# Symbolics Network File System (NFS) User's Guide

## Introduction to Symbolics Network File System (NFS)

Symbolics NFS is a user-transparent remote file access protocol. Symbolics NFS is a fully symmetrical implementation of the Sun Network File System protocol. It is built on the Sun Remote Procedure Call (RPC) protocol which in turn is built on the Sun eXternal Data Representation (XDR) standard, all as specified in the Sun Network File System Protocol Specification, Revision B of 17 February 1986. The transport medium used is UDP/IP. You must have Symbolics RPC and Symbolics IP/TCP installed as part of your system to use Symbolics NFS. Note that you only have to load Symbolics RPC separately on 3600-family machines.

The Symbolics NFS software package includes these systems: Network RPC, NFS Client, NFS Server, and NFS Documentation. Together, they provide the capacity for file transfer, direct file access, directory lookup, and file attribute modification. In addition, Symbolics NFS allows your Symbolics machine to act as an NFS client or server or both.

Because NFS protocol is UNIX oriented, you will find some behavioral differences from the other remote file protocols available on Genera. See the section "File System Access Through Symbolics NFS" for further information.

## Configuring Your System to Use Symbolics NFS

These sections tell you how to complete the Symbolics NFS installation. Be sure you have completed the installation procedure described in the cover letter for layered products before completing these steps.

Note that you can configure your system to be an NFS client, an NFS server, or both. To configure your system as an NFS Client, first load RPC and IP-TCP and then load the NFS Client. To configure your system as an NFS Server, first load RPC and IP-TCP, then the NFS Client and NFS Server systems. For online documentation of Symbolics NFS, load the NFS Documentation system.

**Note:** Ivory machines already have RPC in the world, so for an Ivory machine it is only necessary to load IP-TCP and the appropriate NFS system(s).

## Adding the FILE UDP NFS Service

You must add the service attribute `FILE UDP NFS` to the namespace object for each host that supports NFS as a file protocol in your namespace. This attribute makes NFS the preferred file protocol over FTP when Symbolics machines read and write files from the host.

You can add a service attribute with either the Command Processor command Add Services to Hosts or with the Namespace Editor. The Command Processor command is the easiest method and you can use it by specifying:

```
Add Services to Hosts FILE UDP NFS host1, host2, ...
```

where *host* is the name of your host.

You can use the Namespace Editor by:

> Invoking the Namespace Editor with the `:Edit Namespace Object` command, and add the service attribute `FILE UDP NFS` for each host that supports NFS as a file protocol in your namespace.
>
> Saving out each host object after you have changed it to make your change to the namespace permanent.

### Providing UNIX NFS File Access to a Symbolics Computer

In order for Genera to mount UNIX file systems through NFS, the file systems must be exported by a UNIX file server host. Genera must be able to mount user file systems in order to access user directories.

If you are not using Network Information Service at your site, Genera needs access to some files in the `/etc` directory of UNIX server hosts, so you need to export the root filesystem of these hosts. Network RPC tries to read the `/etc/passwd` and `/etc/group` files for the UNIX name lookup services it provides. The NFS system tries to read the `/etc/mtab` file on UNIX file servers in order to determine mount points.

Edit the `/etc/exports` file and use `exportfs` to allow NFS access to the UNIX filesystems. See the UNIX manual pages for the `exports` file and the `exportfs` program for more information.

### UNIX UID and GID Mapping

Symbolics NFS must establish a username to uid (user id) and groupname to gid (group id) mapping for all foreign server and client hosts. This mapping is necessary because file owners and file groups are communicated by means of UNIX uids and gids. The mapping is also used by the underlying Symbolics RPC implementation of UNIX RPC authentication. Unlike Genera, where users are identified by user name, UNIX RPC authentication identifies users requesting services by user and group id.

Symbolics NFS uses the following mechanisms in descending order to determine uid and gid mapping. Note that Symbolics NFS must use one of these mechanisms to operate properly.

1. Reading the `/etc/passwd` and `/etc/group` files. Symbolics NFS attempts to determine the uid to username and gid to groupname mapping for UNIX hosts by reading the files `/etc/passwd` and `/etc/group` on those machines. The password associated with the uid is verified.

2. Using the Sun NIS. Sun provides a network distributed database called NIS. Symbolics NFS makes use of the Sun NIS if it knows the name of the Sun Yellow Pages domain and the name of the Sun NIS server for that domain.

See the section "Configuring the Genera Namespace to Use Sun Network Information System (NIS)".

3. Consulting the Namespace Database. Symbolics NFS consults the Namespace Database to see if the mapping information is available there. See the section "Adding a Username-to-UID and Username-to-GID Mapping to the Genera Namespace Database".

### Configuring the Genera Namespace to Use Sun Network Information System (NIS)

Sun provides a network distributed database called the Sun Yellow Pages for mapping usernames to uids and group names to gids. Symbolics NFS uses the Sun NIS if it knows the name of the Sun NIS domain and the Sun NIS server for that domain.

You can determine the the Sun NIS domain name by looking in the `/etc/rc.local` file on a Sun workstation. The Sun NIS domain name is the argument to the command `/bin/domainname` in that file. To determine the domain server, type the command `ypwhich` to the Sun workstation. The reply is the Sun NIS domain server.

You can declare the Sun NIS domain name for an entire site, or for individual hosts.

To let Symbolics NFS know the default name of the Sun NIS domain and the Sun NIS server for your site, edit the namespace object for your local site and add the YP-DOMAIN-NAME user property for site objects and the YP-DOMAIN-SERVER user property for site objects. The YP-DOMAIN-SERVER user property for site objects will be used by Symbolics NFS only if it cannot determine the Sun NIS server for a host in the site by asking it.

### Sample Namespace Database Entry for a Site Object Using Yellow Pages

Here is a sample namespace database entry for a site object using Yellow Pages:

```
SITE YOSEMITE
PRETTY-NAME Yosemite
LOCAL-NAMESPACE YOSEMITE
SITE-DIRECTORY "S1:/usr/share/symbolics/sys.sct/site/"
TIMEZONE EST
HOST-FOR-BUG-REPORTS HALF-DOME
USER-PROPERTY YP-DOMAIN-NAME yosemite
USER-PROPERTY YP-DOMAIN-SERVER S1
```

To let the Symbolics NFS know the name of the Sun NIS domain and the Sun NIS server for a specific host, edit the namespace object for the host and add the YP-DOMAIN-NAME user property for host objects and the YP-DOMAIN-SERVER user property for host objects. The YP-DOMAIN-SERVER user property for host objects will be used by Symbolics NFS only if it cannot determine the Sun NIS server for the host by asking it.

### Adding a Username-to-UID and Username-to-GID Mapping to the Genera Namespace Database

Username-to-uid, username-to-gid, username-to-gids, and username-to-homedir mappings can be made in the namespace database on either a host-specific or on a default basis.

To add a host-specific mapping to the namespace database, edit the namespace object for the applicable user and assign these user object user property attributes:

RPC-HOST-UID
RPC-HOST-GID
RPC-HOST-GIDS
RPC-HOST-HOMEDIR

To add default mappings to the namespace database, edit the namespace object for the applicable user, and assign these user object user property attributes:

RPC-UID
RPC-GID
RPC-GIDS
RPC-HOMEDIR

The mappings declared by this user namespace object are used only after Symbolics NFS has failed to find a mapping by reading the `/etc/passwd` file for that host or by using the Sun NIS.

**Note:** If you are using the Sun NIS for lookup, it is a good idea to disable the use of the namespace for determining the mapping. For further information on this topic, see the section "RPC-USE-NAMESPACE User Property".

### UNIX Name Lookup on Symbolics Computers

UNIX name lookup facilities for the Symbolics computers provide the UNIX equivalent of many of the Genera namespace services. These facilities allow for per-host mappings of usernames to UNIX uids, groupnames to gids, the inverse mappings, and assorted other mappings. UNIX name lookup facilities are used for RPC authentication and for NFS.

In looking up UNIX names, Genera searches these databases in this order:

- The UNIX file database, which consists of files such as `/etc/passwd`, `/etc/group`, `/etc/hosts` and so on.
- Sun's NIS
- The Genera namespace

You can use the Genera namespace user properties to assign mappings for sites that do not run NIS and do not have readable files.

The goal of this ordering is to return the same data that a lookup on the UNIX would return. In practice, this may be impossible. For example, to look up a user name, UNIX machines first look in the file `/etc/passwd`. This may not be possible to emulate from the network, as the file may not be accessible to network file access — the root directory of the machine may not be exported for NFS mounting. In this case, the Sun NIS are consulted without first checking the `/etc/passwd` file. In cases where there is an entry in the `/etc/passwd` file that shadows the Sun NIS, lookup on a UNIX machine returns a different value from the value returned on the Symbolics machine.

The lookup mechanism is recursive. For example, consider the problem of determining the uid of the currently logged-in user. The first database to be searched is the file database. Before the file database can be searched, NFS needs to know a uid to use for authentication to read a file, so it will also look up the uid of the currently logged-in user. This lookup skips the file lookup process and continues on to the Sun Yellow Pages database. After the Sun NIS returns a uid, NFS uses it to read the files, and the FILE database lookup is able to return a uid.

### File System Access Through Symbolics NFS

Symbolics NFS provides file transfer, direct file access, directory lookup, and file attribute modification. In addition, Symbolics NFS allows you to access files using the standard Genera activity interfaces (for example, you can use `c-X c-F` in Zmacs, or the Show Directory command from the Command Processor). Because the NFS protocol is UNIX oriented, there are some behavioral differences from the other remote file protocols available on Genera, primarily in status-line states and pathname syntax.

You can modify UNIX required characteristics with customization parameters — variables or user properties that can be assigned to namespace objects. See the section "User Properties for Configuring a UNIX File System".

Symbolics NFS requires that you first load Symbolics IP/TCP and Symbolics RPC. Note that you only have to load Symbolics RPC separately on 3600-family machines.

### UNIX File System Access Through Symbolics NFS

Files on a UNIX file system can be accessed from Genera using pathnames that follow UNIX syntax conventions. All Genera file and directory commands will operate on the UNIX file system.

Typically, you access UNIX files using the format:

*host*:/*directory*/*filename*...

The word *host* represents the name of a UNIX host. *Directory* specifies the directory (or subdirectory) in which the file *file-name* resides. The slash character (/) separates pathname components. This format can be used for local or remote hosts.

### UNIX Pathname Completion

Genera's completion facilities can access the file systems of various hosts, including Sun computers. To perform pathname completion, Genera looks in the host's file system and returns a new, possibly more specific string than the one that you have entered, expanding any unambiguous abbreviations in a host-dependent fashion.

Wildcard syntax is supported for UNIX pathnames. An asterisk (*) is used to specify a canonical type and is equivalent to the keyword argument **:wild**. One common use of the double asterisk (**) is to specify all directories and subdirectories; it is equivalent to the keyword argument **:wild-inferiors**. The double asterisk wildcard is supported only for UNIX files accessed by means of NFS.

Use the c-/ and c-? keystrokes to see multiple possibilities for what you have already typed. For more information about using these keystrokes, see the section "c-? and c-/".

### UNIX Pathname Merging

Merging of UNIX pathnames is identical in most respects to pathname merging of LMFS pathnames. Like files in LMFS, UNIX pathnames are merged against the default. You need to specify only those fields (name, type, or version) that differ from the context-dependent default to create a new pathname. However, unlike LMFS pathnames, UNIX pathnames are case-sensitive. A complete discussion of Genera-style pathname merging appears in the section "Pathname Defaults and Merging".

The next examples show pathname merging when a new host or directory is specified:

| *Default* | *Input* | *Merged* |
|---|---|---|
| ENIAC:/u0/mwra/foo.lisp | MARK-I: | MARK-I:/u0/mwra/foo.lisp |

| *Default* | *Input* | *Merged* |
|---|---|---|
| ENIAC:/u0/mwra/foo.lisp | /etc/ | ENIAC:/etc/foo.lisp |
| ENIAC:/u0/mwra/foo.lisp | patches/ | ENIAC:/u0/mwra/patches/foo.lisp |
| ENIAC:/u0/mwra/foo.lisp | ../mbta/ | ENIAC:/u0/mbta/foo.lisp |

Since UNIX does not recognize that pathnames have a name or type, it is possible for UNIX filenames to have any number of dots in them. Because of this, names and types are not automatically defaulted during the merging process:

| *Default* | *Input* | *Merged* |
|---|---|---|
| ENIAC:/u0/mwra/foo.lisp | bar | ENIAC:/u0/mwra/bar |
| ENIAC:/u0/mwra/foo.lisp | .bin | ENIAC:/u0/mwra/.bin |
| ENIAC:/u0/mwra/foo.lisp | ...dots... | ENIAC:/u0/mwra/...dots... |

However, names can be merged if you explicitly specify this in the input to be merged. You can indicate whether you want the name or type specified by using the double-arrow character "↔", entered with the SYMBOL-1 key combination:

| *Default* | *Input* | *Merged* |
|---|---|---|
| ENIAC:/u0/mwra/foo.lisp | bar.↔ | ENIAC:/u0/mwra/bar.lisp |
| ENIAC:/u0/mwra/foo.lisp | ↔.bin | ENIAC:/u0/mwra/foo.bin |

The double-arrow character (↔) indicates only whether an entire name or type component should be defaulted. If there are more characters in a name or type pathname component, it means nothing special:

| *Default* | *Input* | *Merged* |
|---|---|---|
| ENIAC:/u0/mwra/foo.lisp | bar.b↔ | ENIAC:/u0/mwra/bar.b↔ |
| ENIAC:/u0/mwra/foo.lisp | x↔.bin | ENIAC:/u0/mwra/x↔.bin |

If a filename has more than one dot in it, Genera interprets the last dot as the separator between the name and type component, and the rest of the dots as part of the name component. Therefore, using ↔ at the end of a filename will result in the type being defaulted, but using ↔ anywhere else in the filename has no effect:

| *Default* | *Input* | *Merged* |
|---|---|---|
| ENIAC:/u0/mwra/foo.lisp | ...dots...↔ | ENIAC:/u0/mwra/...dots...lisp |
| ENIAC:/u0/mwra/foo.lisp | ↔...dots... | ENIAC:/u0/mwra/↔...dots... |

### UNIX Backup Files

Through Symbolics NFS, Genera maintains backup file versions for UNIX using a numbering scheme that is compatible with those UNIX programs that do not recognize version numbers.

If you create a new file, foo.c, its creation is recorded as:

```
foo.c       1:00pm
```

When a new version of the file is written, the previous version is assigned a backup version number:

```
foo.c       1:14pm
foo.c.~1~   1:00pm
```

This process continues as you write other new versions of a file:

```
foo.c       2:03pm
foo.c.~2~   1:14pm
foo.c.~1~   1:00pm
```

You can set the number of backup versions that are retained by Genera with the "NFS-GENERATION-RETENTION-COUNT User Property" for host namespace objects.

Files with backup version numbers are not normally represented internally to Genera with pathname version numbers. Consequently, no Genera utility treats backup file copies as files with versions. For example, Dired and the Clean File command do not behave as if file backup copies are older versions of the same file.

### Using SCT with a UNIX File System

The System Construction Tool (SCT) presents a special case in using UNIX backup files. SCT requires a version number for the files it uses. For directories maintained by SCT, files are maintained so that the newest versions of the files have an explicit version number. (See the section "System Construction Tool" for background information on SCT).

Use the name of a UNIX directory to specify that files contained in it are to be maintained by SCT and should all end with explicit version numbers. If the name of a UNIX directory ends with the `.sct` file extension, Symbolics NFS creates all new files under that directory with explicit version numbers.

Here's an example. Suppose a `sys.translations` file looked like this:

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Package: USER -*-

(fs:set-logical-pathname-host "SYS" :translations
  '(("**;" "ACME-UNIX:/usr/share/symbolics/rel-8-0/sys.sct/**/")))
```

In this case, all new files created with Symbolics NFS under `/usr/share/symbolics/rel-8-0/sys.sct/` are created with explicit version numbers in the newest versions of files, meeting SCT requirements. This is true for all subdirectories of a directory whose name ends in `.sct`.

The double-asterisk (**) wildcard can be used only in UNIX pathnames that are accessed by means of NFS. Other UNIX file protocols, such as FTP, do not recognize this wildcard. For further information on the use of the double-asterisk wildcard, see the section "LMFS Pathnames".

Genera considers UNIX pathnames with any directory components ending in the string ".sct" to be for SCT-maintained files, and hence to contain version numbers. The syntax that Genera uses for backup file version numbers in UNIX pathnames is a number surrounded by tildes separated by a dot from the file type.

This pathname is considered to have a version number:

```
ENIAC:/u0/mwra/visi-brain.sct/foo.lisp.~23~
```

So merging with it as a default behaves like this:

| *Default* | *Input* | *Merged* |
|---|---|---|
| ENIAC:/u0/dpw.sct/code/foo.lisp.~23~ | bar.ₖ | ENIAC:/u0/dpw.sct/code/bar.lisp |
| ENIAC:/u0/dpw.sct/code/foo.lisp.~23~ | ₖ.bin | ENIAC:/u0/dpw.sct/code/foo.bin |

Note that backup file version numbers are are not defaulted as part of the merging operation.

You can set the number of backup versions that are retained by Genera with the "NFS-GENERATION-RETENTION-COUNT User Property"for host namespace objects.

## User Properties for Configuring a UNIX File System

### NFS-REMOTE-FILESYSTEM User Property

**Syntax:** NFS-REMOTE-FILESYSTEM *filename qualified-filename*

A user property for host namespace objects. The *filename* argument represents a pathname that specifies a directory on the host. The *qualified-filename* argument specifies a device. A device is defined as a directory on another host that is mounted on *filename*, a directory on the local host.

If the mount points for a UNIX host are unknown when Symbolics NFS accesses a file, it assumes that the file is local to that UNIX host. If the file is on some other host, in a file system that was NFS mounted by the UNIX host, Symbolics NFS can determine where the file really is only in cases where Symbolics NFS knows that the file system has been mounted by the UNIX host.

### NFS-TRANSFER-SIZE User Property

**Syntax:** NFS-TRANSFER-SIZE *integer*

A user property for host namespace objects. It specifies the number of bytes of data requested or sent by each network RPC call when transferring whole files to and from this host. When this property is present on the host namespace object, the value overrides the relevant default.

### NFS-AUTOMOUNT User Property

**Syntax:** NFS-AUTOMOUNT *user-property*

A user property for host namespace objects. It informs the Symbolics client that an NFS server is running the automount program so Symbolics NFS can duplicate the file system layout of the NFS server. The value of the *user-property* argument

should be the arguments given to the automount program running on the NFS server. See the UNIX man page for the automount program for more information about the automount program.

If the value of this user property is a single token that does not begin with the character N, Symbolics NFS assumes the host is running the automount program with no arguments. Symbolics NFS then consults the auto.master NIS map for automount configuration information.

If the value of the user property is more than one token, the value is interpreted as the arguments to the automount program running on the server.

### NFS-GENERATION-RETENTION-COUNT User Property

**Syntax:** NFS-GENERATION-RETENTION-COUNT *integer*

A user property for host namespace objects. Sets the number of file versions retained by NFS when superseding an existing file. The default is 5.

### Network RPC

Symbolics Network RPC includes these features:

- Support for UDP and TCP transport agents
- Support for the RPC portmapper
- Support for RPC Authentication
- Support for RPC Futures

Network RPC lets you communicate with other RPC implementations over a network, specifically Sun and other UNIX implementations through TCP and UDP. To do this, Symbolics RPC supports RPC authentication and the Portmapper. See the Sun Microsystems document *Network Programming* for introductory information on these topics.

### UDP and TCP Transport Agents for Symbolics RPC

Symbolics RPC communication with another host is mediated by an RPC transport agent. Transport agents for the UDP and TCP Internet protocols may be created using the **rpc:host-udp-transport-agent** and **rpc:host-tcp-transport-agent** functions.

For further information, see the section "Selecting the Right RPC Agent for Symbolics UX Applications".

### RPC Portmapper

In UNIX more than one process provides RPC services. Each RPC server process cannot, however, communicate to clients over the same TCP or UDP ports. Instead, each RPC server process listens to its own TCP or UDP port for RPC requests, and services only requests that come from connections to these ports. Because it is not practical to permanently assign a TCP or UDP port for each remote module, port assignment is dynamic. When a UNIX RPC server starts, it obtains a port number from the system and then provides RPC service on that port.

The portmapper tells RPC clients which TCP or UDP port to use to communicate to the right RPC server. It is queried by an RPC client whenever the client needs to make a connection to an RPC server. The portmapper is an RPC server just like any other, except that it always provides the service from the same well-known TCP or UDP ports. Hence, clients always know how to connect to the portmapper.

The portmapper knows the port numbers to which RPC servers are listening because the servers tell the portmapper the information as part of their initialization process. After acquiring a UDP or TCP port, each RPC server process on the Sun registers itself with the portmapper process by making an appropriate RPC call to the portmapper.

The Network RPC system provides portmapper support automatically. The TCP and UDP network agents provide both client and server support for the portmapper. The support is completely transparent.

For client RPC the agents do the work of determining which TCP or UDP port to use, and automatically open network connections to the appropriate ports. Idle network connections are closed automatically to reduce system overhead.

A portmapper server is provided with the Network RPC system so Sun RPC clients that expect a portmapper can communicate with Symbolics RPC servers. However, Genera RPC servers are not required to register with the Genera portmapper, since all the servers can share the same port.

### RPC Authentication

Authentication is a way for RPC clients to identify themselves to RPC servers. This is done by passing identifiers known as "credentials" in the XDR block that makes up an RPC call. Authentication is part of the RPC standard, RFC #1057 "RPC: Remote Procedure Call Protocol specification version 2."

RFC 1057 defines three types of authentication: null, UNIX, and DES (data encryption standard). Null authentication carries no credentials; essentially, it is a placeholder for authentication and carries no information. UNIX authentication carries a credential block consisting of the user's user id (uid), primary group id (gid), and other gids. The UNIX encryption routine is used to prompt for and verify the password associated with the uid. These UNIX-specific concepts have no meaning to Genera, but Genera may supply UNIX authentication deriving the uid and gids from the server. DES authentication is less UNIX-specific, and operates by passing encrypted timestamps between the RPC server and client.

Authentication is actually implemented by the transport agents. The embedded transport agent does not implement any form of authentication. The TCP and UDP transport agents implement **:null** and **:unix** authentication. To request RPC authentication, specify the **:authentication** option to **rpc:define-remote-module** and, optionally, **rpc:define-remote-entry**. The transport agent then supplies the strictest requested form of authentication that it implements.

In order to implement **:unix** authentication, the transport agent must be able to come up with appropriate uids and gids. This is implemented as part of the Network RPC system, which determines the logged-in user's uid and gids based on the namespace information for the user plus other information (described in the section "UNIX UID and GID Mapping"). For further information, see the section "UDP and TCP Transport Agents for Symbolics RPC".

### RPC Futures

Futures are a way of performing non-blocking RPC operations. Non-blocking RPC operations differ from asynchronous RPC operations in that they return values (see the **:asynchronous** option to **rpc:define-remote-entry**); they differ from the default form of RPC operations in that they allow the client to continue running while the RPC operation completes. (This is completely analogous to the Generic Network System futures, which underlie **net:invoke-multiple-services**. See, for example, **net:start-service-access-path-future**.) One common use of RPC futures is to start a number of RPC operations, then go back and check the return status of the earlier operations while the later ones are completing.

To use RPC futures, specify **(:future t)** as a suboption to **(:lisp ...)** in **rpc:define-remote-entry**. For example:

```
(define-remote-entry long-operation examples
  (:lisp (:future t) ...) ...)

(let ((future nil))
  (unwind-protect-case ()
      (progn
        (setq future (start-long-operation-future))

           ... some other computations ...

        (finish-long-operation-future future))
    (:abort
      (when future
        (abort-long-operation-future future)))))
```

**Note**: Futures left unfinished or unaborted consume system resources, so code written using futures should be careful that they are either finished or aborted, just as file-manipulating code has to be careful not to leave files open.

## Host and Site Customization Parameters for Network RPC

### RPC-UDP-CALL-RETRANSMIT-TIMEOUTS User Property

**Syntax:** RPC-UDP-CALL-RETRANSMIT-TIMEOUTS *timeout-integer-value*

A user property for host or site namespace objects. Sets the amount of time (in 60ths of a second) that will elapse before retransmitting a call. Otherwise, the value of **rpc:*default-udp-call-retransmit-timeouts*** is in effect.

### RPC-UDP-CALL-TIMEOUT User Property

**Syntax**: RPC-UDP-CALL-TIMEOUT *timeout-integer-value*

A user property for host or site namespace objects. It sets the amount of time (in 60ths of a second) to elapse before a call is timed out. Otherwise, the value of **rpc:*default-udp-call-timeout*** is in effect.

### RPC-UDP-CONNECTION-IDLE-TIMEOUT User Property

**Syntax:** RPC-UDP-CONNECTION-IDLE-TIMEOUT *timeout-integer-value*

A user property for host or site namespace objects. It sets the amount of idle time (in 60ths of a second) before closing a UDP connection. If no value is set, the value of **rpc:*default-udp-rpc-connection-idle-timeout*** is in effect.

## ID Mapping Customization Parameters for Network RPC

### RPC-GID User Property

**Syntax**: RPC-GID *GID-number*

A user property for user namespace objects. It specifies the default UNIX gid for the user. This property is used only if no uid or username↔gid mapping for the user can be determined by reading the `/etc/passwd` file on the host, by using the Sun NIS database, or by the RPC-HOST-GID user property specified for this user namespace object.

### RPC-GIDS User Property

**Syntax**: RPC-GIDS *GID-Number GID-Number...*

A user property for user namespace objects. It specifies the default UNIX gids for this user. This property is used only when no username→gids mapping for the user can be determined by reading the `/etc/passwd` file on the host, by using the

Sun NIS database, or by the RPC-GIDS user property on the user namespace object.

### RPC-GROUP-FILE User Property

**Syntax:** RPC-GROUP-FILE *filename*

A user property for host or site namespace objects. Determines the group file used by UNIX name lookup facilities for this host or all the hosts in the site. The default is `/etc/group`.

### RPC-HOMEDIR User Property

**Syntax**: RPC-HOMEDIR *directory-pathname*

A user property for user namespace objects. It specifies the home directory for this user. This property is used only if no username↔homedir mapping for the user can be determined either through reading the `/etc/passwd` file on the host or using the Sun Yellow Pages database, or by the RPC-HOST-HOMEDIR user property specified on the user namespace object.

### RPC-HOST-GID User Property

**Syntax:** RPC-HOST-GID *host-name GID-number*

A user property for user namespace objects. It specifies the UNIX gid on the host specified by *host-name*. This property is used only if no username→gid mapping can be determined for this user either by reading the `/etc/passwd` file on the host or by using the Sun NIS database.

For the specific host, it supersedes any RPC-GID user property on the user namespace object.

### RPC-HOST-GIDS User Property

**Syntax:** RPC-HOST-GIDS *host-name GIC-Number GID-Number ...*

A user property for user namespace objects. It specifies the UNIX gids on the host specified by *host-name*. This property is used only if no username→gid mapping for the user can be determined either by reading the `/etc/passwd` file on the host or by using the Sun NIS database.

For the specific host, it supersedes any RPC-GIDS user property on the user namespace object.

### RPC-HOST-HOMEDIR User Property

**Syntax:** RPC-HOST-HOMEDIR *host-name directory-pathname*

A user property for user namespace objects. It specifies the home directory for this user on the host specified by *host-name*. This property is used only if no username↔homedir mapping for the user can be determined either by reading the `/etc/passwd` file on the host or by using the Sun NIS database.

For the specific host, it supersedes any RPC-HOMEDIR user property specified on the user namespace object.

### RPC-HOSTS-FILE User Property

**Syntax:** RPC-HOSTS-FILE *filename*

A user property for host or site namespace objects. It determines the hosts file used by UNIX name lookup facilities for this host or all the hosts in the site. The default is `/etc/hosts`.

### RPC-HOST-UID User Property

**Syntax:** RPC-HOST-UID *host-name UID-number*

A user property for user namespace objects. It specifies the user's UNIX uid on the host specified by the host name. This property is used only if no uid or username↔uid mapping for this user can be determined either by reading the `/etc/passwd` file on the host or by using the Sun NIS database. For the specific host, it supersedes any RPC-UID user property on the user namespace object.

### RPC-PASSWD-FILE User Property

**Syntax:** RPC-PASSWD-FILE *filename*

A user property for host or site namespace objects. It determines the passwd file used by UNIX name lookup facilities for this host or all the hosts in the site. The default is `/etc/passwd`.

### RPC-UID User Property

**Syntax:** RPC-UID *UID-number*

A user property for user namespace objects. It specifies the default UNIX uid for this user. This property is used only if no uid or username↔uid mapping for this user can be determined either by reading the /etc/passwd file on the host or using the Sun NIS database, or if the RPC-HOST-UID user property is specified for this user namespace object.

### RPC-USE-FILE User Property

**Syntax:** RPC-USE-FILE *boolean*

A user property for host or site namespace objects. It controls whether the RPC layer attempts to use the UNIX `/etc/passwd` and `/etc/group` files for determining the Username-uid and Groupname-gid mapping. Usernames and groupnames are transferred as UNIX uids and gids in the RPC and NFS protocol rather than as strings. If this user property is present on a host namespace object and the first character of its value is N, file reading is disabled for mappings related to this host. Otherwise, the value of **rpc:*default-use-file-for-unix-name-lookup*** is in effect.

### RPC-USE-HOSTS-FILE User Property

**Syntax:** RPC-USE-HOSTS-FILE *boolean*

A user property for host or site namespace objects. Determines whether Network RPC attempts to use the hosts file to determine the Username-uid and Groupname-gid mapping. Usernames and groupnames are not transferred as strings in the RPC and NFS protocol, but as UNIX uids and gids. If this user property is present on a host namespace object and the first character of its value is N, the use of the hosts file is disabled for mappings related to this host or all the hosts in the site. Otherwise, the value of **rpc:*default-use-hosts-file-for-unix-name-lookup*** is in effect.

### RPC-USE-NAMESPACE User Property

**Syntax:** RPC-USE-NAMESPACE *boolean*

A user property for host or site namespace objects. It determines whether Network RPC attempts to use the Namespace database to determine the Username-uid and Groupname-gid mapping. Usernames and groupnames are not transferred as strings in the RPC and NFS protocol, but as UNIX uids and gids. If this user property is present on a host namespace object and the first character of its value is N, the use of the Namespace database is disabled for mappings related to this host or all the hosts in the site. Otherwise, the value of **rpc:*default-use-namespace-for-unix-name-lookup*** is in effect.

### RPC-USE-YELLOW-PAGES User Property

**Syntax:** RPC-USE-YELLOW-PAGES *boolean*

A user property for host or site namespace objects. It determines whether Network RPC attempts to use Sun NIS to determine the Username-uid and Groupname-gid mapping. Usernames and groupnames are not transferred as strings in the RPC and NFS protocol, but as UNIX uids and gids.

If this user property is present on a host or site namespace object and the first character of its value is N, Sun NIS are disabled for mappings related to this host or all the hosts in the site. Otherwise, the value of **rpc:*default-use-yellow-pages-for-unix-name-lookup*** is in effect.

### YP-DOMAIN-NAME User Property

**Syntax:** YP-DOMAIN-NAME *domain-name*

A user property for host and site namespace objects. It specifies the Sun NIS domain name for the host or site.

### YP-DOMAIN-SERVER User Property

**Syntax:** YP-DOMAIN-SERVER *server-name*

A user property for host and site namespace objects. It specifies a Sun NIS domain server for this host or site. This parameter is used only if the name of the domain server cannot be determined for this host or site by asking it.

## Dictionary of Network RPC Functions and Variables

### Interface to Network RPC Transport Agents

**rpc:host-tcp-transport-agent** *host* *Function*

Returns a TCP transport agent connected to the specified host. The transport agent can then be passed as the value of the **:transport-agent** keyword to a Lisp RPC stub.

**rpc:host-udp-transport-agent** *host* *Function*

Returns a UDP transport agent connected to the specified host. The transport agent can then be passed as the value of the **:transport-agent** keyword to a Lisp RPC stub. See the function **rpc:host-tcp-transport-agent**.

**rpc:*udp-call-timeout*** *Variable*

Sets the value for the amount of time to elapse before a call is timed out. Setting this variable overrides any setting of the RPC-UDP-CALL-TIMEOUTS host or site user property.

**rpc:*default-udp-call-timeout*** *Variable*

Sets the value for the amount of time to elapse before a call is timed out. The value set by this variable is overridden by a value set by the RPC-UDP-CALL-TIMEOUTS host or site user property. The default value is 1740 (29 seconds).

**rpc:*udp-call-retransmit-timeouts*** *Variable*

Sets the values for the amount of time before retransmitting a call. Setting this variable overrides any setting of the RPC-UDP-CALL-RETRANSMIT-TIMEOUTS host or site user property.

**rpc:*default-udp-call-retransmit-timeouts*** *Variable*

Sets the values for the amount of time before retransmitting a call. The value set by this variable is overridden by any value set by the RPC-UDP-CALL-RETRANSMIT-TIMEOUTS host or site user property. Default values are 60 120 240 480 840.

**rpc:*udp-rpc-connection-idle-timeout*** *Variable*

Sets the amount of idle time before closing a UDP connection. Setting this variable overrides any setting of the RPC-UDP-CONNECTION-IDLE-TIMEOUT host or site user property.

**rpc:*default-udp-rpc-connection-idle-timeout*** *Variable*

Sets the amount of idle time before closing a UDP connection. The value set by this variable is overridden by any value set by the RPC-UDP-CONNECTION-IDLE-TIMEOUT host or site user property. The default value is 10800 (3 minutes).

**rpc:remote-host** *transport-agent* *Function*

Returns the host object on the other end of this transport agent. The value **nil** is returned for embedded transport agents.

**rpc:remote-system-type** *transport-agent* *Function*

Returns the system type of the host at the other end of this transport agent. The system type can be **:macintosh** for MacIvory embedded transport agents, **:UNIX42** for Symbolics UX-family embedded transport agents, or **:LISPM** for Network RPC transport agents.

### Interface to UNIX name Lookup Support

**rpc:host-unix-name-lookup-access-path** *host* *Function*

Returns the UNIX-name-lookup access path for the host.

**rpc:username->passwd** *UNIX-name-lookup-access-path username* *Function*
Returns the decoded passwd entry for the given username.

**rpc:username->password** *UNIX-name-lookup-access-path username* *Function*
Returns the encrypted password for the given username.

**rpc:username->uid** *UNIX-name-lookup-access-path username* *Function*
Returns the UNIX uid for the given username.

**rpc:username->gid** *UNIX-name-lookup-access-path username* *Function*
Returns the UNIX gid (as if from the `/etc/passwd` file) for the given username.

**rpc:username->gecos** *UNIX-name-lookup-access-path username* *Function*
Returns the gecos entry for the given username.

**rpc:username->homedir** *UNIX-name-lookup-access-path username* *Function*
Returns the home directory (as a string) for the given username.

**rpc:username->shell** *UNIX-name-lookup-access-path username* *Function*
Returns the shell for the given username.

**rpc:username->gids** *UNIX-name-lookup-access-path username* *Function*
Returns the UNIX gids (as if from the `/etc/group` file) for the given username.

**rpc:uid->passwd** *UNIX-name-lookup-access-path uid* *Function*
Returns the decoded passwd entry for the given uid.

**rpc:uid->username** *UNIX-name-lookup-access-path uid* *Function*
Returns the username for the given UNIX uid.

**rpc:groupname->group** *UNIX-name-lookup-access-path groupname* *Function*
Returns the decoded group entry for the given username.

**rpc:groupname->gid** *UNIX-name-lookup-access-path groupname* *Function*
Returns the UNIX gid for the given groupname.

**rpc:groupname->password** *UNIX-name-lookup-access-path groupname* *Function*
Returns the encrypted password for the given groupname.

**rpc:groupname->usernames** *UNIX-name-lookup-access-path groupname* *Function*
Returns the usernames for the given groupname.

**rpc:gid->group** *UNIX-name-lookup-access-path gid* *Function*
Returns the decoded group entry for the given gid.

**rpc:gid->groupname** *UNIX-name-lookup-access-path gid* *Function*
Returns the groupname for the given UNIX gid.

**rpc:hostname->host** *UNIX-name-lookup-access-path hostname* *Function*
Returns the parsed host entry for the given hostname.

**rpc:hostname->address** *UNIX-name-lookup-access-path hostname* *Function*
Returns the primary IP address for the given hostname.

**rpc:address->host** *UNIX-name-lookup-access-path address* *Function*
Returns the parsed host entry for the given IP address.

**rpc:address->hostname** *UNIX-name-lookup-access-path address* *Function*
Returns the hostname for the given IP address.

**rpc:reset-all-unix-name-lookup-access-paths** &optional *forget-p* *Function*
Resets all UNIX name lookup access paths.

**rpc:yellow-pages-order** *yellow-pages-UNIX-name-lookup-access-path map* *Function*
Returns the order number for a Sun NIS map. For example, the form:

```
(rpc:yellow-pages-order
        (rpc:host-unix-name-lookup-access-path net:*emb-host*)
        "passwd.byname")
```

Returns

```
620082961
```

**rpc:yellow-pages-lookup** *yellow-pages-UNIX-name-lookup-access-path map key* *Function*

Looks up a key in a Sun NIS map. For example, the form

```
(rpc:yellow-pages-lookup
        (rpc:host-unix-name-lookup-access-path net:*emb-host*)
        "passwd.byname" "daemon")
```

returns

```
"daemon:*:1:1::/:"
```

**rpc:yellow-pages-list** *yellow-pages-UNIX-name-lookup-access-path map* *Function*

Returns the entire contents of a Sun NIS map as a list. For example, the form:

```
(rpc:yellow-pages-list
        (rpc:host-unix-name-lookup-access-path net:*emb-host*)
        "group.byname")
```

returns

```
(("operator" "operator:*:5:kaufman,backup")
 ("nogroup" "nogroup:*:-2:")
 ("daemon" "daemon:*:1:")
 ("wheel" "wheel:*:0:")
 ("staff" "staff:*:10:")
 ("other" "other:*:20:")
 ("audit" "audit:*:9:")
 ("news" "news:*:6:")
 ("kmem" "kmem:*:2:")
 ("tty" "tty:*:4:")
 ("bin" "bin:*:3:")
 ("+" "+:"))
```

### Network RPC Variables for ID Mapping

**rpc:*use-namespace-for-unix-name-lookup*** *Variable*

Determines whether the namespace can be used for UNIX name lookup. The default value is **t**. When set to **nil**, this variable overrides any value set by the corresponding user property, RPC-USE-NAMESPACE.

**rpc:*default-use-namespace-for-unix-name-lookup*** *Variable*

Makes the Namespace the default choice for UNIX name lookup. The default value is **t**. The value set by this variable may be overridden by a value set by the corresponding namespace user property, RPC-USE-NAMESPACE.

**rpc:*use-yellow-pages-for-unix-name-lookup*** *Variable*

Determines whether NIS are used for UNIX name lookup. The default value is **t**. When set to **nil**, this variable overrides any value set by the corresponding user property, RPC-USE-YP.

**rpc:*default-use-yellow-pages-for-unix-name-lookup*** *Variable*

Makes the NIS the default choice for UNIX name lookup. The default value is **t**. The value set by this variable may be overridden by a value set by the corresponding namespace user property, RPC-USE-YP.

**rpc:*use-file-for-unix-name-lookup*** *Variable*

Determines whether passwd and group files are used for UNIX name lookup. The value set by this variable is overridden by a value set by the corresponding namespace user property. The default value is **t**. When set to **nil**, this variable overrides any value set by the corresponding user property, RPC-USE-FILE.

**rpc:*default-use-file-for-unix-name-lookup*** *Variable*

Makes the passwd and group files the default choice for UNIX name lookup. The default value is **t**. The value set by this variable may be overridden by a value set by the corresponding namespace user property, RPC-USE-FILE.

**rpc:*use-hosts-file-for-unix-name-lookup*** *Variable*

Determines whether the hosts file is used for UNIX name lookup. The default value is **t**. When set to **nil**, this variable overrides any value set by the corresponding user property, RPC-USE-FILE.

**rpc:*default-use-hosts-file-for-unix-name-lookup*** *Variable*

Makes the hosts file the default choice for UNIX name lookup. The default value is **nil**. The value set by this variable may be overridden by a value set by the corresponding namespace user property, RPC-USE-FILE.

**Network RPC Error Conditions**

**rpc:udp-rpc-connection-error** *Flavor*

Signaled when a network error occurrs for the UDP connection.

Built on **sys:connection-error** and **rpc:rpc-error**.

**rpc:udp-rpc-host-not-responding** *Flavor*

Signaled if no reply is received for a UDP RPC request within ***udp-rpc-call-timeout***.

Built on **sys:host-not-responding-during-connection** and **rpc:rpc-error**.

**rpc:tcp-rpc-stream-closed** *Flavor*

Signaled when the TCP connection to the RPC server closes for any reason during the execution of an RPC call.

Built on the errors **sys:network-stream-closed** and **rpc:rpc-error**.