NAME

fns – overview of FNS

DESCRIPTION

Federated Naming Service (FNS) provides a method for federating multiple naming services under a single, simple interface for the basic naming operations. The service supports resolution of *composite* names — names that spans multiple naming systems — through the naming interface. In addition to the naming interface, FNS also specifies *policies* for composing names in the enterprise namespace. See **fns_policies**(5) and **fns_initial_context**(5).

Fundamental to the FNS model are the notions of composite names and *contexts.* A context provides operations for:

- associating (binding) names to objects
- resolving names to objects
- removing bindings, listing names, renaming and so on.

A context contains a set of names to reference bindings. A reference contains a list of communication end-points. Every naming operation in the FNS interface is performed on a context object.

The FNS is formed by contexts from one naming system being bound in the contexts of another naming system. Resolution of composite name proceeds from contexts within one naming system to those in the next until the name is resolved.

XFN

XFN is *X/Open Federated Naming* . The programming interface and policies that FNS supports are specified by XFN. See **xfn**(3N) and **fns_policies**(5).

Composite Names

A composite name is a name that spans multiple naming systems. It consists of an ordered list of components. Each component is a name from the namespace of a single naming system. FNS defines the syntax for constructing a composite name using names from component naming systems. Individual naming systems are responsible for the syntax of each component.

The syntax for composite names is that components are composed left to right using the slash character ('/') as the component separator. For example, the composite name **.../Wiz.Com/site/Oceanview.East** consists of four components: **...**, **Wiz.COM**, **site**, and **Oceanview.East**. See **fns_policies**(5) and **fns_initial_context**(5) for more examples of composite names.

Why FNS?

FNS is useful for the following reasons:

- A single uniform naming interface is provided to clients for accessing naming services. Consequently, the addition of new naming services does not require changes to applications or existing naming services. Furthermore, applications that use FNS will be portable across platforms because the interface exported by FNS is XFN, a public, open interface endorsed by other vendors and X/Open.
- Names can be composed in a uniform way (i.e. it supports a model in which composite names are constructed in a uniform syntactic way and can have any number of

components).

- Coherent naming is encouraged through the use of shared contexts and shared names.

FNS and NIS+

In Solaris, the FNS implementation consists of a set of enterprise-level naming services implemented on top of NIS+, the enterprise-wide information service in Solaris. FNS provides the FNS interface for performing naming and attribute operations on FNS enterprise objects (organization, site, user, host and service objects). FNS stores bindings for these objects in NIS+ and uses them in conjunction with existing NIS+ objects.

The command **fncreate** creates NIS+ tables and directories in the NIS+ hierarchy associated with the domain of the host it executes on. The invoker of **fncreate** and other FNS commands is expected to have the necessary NIS+ credentials. See **nis+**(1) and **nisdefaults**(1). The environment variable NIS_GROUP of the process specifies the group owner for the NIS+ objects thus created. In order to facilitate administration of the NIS+ objects, NIS_GROUP should be set to the name of the NIS+ administration group for the domain prior to executing **fncreate** and other FNS commands. Changes to NIS+-related properties, including default access control rights, could be effected using NIS+ administration tools and interfaces after the context has been created. The NIS+ object name that corresponds to an FNS composite name can be obtained using **fnlookup** and **fnlist.**

SEE ALSO

fncreate(1M), **fnlookup**(1), **fnlist**(1), **nis+**(1), **nisdefaults**(1), **nisls**(1), **nischgrp**(1), **nischmod**(1), **nischown**(1), **xfn**(3N), **fns_initial_context**(5), **fns_policies**(5), **fns_references**(5)