BIND Porting Assumptions

for the upgrade of libresolv2 to ISC libbind (9)6.0

For **Sun Microsystems**

By **Ed Posnak, Venev Inc.**
415.254.0086
www.venev.com

version 0.3
April 20, 2009

Introduction

A goal of the resolver library upgrade is to reduce the cost and improve the reliability of future upgrades by making the libresolv2 source code as close as possible to ISC's libbind source code. To accomplish this goal we plan to change the current structure and API of libresolv2. The following provides a concise overview of the assumptions we are operating under and the changes we plan to make as a result.

Assumptions

1. It is OK to update the public header files <netdb.h>, <resolv.h>, <arpa/nameser.h>, <arpa/nameser_compat.h>, and <arpa/inet.h> in order to minimize the number of workarounds that would otherwise be necessary to update the resolver library.
2. It is OK to add to the API of the resolver library as long as existing client applications will continue to run correctly with the updated library. That is, it's not OK to break backwards source code or binary compatibility.
3. It is OK to incorporate workarounds in the updated library where necessary to avoid breaking existing client applications (as defined in 2. above).
4. Nothing in Solaris or any client applications depend on the obsolete cylink and dnssafe libraries. They can and should be removed.
5. It is OK to package the irs code that comes with libbind into libresolv.so.2, as long as (1) the functions that were previously renamed continue to be renamed the same way, and (2) deleted functions do not come back under the name that was deleted.

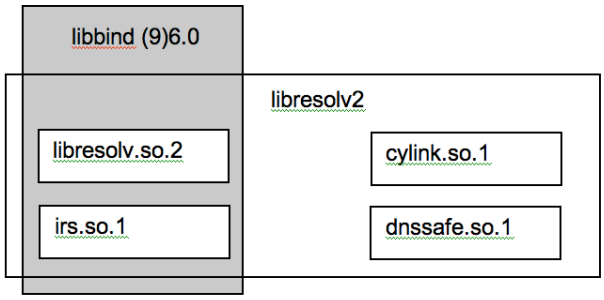
Changes

We will update the Solaris public header files, mentioned above, to incorporate changes made to the more recent ISC versions. We will incorporate as many changes as possible subject to the conditions of assumption 2.

We will create a new libresolv.so.2 such that existing client applications will continue to work without the need to be re-compiled.

We will get rid of the obsolete cylink and dnssafe cryptographic libraries.

We will consolidate all irs functionality, which was formerly split between libresolv.so.2 and irs.so.1, into libresolv.so.2, so that libresolv.so.2 resembles ISC libbind (9)6.0 as illustrated below.



For irs functions that were formerly renamed as res_<function name> we will keep the names the same as libresolv.so.2.

For irs functions that overlap with libsocket/libnsl and were deleted from libresolv.so.2 or moved to irs.so.1 we will rename them as irs_<function name>.

For irs functions in libresolv.so.2 that overlap with libsocket/libnsl but were not removed or renamed we will leave them as-is.

The idea is to use irs functionality exactly how libresolv.so.2 does, keeping any public interfaces the same.

Appendix A: Specific Changes

inet.h.patch

```
--- usr/src/head/arpa/inet.h  2009-04-15 15:59:51.000000000 -0700
+++ usr/src/head.bc/arpa/inet.h     2009-04-15 23:29:18.000000000 -0700
@@ -47,6 +47,16 @@
 #include <sys/byteorder.h>
 #endif /* defined(_XPG4_2) && !defined(__EXTENSIONS__) */

+#ifdef __STDC__
+#ifndef __P
+#define    __P(x)      x
+#endif
+#else
+#ifndef __P
+#define    __P(x)      ()
+#endif
+#endif /* __STDC__ */
+
 #ifdef     __cplusplus
 extern "C" {
 #endif
@@ -66,6 +76,7 @@
                   char *_RESTRICT_KYWD, socklen_t);
 #endif /* !defined(_XPG4_2) || defined(_XPG6) ||
defined(__EXTENSIONS__) */

+
 extern in_addr_t inet_addr(const char *);
 /*
  * With the introduction of CIDR the
@@ -73,11 +84,24 @@
  */
 extern in_addr_t inet_lnaof(struct in_addr);
 extern struct in_addr inet_makeaddr(in_addr_t, in_addr_t);
+
+extern char *           inet_neta __P((u_long, char *, size_t));
+
 extern in_addr_t inet_netof(struct in_addr);
 extern in_addr_t inet_network(const char *);

+extern char            *inet_net_ntop __P((int, const void *, int,
char *, size_t));
+extern int         inet_net_pton __P((int, const char *, void *,
size_t));
+extern char            *inet_cidr_ntop __P((int, const void *, int,
char *, size_t));
+extern int         inet_cidr_pton __P((int, const char *, void *, int
*));
 extern char *inet_ntoa(struct in_addr);
 extern int inet_aton(const char *, struct in_addr *);
+
+extern int         inet_pton __P((int, const char *, void *));
+extern const char      *inet_ntop __P((int, const void *, char *,
size_t));
```

```
+extern u_int            inet_nsap_addr __P((const char *, u_char *,
int));
+extern char            *inet_nsap_ntoa __P((int, const u_char *, char
*));
+
 #else
 unsigned long inet_addr();
 char *inet_ntoa();
@@ -86,13 +110,22 @@
  * following 4 routines are now considered to be Obsolete
  */
 struct     in_addr inet_makeaddr();
+char *            inet_neta();
 unsigned long inet_network();
+char       *inet_net_ntop();
+int         inet_net_pton();
+char       *inet_cidr_ntop();
+int         inet_cidr_pton();
 extern unsigned long inet_lnaof();
 extern unsigned long inet_netof();

 extern int inet_pton();
 extern const char *inet_ntop();
 extern int inet_aton();
+int         inet_pton();
+const char *inet_ntop();
+u_int             inet_nsap_addr();
+char       *inet_nsap_ntoa();
 #endif

 #ifdef     __cplusplus
```

nameser.h.patch

```
--- usr/src/head/arpa/nameser.h     2009-04-15 15:59:09.000000000 -0700
+++ usr/src/head.bc/arpa/nameser.h  2009-04-15 23:26:39.000000000 -0700
@@ -89,6 +89,9 @@
 #define    NS_MAXMSG   65535 /* maximum message size */
 #define    NS_MAXCDNAME      255   /* maximum compressed domain name
*/
 #define    NS_MAXLABEL 63    /* maximum length of domain label */
+#define NS_MAXLABELS   128   /* theoretical max #/labels per domain
name */
+#define NS_MAXNNAME    256   /* maximum uncompressed (binary) domain
name*/
+#define    NS_MAXPADDR (sizeof
"ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff")
 #define    NS_HFIXEDSZ 12    /* #/bytes of fixed data in header */
 #define    NS_QFIXEDSZ 4     /* #/bytes of fixed data in query */
 #define    NS_RRFIXEDSZ      10    /* #/bytes of fixed data in r
record */
@@ -100,6 +103,7 @@
 #define    NS_CMPRSFLGS      0xc0  /* Flag bits indicating name
compression. */
 #define    NS_DEFAULTPORT    53    /* For both TCP and UDP. */

+
 /*
```

```
  * These can be expanded with synonyms, just keep
ns_parse.c:ns_parserecord()
  * in synch with it.
@@ -116,6 +120,18 @@
 } ns_sect;

 /*
+ * Network name (compressed or not) type.  Equivilent to a pointer
when used
+ * in a function prototype.  Can be const'd.
+ */
+typedef u_char ns_nname[NS_MAXNNAME];
+typedef const u_char *ns_nname_ct;
+typedef u_char *ns_nname_t;
+
+struct ns_namemap { ns_nname_ct base; int len; };
+typedef struct ns_namemap *ns_namemap_t;
+typedef const struct ns_namemap *ns_namemap_ct;
+
+/*
  * This is a message handle.  It is caller allocated and has no
dynamic data.
  * This structure is intended to be opaque to all but ns_parse.c, thus
the
  * leading _'s on the member names.  Use the accessor functions, not
the _'s.
@@ -129,6 +145,17 @@
      const uchar_t     *_msg_ptr;
 } ns_msg;

+/*
+ * This is a newmsg handle, used when constructing new messages with
+ * ns_newmsg_init, et al.
+ */
+struct ns_newmsg {
+     ns_msg            msg;
+     const u_char      *dnptrs[25];
+     const u_char      **lastdnptr;
+};
+typedef struct ns_newmsg ns_newmsg;
+
 /* Private data structure - do not use from outside library. */
 struct _ns_flagdata {  int mask, shift;  };
 extern struct _ns_flagdata _ns_flagdata[];
@@ -152,8 +179,23 @@
      const uchar_t     *rdata;
 } ns_rr;

+/*
+ * Same thing, but using uncompressed network binary names, and real C
types.
+ */
+typedef    struct __ns_rr2 {
+     ns_nname    nname;
+     size_t            nnamel;
+     int         type;
+     int         rr_class;
+     u_int       ttl;
```

```
+     int         rdlength;
+     const u_char *    rdata;
+} ns_rr2;
+
 /* Accessor macros - this is part of the public interface. */
 #define    ns_rr_name(rr)    (((rr).name[0] != '\0') ? (rr).name :
".")
+#define ns_rr_nname(rr)      ((const ns_nname_t)(rr).nname)
+#define ns_rr_nnamel(rr) ((rr).nnamel + 0)
 #define    ns_rr_type(rr)    ((ns_type)((rr).type + 0))
 #define    ns_rr_class(rr)   ((ns_class)((rr).rr_class + 0))
 #define    ns_rr_ttl(rr)     ((rr).ttl + 0)
@@ -323,6 +365,17 @@
      ns_t_sink = 40,         /* Kitchen sink (experimentatl) */
      ns_t_opt = 41,          /* EDNS0 option (meta-RR) */
      ns_t_apl = 42,          /* Address prefix list (RFC 3123) */
+     ns_t_ds = 43,           /* Delegation Signer */
+     ns_t_sshfp = 44,  /* SSH Fingerprint */
+     ns_t_ipseckey = 45,     /* IPSEC Key */
+     ns_t_rrsig = 46,  /* RRset Signature */
+     ns_t_nsec = 47,         /* Negative security */
+     ns_t_dnskey = 48, /* DNS Key */
+     ns_t_dhcid = 49,  /* Dynamic host configuratin identifier */
+     ns_t_nsec3 = 50,  /* Negative security type 3 */
+     ns_t_nsec3param = 51,   /* Negative security type 3 parameters */
+     ns_t_hip = 55,          /* Host Identity Protocol */
+     ns_t_spf = 99,          /* Sender Policy Framework */
      ns_t_tkey = 249,  /* Transaction key */
      ns_t_tsig = 250,  /* Transaction signature. */
      ns_t_ixfr = 251,  /* Incremental zone transfer. */
@@ -331,6 +384,7 @@
      ns_t_maila = 254, /* Transfer mail agent records. */
      ns_t_any = 255,         /* Wildcard match. */
      ns_t_zxfr = 256,  /* BIND-specific, nonstandard. */
+     ns_t_dlv = 32769, /* DNSSEC look-aside validatation. */
      ns_t_max = 65536
 } ns_type;

@@ -463,6 +517,7 @@
  * EDNS0 extended flags, host order.
  */
 #define    NS_OPT_DNSSEC_OK  0x8000U
+#define NS_OPT_NSID             3

 /*
  * Inline versions of get/put short/long.  Pointer is advanced.
@@ -514,6 +569,7 @@
 #define    ns_initparse            __ns_initparse
 #define    ns_skiprr         __ns_skiprr
 #define    ns_parserr        __ns_parserr
+#define ns_parserr2          __ns_parserr2
 #define    ns_sprintrr       __ns_sprintrr
 #define    ns_sprintrrf            __ns_sprintrrf
 #define    ns_format_ttl           __ns_format_ttl
@@ -528,6 +584,11 @@
 #define    ns_name_uncompress      __ns_name_uncompress
 #define    ns_name_skip            __ns_name_skip
 #define    ns_name_rollback  __ns_name_rollback
```

```
+#define    ns_name_length          __ns_name_length
+#define    ns_name_eq        __ns_name_eq
+#define    ns_name_owned           __ns_name_owned
+#define    ns_name_map       __ns_name_map
+#define    ns_name_labels          __ns_name_labels
 #define    ns_sign                 __ns_sign
 #define    ns_sign2          __ns_sign2
 #define    ns_sign_tcp       __ns_sign_tcp
@@ -541,6 +602,17 @@
 #define    ns_subdomain            __ns_subdomain
 #define    ns_makecanon            __ns_makecanon
 #define    ns_samename       __ns_samename
+#define    ns_newmsg_init          __ns_newmsg_init
+#define    ns_newmsg_copy          __ns_newmsg_copy
+#define    ns_newmsg_id            __ns_newmsg_id
+#define    ns_newmsg_flag          __ns_newmsg_flag
+#define    ns_newmsg_q       __ns_newmsg_q
+#define    ns_newmsg_rr            __ns_newmsg_rr
+#define    ns_newmsg_done          __ns_newmsg_done
+#define    ns_rdata_unpack         __ns_rdata_unpack
+#define    ns_rdata_equal          __ns_rdata_equal
+#define    ns_rdata_refers         __ns_rdata_refers
+

 int        ns_msg_getflag(ns_msg, int);
 uint_t           ns_get16(const uchar_t *);
@@ -550,6 +622,7 @@
 int        ns_initparse(const uchar_t *, int, ns_msg *);
 int        ns_skiprr(const uchar_t *, const uchar_t *, ns_sect, int);
 int        ns_parserr(ns_msg *, ns_sect, int, ns_rr *);
+int        ns_parserr2 __P((ns_msg *, ns_sect, int, ns_rr2 *));
 int        ns_sprintrr(const ns_msg *, const ns_rr *,
     const char *, const char *, char *, size_t);
 int        ns_sprintrrf(const uchar_t *, size_t, const char *,
@@ -573,6 +646,11 @@
 int        ns_name_skip(const uchar_t **, const uchar_t *);
 void ns_name_rollback(const uchar_t *, const uchar_t **,
                const uchar_t **);
+ssize_t          ns_name_length(ns_nname_ct, size_t);
+int        ns_name_eq(ns_nname_ct, size_t, ns_nname_ct, size_t);
+int        ns_name_owned(ns_namemap_ct, int, ns_namemap_ct, int);
+int        ns_name_map(ns_nname_ct, size_t, ns_namemap_t, int);
+int        ns_name_labels(ns_nname_ct, size_t);
 int        ns_sign(uchar_t *, int *, int, int, void *,
                const uchar_t *, int, uchar_t *, int *, time_t);
 int        ns_sign2(uchar_t *, int *, int, int, void *,
@@ -596,6 +674,25 @@
 int        ns_subdomain(const char *, const char *);
 int        ns_makecanon(const char *, char *, size_t);
 int        ns_samename(const char *, const char *);
+int        ns_newmsg_init(u_char *buffer, size_t bufsiz, ns_newmsg *);
+int        ns_newmsg_copy(ns_newmsg *, ns_msg *);
+void       ns_newmsg_id(ns_newmsg *handle, u_int16_t id);
+void       ns_newmsg_flag(ns_newmsg *handle, ns_flag flag, u_int
value);
+int        ns_newmsg_q(ns_newmsg *handle, ns_nname_ct qname,
+                  ns_type qtype, ns_class qclass);
+int        ns_newmsg_rr(ns_newmsg *handle, ns_sect sect,
```

```
+                       ns_nname_ct name, ns_type type,
+                       ns_class rr_class, u_int32_t ttl,
+                       u_int16_t rdlen, const u_char *rdata);
+size_t          ns_newmsg_done(ns_newmsg *handle);
+ssize_t         ns_rdata_unpack(const u_char *, const u_char *,
ns_type,
+                        const u_char *, size_t, u_char *, size_t);
+int       ns_rdata_equal(ns_type,
+                         const u_char *, size_t,
+                         const u_char *, size_t);
+int       ns_rdata_refers(ns_type,
+                       const u_char *, size_t,
+                       const u_char *);

 #ifdef BIND_4_COMPAT
 #include <arpa/nameser_compat.h>
```

netdb.h.patch

```
--- usr/src/head/netdb.h      2009-04-15 15:55:37.000000000 -0700
+++ usr/src/head.bc/netdb.h   2009-04-19 20:53:23.000000000 -0700
@@ -102,6 +102,31 @@
 #define    h_addr      h_addr_list[0]    /* address, for backward
compatiblity */
 };

+/*%
+ * Assumption here is that a network number
+ * fits in an unsigned long -- probably a poor one.
+ */
+struct     netent {
+     char        *n_name;    /* official name of net */
+     char        **n_aliases;      /* alias list */
+     int         n_addrtype; /* net address type */
+     unsigned long     n_net;            /* network # */
+};
+
+struct     servent {
+     char  *s_name;    /* official service name */
+     char  **s_aliases;      /* alias list */
+     int   s_port;           /* port # */
+     char  *s_proto;   /* protocol to use */
+};
+
+struct     protoent {
+     char  *p_name;    /* official protocol name */
+     char  **p_aliases;      /* alias list */
+     int   p_proto;    /* protocol # */
+};
+
+

 /*
  * addrinfo introduced with IPv6 for Protocol-Independent Hostname
@@ -122,6 +147,20 @@
      struct sockaddr *ai_addr;     /* binary address */
      struct addrinfo *ai_next;     /* next structure in linked list */
 };
```

```
+
+/*%
+ * Error return codes from gethostbyname() and gethostbyaddr()
+ * (left in extern int h_errno).
+ */
+
+#define    NETDB_INTERNAL    -1    /* see errno */
+#define    NETDB_SUCCESS     0     /* no problem */
+#define    HOST_NOT_FOUND    1 /* Authoritative Answer Host not found
*/
+#define    TRY_AGAIN   2 /* Non-Authoritive Host not found, or
SERVERFAIL */
+#define    NO_RECOVERY 3 /* Non recoverable errors, FORMERR, REFUSED,
NOTIMP */
+#define    NO_DATA           4 /* Valid name, no data record of
requested type */
+#define    NO_ADDRESS  NO_DATA           /* no address, look for MX
record */
+
 /* addrinfo flags */
 #define    AI_PASSIVE  0x0008      /* intended for bind() + listen()
*/
 #define    AI_CANONNAME      0x0010      /* return canonical version
of host */
@@ -133,6 +172,7 @@
 #define    AI_ALL            0x0002      /* IPv6 and IPv4 mapped
addresses */
 #define    AI_ADDRCONFIG     0x0004      /* AAAA or A records only if
IPv6/IPv4 cnfg'd */

+
 /*
  * These were defined in RFC 2553 but not SUSv3
  * or RFC 3493 which obsoleted 2553.
@@ -154,6 +194,8 @@
 #define    EAI_SOCKTYPE      10    /* ai_socktype not supported */
 #define    EAI_SYSTEM  11    /* system error in errno */
 #define    EAI_OVERFLOW      12    /* argument buffer overflow */
+#define EAI_PROTOCOL   13
+#define EAI_MAX        14

 /* getnameinfo flags */
 #define    NI_NOFQDN   0x0001
@@ -173,6 +215,12 @@
 #endif /* !defined(_XPG6) || defined(__EXTENSIONS__) */
 #endif /* !defined(_XPG4_2) || defined(_XPG6) ||
defined(__EXTENSIONS__) */

+/*%
+ * Scope delimit character
+ */
+#define SCOPE_DELIMITER      '%'
+
+
 /*
  * Algorithm entry for /etc/inet/ipsecalgs which defines IPsec
protocols
  * and algorithms.
```

resolv.h.patch

```
--- usr/src/head/resolv.h     2009-04-15 15:56:34.000000000 -0700
+++ usr/src/head.bc/resolv.h  2009-04-19 21:21:42.000000000 -0700
@@ -219,10 +219,11 @@
      int         res_h_errno;      /* last one set for this context */
      int         _vcsock;    /* PRIVATE: for res_send VC i/o */
      uint_t      _flags;           /* PRIVATE: see below */
+     u_char      _rnd[16];         /* PRIVATE: random state */
      uint_t      _pad;             /* make _u 64 bit aligned */
      union {
            /* On an 32-bit arch this means 512b total. */
-           char  pad[72 - 4*sizeof (int) - 2*sizeof (void *)];
+           char  pad[56 - 4*sizeof (int) - 2*sizeof (void *)];
            struct {
                  uint16_t          nscount;
                  uint16_t          nstimes[MAXNS];   /* ms. */
@@ -365,6 +366,10 @@
 int         dn_skipname __P((const uchar_t *, const uchar_t *));
 void        putlong __P((unsigned int, uchar_t *));
 void        putshort __P((unsigned short, uchar_t *));
+#ifndef __ultrix__
+u_int16_t  _getshort __P((const u_char *));
+u_int32_t  _getlong __P((const u_char *));
+#endif
 const char *p_class __P((int));
 const char *p_time __P((unsigned int));
 const char *p_type __P((int));
@@ -383,7 +388,9 @@
                       uchar_t **, uchar_t **));
 int         dn_expand __P((const uchar_t *, const uchar_t *,
                       const uchar_t *, char *, int));
+void        res_rndinit __P((res_state));
 uint_t            res_randomid __P((void));
+u_int             res_nrandomid __P((res_state));
 int         res_nameinquery __P((const char *, int, int,
                       const uchar_t *, const uchar_t *));
 int         res_queriesmatch __P((const uchar_t *, const uchar_t *,
@@ -420,6 +427,8 @@
                       int));
 void        res_nclose __P((res_state));
 int         res_nopt __P((res_state, int, uchar_t *, int, int));
+int         res_nopt_rdata __P((res_state, int, u_char *, int, u_char
*,
+                        u_short, u_short, u_char *));
 void        res_send_setqhook __P((res_send_qhook hook));
 void        res_send_setrhook __P((res_send_rhook hook));
 int         __res_vinit __P((res_state, int));
```