Bind 9 - DLZ Phase 2 Project Plan

Objectives:

To add additional drivers and documentation to Bind-DLZ, allowing it to be used with a wider variety of backend databases. Also, to show the impact of using DLZ by doing performance testing with a variety of backend databases.

History:

For a complete history of the DLZ project, refer to the original DLZ project proposal in addition to this proposal for DLZ – Phase 2.

DLZ – phase 1 was completed in June of 2002. Having successfully implemented DLZ, it was decided to continue the project with a second phase. This phase will continue to build upon the first by adding additional drivers and documentation to the DLZ project. Ideally, this will lead to wider adoption and use of DLZ.

NLnet has graciously offered to continue sponsoring the development of DLZ. NLnet is a non-profit organization committed to network (Internet) related research and development. As part of this commitment, they sponsor the development of software that would benefit the Internet and make that software available as open source.

Requirements:

1. **It will be open source.**
   Open source designates software for which the source is freely available for reuse. For the definition, see http://www.opensource.org here. This is an essential condition of funding, as the NLnet Foundation supports network software projects that are open for reuse. The source will be released under a BSD style license.

2. **It will use the new DLZ / SDLZ database interface developed in phase 1.**
   DLZ added a new interface to Bind 9 to allow zones to be looked up in an external database. This capability was used in the development of a PostgreSQL driver in Phase 1 of the DLZ project.

3. **It may modify the existing DLZ / SDLZ database interface.**
   The interface developed in Phase 1 of the DLZ project is still subject to modification. It was built and designed to allow nearly any driver to be developed for use with DLZ. However, if an error or limitation of DLZ is found, then phase 2 may modify the DLZ interface to fix an error or remove a limitation.

4. **It will be merged into Bind 9 public source.**
   The goal of this project is to develop code that will improve Bind's functionality. Ideally, these changes will become part of Bind 9 and not need to be maintained as a separate patch. To that end, source code enhancements will always attempt

to follow the current Bind 9 coding style and standard. However, we cannot guarantee that our modifications will become part of Bind. So this is not a requirement of funding because it is out of our control. The best we can do is to develop the code and present it for inclusion into Bind 9.

5. **To the extent possible, the drivers should support multiple operating systems.**
   This will be limited to the developer's ability to test the drivers on multiple operating systems. However, every effort should be made to allow the drivers to operate on any system that the database it supports operates on.

6. **Dynamically loadable zone drivers may only implement limited functionality.**
   DLZ provides an interface to allow domain name lookups and full zone transfers (not incremental zone transfers). To the extent possible, all drivers developed as part of this project will implement and provide full DLZ capabilities. However, some drivers may not be able to provide zone transfer capabilities because of a limitation of the database backend or something similar.

7. **It will be properly documented for both users and future developers.**
   Good documentation is essential for everyone. Unfortunately, Bind 9's source is not very well documented, and that makes understanding the code and modifications to it difficult. The modifications to Bind in phase 1 of DLZ were documented in the source as code comments in an effort to allow future developers to understand what is going on. Where possible, comments were added which explained relevant parts of Bind 9's existing code.

   Several additional documents were created to explain what DLZ / SDLZ was and explain the use of the new API(s). Additional documentation was created for the PostgreSQL driver implemented in phase 1. DLZ – phase 2 will continue in this tradition, creating documentation for each of the drivers developed as part of this project. Existing documentation from phase 1 will also be augmented with more examples.

8. **After each new driver is finished, a full release of DLZ will be published.**
   After each driver is finished, a full release of the project will be published on the DLZ website on Sourceforge.net. Additional releases may be made after the accomplishment of any suitable milestone.

9. **DLZ will keep pace with Bind 9 releases.**
   Bind 9 continues to be developed separately from DLZ. As new releases of Bind are published, DLZ will be brought in line as part of phase 2. Any work in progress developing a new driver will be completed and released to the DLZ website before DLZ is brought in line with the newest Bind 9 release. This will be done so that a unit of work (a driver in development) can be completed without the possibility of introducing new bugs that could be caused during migration to a newer version of Bind.

10. A MySQL Driver will be implemented.

MySQL is a popular open source database. A DLZ driver for MySQL has been the most requested item when people ask about the project before joining the mailing list. Development of a MySQL driver would make use of DLZ much easier for a large group of people. The driver will allow looking up zones in the database, looking up zone records, and zone transfers from the same database. Ideally, this driver will be included into Bind's distribution as a built in driver.

11. A Berkeley DB Driver will be implemented.

Evidence of interest in supporting drivers for Berkeley DB has been witnessed on Bind mailing lists. A Berkeley DB driver will provide a very high-speed database for use with DLZ. The full capabilities of Berkeley DB will be used, allowing multiple separate processes to simultaneously access and update the database. It will NOT use the older "dbm" style API. I.E. Berkeley DB 4.0 and the concurrent data store model will be used. The driver will allow looking up zones in the database, as well as looking up zone records in the same database. The Berkeley DB driver should be able to support zone transfer. Ideally, this driver will be included into Bind's distribution as a built in driver. Additionally, a utility API (and possibly a command line tool) will be built for adding / removing zones, records, etc from the Berkeley DB database.

12. A File System driver will be implemented.

DNS data is hierarchal in nature, just as a file system is. New advanced file systems such as Reiser and memory file systems provide very high speed file and data access. The file system driver will provide a high-speed database for use with DLZ. This driver can be used with a Reiser file system when permanent storage of data is required. It can also be used with the memory file system to act as a high performance data cache for DLZ. When used in this manner, a secondary database of some sort (PostgreSQL, MySQL, Reiser, etc) would be used as permanent storage. Some separate utility would be responsible for updating the data in the cache. In the interest of speed, data will be gleaned from directory entries and not from data within files. Listing directory entries is faster than opening several files. Additionally, the Reiser file system can pack zero length files very tightly by design, allowing a maximum of data to be held in a minimum of physical drive space. The file system driver should be able to support zone transfer. Ideally, this driver will be included into Bind's distribution as a built in driver.

13. A LDAP driver will be implemented.

Many larger ISP's may be interested in support for Bind's new features. Many of these larger ISP's use LDAP for login and other configuration information. An LDAP driver would allow Bind 9 to easily integrate with existing systems for these large ISP's. The LDAP driver will allow looking up zones in the database, as well as looking up zone records in the same database. If possible, this driver will also allow zone transfer. Ideally, this driver will be included into Bind's distribution as a built in driver.

14. (Optional) An ODBC database driver will be implemented.

ODBC provides an interface to many different databases through a single interface. By developing an ODBC driver, we can support many databases with one driver. This driver will be of most use on Windows based systems, where ODBC is widespread. The driver will allow looking up zones in the database, as well as looking up zone records in the same database. The ODBC driver should be able to support zone transfer. Ideally, this driver will be included into Bind's distribution as a built in driver.

15. (Optional) Performance testing will be performed.

One hindrance to the acceptance of DLZ is a perceived performance loss as compared to standard Bind. Performance testing using a variety of DLZ drivers will show the performance of DLZ when using each database / driver combination. This will allow new DLZ users to best decide which driver / database combination will best meet their needs for speed / maintainability. Performance test results will be published on the DLZ website. Additionally, an article will be written about DLZ and the performance tests in the hopes of having it published in an online magazine.

Resources:

- Several computers running Linux.
- A computer running Windows NT/2000.
- PostgreSQL database
- Berkeley Database
- MySQL Database
- LDAP server
- ODBC Database (optional)
- Contact with the ISC, or a developer with commit authority to the Bind 9 source tree and documentation.
- Beta testers on a variety of operating systems.
- Website and mailing list.

Existing resources available:

- At least 1 computer running Windows 2000.
- Several computers running Linux.
- PostgreSQL database (open source)
- Berkeley Database (open source)
- ODBC Database (MS Access or Microsoft Data Engine)
- LDAP server (openLDAP open source)
- MySQL database (open source)
- 1 contact with ISC / developer with commit authority (???)
- Website and mailing list. (DLZ website on Sourceforge.net, DLZ mailing list on Sourceforge.net, Bind9-workers & Bind9-users mailing lists.)

Development Timeline:
15.5 man weeks total, approx 620 hours including all optional components
11 man weeks total, approx 440 hours including no optional components

2 weeks

Development of MySQL driver, documentation & testing

2.5 weeks

Development of File System driver, documentation & testing
Fix any bugs in MySQL driver.

3.5 weeks

Development of Berkeley DB driver, documentation & testing.
Development of Berkeley DB utility API and possibly a command line tool, documentation & testing.
Fix any bugs in previous drivers.

3 weeks

Development of LDAP driver, documentation & testing
Fix any bugs in previous drivers.

3 weeks (Optional)

Development of ODBC driver, documentation & testing

1 week (Optional)

Setup several Linux machines to use in performance testing. Perform tests, aggregate and analyze results. Publish results to web and develop article for publication in online magazine.

0.5 weeks (Optional)

If possible, modify Bind configuration script to allow options such as "--with-dlz-postgres", "--with-dlz-mysql", etc. This would allow better integration with Bind, as DLZ could be integrated with the Bind distribution but not built by default. If a user wanted DLZ, they could specify the correct options to build Bind with DLZ and the DLZ drivers they need. Also, add more example configurations for the existing PostgreSQL driver.

The timeline above is in logical weeks. The actual development time will take much longer, as the project will not be the primary focus of the developer.