The FNS Model 2

This chapter describes the FNS naming model from several perspectives. Some aspects are graphically presented as a way of helping you visualize FNS concepts.

FNS Architectural Model	*page 11*
User's View	*page 17*
Application View	*page 19*
API Usage Model	*page 22*

FNS Architectural Model

The primary services provided by a federated naming system are mapping a composite name to a reference and providing access to attributes associated with a named object. This section defines the elements of the FNS naming model. The FNS naming model is the same as the XFN naming model.

References

A reference is the information on how to reach an object. It contains a list of addresses. An address identifies a communication endpoint. Multiple addresses identify multiple communication endpoints for a single conceptual object or service. For example, a list of addresses may be required because the object is distributed or because the object can be accessed through more than one communication mechanism.

Note – FNS cannot guarantee specific properties of addresses such as their stability, validity, or reachability. A client may be able to look up a name but not be able to use the returned reference because the client may not have support for any of the necessary communication mechanisms, or may lack the necessary network connectivity to reach the address. Further, the address may be invalid from that origin, or stale; these issues are the province of convention between the name's binder, the clients, and the service provider specified in the address.

Contexts

An XFN context is an object that exports the XFN base context programming interface. A context contains, an atomic name is bound to an XFN reference, as shown in Figure 2-1. An atomic name may have zero or more attributes. Contexts are at the heart of the lookup and binding operations, described extensively in Chapter 10, “Interfaces for Writing FNS Applications.”

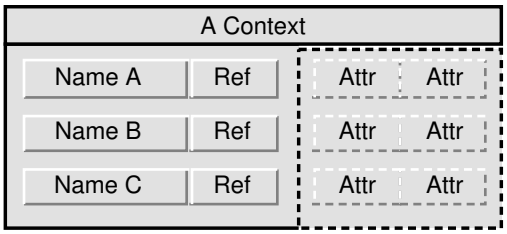

Figure 2-1 An XFN Context

Attributes

Each named object is associated with zero or more attributes. Thus, attributes are optional, as shown by the dotted lines in Figure 2-1 on page 12. Each attribute has a unique attribute identifier, an attribute syntax, and a set of zero or more distinct attribute values. XFN defines the base attribute interface for examining and modifying the values of attributes associated with existing named objects. These objects may be contexts or other types of objects. Associated with a context are syntax attributes that describe how the context parses compound names.

Compound Names

A compound name represents a sequence of one or more atomic names. An atomic name in one context object can be bound to a reference to another context object of the same type, called a subcontext. Objects in the subcontext are named using a compound name. Compound names are resolved by looking up each successive atomic name in each successive context.

A familiar analogy for UNIX users is the file naming model, where directories are analogous to contexts, and path names serve as compound names. Furthermore, contexts can be arranged in a "tree" structure, just as directories are, with the compound names forming a hierarchical naming system.

- **UNIX example**: `usr/local/bin`. UNIX atomic names are ordered from left to right and are delimited by slash ("`/`") characters. The name `usr` is bound to a context in which `local` is bound. The name `local` is bound to a context in which `bin` is bound.
- **DNS example**: `sales.Wiz.COM`. DNS atomic names are ordered from right to left, and are delimited by dot ("`.`") characters. The domain name `COM` is bound to a context in which `Wiz` is bound. `Wiz` is bound to a context in which `sales` is bound.
- **X.500 example**: `c=us/o=wiz/ou=sales`. An X.500 atomic name comprises an attribute type and an attribute value. Atomic names are known as *relative distinguished names* in X.500. In this string representation X.500 atomic names are ordered from left to right, and are delimited by slash ("`/`" characters. An attribute type is separated from an attribute value by an equals ("`=`") character. Abbreviations are defined for commonly

- used attribute types (for example, "c" represents country name). The country name `US` is bound to a context in which `wiz` is bound. The organization name `wiz` is bound to a context in which the organizational unit name `sales` is bound.

Figure 2-2 shows an example of a hierarchical naming system with compound names.

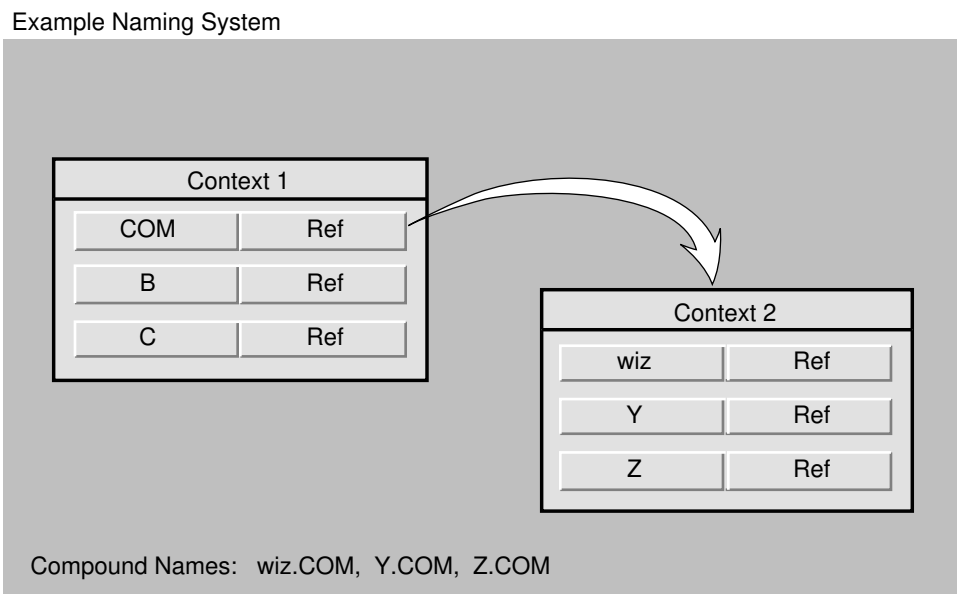

Figure 2-2 Hierarchical Name System With Compound Names

Composite Names

Fundamental to XFN and the notion of "federation" is the concept of a composite name. An XFN composite name names an object in the federated namespace exported by the underlying name systems. A composite name, then, is a name that spans multiple name systems. It consists of an ordered list of zero or more components. Each component is a name from the namespace of a single naming system. Composite name resolution is the process of resolving a name that spans multiple name systems. Chapter 11, "XFN Composite Names," and Appendix A, "XFN Composite Names Syntax" supply more detail about composite names.

A federated naming system is an aggregation of autonomous name systems that cooperate to support name resolution of composite names through a standard interface. Each member of a federation has autonomy in its choice of operations other than name resolution. Individual name systems are responsible for the resolution of each component.

Components are slash-separated (/) and ordered from left to right, according to XFN composite name syntax. For example, the composite name

```
sales.Wiz.COM/usr/local/bin
```

has two components, a DNS name (`sales.Wiz.COM`) and a UNIX path name (`usr/local/bin`).

Figure 2-3 on page 16 shows an example of a federated naming system with composite names. Note that the position of the name within a context has no inherent significance in this illustration.

XFN Links

An XFN link is a special form of a reference that is bound to an atomic name in a context. Instead of an address, a link contains a composite name. Many name systems support a native notion of link that may be used within the naming system itself. FNS does not specify whether there is any relationship between such native links and XFN links.

"XFN Links" on page 161 describes links in detail.

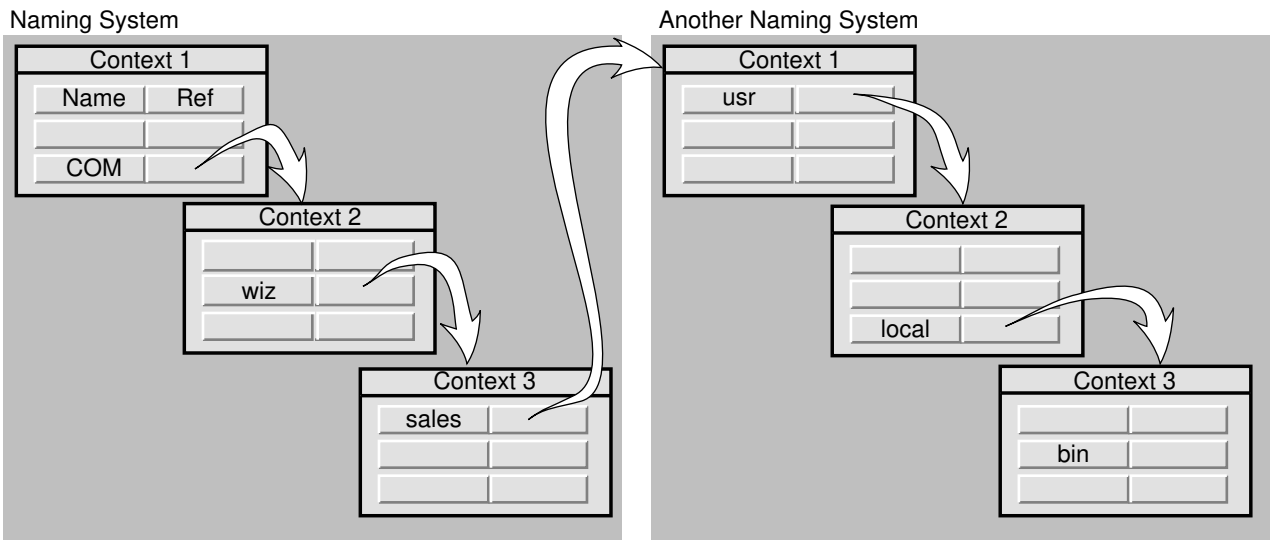

Figure 2-3 Federated Naming System With Composite Names

Initial Context

Every FNS name is interpreted relative to some context, and every FNS naming operation is performed on a context object. The *initial context* object provides a starting point for the resolution of composite names. The FNS interface provides a function that allows the client to obtain an initial context.

The policies described in Part 2 of this guide specify a set of names that the client can expect to find in this context and the semantics of their bindings. This provides the initial pathway to other FNS contexts.

User's View

Users experience federated naming through applications, as shown in Figure 2-4. The user can interact with FNS-aware applications in a simple, intuitive, and consistent manner. Typically, the user does not need to compose or know the full composite name of the objects because the application takes care of constructing the composite names.

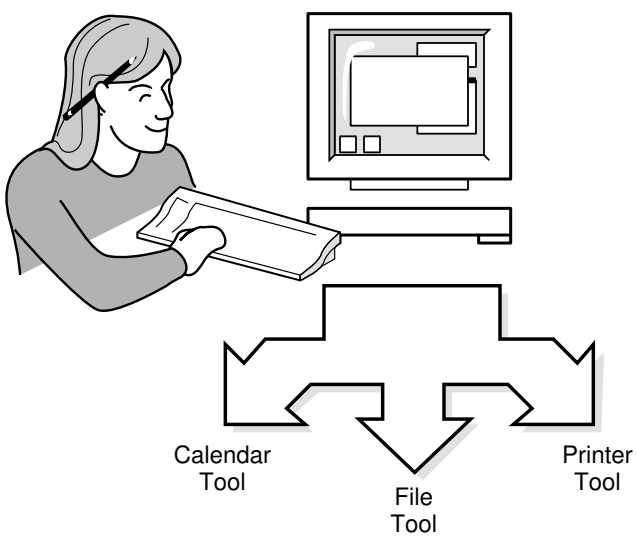

Figure 2-4 User View of FNS

For example, if the application is expecting you to type a user name, the application may prepend the `user/` string to names that you enter. Furthermore, if the application needs to name one of the user's services, such as the user's default fax machine, it can append the rest of the name, `/service/fax`, to the input supplied. Hence, a fax tool may take as input `jsmith` and compose the name `user/jsmith/service/fax` for the default fax of the user `jsmith`. Policies such as `user/` and `service/` are described in Part 2, "FNS Policies."

Similarly, to access a person's calendar, you just need to type the person's user name. The application takes the input, `jsmith`, and uses it to construct the composite name of the object, in this case, `user/jsmith/service/calendar`, as shown in Figure 2-5.

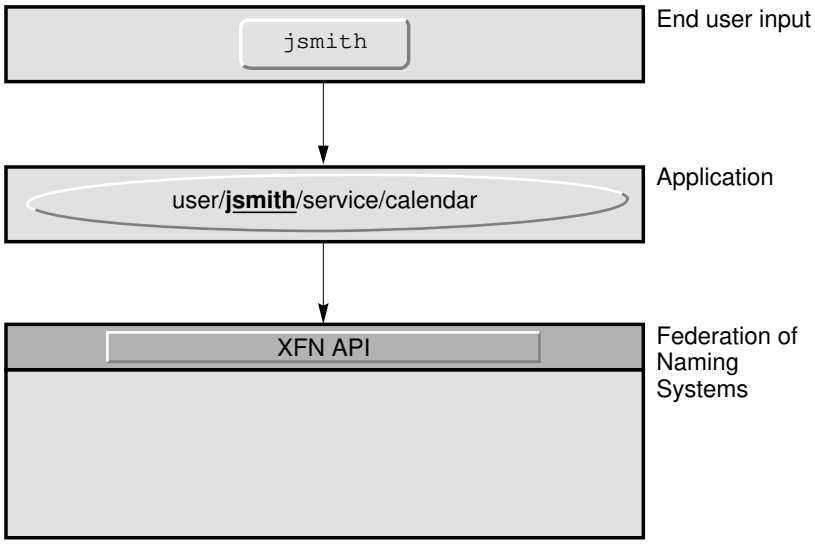

Figure 2-5 User Interaction with FNS

File System View

Users and applications also experience federated naming through the file system. The initial context is located under `/xfn` in the root directory. For example, user `jsmith`'s "to do" file has the FNS name, `user/jsmith/fs/to_do`. To read this file, a person could type:

```
% cat /xfn/user/jsmith/fs/to_do
```

Applications would access the files under `/xfn` just as they would any other files. Applications do not need to be modified in any way, nor do they need to use the XFN API.

Application View

The way that client applications interact with FNS to access different naming systems is illustrated in a series of figures. Figure 2-6 shows an application that uses the XFN API and library.

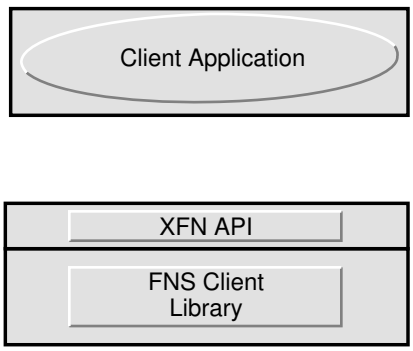

Figure 2-6 Client Application Interaction With FNS

Figure 2-7 shows the details beneath the API. A name service that is federated is accessed through the FNS client library and a *context shared object module*. This module translates the FNS calls into naming-service-specific calls.

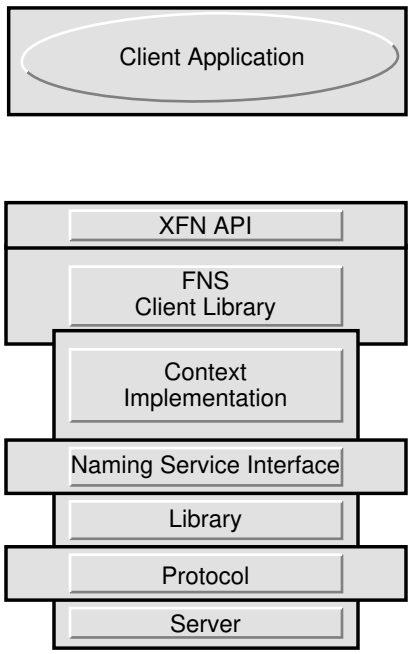

Figure 2-7 Details Beneath XFN API

X.500, DNS, and NIS+ are the name services that have been federated in the example shown in Figure 2-8.

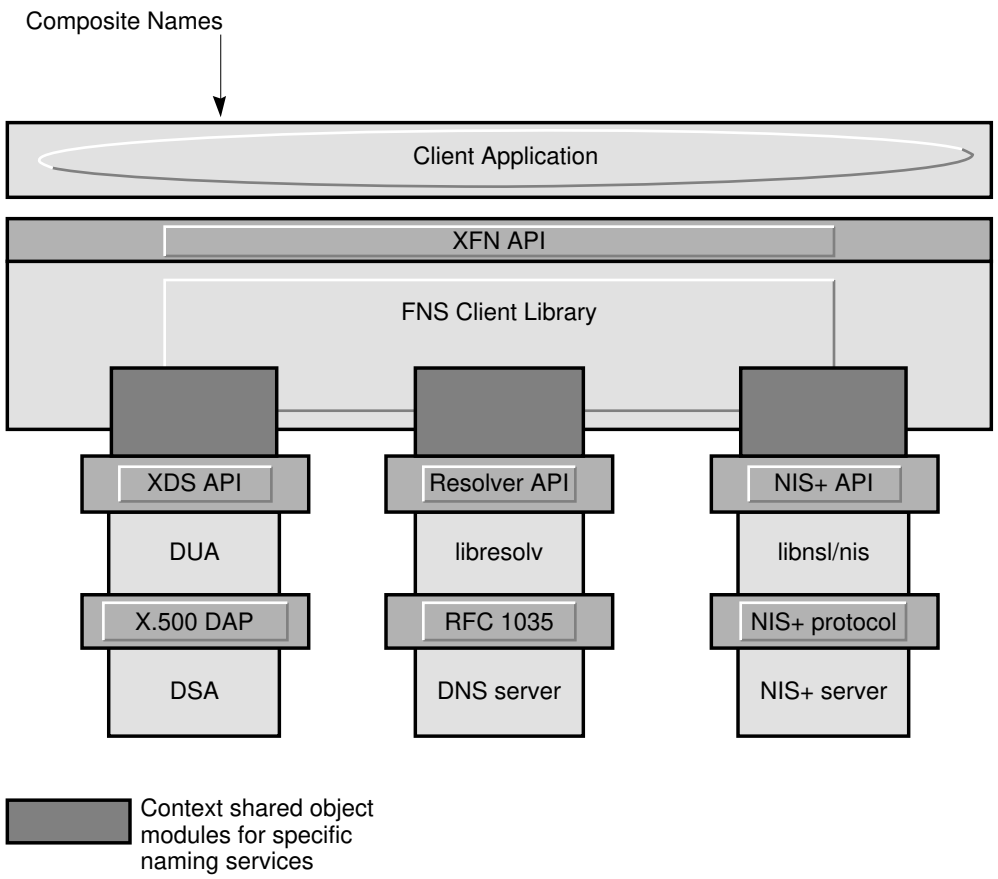

Figure 2-8 FNS Implementation Examples

2

API Usage Model

Many clients of the XFN interface are only interested in lookups. Their usage of the interface amounts to:

- Obtaining the initial context
- Looking up one or more names relative to the initial context

Once the client obtains a desired reference, it constructs a client-side representation of the object from the reference. This need not be code within the application layer but may be code inside the service layer. For example, RPC services can provide clients with a means of constructing client-side handles from a composite name for the service, or from a reference containing an RPC address for the service. After getting this handle, the client performs all further operations on the object or service by supplying the handle.

This is the basic model of how the XFN interface is expected to be used. The enterprise policies presented in Chapter 4 further encourage a bind/lookup model for how services and clients may rendezvous through the use of the name service.