

Network File System Version 4 Working Group  
Internet-Draft  
Intended status: Draft Standard  
Obsoletes: [1831](#)  
Expires: September 5, 2009

R. Thurlow  
Sun Microsystems

March 5, 2009

**RPC: Remote Procedure Call Protocol Specification Version 2**  
**draft-ietf-nfsv4-rfc1831bis-13.txt**

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

This document will expire in July, 2009.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

## Abstract

This document describes the ONC (Open Network Computing) Remote Procedure Call (ONC RPC Version 2) protocol as it is currently deployed and accepted. This document obsoletes [[RFC1831](#)].

## Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">1</a>
<a href="#">2.</a>	Changes since <a href="#">RFC 1831</a> . . . . .	<a href="#">1</a>
<a href="#">3.</a>	Terminology . . . . .	<a href="#">1</a>
<a href="#">4.</a>	The RPC Model . . . . .	<a href="#">1</a>
<a href="#">5.</a>	Transports and Semantics . . . . .	<a href="#">1</a>
<a href="#">6.</a>	Binding and Rendezvous Independence . . . . .	<a href="#">1</a>
<a href="#">7.</a>	Authentication . . . . .	<a href="#">1</a>
<a href="#">8.</a>	RPC Protocol Requirements . . . . .	<a href="#">1</a>
<a href="#">8.1.</a>	RPC Programs and Procedures . . . . .	<a href="#">1</a>
<a href="#">8.2.</a>	Authentication, Integrity and Privacy . . . . .	<a href="#">1</a>
<a href="#">8.3.</a>	Program Number Assignment . . . . .	<a href="#">1</a>
<a href="#">8.4.</a>	Other Uses of the RPC Protocol . . . . .	<a href="#">1</a>
<a href="#">8.4.1.</a>	Batching . . . . .	<a href="#">1</a>
<a href="#">8.4.2.</a>	Broadcast Remote Procedure Calls . . . . .	<a href="#">1</a>
<a href="#">9.</a>	The RPC Message Protocol . . . . .	<a href="#">1</a>
<a href="#">10.</a>	Authentication Protocols . . . . .	<a href="#">1</a>
<a href="#">10.1.</a>	Null Authentication . . . . .	<a href="#">1</a>
<a href="#">11.</a>	Record Marking Standard . . . . .	<a href="#">1</a>
<a href="#">12.</a>	The RPC Language . . . . .	<a href="#">1</a>
<a href="#">12.1.</a>	An Example Service Described in the RPC Language . . . . .	<a href="#">1</a>
<a href="#">12.2.</a>	The RPC Language Specification . . . . .	<a href="#">1</a>
<a href="#">12.3.</a>	Syntax Notes . . . . .	<a href="#">1</a>
<a href="#">13.</a>	IANA Considerations . . . . .	<a href="#">1</a>
<a href="#">13.1.</a>	Numbering Requests to IANA . . . . .	<a href="#">1</a>
<a href="#">13.2.</a>	Protecting Past Assignments . . . . .	<a href="#">1</a>
<a href="#">13.3.</a>	RPC Number Assignment . . . . .	<a href="#">1</a>
<a href="#">13.3.1.</a>	To be assigned by IANA . . . . .	<a href="#">1</a>
<a href="#">13.3.2.</a>	Defined by local administrator . . . . .	<a href="#">1</a>
<a href="#">13.3.3.</a>	Transient block . . . . .	<a href="#">1</a>
<a href="#">13.3.4.</a>	Reserved block . . . . .	<a href="#">1</a>
<a href="#">13.3.5.</a>	RPC Number Sub-Blocks . . . . .	<a href="#">1</a>
<a href="#">13.4.</a>	RPC Authentication Flavor Number Assignment . . . . .	<a href="#">1</a>
<a href="#">13.4.1.</a>	Assignment Policy . . . . .	<a href="#">1</a>
<a href="#">13.4.2.</a>	Auth Flavors vs. Pseudo-flavors . . . . .	<a href="#">1</a>
<a href="#">13.5.</a>	Authentication Status Number Assignment . . . . .	<a href="#">1</a>
<a href="#">13.5.1.</a>	Assignment Policy . . . . .	<a href="#">1</a>
<a href="#">14.</a>	Security Considerations . . . . .	<a href="#">1</a>
<a href="#">15.</a>	<a href="#">Appendix A</a> : System Authentication . . . . .	<a href="#">1</a>
<a href="#">16.</a>	<a href="#">Appendix B</a> : Requesting RPC-related numbers from IANA . . . . .	<a href="#">1</a>
<a href="#">17.</a>	<a href="#">Appendix C</a> : Current number assignments . . . . .	<a href="#">1</a>
<a href="#">18.</a>	Normative References . . . . .	<a href="#">1</a>
<a href="#">19.</a>	Informative References . . . . .	<a href="#">1</a>
<a href="#">20.</a>	Author's Address . . . . .	<a href="#">1</a>



## **1. Introduction**

This document specifies version two of the message protocol used in ONC Remote Procedure Call (RPC). The message protocol is specified with the eXternal Data Representation (XDR) language [[RFC4506](#)]. This document assumes that the reader is familiar with XDR. It does not attempt to justify remote procedure calls systems or describe their use. The paper by Birrell and Nelson [[XRPC](#)] is recommended as an excellent background for the remote procedure call concept.

## **2. Changes since [RFC 1831](#)**

This document obsoletes [RFC 1831](#) as the authoritative document describing RPC, without introducing any over-the-wire protocol changes. The main changes from [RFC 1831](#) are:

- o Addition of an Appendix which describes how an implementor can request new RPC program numbers, authentication flavor numbers and authentication status numbers from IANA, rather than from Sun Microsystems
- o Addition of an "IANA Considerations" section which describes past number assignment policy and how IANA is intended to assign them in the future
- o Clarification of the RPC Language Specification to match current usage
- o Enhancement of the "Security Considerations" section to reflect experience with strong security flavors
- o Specification of new authentication errors that are in common use in modern RPC implementations
- o Updates for the latest IETF intellectual property statements

## **3. Terminology**

This document discusses clients, calls, servers, replies, services, programs, procedures, and versions. Each remote procedure call has two sides: an active client side that makes the call to a server, which sends back a reply. A network service is a collection of one or more remote programs. A remote program implements one or more remote procedures; the procedures, their parameters, and results are documented in the specific program's protocol specification. A server may support more than one version of a remote program in order to be compatible with changing protocols.



For example, a network file service may be composed of two programs. One program may deal with high-level applications such as file system access control and locking. The other may deal with low-level file input and output and have procedures like "read" and "write". A client of the network file service would call the procedures associated with the two programs of the service on behalf of the client.

The terms client and server only apply to a particular transaction; a particular hardware entity (host) or software entity (process or program) could operate in both roles at different times. For example, a program that supplies remote execution service could also be a client of a network file service.

#### **4. The RPC Model**

The ONC RPC protocol is based on the remote procedure call model, which is similar to the local procedure call model. In the local case, the caller places arguments to a procedure in some well-specified location (such as a register window). It then transfers control to the procedure, and eventually regains control. At that point, the results of the procedure are extracted from the well-specified location, and the caller continues execution.

The remote procedure call model is similar. One thread of control logically winds through two processes: the caller's process, and a server's process. The caller process first sends a call message to the server process and waits (blocks) for a reply message. The call message includes the procedure's parameters, and the reply message includes the procedure's results. Once the reply message is received, the results of the procedure are extracted, and caller's execution is resumed.

On the server side, a process is dormant awaiting the arrival of a call message. When one arrives, the server process extracts the procedure's parameters, computes the results, sends a reply message, and then awaits the next call message.

In this model, only one of the two processes is active at any given time. However, this model is only given as an example. The ONC RPC protocol makes no restrictions on the concurrency model implemented, and others are possible. For example, an implementation may choose to have RPC calls be asynchronous, so that the client may do useful work while waiting for the reply from the server. Another possibility is to have the server create a separate task to process an incoming call, so that the original server can be free to receive other requests.





There are a few important ways in which remote procedure calls differ from local procedure calls:

- o Error handling: failures of the remote server or network must be handled when using remote procedure calls.
- o Global variables and side-effects: since the server does not have access to the client's address space, hidden arguments cannot be passed as global variables or returned as side effects.
- o Performance: remote procedures usually operate one or more orders of magnitude slower than local procedure calls.
- o Authentication: since remote procedure calls can be transported over unsecured networks, authentication may be necessary. Authentication prevents one entity from masquerading as some other entity.

The conclusion is that even though there are tools to automatically generate client and server libraries for a given service, protocols must still be designed carefully.

## **5. Transports and Semantics**

The RPC protocol can be implemented on several different transport protocols. The scope of the definition of the RPC protocol excludes how a message is passed from one process to another, and includes only the specification and interpretation of messages. However, the application may wish to obtain information about (and perhaps control over) the transport layer through an interface not specified in this document. For example, the transport protocol may impose a restriction on the maximum size of RPC messages, or it may be stream-oriented like TCP [[RFC793](#)] with no size limit. The client and server must agree on their transport protocol choices.

It is important to point out that RPC does not try to implement any kind of reliability and that the application may need to be aware of the type of transport protocol underneath RPC. If it knows it is running on top of a reliable transport such as TCP, then most of the work is already done for it. On the other hand, if it is running on top of an unreliable transport such as UDP [[RFC768](#)], it must implement its own time-out, retransmission, and duplicate detection policies as the RPC protocol does not provide these services.

Because of transport independence, the RPC protocol does not attach specific semantics to the remote procedures or their execution requirements. Semantics can be inferred from (but should be



explicitly specified by) the underlying transport protocol. For example, consider RPC running on top of an unreliable transport such as UDP. If an application retransmits RPC call messages after time-outs, and does not receive a reply, it cannot infer anything about the number of times the procedure was executed. If it does receive a reply, then it can infer that the procedure was executed at least once.

A server may wish to remember previously granted requests from a client and not regrant them in order to insure some degree of execute-at-most-once semantics. A server can do this by taking advantage of the transaction ID that is packaged with every RPC message. The main use of this transaction ID is by the client RPC entity in matching replies to calls. However, a client application may choose to reuse its previous transaction ID when retransmitting a call. The server may choose to remember this ID after executing a call and not execute calls with the same ID in order to achieve some degree of execute-at-most-once semantics. The server is not allowed to examine this ID in any other way except as a test for equality.

On the other hand, if using a "reliable" transport such as TCP, the application can infer from a reply message that the procedure was executed exactly once, but if it receives no reply message, it cannot assume that the remote procedure was not executed. Note that even if a connection-oriented protocol like TCP is used, an application still needs time-outs and reconnection to handle server crashes.

There are other possibilities for transports besides datagram- or connection-oriented protocols. For example, a request-reply protocol such as [\[VMTP\]](#) is perhaps a natural transport for RPC. ONC RPC currently uses both TCP and UDP transport protocols. [Section 10](#) (Record Marking Standard) describes the mechanism employed by ONC RPC to utilize a connection-oriented, stream-oriented transport such as TCP. The mechanism by which future transports having different structural characteristics should be used to transfer ONC RPC messages should be specified by means of a standards-track RFC, once such additional transports are defined.

## **6. Binding and Rendezvous Independence**

The act of binding a particular client to a particular service and transport parameters is NOT part of this RPC protocol specification. This important and necessary function is left up to some higher-level software.

Implementors could think of the RPC protocol as the jump-subroutine instruction ("JSR") of a network; the loader (binder) makes JSR useful, and the loader itself uses JSR to accomplish its task.



Likewise, the binding software makes RPC useful, possibly using RPC to accomplish this task.

## **7. Authentication**

The RPC protocol provides the fields necessary for a client to identify itself to a service, and vice-versa, in each call and reply message. Security and access control mechanisms can be built on top of this message authentication. Several different authentication protocols can be supported. A field in the RPC header indicates which protocol is being used. More information on specific authentication protocols is in [section 8.2](#): "Authentication, Integrity and Privacy".

## **8. RPC Protocol Requirements**

The RPC protocol must provide for the following:

- o Unique specification of a procedure to be called.
- o Provisions for matching response messages to request messages.
- o Provisions for authenticating the caller to service and vice-versa.

Besides these requirements, features that detect the following are worth supporting because of protocol roll-over errors, implementation bugs, user error, and network administration:

- o RPC protocol mismatches.
- o Remote program protocol version mismatches.
- o Protocol errors (such as misspecification of a procedure's parameters).
- o Reasons why remote authentication failed.
- o Any other reasons why the desired procedure was not called.

### **8.1. RPC Programs and Procedures**

The RPC call message has three unsigned integer fields -- remote program number, remote program version number, and remote procedure number -- which uniquely identify the procedure to be called. Program numbers are administered by a central authority (IANA). Once implementors have a program number, they can implement their remote



program; the first implementation would most likely have the version number 1 but MUST NOT be the number zero. Because most new protocols evolve, a version field of the call message identifies which version of the protocol the caller is using. Version numbers enable support of both old and new protocols through the same server process.

The procedure number identifies the procedure to be called. These numbers are documented in the specific program's protocol specification. For example, a file service's protocol specification may state that its procedure number 5 is "read" and procedure number 12 is "write".

Just as remote program protocols may change over several versions, the actual RPC message protocol could also change. Therefore, the call message also has in it the RPC version number, which is always equal to two for the version of RPC described here.

The reply message to a request message has enough information to distinguish the following error conditions:

- o The remote implementation of RPC does not support protocol version 2. The lowest and highest supported RPC version numbers are returned.
- o The remote program is not available on the remote system.
- o The remote program does not support the requested version number. The lowest and highest supported remote program version numbers are returned.
- o The requested procedure number does not exist. (This is usually a client side protocol or programming error.)
- o The parameters to the remote procedure appear to be garbage from the server's point of view. (Again, this is usually caused by a disagreement about the protocol between client and service.)

## **8.2. Authentication, Integrity and Privacy**

Provisions for authentication of caller to service and vice-versa are provided as a part of the RPC protocol. The call message has two authentication fields, the credential and verifier. The reply message has one authentication field, the response verifier. The RPC protocol specification defines all three fields to be the following opaque type (in the eXternal Data Representation (XDR) language [\[RFC4506\]](#)):





```

enum auth_flavor {
    AUTH_NONE      = 0,
    AUTH_SYS       = 1,
    AUTH_SHORT     = 2,
    AUTH_DH        = 3,
    RPCSEC_GSS     = 6
    /* and more to be defined */
};

struct opaque_auth {
    auth_flavor flavor;
    opaque body<400>;
};

```

In other words, any "opaque\_auth" structure is an "auth\_flavor" enumeration followed by up to 400 bytes which are opaque to (uninterpreted by) the RPC protocol implementation.

The interpretation and semantics of the data contained within the authentication fields is specified by individual, independent authentication protocol specifications.

If authentication parameters were rejected, the reply message contains information stating why they were rejected.

As demonstrated by RPCSEC\_GSS, it is possible for an "auth\_flavor" to also support integrity and privacy.

### **8.3. Program Number Assignment**

Program numbers are given out in groups according to the following chart:

0x00000000	Reserved
0x00000001 - 0x1fffffff	To be assigned by IANA
0x20000000 - 0x3fffffff	Defined by local administrator (some blocks assigned here)
0x40000000 - 0x5fffffff	Transient
0x60000000 - 0x7fffffff	Reserved
0x7f000000 - 0x7fffffff	Assignment outstanding
0x80000000 - 0xffffffff	Reserved

The first group is a range of numbers administered by IANA and should be identical for all sites. The second range is for applications peculiar to a particular site. This range is intended primarily for debugging new programs. When a site develops an application that might be of general interest, that application should be given an assigned number in the first range. Application developers may apply



for blocks of RPC program numbers in the first range by methods described in [Appendix B](#). The third group is for applications that generate program numbers dynamically. The final groups are reserved for future use, and should not be used.

#### **[8.4.](#) Other Uses of the RPC Protocol**

The intended use of this protocol is for calling remote procedures. Normally, each call message is matched with a reply message. However, the protocol itself is a message-passing protocol with which other (non-procedure call) protocols can be implemented.

##### **[8.4.1.](#) Batching**

Batching is useful when a client wishes to send an arbitrarily large sequence of call messages to a server. Batching typically uses reliable byte stream protocols (like TCP) for its transport. In the case of batching, the client never waits for a reply from the server, and the server does not send replies to batch calls. A sequence of batch calls is usually terminated by a legitimate remote procedure call operation in order to flush the pipeline and get positive acknowledgement.

##### **[8.4.2.](#) Broadcast Remote Procedure Calls**

In broadcast protocols, the client sends a broadcast call to the network and waits for numerous replies. This requires the use of packet-based protocols (like UDP) as its transport protocol. Servers that support broadcast protocols usually respond only when the call is successfully processed and are silent in the face of errors, but this varies with the application.

The principles of broadcast RPC also apply to multicasting - an RPC request can be sent to a multicast address.

#### **[9.](#) The RPC Message Protocol**

This section defines the RPC message protocol in the XDR data description language [[RFC4506](#)].

```
enum msg_type {  
    CALL    = 0,  
    REPLY   = 1  
};
```

A reply to a call message can take on two forms: The message was either accepted or rejected.



```
enum reply_stat {
    MSG_ACCEPTED = 0,
    MSG_DENIED   = 1
};
```

Given that a call message was accepted, the following is the status of an attempt to call a remote procedure.

```
enum accept_stat {
    SUCCESS          = 0, /* RPC executed successfully */
    PROG_UNAVAIL     = 1, /* remote hasn't exported program */
    PROG_MISMATCH    = 2, /* remote can't support version # */
    PROC_UNAVAIL     = 3, /* program can't support procedure */
    GARBAGE_ARGS     = 4, /* procedure can't decode params */
    SYSTEM_ERR       = 5 /* e.g. memory allocation failure */
};
```

Reasons why a call message was rejected:

```
enum reject_stat {
    RPC_MISMATCH = 0, /* RPC version number != 2 */
    AUTH_ERROR   = 1 /* remote can't authenticate caller */
};
```

Why authentication failed:

```
enum auth_stat {
    AUTH_OK          = 0, /* success */
    /*
     * failed at remote end
     */
    AUTH_BADCRED      = 1, /* bad credential (seal broken) */
    AUTH_REJECTEDCRED = 2, /* client must begin new session */
    AUTH_BADVERF      = 3, /* bad verifier (seal broken) */
    AUTH_REJECTEDVERF = 4, /* verifier expired or replayed */
    AUTH_TOOWEAK      = 5, /* rejected for security reasons */
    /*
     * failed locally
     */
    AUTH_INVALIDRESP  = 6, /* bogus response verifier */
    AUTH_FAILED       = 7, /* reason unknown */
    /*
     * AUTH_KERB errors; deprecated. See [RFC2695]
     */
    AUTH_KERB_GENERIC = 8, /* kerberos generic error */
    AUTH_TIMEEXPIRE   = 9, /* time of credential expired */
    AUTH_TKT_FILE     = 10, /* problem with ticket file */
    AUTH_DECODE       = 11, /* can't decode authenticator */
};
```



```
    AUTH_NET_ADDR = 12,      /* wrong net address in ticket */
    /*
    * RPCSEC_GSS GSS related errors
    */
    RPCSEC_GSS_CREDPROBLEM = 13, /* no credentials for user */
    RPCSEC_GSS_CTXPROBLEM = 14  /* problem with context */
};
```

As new authentication mechanisms are added, there may be a need for more status codes to support them. IANA will hand out new auth\_stat numbers on a simple first-come, first-served basis as defined in the "IANA Considerations" and [Appendix B](#).

The RPC message:

All messages start with a transaction identifier, xid, followed by a two-armed discriminated union. The union's discriminant is a msg\_type which switches to one of the two types of the message. The xid of a REPLY message always matches that of the initiating CALL message. NB: The xid field is only used for clients matching reply messages with call messages or for servers detecting retransmissions; the service side cannot treat this id as any type of sequence number.

```
struct rpc_msg {
    unsigned int xid;
    union switch (msg_type mtype) {
        case CALL:
            call_body cbody;
        case REPLY:
            reply_body rbody;
    } body;
};
```

Body of an RPC call:

In version 2 of the RPC protocol specification, rpcvers MUST be equal to 2. The fields prog, vers, and proc specify the remote program, its version number, and the procedure within the remote program to be called. After these fields are two authentication parameters: cred (authentication credential) and verf (authentication verifier). The two authentication parameters are followed by the parameters to the remote procedure, which are specified by the specific program protocol.

The purpose of the authentication verifier is to validate the authentication credential. Note that these two items are historically separate, but are always used together as one logical





entity.

```
struct call_body {
    unsigned int rpcvers;      /* must be equal to two (2) */
    unsigned int prog;
    unsigned int vers;
    unsigned int proc;
    opaque_auth cred;
    opaque_auth verf;
    /* procedure specific parameters start here */
};
```

Body of a reply to an RPC call:

```
union reply_body switch (reply_stat stat) {
case MSG_ACCEPTED:
    accepted_reply areply;
case MSG_DENIED:
    rejected_reply rreply;
} reply;
```

Reply to an RPC call that was accepted by the server:

There could be an error even though the call was accepted. The first field is an authentication verifier that the server generates in order to validate itself to the client. It is followed by a union whose discriminant is an enum accept\_stat. The SUCCESS arm of the union is protocol specific. The PROG\_UNAVAIL, PROC\_UNAVAIL, GARBAGE\_ARGS, and SYSTEM\_ERR arms of the union are void. The PROG\_MISMATCH arm specifies the lowest and highest version numbers of the remote program supported by the server.

```
struct accepted_reply {
    opaque_auth verf;
    union switch (accept_stat stat) {
case SUCCESS:
    opaque results[0];
    /*
     * procedure-specific results start here
     */
case PROG_MISMATCH:
    struct {
        unsigned int low;
        unsigned int high;
    } mismatch_info;
default:
    /*
     * Void. Cases include PROG_UNAVAIL, PROC_UNAVAIL,
```



```
        * GARBAGE_ARGS, and SYSTEM_ERR.
        */
        void;
    } reply_data;
};
```

Reply to an RPC call that was rejected by the server:

The call can be rejected for two reasons: either the server is not running a compatible version of the RPC protocol (RPC\_MISMATCH), or the server rejects the identity of the caller (AUTH\_ERROR). In case of an RPC version mismatch, the server returns the lowest and highest supported RPC version numbers. In case of invalid authentication, failure status is returned.

```
union rejected_reply switch (reject_stat stat) {
case RPC_MISMATCH:
    struct {
        unsigned int low;
        unsigned int high;
    } mismatch_info;
case AUTH_ERROR:
    auth_stat stat;
};
```

## **10. Authentication Protocols**

As previously stated, authentication parameters are opaque, but open-ended to the rest of the RPC protocol. This section defines two standard "flavors" of authentication. Implementors are free to invent new authentication types, with the same rules of flavor number assignment as there is for program number assignment. The "flavor" of a credential or verifier refers to the value of the "flavor" field in the opaque\_auth structure. Flavor numbers, like RPC program numbers, are also administered centrally, and developers may assign new flavor numbers by methods described in [Appendix B](#). Credentials and verifiers are represented as variable length opaque data (the "body" field in the opaque\_auth structure).

In this document, two flavors of authentication are described. Of these, Null authentication (described in the next subsection) is mandatory - it MUST be available in all implementations. System authentication (AUTH\_SYS) is described in [Appendix A](#). Implementors MAY include AUTH\_SYS in their implementations to support existing applications. See "Security Considerations" for information about other, more secure, authentication flavors.



### **10.1. Null Authentication**

Often calls must be made where the client does not care about its identity or the server does not care who the client is. In this case, the flavor of the RPC message's credential, verifier, and reply verifier is "AUTH\_NONE". Opaque data associated with "AUTH\_NONE" is undefined. It is recommended that the length of the opaque data be zero.

## **11. Record Marking Standard**

When RPC messages are passed on top of a byte stream transport protocol (like TCP), it is necessary to delimit one message from another in order to detect and possibly recover from protocol errors. This is called record marking (RM). One RPC message fits into one RM record.

A record is composed of one or more record fragments. A record fragment is a four-byte header followed by 0 to  $(2^{31}) - 1$  bytes of fragment data. The bytes encode an unsigned binary number; as with XDR integers, the byte order is from highest to lowest. The number encodes two values -- a boolean which indicates whether the fragment is the last fragment of the record (bit value 1 implies the fragment is the last fragment) and a 31-bit unsigned binary value which is the length in bytes of the fragment's data. The boolean value is the highest-order bit of the header; the length is the 31 low-order bits. (Note that this record specification is NOT in XDR standard form!)

## **12. The RPC Language**

Just as there was a need to describe the XDR data-types in a formal language, there is also need to describe the procedures that operate on these XDR data-types in a formal language as well. The RPC Language is an extension to the XDR language, with the addition of "program", "procedure", and "version" declarations. The keywords "program" and "version" are reserved in the RPC Language, and implementations of XDR compilers MAY reserve these keywords even when provided pure XDR, non-RPC, descriptions. The following example is used to describe the essence of the language.

### **12.1. An Example Service Described in the RPC Language**

Here is an example of the specification of a simple ping program.

```
program PING_PROG {  
    /*  
     * Latest and greatest version  
     */
```



```

    version PING_VERS_PINGBACK {
        void
        PINGPROC_NULL(void) = 0;
        /*
         * Ping the client, return the round-trip time
         * (in microseconds). Returns -1 if the operation
         * timed out.
         */
        int
        PINGPROC_PINGBACK(void) = 1;
    } = 2;

    /*
     * Original version
     */
    version PING_VERS_ORIG {
        void
        PINGPROC_NULL(void) = 0;
    } = 1;
} = 1;

const PING_VERS = 2;      /* latest version */

```

The first version described is PING\_VERS\_PINGBACK with two procedures, PINGPROC\_NULL and PINGPROC\_PINGBACK. PINGPROC\_NULL takes no arguments and returns no results, but it is useful for computing round-trip times from the client to the server and back again. By convention, procedure 0 of any RPC protocol should have the same semantics, and never require any kind of authentication. The second procedure is used for the client to have the server do a reverse ping operation back to the client, and it returns the amount of time (in microseconds) that the operation used. The next version, PING\_VERS\_ORIG, is the original version of the protocol and it does not contain PINGPROC\_PINGBACK procedure. It is useful for compatibility with old client programs, and as this program matures it may be dropped from the protocol entirely.

## **12.2. The RPC Language Specification**

The RPC language is identical to the XDR language defined in [RFC 4506](#), except for the added definition of a "program-def" described below.

```

program-def:
    "program" identifier "{"
        version-def
        version-def *
    "}" "=" constant ";

```





```
version-def:
    "version" identifier "{"
        procedure-def
        procedure-def *
    "}" "=" constant ";"

procedure-def:
    proc-return identifier "(" proc-firstarg
        ("," type-specifier)* ")" "=" constant ";"

proc-return: "void" | type-specifier

proc-firstarg: "void" | type-specifier
```

### **12.3. Syntax Notes**

- o The following keywords are added and cannot be used as identifiers: "program" and "version";
- o A version name cannot occur more than once within the scope of a program definition. Nor can a version number occur more than once within the scope of a program definition.
- o A procedure name cannot occur more than once within the scope of a version definition. Nor can a procedure number occur more than once within the scope of version definition.
- o Program identifiers are in the same name space as constant and type identifiers.
- o Only unsigned constants can be assigned to programs, versions and procedures.
- o Current RPC language compilers do not generally support more than one type-specifier in procedure argument lists; the usual practice is to wrap arguments into a structure.

### **13. IANA Considerations**

The assignment of RPC program numbers, authentication flavor numbers and authentication status numbers has in the past been performed by Sun Microsystems, Inc (Sun). This is inappropriate for an IETF standards-track protocol, as such work is done well by the Internet Assigned Numbers Authority (IANA). This document proposes the transfer of authority over RPC program numbers, authentication flavor



numbers and authentication status numbers described here from Sun Microsystems, Inc. to IANA and proposes how IANA will maintain and assign these numbers. Users of RPC protocols will benefit by having an independent body responsible for these number assignments.

### **13.1. Numbering Requests to IANA**

[Appendix B](#) of this document describes the information to be sent to IANA to request one or more RPC numbers and the rules that apply. IANA should store the request for documentary purposes, and put the following information into the public registry:

- o The short description of purpose and use
- o The program number(s) assigned
- o The short identifier string(s)

### **13.2. Protecting Past Assignments**

Sun has made assignments in both number spaces since the original deployment of RPC. The assignments made by Sun Microsystems are still valid, and will be preserved. Sun will communicate all current assignments in both number spaces to IANA before final handoff of number assignment is done. Current program and auth number assignments are provided in [Appendix C](#). Current authentication status numbers are listed in [Section 9](#) of this document in the "enum auth\_stat" definition.

### **13.3. RPC Number Assignment**

Future IANA practice should deal with the following partitioning of the 32-bit number space as listed in [Section 8.3](#). Detailed information for the administration of the partitioned blocks in [Section 8.3](#). is given below.

#### **13.3.1. To be assigned by IANA**

The first block will be administered by IANA, with previous assignments by Sun protected. Previous assignments were restricted to the range decimal 100000-399999 (0x000186a0 to 0x00061a7f), therefore IANA should begin assignments at decimal 400000. Individual numbers should be granted on a first-come, first-served basis, and blocks should be granted under rules related to the size of the block.



### **13.3.2. Defined by local administrator**

The "Defined by local administrator" block is available for any local administrative domain to use, in a similar manner to IP address ranges reserved for private use. The expected use would be through the establishment of a local domain "authority" for assigning numbers from this range. This authority would establish any policies or procedures to be used within that local domain for use or assignment of RPC numbers from the range. The local domain should be sufficiently isolated that it would be unlikely that RPC applications developed by other local domains could communicate with the domain. This could result in RPC number contention, which would cause one of the applications to fail. In the absence of a local administrator, this block can be utilized in a "Private Use" manner per [[RFC5226](#)].

### **13.3.3. Transient block**

The "Transient" block can be used by any RPC application on a "as available" basis. This range is intended for services that can communicate a dynamically selected RPC program number to clients of the service. Any mechanism can be used to communicate the number. Examples include shared memory when the client and server are located on the same system, or a network message (either RPC or otherwise) that disseminates the selected number.

The transient block is not administered. An RPC service uses this range by selecting a number in the transient range and attempting to register that number with the local system's RPC bindery (see the `RPCBPROC_SET` or `PMAPPROC_SET` procedures in "Binding Protocols for ONC RPC", [[RFC1833](#)]). If successful, no other RPC service was using that number and the RPC Bindery has assigned that number to the requesting RPC application. The registration is valid until the RPC Bindery terminates, which normally would only happen if the system reboots causing all applications, including the RPC service using the transient number, to terminate. If the transient number registration fails, another RPC application is using the number and the requestor must select another number and try again. To avoid conflicts, the recommended method is to select a number randomly from the transient range.

### **13.3.4. Reserved block**

The "Reserved" blocks are available for future use. RPC applications must not use numbers in these ranges unless their use is allowed by future action by the IESG.



### **13.3.5. RPC Number Sub-Blocks**

RPC numbers are usually assigned for specific RPC services. Some applications, however, require multiple RPC numbers for a service. The most common example is an RPC service that needs to have multiple instances of the service active simultaneously at a specific site. RPC does not have an "instance identifier" in the protocol, so either a mechanism must be implemented to multiplex RPC requests amongst various instances of the service, or unique RPC numbers must be used by each instance.

In these cases, the RPC protocol used with the various numbers may be different or the same. The numbers may be assigned dynamically by the application, or as part of a site-specific administrative decision. If possible, RPC services that dynamically assign RPC numbers should use the "Transient" RPC number block defined in [section 2](#). If not possible, RPC number sub-blocks may be requested.

Assignment of RPC Number Sub-Blocks is controlled by the size of the sub-block being requested. "Specification Required" and "IESG Approval" are used as defined by [\[RFC5226\] Section 4.1](#).

Size of sub-block	Assignment Method	Authority
-----	-----	-----
Up to 100 numbers	First Come First Served	IANA
Up to 1000 numbers	Specification Required	IANA
More than 1000 numbers	IESG Approval required	IESG

Note: sub-blocks can be any size. The limits given above are maximums and smaller size sub-blocks are allowed.

Sub-blocks sized up to 100 numbers may be assigned by IANA on a First Come First Served basis. The RPC Service Description included in the range must include an indication of how the sub-block is managed. At a minimum, the statement should indicate whether the sub-block is used with a single RPC protocol or multiple RPC protocols, and whether the numbers are dynamically assigned or statically (through administrative action) assigned.

Sub-blocks of up to 1000 numbers must be documented in detail. The documentation must describe the RPC protocol or protocols that are to be used in the range. It must also describe how the numbers within the sub-block are to be assigned or used.

Sub-blocks sized over 1000 numbers must be documented as described above, and the assignment must be approved by the IESG. It is expected that this will be rare.





In order to avoid multiple requests of large blocks of numbers the following rule is proposed.

Requests up to and including 100 RPC numbers are handled via the First Come First Served assignment method. This 100 number threshold applies to the total number of RPC numbers assigned to an individual or entity. For example, if an individual or entity first requests say 70 numbers, and then later requests 40 numbers, then the request for the 40 numbers will be assigned via the Specification Required method. As long as the total number of numbers assigned does not exceed 1000, IANA is free to waive the Specification Required assignment for incremental requests of less than 100 numbers.

If an individual or entity has under 1000 numbers and later requests an additional set of numbers such that the individual or entity would be granted over 1000 numbers, then the additional request will require IESG Approval.

#### **13.4. RPC Authentication Flavor Number Assignment**

The second number space is the authentication mechanism identifier, or "flavor", number. This number is used to distinguish between various authentication mechanisms which can be optionally used with an RPC message. An authentication identifier is used in the "flavor" field of the "opaque\_auth" structure.

##### **13.4.1. Assignment Policy**

[Appendix B](#) of this document describes the information to be sent to IANA to request one or more RPC auth numbers and the rules that apply. IANA should store the request for documentary purposes, and put the following information into the public registry:

- o The short identifier string(s)
- o The auth number(s) assigned
- o The short description of purpose and use

##### **13.4.2. Auth Flavors vs. Pseudo-flavors**

Recent progress in RPC security has moved away from new auth flavors as used by AUTH\_DH [[DH](#)], and focused on using the existing RPCSEC\_GSS [[RFC2203](#)] flavor and inventing novel GSS-API mechanisms which can be used with it. Even though RPCSEC\_GSS is an assigned authentication flavor, use of a new RPCSEC\_GSS mechanism with NFS ([\[RFC1094\]](#) [[RFC1813](#)] and [[RFC3530](#)]) will require the registration of 'pseudo-



flavors' which are used to negotiate security mechanisms in an unambiguous way, as defined by [\[RFC2623\]](#). Existing pseudo-flavors have been granted in the decimal range 390000-390255. New pseudo-flavor requests should be granted by IANA within this block on a First Come First Served basis.

For non-pseudo-flavor requests, IANA should begin granting RPC authentication flavor numbers at 400000 on a First Come First Served basis to avoid conflicts with currently granted numbers.

For authentication flavors or RPCSEC\_GSS mechanisms to be used on the Internet, it is strongly advised that an informational or standards-track RFC be published describing the authentication mechanism behaviour and parameters.

### **[13.5.](#) Authentication Status Number Assignment**

The final number space is the authentication status or "auth\_stat" values which describe the nature of a problem found during an attempt to authenticate or validate authentication. The complete initial list of these values is found in [Section 9](#) of this document, in the "auth\_stat" enum listing. It is expected that it will be rare to add values, but that a small number of new values may be added from time to time as new authentication flavors introduce new possibilities. Numbers should be granted on a First Come First Served basis to avoid conflicts with currently granted numbers.

#### **[13.5.1.](#) Assignment Policy**

[Appendix B](#) of this document describes the information to be sent to IANA to request one or more auth\_stat values and the rules that apply. IANA should store the request for documentary purposes, and put the following information into the public registry:

- o The short identifier string(s)
- o The auth\_stat number(s) assigned
- o The short description of purpose and use

### **[14.](#) Security Considerations**

AUTH\_SYS as described in [Appendix A](#) is known to be insecure due to the lack of a verifier to permit the credential to be validated. AUTH\_SYS SHOULD NOT be used for services which permit clients to modify data. AUTH\_SYS MUST NOT be specified as RECOMMENDED or REQUIRED for any standards-track RPC service.



AUTH\_DH as mentioned in sections [8.2](#) and [13.4.2](#) is considered obsolete and insecure; see [[RFC2695](#)]. AUTH\_SYS SHOULD NOT be used for services which permit clients to modify data. AUTH\_DH MUST NOT be specified as RECOMMENDED or REQUIRED for any standards-track RPC service.

[RFC2203] defines a new security flavor, RPCSEC\_GSS, which permits GSS-API [[RFC2743](#)] mechanisms to be used for securing RPC. All non-trivial RPC programs developed in the future should implement RPCSEC\_GSS-based security appropriately. [[RFC2623](#)] describes how this was done for a widely deployed RPC program.

Standards-track RPC services MUST mandate support for RPCSEC\_GSS, and MUST mandate support for an authentication pseudo-flavor with appropriate levels of security, depending on the need for simple authentication, integrity a.k.a. non-repudiation, or data privacy.

## **[15. Appendix A: System Authentication](#)**

The client may wish to identify itself, for example, as it is identified on a UNIX(tm) system. The flavor of the client credential is "AUTH\_SYS". The opaque data constituting the credential encodes the following structure:

```
struct authsys_parms {
    unsigned int stamp;
    string machinename<255>;
    unsigned int uid;
    unsigned int gid;
    unsigned int gids<16>;
};
```

The "stamp" is an arbitrary ID which the caller machine may generate. The "machinename" is the name of the caller's machine (like "krypton"). The "uid" is the caller's effective user ID. The "gid" is the caller's effective group ID. The "gids" is a counted array of groups which contain the caller as a member. The verifier accompanying the credential should have "AUTH\_NONE" flavor value (defined above). Note this credential is only unique within a particular domain of machine names, uids, and gids.

The flavor value of the verifier received in the reply message from the server may be "AUTH\_NONE" or "AUTH\_SHORT". In the case of "AUTH\_SHORT", the bytes of the reply verifier's string encode an opaque structure. This new opaque structure may now be passed to the server instead of the original "AUTH\_SYS" flavor credential. The server may keep a cache which maps shorthand opaque structures



(passed back by way of an "AUTH\_SHORT" style reply verifier) to the original credentials of the caller. The caller can save network bandwidth and server cpu cycles by using the shorthand credential.

The server may flush the shorthand opaque structure at any time. If this happens, the remote procedure call message will be rejected due to an authentication error. The reason for the failure will be "AUTH\_REJECTEDCRED". At this point, the client may wish to try the original "AUTH\_SYS" style of credential.

It should be noted that use of this flavor of authentication does not guarantee any security for the users or providers of a service, in itself. The authentication provided by this scheme can be considered legitimate only when applications using this scheme and the network can be secured externally, and privileged transport addresses are used for the communicating end-points (an example of this is the use of privileged TCP/UDP ports in Unix systems - note that not all systems enforce privileged transport address mechanisms).

#### **16. [Appendix B](#): Requesting RPC-related numbers from IANA**

RPC program numbers, authentication flavor numbers and authentication status numbers which must be unique across all networks are assigned by the Internet Assigned Number Authority. To apply for a single number or a block of numbers, electronic mail must be sent to IANA <iana@iana.org> with the following information:

- o The type of number(s) (program number or authentication flavor number or authentication status number) sought
- o How many numbers are sought
- o The name of person or company which will use the number
- o An "identifier string" which associates the number with a service
- o Email address of the contact person for the service which will be using the number.
- o A short description of the purpose and use of the number
- o If an authentication flavor number is sought, and the number will be a 'pseudo-flavor' intended for use with RPCSEC\_GSS and NFS, mappings analogous to those in [Section 4.2 of \[RFC2623\]](#) are required.

Specific numbers cannot be requested. Numbers are assigned on a





First Come First Served basis.

For all RPC authentication flavor and authentication status numbers to be used on the Internet, it is strongly advised that an informational or standards-track RFC be published describing the authentication mechanism behaviour and parameters.

## 17. [Appendix C](#): Current number assignments

```
#
# Sun-assigned RPC numbers
#
# Description/Owner          RPC Program Number      Short Name
# -----
portmapper                   100000      pmapprog portmap rpcbind
remote stats                 100001      rstatprog
remote users                 100002      rusersprog
nfs                          100003      nfs
yellow pages (NIS)          100004      ypprog ypserv
mount demon                 100005      mountprog
remote dbx                  100006      dbxprog
yp binder (NIS)             100007      ypbindprog ypbind
shutdown msg                100008      wall
yppasswd server             100009      yppasswdprog yppasswdd
ether stats                 100010      etherstatprog
disk quotas                 100011      rquota
spray packets               100012      spray
3270 mapper                 100013      ibm3270prog
RJE mapper                  100014      ibmrjeprog
selection service           100015      selnsvcp prog
remote database access      100016      rdatabaseprog
remote execution            100017      rexec
Alice Office Automation     100018      aliceprog
scheduling service          100019      schedprog
local lock manager          100020      lockprog llockmgr
network lock manager        100021      netlockprog nlockmgr
x.25 inr protocol          100022      x25prog
status monitor 1            100023      statmon1
status monitor 2            100024      statmon2
selection library           100025      selnlibprog
boot parameters service     100026      bootparam
mazewars game               100027      mazeprog
yp update (NIS)             100028      ypupdateprog ypupdate
key server                  100029      keyserveprog
secure login                100030      securecmdprog
nfs net forwarder init      100031      netfwdiprog
nfs net forwarder trans     100032      netfwdtprog
sunlink MAP                 100033      sunlinkmap
```



network monitor	100034	netmonprog
lightweight database	100035	dbaseprog
password authorization	100036	pwdauthprog
translucent file svc	100037	tfsprog
nse server	100038	nseprog
nse activate daemon	100039	nse_activate_prog
sunview help	100040	sunview_help_prog
pnf install	100041	pnf_prog
ip addr allocator	100042	ipaddr_alloc_prog
show filehandle	100043	filehandle
MVS NFS mount	100044	mvsnfsprog
remote user file operations	100045	rem_fileop_user_prog
batched ypupdate	100046	batch_ypupdateprog
network execution mgr	100047	nem_prog
raytrace/mandelbrot remote daemon	100048	raytrace_rd_prog
raytrace/mandelbrot local daemon	100049	raytrace_ld_prog
remote group file operations	100050	rem_fileop_group_prog
remote system file operations	100051	rem_fileop_system_prog
remote system role operations	100052	rem_system_role_prog
gpd lego fb simulator	100053	[unknown]
gpd simulator interface	100054	[unknown]
ioadmd	100055	ioadmd
filemerge	100056	filemerge_prog
Name Binding Program	100057	namebind_prog
sunlink NJE	100058	njeprog
MVSNFS get attribute service	100059	mvsattrprog
SunAccess/SunLink resource manager	100060	rmgrprog
UID allocation service	100061	uidallocprog
license broker	100062	lbserverprog
NETlicense client binder	100063	lbbinderprog
GID allocation service	100064	gidallocprog
SunIsam	100065	sunisamprog
Remote Debug Server	100066	rdbsrvprog
Network Directory Daemon	100067	[unknown]
Network Calendar Program	100068	cmsd cm
ypxfrd	100069	ypxfrd
rpc.timed	100070	timedprog
bugtraqd	100071	bugtraqd
	100072	[unknown]
Connectathon Billboard - NFS	100073	[unknown]
Connectathon Billboard - X	100074	[unknown]
Sun tool for scheduling rooms	100075	schedroom
Authentication Negotiation	100076	authnegotiate_prog
Database manipulation	100077	attribute_prog
Kerberos authentication daemon	100078	kerbprog
Internal testing product (no name)	100079	[unknown]
Sun Consulting Special	100080	autodump_prog
Event protocol	100081	event_svc



bugtraq_qd	100082	bugtraq_qd
ToolTalk and Link Service Project	100083	database service
Consulting Services	100084	[unknown]
Consulting Services	100085	[unknown]
Consulting Services	100086	[unknown]
Jupiter Administration	100087	adm_agent admin
	100088	[unknown]
	100089	[unknown]
Dual Disk support	100090	libdsd/dsd
DocViewer 1.1	100091	[unknown]
ToolTalk	100092	remote_activation_svc
Consulting Services	100093	host_checking
SNA peer-to-peer	100094	[unknown]
Roger Riggs	100095	searchit
Robert Allen	100096	mesgtool
SNA	100097	[unknown]
SISU	100098	networked version of CS5
NFS Automount File System	100099	autofs
	100100	msgboard
event dispatching agent [eventd]	100101	netmgt_eventd_prog
statistics/event logger [netlogd]	100102	netmgt_netlogd_prog
topology display manager [topology]	100103	netmgt_topology_prog
syncstat agent [syncstatd]	100104	netmgt_syncstatd_prog
ip packet stats agent [ippktd]	100105	netmgt_ippktd_prog
netmgt config agent [configd]	100106	netmgt_configd_prog
restat agent [restatd]	100107	netmgt_restatd_prog
lpq agent [lprstatd]	100108	netmgt_lprstatd_prog
netmgt activity agent [mgtlogd]	100109	netmgt_mgtlogd_prog
proxy DECnet NCP agent [proxydni]	100110	netmgt_proxydni_prog
topology mapper agent [mapperd]	100111	netmgt_mapperd_prog
netstat agent [netstatd]	100112	netmgt_netstatd_prog
sample netmgt agent [sampled]	100113	netmgt_sampled_prog
X.25 statistics agent [vcstatd]	100114	netmgt_vcstatd_prog
Frame Relay	100128	[unknown]
PPP agent	100129	[unknown]
localhad	100130	rpc.localhad
layers2	100131	na.layers2
token ring agent	100132	na.tr
related to lockd and statd	100133	nsm_addr
Kerberos project	100134	kwarn
etherif2	100135	na.etherif2
hostmem2	100136	na.hostmem2
iostat2	100137	na.iostat2
snmpv2	100138	na.snmpv2
Cooperative Console	100139	cc_sender
na.cpushat	100140	na.cpushat
Sun Cluster SC3.0	100141	rgmd_receptionist
	100142	fed



Network Storage	100143	rdc
Sun Cluster products	100144	nafo
SunCluster 3.0	100145	scadmd
ASN.1	100146	amiserv
	100147	amiaux # BER and DER encode and decode
Delegate Management Server	100148	dm
	100149	rkstat
	100150	ocfserv
	100151	sccheckd
	100152	autoclientd
	100153	sunvts
	100154	ssmond
	100155	smsserverd
	100156	test1
	100157	test2
	100158	test3
	100159	test4
	100160	test5
	100161	test6
	100162	test7
	100163	test8
	100164	test9
	100165	test10
	100166	nfsmapid
	100167	SUN_WBEM_C_CIMON_HANDLE
	100168	sacmmd
	100169	fmd_adm
	100170	fmd_api
	100171	[unknown]
	100172	idmapd
<a href="#">[available]</a>	100173 - 100174	
snmptrap	100175	na.snmptrap
<a href="#">[available]</a>	100176-100199	
<a href="#">[available]</a>	100200	
MVS/NFS Memory usage stats server	100201	[unknown]
Netapp	100202-100207	
<a href="#">[available]</a>	100208-100210	
8.0 SunLink SNA RJE	100211	[unknown]
8.0 SunLink SNA RJE	100212	[unknown]
	100213	ShowMe
	100214	[unknown]
	100215	[unknown]
AUTH_RSA Key service	100216	keyrsa
SunSelect PC license service	100217	[unknown]
WWCS (Corporate)	100218	sunsolve
	100219	cstatd





X/Open Federated Naming	100220	xfn_server_prog
Kodak Color Management System	100221	kcs_network_io kcs
HA-DBMS	100222	ha_dbms_serv
	100223-100225	[unknown]
	100226	hafaultd
NFS ACL Service	100227	nfs_acl
distributed lock manager	100228	dlmd
	100229	metad
	100230	metamhd
	100231	nfsauth
	100232	sadmin
	100233	ufsd
	100234	grpserve
	100235	cachefs
	100236	mshprog Media_Server
	100237	ihnamed
	100238	ihnetd
	100239	ihsecured
	100240	ihclassmgrd
	100241	ihrepositoryd
	100242	metamedd rpc.metamedd
	100243	contentmanager cm
	100244	symon
	100245	pld genesil
	100246	ctid
		cluster_transport_interface
	100247	ccd
		cluster_configuration_db
	100248	pmfd
	100249	dmi2_client
	100250	mfs_admin
	100251	ndshared_unlink
	100252	ndshared_touch
	100253	ndshared_slink
	100254	cbs control_board_server
	100255	skiserv
	100256	nfsxa nfsxattr
	100257	ndshared_disable
	100258	ndshared_enable
	100259	sms_account_admin
	100260	sms_modem_admin
	100261	sms_r_login
	100262	sms_r_subaccount_mgt
	100263	sms_service_admin
	100264	session_admin
	100265	canci_ancs_program
	100266	canci_sms_program
	100267	msmp



	100268	halck
	100269	halogmsg
	100270	nfs_id_map
	100271	ncall
	100272	hmip
	100273	repl_mig
	100274	repl_mig_cb
NIS+	100300	nisplus
NIS+	100301	nis_cachemgr
NIS+ call back protocol	100302	[unknown]
NIS+ Password Update Daemon	100303	nispaswdd
FNS context update in NIS	100304	fnsypd
	100305	[unknown]
	100306	[unknown]
	100307	[unknown]
	100308	[unknown]
	100309	[unknown]
<a href="#">[available]</a>	100310 - 100398	
nfscksum	100399	nfscksum
network utilization agent	100400	netmgt_netu_prog
network rpc ping agent	100401	netmgt_rping_prog
	100402	na.shell
picsprint	100403	na.picslp
	100404	traps
	100405 - 100409	[unknown]
	100410	jdsagent
	100411	na.haconfig
	100412	na.halhost
	100413	na.hadtsrv
	100414	na.hamdstat
	100415	na.neoadmin
	100416	ex1048prog
	100417	rdmaconfig
IETF NFSv4 Working Group - FedFS	100418 - 100421	
	100422	mdcommd
	100423	kiprop krb5_iprop
	100424	stsf
<a href="#">[available]</a>	100425 - 100499	
Sun Microsystems	100500 - 100531	[unknown]
	100532	ucmmstate
	100533	srcmd
<a href="#">[available]</a>	100534 - 100999	
nse link daemon	101002	nselinktool
nse link application	101003	nselinkapp
<a href="#">[available]</a>	101004 - 101900	
	101901	[unknown]
<a href="#">[available]</a>	101902 - 101999	
AssetLite	102000	[unknown]



PagerTool	102001	[unknown]
Discover	102002	[unknown]
<a href="#">[available]</a>	102003 - 105000	
ShowMe	105001	sharedapp
Registry	105002	REGISTRY_PROG
Print-server	105003	print-server
Proto-server	105004	proto-server
Notification-server	105005	notification-server
Transfer-agent-server	105006	transfer-agent-server
<a href="#">[available]</a>	105007 - 110000	
	110001	tsolrpcb
	110002	tsolpeerinfo
	110003	tsolboot
	120001	cmip na.cmip
	120002	na.osidiscover
	120003	cmiptrap
<a href="#">[available]</a>	120004 - 120099	
	120100	eserver
	120101	repserver
	120102	swserver
	120103	dmd
	120104	ca
<a href="#">[available]</a>	120105 - 120125	
	120126	nf_fddi
	120127	nf_fddismt7_2
<a href="#">[available]</a>	120128 - 150000	
pc passwd authorization	150001	pcnfsdprog
TOPS name mapping	150002	[unknown]
TOPS external attribute storage	150003	[unknown]
TOPS hierarchical file system	150004	[unknown]
TOPS NFS transparency extensions	150005	[unknown]
PC NFS License	150006	pcnfslicense
RDA	150007	rdaprogram
WabiServer	150008	wsprog
WabiServer	150009	wsrlprog
<a href="#">[available]</a>	150010 - 160000	
	160001	nihon-cm
	160002	nihon-ce
<a href="#">[available]</a>	160003 - 170099	
	170100	domf_daemon0
	170101	domf_daemon1
	170102	domf_daemon2
	170103	domf_daemon3
	170104	domf_daemon4
	170105	domf_daemon5
<a href="#">[available]</a>	170106 - 179999	
	180000	cecprog
	180001	cecsysprog



180002	cec2cecprog
180003	cesprog
180004	ces2cesprog
180005	cet2cetprog
180006	cet2cetdoneprog
180007	cetcomprog
180008	cetsysprog
180009	cghapresenceprog
180010	cgdmssyncprog
180011	cgdmcsnsccliprog
180012	cgdmcrscscliprog
180013	cgdmcrscssvcproG
180014	chmprog
180015	chmsysprog
180016	cracsapiproG
180017	ckptmprog
180018	crimcomponentprog
180019	crimqueryprog
180020	crimsecondaryprog
180021	crimservicesprog
180022	crimsyscomponentprog
180023	crimsysservicesprog
180024	csmagtapiproG
180025	csmagtcallbackprog
180026	csmreplicaproG
180027	csmsrvprog
180028	cssccltprog
180029	csscsvrprog
180030	csscopresultprog
180031 - 199999	
200000	pyramid_nfs
200001	pyramid_reserved
200002	cadds_image
200003	stellar_name_prog
200004	[unknown]
200005	[unknown]
200006	pac1
200007	lookupids
200008	ax_statd_prog
200009	ax_statd2_prog
200010	edm
200011	dtedirwd
200012	[unknown]
200013	[unknown]
200014	[unknown]
200015	[unknown]
200016	easerpcd
200017	rlxnfs

[[available](#)]





	200018	sascuiddprog
	200019	knfsd
	200020	ftnfsd ftnfsd_program
	200021	ftsyncd ftsyncd_program
	200022	ftstatd ftstatd_program
	200023	exportmap
	200024	nfs_metadata
[ <a href="#">available</a> ]	200025 - 200200	
	200201	ecoad
	200202	eamon
	200203	ecolic
	200204	cs_printstatus_svr
	200205	ecodisc
[ <a href="#">available</a> ]	200206 - 300000	
	300001	adt_rflockprog
	300002	columbine1
	300003	system33_prog
	300004	frame_prog1
	300005	uimxprog
	300006	rvd
	300007	entombing daemon
	300008	account mgmt system
	300009	frame_prog2
	300010	beeper access
	300011	dptuprog
	300012	mx-bcp
	300013	instrument-file-access
	300014	file-system-statistics
	300015	unify-database-server
	300016	tmd_msg
	300017	[unknown]
	300018	[unknown]
	300019	automounter access
	300020	lock server
	300021	[unknown]
	300022	office-automation-1
	300023	office-automation-2
	300024	office-automation-3
	300025	office-automation-4
	300026	office-automation-5
	300027	office-automation-6
	300028	office-automation-7
	300029	local-data-manager
	300030	chide
	300031	csi_program
	300032	[unknown]
	300033	online-help
	300034	case-tool



300035	delta
300036	rgi
300037	instrument-config-server
300038	[unknown]
300039	[unknown]
300040	dtia-rpc-server
300041	cms
300042	viewer
300043	aqm
300044	exclaim
300045	masterplan
300046	fig_tool
300047	[unknown]
300048	[unknown]
300049	[unknown]
300050	remote-lock-manager
300051	[unknown]
300052	gdebug
300053	ldebug
300054	rscanner
300055	[unknown]
300056	[unknown]
300057	[unknown]
300058	[unknown]
300059	[unknown]
300060	[unknown]
300061	[unknown]
300062	[unknown]
300063	[unknown]
300064	[unknown]
300065	[unknown]
300066	nSERVER
300067	[unknown]
300068	[unknown]
300069	[unknown]
300070	[unknown]
300071	BioStation
300072	[unknown]
300073	NetProb
300074	Logging
300075	Logging
300076	[unknown]
300077	[unknown]
300078	[unknown]
300079	[unknown]
300080	[unknown]
300081	[unknown]
300082	sw_twin



300083	remote_get_login
300084	odcprog
300085	[unknown]
300086	[unknown]
300087	[unknown]
300088	[unknown]
300089	[unknown]
300090	[unknown]
300091	smartdoc
300092	superping
300093	distributed-chembench
300094	uacman/alfil-uacman
300095	ait_rcagent_prog
300096	ait_rcagent_appl_prog
300097	smart
300098	ecoprogram
300099	leonardo
300100	[unknown]
300101	[unknown]
300102	[unknown]
300103	[unknown]
300104	[unknown]
300105	[unknown]
300106	[unknown]
300107	[unknown]
300108	wingz
300109	teidan
300110	[unknown]
300111	[unknown]
300112	[unknown]
300113	[unknown]
300114	[unknown]
300115	[unknown]
300116	cadc_fhlockprog
300117	highscan
300118	[unknown]
300119	[unknown]
300120	[unknown]
300121	opennavigator
300122	aarpcxfer
300123	[unknown]
300124	[unknown]
300125	[unknown]
300126	groggs
300127	licsrv
300128	issdemon
300129	[unknown]
300130	maximize



300131	cgm_server
300132	[unknown]
300133	agent_rpc
300134	docmaker
300135	docmaker
300136	[unknown]
300137	[unknown]
300138	[unknown]
300139	iesx
300140	[unknown]
300141	[unknown]
300142	[unknown]
300143	[unknown]
300144	smart-mbs
300145	[unknown]
300146	[unknown]
300147	docimage
300148	[unknown]
300149	dmc-interface
300150	[unknown]
300151	jss
300152	[unknown]
300153	arimage
300154	xdb-workbench
300155	frontdesk
300156	dmc
300157	expressight-6000
300158	graph service program
300159	[unknown]
300160	[unknown]
300161	[unknown]
300162	[unknown]
300163	[unknown]
300164	[unknown]
300165	[unknown]
300166	[unknown]
300167	[unknown]
300168	[unknown]
300169	[unknown]
300170	[unknown]
300171	[unknown]
300172	[unknown]
300173	[unknown]
300174	[unknown]
300175	[unknown]
300176	rlpr
300177	nx_hostdprog
300178	netuser-x





300179	rmntprog
300180	[unknown]
300181	mipe
300182	[unknown]
300183	collectorprog
300184	uslookup_PROG
300185	viewstation
300186	iate
300187	[unknown]
300188	[unknown]
300189	[unknown]
300190	imsvtprog
300191	[unknown]
300192	[unknown]
300193	[unknown]
300194	pmdb
300195	pmda
300196	[unknown]
300197	[unknown]
300198	trend_idbd
300199	rres
300200	sd.masterd
300201	sd.executiond
300202	sd.listend
300203	sd.reserve1
300204	sd.reserve2
300205	msbd
300206	stagedprog
300207	mountprog
300208	watchdprog
300209	pms
300210	[unknown]
300211	session_server_program
300212	session_program
300213	debug_serverprog
300214	[unknown]
300215	[unknown]
300216	paceprog
300217	[unknown]
300218	mbus
300219	aframes2ps
300220	npartprog
300221	cm1server
300222	cm1bridge
300223	sailfrogfaxprog
300224	sailfrogphoneprog
300225	sailfrogvmailprog
300226	wserviceprog arcstorm



300227	hld
300228	alive
300229	radsp
300230	radavx
300231	radview
300232	rsys_prog
300233	rsys_prog
300234	fm_rpc_prog
300235	aries
300236	uapman
300237	ddman
300238	top
300239	[unknown]
300240	trendlink
300241	licenseprog
300242	statuslicenseprog
300243	oema_rmpf_svc
300244	oema_smpf_svc
300245	oema_rmsg_svc
300246	grapes-sd
300247	ds_master
300248	ds_transfer
300249	ds_logger
300250	ds_query
300251	[unknown]
300252	[unknown]
300253	nsd_prog
300254	browser
300255	epoch
300256	floorplanner
300257	reach
300258	tactic
300259	cachescientific1
300260	cachescientific2
300261	desksrc_prog
300262	photo3d1
300263	photo3d2
300264	[unknown]
300265	soundmgr
300266	s6k
300267	aims_referenced_ text_processor
300268	xess
300269	ds_queue
300270	[unknown]
300271	orionscanplus
300272	openlink-xx
300273	kbmsprog



300274	[unknown]
300275	futuresource
300276	the_xprt
300277	cmg_srvprog
300278	[unknown]
300279	[unknown]
300280	front
300281	[unknown]
300282	[unknown]
300283	[unknown]
300284	conmanprog
300285	jincv2
300286	isls
300287	systemstatprog
300288	fxpsprog
300289	callpath
300290	axess
300291	armor_rpcd
300292	armor_dictionary_rpcd
300293	armor_miscd
300294	filetransfer_prog
300295	bl_swda
300296	bl_hwda
300297	[unknown]
300298	[unknown]
300299	[unknown]
300300	filemon
300301	acunetprog
300302	rbuild
300303	assistprog
300304	tog
300305	[unknown]
300306	sns7000
300307	igprog
300308	tgprog
300309	plc
300310	pxman pxlsprog
300311	hde_server hdeserver
300312	tsslicenseprog
300313	rpc.explorerd
300314	chrd
300315	tbisam
300316	tbis
300317	adsprog
300318	sponsorprog
300319	querycmprog
300320	[unknown]
300321	[unknown]



300322	mobili
300323	sld
	service_locator_daemon
300324	linkprog
300325	codexdaemonprog
300326	drprog
300327	ressys_commands
300328	stamp
300329	matlab
300330	sched1d
300331	upcprog
300332	xferbkch
300333	xfer
300334	qbthd
300335	qbabort
300336	lsd
300337	geomgrd
300338	generic_fts
300339	ft_ack
300340	lymb
300341	vantage
300342	cltstd clooptstdprog
300343	clui clui_prog
300344	testerd tstdprog
300345	extsim
300346	cmd_dispatch maxm_ems
300347	callpath_receive_program
300348	x3270prog
300349	sbc_lag
300350	sbc_frsa
300351	sbc_frs
300352	atommgr
300353	geostrat
300354	dbvialu6.2
300355	[unknown]
300356	fxncprog
300357	infopolic
300358	[unknown]
300359	aagns
300360	aagms
300361	[unknown]
300362	clariion_mgr
300363	setcimrpc
300364	virtual_protocol_adapter
300365	unibart
300366	uniarch
300367	unifile
300368	unisrex





300369	uniscmd
300370	rsc
300371	set
300372	desaf-ws/key
300373	reelddb
300374	nl
300375	rmd
300376	agcd
300377	rsynd
300378	rcnlib
300379	rcnlib_attach
300380	evergreen_mgmt_agent
300381	fx104prog
300382	rui
	remote_user_interface
300383	ovomd
300384	[unknown]
300385	[unknown]
300386	system_server
300387	pipecs cs_pipeprog
	ppktrpc
300388	uv-net univision
300389	auexe
300390	audip
300391	mqi
300392	eva
300393	eeee_reserved_1
300394	eeee_reserved_2
300395	eeee_reserved_3
300396	eeee_reserved_4
300397	eeee_reserved_5
300398	eeee_reserved_6
300399	eeee_reserved_7
300400	eeee_reserved_8
300401	cprlm
300402	wg_idms_manager
300403	timequota
300404	spiff
300405-300414	ov_oem_svc
300415	ov_msg_ctlg_svc
300416	ov_advt_reg_svc
300417-300424	showkron
300425	daatd
300426	swiftnet
300427	ovomdel
300428	ovomreq
300429	msg_dispatcher
300430	pcshare server



300431	rcvs
300432	fdfserver
300433	bssd
300434	drdd
300435	mif_gutsprog
300436	mif_guiprog
300437	twolfd
300438	twscd
300439	nwsbumv
300440	dgux_mgr
300441	pfxd
300442	tds
300443	ovomadmind
300444	ovomgate
300445	omadmind
300446	nps
300447	npd
300448	tsa
300449	cdaimc
300450-300452	
300453	ckt_implementation
300454	mda-tactical
300455-300458	
300459	attrun
300460	RoadRunner
300461	nas
300462	undeleste
300463	ovacadd
300464	tbdesmai
300465	arguslm
300466	dmd
300467	drd
300468	fm_help
300469	ftransrpc_prog
300470	finrisk
300471	dg_pc_idisched
300472	dg_pc_idiserv
300473	apd
300474	ap_sspd
300475	callpatheventrecorder
300476	flc
300477	dg_osm
300478	dspnamed
300479	iqddsrv
300480	iqjobsrv
300481	tacosxx
300482	wheeldbmg
300483	cnxmgr_nm_prog



300484	cnxmgr_cfg_prog
300485	3dsmapper
300486	ids
300487	imagine_rpc_svc
300488	lfn
300489	salesnet
300490	defaxo
300491	dbqtsd
300492	kms
300493	rpc.iced
300494	calc2s
300495	ptoidprog
300496	docsls
300497	new
300498	collagebdg
300499	ars_server
300500	ars_client
300501	vr_catalog
300502	vr_tdb
300503	ama
300504	evama
300505	conama
300506	service_process
300507	reuse_proxy
300508	mars_ctrl
300509	mars_db
300510	mars_com
300511	mars_admch
300512	tbpipcip
300513	top_acs_svc
300514	inout_svc
300515	csoft_wp
300516	mcfs
300517	eventprog
300518	dg_pc_idimsg
300519	dg_pc_idiaux
300520	atsr_gc
300521	alarm alarm_prog
300522	fts_prog
300523	dcs_prog
300524	ihb_prog
300525	[unknown]
300526	[unknown]
300527	clu_info_prog
300528	rmfm
300529	c2sdocd
300530	interahelp
300531	callpathasynmsgshandler



300532	optix_arc
300533	optix_ts
300534	optix_wf
300535	maxopenc
300536	cev cev_server
300537	sitewideprog
300538	drs
300539	drsdm
300540	dasgate
300541	dcdbd
300542	dcpsd
300543	supportlink_prog
300544	broker
300545	listner
300546	multiaccess
300547	spai_interface
300548	spai_adaption
300549	chimera_ci
	chimera_clientinterface
300550	chimera_pi
	chimera_processinvoker
300551	teamware_fl
	teamware_foundationlevel
300552	teamware_sl
	teamware_systemlevel
300553	teamware_ui
	teamware_userinterface
300554	lprm
300555	mpsprog
	Mensuration_Proxy_Server
300556	mo_symdis
300557	retsideprog
300558	slp
300559	slm-api
300560	im_rpc teamconference
300561	license_prog license
300562	stuple stuple_prog
300563	upasswd_prog
300564	gentranmentorsecurity
300565	gentranmentorprovider
300566	latituded
	latitude_license_server
300567	gentranmentorreq1
300568	gentranmentorreq2
300569	gentranmentorreq3
300570	rj_server
300571	gws-rdb
300572	gws-mpmd





300573	gws-spm
300574	vwcalcd
300575	vworad
300576	vwsybd
300577	vwave
300578	online_assistant
300579	internet_assistant
300580	spawnd
300581	procmgrg
300582	cfgdbd
300583	logutild
300584	ibis
300585	ibisau
300586	aapi
300587	rstrt
300588	hbeat
300589	pcspu
300590	empres
300591	schd_server
	LiveScheduler
300592	path_server
	LiveScheduler
300593	c2sdmd
300594	c2scf
300595	btsas
300596	sdtas
300597	appie
300598	dmi
300599	pacd
	panther software corp daemon
300600	sisd
300601	cpwebserver
300602	wwcommo
300603	mx-mie
300604	mx-mie-debug
300605	idmn
300606	ssrv
300607	vpnserv
300608	samserv
300609	sams_server
300610	chrysalis
300611	ddm
300612	ddm-is
300613	mx-bcp-debug
300614	upmrd
300615	upmdsd
300616	res
300617	colortron



300618	zrs
300619	afpsrv
300620	apxft
300621	nrp
300622	hpid
300623	mailwatch
300624	fos bc_fcrb_receiver
300625	cs_sysadmin_svr
300626	cs_controller_svr
300627	nokia_nms_eai
300628	dbg
300629	remex
300630	cs_bind
300631	idm
300632	prpasswd
300633	iw-pw
300634	starrb
300635	Impress_Server
300636	colorstar
300637	gwugui
300638	gwsgui
300639	dai_command_proxy
300640	dai_alarm_server
300641	dai_fui_proxy
300642	spai_command_proxy
300643	spai_alarm_server
300644	iris
300645	hcxttp
300646	updatedb rsched
300647	urnd urn
300648	iqwpsrv
300649	dskutild
300650	online
300651	nlserv
300652	acsm
300653	dg_clar_sormsg
300654	wwpollerrpc
300655	wwmodelrpc
300656	nsprofd
300657	nsdistd
300658	recollect
300659	lssexecd lss_res
300660	lssagend lss_rea
300661	cdinfo
300662	sninsr_addon
300663	mm-sap
300664	ks
300665	psched



300666	tekdvfs
300667	storxll
300668	nisse
300669	lbadvise
300670	atcinstaller
300671	atntstarter
300672	NetML
300673	tdmesmge
300674	tdmesmgd
300675	tdmesmgt
300676	olm
300677	mediamanagement
300678	rdbprog fieldowsrv
300679	rpwdprog rpwd
300680	sapi-trace
300681	sapi-master-daemon
300682	omdcuprog om-dcu
300683	wwprocmon
300684	tndidprog
300685	rkey_setsecretprog
300686	asdu_server_prog
300687	pwrctrl
300688	siunixd
300689	wmapi
300690	cross_reference_ole
300691	rtc
300692	disp
300693	sql_compilation_agent
300694	tnsysprog
300695	ius-sapimd
300696	apteam-dx
300697	rmsrpc
300698	seismic_system
300699	remote
300700	tt1_ts_event nokia_nms
300701	fxrs
300702	onlicense
300703	vxkey
300704	dinis
300705	sched2d schedule-2
300706	sched3d schedule-3
300707	sched4d schedule-4
300708	sched5d schedule-5
300709	sched6d schedule-6
300710	sched7d schedule-7
300711	sched8d schedule-8
300712	sched9d schedule-9
300713	adtsqry



300714	adserv
300715	adrepser
300716	[unknown]
300717	caad
300718	caui
300719	cescda
300720	vcapiadmin
300721	vcapi20
300722	tcfs
300723	csed
300724	nothand
300725	hacb
300726	nfauth
300727	imlm
300728	bestcomm
300729	lprpasswd
300730	rprpasswd
300731	proplistd
300732	mikomomc
300733	arepa-cas
300734	[unknown]
300735	[unknown]
300736	ando_ts
300737	intermezzo
300738	ftel-sdh-request
300739	ftel-sdh-response
300740	[unknown]
300741	[unknown]
300742	[unknown]
300743	[unknown]
300744	[unknown]
300745	vrc_abb
300746	vrc_comau
300747	vrc_fanuc
300748	vrc_kuka
300749	vrc_reis
300750	hp_sv6d
300751	correntmgr01
300752	correntike
300753	[unknown]
300754	[unknown]
300755	intransa_location
300756	intransa_management
300757	intransa_federation
300758	portprot
300759	ipmiprot
300760	aceapi
300761	f6000pss





	300762	vsmapi_program
	300763	ubertuple
	300764	ctconcrpcif
	300765	mfuadmin
	300766	aiols
	300767	dsmrootd
	300768	htdl
	300769	caba
	300770	vrc_cosimir
	300771	cmhelmd
	300772	polynsm
	300773	[unknown]
	300774	[unknown]
	300775	[unknown]
	300776	[unknown]
	300777	[unknown]
	300778	[unknown]
	300779	[unknown]
	300780	[unknown]
	300781	dsmrecalld
	300782	[unknown]
	300783	[unknown]
	300784	twrgcontrol
	300785	twrled
	300786	twrcfgdb
BMC software	300787-300886	
<a href="#">[available]</a>	300887 - 300999	
Sun Microsystems	301000-302000 [ 2000 numbers ]	
<a href="#">[available]</a>	302001-349999	
American Airlines	350000 - 350999	
Acucobol Inc.	351000 - 351099	
The Bristol Group	351100 - 351249	
Amteva Technologies	351250 - 351349	
	351350	wfmMgmtApp
	351351	wfmMgmtDataSrv
	351352	wfmMgmtFut1
	351353	wfmMgmtFut1
	351354	wfmAPM
	351355	wfmIAMgr
	351356	wfmECMgr
	351357	wfmLookOut
	351358	wfmAgentFut1
	351359	wfmAgentFut2
<a href="#">[available]</a>	351360 - 351406	
Sterling Software ITD	351407	csed
	351360	sched10d
	351361	sched11d
	351362	sched12d



351363	sched13d
351364	sched14d
351365	sched15d
351366	sched16d
351367	sched17d
351368	sched18d
351369	sched19d
351370	sched20d
351371	sched21d
351372	sched22d
351373	sched23d
351374	sched24d
351375	sched25d
351376	sched26d
351377	sched27d
351378	sched28d
351379	sched29d
351380	sched30d
351381	sched31d
351382	sched32d
351383	sched33d
351384	sched34d
351385	sched35d
351386	sched36d
351387	sched37d
351388	sched38d
351389	sched39d
351390	consoleserver
351391	scheduleserver
351392	RDELIVER
351393	REVENTPROG
351394	RSENDEVENTPROG
351395	snapp
351396	snapad
351397	sdsoodb
351398	sdsmain
351399	sdssrv
351400	sdscInt
351401	sdsreg
351402	fsbatch
351403	fsmonitor
351404	fsdisp
351405	fsession
351406	fslog
351407	svdpappserv
351408	gns
351409	[unkonwn]
351410	[unkonwn]



351411	[unkonwn]
351412	axi
351413	rpcxfr
351414	slm
351415	smbpasswd
351416	tbdbserv
351417	tbprojserv
351418	genericserver
351419	dynarc_ds
351420	dnscmdr
351421	ipcmdr
351422	faild
351423	failmon
351424	faildebug
351425	[unknown]
351426	[unknown]
351427	siemens_srs
351428	bsproxy
351429	ifsrpc
351430	CesPvcSm
351431	FrPvcSm
351432	AtmPvcSm
351433	radius
351434	auditor
351435	sft
351436	voicemail
351437	kis
351438	SOFTSERV_NOTIFY
351439	dynarpc
351440	hc
351441	iopas
351442	iopcs
351443	iopss
351444	spcnfs
351445	spcvss
351446	matilda_sms
351447	matilda_brs
351448	matilda_dbs
351449	matilda_sps
351450	matilda_svs
351451	matilda_sds
351452	matilda_vvs
351453	matilda_stats
351454	xtrade
351455	mapsvr
351456	hp_graphicsd
351457	berkeley_db
	berkeley_db_svc



	351458	io_server
	351459	rpc.niod
	351460	rpc.kill
	351461	hmdisproxy
	351462	smdisproxy
	351463	avatard
	351464	namu
	351465	BMCsess
	351466	FENS_Sport
	351467	EM_CONFIG
	351468	EM_CONFIG_RESP
	351469	lodge_proof
	351470	ARCserveIT-Queue
	351471	ARCserveIT-Device
	351472	ARCserveIT-Discover
	351473	ARCserveIT-Alert
	351474	ARCserveIT-Database
	351475	scand1
	351476	scand2
	351477	scand3
	351478	scand4
	351479	scand5
	351480	dscv
	351481	cb_svc
	351482	[unknown]
	351483	iprobe
	351484	omniconf
	351485	isan
BG Partners	351486 - 351500	
	351501	mond
	351502	iqlremote
	351503	iqlalarm
<a href="#">[available]</a>	351504 - 351599	
Orion Multisystems	351600-351855	
<a href="#">[available]</a>	351856 - 351899	
NSP lab	351900 - 351999	
<a href="#">[available]</a>	351999 - 352232	
	352233	asautostart
	352234	asmediad1
	352235	asmediad2
	352236	asmediad3
	352237	asmediad4
	352238	asmediad5
	352239	asmediad6
	352240	asmediad7
	352241	asmediad8
	352242	asmediad9
	352243	asmediad10





352244	asmediad11
352245	asmediad12
352246	asmediad13
352247	asmediad14
352248	asmediad15
352249	asmediad16
352250	waruser
352251	warlogd
352252	warsvrmgr
352253	warvfsysd
352254	warftpd
352255	warnfsd
352256	bofproxyc0
352257	bofproxys0
352258	bofproxyc1
352259	bofproxys1
352260	bofproxyc2
352261	bofproxys2
352262	bofproxyc3
352263	bofproxys3
352264	bofproxyc4
352265	bofproxys4
352266	bofproxyc5
352267	bofproxys5
352268	bofproxyc6
352269	bofproxys6
352270	bofproxyc7
352271	bofproxys7
352272	bofproxyc8
352273	bofproxys8
352274	bofproxyc9
352275	bofproxys9
352276	bofproxyc10
352277	bofproxys10
352278	bofproxyc11
352279	bofproxys11
352280	bofproxyc12
352281	bofproxys12
352282	bofproxyc13
352283	bofproxys13
352284	bofproxyc14
352285	bofproxys14
352286	bofproxyc15
352287	bofproxys15
352288	bofproxyc16
352289	bofproxys16
352290	bofproxyc17
352291	bofproxys17



[[available](#)]

352292 bofproxyp04

352293-370000

370001 [unknown]

370002 [unknown]

370003 [unknown]

370004 [unknown]

370005 [unknown]

370006 [unknown]

370007 [unknown]

370008 [unknown]

370009 [unknown]

370010 [unknown]

370011 [unknown]

370012 [unknown]

370013 [unknown]

370014 [unknown]

370015 [unknown]

370016 [unknown]

370017 [unknown]

370018 [unknown]

370019 [unknown]

370020 [unknown]

370021 [unknown]

370022 [unknown]

370023 [unknown]

370024 [unknown]

370025 [unknown]

370026 [unknown]

370027 [unknown]

[[available](#)]

370028 - 379999

380000 opensna

380001 probenet

380002 [unknown]

380003 license

380004 na.3com-remote

380005 na.ntp

380006 probeutil

380007 na.vlb

380008 cds\_mhs\_agent

380009 cds\_x500\_agent

380010 cds\_mailhub\_agent

380011 codex\_6500\_proxy

380012 codex\_6500\_trapd

380013 na.nm212

380014 cds\_mta\_metrics\_agent

380015 [unknown]

380016 na.cable

380017 codexcabletrap



Swiss Re	380018-380028
	380029 ncstat
	380030 ncnfsstat
	380031 ftams
	380032 na.isotp
	380033 na. <a href="#">rfc1006</a>
[ <a href="#">available</a> ]	380034 - 389999
Epoch Systems	390000 - 390049
Quickturn Systems	390050 - 390065
Team One Systems	390066 - 390075
General Electric CRD	390076 - 390085
TSIG NFS subcommittee	390086 - 390089
SoftLab ab	390090 - 390099
Legato Network Services	390100 - 390115
	390116 cdsmonitor
	390117 cdslock
	390118 cdslicense
	390119 shm
	390120 rws
	390121 cdc
Data General	390122 - 390141
Perfect Byte	390142 - 390171
JTS Computer Systems	390172 - 390181
Parametric Technology	390182 - 390191
Voxem	390192 - 390199
Effix Systems	390200 - 390299
Motorola	390300 - 390309
Mobile Data Intl.	390310 - 390325
Physikalisches Institut	390326 - 390330
Ergon Informatik AG	390331 - 390340
Analog Devices Inc.	390341 - 390348
Interphase Corporation	390349 - 390358
NeWsware	390359 - 390374
Qualix Group	390375 - 390379
Xerox Imaging Systems	390380 - 390389
Noble Net	390390 - 390399
Legato Network Services	390400 - 390499
Client Server Tech.	390500 - 390511
Atria	390512 - 390517
GE NMR Instruments	390518 - 390525
Harris Corp.	390526 - 390530
Unisys	390531 - 390562
Aggregate Computing	390563 - 390572
Interactive Data	390573 - 390580
OKG AB	390581 - 390589
K2 Software	390591 - 390594
Collier Jackson	390595 - 390599
Remedy Corporation	390600 - 390699



Mentor Graphics	390700 - 390799
AT&T Bell Labs (Lucent)	390800 - 390899
Xerox	390900 - 390999
Silicon Graphics	391000 - 391063
Data General	391064 - 391095
Computer Support Corp.	391096 - 391099
Quorum Software Systems	391100 - 391199
InterLinear Technology	391200 - 391209
Highland Software	391210 - 391229
Boeing Comp. Svcs.	391230 - 391249
IBM Sweden	391250 - 391259
Signature Authority Svc	391260 - 391271
ZUMTOBEL Licht GmbH	391272 - 391283
NOAA/ERL	391284 - 391299
NCR Corp.	391300 - 391399
FTP Software	391400 - 391409
Cadre Technologies	391410 - 391433
Visionware Ltd (UK)	391434 - 391439
IBR-Partner AG	391440 - 391449
CAP Programator AB	391450 - 391459
Reichle+De-Massari AG	391460 - 391474
Swiss Bank Corp (London)	391475 - 391484
Unisys Enterprise Svr	391485 - 391489
Intel - Test Dev. Tech.	391490 - 391499
Ampex	391500 - 391755
	391756 naas-spare
	391757 naas-admin
	391758 isps
	391759 isps-admin
	391760 mars
	391761 mars-admin
	391762 attcis_spare0
	391763 attcis_spare1
	391764 mail-server
	391765 mail-server-spare
	391766 attcis_spare2
	391767 attcis_spare3
	391768 attcis_spare4
	391769 attcis_spare5
	391770 attcis_spare6
	391771 attcis_spare7
Integrated Systems, Inc.	391772 - 391779
Parametric Tech., Inc.	391780 - 391789
Ericsson Telecom AB	391790 - 391799
SLAC	391800 - 391849
	391850 qhrdata
	391851 qhrbackup
	391852 minutedata





	391853	prefecture
	391854	supc
	391855	suadmincrw
	391856	suadminotas
	391857	sumessage
	391858	sublock
	391859	sumotd
staffware dev. (uk)	391860 - 391869	
Staffware Dev. (UK)	391870 - 391879	
	391880	namesrvr
	391881	disksrvr
	391882	tapesrvr
	391883	migsrvr
	391884	pdmsrvr
	391885	pvrsvr
	391886	repacksrvr
	391887	[unknown]
Convex Computer Corp.	391888 - 391951	
	391952	lookoutsrv
	391953	lookoutagnt
	391954	lookoutprxy
	391955	lookoutsntp
	391956	lookoutrmon
	391957	lookoutfut1
	391958	lookoutfut2
windward	391959 - 391967	
	391968	sra_legato
	391969	sra_legato_imgsvr
	391970	sra_legato_0
	391971	sra_legato_1
	391972	sra_legato_2
	391973	sra_legato_3
	391974	sra_legato_4
	391975	sra_legato_5
	391976	sra_legato_6
	391977	sra_legato_7
	391978	sra_legato_8
	391979	sra_legato_9
Brooktree Corp.	391980 - 391989	
Cadence Design Systems	391990 - 391999	
J. Frank & Associates	392000 - 392999	
Cooperative Solutions	393000 - 393999	
Xerox Corp.	394000 - 395023	
	395024	odbc_sqlretriever
3M	395025 - 395091	
Digital Zone Intl.	395092 - 395099	
Software Professionals	395100 - 395159	
Del Mar Solutions	395160 - 395164	



	395165	ife-es
	395166	ife-resmgr
	395167	ife-aes
	395168	ife-bite
	395169	ife-loader
	395170	ife-satcom
	395171	ife-seat
	395172	ife-dbmgr
	395173	ife-testmgr
	395174	atrium_server
	395175	ase_director
	395176	ase_agent
	395177	ase_hsm
	395178	ase_mgr
	395179	ase_sim
Hewlett-Packard	395180 - 395194	
XES, Inc.	395195 - 395199	
Unitech Products	395200 - 395249	
TransSys	395250 - 395505	
Unisys Govt Systems	395506 - 395519	
Bellcore	395520 - 395529	
IBM	395530 - 395561	
AT&T Network Services	395562 - 395571	
Data General	395572 - 395577	
Swiss Bank Corp	395578 - 395597	
Swiss Bank Corp	395598 - 395637	
Novell	395638 - 395643	
Computer Associates	395644 - 395650	
Omneon Video Networks	395651 - 395656	
<a href="#">[available]</a>	395657 - 395908	
UK Post Office	395909 - 395924	
AEROSPATIALE	395925 - 395944	
Result d.o.o.	395945 - 395964	
DataTools, Inc.	395965 - 395980	
CADIS, Inc.	395981 - 395990	
Cummings Group, Inc.	395991 - 395994	
Cadre Technologies	395995 - 395999	
American Airlines	396000 - 396999	
Ericsson Telecom TM Div	397000 - 398023	
IBM	398024 - 398028	
Toshiba OME Works	398029 - 398033	
TUSC Computer Systems	398034 - 398289	
AT&T	398290 - 398320	
Ontario Hydro	398321 - 398346	
Micrion Corporation	398347 - 398364	
<a href="#">[available]</a>	398365 - 398591	
Pegasystems, Inc.	398592 - 399616	
Spectra Securities Soft	399617 - 399850	



QualCom	399851 - 399866	
[ <a href="#">available</a> ]	399867 - 399884	
Altris Software Ltd.	399885 - 399899	
ISO/IEC WG11	399900 - 399919	
Parametric Technology	399920 - 399949	
Dolby Laboratories	399950 - 399981	
[ <a href="#">available</a> ]	399982 - 399991	
Xerox PARC	399992 - 399999	
#		
Next Inc.	200100000 - 200199999	
Netwise (RPCtool)	200200000	
Concurrent Computer Corp	200200001 - 200200007	
AIM Technology	200300000 - 200399999	
TGV	200400000 - 200499999	
#		
# Sun-assigned authentication flavor numbers		
#		
AUTH_NONE	0	/* no authentication, see <a href="#">RFC 1831</a> */
		/* a.k.a. AUTH_NULL */
AUTH_SYS	1	/* unix style (uid+gids), <a href="#">RFC 1831</a> */
		/* a.k.a. AUTH_UNIX */
AUTH_SHORT	2	/* short hand unix style, <a href="#">RFC 1831</a> */
AUTH_DH	3	/* des style (encrypted timestamp) */
		/* a.k.a. AUTH_DES, see <a href="#">RFC 2695</a> */
AUTH_KERB	4	/* kerberos auth, see <a href="#">RFC 2695</a> */
AUTH_RSA	5	/* RSA authentication */
RPCSEC_GSS	6	/* GSS-based RPC security for auth, integrity and privacy, RPC 5403 */
AUTH_NW	30001	NETWARE
AUTH_SEC	200000	TSIG NFS subcommittee
AUTH_ESV	200004	SVr4 ES
AUTH_NQNFS	300000	Univ. of Guelph - Not Quite NFS
AUTH_GSSAPI	300001	OpenVision <john.linn@ov.com>
AUTH_ILU_UGEN	300002	Xerox <janssen@parc.xerox.com> - ILU Unsecured Generic Identity
#		
# Small blocks are assigned out of the 39xxxx series of numbers		
#		
AUTH_SPNEGO	390000	
	390000 - 390255	NFS 'pseudo' flavors for RPCSEC_GSS
	390003	kerberos_v5 authentication, <a href="#">RFC 2623</a>
	390004	kerberos_v5 with data integrity, <a href="#">RFC 2623</a>
	390005	kerberos_v5 with data privacy, <a href="#">RFC 2623</a>
	200000000	reserved
	200100000	NeXT Inc.



## **18. Normative References**

[RFC4506]

Eisler, M., "XDR: External Data Representation Standard", [RFC 4506](#), May 2006

[RFC2203]

Eisler, M., Chiu, A., Ling, L., "RPCSEC\_GSS Protocol Specification", [RFC 2203](#), September 1997

## **19. Informative References**

[XRPC]

Birrell, A. D. & Nelson, B. J., "Implementing Remote Procedure Calls", XEROX CSL-83-7, October 1983.

[VMTP]

Cheriton, D., "VMTP: Versatile Message Transaction Protocol", Preliminary Version 0.3, Stanford University, January 1987.

[DH]

Diffie & Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory IT-22, November 1976.

[RFC768]

Postel, J., "User Datagram Protocol", STD 6, [RFC 768](#), USC/Information Sciences Institute, August 1980.

[RFC793]

Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", STD 7, [RFC 793](#), USC/Information Sciences Institute, September 1981.

[RFC1094]

Sun Microsystems, Inc., "NFS: Network File System Protocol Specification", [RFC 1094](#), March 1989.

[RFC1813]





Callaghan, B., Pawlowski, B., Staubach, P., "NFS Version 3 Protocol Specification", [RFC 1813](#), June 1995.

[RFC1831]

R. Srinivasan, "RPC: Remote Procedure Call Protocol Specification Version 2", [RFC 1831](#), August 1995.

[RFC1833]

R. Srinivasan, "Binding Protocols for ONC RPC Version 2", [RFC 1833](#), August 1995.

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [RFC 2119](#), March 1997

[RFC2623]

Eisler, M., "NFS Version 2 and Version 3 Security Issues and the NFS Protocol's Use of RPCSEC\_GSS and Kerberos V5", [RFC 2623](#), June 1999.

[RFC2695]

Chiu, A., "Authentication Mechanisms for ONC RPC", [RFC 2695](#), September 1999.

[RFC2743]

Linn. J., "Generic Security Service Application Program Interface Version 2, Update 1", [RFC 2743](#), January 2000.

[RFC3530]

Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., Noveck, D., "Network File System (NFS) version 4 Protocol", [RFC 3530](#), April 2003.

[RFC5226]

Narten, T. and Alvestrand, H., "Guidelines for Writing an IANA Considerations Section in RFCs", [RFC 5226](#), May 2008.



## **20. Author's Address**

Address comments related to this memorandum to:

`nfsv4@ietf.org`

Robert Thurlow  
Sun Microsystems, Inc.  
500 Eldorado Boulevard, UBRM05-171  
Broomfield, CO 80021

Phone: 877-718-3419  
E-mail: `robert.thurlow@sun.com`