types.h

```
Typedef unsigned char uchar_t;
typedef unsigned short ushort_t;
typedef unsigned int uint_t;
typedef unsigned long ulong_t;
typedef char *  caddr_t;
typedef long long  longlong_t;
typedef unsigned long long u_longlong_t;
typedef long uid_t;
typedef uid_t gid_t;
typedef long pid_t;

typedef unsigned char u_char;
typedef unsigned short u_short;
typedef unsigned int u_int;
typedef unsigned long u_long;
```

in.h

```
struct sockaddr_in {
 short sin_family;
 u_short sin_port;
 struct in_addr sin_addr;
 char sin_zero[8];
};
```

netdb.h

```
struct hostent {
 char *h_name;
 char **h_aliases;
 int h_addrtype;
 int h_length;
 char **h_addr_list;
#define h_addr h_addr_list[0]
};

struct hostent *gethostbyname(const char *);
struct hostent *gethostbyaddr(const char *,
int, int);
```

time.h

```
struct timeval {
 long tv_sec;  /* seconds */
 long tv_usec; /* and microseconds */
};
int gettimeofday(struct timeval *);
int settimeofday(struct timeval *);
```

select.h

```
typedef long fd_mask;
typedef struct fd_set {
 fd_mask fds_bits[howmany(FD_SETSIZE,
NFDBITS)];
} fd_set;

#define FD_SET(n, p) ((p)-> fds_bits
[(n)/NFDBITS] |= (unsigned)1 << ((n) %
NFDBITS)))

#define FD_CLR(n, p) ((p)-> fds_bits
[(n)/NFDBITS] &= ~((unsigned)1 << ((n) %
NFDBITS)))

#define FD_ISSET(n, p) ((p)->
fds_bits[(n)/NFDBITS] & ((unsigned)1 << ((n)
% NFDBITS)))

#define FD_ZERO(p) bzero((char *)(p), sizeof
(*(p)))

extern int select(int, fd_set *, fd_set *,
fd_set *, struct timeval *);
```

socket.h

```
#define AF_UNSPEC 0
#define AF_UNIX  1
#define AF_INET  2

struct sockaddr {
 u_short sa_family;
 char sa_data[14];
 };

#define PF_UNSPEC AF_UNSPEC
#define PF_UNIX  AF_UNIX
#define PF_INET  AF_INET

#define SOMAXCONN 5

extern int accept(int, struct sockaddr *, int
*);
extern int bind(int, struct sockaddr *, int);
extern int connect(int, struct sockaddr *,
int);
extern int listen(int, int);
extern int recvfrom(int, char *, int, int,
struct sockaddr *, int *);
extern int sendto(int, const char *, int,
int, const struct sockaddr *, int);
extern int socket(int, int, int);
extern int shutdown(int, int);
```

xdr.h

```
enum xdr_op {
 XDR_ENCODE = 0,
 XDR_DECODE = 1,
 XDR_FREE = 2
};

typedef bool_t (*xdrproc_t)();
extern bool_t xdr_void(void);
extern bool_t xdr_int(XDR *, int *);
...

extern void   xdrmem_create(XDR *, const
caddr_t, const u_int, const enum
xdr_op);
extern void   xdrstdio_create(XDR *, FILE *,
const enum xdr_op);
```

clnt.h

```
enum clnt_stat {
 RPC_SUCCESS = 0,
 RPC_CANTENCODEARGS = 1,
 RPC_CANTDECODERES = 2,
 RPC_CANTSEND = 3,
 RPC_CANTRECV = 4,
 RPC_TIMEDOUT = 5,
 RPC_VERSMISMATCH = 6,
        ...

typedef struct __client {
 AUTH *cl_auth;
 struct clnt_ops {
        ...
} CLIENT;

/*
 * bool_t
 * clnt_control(cl, request, info)
 * CLIENT *cl;
 * u_int request;
 * char *info;
 */
#define CLSET_TIMEOUT  1
#define CLGET_TIMEOUT  2
#define CLGET_SERVER_ADDR 3
#define CLSET_RETRY_TIMEOUT 4
#define CLGET_RETRY_TIMEOUT 5
```

```
extern  CLIENT * clnt_create(const char *,
const u_long, const u_long, const
char *);

void clnt_pcreateerror(const char *); /*
stderr */
void clnt_perror(const CLIENT *, const char
*);
extern enum clnt_stat callrpc(const char *,
const u_long, const u_long, const u_long,
const xdrproc_t, const char *, const
xdrproc_t, char *);

extern enum clnt_stat clnt_broadcast(const
u_long, const u_long, const u_long, const
xdrproc_t, caddr_t, const xdrproc_t, caddr_t,
const resultproc_t);
```

svc.h

```
struct svc_req {
 u_long  rq_prog;
 u_long  rq_vers;
 u_long  rq_proc;
 struct opaque_auth rq_cred;
 caddr_t  rq_clntcred;
 struct __svcxprt *rq_xprt;
};

registerrpc(u_long, u_long, u_long, char
*(*)(char *), xdrproc_t, xdrproc_t);

extern bool_t rpc_reg(const u_long,   const
u_long, const u_long, char *(*)(char *),
const xdrproc_t, const xdrproc_t, const char
*);

extern bool_t svc_sendreply(const SVCXPRT *,
const xdrproc_t, const caddr_t);
extern void svcerr_decode(const SVCXPRT *);
extern void svcerr_weakauth(const SVCXPRT *);
extern void svcerr_noproc(const SVCXPRT *);
extern void svcerr_progvers(const SVCXPRT *,
const u_long, const u_long);
extern void svcerr_auth(const SVCXPRT *,
const enum auth_stat);
extern void svcerr_noprog(const SVCXPRT *);
extern void svcerr_systemerr(const SVCXPRT
*);
extern void svc_run(void);
```

auth.h

```
enum auth_stat {
 AUTH_OK = 0,
 AUTH_BADCRED = 1,
 AUTH_REJECTEDCRED = 2,
 AUTH_BADVERF = 3,
 AUTH_REJECTEDVERF = 4,
 AUTH_TOOWEAK = 5,
       ....
};
struct opaque_auth {
 enum_t oa_flavor;
 caddr_t oa_base;
 u_int oa_length;
};

typedef struct __auth {
 struct opaque_auth ah_cred;
 struct opaque_auth ah_verf;
 union des_block ah_key;
 struct auth_ops {
       ....
} AUTH;

extern AUTH *authunix_create(const char *,
const uid_t, const gid_t, const int,
const gid_t *);
extern AUTH *authunix_create_default(void);
extern AUTH *authnone_create(void);

struct authunix_parms {
 u_long  aup_time;
 char *aup_machname;
 uid_t  aup_uid;
 gid_t  aup_gid;
 u_int  aup_len;
 gid_t *aup_gids;
};
```

pmap_prot.h

```
#define PMAPPORT 111

struct pmap {
 u_long pm_prog;
 u_long pm_vers;
 u_long pm_prot;
 u_long pm_port;
};
typedef struct pmap pmap;
typedef pmap PMAP;


#define PMAP_IPPROTO_TCP 6

#define PMAP_IPPROTO_UDP 17

struct pmaplist {
 PMAP pml_map;
 struct pmaplist *pml_next;
};
typedef struct pmaplist pmaplist;
typedef struct pmaplist PMAPLIST;
```

pmap_clnt.h

```
extern bool_t pmap_set(unsigned long,
unsigned long, int, unsigned short port);
extern bool_t pmap_unset(u_long, u_long);
extern struct pmaplist *pmap_getmaps(struct
sockaddr_in *);
```

dce/rpc.h

```
extern void rpc_server_register_if(
    /* [in] */ rpc_if_handle_t if_spec,
    /* [in] */ uuid_p_t mgr_type_uuid,
    /* [in] */ rpc_mgr_epv_t mgr_epv,
    /* [out] */ unsigned32 *status
);
extern void rpc_server_use_protseq(
    /* [in] */ unsigned_char_p_t protseq,
    /* [in] */ unsigned32 max_call_requests,
    /* [out] */ unsigned32 *status
);
extern void rpc_server_use_all_protseqs(
    /* [in] */ unsigned32 max_call_requests,
    /* [out] */ unsigned32 *status
);
extern void rpc_server_inq_bindings(
    /* [out] */ rpc_binding_vector_p_t *vector,
    /* [out] */ unsigned32 *status
);
extern void rpc_ep_register(
    /* [in] */ rpc_if_handle_t if_spec,
    /* [in] */ rpc_binding_vector_p_t vector,
    /* [in] */ uuid_vector_p_t object_uuid_vec,
    /* [in] */ unsigned_char_p_t annotation,
    /* [out] */ unsigned32 *status
);
extern void rpc_server_listen(
    /* [in] */ unsigned32 max_calls_exec,
    /* [out] */ unsigned32 *status
);
extern void rpc_mgmt_stop_server_listening(
    /* [in] */ rpc_binding_handle_t handle,
    /* [out] */ unsigned32 *status
);
extern void rpc_mgmt_ep_unregister(
    /* [in] */ rpc_binding_handle_t ep_binding,
    /* [in] */ rpc_if_id_p_t if_id,
    /* [in] */ rpc_binding_handle_t binding,
    /* [in] */ uuid_p_t object_uuid,
    /* [out] */ unsigned32 *status
);
extern void rpc_server_unregister_if(
    /* [in] */ rpc_if_handle_t if_spec,
    /* [in] */ uuid_p_t mgr_type_uuid,
    /* [out] */ unsigned32 *status
);
```

```
extern void rpc_string_binding_compose(
    /* [in] */ unsigned_char_p_t object_uuid,
    /* [in] */ unsigned_char_p_t protseq,
    /* [in] */ unsigned_char_p_t netaddr,
    /* [in] */ unsigned_char_p_t endpoint,
    /* [in] */ unsigned_char_p_t options,
    /* [out] */ unsigned_char_p_t *str_binding,
    /* [out] */ unsigned32 *status
);
extern void rpc_string_free(
    /* [in, out] */ unsigned_char_p_t *string,
    /* [out] */ unsigned32 *status
);
extern void rpc_binding_from_string_binding(
    /* [in] */ unsigned_char_p_t string_binding,
    /* [out] */ rpc_binding_handle_t *handle,
    /* [out] */ unsigned32 *status
);
extern void rpc_binding_to_string_binding(
    /* [in] */ rpc_binding_handle_t handle,
    /* [out] */ unsigned_char_p_t *str_binding,
    /* [out] */ unsigned32 *status
);
extern void rpc_ep_resolve_binding(
    /* [in] */ rpc_binding_handle_t binding_h,
    /* [in] */ rpc_if_handle_t if_spec,
    /* [out] */ unsigned32 *status
);
```

dce/rpcbase.h

```
typedef struct  {
  unsigned32 count;
  rpc_binding_handle_t binding_h[1];
} rpc_binding_vector_t;
typedef rpc_binding_vector_t *rpc_binding_vector_p_t;

typedef unsigned char idl_char ;
typedef idl_char unsigned_char_t;
typedef idl_char *unsigned_char_p_t;
```