, clones serve a read-only copy of the database. How does a clone know it's a clone? By examining the master property. Each NetInfo domain has a master property in the root directory. The value of this property identifies the master server of the domain. Looking at figure 2 again, you see that the value of the master property in the root domain is olympus/network. This indicates that the master copy of the domain is served by the host olympus from the database tagged network.

propagating
At this point you may find yourself wondering, "If a clone server only has read access to the database, how does it maintain consistency with the master server?" Good question. Whenever a request to the master server results in a change to the database, the request is immediately passed on to all clone servers, where the change is duplicated. Modifications to a domain are therefore propagated almost instantly.

"But," you say, "what if the clone server is unavailable for a time and misses some changes?" Another good question. Whenever the daemon process for a clone server is started, it compares its copy of the database with the copy on the master server (for the curious, it uses checksum). If it finds that its database doesn't match the master, it loads a complete copy of the master database into its own /netinfo directory. The same comparison is made at least every half hour.

With all write requests being passed to clone servers and periodic consistency checks made, you can feel pretty secure that your clone databases will always be in sync with the master.

the nitty-gritty

So far, we've been talking about a child domain binding to its parent. Now it's time to come clean: That's still not the whole story. When we talk about a child domain binding to its parent, we're talking specifically about a netinfod process binding to its parent. Many other processes and programs on a NeXT computer need access to the parent NetInfo domain, and each can bind separately.

For example, requests for information are handled by the lookupd daemon, which searches not only NetInfo but also the DNS and NIS. Because lookupd never writes information, it never needs to bind to the master domain server. In contrast, when you use PrintManager to export a printer to the network, the application must have write access to the appropriate domain. PrintManager must bind to the master server and can do so even if netinfod is bound to a clone server.

in conclusion

Now you've had a peek inside the mysterious world of NetInfo binding and propagating. You've seen how children find their parents, how clones come into play, and how the different databases are kept consistent. You might feel you're gotten too many details, but tuck the data away in the back of your mind. Knowledge of the underpinnings of NetInfo can come in handy when troubles arise (or when you want to impress your friends).