A General Porting Strategy

to address interface incompatibilities between
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 14, 2009

Introduction

The BIND update project involves upgrading the OpenSolaris resolver library, libresolv2, to the latest version the ISC resolver library, which we'll refer to in this document as ISC libbind (9)6.0.

A goal of this port 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. A second goal is to maintain the existing interface to the resolver library.

A tradeoff between these goals is necessitated by the incompatibilities between the current libresolv2 interface with the more recent ISC libbind code. These incompatibilities arise from a few general problems:

- differences between the OpenSolaris public header files and ISC libbind headers
- the splitting of libbind into separate dynamic libraries on OpenSolaris
- overlap between ISC libbind functions and OpenSolaris' libnsl and libsocket

As a result, upgrading libresolv2 to a new ISC version while maintaining the existing interface involves re-applying a number of workarounds to the ISC code. These include:

- Changing the types of arguments and return values of functions to match older OpenSolaris public header files
- Writing adapters for data structures that convert from ISC structures to those defined in older OpenSolaris public header files
- Changing the names of or deleting functions that conflict with names in OpenSolaris' libnsl and libsocket
- Undoing ISC's ANSI C identifier hiding
- Writing code to load libraries and invoke functions to handle dependencies across the split of libraries created for OpenSolaris

The process of teasing out and re-applying these workarounds each time libresolv2 is upgraded to a newer version of ISC libbind is a human-intensive effort that increases the cost and the risk of introducing defects in every upgrade. Moreover, as ISC's libbind evolves independently, more workarounds need to be incorporated with each upgrade.

Updating the OpenSolaris public header files and consolidating libbind into a single dynamic library would eliminate virtually all of these workarounds, and would dramatically reduce the cost and improve reliability of future ports. However, these changes, taken in whole, would break the existing API/ABI.

Since it is not clear to what extent the headers can be updated, we will adopt a flexible approach that does not depend on changes that would break API/ABI. This document presents our strategy for updating libresolv2 to meet our dual goals of preserving the interface while minimizing changes to the ISC source code.

General Problems

Differences in Public Headers

Currently, there are five OpenSolaris public header files that libresolv2 uses in place of the versions distributed with ISC libbind. These are: <netdb.h>, <resolv.h>, <arpa/nameser.h>, <arpa/nameser_compat.h>, <arpa/inet.h>. There are many differences between the OpenSolaris versions and the ISC versions, which are newer (see Appendix A for a detailed list). Some of these differences constitute a change of API.

Porting the latest version of ISC libbind without changing the API in these public headers requires workarounds, which will have to be teased out and re-applied each time libresolv is upgraded to a new version of ISC libbind. All of these changes are documented, but new ones inevitably come up with each upgrade.

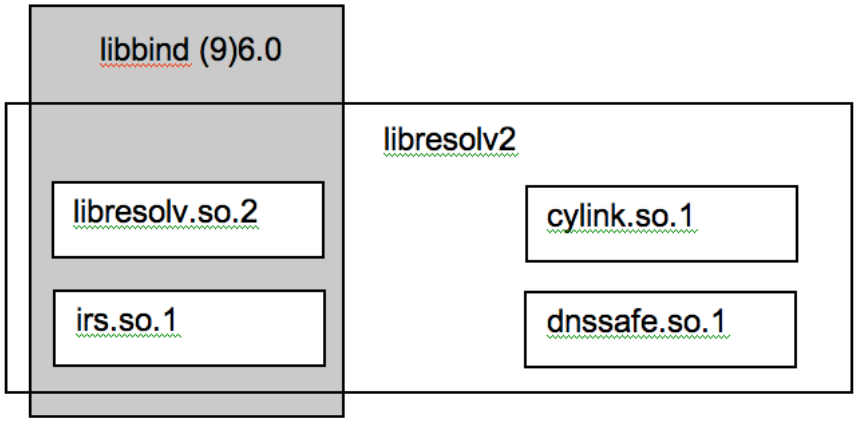
Splitting of Libraries

Currently, libresolv2 consists of four dynamic libraries:

1. libresolv2.so.2: contains the client side functionality of the old libresolv library. Does not include IRS and crypto functions (described below).
2. irs.so.1 - - Information Retrieval Service - provides access to multiple information services
3. cylink.so.1 - CYLINK crypto library for DNSSEC/TSIG.
4. dnssafe.so.1- DNSSAFE crypto library for DNSSEC/TSIG

The splitting of libresolv2 into separate dynamic libraries, was apparently done in order to reduce the size increase of libresolv.so.2 for the on81 BIND 8.2.2-p5 port.

ISC libbind, has no such split; ISC packages all the functionality in a single library. Whereas the code for all four libraries were included in libbind 8.4.1, only the first two are in libbind (9)6.0. The figure below illustrates this relationship.



Older versions of ISC BIND included the source code for the dnssafe and cylink crypto libraries. This source code is currently part of the Solaris libresolv2 source tree, but is compiled into separate libraries: dsnsafe.so.1 and cylink.so.1.

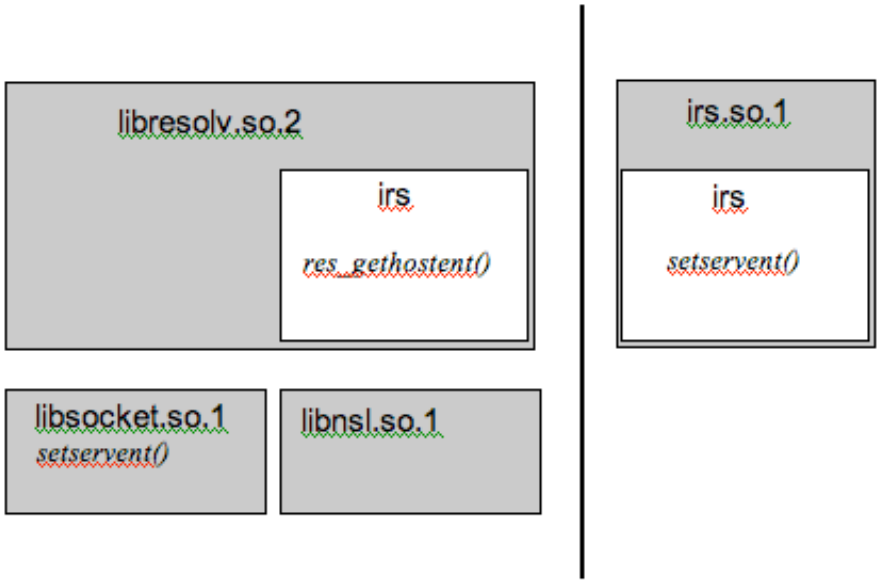
The dnssafe and cylink source code are no longer part of the ISC distribution and are likely to have changed. Maintaining them represents an increase in scope of the porting effort that may not be necessary or desired.

Overlap with libsocket/libnsl

The irs functionality in ISC libbind was not completely extracted into irs.so.1. Instead, only about half of the irs functions were actually split out into irs.so.1; the other half remained in libresolv2.so.2. The criteria for determining which functions were split out is not documented, but the rationale appears to have something to do with overlap with libnsl and libsocket.

Many of the irs functions that overlap with libnsl and libsocket were moved into irs.so.1 (e.g. setservent), but some remained in libresolv2.so.2 and of those some were renamed (e.g. gethostent became res_gethostent). The result is the ISC code ends up calling the libsocket version of some functions (e.g. setservent) but the ISC version of others (e.g. res_gethostent).

The situation with irs splitting is illustrated in the figure below.



Dependencies between the two pieces of irs cause a porting problem. The way this was dealt with in the last port was to write stub functions that reside in libresolv2. When called, the stubs load irs.so.1 and call the missing function in that library.

Consolidating the irs functions into libresolv2 might simplify future porting efforts.

Proposed Strategy

This section outlines a unified strategy for addressing each of the incompatibility problems described above, with the goal of eliminating workarounds from the ISC code. It is based on the following assumptions:

1. the interface defined by Solaris public header files can be extended (but not reduced)
2. ISC libbind functions conflicting with libnsl or libsocket must be renamed
3. the dynamic library structure can be changed. (e.g. irs.so can be folded back into libresolv.so)

If any of these assumptions were to change, the strategy outlined below would also change.

A large number of workarounds in the ISC code arise from trying to compile it against the OpenSolaris public header files. Since our goal is to reduce the number of workarounds in ISC code, we should consider doing something different. Ideally, the public header files would be updated to match the latest ISC versions, but this cannot be done fully without breaking the API/ABI. So as a tradeoff, we propose doing the following:

1. Update the OpenSolaris public headers as much as possible without breaking the API/ABI (see Appendix A for details).
2. Compile the ISC code against the function prototypes defined in its own header files, using compiler directives to rename functions.
3. Where necessary, implement adapters that wrap ISC functions to provide backward compatibility.

This strategy can also be used to extract other workarounds from the ISC source code; for example; workarounds related to overlap with libnsl/libsocket, library splitting, and the undoing of ISC's identifier hiding mechanisms. The examples given in the next section show how the adapter strategy is implemented.

The proposed strategy addresses all the aforementioned issues in a unified way, reducing the number of workarounds required, and moving changes outside the ISC source code. Compatibility with the existing interface is preserved, and the evolution of the interface is simplified - adapters can be updated and/or removed without touching the ISC code. The net result is reduced cost and improved reliability of future porting efforts.

Example:

To illustrate our approach we consider the porting issues associated with two functions from libresolv2. The examples show how the issues are currently addressed in libresolv2 and how they would be addressed with an adapter strategy.

Consider the inet_ntoa(), defined in inet/inet_ntoa.c. This function has one porting issue: it conflicts with a function of the same name in libnsl. In the current implementation, this issue is addressed with the following change to inet/inet_ntoa.c.

```
#ifndef SUNW_LIBNSL /* We don't need this if linked with libnsl */
char * inet_ntoa(struct in_addr in) {
      ...
}
#endif      /* SUNW_LIBNSL */
```

The main problem with this solution is that it resides in the source ISC code. It must be kept track of and re-applied each time libresolv2 is upgraded to a new version of the ISC code. A better solution is to just change the makefile so the file is not compiled. That way the change persists and nothing needs to be done for future ports. Moreover, if we decide one day that we like ISC's inet_ntoa, all we have to do is compile it.

Now, for a more interesting case, let's take a look at the function sethostent(), defined in irs/gethostent.c. This function also conflicts with a function of the same name in libnsl, but this time the issue is resolved by passing the following directive to the compiler:

```
-Dsethostent=res_sethostent
```

This is a big improvement, as the change is implemented outside the ISC source code. However, there is another problem: ISC defines sethostent as returning void, while the OpenSolaris public header defines it as returning int. Renaming the function in the Makefile does not solve this problem and so the current implementation makes the following change to irs/gethostent.c

```
#ifdef ORIGINAL_ISC_CODE
void
#else
int
#endif
sethostent(int stayopen) {
      struct net_data *net_data = init();
      sethostent_p(stayopen, net_data);
#ifdef      ORIGINAL_ISC_CODE
#else
      return (0);
#endif /* ORIGINAL_ISC_CODE */
}
```

This is another change that will have to be tracked and added for each new upgrade.

Fortunately, all of these issues can be resolved without any changes to the ISC code. Using the adapter strategy, we just add the following three lines to the header file port_after.h:

```
#define inet_ntoa irs_inet_ntoa /* eliminate conflict with libnsl */
#define sethostent res_sethostent /* eliminate conflict with libnsl */
void sethostent(int); /* use ISC definition */
```

The first two lines simply rename the functions so they don't conflict with libnsl. The key here is that, because port_after.h is loaded *after* all other header files, the functions defined in the OpenSolaris headers have different names and thus may have different return types. The third line simply provides the ISC definition of the (renamed) function.

In this case, because the sethostent function conflicted with libnsl and was renamed, there was no need to do anything regarding the difference between the ISC definition and OpenSolaris the public header definition. However, there are cases where we want to expose functions where the ISC definition differs from the OpenSolaris header definition. In this case we do the above renaming and add a simple wrapper like this:

```
int sethostent(int stayopen) {
   res_sethostent(stayopen);
   return 0;
}
```

The API/ABI is preserved and no changes are made to the ISC code.

Appendix A: Changes To Public Headers

The first part of the strategy outlined involves updating the OpenSolaris public headers as much as possible without breaking the API/ABI. To do this we must identify the differences between ISC's headers and the OpenSolaris headers and determine the impact of each.

Changes that would break the existing libresolv2 API are highlighted.

There are five public header files that libresolv2 uses in place of the ISC versions. These are: <netdb.h>, <resolv.h>, <arpa/nameser.h>, <arpa/nameser_compat.h>, <arpa/inet.h>

The following sections list, for each header file, the changes that would need to be made to the OpenSolaris version to bring it up to date with ISC libbind (9)6.0. The changes are categorized based on whether or not the change appears to be essential to make ISC libbind work. Of course all the changes could be considered essential for providing a fully reliable and maintainable port, and we should incorporate as many of them as possible. However, for this analysis changes are categorized as essential if and only if the ISC code would not compile without them.

Changes to resolv.h

Essential:

1. add a field to res_state. _rnd (16 bytes) was squeezed in just before _pad, which is now 16 bytes smaller, but aligned 16 bytes to the right. **MAY BREAK API.**
2. add res_rdninit, res_nrandomid, and res_nopt_rdata
3. add RES_NSID, replacing RES_NO_NIBBLE. Technically, this breaks the interface, but RES_NO_NIBBLE was already deleted in BIND 8.4.x and libresolv2 only references RES_NO_NIBBLE2. The public header just wasn't updated. **MAY BREAK API.**
4. remove RES_NO_BITSTRING. **MAY BREAK API.**
5. add "| RES_NO_NIBBLE2" to RES_DEFAULT
6. add prototypes for _getshort and _getlong

Other changes:

1. add #includes of <sys/cdefs.h> <sys/socket.h>
2. replace uint_t, ulong_t, uchar_t with uint, ulong, u_char. **MAY BREAK API.**
3. replace unsigned int with u_int32_t. **MAY BREAK API.**
4. replace extern "C" { ... } with __BEGIN_DECLS ... __END_DECLS
5. add ANSI C idenitifier hiding for about 70 symbols: e.g. #define b64_ntop __b64_ntop

Changes to netdb.h

Essential:

1. add EAI_OVERFLOW (replacing EAI_BADHINTS) **MAY BREAK API.**
2. add EAI_MAX, EAI_PROTOCOL
3. add AI_MASK
4. add #define SCOPE_DELIMITER '%'
5. change the value of several AI_* #defines. **MAY BREAK API.**

Other changes:

1. add #includes of <sys/cdefs.h> and <stdio.h>
2. change the signature of sethostent, setservent, setnetent, setprotoent, setnetgrent, endhostent, endservent, endnetent, endprotoent, and endnetgrent. These return void, not int. (NOTE: return value is toggled in public headers by #if !defined(_XPG4_2) || defined(__EXTENSIONS__) **MAY BREAK API.**
3. use __P around arguments
4. replace socklen_t/size_t with int. **MAY BREAK API.**
5. replace void* with char* **MAY BREAK API.**
6. replace in_addr_t with unsigned long **MAY BREAK API.**
7. replace extern "C" { ... } with __BEGIN_DECLS ... __END_DECLS
8. add hostent_data, netent_data, protoent_data, servent_data (for hpux, osf, aix)
9. replace MAX_ADDRS and MAX_ALIASES with _MAX_ADDRS and _MAX_ALIASES. (for hpux, osf, aix) **MAY BREAK API.**
10. add MAX_LINELEN (for hpux, osf, aix)
11. add #ifndef protection against redefine
12. ISC netdb.h doesn't have some #defines (e.g. IPSEC_PROTO, _PATH_IPNODES, _PATH_IPSECALGS, _PATH_NETMASKS etc.) and structs (ipsecalgent) **REMOVAL MAY BREAK API.**

Changes to arpa/nameser.h

Essential:

1. add #define NS_MAXLABELS, NS_MAXNNAME, and NS_MAXPADDR
2. add definitions of ns_nname, ns_nname_ct, ns_nname_t, ns_namemap ns_namemap_t, and ns_namemap_ct
3. add __ns_rr2 struct
4. add about 10 new values to enum __ns_type
5. add NS_OPT_NSID
6. add ns_newmsg struct
7. add ns_rr_nname(rr) and ns_rr_nnamel(rr) macros
8. add ns_parseerr2, ns_name_eq, ns_name_owned, ns_name_map, ns_name_labels
9. add ns_newmsg_init, ns_newmsg_copy, ns_newmsg_q, ns_newmsg_rr, ns_newmsg_done, ns_newmsg_flag etc.
10. add ns_rdata_unpack, ns_rdata_equal, ns_rdata_refers

Other Changes:

1. ISC doesn't have ns_updrec. The ISC definition is in res_update.h and is different from the public one. **REMOVAL MAY BREAK API.**
2. add #includes of <sys/cdefs.h>
3. replace u_char_t with u_char. **MAY BREAK API.**
4. replace uint16_t with u_int16_t. **MAY BREAK API.**
5. add ANSI C idenitifier hiding for about 20 symbols: e.g. #define ns_name_length __ns_name_length
6. replace extern "C" { ... } with __BEGIN_DECLS ... __END_DECLS

Changes to arpa/inet.h

Essential:

1. add #includes of <sys/cdefs.h> (needs #define __P(x))
2. add inet_neta, inet_net_ntop, inet_cidr_ntop, inet_cidr_pton, inet_nsap_addr, inet_nsap_ntoa

Other Changes:

1. replace in_addr_t with u_long/unsigned long. **MAY BREAK API.**
2. add ANSI C idenitifier hiding for about 20 symbols e.g. #define inet_addr __inet_addr
3. functions are not declared extern
4. replace extern "C" { ... } with __BEGIN_DECLS ... __END_DECLS

Changes to arpa/nameser_compat.h

Essential:

No major changes:

Other Changes:

1. ISC does not use extern "C" { ... }