NAME

ypclnt, yp_get_default_domain, yp_bind, yp_unbind, yp_match, yp_first, yp_next, yp_all, yp_order, yp_master, yperr_string, ypprot_err – NIS Version 2 client interface

SYNOPSIS

cc [*flag* ...] *file* ... –**lnsl** [*library* ...]

#include <rpcsvc/ypclnt.h>
#include <rpcsvc/yp_prot.h>

MT-LEVEL

Unsafe

DESCRIPTION

This package of functions provides an interface to NIS, Network Information Service Version 2, formerly referred to as YP. In this version of SunOS, NIS version 2 is supported only for compatibility with previous versions. The recommended enterprise level information service is NIS+ or NIS version 3, see **nis+**(1). Moreover, this version of SunOS supports only the client interface to NIS version 2. It is expected that this client interface will be served either by an existing **ypserv** process running on another machine on the network that has an earlier version of SunOS or by an NIS+ server, see **rpc.nisd**(1M), running in "YP-compatibility mode". Refer to the **NOTES** section in **ypfiles**(4) for implications of being an NIS client of an NIS+ server in "YP-compatibility mode", and to **ypbind**(1M), **ypwhich**(1), **ypmatch**(1), and **ypcat**(1) for commands to access NIS from a client machine. The package can be loaded from the standard library, **/usr/lib/libnsl.so.1**.

All input parameter names begin with *in*. Output parameters begin with *out*. Output parameters of type **char** ** should be addresses of uninitialized character pointers. Memory is allocated by the NIS client package using **malloc**(3C), and may be freed by the user code if it has no continuing need for it. For each *outkey* and *outval*, two extra bytes of memory are allocated at the end that contain NEWLINE and null, respectively, but these two bytes are not reflected in *outkeylen* or *outvallen*. *indomain* and *inmap* strings must be non-null and null-terminated. String parameters which are accompanied by a count parameter may not be null, but may point to null strings, with the count parameter indicating this. Counted strings need not be null-terminated.

All functions in this package of type *int* return **0** if they succeed, and a failure code (**YPERR_***xxxx*) otherwise. Failure codes are described under **ERRORS** below.

Routines

yp_bind (char **indomain***);**

To use the NIS name services, the client process must be “bound” to an NIS server that serves the appropriate domain using **yp_bind( )**. Binding need not be done explicitly by user code; this is done automatically whenever an NIS lookup function is called. **yp_bind( )** can be called directly for processes that make use of a backup strategy (for example, a local file) in cases when NIS services are not available. If a process calls **yp_bind( ),** it should call **yp_unbind( )** when it is done using NIS in order to free up resources.

void yp_unbind(char **indomain***);**

Each binding allocates (uses up) one client process socket descriptor; each bound domain costs one socket descriptor. However, multiple requests to the same domain use that same descriptor. **yp_unbind( )** is available at the client interface for processes that explicitly manage their socket descriptors while accessing multiple domains. The call to **yp_unbind( )** makes the domain *unbound*, and frees all per-process and per-node resources used to bind it.

If an RPC failure results upon use of a binding, that domain will be unbound automatically. At that point, the **ypclnt( )** layer will retry a few more times or until the operation succeeds, provided that **rpcbind**(1M) and **ypbind**(1M) are running, and either

- the client process cannot bind a server for the proper domain, or
- RPC requests to the server fail.

If an error is not RPC-related, or if **rpcbind** is not running, or if **ypbind** is not running, or if a bound **ypserv** process returns any answer (success or failure), the **ypclnt** layer will return control to the user code, either with an error code, or a success code and any results.

yp_get_default_domain (char ***outdomain***);**

The NIS lookup calls require a map name and a domain name, at minimum. It is assumed that the client process knows the name of the map of interest. Client processes should fetch the node's default domain by calling **yp_get_default_domain( )**, and use the returned *outdomain* as the *indomain* parameter to successive NIS name service calls. The domain thus returned is the same as that returned using the **SI_SRPC_DOMAIN** command to the **sysinfo**(2) system call.

yp_match(char **indomain*, **char** **inmap*, **char** **inkey*, **int** *inkeylen*, **char** ***outval*, **int** **outvallen***);**

yp_match( ) returns the value associated with a passed key. This key must be exact; no pattern matching is available. **yp_match( )** requires a full YP map name; for example, **hosts.byname** instead of the nickname **hosts**.

yp_first(char **indomain*, **char** **inmap*, **char** ***outkey*, **int** **outkeylen*, **char** ***outval*, **int** **outvallen***);**

yp_first( ) returns the first key-value pair from the named map in the named domain.

yp_next(char **indomain*, **char** **inmap*, **char** **inkey*, **int** *inkeylen*, **char** ***outkey*, **int** **outkeylen*, **char** ***outval*, **int** **outvallen***);**

yp_next( ) returns the next key-value pair in a named map. The *inkey* parameter must be the *outkey* returned from an initial call to **yp_first( )** (to get the second key-value pair) or the one returned from the *n*th call to **yp_next( )** (to get the *n*th + second key-value pair). Similarly, the *inkeylen* parameter must be the *outkeylen* returned from the earlier **yp_first( )** or **yp_next( )** call.

The concept of first (and, for that matter, of next) is particular to the structure of the NIS map being processing; there is no relation in retrieval order to either the lexical order within any original (non-NIS name service) data base, or to any obvious numerical sorting order on the keys, values, or key-value pairs. The only ordering guarantee made is that if the **yp_first( )** function is called on a particular map, and then the **yp_next( )** function is repeatedly called on the same map at the same server until the call fails with a reason of **YPERR_NOMORE**, every entry in the data base will be seen exactly once. Further, if the same sequence of operations is performed on the same map at the same server, the entries will be seen in the same order.

Under conditions of heavy server load or server failure, it is possible for the domain to become unbound, then bound once again (perhaps to a different server) while a client is running. This can cause a break in one of the enumeration rules; specific entries may be seen twice by the client, or not at all. This approach protects the client from error messages that would otherwise be returned in the midst of the enumeration. The next paragraph describes a better solution to enumerating all entries in a map.

yp_all(char **indomain*, **char** **inmap*, **struct ypall_callback** **incallback*);

yp_all( ) provides a way to transfer an entire map from server to client in a single request using TCP (rather than UDP as with other functions in this package). The entire transaction take place as a single RPC request and response. **yp_all( )** can be used just like any other NIS name service procedure, identify the map in the normal manner, and supply the name of a function which will be called to process each key-value pair within the map. The call to **yp_all( )** returns only when the transaction is completed (successfully or unsuccessfully), or the **foreach( )** function decides that it does not want to see any more key-value pairs.

The third parameter to **yp_all( )** is

```
struct ypall_callback *incallback {
        int (*foreach)();
        char *data;
};
```

The function **foreach( )** is called

```
foreach(int instatus, char *inkey, int inkeylen, char *inval, int invallen,
      char *indata);
```

The *instatus* parameter will hold one of the return status values defined in **<rpcsvc/yp_prot.h** — either **YP_TRUE** or an error code. (See **ypprot_err( )**, below, for a function which converts an NIS name service protocol error code to a **ypclnt** layer error code.)

The key and value parameters are somewhat different than defined in the synopsis section above. First, the memory pointed to by the *inkey* and *inval* parameters is private to the **yp_all( )** function, and is overwritten with the arrival of each new key-value pair. It is the responsibility of the **foreach( )** function to do something useful with the contents of that memory, but it does not own the memory itself. Key and value objects presented to the **foreach( )** function look exactly as they do in the server's map — if they were not NEWLINE-terminated or null-terminated in the map, they will not be here either.

The *indata* parameter is the contents of the *incallback→data* element passed to **yp_all( )**. The **data** element of the callback structure may be used to share state information between the **foreach( )** function and the mainline code. Its use is optional, and no part of the NIS client package inspects its contents — cast it to something useful, or ignore it.

The **foreach( )** function is a Boolean. It should return **0** to indicate that it wants to be called again for further received key-value pairs, or non-zero to stop the flow of key-value pairs. If **foreach( )** returns a non-zero value, it is not called again; the functional value of **yp_all( )** is then **0**.

yp_order(char **indomain*, **char** **inmap*, **unsigned long** **outorder***);**

yp_order( ) returns the order number for a map. This function is not supported if the **ypbind** process on the client's system is bound to an NIS+ server running in "YP-compatibility mode".

yp_master(char **indomain*, **char** **inmap*, **char** ***outname***);**

yp_master( ) returns the machine name of the master NIS server for a map.

char ***yperr_string(int** *incode***);**

yperr_string( ) returns a pointer to an error message string that is null-terminated but contains no period or NEWLINE.

ypprot_err (unsigned int *incode***);**

ypprot_err( ) takes an NIS name service protocol error code as input, and returns a ypclnt layer error code, which may be used in turn as an input to **yperr_string( )**.

RETURN VALUES

All integer functions return **0** if the requested operation is successful, or one of the following errors if the operation fails.

YPERR_ACCESS	**15**	**/* access violation */**
YPERR_BADARGS	**1**	**/* args to function are bad */**
YPERR_BADDB	**13**	**/* yp database is bad */**
YPERR_BUSY	**16**	**/* database busy */**
YPERR_DOMAIN	**3**	**/* can't bind to server on this domain */**
YPERR_KEY	**5**	**/* no such key in map */**
YPERR_MAP	**4**	**/* no such map in server's domain */**
YPERR_NODOM	**12**	**/* local domain name not set */**
YPERR_NOMORE	**8**	**/* no more records in map database */**
YPERR_PMAP	**9**	**/* can't communicate with rpcbinder */**
YPERR_RESRC	**7**	**/* resource allocation failure */**
YPERR_RPC	**2**	**/* RPC failure – domain has been unbound */**
YPERR_YPBIND	**10**	**/* can't communicate with ypbind */**
YPERR_YPERR	**6**	**/* internal yp server or client error */**
YPERR_YPSERV	**11**	**/* can't communicate with ypserv */**
YPERR_VERS	**14**	**/* yp version mismatch */**

FILES

/usr/lib/libnsl.so.1

SEE ALSO

nis+(1), **ypcat**(1), **ypmatch**(1), **ypwhich**(1), **rpcbind**(1M), **rpc.nisd**(1M), **ypbind**(1M), **sysinfo**(2), **malloc**(3C), **ypfiles**(4)

NOTES

This interface is unsafe in multithreaded applications. Unsafe interfaces should be called only from the main thread.