Release Bulletin
SYBASE *Production* Release 4.0.2
NeXT DB-Library

1. Product Summary

Enclosed is the NeXT DB-Library Release disk. It contains the Sybase 4.0.2 Production DB-Library.

This document contains release information for the SYBASE *Production* Release 4.0.2.

2. Release Contents

The following is a summary of the new features in the Release 4.0.2 DB-Library.

2.1. Support for text and image Datatypes

Modifications to existing routines to return *dbint* instead of *int*. A new option, DBTEXTLIMIT, to set maximum datalength in the front end, is provided for *dbsetopt()*. Seven new DB-Library routines are provided to return text pointer length, text pointers, text timestamps, change text values in the database, etc.

2.2. *BROWSE* Mode

Ten new DB-Library routines are provided to support the new database *browse* capability.

2.3. Return Parameters and Status

New DB-Library routines are provided to support the return of status and return parameters from stored procedures.

2.4. Remote Procedure Call Support

New DB-Library routines are provided to support Remote Procedure Call execution. ``Additional Error/Informational Data'' User message handling routines installed using *dbmsghandle()* will get additional information, including the server name, procedure name and line number, as additional parameters.

2.5. BCP Features

Refer to information below in Section 4.3. The current *bcp* routines have been replaced with new *bcp* routines that support re-use of a DBPROCESS structure and the DB-Library-like error-handling under program control. The old *bcp* routines are provided for compatibility, but new functionality such as support for text and image is provided in the new library only.

3. Problem resolutions

Problem resolutions reports for this release are located in the file cpr_dblib that is in the *~sybase/install/SPR* directory included on your disk.

4. Documentation Clarifications/Updates

4.1. Release 4.0.2 SQL Toolset and SQL Server Compatibility

Release 4.0.2 includes changes in the logical communication protocol (Tabular Data Stream™ - TDS) used between client programs and the SQL Server. Client programs include any program that uses the client library, DB-Library. The standalone utilities (*bcp, isql, etc.*) use DB-Library and are therefore client programs. Any user program that uses DB-Library is also considered a client program. The following table shows which release versions are compatible:

Notice that any client program can communicate with a pre- 4.0 SQL Server. All combinations are compatible. If a Release 4.0.2 front-end program is connected to a pre-4.0 server, 4.0 server features such as Remote Procedure Calls will not be available.

Question:

Will my existing client applications run with a Release 4.0.1 SQL Server?

Answer:

Yes, any existing client applications will run with a Release 4.0 SQL Server. There is no need to re-compile or re-link your client applications. However, pre-4.0 client applications will not be able to use some of the new 4.0 features such as such as text and image datatypes, return values from stored procedures, *browse* mode, etc.

Question:

If I have two SQL Servers at my site, one Release pre-4.0 and one Release 4.0, how will client programs that need to use both SQL Servers be affected?

Answer:

You can leave your client programs at pre-4.0 release level. They will then work fine with both release levels of SQL Servers. However, if you want to use some of the new Release 4.0 SQL Server features mentioned above, you will need to use 4.0 or later client programs. If you want to use a new client program that uses the new 4.0 features on both SQL Servers, you will have to upgrade your pre-4.0 SQL Server to Release 4.0.

Question:

Will my 4.0.2 client applications work with a pre- 4.0 server?

Answer:

Yes. However, 4.0 server features such as Remote Procedure Calls will not be available.

4.2. Changes to DB-Library Functions

Problem:

Prior to release 4.0, *dbresults()* was not always treating stored procedures like ``black boxes.'' In some circumstances, an invocation of a stored procedure would require more calls to *dbresults()* than the documentation indicated. In other cases, after the execution of a stored procedure, the DBCOUNT() macro would return the number of rows affected by the last SQL command in the stored procedure, even if that last command was not a SELECT statement.

Change or Fix:

Starting with release 4.0, *dbresults()* now conforms to its documented behavior: It must be called once for each SELECT statement executed by
a stored procedure (even if that SELECT statement returns zero rows), or once for a stored procedure that never executes a SELECT statement. As before, the easiest way to do this is to call *dbresults()* until it returns NO_MORE_RESULTS.

Here are some illustrative examples:

This stored procedure:

```
create proc proc1 as
```

```
select * from foo
insert foo values(1)
insert foo values(1)
select * from foo
```

requires 2 calls to *dbresults()* because it executes 2 SELECT statements.
This stored procedure:

```
create proc proc2 as
select * from foo
insert foo values(1)
insert foo values(1)
```

requires 1 call to *dbresults()* because it executes 1 SELECT statement.

This stored procedure:

```
create proc proc3 as
insert foo values(1)
insert foo values(1)
select * from foo
```

requires 1 call to *dbresults()* because it executes 1 SELECT statement.
This stored procedure:

```
create proc proc4 as
select * from foo
select * from foo
```

requires 2 calls to *dbresults()* because it executes 2 SELECT statements.

This stored procedure:

```
create proc proc5 as
insert foo values(1)
insert foo values(1)
```

requires 1 call to *dbresults()* because it contains no SELECT statements. Every stored procedure requires at least 1 call to *dbresults()*.

DBCOUNT() now returns the number of rows returned by the latest SELECT statement executed by a stored procedure, and -1 after a stored procedure that does not execute any SELECT statements. Note that a stored procedure that contains no SELECT statements may execute a SELECT by calling another stored procedure that contains a SELECT.

Programs that took advantage of DBCOUNT()'s previous behavior in order to examine the rowcount of a stored procedure's last INSERT, UPDATE, or DELETE statement should modify their stored procedures and DB-Library applications to do one of the following:

1. explicitly select @@rowcount after the operations of interest,
2. return @@rowcount as the stored procedure's return value, or
3. return @@rowcount as one of the stored procedure's output parameters.

Problem:

Conversions from floating-point and datetime values to character strings were not precise enough to use in WHERE clauses of SQL statements.

Change or fix:

dbbind() and *dbconvert()* now produce a fully precise character string which will match an original data value retrieved from the SQL Server. Note that some application programs may have some character buffers that need to be lengthened to hold these fully precise values: The addition of seconds and milliseconds to datetime strings makes them 7 characters longer.

4.3. Changes to BCP Subroutines

The C language interface to the bulk-copy library has changed. All of the new functions begin with ``*bcp_*'', where the old names began with ``*bcp*''.

Here are some of the problems in previous releases that were fixed by changing the library's interface:

Problem:
Wrote all error messages to a host file, so errors could not be handled under program control.

Change or fix:
Now uses the DB-LIBRARY error-handling mechanism.

Problem:

Lengths of host file columns were either fixed or determined by a terminator. This meant that NULL values for numerical or binary data types could not be represented, unless the host file column was in ASCII. This had two disadvantages:

1) Conversion from DBFLT8 to ASCII and back can result in a loss of precision.

2) ASCII files are bigger and take longer to read and write than binary files.

Change or fix:

Host file columns can now contain an optional 1, 2, or 4-byte length-prefix, which can be set to 0 for NULL data values.

Problem:

bcpbind() and *bcpformat()* used *strlen()* to determine the length of terminator sequences, even for binary data. This means that terminator sequences could not contain any zero-valued bytes.

Change or fix:

bcp_bind() and *bcp_colfmt()* now receive an additional ``termlen'' parameter.

Problem:

Used constants IN and OUT, likely to collide with users' definitions.

Change or fix:

Now uses DB_IN and DB_OUT.

Problem:

bcp row-batching was based on the number of rows sent to the Server. Some customers doing telemetry applications wanted to do batching based on time as well.

Change or fix:

Batching behavior is now user-controllable.

Problem:

bcpsendrow() could not send NULL data values, because a data-length of 0 meant ``default datalength,'' and a NULL data pointer was illegal.

Change or fix:

Datalength of 0 now means ``zero-length data'', and -1 means ``use the length-prefix and/or terminator to determine the data's length''.

Problem:

bcpinit() used to open a new DBPROCESS connection for each *bcp* operation. This made it impossible to copy data into or out of temporary tables, or to embed a ``slow'' *bcp* operation within a transaction.

Change or fix:

BCP-Library now uses a normal DBPROCESS for its communications with the server. The application programmer can use this DBPROCESS for any purpose either before or after the *bcp* operation.

4.4. Changes to BCP User Interface

The user interface of the *bcp* utility program has changed. Here are some of the problems which were solved by this change:

Problem:

In the standalone *bcp* program, file formats were specified interactively, with no straightforward way to save file formats. This is a problem because the file format used to create a host file must be identical to the file format used to read that host file.

Change or fix:

The standalone *bcp* program now reads and writes format files.

Problem:

In the standalone *bcp* program, arguments such as the table name, host file name, etc. were specified interactively, so it was impossible to call the standalone *bcp* program from a script. Some users had managed to ``reverse-engineer'' some input files, but there was no easy way to do this.

Change or fix:

The standalone *bcp* program now accepts arguments on the command-line, and format information from a host file.

We have provided some software to help users convert over to the new interface. The ``bcptrans'' program exhibits the same user interface as the old *bcp* program, and outputs two files: a command line for the new *bcp* program (bcptrans.cmd), and a format file referenced by that command-line (bcptrans.fmt). These two files can then be used as input to the new program. To fix a bug related to data-truncation, the formats for data files created via this method differ from old data files in one regard:

If the default host-type and length were used for float or datetime values, the old program specified a host file field-width of 20 characters. *Bcptrans* will specify 25 characters for float, and 26 characters for datetime.

4.5. Conversion of Input Scripts with BCPTRANS

If you hand-created input scripts for use as format files with pre- 4.0 *bcp*, you can use *bcptrans* to create new format files for use with the new *bcp*. The syntax of *bcptrans* is:

```
bcptrans [[database_name.]owner.]table-name [ -3]
   {in | out} file_name [errors]
```

The conversion program produces two files:

1. *bcptrans.cmd* contains the appropriate command line for the new *bcp* program.

2. *bcptrans.fmt* contains the appropriate format file for the new *bcp* program.

These two files can then be used as input to the new *bcp*.

The -3 flag to *bcptrans* must be used to convert an old format file when you are planning to copy IN an old datafile you created using the default type and length for float and datetime data. (Otherwise, the format file *bcptrans* creates will use the new default lengths for float and datetime data, which have been changed from 20 characters to 25 characters for float, and 26 characters for datetime). Do not use the -3 flag for converting format files for any data that does not match this specification.

The following table is a guide to using the ``-3'' flag with *bcptrans*:

4.6. Changes in type-definitions in sybdb.h

Some undocumented type-definitions have been removed from *include/sybdb.h*. Following is a list of the definitions that have been deleted, and the documented definition that should be used instead:

Old definition:	*New definition:*
MONY	DBMONEY
DATE	DBDATETIME

In addition, one type-definition has been changed to avoid collisions with users' types:

Old definition:	*New definition:*
BOOL	DBBOOL

``BOOL'' has been removed from this version of sybdb.h.

4.7. Include-File Name Changes

Two files which are indirectly included by DB-Library application programs have changed their names: *sybloginrec.h* is now *syblogin.h*, and *sybdbtokens.h* is now *sybdbtoken.h*. These filenames were shortened to improve their portability to operating systems with restrictions on the length of filenames. Since these two files are not included directly by DB-Library application programs, no source-code changes will be necessary, and the majority of DB-Library programmers will not notice any change at all.

Only customers whose makefiles' dependency-lists include all include-files, even those that are included only by other include-files, are affected. If your makefiles contain references to *sybloginrec.h* or *sybdbtokens.h*, you should edit the makefile to use the new, shortened versions of the names. If your makefiles do not contain references to these two files, no change is required.

5. Highlighted Known Problems

5.1. Problem Reports

Refer to the following files in the */usr/sybase/install/SPR* directory included on your tapes for complete listings of known problems.

spr_dblib (DB-Library)

5.2 Installing Samples with a 4.0 Server

The 4.0.2 distribution of DB-Library includes the script
/usr/sybase/sample/dblibrary/installsample.
It is used to set up the server environment to run the sample programs in
/usr/sybase/sample/dblibrary.

The *installsample* script is designed to work with a 4.0.1 version of *startserver*, and does not work correctly with a pre-4.0.1 *startserver*.

To set up the server environment correctly, follow the instructions in the ``Notes'' section at the end of the file:

/usr/sybase/sample/dblibrary/README.

6. Technical Support

The bundled SQL Server and DB-Library software on NeXT do not include Sybase technical support. For information about purchasing support or additional products from Sybase, call 1-800-447-9227. For help with NeXT and NeXT bundled software, contact NeXT.