IBM® DB2 Universal Database™

Application Development Guide: Building and Running Applications

Version 8.2

SC09-4825-01

IBM® DB2 Universal Database™

Application Development Guide: Building and Running Applications

Version 8.2

SC09-4825-01

Before using this information and the product it supports, be sure to read the general information under *Notices*.

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide

To order DB2 publications from DB2 Marketing and Sales in the United States or Canada, call 1-800-IBM-4YOU (426-4968).

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Welcome to DB2 application development

This preface provides the information you need to get started with DB2 application development, specifically with the DB2 Developer's Edition products. It also gives an overview of all three volumes of the Application Development Guide.

For instructions on how to install a product that is available with DB2 Developer's Edition, either refer to the appropriate *Quick Beginnings* book, which is available from the PDF CD, or check the product CD itself for installation instructions.

If you want to access DB2 documentation on your computer and you have not yet installed the DB2 Information Center, then refer to either "Installing the DB2 Information Center using the DB2 Setup wizard (UNIX)" on page 292 or "Installing the DB2 Information Center using the DB2 Setup wizard (Windows)" on page 294. The DB2 Information Center contains documentation for DB2 Universal Database and DB2 related products.

DB2 Developer's Edition Products

DB2® Universal Database provides two product packages for application development: DB2 Personal Developer's Edition and DB2 Universal Developer's Edition. The Personal Developer's Edition provides the DB2 Universal Database™ and DB2 Connect™ Personal Edition products that run on Linux and Windows® operating systems. The DB2 Universal Developer's Edition provides DB2 products on these platforms as well as on AIX®, HP-UX, and the Solaris Operating Environment. Contact your IBM® representative for a full list of supported platforms.

Using the software that comes with these products, you can develop and test applications that run on one operating system and access databases on the same or on a different operating system. For example, you can create an application that runs on a Windows operating system but accesses a database on a UNIX® platform such as AIX. See your License Agreement for the terms and conditions of use for the Developer's Edition products.

The Personal Developer's Edition contains several CD-ROMs with all the code that you need to develop and test your applications. In each Media Pack, you will find:

- The DB2 Universal Database product CD-ROMs for Linux and Windows operating systems. Each CD-ROM contains the DB2 server and Application Development Client for the supported operating system. These CD-ROMs are provided to you for testing your applications only. If you need to install and use a database, you have to get a valid license by purchasing the Universal Database product.
- DB2 Connect Personal Edition. If you need to install and use this product, you have to get a valid license by purchasing DB2 Connect Personal Edition.
- A DB2 publications CD-ROM containing DB2 books in PDF format
- A DB2 Information Center CD-ROM containing DB2 documentation in HTML format
- DB2 Net Search Extender (Windows only)

- DB2 Spatial Extender (Windows only)
- VisualAge® for Java™, Entry Edition

The Universal Developer's Edition contains CD-ROMs for all the operating systems supported by DB2, and include the following:

- DB2 Universal Database Personal Edition, Workgroup Server Edition, and Enterprise Server Edition
- DB2 Connect Personal Edition and DB2 Connect Enterprise Edition
- Administration clients for all platforms. These clients contain tools for administering databases, such as the Control Center and the Event Analyzer. These clients also allow you to run applications on any system.
- Application development clients for all platforms. These clients have application development tools, sample programs, and header files. Each DB2 AD client includes everything you need to develop your applications.
- Run-time clients for all platforms. An application can be run from a run-time client on any system. The run-time client does not have some of the features of the administration client, such as the DB2 Control Center and Event Analyzer, and so takes up less space.
- A DB2 publications CD-ROM containing DB2 books in PDF format
- A DB2 Information Center CD-ROM containing DB2 documentation in HTML format
- DB2 Net Search Extender
- DB2 Spatial Extender

In addition, for both Developer's Editions you get copies of other software that you might find useful for developing applications. This software can vary from time to time, and is accompanied by license agreements for use.

Related concepts:

- "DB2 UDB Enterprise Server Edition" in the *Quick Beginnings for DB2 Servers*
- "DB2 Run-Time Client" in the *Quick Beginnings for DB2 Clients*
- "DB2 Administration Client" in the *Quick Beginnings for DB2 Clients*
- "DB2 Workgroup Server Edition" in the *Quick Beginnings for DB2 Servers*

Related tasks:

- "Installing DB2 Personal Edition - overview (Windows)" in the *Quick Beginnings for DB2 Personal Edition*
- "Installing DB2 Personal Edition - overview (Linux)" in the *Quick Beginnings for DB2 Personal Edition*

Related reference:

- "DB2 Application Development Client" on page 3

About this book

The *Application Development Guide* is a three-volume book that describes what you need to know about coding, debugging, building, and running DB2 applications:

- *Application Development Guide: Programming Client Applications* contains what you need to know to code standalone DB2 applications that run on DB2 clients. It includes information on:

  - Programming interfaces that are supported by DB2. High-level descriptions are provided for DB2 Developer's Edition, supported programming interfaces, facilities for creating Web applications, and DB2-provided programming features, such as routines and triggers.
  - The general structure that a DB2 application should follow. Recommendations are provided on how to maintain data values and relationships in the database, authorization considerations are described, and information is provided on how to test and debug your application.
  - Embedded SQL, both dynamic and static. The general considerations for embedded SQL are described, as well as the specific issues that apply to the usage of static and dynamic SQL in DB2 applications.
  - Supported host and interpreted languages, such as C/C++, COBOL, Perl, and REXX, and how to use embedded SQL in applications that are written in these languages.
  - The DB2 .NET Data Provider, and the OLE DB .NET and ODBC .NET data providers.
  - Java (both JDBC and SQLJ) and considerations for building Java applications for use on WebSphere Application Servers.
  - The IBM OLE DB Provider for DB2 Servers. General information is provided about IBM OLE DB Provider support for OLE DB services, components, and properties. Specific information is also provided about Visual Basic and Visual C++ applications that use the OLE DB interface for ActiveX Data Objects (ADO).
  - National language support issues. General topics, such as collating sequences, the derivation of code pages and locales, and character conversions are described. More specific issues such as DBCS code pages, EUC character sets, and issues that apply in Japanese and Traditional Chinese EUC and UCS-2 environments are also described.
  - Transaction management. Issues that apply to applications that perform multisite updates, and to applications that perform concurrent transactions, are described.
  - Applications in partitioned database environments. Directed DSS, local bypass, buffered inserts, and troubleshooting applications in partitioned database environments are described.
  - Commonly used application techniques. Information is provided on how to use generated and identity columns, declared temporary tables, and how to use savepoints to manage transactions.
  - The SQL statements that are supported for use in embedded SQL applications.
  - Applications that access host and iSeries environments. The issues that pertain to embedded SQL applications that access host and iSeries envirionments are described.
  - The simulation of EBCDIC binary collation.
- *Application Development Guide: Programming Server Applications* contains what you need to know about programming using server-side objects, including routines, large objects, user-defined types, and triggers. It includes information on:
  - Routines (stored procedures, user-defined functions, and methods), including:
    - Routine performance, security, library management considerations, and restrictions.
    - Creating routines, including external routines, and the CREATE statement.
    - Procedure parameter modes and parameter handling.

- Procedure result sets.
- SQL procedures including debugging and condition handling.
- User-defined scalar and table functions.
- User-defined scalar and table function calls (FIRST call, FINAL call,...) and scratchpads.
- Methods.
- Authorizations and binding of external routines.
- Language-specific considerations for C, Java, .NET common language runtime, and OLE automation routines.
- Invoking routines.
- Function selection.
- Passing distinct types and LOBs to functions.
- Code pages and routines.

– Large objects, including LOB usage and locators, reference variables, and CLOB data.
– User-defined distinct types, including strong typing, defining and dropping UDTs, creating tables with structured types, using distinct types and typed tables for specific applications, manipulating distinct types and casting between them, and performing comparisons and assignments with distinct types, including UNION operations on distinctly typed columns.
– User-defined structured types, including storing instances and instantiation, structured type hierarchies, defining structured type behavior, the dynamic dispatch of methods, the comparison, casting, and constructor functions, and mutator and observer methods for structured types.
– Typed tables, including creating, dropping, substituting, storing objects, defining system-generated object identifiers, and constraints on object identifier columns.
– Reference types, including relationships between objects in typed tables, semantic relationships with references, and referential integrity versus scoped references.
– Typed tables and typed views, including structured types as column types, transform functions and transform groups, host language program mappings, and structured type host variables.
– Triggers, including INSERT, UPDATE, and DELETE triggers, interactions with referential constraints, creation guidelines, granularity, activation time, transition variables and tables, triggered actions, multiple triggers, and synergy between triggers, constraints, and routines.

• *Application Development Guide: Building and Running Applications* contains what you need to know to build and run DB2 applications on the operating systems supported by DB2:
  – AIX
  – HP-UX
  – Linux
  – Solaris
  – Windows

  It includes information on:
  – DB2 supported servers and software to build applications, including supported compilers and interpreters.

- The DB2 sample program files, makefiles, build files, and error-checking utility files.
- How to set up your application development environment, including specific instructions for Java and WebSphere MQ functions.
- How to set up the sample database
- How to migrate your applications from previous versions of DB2.
- How to build and run Java applets, applications, and routines.
- How to build and run SQL procedures.
- How to build and run C/C++ applications and routines.
- How to build and run IBM and Micro Focus COBOL applications and routines.
- How to build and run REXX applications on AIX and Windows.
- How to build and run C# and Visual Basic .NET appllcations and CLR .NET routines on Windows.
- How to build and run applications with ActiveX Data Objects (ADO) using Visual Basic and Visual C++ on Windows.
- How to build and run applications with remote data objects using Visual C++ on Windows.

Part 1. The Application Development Environment

Chapter 1. DB2 environment support

This volume of the Application Development Guide describes DB2 support for application development. It provides the information you need to set up your environment for developing DB2 applications, and gives step-by-step instructions to compile, link, and run these applications in this environment. It explains how to build applications using the DB2 Application Development (DB2 AD) Client for DB2 Universal Database Version 8.2 on the following platforms:

- AIX
- HP-UX
- Linux
- Solaris Operating Environment
- Windows

DB2 Application Development Client

The DB2 Application Development (DB2 AD) Client provides the tools and environmental support to develop applications that access DB2 servers and application servers that implement the Distributed Relational Database Architecture (DRDA).

You can build and run DB2 applications with a DB2 AD Client installed. You can also run DB2 applications on these DB2 clients:

- DB2 Run-Time Client
- DB2 Administration Client

The DB2 AD Clients for supported platforms include the following:

- **Precompilers for C/C++, COBOL, and Fortran**, (providing the language is supported for that platform).
- **Embedded SQL application support**, including programming libraries, include files and code samples.
- **DB2 Call Level Interface (DB2 CLI) application support**, including programming libraries, include files, and code samples to develop applications which are easily ported to ODBC and compiled with an ODBC SDK. An ODBC SDK is available from Microsoft for Windows operating systems, and from various other vendors for many of the other supported platforms. For Windows operating systems, DB2 clients contain an ODBC driver that supports applications developed with the Microsoft ODBC Software Developer's Kit. For all other platforms, DB2 clients contain an optionally installed ODBC driver that supports applications that can be developed with an ODBC SDK for that platform, if one exists. Only DB2 Clients for Windows operating systems contain an ODBC driver manager.
- **DB2 Java Enablement**, which includes DB2 Java Database Connectivity (DB2 JDBC) support to develop Java applications and applets, and DB2 embedded SQL for Java (DB2 SQLJ) support to develop Java embedded SQL applications and applets.

- **Java Development Kit**, or equivalent, is shipped with DB2 for all supported operating systems. For specific version details, please see the ''supported development software' topic for your operating system:
  - "AIX supported development software" on page 8
  - "HP-UX supported development software" on page 10
  - "Linux supported development software" on page 12
  - "Solaris supported development software" on page 17
  - "Windows supported development software" on page 19
- **ActiveX Data Objects (ADO) and Object Linking and Embedding (OLE) automation routines (UDFs and Stored Procedures)** on Windows operating systems, including code samples implemented in Microsoft Visual Basic and Microsoft Visual C++. Also, code samples with Remote Data Objects (RDO) implemented in Microsoft Visual Basic.
- **Object Linking and Embedding Database (OLE DB) table functions** on Windows operating systems.
- **C# and Visual Basic .NET applications and CLR .NET routines** on Windows operating systems.
- **DB2 Development Center**, a graphical application that supports the rapid development of routines (stored procedures and user-defined functions), and structured types. The Development Center provides a single development environment that supports the entire DB2 family ranging from the workstation to z/OS. You can launch the Development Center as a stand-alone application or from a DB2 Universal Database center, such as the Control Center, the Command Editor, or the Task Center. The Development Center is implemented with Java, and all database connections are managed by using a Java Database Connectivity (JDBC) API. The Development Center also provides a DB2 Development Add-In for each of the following development environments:
  - Microsoft Visual C++, Version 6
  - Microsoft Visual Basic, Version 6
  - Microsoft Visual InterDev, Version 6
- **Interactive SQL** through the Command Editor or Command Line Processor (CLP) to prototype SQL statements or to perform ad hoc queries against the database.
- **A set of documented APIs** to enable other application development tools to implement precompiler support for DB2 directly within their products. For example, IBM COBOL on AIX uses this interface. Information on the set of Precompiler Services APIs is available from the PDF file, prepapi.pdf, at the DB2 application development Web site:

  `http://www.ibm.com/software/data/db2/udb/ad`
- **An SQL92 and MVS Conformance Flagger**, which identifies embedded SQL statements in applications that do not conform to the ISO/ANSI SQL92 Entry Level standard, or which are not supported by DB2 UDB for z/OS and OS/390. If you migrate applications developed on a workstation to another platform, the Flagger saves you time by showing syntax incompatibilities.

Related reference:

- "PRECOMPILE Command" in the *Command Reference*
- "AIX supported development software" on page 8
- "HP-UX supported development software" on page 10
- "Linux supported development software" on page 12
- "Solaris supported development software" on page 17

• "Windows supported development software" on page 19

Database manager instances

DB2® supports multiple database manager instances on the same machine. A database manager instance has its own configuration files, directories, and databases.

Each database manager instance can manage several databases. However, a given database belongs to only one instance. The following figure shows this relationship.

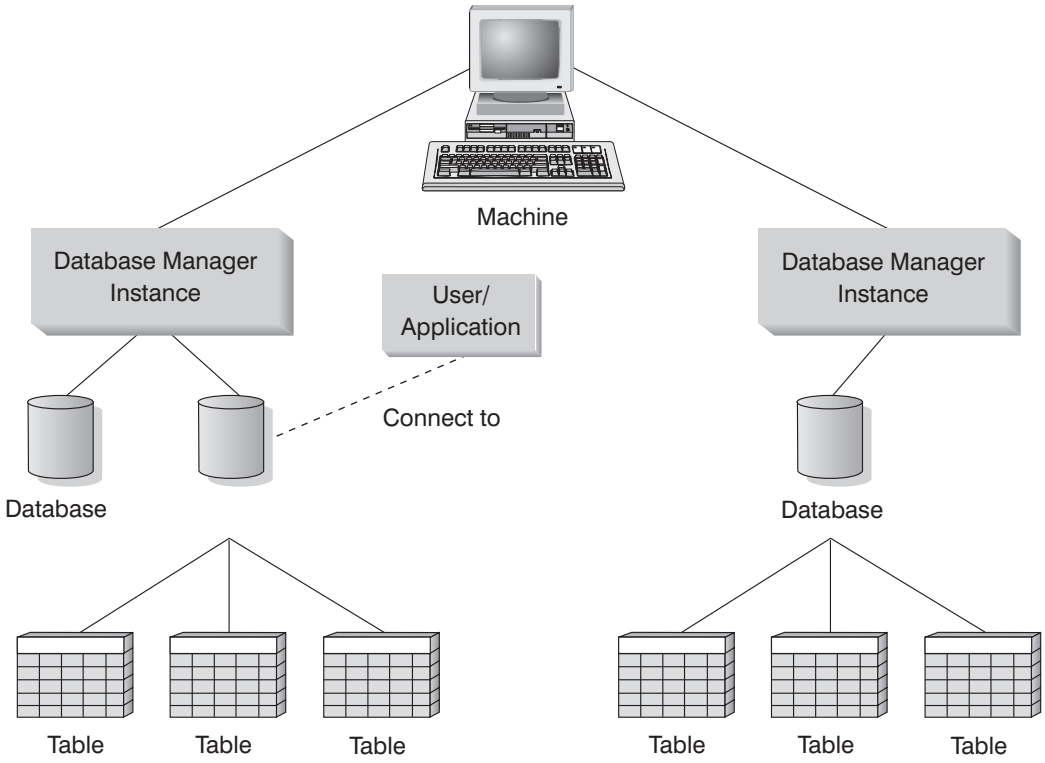

Figure 1. Database manager instances

Database manager instances give you the flexibility to have multiple database environments on the same machine. For example, you can have one database manager instance for development, and another instance for production.

With UNIX® Enterprise Server Edition (ESE) servers you can have different DB2 versions on different database manager instances. For example, you can have one database manager instance running DB2 Universal Database Version 7.1, and another running DB2 Universal Database Version 8.2. Prior to DB2 Version 8, within a version level only one release and FixPak level are supported. For example, DB2 Version 7.1 and DB2 Version 7.2 cannot coexist on a UNIX server. With DB2 Version 8, multiple FixPak levels can coexist on the same UNIX server, such as DB2 Version 8.1 and DB2 Version 8.2.

With Windows® servers, you must have the same DB2 version, release, and FixPak level on each database manager instance. You cannot have one database manager instance running DB2 Universal Database Version 7.1 and another instance running DB2 Universal Database Version 8.2.

You need to know the following for each instance you use:

instance name
For UNIX platforms, this is a valid user name that you specify when you create the database manager instance.

For Windows operating systems, this is an alphanumeric string of up to eight characters. An instance named ″DB2″ is created for you during install.

instance directory
The home directory where the instance is located.

For UNIX platforms, the instance directory is `$HOME/sqllib`, where $HOME is the home directory of the instance owner.

For Windows operating systems, the instance directory is `%DB2PATH%\`*instance_name*. The variable `%DB2PATH%` determines where DB2 is installed. The default installation value for `%DB2PATH%` is `\Program Files\IBM\SQLLIB`, so depending on which drive DB2 is installed, `%DB2PATH%` will point to *drive*`:\Program Files\IBM\SQLLIB`, unless the default value is changed.

The instance path on Windows servers is created based on either:

```
%DB2PATH%\%DB2INSTANCE%
```

(for example, `C:\Program Files\IBM\SQLLIB\DB2`)

or, if DB2INSTPROF is defined:

```
%DB2INSTPROF%\%DB2INSTANCE%
```

(for example, `C:\PROFILES\DB2`)

The DB2INSTPROF environment variable is used on Windows servers to support running DB2 on a network drive for which the client machine has only read access. In this case, DB2 will be set to point to *drive*`:\Program Files\IBM\SQLLIB`, and DB2INSTPROF will be set to point to a local path (for example, `C:\PROFILES`) which will contain all instance specific information such as catalogs and configurations, since DB2 requires update access to these files.

Related concepts:
- “Multiple instances of the database manager” in the *Administration Guide: Implementation*
- “Multiple instances on a UNIX operating system” in the *Administration Guide: Implementation*
- “Multiple instances on a Windows operating system” in the *Administration Guide: Implementation*

Related tasks:
- “Updating the database manager configuration file” on page 25
- “Setting the current instance” in the *Administration Guide: Implementation*
- “Creating additional instances” in the *Administration Guide: Implementation*

Related reference:

- "GET DATABASE MANAGER CONFIGURATION Command" in the *Command Reference*
- "UPDATE DATABASE MANAGER CONFIGURATION Command" in the *Command Reference*

DB2 supported servers

You use the DB2 AD client to develop applications that will run on a specific operating system. However, your applications can access remote databases on the following operating system servers:

- DB2 for AIX
- DB2 for HP-UX
- DB2 for Linux
- DB2 for OS/2
- DB2 for NUMA-Q
- DB2 for Solaris
- DB2 for Windows NT
- DB2 for Windows 2000
- DB2 for Windows XP
- DB2 for Windows Server 2003
- Distributed Relational Database Architecture (DRDA)-compliant application servers, such as:
  - DB2 for z/OS and OS/390
  - DB2 for AS/400 and iSeries
  - DB2 Server for VSE & VM (formerly SQL/DS for VM and VSE)
  - DRDA-compliant application servers from database vendors other than IBM.

Notes:

1. DB2 Version 8 HP-UX 64-bit servers do not support running DB2 Version 7 64-bit local applications.
2. DB2 for OS/2 is not available for DB2 Version 8
3. DB2 for NUMA-Q runs on the PTX operating system, and is only available for DB2 Version 7
4. DB2 Version 8 Windows 64-bit servers support connections from DB2 Version 6 and Version 7 32-bit clients only for SQL requests. Connections from Version 7 64-bit clients are not supported.

Related tasks:

- "Configuring remote access to a server database" in the *Installation and Configuration Supplement*

Related reference:

- "DB2 supported development software" on page 7

DB2 supported development software

DB2 Version 8 supports compilers, interpreters, and related software for the following operating systems:

- AIX
- HP-UX
- Linux
- Solaris Operating Environment

- Windows

DB2 supports 32-bit and 64-bit versions of each of these operating systems. There are differences for building applications in 32-bit and 64-bit environments in most cases on these operating systems. DB2 does not support running COBOL applications or routines (stored procedures and user-defined functions) in 64-bit operating system environments. For other languages, DB2 supports running 32-bit applications and routines on all supported 64-bit operating system environments except Linux IA64 and Linux zSeries.

With the exception of IBM COBOL on Windows, the compiler information given for each of these operating systems assumes that you are using the DB2 precompiler for that operating system, and not the precompiler support that might be built into one of the listed compilers. For IBM COBOL on Windows, DB2 supports the IBM COBOL precompiler as well as the DB2 precompiler.

For the latest DB2 compiler information and related software updates, visit the DB2 application development Web site at:

`http://www.ibm.com/software/data/db2/udb/ad`

Note the following points about software support:

- **Fortran and REXX.** DB2 will not enhance features for Fortran and REXX beyond the level of support for these languages in DB2 Universal Database Version 5.2.
- **Perl**. At the time of printing, Release 0.76 of the DB2 UDB driver (DBD::DB2) for the Perl Database Interface (Perl DBI) Version 0.93 or later is available for AIX, HP-UX, Linux, Solaris and Windows. The latest driver can be downloaded from:

  `http://www.ibm.com/software/data/db2/perl`
- **PHP**. PHP can be used as a method to access DB2 from web-based applications or the command line, and is available for AIX, HP-UX, Linux, Solaris and Windows. At the time of printing, the latest version is PHP 4.3.4. You can download the latest version of PHP from:

  `http://www.php.net`

Related reference:

- "AIX supported development software" on page 8
- "HP-UX supported development software" on page 10
- "Linux supported development software" on page 12
- "Solaris supported development software" on page 17
- "Windows supported development software" on page 19

AIX supported development software

DB2 for AIX supports the following operating systems:

AIX Version 4.3.3 (32-bit only)
with Maintenance level 11

Plus the March 2003 C++ Runtime PTF (see the C++ section below for the download link), and:

For JFS filesystems:
APAR IY49385

For Java:

- OpenGL.OpenGL_X.rte.base

- OpenGL.OpenGL_X.rte.soft
- X11.adt.lib

AIX Version 5.1.0 (32-bit and 64-bit)
with Maintenance level 5

Plus the March 2003 C++ Runtime PTF (see the C++ section below for the download link), and:

For JFS filesystems:
APAR IY48735

For JFS2 filesystems:
APAR IY49254

For Java:
Recommended Maintenance Package AIX 5100-04 and APAR IY46667

For running more than 1000 db2agents:
APAR IY49220, and specify ″vmtune -T 0″ before db2start or in AIX bootup

AIX Version 5.2.0 (32-bit and 64-bit)
with Maintenance level 2, and:

For Concurrent I/O (CIO) and Direct I/O (DIO) mounted volume:
APARs IY49129 and IY49346

For JFS filesystems:
APAR IY48339

For JFS2 filesystems:
APAR IY49304

For Java:
Recommended Maintenance Package AIX 5200-01 and APAR IY46668

For running more than 1000 db2agents and using the 32-bit AIX kernel:
APAR IY49885, and specify ″vmo -o pta_balance_threshold=0″ before db2start or in AIX bootup

Note: You can query your system to see if a particular APAR is installed with the following command:

```
instfix -v -i -k <APAR>
```

For example: `instfix -v -i -k IY31254`

DB2 for AIX supports the following programming languages and compilers:

C IBM C for AIX Version 5.0.2.3

IBM C for AIX Version 6.0

C++ IBM VisualAge C++ Version 5.0.2.3 with the March 2003 C++ Runtime PTF:
`http://www-1.ibm.com/support/docview.wss?rs=0&q=xlC.rte&uid=swg24004427`

IBM VisualAge C++ Version 6.0 with the March 2003 C++ Runtime PTF:
`http://www-1.ibm.com/support/docview.wss?rs=0&q=xlC.rte&uid=swg24004427`

COBOL
IBM COBOL Set for AIX Version 1.1

Micro Focus COBOL Server Express Version 2.2 with Service Pack 1

Note: COBOL support is for 32-bit only.

Fortran
IBM XL Fortran for AIX Versions 4.1 (32-bit only) and 5.1.0 (for 32-bit and 64-bit)

Java Java Developer Kit Version 1.3.1 and Java Runtime Environment Version 1.3.1 for AIX from IBM

IBM Developer Kit for AIX, Java Technology Edition, Version 1.4.1 Service Release 1 (for AIX 5.1 and 5.2 only)

Note: DB2 will install the latest supported version of the developer kit if it is not already installed, unless the DB2 installation is an update of a previous DB2 Version 8 installation. If a previous DB2 Version 8 installation is being updated, the supported developer kit must be manually installed from the CD-ROM.

Perl Perl 5.004_04 or later, DBI 0.93 or later

PHP PHP 4.3.4 or later

REXX IBM AIX REXX/6000 AISPO Product Number: 5764-057

IBM Object REXX for AIX Version 1.1

REXXSAA 4.00

Note: REXX support is for 32-bit only.

For DB2 for AIX software support updates visit the DB2 application development Web site:

```
http://www.ibm.com/software/data/db2/udb/ad
```

Related reference:

- "DB2 supported development software" on page 7
- "AIX C application compile and link options" on page 168
- "AIX C routine compile and link options" on page 170
- "AIX C++ application compile and link options" on page 173
- "AIX C++ routine compile and link options" on page 174
- "AIX IBM COBOL application compile and link options" on page 183
- "AIX IBM COBOL routine compile and link options" on page 186
- "AIX Micro Focus COBOL application compile and link options" on page 188
- "AIX Micro Focus COBOL routine compile and link options" on page 190
- "Installation requirements for DB2 servers (AIX)" in the *Quick Beginnings for DB2 Servers*

HP-UX supported development software

DB2 for HP-UX supports the following operating systems:

HP-UX on PA-RISC
HP-UX Version 11i(11.11) for PA-RISC 2 with:

June 2003 GOLDBASE11i and June 2003 GOLDAPPS11i bundles and patches PHSS_26560, PHKL_28489, PHCO_27434 and PHCO_29960

HP-UX on IA64
HP-UX Version 11i Version 2 (B.11.23) for Intel Itanium with patch PHKL_30065

Note: Attempting to run DB2 applications or routines for HP-UX on IA64 that were built for PA-RISC under the aries binary translator is not supported.

DB2 for HP-UX on PA-RISC supports the following programming languages and compilers:

C HP C Compiler version B.11.11.02

C++ HP aC++ Version A.03.52

COBOL
Micro Focus COBOL Server Express Version 2.2 with Service Pack 1

Note: COBOL support is for 32-bit only.

Fortran
HP-UX f90 B.11.01.06

Java HP-UX 32-bit: Software Developer's Kit and Runtime Environment 1.3.1 and 1.4.2.01 for HP-UX 11.0 and 11i PA-RISC from Hewlett-Packard

HP-UX 64-bit: Software Developer's Kit and Runtime Environment 1.4.2.01 for HP-UX 11.0 and 11i PA-RISC from Hewlett-Packard

Notes:

1. 32-bit Java routines are supported for Software Developer's Kit 1.3.1 only.
2. DB2 will install the latest supported version of the developer kit if it is not already installed, unless the DB2 installation is an update of a previous DB2 Version 8 installation. If a previous DB2 Version 8 installation is being updated, the developer kit must be manually installed from the CD-ROM.

Perl Perl 5.004_04 or later, DBI 0.93 or later

PHP PHP 4.3.4 or later

DB2 for HP-UX on IA64 supports the following programming languages and compilers:

C HP C Compiler version A.05.52

C++ HP (aCC) aC++ Version A.05.52

Note: DB2 for HP-UX on IA64 does not support any C++ application or third party C++ library that is built with the deprecated option `-AP` or has a dependency on `libstd` (instead of `libstd_v2`).

Fortran
HP-UX F90 B.11.23

Notes:

1. To generate 64-bit code you need the `+DD64` compile option.
2. To generate 32-bit code you need the `+DD32` compile option.

Java Software Developer's Kit and Runtime Environment 1.4.2.01 or greater for HP-UX 11 PA-RISC and Itanium-based systems

Note: DB2 will install the supported version of the developer kit if it is not already installed, unless the DB2 installation is an update of a previous DB2 Version 8 installation. If a previous DB2 Version 8 installation is being updated, the developer kit must be manually installed from the CD-ROM into `/opt/java1.4`.

Perl Perl 5.004_04 or later, DBI 0.93 or later

PHP PHP 4.3.4 or later

For DB2 for HP-UX software support updates visit the DB2 application development Web site:

```
http://www.ibm.com/software/data/db2/udb/ad
```

Related reference:

- "DB2 supported development software" on page 7
- "HP-UX C application compile and link options" on page 194
- "HP-UX C routine compile and link options" on page 196
- "HP-UX C++ application compile and link options" on page 199
- "HP-UX C++ routine compile and link options" on page 201
- "HP-UX Micro Focus COBOL application compile and link options" on page 204
- "HP-UX Micro Focus COBOL routine compile and link options" on page 206
- "Installation requirements for DB2 servers (HP-UX)" in the *Quick Beginnings for DB2 Servers*

Linux supported development software

DB2 for Linux supports the following operating system architectures:

- **Linux on Intel x86 (32-bit)**
- **Linux on s/390 and zSeries**
- **Linux on AMD64**
- **Linux on PowerPC**
- **Linux on IA64**

For information on supported distributions, and kernel and library levels for each of these architectures, please visit:

```
http://www.ibm.com/db2/linux/validate
```

The following tables describe the DB2 Linux architecture support at the time of printing. You are strongly urged to check the validate Website (above) for updates to this support:

Table 1. Linux on Intel x86 (32-bit)

Distributions	Kernel	Library	Comments
Conectiva Linux Enterprise Edition (CLEE)	2.4.19	glibc 2.2.5	Powered by United Linux 1.0
LINX Rocky Secure Server 2.1	2.4.21	glibc 2.2.5	
Red Flag Advanced Server 4.0	2.4.21-as.2	glibc 2.2.93-5	
Red Flag Function Server 4.0	2.4.20-8smp	glibc 2.2.93-5	

Table 1. Linux on Intel x86 (32-bit) (continued)

Distributions	Kernel	Library	Comments
Red Hat Enterprise Linux 2.1 AS/ES/WS	2.4.9-e16	glibc 2.2.4	
Red Hat Enterprise Linux (RHEL) 3 AS/ES/WS	2.4.21-7.EL	glibc-2.3.2-95.3	
SCO Linux 4.0	2.4.19	glibc 2.2.5	Powered by United Linux 1.0
SuSE Pro 8.0	2.4.18	glibc 2.2.5	
SuSE Pro 8.1	2.4.19	glibc 2.2.5	
SuSE Linux Enterprise Server (SLES) 7	2.4.7	glibc 2.2.2	
SuSE Linux Enterprise Server (SLES) 8	2.4.19	glibc 2.2.5	Validated up to SuSE Service Pack 2 level
Turbolinux 7 Server	2.4.9	glibc 2.2.4	
Turbolinux 8 Server	2.4.18-5	glibc 2.2.5	
Turbolinux Enterprise Server 8	2.4.19	glibc 2.2.5	
United Linux 1.0	2.4.19	glibc 2.2.5	

Table 2. Linux on Intel x86 (32-bit) non-enterprise distributions (no longer supported by the vendor)

Distributions	Kernel	Library	Comments
Red Hat 7.2	2.4.9-34	glibc 2.2.4	
Red Hat 7.3	2.4.18	glibc 2.2.5	
Red Hat 8.0	2.4.18-14	glibc 2.2.93-5	
SuSE 7.3	2.4.10	glibc 2.2.4	

Table 3. Linux on s/390 and zSeries (31-bit kernel version supported on s/390; 64-bit on zSeries)

Distributions	Kernel	Library	Comments
Red Hat 7.2	2.4.9-38	glibc 2.2.4	
SuSE Linux Enterprise Server (SLES) 7	2.4.7-58	glibc 2.2.4	compat.rpm contains libstdc++ 6.1. Use JDK 1.3.1 SR 1 for Java
SuSE Linux Enterprise Server (SLES) 8	2.4.19	glibc 2.2.5	Powered by United Linux 1.0
Turbo Linux Enterprise Server (TLES) 8	2.4.19	glibc 2.2.5	Powered by United Linux 1.0
United Linux 1.0	2.4.19	glibc 2.2.5	

Table 4. Linux on AMD64

Distributions	Kernel	Library	Comments
Red Hat Enterprise Linux (RHEL) 3 AS/ES/WS	2.4.21-7.EL	glibc-2.3.2-95.3	

Table 4. Linux on AMD64 (continued)

Distributions	Kernel	Library	Comments
SuSE Linux Enterprise Server (SLES) 8.0	2.4.19-SMP	glibc 2.2.5-16	

Table 5. Linux on PowerPC (iSeries and pSeries)

Distributions	Kernel	Library	Comments
Red Hat Enterprise Linux (RHEL) 3 AS	2.4.21-7.EL	glibc-2.3.2-95.3	
SuSE Enterprise Server (SLES) 8	2.4.19-16	glibc 2.2.5	Powered by United Linux 1.0
Turbolinux Enterprise Server 8	2.4.19-16	glibc 2.2.5	Powered by United Linux 1.0
United Linux 1.0	2.4.19	glibc 2.2.5	

Table 6. Linux on IA64

Distributions	Kernel	Library	Comments
Red Hat Enterprise Linux 2.1 AS/ES/WS	2.4.18-e.12smp	glibc	
Red Hat Enterprise Linux (RHEL) 3 AS/ES/WS	2.4.21-7.EL	glibc-2.3.2-95.3	
SuSE Linux Enterprise Server (SLES) 8	2.4.19-SMP	glibc 2.2.5	Powered by United Linux 1.0
United Linux 1.0	2.4.19	glibc 2.2.5	

DB2 for Linux for Intel x86 supports the following programming languages and compilers:

C GNU/Linux gcc versions 2.95.3 and 2.96

C++ GNU/Linux g++ versions 2.95.3 and 2.96

COBOL
Micro Focus COBOL Server Express Version 2.2 with Service Pack 1

Java IBM Developer Kit and Runtime Environment for Linux, Java 2 Technology Edition, Version 1.3.1 and 1.4.1 Service Release 1, 32-bit version

Note: DB2 will install the latest supported version of the developer kit if it is not already installed, unless the DB2 installation is an update of a previous DB2 Version 8 installation. If a previous DB2 Version 8 installation is being updated, the supported developer kit must be manually installed from the CD-ROM.

Perl Perl 5.004_04 or later, DBI 0.93 or later

PHP PHP 4.3.4 or later

REXX Object REXX Interpreter for Linux Version 2.1

A 32-bit instance on **DB2 for Linux on s/390** or **DB2 for Linux on zSeries** supports the following programming languages and compilers:

C GNU/Linux gcc version 2.95.3

C++ GNU/Linux g++ version 2.95.3

COBOL
Micro Focus COBOL Server Express Version 2.2 with Service Pack 1

Java IBM zSeries Developer Kit for Linux, Java 2 Technology Edition (at the Sun 1.3.1 and 1.4.1 Service Release 1 SDK level)

Note: DB2 will install the latest supported version of the developer kit if it is not already installed, unless the DB2 installation is an update of a previous DB2 Version 8 installation. If a previous DB2 Version 8 installation is being updated, the supported developer kit must be manually installed from the CD-ROM.

Perl Perl 5.004_04 or later, DBI 0.93 or later

PHP PHP 4.3.4 or later

REXX Object REXX 2.2.0 for Linux/390

A 64-bit instance on **DB2 for Linux on zSeries** supports the following programming languages and compilers:

C GNU/Linux gcc version 3.2

C++ GNU/Linux g++ version 3.2

Java IBM zSeries Developer Kit for Linux, Java 2 Technology Edition (at the Sun 1.4.1 Service Release 1 SDK level)

Note: DB2 will install the latest supported version of the developer kit if it is not already installed, unless the DB2 installation is an update of a previous DB2 Version 8 installation. If a previous DB2 Version 8 installation is being updated, the supported developer kit must be manually installed from the CD-ROM.

Perl Perl 5.8

A 32-bit instance on **DB2 for Linux on AMD64** supports the following programming languages and compilers:

C GNU/Linux gcc versions 3.2 and 3.3

Note: The ″-m32″ compiler option must be used to generate 32-bit applications or routines (stored procedures and user-defined functions).

C++ GNU/Linux g++ versions 3.2 and 3.3

Notes:

1. These versions of the GNU/Linux g++ compiler do not accept integer parameters for some fstream functions. Consult the compiler documentation for more information.
2. The ″-m32″ compiler option must be used to generate 32-bit applications or routines (stored procedures and user-defined functions).

Java IBM Developer Kit and Runtime Environment for Linux x86, Java 2 Technology Edition, Version 1.3.1 Service Release 4, 32-bit version, and Version 1.4.1 Service Release 1, 32-bit version.

Notes:

1. DB2 will install the latest supported version of the developer kit if it is not already installed, unless the DB2 installation is an update of a previous DB2 Version 8 installation. If a previous DB2 Version 8 installation is being updated, the supported developer kit must be manually installed from the CD-ROM.
2. If you have installed the developer kit 1.3.1 Service Release supplied by SuSE SLES 8, you must uninstall it before installing DB2, otherwise DB2 will not be able to install the recommended developer kit. If DB2 has been installed while the developer kit supplied by SuSE SLES 8 was not uninstalled, please see the instructions in the README to update the developer kit manually.

Perl Perl 5.8

PHP PHP 4.3.4 or later

A 64-bit instance on **DB2 for Linux on AMD64** supports the following programming languages and compilers:

C GNU/Linux gcc versions 3.2 and 3.3

C++ GNU/Linux g++ versions 3.2 and 3.3

Note: These versions of the GNU/Linux g++ compiler do not accept integer parameters for some fstream functions. Consult the compiler documentation for more information.

Java DB2 does not currently support any 64-bit Java Developer Kit for Linux on AMD64.

Perl Perl 5.8

PHP PHP 4.3.4 or later

DB2 for Linux on PowerPC supports the following programming languages and compilers:

C GNU/Linux gcc version 3.2

C++ GNU/Linux g++ version 3.2

Note: This version of the GNU/Linux g++ compiler does not accept integer parameters for some fstream functions. Consult the compiler documentation for more information.

Java For a 32-bit instance: IBM Developer Kit and Runtime Environment for Linux Java 2 Technology Edition, Versions 1.3.1 and 1.4.1 Service Release 1, 32-bit version for PowerPC.

For a 64-bit instance: IBM Developer Kit and Runtime Environment for Linux Java 2 Technology Edition, Version 1.4.1 Service Release 1, 64-bit version for PowerPC.

Note: DB2 will install the latest supported version of the developer kit if it is not already installed, unless the DB2 installation is an update of a previous DB2 Version 8 installation. If a previous DB2 Version 8 installation is being updated, the supported developer kit must be manually installed from the CD-ROM.

Perl Perl 5.8

PHP PHP 4.3.4 or later

DB2 for Linux on IA64 supports the following programming languages and compilers:

C GNU/Linux gcc version 3.2

C++ GNU/Linux g++ version 3.2

Note: This version of the GNU/Linux g++ compiler does not accept integer parameters for some fstream functions. Consult the compiler documentation for more information.

Java IBM Developer Kit and Runtime Environment for Linux, Java 2 Technology Edition, Version 1.3.1, 64-bit version. To use this Developer Kit, you must also have installed gcc 3.2 and the gcc3 libstdc++ runtime libraries.

Note: DB2 will install the the developer kit if it is not already installed, unless the DB2 installation is an update of a previous DB2 Version 8 installation. If a previous DB2 Version 8 installation is being updated, the developer kit must be manually installed from the CD-ROM.

Perl Perl 5.8

Note: Running DB2 32-bit applications or routines (stored procedures and user-defined functions) is not supported on Linux IA64.

For DB2 for Linux software support updates visit the DB2 application development Web site:

```
http://www.ibm.com/software/data/db2/udb/ad
```

Related reference:

- "DB2 supported development software" on page 7
- "Linux C application compile and link options" on page 208
- "Linux C routine compile and link options" on page 210
- "Linux C++ application compile and link options" on page 213
- "Linux C++ routine compile and link options" on page 215
- "Installation requirements for DB2 Personal Edition (Linux)" in the *Quick Beginnings for DB2 Personal Edition*
- "Installation requirements for DB2 servers (Linux)" in the *Quick Beginnings for DB2 Servers*
- "Linux Micro Focus COBOL application compile and link options" on page 219
- "Linux Micro Focus COBOL routine compile and link options" on page 220

Solaris supported development software

DB2 for Solaris supports the following operating system:

Solaris

DB2 Workgroup Server Edition is supported on the following Solaris Operating Environment versions:

Solaris 7 (32-bit) with "Recommended & Security Patches" + 107226-17 + 107153-01 + 106327-10

Solaris 8 (32-bit) with ″Recommended & Security Patches″ + 108921-12 + 108940-24 + 108434-03 + 108528-12

Solaris 9 (32-bit)

DB2 Enterprise Server Edition is supported on the following Solaris Operating Environment versions:

Solaris 7 (32-bit) with ″Recommended & Security Patches″ + 107226-17 + 107153-01 + 106327-10

Solaris 7 (64-bit) with ″Recommended & Security Patches″ + 107226-17 + 107153-01 + 106300-11 + 106327-10

Solaris 8 (32-bit) with ″Recommended & Security Patches″ + 108921-12 + 108940-24 + 108434-03 + 108528-12

Solaris 8 (64-bit) with ″Recommended & Security Patches″ + 108921-12 + 108940-24 + 108435-03 + 108434-03 + 108528-12

Solaris 9 (32-bit)

Solaris 9 (64-bit)

″Recommended & Security Patches″ can be obtained from:

```
http://sunsolve.sun.com
```

From the SunSolve Online Web site, click on the ″Patches″ menu item in the left panel and select ″Recommended & Security Patches″ from the ″Downloads″ section.

The J2SE Solaris Patch Clusters are also required. They can be obtained from the `http://sunsolve.sun.com` Web site. From the SunSolve Online Web site, click on the ″Patches″ menu item in the left panel and select ″Recommended Patch Clusters″ from the ″Browse & Download Patches″ section.

The SUNWlibC software is required to install DB2 on Solaris.

For DB2 on 64-bit Fujitsu PRIMEPOWER systems, you also require the following:

- Solaris 8 Kernel Update Patch 108528-16 or later to get the fix for patch 912040-01
- Solaris 9 Kernel Update Patch 112233-01 or later to get the fix for patch 912041-01

The Fujitsu PRIMEPOWER patches for Solaris can be downloaded from FTSI at this link:

```
https://download.ftsi.fujitsu.com/
```

DB2 for Solaris supports the following programming languages and compilers:

C Forte C Versions 5.0, 6, 6.1, and 6.2

Sun ONE Studio 7 and 8, Compiler Collection

C++ Forte C++ Versions 5.0, 6, 6.1, and 6.2

Sun ONE Studio 7 and 8, Compiler Collection

COBOL

Micro Focus COBOL Server Express Version 2.2 with Service Pack 1

Note: COBOL support is for 32-bit only.

Fortran
SPARCompiler Fortran Versions 4.2 and 5.0

Java Solaris 32-bit: Java Development Kit (JDK) Versions 1.3.1 and 1.4.2 for Solaris from Sun Microsystems

Solaris 64-bit: Java Development Kit (JDK) Version 1.4.2 for Solaris from Sun Microsystems

Note: DB2 will install the latest supported version of the JDK if it is not already installed, unless the DB2 installation is an update of a previous DB2 Version 8 installation. If a previous DB2 Version 8 installation is being updated, the JDK must be manually installed from the CD-ROM.

Perl Perl 5.004_04 or later, DBI 0.93 or later

PHP PHP 4.3.4 or later

For DB2 for Solaris software support updates visit the DB2 application development Web site:

```
http://www.ibm.com/software/data/db2/udb/ad
```

Related reference:

- "DB2 supported development software" on page 7
- "Solaris C application compile and link options" on page 222
- "Solaris C routine compile and link options" on page 224
- "Solaris C++ application compile and link options" on page 227
- "Solaris C++ routine compile and link options" on page 229
- "Solaris Micro Focus COBOL application compile and link options" on page 232
- "Solaris Micro Focus COBOL routine compile and link options" on page 234
- "Installation requirements for DB2 servers (Solaris Operating Environment)" in the *Quick Beginnings for DB2 Servers*

Windows supported development software

DB2 for Windows 32-bit operating systems supports the following:

Microsoft Windows XP

Microsoft Windows Server 2003

Microsoft Windows 2000
Service Pack 2 is required for Windows Terminal Server.

Microsoft Windows NT
Version 4.0 with Service Pack 6a or later.

Microsoft Windows ME

Microsoft Windows 98

DB2 for Windows 32-bit operating systems supports the following programming languages:

Basic Microsoft Visual Basic 6.0 Professional Edition

Microsoft Visual Basic .NET 7.0 and 7.1 for Microsoft .NET Framework versions 1.0 and 1.1 respectively

Note: .NET Framework must be installed before using the DB2 Install program to install the DB2 .NET Data Provider.

C# Microsoft Visual C# .NET Compiler versions 7.0 and 7.1 for Microsoft .NET Framework versions 1.0 and 1.1 respectively

Note: .NET Framework must be installed before using the DB2 Install program to install the DB2 .NET Data Provider.

C/C++ Microsoft Visual C++ Version 6.0

Microsoft Visual C++ .NET 2002 and 2003

Intel C++ Compiler for 32-bit applications Version 6 or later

COBOL

Micro Focus COBOL Version 4.0.20

Micro Focus COBOL Net Express Version 3.1.0

IBM VisualAge COBOL Version 3.0.4 or later

Java IBM Developer Kit and Runtime Environment for Windows, Java 2 Technology Edition, Versions 1.3.1 and 1.4.1 Service Release 1, 32-bit version

Note: DB2 will install the latest supported version of the developer kit if it is not already installed.

Java Development Kit (JDK) 1.3.1 for Win32 from Sun Microsystems

Perl Perl 5.004_04, DBI 0.93

PHP PHP 4.3.4 or later

REXX IBM Object REXX for Windows NT/95 Version 1.1

For information on obtaining IBM Object REXX for Windows, visit:

```
http://www.ibm.com/software/ad/obj-rexx/
```

Microsoft Windows Scripting Host

Version 5.1

Note: For Windows 2000 and Windows XP: to run DB2 applications that have COM+ objects using ODBC or applications that use the OLE DB Provider for ODBC with OLE DB resource pooling disabled, you must use Windows 2000 service pack 3 or Windows XP service pack 1. If you are not sure your application environment qualifies, you are recommended to install the appropriate Windows service level. For more information, refer to the following Microsoft Knowledge Base article:

```
http://support.microsoft.com/default.aspx?scid=KB;EN-US;306414
```

These service packs are not required for DB2 server itself or for applications shipped as part of DB2 products.

DB2 for Windows 64-bit operating systems supports the following:

Microsoft Windows XP 64-bit Edition

Microsoft Windows Server 2003

DB2 for Windows 64-bit operating systems supports the following programming languages:

C/C++ Intel C++ Compiler for Itanium Version 7.1 with Microsoft Software Developer's Kit 3790 or later.

Microsoft's C/C++ compiler for the Intel Itanium architecture (available with Microsoft Software Developer's Kit 3790)

Java IBM Developer Kit and Runtime Environment for Windows, Java 2 Technology Edition, Version 1.4.1 Service Release 1, 64-bit version

Note: DB2 will install the developer kit if it is not already installed.

Perl Perl 5.004_04, DBI 0.93

PHP PHP 4.3.4 or later

Microsoft Windows Scripting Host
Version 5.1

Note: Windows Server 2003 support includes the following:
- Windows Server 2003, Standard Edition
- Windows Server 2003, Enterprise Edition
- Windows Server 2003, Datacenter Edition

For DB2 for Windows software support updates visit the DB2 application development Web site:

```
http://www.ibm.com/software/data/db2/udb/ad
```

Related reference:

- "DB2 supported development software" on page 7
- "Installation requirements for DB2 servers (Windows)" in the *Quick Beginnings for DB2 Servers*
- "Windows C/C++ application compile and link options" on page 262
- "Windows C/C++ routine compile and link options" on page 266
- "Windows IBM COBOL application compile and link options" on page 272
- "Windows IBM COBOL routine compile and link options" on page 276
- "Windows Micro Focus COBOL application compile and link options" on page 279
- "Windows Micro Focus COBOL routine compile and link options" on page 282
- "Installation requirements for DB2 Personal Edition (Windows)" in the *Quick Beginnings for DB2 Personal Edition*
- "Visual Basic .NET application compile and link options" on page 250
- "C# .NET application compile and link options" on page 247
- "CLR .NET routine compile and link options" on page 255

Chapter 2. Setup

General setup information

For DB2 CLI setup information, see the *CLI Guide and Reference*.

Setting up the application development environment

In order to build and run DB2 applications, you must use a compiler or interpreter for one of the supported programming languages for your operating system, unless you use command line processor (CLP) scripting or SQL procedures (see below). You have to set up the DB2 environment and configure it for your development requirements. There are certain procedures to follow in order to migrate DB2 applications from a previous version of DB2. Also, you might want to create the DB2 sample database for testing purposes.

Prerequisites:

Ensure the environment for the DB2-supported compiler or interpreter you plan to use is correctly set up by first building a non-DB2 application. Then, if you encounter any problems, please see the documentation that comes with your compiler or interpreter.

Install the Application Development client on the client or server workstation you are using. If you are developing applications from a remote client, ensure your client machine can reach the machine on which the DB2 database server resides. Also ensure your client can successfully connect to the database. You can use the command line processor (CLP) or client configuration assistant (CCA) to test connectivity.

Procedure:

To set up your application development environment:

1. Unless the defaults are acceptable, follow the instructions in "Updating the database manager configuration file" on page 25
2. If you will be programming with DB2 CLI, Java, or WebSphere MQ functions, you have to configure your environment. Before you perform any platform-specific changes, follow the instructions in the following:
   - Setting up the CLI environment
   - "Setting up the Java environment" on page 26
   - "Setting up DB2 WebSphere MQ functions" on page 28
3. Configure your operating system environment with the instructions in the following:
   - "Setting up the UNIX application development environment" on page 31
   - "Setting up the Windows application development environment" on page 39
4. Optional: "Setting up the sample database" on page 44

SQL procedures

Beginning with DB2 Version 8.2, the creation of SQL procedures does not require a C or C++ compiler on the server, so no C or C++ compiler setup is required. When an SQL procedure is created, its procedural statements are converted to a native representation that is stored in the database catalogs, as is done with other SQL statements. When an SQL procedure is called, this representation is loaded from the catalogs and is executed by the DB2 engine.

Related concepts:

- "Database manager instances" on page 5
- "Migrating applications to DB2 Version 8" on page 49

Related tasks:

- "Updating the database manager configuration file" on page 25
- "Setting up the CLI environment" in the *CLI Guide and Reference, Volume 1*
- "Setting up the Java environment" on page 26
- "Setting up DB2 WebSphere MQ functions" on page 28
- "Setting up the UNIX application development environment" on page 31
- "Setting up the Windows application development environment" on page 39
- "Setting up the sample database" on page 44

Related reference:

- "DB2 Application Development Client" on page 3
- "DB2 supported servers" on page 7
- "DB2 supported development software" on page 7
- "AIX supported development software" on page 8
- "HP-UX supported development software" on page 10
- "Linux supported development software" on page 12
- "Solaris supported development software" on page 17
- "Windows supported development software" on page 19

Rebuilding DB2 routine shared libraries

DB2® will cache the shared libraries used for stored procedures and user-defined functions once loaded. If you are developing a routine, you might want to test loading the same shared library a number of times, and this caching can prevent you from picking up the latest version of a shared library. The way to avoid caching problems depends on the type of routine:

1. **Fenced, not threadsafe routines.** The database manager configuration keyword KEEPFENCED has a default value of YES. This keeps the fenced mode process alive. This default setting can interfere with reloading the library. It is best to change the value of this keyword to NO while developing fenced, not threadsafe routines, and then change it back to YES when you are ready to load the final version of your shared library. For more information, see "Updating the database manager configuration file" on page 25.
2. **Trusted or threadsafe routines.** Except for SQL routines (including SQL procedures), the only way to ensure that an updated version of a DB2 routine library is picked up when that library is used for trusted, or threadsafe routines, is to recycle the DB2 instance by entering db2stop followed by db2start on the command line. This is not needed for an SQL routine because when it is recreated, the compiler uses a new unique library name to prevent possible conflicts.

For routines other than SQL routines, you can also avoid caching problems by creating the new version of the routine with a differently named library (for example foo.a becomes foo.1.a), and then using either the ALTER PROCEDURE or ALTER FUNCTION SQL statement with the new library.

Related tasks:

- "Updating the database manager configuration file" on page 25

Related reference:

- "ALTER FUNCTION statement" in the *SQL Reference, Volume 2*
- "ALTER PROCEDURE statement" in the *SQL Reference, Volume 2*

Updating the database manager configuration file

This file contains important settings for application development.

The keyword `KEEPFENCED` has the default value `YES`. For fenced, not threadsafe routines (stored procedures and UDFs), this keeps the routine process alive. It is best to change the value of this keyword to `NO` while developing these routines, and then change it back to `YES` when you are ready to load the final version of your shared library. For more information, see "Rebuilding DB2 routine shared libraries" on page 25.

Note: `KEEPFENCED` was known as `KEEPDARI` in previous versions of DB2.

For Java application development, you need to update the `JDK_PATH` keyword with the path where the Java Development Kit is installed.

Note: `JDK_PATH` was known as `JDK11_PATH` in previous versions of DB2.

Procedure:

To change these settings enter:

```
db2 update dbm cfg using <keyword> <value>
```

For example, to set the keyword KEEPFENCED to NO:

```
db2 update dbm cfg using KEEPFENCED NO
```

To set the JDK_PATH keyword to the directory /home/db2inst/jdk13:

```
db2 update dbm cfg using JDK_PATH /home/db2inst/jdk13
```

To view the current settings in the database manager configuration file, enter:

```
db2 get dbm cfg
```

Note: On Windows, you need to enter these commands in a DB2 command window.

Related concepts:

- "Rebuilding DB2 routine shared libraries" on page 25
- "Database manager instances" on page 5

Related tasks:

- "Setting up the Java environment" on page 26

Related reference:

- "CREATE FUNCTION statement" in the *SQL Reference, Volume 2*
- "CREATE PROCEDURE statement" in the *SQL Reference, Volume 2*
- "GET DATABASE MANAGER CONFIGURATION Command" in the *Command Reference*
- "RESET DATABASE MANAGER CONFIGURATION Command" in the *Command Reference*

Setting up the Java environment

You can develop Java programs to access DB2 databases with the appropriate Java Developer Kit for your platform. The Developer Kit includes Java Database Connectivity (JDBC), a dynamic SQL API for Java.

DB2 JDBC support is provided as part of the Java Enablement option on DB2 clients and servers. With this support, you can build and run JDBC applications and applets. These contain dynamic SQL only, and use a Java call interface to pass SQL statements to DB2.

DB2 embedded SQL for Java (SQLJ) support is also provided as part of Java Enablement. With DB2 SQLJ support, in addition to DB2 JDBC support, you can build and run SQLJ applets and applications. These contain static SQL and use embedded SQL statements that are bound to a DB2 database.

The SQLJ support provided by the DB2 AD Client includes:

- The DB2 SQLJ translator, **sqlj**, which replaces embedded SQL statements in the SQLJ program with Java source statements, and generates a serialized profile which contains information about the SQL operations found in the SQLJ program.

- The DB2 SQLJ profile customizer, `db2sqljcustomize`, which precompiles the SQL statements stored in the serialized profile, customizes them into runtime function calls, and generates a package in the DB2 database.

  Note: The DB2 SQLJ profile customizer was called `db2profc` in previous versions of DB2.
- The DB2 SQLJ profile printer, `db2sqljprint`, which prints the contents of a DB2 customized version of a profile in plain text.

  Note: The DB2 SQLJ profile printer was called `db2profp` in previous versions of DB2.
- The DB2 SQLJ profile binder, `db2sqljbind`, which generates packages from a previously customized SQLJ program.

Note: The CLI-based Type 2 and Type 3 JDBC drivers are deprecated. No new features or enhancements will be put into these drivers and they will not be available in future DB2 releases. The completely redesigned Universal JDBC driver is provided to replace these legacy drivers with more enhanced features. You are encouraged to migrate applications to use the new driver as soon as possible.

Procedure:

To build applications with JDBC Universal Type 2 or JDBC Universal Type 4 connectivity, and to build applets with JDBC Universal Type 4 connectivity, the TCP/IP listener must be running. To ensure this, do the following:

1. Set the environment variable DB2COMM to TCPIP as follows:

   ```
   db2set DB2COMM=TCPIP
   ```
2. Update the database manager configuration file with the TCP/IP service name as specified in the services file:

   ```
   db2 update dbm cfg using SVCENAME <TCP/IP service name>
   ```

   You must do a ″db2stop″ and ″db2start″ for this setting to take effect.

Note: The port number used for applets and SQLJ programs needs to be the same as the TCP/IP SVCENAME number used in the database manager configuration file.

To run DB2 Java applications, you must install and invoke a Java Virtual Machine (JVM) that provides native threads support. To execute a Java application using native threads, you can use the `-native` option in your command. For example, to run the Java sample application, `DbInfo.class`, you can use the following command:

```
java -native DbInfo
```

You can specify native threads as the default thread support for some Java Virtual Machines by setting the THREADS_FLAG environment variable to ″native″. This documentation assumes native threads support is the default. Please refer to your JVM documentation for instructions on making native threads the default on your system.

To run DB2 Java applets, you can invoke a Java Virtual Machine that provides either native threads or green threads support.

When the above are installed and working, you can set up your particular operating system Java environment by following the steps in one of the following:
- "Setting up the UNIX Java environment" on page 32
- "Setting up the Windows Java environment" on page 42

For the latest DB2 Java application development updates, visit the Web page at:

`http://www.ibm.com/software/data/db2/udb/ad/v8/java`

Related tasks:

- "Setting up the UNIX Java environment" on page 32
- "Setting up the Windows Java environment" on page 42
- "Installing the DB2 Universal JDBC Driver" in the *Application Development Guide: Programming Client Applications*

Related reference:

- "db2sqljcustomize - DB2 SQLJ Profile Customizer Command" in the *Command Reference*
- "db2sqljprint - DB2 SQLJ Profile Printer Command" in the *Command Reference*
- "db2sqljbind - DB2 SQLJ Profile Binder Command" in the *Command Reference*

Setting up DB2 WebSphere MQ functions

DB2 and WebSphere MQ can be used to construct applications that combine messaging and database access. MQ functions are similar to user-defined functions (UDFs), and can be optionally enabled in DB2. By using these basic functions, it is possible to support a wide range of applications, from simple event notification to data warehousing, to updating federated data sources.

Procedure:

To set up DB2 WebSphere MQ functions:

1. Install WebSphere MQ on each physical machine.

   Ensure that a minimum of WebSphere MQ Version 5.1 with the latest FixPak is installed on your DB2 Universal Database server. If this version of WebSphere MQ is already installed then skip to the next step, "Install WebSphere MQ AMI." DB2 Version 8 includes a copy of the WebSphere MQ server to use with DB2. Platform-specific instructions for installing WebSphere MQ or for upgrading an existing WebSphere MQ installation can be found in a platform-specific Quick Beginnings book at http://www.ibm.com/software/ts/mqseries/library/manuals. Be sure to set up a default queue manager as you go through the installation process.
2. Install the WebSphere MQ Application Messaging Interface AMI on each physical machine.

   This is an extension to the WebSphere MQ programming interfaces that provide a clean separation of administrative and programming tasks. The DB2 WebSphere MQ functions require the installation of this interface. If the WebSphere MQ AMI is already installed on your DB2 server then skip to the next step, "Enable and Configure the DB2 WebSphere MQ User-defined Functions." If the WebSphere MQ AMI is not installed then you can install it from either the installation package provided with DB2, or by downloading a copy of the AMI from the WebSphere MQ SupportPacs web site at http://www.ibm.com/software/ts/mqseries/txppacs. The AMI can be found

under “Category 3 – Product Extensions”. For convenience, a copy of the WebSphere MQ AMI is provided with DB2. This file is located in the sqllib/cfg/mq directory.

The name of the file is operating system dependent:

Table 7. WebSphere MQ AMI

Operating system	Filename
AIX Version 4.3 and greater	ma0f_ax.tar.Z
HP-UX	ma0f_hp.tar.Z
Solaris Operating Environment	ma0f_sol7.tar.Z or mq0f_sol26.tar.Z
Windows	ma0f_win.zip

Note: DB2 WebSphere MQ functions are not supported on Linux

Follow the normal AMI installation process as outlined in the AMI readme file contained in the compressed installation image.

3. Enable and Configure the DB2 WebSphere MQ User-defined Functions.

   The **enable_MQFunctions** utility is a flexible command that performs the following actions:

   a. Checks that the proper WebSphere MQ environment has been set up
   b. Installs and creates a default configuration for the DB2 WebSphere MQ functions
   c. Enables the specified database with these functions
   d. Confirms that the configuration works

   On UNIX 64-bit, the runtime library path must be modified to include `$HOME/sqllib/lib32` in order to execute the enable/disable_MQFunctions. The following settings do this.

   AIX

   ```
   LIBPATH=$HOME/sqllib/lib32 enable_MQFunctions -n dbname \
      -u userid -p passwd -v 0pc [-q qMgr -force -noValidate]
   LIBPATH=$HOME/sqllib/lib32 disable_MQFunctions -n dbname \
      -u userid -p passwd -v 0pc
   ```

   HP-UX

   ```
   SHLIB_PATH=$HOME/sqllib/lib32 enable_MQFunctions -n dbname \
      -u userid -p passwd -v 0pc [-q qMgr -force -noValidate]
   SHLIB_PATH=$HOME/sqllib/lib32 disable_MQFunctions -n dbname \
      -u userid -p passwd -v 0pc
   ```

   Solaris

   ```
   LD_LIBRARY_PATH=$HOME/sqllib/lib32 enable_MQFunctions -n dbname \
      -u userid  -p passwd -v 0pc [-q qMgr -force -noValidate]
   LD_LIBRARY_PATH=$HOME/sqllib/lib32 disable_MQFunctions -n dbname \
      -u userid -p passwd -v 0pc
   ```

   During the enable step, you configure and enable a database for the DB2 WebSphere MQ functions with the following steps:

   a. For Windows, go to step ’c’.
   b. Enable the WebSphere MQ functions on UNIX by adding the DB2 instance owner (often db2inst1) and the user ID associated with fenced user-defined functions (often db2fenc1) to the WebSphere MQ group mqm. This is needed for the DB2 functions to access WebSphere MQ.
   c. Add the AMT_DATA_PATH environment variable to the list understood by DB2. You can edit the file $HOME/sqllib/profile.env (UNIX) or

%DB2PATH%\profile.env (Windows), and add AMT_DATA_PATH to DB2ENVLIST. You can also use the **db2set** command.

d. Restart the database instance for the environment variable changes to take effect.

e. Change directory to $HOME/sqllib/cfg (UNIX) or %DB2PATH%\cfg (Windows).

f. Run the command **enable_MQFunctions** to configure and enable a database for the DB2 WebSphere MQ functions. In a DB2 ESE environment, only perform this step on the catalog node. Refer to the topic ″enable_MQFunctions″ for a complete description of this command. Some common examples are given below. After successful completion, the specified database is enabled and the configuration is tested.

g. To test these functions using the Command Line Processor, issue the following commands after you have connected to the enabled database:

```
values DB2MQ.MQSEND('a test')
values DB2MQ.MQRECEIVE()
```

The first statement sends the message “a test” to the DB2MQ_DEFAULT_Q queue and the second receives it back.

Note: After running enable_MQFunctions, the utility establishes a default WebSphere MQ environment. The utility also creates the WebSphere MQ queue manager DB2MQ_DEFAULT_MQM and the default queue DB2MQ_DEFAULT_Q. The utility installs the files amt.xml, amthost.xml, and amt.dtd if they do not already exist in the directory pointed to by AMT_DATA_PATH. If an amthost.xml file does exist, and does not contain a definition for connection DB2MQ, then a line is added to the file with the appropriate information. A copy of the original file is saved as DB2MQSAVE.amthost.xml.

4. If you want to use transactional MQ UDFs, make sure that the database is configured for federated operations. Do this with the following command

```
update dbm cfg using federated yes
```

5. To make use of the publish/subscribe capabilities provided by the DB2 WebSphere MQ functions, you must also install either MQSeries Integrator or the WebSphere MQ Publish/Subscribe product extensions on each physical machine. Information on MQSeries Integrator can be found at

```
http://www.ibm.com/software/ts/mqseries/integrator
```

Information on the WebSphere MQ Publish/Subscribe feature can be found at

```
http://www.ibm.com/software/ts/mqseries/txppacs
```

Related concepts:

- “MQSeries Enablement” in the *Application Development Guide: Programming Client Applications*
- “WebSphere MQ Functional Overview” in the *Application Development Guide: Programming Client Applications*
- “WebSphere MQ Messaging” in the *Application Development Guide: Programming Client Applications*
- “Sending Messages with WebSphere MQ Functions” in the *Application Development Guide: Programming Client Applications*
- “Retrieving Messages with WebSphere MQ Functions” in the *Application Development Guide: Programming Client Applications*

- "WebSphere MQ Application-to-application Connectivity" in the *Application Development Guide: Programming Client Applications*
- "Request/Reply Communications with WebSphere MQ Functions" in the *Application Development Guide: Programming Client Applications*
- "Publish/Subscribe with WebSphere MQ Functions" in the *Application Development Guide: Programming Client Applications*
- "How to use WebSphere MQ functions within DB2" in the *IBM DB2 Information Integrator Application Developer's Guide*

Related reference:

- "db2mqlsn - MQ Listener Command" in the *Command Reference*
- "enable_MQFunctions" in the *Command Reference*
- "disable_MQFunctions" in the *Command Reference*

UNIX

For UNIX DB2 CLI setup information, see the *CLI Guide and Reference*.

Setting up the UNIX application development environment

You need to set environment variables for your database instance. Each database manager instance has two files, `db2profile` and `db2cshrc`, which are scripts to set the environment variables for that instance.

Procedure:

Run the correct script for the shell you are using:

For bash or Korn shell:

```
. $HOME/sqllib/db2profile
```

For C shell:

```
source $HOME/sqllib/db2cshrc
```

where `$HOME` is the home directory of the instance owner.

If you include this command in your `.profile` or `.login` file, the command runs automatically when you log on.

If you will be using ODBC, DB2 CLI, or Java, do the steps in one of the following topics:

- Setting Up the UNIX ODBC Environment
- Setting Up the UNIX Java Environment

Related concepts:

- "UNIX environment variable settings" on page 32

Related tasks:

- "Setting up the UNIX ODBC environment" in the *CLI Guide and Reference, Volume 1*
- "Setting up the UNIX Java environment" on page 32

Related reference:

- "AIX supported development software" on page 8

- "HP-UX supported development software" on page 10
- "Linux supported development software" on page 12
- "Solaris supported development software" on page 17

UNIX environment variable settings

Depending on the UNIX® platform you are on, values for the following environment variables are set, either in `db2profile` (for bash or korn shell) or `db2cshrc` (for C shell), and a call to these files are put in the instance owner's `.profile` (bash or korn shell) or `.login` (C shell) file.

AIX®:

- PATH, includes several DB2® directories including `sqllib/bin`
- LIBPATH, includes the directory `sqllib/lib` (see note below)

HP-UX:

- PATH, includes several DB2 directories including `sqllib/bin`
- SHLIB_PATH (32-bit and 64-bit) or LD_LIBRARY_PATH (64-bit), includes the directory `sqllib/lib` (see note below)

Linux and Solaris:

- PATH, includes several DB2 directories including `sqllib/bin`
- LD_LIBRARY_PATH, includes the directory `sqllib/lib` (see note below)

Note: If you are running a local 32-bit application in a 64-bit DB2 instance, see "Migrating applications from 32-bit to 64-bit environments" on page 51.

The blank files `sqllib/userprofile` and `sqllib/usercshrc` are created during instance creation to allow users to place their own instance environmental settings. These files will not be modified during an instance update (`db2iupdt`) in any DB2 FixPak or future version install. If you do not want the new environment settings in the `db2profile` or `db2cshrc` scripts, you can override them using the corresponding "user" script, which is called at the end of the `db2profile` or `db2cshrc` script. During an instance migration (`db2imigr`), the user scripts are copied over so that your environment modifications will still be in use.

Related tasks:

- "Setting up the UNIX application development environment" on page 31
- "Migrating applications from 32-bit to 64-bit environments" on page 51

Setting up the UNIX Java environment

To run JDBC and SQLJ programs on UNIX with DB2 JDBC support, commands to update your Java environment are included in the database manager files `db2profile` and `db2cshrc`. When a DB2 instance is created, `.bashrc`, `.profile`, and `.cshrc` are modified so that:

1. THREADS_FLAG is set to "native". (on Solaris only)
2. CLASSPATH includes:
   - "." (the current directory)
   - the file `sqllib/java/db2java.zip`
   - the file `sqllib/java/db2jcc.jar`
   - the file `sqllib/java/db2jcc_license_cu.jar`

Note: db2jcc_license_cisuz.jar is also included in the CLASSPATH for DB2 Connect Personal Edition, DB2 Connect Enterprise Edition, and DB2 ESE. This provides additional connectivity to DB2 for z/OS and OS/390, DB2 for AS/400 and iSeries, and DB2 for VSE & VM.

To build SQLJ programs, CLASSPATH is also updated to include the file:

```
sqllib/java/sqlj.zip
```

To build Data Source programs with Java Developer Kit 1.3 or 1.4, you must also obtain and install the following:

JNDI 1.2.1 class Libraries (jndi.jar and providerutil.jar)
http://java.sun.com/products/jndi/#download

File System Service Provider 1.2 (fscontext.jar)
http://java.sun.com/products/jndi/#download

For Java Developer Kit 1.3, you must additionally obtain and install the following:

JDBC 2.0 Optional Package
http://java.sun.com/products/jdbc/download.html#spec

Note: The JDBC 2.0 Optional Package is not required to build Data Source programs with Java Developer Kit 1.4.

For Data Source programs, you must also update your CLASSPATH to include the following files:

- jndi.jar
- fscontext.jar
- providerutil.jar

For Java Developer Kit 1.3, you must also update your CLASSPATH to include one of the following:

- jdbc2_0-stdext.jar
- j2ee.jar

Notes:

1. For Java Developer Kit 1.3, If you have already updated your CLASSPATH with j2ee.jar, you do not need jdbc2_0-stdext.jar.
2. For Java Developer Kit 1.4, jdbc2_0-stdext.jar and j2ee.jar are not required in your CLASSPATH.

Data Source sample programs are included in the `sqllib/samples/java/sqlj` directory. For details, see the samples README file in `sqllib/samples/java`.

Notes:

1. If other files are included in CLASSPATH, make sure the above files are specified first.
2. You should remove any direct reference to the `sqllib/java12` directory since this directory will not exist in future versions of DB2. Instead, reference the `sqllib/java` directory.
3. DB2 Java Enablement includes the JDBC package binder utility, db2jdbcbind. JDBC packages are bound automatically on DB2 Version 8 servers. The utility is to create the JDBC packages on downlevel servers, such as DB2 Versions 6 and 7.

Procedure:

To run DB2 Java routines (stored procedures and UDFs), you need to update the DB2 database manager configuration on the server to include the path where the Java Developer Kit is installed on that machine. You can do this by entering the following on the server command line:

```
db2 update dbm cfg using JDK_PATH /home/db2inst/jdk13
```

where */home/db2inst/jdk13* is the path where the Java Developer Kit is installed.

You can check the DB2 database manager configuration to verify the correct value for the JDK_PATH field by entering the following command on the server:

```
db2 get dbm cfg
```

You might want to redirect the output to a file for easier viewing. The JDK_PATH field appears near the beginning of the output.

When the above are installed and working, you can set up your specific UNIX operating system environment by following the steps in one of the following:
- Setting up the AIX Java Environment
- Setting up the HP-UX Java Environment
- Setting up the Linux Java Environment
- Setting up the Solaris Java Environment

Related tasks:
- "Setting up the AIX Java environment" on page 34
- "Setting up the HP-UX Java environment" on page 35
- "Setting up the Linux Java environment" on page 37
- "Setting up the Solaris Java environment" on page 38
- "Updating the database manager configuration file" on page 25
- "Installing the DB2 Universal JDBC Driver" in the *Application Development Guide: Programming Client Applications*

Related reference:
- "GET DATABASE MANAGER CONFIGURATION Command" in the *Command Reference*
- "RESET DATABASE MANAGER CONFIGURATION Command" in the *Command Reference*
- "db2jdbcbind - DB2 JDBC Package Binder Command" in the *Command Reference*

Setting up the AIX Java environment

Before implementing these instructions, do the setup in "Setting up the UNIX Java environment" on page 32.

Procedure:

To build Java applications on AIX with DB2 JDBC support, you need:

1. One of the supported developer kits listed in "AIX supported development software" on page 8.
2. DB2 Java Enablement, provided on DB2 Universal Database Version 8 for AIX clients and servers.

Related concepts:

- "Java sample programs" on page 107
- "Java applet considerations" on page 108

Related tasks:

- "Setting up the UNIX Java environment" on page 32
- "Setting up the sample database" on page 44

Related reference:

- "AIX supported development software" on page 8

Setting up the HP-UX Java environment

Before implementing these instructions, do the setup in "Setting up the UNIX Java environment" on page 32.

Procedure:

To build Java applications on HP-UX with DB2 JDBC support, you need to install and configure on your development machine:

1. One of the supported developer kits listed in "HP-UX supported development software" on page 10.
2. DB2 Java Enablement, provided on DB2 Universal Database Version 8 for HP-UX clients and servers.

For HP-UX 32-bit Java routines (stored procedures and user-defined functions) on HP-UX, the minimum JAVA_HEAP_SZ is 2048.

For HP-UX 64-bit, DB2 hardcodes the minimum heap setting to equal the maximum heap setting. On DB2 for HP-UX on IA64, the database manager configuration variable `JAVA_HEAP_SZ` should be set to at least 4096.

To run Java routines on HP-UX 32-bit, make sure the shared library path is similar to the following:

```
export SHLIB_PATH=$JAVADIR/jre/lib/PA_RISC:\
                  $JAVADIR/jre/lib/PA_RISC/classic:\
                  $HOME/sqllib/lib:\
                  /usr/lib:$SHLIB_PATH
```

where `$JAVADIR` is normally set to `/opt/java1.3` (the default location of the Java SDK for HP-UX 32-bit).

To run Java routines on HP-UX 64-bit, enable the `db2hpjv` tool by issuing this command on the command line:

```
db2hpjv -e
```

The following command disables this support:

```
db2hpjv -d
```

You must do a `db2stop` and `db2start` after issuing `db2hpjv -e` or `db2hpjv -d` in order for the changes to take effect. Java routine support is disabled by default.

Note: DB2 for HP-UX will not run if Java routine support is enabled with `db2hpjv -e` but Java is uninstalled on the system.

For HP-UX 64-bit on PA-RISC, symbolic links to the following libraries should be created in /usr/lib/pa20_64, otherwise you can encounter an SQL4301N error:

```
/opt/java1.4/jre/lib/PA_RISC2.0W/libnet.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libzip.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/librmi.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libnio.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libverify.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libmlib_image.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libhprof.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libjaas_unix.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libawt.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libcmm.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libdcpr.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libdt_socket.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libfontmanager.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libioser12.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libmawt.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libjsound.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libjava.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libjawt.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libjcov.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libjcpm.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libjdwp.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/libjpeg.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/hotspot/libjsig.sl
/opt/java1.4/jre/lib/PA_RISC2.0W/hotspot/libjvm.sl
```

If the links do not exist, they can be created with the following commands (which require root authority to execute):

```
ln -s /opt/java1.4/jre/lib/PA_RISC2.0W/*.sl    /usr/lib/pa20_64
ln -s /opt/java1.4/jre/lib/PA_RISC2.0W/hotspot/*.sl   /usr/lib/pa20_64
```

For HP-UX on IA64, symbolic links to the following libraries should be created in /usr/lib/hpux64, otherwise you can encounter an SQL4301N error:

```
/opt/java1.4/jre/lib/IA64W/hotspot/libjunwind.so
/opt/java1.4/jre/lib/IA64W/hotspot/libjvm.so
/opt/java1.4/jre/lib/IA64W/hotspot/libjsig.so
/opt/java1.4/jre/lib/IA64W/libJdbcOdbc.so
/opt/java1.4/jre/lib/IA64W/libverify.so
/opt/java1.4/jre/lib/IA64W/librmi.so
/opt/java1.4/jre/lib/IA64W/libzip.so
/opt/java1.4/jre/lib/IA64W/libawt.so
/opt/java1.4/jre/lib/IA64W/libcmm.so
/opt/java1.4/jre/lib/IA64W/libmawt.so
/opt/java1.4/jre/lib/IA64W/libjava.so
/opt/java1.4/jre/lib/IA64W/libjcov.so
/opt/java1.4/jre/lib/IA64W/libjcpm.so
/opt/java1.4/jre/lib/IA64W/libjdwp.so
/opt/java1.4/jre/lib/IA64W/libjpeg.so
/opt/java1.4/jre/lib/IA64W/libjsound.so
/opt/java1.4/jre/lib/IA64W/libmlib_image.so
/opt/java1.4/jre/lib/IA64W/libnet.so
/opt/java1.4/jre/lib/IA64W/libnio.so
/opt/java1.4/jre/lib/IA64W/libjaas_unix.so
/opt/java1.4/jre/lib/IA64W/libioser12.so
/opt/java1.4/jre/lib/IA64W/libhprof.so
/opt/java1.4/jre/lib/IA64W/libfontmanager.so
/opt/java1.4/jre/lib/IA64W/libdt_socket.so
/opt/java1.4/jre/lib/IA64W/libdcpr.so
/opt/java1.4/jre/lib/IA64W/libjawt.so
```

If the links do not exist, they can be created with the following commands (which require root authority to execute):

```
ln -s /opt/java1.4/jre/lib/IA64W/*.so    /usr/lib/hpux64
ln -s /opt/java1.4/jre/lib/IA64W/hotspot/*.so   /usr/lib/hpux64
```

Related concepts:

- "Java sample programs" on page 107
- "Java applet considerations" on page 108

Related tasks:

- "Setting up the UNIX Java environment" on page 32
- "Setting up the sample database" on page 44

Related reference:

- "HP-UX supported development software" on page 10

Setting up the Linux Java environment

Before implementing these instructions, do the setup in "Setting up the UNIX Java environment" on page 32.

Procedure:

To build Java applications on Linux with DB2 JDBC support, you need to install and configure on your development machine:

1. One of the following:
   - One of the supported developer kits listed in "Linux supported development software" on page 12.
   - DB2 Java Enablement, provided on DB2 Universal Database Version 8 for Linux clients and servers.

To run Java stored procedures or user-defined functions, the Linux run-time linker must be able to access certain Java shared libraries, and DB2 must be able to load both these libraries and the Java virtual machine. Since the program that does this loading runs with setuid privileges, it will only look for the dependent libraries in `/usr/lib`.

Create symbolic links in `/usr/lib` to point to the Java shared libraries. The following are essential shared libraries you need to link to. Depending on the applications you are building and running, you might need to link to additional shared libraries.

For the IBM Developer Kit 1.3, you need symbolic links to `libjava.so`, `libjvm.so`, and `libhpi.so`. You can create the symbolic links by running the following commands as root:

```
cd /usr/lib
ln -fs JAVAHOME/jre/bin/libjava.so .
ln -fs JAVAHOME/jre/bin/classic/libjvm.so .
ln -fs JAVAHOME/jre/bin/libhpi.so .
```

where *JAVAHOME* is the base directory for the IBM Developer Kit. If DB2 cannot find these libraries, you will get a -4301 error when trying to run a Java routine, and there will be messages in the administration notification log about libraries not found.

Note: An alternative is to add the Java shared libraries to `/etc/ld.so.conf` instead of creating links in `/usr/lib`. If you do this, you must run `ldconfig` as root after changing `/etc/ld.so.conf`, otherwise it will not work, as the call to the routine will hang (not complete). This alternative method can still not work in specific instances, also resulting in the routine hanging (not completing). If you encounter this, please create the links in the `/usr/lib` directory as instructed above.

Related concepts:

- "Java sample programs" on page 107
- "Java applet considerations" on page 108

Related tasks:

- "Setting up the UNIX Java environment" on page 32
- "Setting up the sample database" on page 44

Related reference:

- "Linux supported development software" on page 12

Setting up the Solaris Java environment

Before implementing these instructions, do the setup in "Setting up the UNIX Java environment" on page 32.

Procedure:

To build Java applications in the Solaris operating environment with DB2 JDBC support, you need to install and configure the following on your development machine:

1. One of the supported developer kits listed in "Solaris supported development software" on page 17.
2. DB2 Java Enablement, provided on DB2 Universal Database Version 8 for Solaris clients and servers.

Related concepts:

- "Java sample programs" on page 107
- "Java applet considerations" on page 108

Related tasks:

- "Setting up the UNIX Java environment" on page 32
- "Setting up the sample database" on page 44

Related reference:

- "Solaris supported development software" on page 17

Windows

For Windows DB2 CLI setup information, see the *CLI Guide and Reference*.

Setting up the Windows application development environment

When you install the DB2 AD Client on Windows NT, Windows 2000, Windows XP, or Windows Server 2003, the install program updates the configuration registry with the environment variables INCLUDE, LIB, and PATH. The system-wide environment variable, DB2INSTANCE, is set by install to the default instance created, called DB2. DB2PATH is set inside a DB2 command window when the window is opened. When you install the DB2 AD Client on Windows 98 or Windows ME, the install program updates the autoexec.bat file.

You can override these environment variables to set the values for the machine or the currently logged-on user. Exercise caution when changing these environment variables. Do not change the DB2PATH environment variable. DB2INSTANCE is defined as a system-level environment variable. You do not need to make use of the DB2INSTDEF DB2 registry variable which defines the default instance name to use if DB2INSTANCE is not set.

Procedure:

To override the environment variable settings, use any of the following:

- The Windows XP control panel
- The Windows Server 2003 control panel
- The Windows NT control panel
- The Windows 2000 control panel
- The Windows 98 or Windows ME command window
- The Windows 98 or Windows ME autoexec.bat file

When using the variable %DB2PATH% in a command, put the full path in quotes, as in set LIB="%DB2PATH%\lib";%LIB%. The default installation value for this variable is `\Program Files\IBM\SQLLIB`, which contains a space, so not using quotes can cause an error.

In addition, you must take the following specific steps for running DB2 applications:

- When building C or C++ programs, you must ensure that the `INCLUDE` environment variable contains `%DB2PATH%\INCLUDE` as the first directory.

  To do this, update the environment setup file for your compiler:

  Microsoft Visual C++ 6.0

  ```
  "C:\Program Files\Microsoft Visual Studio\VC98\bin\vcvars32.bat"
  ```

  Microsoft Visual C++ .NET

  ```
  "C:\Program Files\Microsoft Visual Studio
  .NET\Common7\Tools\vsvars32.bat"
  ```

  These files have the following commands:

  Microsoft Visual C++ 6.0

  ```
  set INCLUDE=%MSVCDir%\ATL\INCLUDE;%MSVCDir%\INCLUDE;
     %MSVCDir%\MFC\INCLUDE;%INCLUDE%
  ```

  Microsoft Visual C++ .NET

  ```
  @set INCLUDE=%MSVCDir%\ATLMFC\INCLUDE;...;
     %FrameworkSDKDir%\include;%INCLUDE%
  ```

To use either file with DB2, first move %INCLUDE%, which sets the %DB2PATH%\INCLUDE path, from the end of the list to the beginning, as follows:

Microsoft Visual C++ 6.0

```
set INCLUDE=%INCLUDE%;%MSVCDir%\ATL\INCLUDE;
   %MSVCDir%\INCLUDE;%MSVCDir%\MFC\INCLUDE
```

Microsoft Visual C++ .NET

```
@set INCLUDE=%INCLUDE%;%MSVCDir%\ATLMFC\INCLUDE;...;
   %FrameworkSDKDir%\include
```

- When building Micro Focus COBOL programs, set the COBCPY environment variable to point to %DB2PATH%\INCLUDE\cobol_mf.
- When building IBM COBOL programs, set the SYSLIB environment variable to point to %DB2PATH%\INCLUDE\cobol_a.
- Ensure the LIB environment variable points to %DB2PATH%\lib by using:

  ```
  set LIB="%DB2PATH%\lib";%LIB%
  ```

  Note: To enable cross-developing 64-bit applications from a 32-bit environment, see "Migrating applications from 32-bit to 64-bit environments" on page 51.
- Ensure that the DB2COMM environment variable is set at the server of a remote database.
- Ensure that the security service has started at the server for SERVER authentication, and at the client, when using CLIENT authentication.

  Note: Because CLIENT authentication happens on the client side instead of the server side, the client application is running under the context of the user. The Win32 authentication API requires certain privileges that might or might not be held by the user. In order to make sure that the CLIENT authentication takes place successfully, the authentication requests are passed from the client application to the security server (which runs under a privileged account local system by default, and has the right to call the authentication API).

  To start the security service manually, use the NET START DB2NTSECSERVER command.

  Normally, the only time you would want to set the security service to start automatically is if the workstation is acting as a DB2 client connecting to a server that is configured for Client Authentication. To have the security service start automatically, do the following:

  Windows NT

  1. Click the "Start" button.
  2. Click "Settings".
  3. Click "Control Panel".
  4. In the Control Panel, click "Services".
  5. In the Services window, highlight "DB2 Security Server".
  6. If it does not have the settings "Started" and "Automatic" listed, click "Startup".
  7. Click "Automatic".
  8. Click "OK".
  9. Reboot your machine to have the settings take effect.

  Windows 2000 and Windows Server 2003

  1. Click the "Start" button.

2. For Windows 2000, click ″Settings″ and then click ″Control Panel″.
   For Windows Server 2003, click ″Control Panel″.
3. Click ″Administrative Tools″.
4. Click ″Services″.
5. In the Services window, highlight ″DB2 Security Server″.
6. If it does not have the settings ″Started″ and ″Automatic″ listed, click ″Action″ from the top menu.
7. Click ″Properties″.
8. Make sure you are in the ″General″ tab.
9. Choose ″Automatic″ from the 'Startup Type' drop-down menu.
10. Click ″OK″.
11. Reboot your machine to have the settings take effect.

Windows XP

1. Click the ″Start″ button.
2. Click ″Settings″.
3. Click ″Control Panel″.
4. Click ″Performance and Maintenance″.
5. Click ″Administrative Tools″.
6. Click ″Services″.
7. In the Services window, highlight ″DB2 Security Server″.
8. If it does not have the settings ″Started″ and ″Automatic″ listed, click ″Action″ from the top menu.
9. Click ″Properties″.
10. Make sure you are in the ″General″ tab.
11. Choose ″Automatic″ from the 'Startup Type' drop-down menu.
12. Click ″OK″.
13. Reboot your machine to have the settings take effect.

The database manager on a Windows XP, Windows Server 2003, Windows NT, or a Windows 2000 environment is implemented as a service, and hence does not return errors or warnings when the service is started, though problems might have occurred. This means that when you run the `db2start` or the `NET START` command, no warnings will be returned if any communication subsystem failed to start. Therefore, the user should always examine the event logs or the DB2 Administration Notification log for any errors that might have occurred during the running of these commands.

If you will be using DB2 CLI or Java, proceed to the appropriate task:

– Setting Up the Windows CLI Environment
– "Setting up the Windows Java environment" on page 42

Related tasks:

- "Migrating applications from 32-bit to 64-bit environments" on page 51
- "Setting up the Windows CLI environment" in the *CLI Guide and Reference, Volume 1*
- "Setting up the Windows Java environment" on page 42

Related reference:

- "DB2 Application Development Client" on page 3

- "Windows supported development software" on page 19

Setting up the Windows Java environment

This topic provides the information you need to build and run DB2 Java programs in a Windows environment.

To run JDBC and SQLJ programs on a supported Windows® platform with DB2® JDBC support, CLASSPATH is automatically updated when DB2 is installed to include:

- "." (the current directory)
- the file `sqllib\java\db2java.zip`
- the file `sqllib\java\db2jcc.jar`
- the file `sqllib\java\db2jcc_license_cu.jar`

Note: db2jcc_license_cisuz.jar is also included in the CLASSPATH for DB2 Connect Personal Edition, DB2 Connect Enterprise Edition, and DB2 ESE. This provides additional connectivity to DB2 for z/OS and OS/390, DB2 for AS/400 and iSeries, and DB2 for VSE & VM.

To build SQLJ programs, CLASSPATH is also updated to include the file:

```
sqllib\java\sqlj.zip
```

To build Data Source programs with Java Developer Kit 1.3 or 1.4, you must also obtain and install the following:

JNDI 1.2.1 class Libraries (jndi.jar and providerutil.jar)
http://java.sun.com/products/jndi/#download

File System Service Provider 1.2 (fscontext.jar)
http://java.sun.com/products/jndi/#download

For Java Developer Kit 1.3, you must additionally obtain and install the following:

JDBC 2.0 Optional Package (jdbc2_0-stdext.jar)
http://java.sun.com/products/jdbc/download.html#spec

Note: The JDBC 2.0 Optional Package is not required to build Data Source programs with Java Developer Kit 1.4.

For Data Source programs, you must also update your CLASSPATH to include the following files:

- jndi.jar
- fscontext.jar
- providerutil.jar

For Java Developer Kit 1.3, you must also update your CLASSPATH to include one of the following:

- jdbc2_0-stdext.jar
- j2ee.jar

Notes:

1. For Java Developer Kit 1.3, If you have already updated your CLASSPATH with j2ee.jar, you do not need jdbc2_0-stdext.jar.

2. jdbc2_0-stdext.jar or j2ee.jar are not required in your CLASSPATH when using Java Developer Kit 1.4.

Data Source sample programs are included in the `sqllib\samples\java\sqlj` directory. For details, see the samples README file in `sqllib\samples\java`.

Notes:

1. If other files are included in CLASSPATH, make sure the above files are specified first.
2. You should remove any direct reference to the `sqllib\java12` directory since this directory will not exist in future versions of DB2. Instead, reference the `sqllib\java` directory.
3. DB2 Java Enablement includes the JDBC package binder utility, db2jdbcbind. JDBC packages are bound automatically on DB2 Version 8 servers. The utility is to create the JDBC packages on downlevel servers, such as DB2 Versions 6 and 7.
4. The Microsoft Software Developer's Kit for Java is not supported in DB2 Version 8. It cannot be used for SQLJ customization nor for running type 2 JDBC applications.

Procedure:

To build Java applications on a Windows operating system with DB2 JDBC support, you need to install and configure the following on your development machine:

1. One of the supported developer kits listed in "Windows supported development software" on page 19.
2. DB2 Java Enablement, provided on DB2 Universal Database Version 8 for Windows clients and servers.

To run DB2 Java routines (stored procedures and UDFs), you need to update the DB2 database manager configuration on the server to include the path where the Java Developer Kit is installed on that machine. You can do this by entering the following on the server command line:

```
db2 update dbm cfg using JDK_PATH c:\jdk13
```

where *c:\jdk13* is the path where the Java Developer Kit is installed.

If the path where the Java Developer Kit is installed contains a directory name with one or more spaces, you can either put the path in single quotes. For example:

```
db2 update dbm cfg using JDK_PATH 'c:\Program Files\jdk13'
```

or use the short-form of the directory name which does not have the space:

```
db2 update dbm cfg using JDK_PATH c:\progra~1\jdk13
```

You can check the DB2 database manager configuration to verify the correct value for the `JDK_PATH` field by entering the following command on the server:

```
db2 get dbm cfg
```

You might want to redirect the output to a file for easier viewing. The `JDK_PATH` field appears near the beginning of the output.

The following commands can be put into a batch file to set your Java environment for the IBM Java Developer Kits. The batch file must be run in a DB2 command window. Make sure you make all necessary path changes to suit your particular environment. Similar commands can be used for other supported Java Developer Kits.

Here are the commands for an example batch file to set up the Sun JDK 1.3.1 environment:

```
set JDKPATH=D:\JAVA\SUNjdk131
set PATH=%JDKPATH%\bin;%PATH%
set CLASSPATH=%CLASSPATH%;%JDKPATH%\lib\jdbc2_0-stdext.jar
db2 update dbm cfg using JDK_PATH %JDKPATH%
db2 terminate
db2stop
db2start
```

The batch file must be run in a DB2 Command window.

Related concepts:

- "Java sample programs" on page 107
- "Java applet considerations" on page 108

Related tasks:

- "Setting up the sample database" on page 44
- "Installing the DB2 Universal JDBC Driver" in the *Application Development Guide: Programming Client Applications*

Related reference:

- "Windows supported development software" on page 19

Sample database

Setting up the sample database

To use the sample programs shipped with DB2, you need to create the `sample` database on a server workstation. This will require approximately 23.5 megabytes of hard-drive space. An empty DB2 database requires approximately 21.5 megabytes of hard-drive space.

Also, if you will be using a remote client to access the `sample` database on a server that is running a different version of DB2, or running on a different operating system, you need to bind the database utilities, including the DB2 CLI utility files, to the `sample` database.

Procedure:

Here are the steps to set up the `sample` database:

1. "Creating the sample database" on page 45
2. "Cataloging the sample database" on page 47
3. "Binding the sample database utilities" on page 47

Related tasks:

- "Creating the sample database" on page 45

- "Cataloging the sample database" on page 47
- "Binding the sample database utilities" on page 47

Related reference:
- "The SAMPLE database" in the *SQL Reference, Volume 1*
- "db2sampl - Create Sample Database Command" in the *Command Reference*

Creating the sample database

You create the `sample` database on the command line with the `db2sampl` command.

Prerequisites:

You must have System Administrator (SYSADM) or System Control (SYSCTRL) authority to create a database. SYSADM and SYSCTRL are, respectively, the first and second highest levels of authority for DB2.

Procedure:

To create the database, do the following on the server:

1. Ensure that the location of `db2sampl` (the program that creates the `sample` database) is in your path. The `db2profile` or `db2cshrc` file will put `db2sampl` in your path, so it will remain there unless you change it.
   - On UNIX servers, `db2sampl` is located in:
     ```
     $HOME/sqllib/bin
     ```
     where `$HOME` is the home directory of the DB2 instance owner.
   - On Windows, `db2sampl` is located in:
     ```
     %DB2PATH%\bin
     ```
     where `%DB2PATH%` is where DB2 is installed.
2. Ensure that the DB2INSTANCE environment variable is set to the name of the instance where you want to create the `sample` database. If it is not set, you can set it with the following commands:
   - On UNIX:

     you can do this for the bash or Korn shell by entering:
     ```
     DB2INSTANCE=instance_name
     export DB2INSTANCE
     ```
     and for the C shell by entering:
     ```
     setenv DB2INSTANCE instance_name
     ```
   - On Windows, enter:
     ```
     set DB2INSTANCE=instance_name
     ```

   where *instance_name* is the name of the database instance.
3. Create the `sample` database by entering `db2sampl` followed by where you want to create the sample database. On UNIX platforms, this is a *path*, for example: "$HOME", and would be entered as:
   ```
   db2sampl path
   ```
   For example:
   ```
   db2sampl $HOME
   ```

On Windows, this is a *drive*, for example: "C:", and would be entered as:

```
db2sampl drive
```

For example:

```
db2sampl C:
```

If you do not specify the path or drive, the installation program installs the sample tables in the default path or drive specified by the DFTDBPATH parameter in the database manager configuration file. The authentication type for the database is the same as the instance in which it is created.

Related tasks:

- "Creating the sample database on Host or AS/400 and iSeries servers" on page 46
- "Cataloging the sample database" on page 47
- "Binding the sample database utilities" on page 47

Creating the sample database on Host or AS/400 and iSeries servers

If you want to run the sample programs against a Host server such as DB2 UDB for z/OS and OS/390, or an AS/400 and iSeries server, you need to create a database that contains the sample tables described in the SQL Reference.

Note: You need DB2 Connect to connect to a host server.

Restrictions:

There are some SQL syntax and DB2 command differences between DB2 on the workstation and DB2 on host systems. When accessing databases on DB2 UDB for z/OS and OS/390 or DB2 for AS/400 and iSeries, make sure your programs use SQL statements and precompile/bind options that are supported on these database systems.

Procedure:

To create the database:

1. Create the `sample` database in a DB2 workstation server instance using `db2sampl`.
2. Connect to the `sample` database.
3. Export the sample table data to a file.
4. Connect to the host database.
5. Create the sample tables.
6. Import the sample table data from the file where you exported the data on the workstation server.

Related concepts:

- "Export Overview" in the *Data Movement Utilities Guide and Reference*
- "Import Overview" in the *Data Movement Utilities Guide and Reference*

Related tasks:

- "Cataloging the sample database" on page 47

- "Binding the sample database utilities" on page 47

Related samples:

- "expsamp.sqb -- Export and import tables with table data to a DRDA database (IBM COBOL)"
- "tbmove.sqc -- How to move table data (C)"
- "tbmove.sqC -- How to move table data (C++)"

Cataloging the sample database

To access the `sample` database on the server from a remote client, you need to catalog the `sample` database on the client workstation.

You do not need to catalog the `sample` database on the server workstation because it was cataloged on the server when you created it.

Cataloging updates the database directory on the client workstation with the name of the database that the client application wants to access. When processing client requests, the database manager uses the cataloged name to find and connect to the database.

Procedure:

To catalog the sample database on the remote client workstation, enter:

```
db2 catalog database sample as sample at node nodename
```

where *nodename* is the name of the server node.

You must also catalog the remote node before you can connect to the database.

Related tasks:

- "Cataloging a TCP/IP node from the DB2 client" in the *Installation and Configuration Supplement*
- "Cataloging a database from a DB2 client using the CLP" in the *Installation and Configuration Supplement*
- "Cataloging a NetBIOS node from the DB2 client" in the *Installation and Configuration Supplement*
- "Cataloging a Named Pipes node from the client" in the *Installation and Configuration Supplement*
- "Binding the sample database utilities" on page 47

Binding the sample database utilities

If you will be accessing the `sample` database on the server from a remote client that is running a different version of DB2, you need to bind the database utilities, including the DB2 CLI utilities, to the `sample` database.

Binding creates the package that the database manager needs in order to access the database when an application is executed. Binding can be done explicitly by running the `BIND` command against the bind file created during precompilation.

Procedure:

You bind the database utilities differently depending on the platform of the client workstation you are using.

On a UNIX client workstation:

1. Connect to the sample database by entering:

   ```
   db2 connect to sample user userid using password
   ```

   where *userid* and *password* are the user ID and password of the instance where the sample database is located.
2. Bind the utilities to the database by entering:

   ```
   db2 bind BNDPATH/@db2ubind.lst blocking all sqlerror continue \
   messages bind.msg grant public

   db2 bind BNDPATH/@db2cli.lst blocking all sqlerror continue \
   messages cli.msg grant public
   ```

   where *BNDPATH* is the path where the bind files are located, such as $HOME/sqllib/bnd, where $HOME is the home directory of the DB2 instance owner.
3. Verify that the bind was successful by checking the bind message files bind.msg and cli.msg.

On a client workstation running a Windows operating system:

1. From the Start Menu, select Programs.
2. From the Programs Menu (or from 'All Programs' on Windows XP), select IBM DB2.
3. From the IBM DB2 menu, select "Command Line Tools".
4. From the "Command Line Tools" menu, select the DB2 Command Window.
   The command window displays.
5. Connect to the sample database by entering:

   ```
   db2 connect to sample user userid using password
   ```

   where *userid* and *password* are the user ID and password of the instance where the sample database is located.
6. Bind the utilities to the database by entering:

   ```
   db2 bind "%DB2PATH%\bnd\@db2ubind.lst" blocking all
   sqlerror continue messages bind.msg grant public

   db2 bind "%DB2PATH%\bnd\@db2cli.lst" blocking all
   sqlerror continue messages cli.msg grant public
   ```

   where %DB2PATH% is the path where DB2 is installed.
7. Exit the command window, and verify that the bind was successful by checking the bind message files, bind.msg and cli.msg.

For all clients accessing host servers, specify one of the following .lst files instead of db2ubind.lst:

ddcsmvs.lst
for DB2 for z/OS and OS/390

ddcsvm.lst
for DB2 for VM

ddcsvse.lst
for DB2 for VSE

ddcs400.lst
for DB2 for AS/400 and iSeries

For example:

- If accessing a DB2 for z/OS and OS/390 server from a UNIX client, enter:

```
db2 bind BNDPATH/@ddcsmvs.lst blocking all sqlerror continue \
  messages bind.msg grant public
```

- If accessing a DB2 for z/OS and OS/390 server from a Windows client, enter:

```
db2 bind "%DB2PATH%\bnd\@ddcsmvs.lst" blocking all
   sqlerror continue messages bind.msg grant public
```

Related tasks:

- "Creating the sample database" on page 45
- "Creating the sample database on Host or AS/400 and iSeries servers" on page 46
- "Cataloging the sample database" on page 47

Related reference:

- "BIND Command" in the *Command Reference*

Migrating applications

Migrating applications to DB2 Version 8

DB2® Version 8 supports the following DB2 versions for migration:

- DB2 Version 6
- DB2 Version 7.1
- DB2 Version 7.2
- DataJoiner® Version 2.1.1

When you migrate to a later version of DB2, your database and node directories are migrated automatically. To migrate from any other previous version of DB2, you must first migrate to one of the above supported versions, and then migrate from that version to DB2 version 8.

HP-UX

If you are migrating DB2 from HP-UX Version 10 or earlier to HP-UX Version 11, your DB2 programs must be re-precompiled with DB2 on HP-UX Version 11 (if they include embedded SQL), and must be re-compiled. This includes all DB2 applications, stored procedures, user-defined functions and user exit programs. As well, DB2 programs that are compiled on HP-UX Version 11 cannot run on HP-UX Version 10 or earlier. DB2 programs that are compiled and run on HP-UX Version 10 can connect remotely to HP-UX Version 11 servers.

Micro Focus COBOL

Any existing applications precompiled with DB2 Version 2.1.1 or earlier and compiled with Micro Focus COBOL should be re-precompiled with the current version of DB2, and then recompiled with Micro Focus COBOL. If these

applications built with the earlier versions of the IBM® precompiler are not re-precompiled, there is a possibility of database corruption if abnormal termination occurs.

Related concepts:

- "Migration recommendations" in the *Quick Beginnings for DB2 Servers*
- "Migrating Java applications, routines, and applets" on page 50
- "Running applications on two versions of DB2" on page 55

Related tasks:

- "Migrating databases" in the *Quick Beginnings for DB2 Servers*
- "Migrating instances (UNIX)" in the *Quick Beginnings for DB2 Servers*
- "Migrating DB2 UDB (Windows)" in the *Quick Beginnings for DB2 Servers*
- "Migrating DB2 UDB (UNIX)" in the *Quick Beginnings for DB2 Servers*
- "Migrating applications from 32-bit to 64-bit environments" on page 51
- "Ensuring application portability" on page 54

Related reference:

- "Administrative APIs and application migration" in the *Administrative API Reference*
- "Migration restrictions" in the *Quick Beginnings for DB2 Servers*
- "Version 8 incompatibilities with previous releases" in the *Administration Guide: Planning*

Migrating Java applications, routines, and applets

SQLJ applications and routines

In DB2® Version 8, SQLJ is based on a new, platform-independent architecture supporting greatly improved performance and portability of customized SQLJ applications and routines across DB2 supported operating systems. Several changes have been made from DB2 Version 7 SQLJ support. These changes might require some modification of SQLJ applications and routines, and will require retranslation and recustomization of SQLJ applications and routines being migrated to DB2 Version 8.

To migrate existing SQLJ applications and routines from Version 7 to Version 8, the following steps must be taken:

1. Change all VALUES statements to a dummy select. For example:

   ```
   #sql [ctxt] hv = {VALUES (DUMMY(1))}
   #sql [ctxt] {SELECT DUMMY(1)  INTO :hv FROM SYSIBM.SYSDUMMY1}
   ```

   Note: In DB2 Version 8, the `VALUES` statement and compound SQL are no longer supported with SQLJ.
2. Remove all BLOCK statements. Change to individual SQL statements. If the only two statements blocked were an executable SQL statement and a COMMIT, a single statement can be used with autocommit turned on for the connection. Optionally replace the use of BLOCK statements with calls to SQLJ batching APIs to achieve blocking objectives.
3. Retranslate applications and routines using the Version 8 SQLJ utilities with the **sqlj** command, and recustomize these applications and routines using the `db2sqljcustomize` command. This step must always be taken even if there are

no changes to the source code, so that the binaries will be runnable by the new cross-platform support that comes with Version 8.

Java™ applets

The type 3 JDBC driver, formerly known as the "net" driver, is deprecated. DB2 Java applets should be migrated to the DB2 Universal JDBC driver, which contains Type 4 connectivity. To convert a type 3 JDBC applet to use the new DB2 Universal JDBC driver, make the following changes:

1. The DB2 Universal JDBC driver archive is `db2jcc.jar`. In the `.html` file associated with the applet, change the archive from `db2java.zip` to `db2jcc.jar`. Copy `db2jcc.jar` to the web server.
2. The DB2 Universal JDBC driver class name is `com.ibm.db2.jcc.DB2Driver`. In the applet `.java` files, change the type 3 JDBC driver class name, `COM.ibm.db2.jdbc.net.DB2Driver`, to the DB2 Universal JDBC driver class name. There might not be a reference to the JDBC driver class if the applet uses `javax.sql.DataSource` to obtain connections.
3. Both the type 3 and DB2 Universal JDBC driver use a data source URL of the same form: `jdbc:db2://server:portnumber/dbname`. However, the three parts: `server`, `portnumber`, and `dbname` have a different meaning in the two drivers.

   The type 3 JDBC driver is a three tier model with a client (the browser running the applet), a JDBC Applet Server, and a DB2 server. The `server` and `portnumber` in the URL refer to the JDBC Applet Server. The `dbname` is the database alias cataloged on the system running the JDBC Applet Server.

   The DB2 Universal JDBC driver client connects directly to the DB2 server so the `server` and `portnumber` are those of the DB2 server TCP/IP listener. The `dbname` is the database alias cataloged on the DB2 server system.

   If the applet uses `DriverManager.getConnection` to connect to DB2, update the `.java` files and (if necessary) the `.html` file with the new URL for the DB2 Universal JDBC driver.
4. If the applet makes use of `COM.ibm.db2.jdbc.DB2DataSource`, new `javax.sql.DataSource` objects of the class `com.ibm.jcc.db2.DB2SimpleDataSource` must be created. The applet must be updated to use this new class.

Related concepts:

- "Migrating applications to DB2 Version 8" on page 49
- "Java applet considerations" on page 108

Related tasks:

- "Building SQLJ applications" on page 117
- "Building SQLJ routines" on page 121
- "Building SQLJ programs" on page 114

Migrating applications from 32-bit to 64-bit environments

Windows 32-bit applications can run as is on Windows 64-bit without any changes to the 64-bit environment. On UNIX, in all 64-bit DB2 instances except Linux for IA64 and Linux for zSeries, you can migrate your existing 32-bit local applications by rebinding the application, and running it with the correct library path setting. On HP-UX, you can only do this if the application was linked with the `+s` option.

If the application was not linked with the +s option, you must rebuild the application with the +s option, or with the embedded runtime path to include the 32-bit DB2 library (see below).

Procedure:

On UNIX, the correct library path for 32-bit applications in a 64-bit environment is lib32. You probably will not want to change the environment variable setting to lib32 since this will affect all applications in the instance environment (32-bit and 64-bit). To avoid this, you can use a wrapper script to set the environment variable just for the application that you also run from the script.

This is an example of a wrapper script you could use:

```
#! /bin/sh
echo <ENV_VAR_SETTING>
export <ENV_VAR_SETTING>
rm -rf $HOME/sqllib/db2dump/* > /dev/null 2>&1
echo
echo Running application...
$1
echo ...Done running application.
```

where <ENV_VAR_SETTING> is the environment variable setting for your platform:

AIX: LIBPATH=$HOME/sqllib/lib32:$LIBPATH

HP-UX (one of):

SHLIB_PATH=$HOME/sqllib/lib32:$SHLIB_PATH

LD_LIBRARY_PATH=$HOME/sqllib/lib32:$LD_LIBRARY_PATH

Note: The wrapper will only work on HP-UX if the application was linked with the +s option.

Linux: LD_LIBRARY_PATH=$HOME/sqllib/lib32:$LD_LIBRARY_PATH

Solaris:

LD_LIBRARY_PATH=$HOME/sqllib/lib32:$LD_LIBRARY_PATH

After rebinding the application, you can run this wrapper program by entering the wrapper script name on the command line followed by the executable name:

```
<wrapper_script> <executable>
```

Changing the environment variable inside a wrapper might not work for applications that invoke other executables (such as with a C system() call) if the wrapper script's library path is not compatible with the invoked executable. To migrate these applications, the object file must be relinked, and then the application rebound.

The runtime library path with lib32 should be used to link the object file, rather than the environment variable for the platform. The C, C++, and CLI build scripts for the sample programs use the appropriate runtime path to allow new applications to easily port to 64-bit environments.

The same link options should be used to link object files of existing 32-bit applications in 64-bit environments (see the build scripts in the samples related links below). The following flags can be used to include the 32-bit db2 library in the runtime library path:

AIX: -L$DB2PATH/lib32

Alternately, one can use the `-blibpath` linker option to specify a complete runtime library path. The AIX sample build scripts use the former method.

HP-UX:
-Wl,+b$DB2PATH/lib32

Linux: -Wl,-rpath,$DB2PATH/lib32

Solaris:
-R$DB2PATH/lib32

Notes:

1. Each of these commands assume that the compiler is used for the linking as opposed to linking with `ld` directly.
2. On Solaris, LD_LIBRARY_PATH and LD_LIBRARY_PATH_32 must be unset before you link the application with the runtime path. If not, the LD_LIBRARY_PATH or LD_LIBRARY_PATH_32 setting will be used instead of the runtime path setting.
3. On Linux, if you are using the `--enable-new-dtags` link option, unset LD_LIBRARY_PATH before running the 32-bit executable. If you do not, the LD_LIBRARY_PATH setting will be used instead of the runtime path setting.

Cross-development on Windows

To develop 64-bit applications on Windows 32-bit environments, ensure that the `LIB` environment variable points to `%DB2PATH%\lib\Win64`. By default the `%DB2PATH%\lib` (32-bit path) will be added to your LIB path so you must make certain that the `%DB2PATH%\lib\Win64` path is ahead of the default 32-bit path when doing cross-development of 64-bit applications in 32-bit environments. `%DB2PATH%\lib` is for developing 32-bit applications on 32-bit environments, or 64-bit applications on 64-bit environments.

Changing Long Data Types

If you want to migrate a 32-bit application for use in a 64-bit operating environment while still running on a 32-bit server, use the LONGERROR precompile option to prepare for porting the application. Set LONGERROR to YES in the 32-bit environment so that the precompiler returns an error whenever it encounters a host variable of the long type. Then follow these steps:

1. Prune the use of long types for host variables, unless long types are necessary. Instead, use the new portable host variables, sqlint32 or sqluint32 . For example:

```
EXEC SQL BEGIN DECLARE SECTION;
 long y;     /* this declaration generates an error on 64 bit */
 sqlint32 x; /* this declaration is acceptable for 64 bit */
EXEC SQL END DECLARE SECTION;
```

2. Precompile the application against a database on a 64-bit server. This creates a new package for the application that is being ported.
3. Compile the application in 64-bit mode.
4. Link the application with the new 64-bit DB2 libraries.
5. Bind the application to a database on a 64-bit server.

Note: DB2 does not support running a 64-bit application in a 32-bit instance.

Related concepts:

- "UNIX environment variable settings" on page 32
- "Migrating applications to DB2 Version 8" on page 49

Related samples:

- "bldapp -- Builds AIX C application programs (C)"
- "bldapp -- Builds HP-UX C applications (C)"
- "bldapp -- Builds Linux C applications (C)"
- "bldapp -- Builds Solaris C applications (C)"

Ensuring application portability

Here are points to keep in mind when developing your applications. They will help make your applications portable.

Procedure:

- On UNIX, use only the default library search path, `/usr/lib:/lib`, in your applications. On Windows® operating systems, ensure the LIB environment variable points to `%DB2PATH%\lib` by using:

  ```
  set LIB=%DB2PATH%\lib;%LIB%
  ```

  Also, create symbolic links between the default path and the version of DB2 you are using. Ensure that the link is to the minimum level of DB2 required by your applications. Refer to the *Quick Beginnings* book, or installation topic, for your platform for information about setting links.
- If your application requires a particular version of DB2, use the path that specifies the DB2 version in your application. For example, if your AIX® application requires DB2 Version 5, use /usr/lpp/db2_05_00/lib. Ordinarily, you do not need to do this.
- When you are building an application for production, rather than internal development, the path in your application should not point to the instance owner's copy of the `sqllib/lib` directory on UNIX, or the `sqllib\lib` directory on Windows operating systems. This makes applications highly dependent on specific user names and environments.
- Generally, do not use the following environment variables to alter search paths in a particular environment: LIBPATH (AIX), SHLIB_PATH (HP-UX 32-bit), LD_LIBRARY_PATH (HP-UX 64-bit, Linux, and Solaris), and LIB (Windows). These variables override the search paths specified in the applications running in the environment, so applications might not be able to find the libraries or the files they need.
- In DB2 Universal Database™ Versions 6, 7, and 8, all character array items with string semantics have type char, instead of other variations, such as unsigned char. Any applications you code with DB2 Universal Database Version 6, Version 7, or Version 8, should follow this practice.

  If you have DB2 Version 1 applications which use unsigned char, your compiler might produce warnings or errors because of type clashes between unsigned char in Version 1 applications and char in Version 6, Version 7, or Version 8 function prototypes. If this occurs, use the compiler option `-DSQLOLDCHAR` to eliminate the problem.

Related concepts:

- "UNIX environment variable settings" on page 32
- "Migrating applications to DB2 Version 8" on page 49

Related tasks:

- "Setting up the Windows application development environment" on page 39
- "Migrating applications from 32-bit to 64-bit environments" on page 51

Running applications on two versions of DB2

On UNIX® platforms, if you have applications from a previous DB2® release version and you want them to run in both a database instance of the previous version as well as a DB2 Version 8 instance on the same machine, you might need to make some changes to your environment. To determine what changes to make, answer the following questions, and then review the "Conditions" section to see if any of the conditions apply to your situation.

An AIX® system is used to explain the points raised. The same concepts apply to other UNIX platforms, but the details can differ, such as environment variables and specific commands.

Questions

Question 1: How was the application on the previous DB2 version linked to the DB2 client run-time library, for example, libdb2.a on AIX?

To determine the embedded shared library search path for an executable, use one of the following system commands in the directory where the executable resides (which can be `/usr/bin` or your instance directory):

AIX dump -H *executable_filename*

HP-UX
chatr *executable_filename*

Linux objdump -p *executable_filename*

Solaris
dump -Lv *executable_filename*

where *executable_filename* is the name of the executable file for the application.

The following is a sample dump listing from DB2 Version 7.2 for the AIX C sample application, `dbcat`, taken in the samples sub-directory of the DB2 instance, `/home/dbinst/samples/c`:

Sample Dump Listing for C application dbcat

```
---------------------------------------------------------------------

    dbcat:

                          ***Loader Section***
                       Loader Header Information
    VERSION#         #SYMtableENT     #RELOCent        LENidSTR
    0x00000001       0x0000000f       0x00000015       0x00000047

    #IMPfilID        OFFidSTR         LENstrTBL        OFFstrTBL
    0x00000003       0x00000284       0x0000007f       0x000002cb


                          ***Import File Strings***
    INDEX  PATH                            BASE               MEMBER
    0      /home/db2inst/sqllib/lib:/usr/lib:/lib

    1                                      libc.a             shr.o
    2                                      libdb2.a           shr.o

 ---------------------------------------------------------------------
```

Line 0 (zero) shows the directory paths that the executable searches to find the shared libraries to which it is linked. Lines 1, and 2 show the shared libraries to which the application is linked.

Depending on how the application was built, you could see the following paths: `/usr/lpp/db2_07_01_0000/lib`, *INSTHOME*`/sqllib/lib` (where *INSTHOME* is the home directory of the database instance owner), or just the `/usr/lib:/lib` combination.

Question 2: How are the DB2 run-time libraries configured on your system?

When either of DB2 Versions 1, 2, 5, 6, 7, or 8 is installed, there is an optional step which creates symbolic links from the system default shared library path `/usr/lib` to the DB2 install path which contains the DB2 client run-time libraries.

The install paths for the different DB2 versions on AIX are as follows:

Version 1
`/usr/lpp/db2_01_01_0000/lib`

Version 2
`/usr/lpp/db2_02_01/lib`

Version 5
`/usr/lpp/db2_05_00/lib`

Version 6.1
`/usr/lpp/db2_06_01/lib`

Version 7
`/usr/lpp/db2_07_01/lib`

Version 8
`/usr/opt/db2_08_01/lib`

In all cases, the run-time shared libraries are named `libdb2.a`.

Only one version of these libraries can be the default at any one time. DB2 provides this default so that when you build an application, it does not depend on a particular version of DB2.

Question 3: Do you specify different search paths in your environment?

You can override the shared library search path coded in your application using the LIBPATH environment variable on AIX, SHLIB_PATH on HP-UX 32-bit, SHLIB_PATH or LD_LIBRARY_PATH on HP-UX 64-bit, or LD_LIBRARY_PATH on Linux and Solaris. On Solaris, you can also use LD_LIBRARY_PATH_32 for 32-bit applications and LD_LIBRARY_PATH_64 for 64-bit applications. You can see the library search path using the appropriate system command for your platform given in the answer to Question 1.

Conditions

Once you have the answers to the questions above, you might need to make changes to your environment. Read the conditions listed below. If one of the conditions applies to your situation, make the necessary changes.

Condition 1: If a Version 7 application loads a shared library out of the AIX default shared library path `/usr/lib/libdb2.a`, and

- If there is a symbolic link from `/usr/lib/libdb2.a` to `/usr/lpp/db2_07_01/lib/libdb2.a`, and the database server is DB2 Universal Database Version 8 for AIX, do one of the following:
  - Change the symbolic link to point to:

    ```
    /usr/opt/db2_08_01/lib/libdb2.a
    ```

    As root, you can change links using the "db2ln" command as follows:

    ```
    /usr/opt/db2_08_01/cfg/db2ln
    ```
  - Set the LIBPATH environment variable to point to `/usr/opt/db2_08_01/lib` or *INSTHOME*`/sqllib/lib`, where *INSTHOME* is the home directory of the Version 8 DB2 instance owner.
  - Configure a TCP/IP connection from the application (client) instance to the server instance.
- If there is a symbolic link from `/usr/lib/libdb2.a` to `/usr/opt/db2_08_01/lib/libdb2.a`, and the database server is DB2 Version 7, configure a TCP/IP connection from the application (client) instance to the server instance.

Condition 2: If a Version 7 application loads a shared library out of the $HOME path of a DB2 Version 7 instance owner (`$HOME/sqllib/lib/libdb2.a`), and the database server is DB2 Universal Database™ Version 8 for AIX, do one of the following:

- Migrate the application instance to the same version as the database server instance.
- Set the LIBPATH environment variable to point to `/usr/opt/db2_08_01/lib` or *INSTHOME*`/sqllib/lib`, where *INSTHOME* is the home directory of the Version 8 instance owner.
- Configure a TCP/IP connection from the application (client) instance to the server instance.

Condition 3: If a Version 7 application loads a shared library out of the DB2 Version 7 install path (`/usr/lpp/db2_07_01/lib/libdb2.a`), and the database server is DB2 Universal Database Version 8 for AIX, do one of the following:

- Set the LIBPATH environment variable to point to `/usr/opt/db2_08_01/lib` or *INSTHOME*`/sqllib/lib`, where *INSTHOME* is the home directory of the database instance owner.
- Configure a TCP/IP connection from the application (client) instance to the server instance.

Condition 4: If a Version 7 application loads a shared library out of the DB2 Universal Database Version 8 for AIX install path (`/usr/opt/db2_08_01/lib/libdb2.a`), and the database server is DB2 Version 7, configure a TCP/IP connection from the application (client) instance to the server instance.

Related concepts:

- "UNIX environment variable settings" on page 32
- "Database manager instances" on page 5
- "Migrating applications to DB2 Version 8" on page 49

Related tasks:

- "Setting up the Windows application development environment" on page 39

Where to go next

Once your environment is set up, you are ready to build your DB2 applications. The following chapter discusses the sample programs and related files, including the build files. The chapters following this use the build files and samples to show you how to compile, link, and run your applications in your programming environment. Read the specific chapter for your particular application development needs.

Chapter 3. Sample programs and related files

This chapter describes sample programs and related files for the programming languages for all platforms supported by DB2. It presents the design for the samples based on the component structure of DB2, and gives a listing of the DB2 samples with a description of each one. It also explains the uses of the build files, makefiles and error-checking utilities that come with DB2.

Sample files

The sample programs come with the DB2® Application Development (DB2 AD) Client. Not all sample programs are available on all platforms or supported programming languages. You can use the sample programs as templates to create your own applications, and as a learning tool to understand DB2 functionality.

DB2 sample programs are provided "as is" without any warranty of any kind. The user, and not IBM®, assumes the entire risk of quality, performance, and repair of any defects.

Besides the sample program files, other sample files are provided by DB2 in the samples directories under `sqllib/samples` (UNIX) and under `sqllib\samples` (Windows). These include build files and makefiles to compile and link the sample programs, error-checking utility files which are linked in to most sample programs, and various script files to assist in application development. For example, scripts are provided to catalog and uncatalog stored procedures and UDFs in several language sub-directories. Each samples directory has a README file which describes the files contained in the directory.

HTML versions of most sample program source files are provided, accessible in the online documentation. These 'samples in HTML' are linked into the documentation topics to demonstrate the functionality described by them. Keywords, such as SQL statements and DB2 APIs, are hot-linked within the samples in HTML so the user can go directly to the documentation describing them. Most samples in HTML have a link in the comment section at the top of the file to a sample output file showing typical results of running the compiled sample program. Please note that the actual output is, in many cases, machine and platform-dependent, so the output you receive from running the same program might vary.

Below is a table showing the sample directories and README files for the main supported programming languages/APIs by platform. The README files are hot-linked in the online documentation, and the samples listings within them have hotlinks to the sample file source code. You can also access the sample files in the listed samples directories. For the directory paths, the UNIX®-style slashes are used, as in `samples/c`, except where the directory is for Windows® only, as in `samples\VB\ADO`.

Table 8. Sample README files by platform

Platform —> Language	**AIX®**	**HP-UX**	**Linux**	**Solaris**	**Windows**
C samples/c	README	README	README	README	README
C++ samples/cpp	README	README	README	README	README
C# samples\.NET\cs	n/a	n/a	n/a	n/a	README
CLI samples/cli	README	README	README	README	README
CLP samples/clp	README	README	README	README	README
IBM COBOL samples/cobol	README	n/a	n/a	n/a	README
Micro Focus COBOL samples/cobol_mf	README	README	README	README	README
JDBC samples/java/jdbc	README	README	README	README	README
SQLJ samples/java/sqlj	README	README	README	README	README
Perl samples/perl	README	README	README	README	README
PHP samples/php	README	README	README	README	README
SQL procedures samples/sqlproc	README	README	README	README	README
Visual Basic samples\VB\ADO	n/a	n/a	n/a	n/a	ReadMe.txt
Visual Basic .NET samples\.NET\vb	n/a	n/a	n/a	n/a	README

Sample program file extensions differ for each supported language, and for embedded SQL and non-embedded SQL programs within each language. File extensions might also differ for groups of programs within a language. These different sample file extensions are categorized in the following tables:

Sample file extensions by language
Table 9 on page 61.

Sample file extensions by program group
Table 10 on page 61.

Table 9. Sample file extensions by language

Language	Directory	Embedded SQL programs	Non-embedded SQL programs
C	`samples/c` `samples/cli` (CLI programs)	`.sqc`	`.c`
C++	`samples/cpp`	`.sqC` (UNIX) `.sqx` (Windows)	`.C` (UNIX) `.cxx` (Windows)
C#	`samples\.NET\cs`		`.cs`
COBOL	`samples/cobol` `samples/cobol_mf`	`.sqb`	`.cbl`

Java™	`samples/java/jdbc`
	`samples/java/sqlj`
	`samples/java/WebSphere`
	`samples/java/plugin`
		`.sqlj`	`.java`

REXX	`samples/rexx`	`.cmd`	`.cmd`

Visual Basic	`samples\VB\ADO`
	`samples\VB\MTS`
	`samples\VB\RDO`
			`.bas .frm .vbp`

Visual Basic .NET	`samples\.NET\vb`		`.vb`
Visual C++	`samples\VC\ADO`		`.cpp .dsp .dsw`

Table 10. Sample file extensions by program group

Sample group	Directory	File Extension
CLP	`samples/clp`	`.db2`
OLE	`samples\ole\msvb` (Visual Basic) `samples\ole\msvc` (Visual C++)	`.bas .vbp` (Visual Basic) `.cpp` (Visual C++)
OLE DB	`samples\oledb`	`.db2`
SQL procedures	`samples/sqlproc`	`.db2` (SQL Procedure scripts) `.c` (CLI Client Applications) `.sqc` (embedded C Client Applications) `.java` (JDBC Client Applications)
User exit	`samples/c`	`.ctsm` (UNIX & Windows) `.cdisk` (UNIX & Windows) `.ctape` (UNIX) `.cxbsa` (UNIX)

Note:

Directory delimiters
The directory delimiter on UNIX is a /. On Windows it is a \. In the tables, the UNIX delimiters are used unless the directory is only available on Windows.

Embedded SQL programs
Require precompilation, except for REXX embedded SQL programs where the embedded SQL statements are interpreted when the program is run.

IBM COBOL samples
Are only supplied for AIX and Windows 32-bit operating systems in the `cobol` subdirectory.

Micro Focus COBOL samples
Are only supplied for AIX, HP-UX, Solaris Operating Environment, and Windows 32-bit operating systems in the `cobol_mf` subdirectory.

Java samples
Are Java Database Connectivity (JDBC) applets, applications, and routines, embedded SQL for Java (SQLJ) applets, applications, and routines. Also, WebSphere® samples and plugin example files for the DB2 Control Center. Java samples are available on all supported DB2 platforms.

REXX samples
Are only supplied for AIX and Windows 32-bit operating systems.

CLP samples
Are Command Line Processor scripts that execute SQL statements.

OLE samples
Are for Object Linking and Embedding (OLE) in Microsoft® Visual Basic and Microsoft Visual C++, supplied for Windows operating systems only.

Visual Basic samples
Are ActiveX Data Objects, Remote Data Objects, and Microsoft Transaction Server samples, supplied on Windows operating systems only.

Visual C++ samples
Are ActiveX Data Object samples, supplied on Windows operating systems only.

User exit samples
Are Log Management User Exit programs used to archive and retrieve database log files. The files must be renamed with a `.c` extension and compiled as C language programs.

The sample programs directory is typically read-only on most platforms. Before you alter or build the sample programs, copy them to your working directory.

Structure and design

Most of the DB2 samples in C, CLI, C++, C#, Java, Perl, PHP, Visual Basic ADO, and Visual Basic .NET are organized to reflect an object-based design model of the database components.. The samples are grouped in categories representing different levels of DB2. The level to which a sample belongs is indicated by a two character prefix at the beginning of the sample name. Not all levels are represented in the samples for each Application Programming Interface, but for the samples as a whole, the levels are represented as follows:

prefix	DB2 Level
il	Installation Image Level
cl	Client Level
in	Instance Level
db	Database Level
ts	Table Space Level
tb	Table Level

dt Data Type Level

The levels show a hierarchical structure. The Installation image level is the top level of DB2. Below this level, a client-level application can access different instances; an instance can have one or more databases; a database has table spaces within which tables exist, and which in turn hold data of different data types.

This design does not include all DB2 samples. The purpose of some samples is to demonstrate different methods for accessing data. These methods are the main purpose of these samples so they are represented by these methods in a similar manner as the above:

prefix Programming method

fn SQL function

sp Stored procedure

ud User-defined function

Besides these categories, there are a set of tutorial samples to introduce basic concepts of database programming. These samples use some of the simpler functions represented in the samples design, and begin with the characters: "tut".

There are other samples not included in this design, such as the Log Management User Exit samples, samples in COBOL, Visual C++, REXX, Object Linking and Embedding (OLE) samples, CLP scripts, and SQL Procedures.

Note: Java, C#, and Visual Basic .NET program names have the first character, and sometimes other characters, in uppercase. Java programs do not have the underscores in the tutorial sample names. Visual Basic ADO samples have some characters in uppercase (but not the first character).

For the latest samples updates, visit the DB2 application development samples Web page:

```
http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html
```

Related concepts:

- "Build files" on page 97
- "Makefiles" on page 99
- "Error-checking utilities" on page 102

Related reference:

- "C samples" on page 64
- "CLI samples" on page 71
- "JDBC samples" on page 78
- "SQLJ samples" on page 80
- "SQL procedure samples" on page 90
- "Visual Basic samples" on page 93
- "Visual C++ samples" on page 96
- "Object Linking and Embedding (OLE) samples" on page 85
- "Object Linking and Embedding Database (OLE DB) table function samples" on page 86
- "Command Line Processor (CLP) samples" on page 73

- "Log management user exit samples" on page 83
- "COBOL samples" on page 73
- "Java WebSphere samples" on page 82
- "Java plug-in samples" on page 83
- "Windows Management Instrumentation samples" on page 96
- "REXX samples" on page 88
- "Dynamic reconfiguration samples" on page 77
- "C# samples" on page 70
- "Visual Basic .NET samples" on page 94
- "Perl Samples" on page 86
- "PHP samples" on page 87
- "Security plug-in samples" on page 90

Sample programs by language and application interface

C samples

UNIX directory: `sqllib/samples/c`. Windows directory: `sqllib\samples\c`.

File extensions: `.c` (no embedded SQL); `.sqc` (embedded SQL)

Table 11. C sample program files

Sample program name	Program description
Tutorial samples - Programs that demonstrate basic database operations.	
`tut_mod.sqc`	How to modify table data.
`tut_read.sqc`	How to read tables.
`tut_use.sqc`	How to use a database.
Client level - Samples that deal with the client level of DB2.	
`cli_info.c`	How to get and set client level information.
`clisnap.c`	How to capture a snapshot at the client level.
Instance level - Samples that deal with the instance level of DB2.	
`inattach.c`	How to attach to/detach from an instance.
`inauth.sqc`	How to display authorities at instance level.
`ininfo.c`	How to get and set instance level information.
`insnap.c`	How to capture a snapshot at the instance level.
`instart.c`	How to stop and start the current local instance.
Database level - Samples that deal with database objects in DB2.	
`dbauth.sqc`	How to grant/display/revoke authorities at the database level
`dbcfg.sqc`	How to configure database and database manager parameters.
`dbconn.sqc`	How to connect and disconnect from a database.
`dbcreate.c`	How to create and drop databases.
`dbhistfile.sqc`	How to read and update a database recovery history file entry.
`dbinfo.c`	How to get and set information at a database level.
`dbinline.sqc`	How to use inline SQL procedure language.

Table 11. C sample program files (continued)

Sample program name	Program description
dbinspec.sqc	How to check architectural integrity with the DB2 API db2Inspect.
dblogconn.sqc	How to read database log files asynchronously with a database connection.
dblognoconn.sqc	How to read database log files asynchronously with no database connection.
dbmcon.sqc	How to connect to and disconnect from multiple databases.
dbmcon1.h	Header file for dbmcon1.sqc.
dbmcon1.sqc	Support file for dbmcon.sqc.
dbmcon2.h	Header file for dbmcon2.sqc.
dbmcon2.sqc	Support file for dbmcon.sqc.
dbmigrat.c	How to migrate a database.
dbpkg.sqc	How to work with packages.
dbrec.sqc	How to use the db2GetRecommendations API.
dbrecov.sqc	How to recover a database.
dbredirect.sqc	How to perform Redirected Restore of a database.
dbrestore.sqc	How to restore a database from a backup.
dbrollfwd.sqc	How to perform rollforward after a restore of a database.
dbsample.sqc	How to create the sample database including Host and AS/400 tables and views.
dbsnap.c	How to capture a snapshot at the database level.
dbthrds.sqc	How to use multiple context APIs on UNIX.
dbthrds.sqc	How to use multiple context APIs on Windows.
dbuse.sqc	How to use database objects.
Table space level - Samples that deal with the table space level of DB2.	
tscreate.sqc	How to create and drop buffer pools and table spaces.
tsinfo.sqc	How to get information at the table space level.
Table level - Samples that deal with table objects in DB2.	
tbast.sqc	How to use a staging table for updating a deferred Automatic Summary Table.
tbcompress.sqc	How to create tables with null and default value compression options.
tbconstr.sqc	How to work with table constraints.
tbcreate.sqc	How to create, alter and drop tables.
tbident.sqc	How to use identity columns.
tbinfo.sqc	How to get and set information at a table level.
tbintrig.sqc	How to use an 'INSTEAD OF' trigger on a view.
tbload.sqc	How to load into a partitioned database.
tbmerge.sqc	How to use the MERGE statement.
tbmod.sqc	How to modify information in a table.
tbmove.sqc	How to move a table data.
tbonlineinx.sqc	How to create and reorganize indexes on a table.

Table 11. C sample program files (continued)

Sample program name	Program description
tbpriv.sqc	How to grant/display/revoke table level privileges.
tbread.sqc	How to read information in a table.
tbreorg.sqc	How to reorganize a table.
tbrunstats.sqc	How to perform runstats on a table.
tbsavept.sqc	How to use external savepoints.
tbsel.sqc	How to select from each of: insert, update, delete.
tbselcreate.db2	How to create the tables for the tbsel program.
tbseldrop.db2	How to drop the tables for the tbsel program.
tbtemp.sqc	How to use a declared temporary table.
tbtrig.sqc	How to use a trigger on a table.
tbumqt.sqc	How to use user materialzed query tables (summary tables).
tbunion.sqc	How to insert through a UNION ALL view.
tbxload.sqc	How to simultaneously return data from a select statement and load it into a table.
Data type level - Samples that deal with data types.	
dtformat.sqc	How to use load and import data format extensions.
dtlob.sqc	How to read and write LOB data.
dtudt.sqc	How to create, use, and drop user-defined distinct types.
DB2 function level	
fnuse.sqc	How to use SQL functions.
Stored procedure level - Samples that demonstrate stored procedures.	
spcat	Stored procedure catalog script for the spserver program. This script calls spdrop.db2 and spcreate.db2.
spcreate.db2	CLP script to issue CREATE PROCEDURE statements.
spdrop.db2	CLP script to drop stored procedures from the catalog.
spclient.sqc	Client program used to call the server routines declared in spserver.sqc.
spserver.sqc	Stored procedure routines built and run on the server.
UDF level - Samples that demonstrate user-defined functions.	
udfcli.sqc	Client application which calls the user-defined function in udfsrv.c, udfsrv.C.
udfsrv.c	User-defined function ScalarUDF called by udfcli.sqc
udfemcli.sqc	Client application which calls the embedded SQL user defined function library udfemsrv.
udfemsrv.sqc	Embedded SQL User-defined function library called by udfemcli.
Other	
evm.sqc	How to create and parse file, pipe, and table event monitors.
utilrecov.c	Utilities for the backup, restore and log file samples.
utilsnap.c	Utilities for the snapshot monitor samples.

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Related concepts:

- "Build files" on page 97
- "Makefiles" on page 99
- "Error-checking utilities" on page 102
- "Sample files" on page 59

C++ samples

UNIX directory: `sqllib/samples/cpp`. Windows directory: `sqllib\samples\cpp`.

UNIX file extensions: `.C` (no embedded SQL); `.sqC` (embedded SQL)

Windows file extensions: `.cxx` (no embedded SQL); `.sqx` (embedded SQL)

Table 12. C++ Sample Program Files

Sample program name	Program description
Tutorial samples - Programs that demonstrate basic database operations.	
`tut_mod.sqC`	How to modify table data.
`tut_read.sqC`	How to read tables.
`tut_use.sqC`	How to use a database.
Client level - Samples that deal with the client level of DB2.	
`cli_info.C`	How to get and set client level information.
`clisnap.C`	How to capture a snapshot at the client level.
Instance level - Samples that deal with the instance level of DB2.	
`inattach.C`	How to attach to/detach from an instance.
`inauth.sqC`	How to display authorities at instance level.
`ininfo.C`	How to get and set instance level information.
`insnap.C`	How to capture a snapshot at the instance level.
`instart.C`	How to stop and start the current local instance.
Database level - Samples that deal with database objects in DB2.	
`dbauth.sqC`	How to grant/display/revoke authorities at the database level
`dbcfg.sqC`	How to configure database and database manager parameters.
`dbconn.sqC`	How to connect and disconnect from a database.
`dbcreate.C`	How to create and drop databases.
`dbhistfile.sqC`	How to read and update a database recovery file entry.
`dbinfo.C`	How to get and set information at a database level.
`dbinline.sqC`	How to use inline SQL procedure language.
`dbinspec.sqC`	How to check architectural integrity with the DB2 API db2Inspect.
`dblogconn.sqC`	How to read database log files asynchronously with a database connection.

Table 12. C++ Sample Program Files (continued)

Sample program name	Program description
dblognoconn.sqC	How to read database log files asynchronously with no database connection.
dbmcon.sqC	How to connect to and disconnect from multiple databases.
dbmcon1.h	Header file for dbmcon1.sqC
dbmcon1.sqC	Support file for dbmcon.sqC.
dbmcon2.h	Header file for dbmcon2.sqC
dbmcon2.sqC	Support file for dbmcon.sqC.
dbmigrat.C	How to migrate a database.
dbpkg.sqC	How to work with packages.
dbrec.sqC	How to use the db2GetRecommendations API.
dbrecov.sqC	How to recover a database.
dbredirect.sqC	How to recover a database using redirected restore.
dbrestore.sqC	How to recover a database.
dbrollfwd.sqC	How to recover a database using rollforward recovery.
dbsample.sqC	How to create the sample database including Host and AS/400 tables and views.
dbsnap.C	How to capture a snapshot at the database level.
dbthrds.sqC	How to use multiple context APIs on UNIX.
dbthrds.sqC	How to use multiple context APIs on Windows.
dbuse.sqC	How to use database objects.
Table space level - Samples that deal with the table space level of DB2.	
tscreate.sqC	How to create and drop buffer pools and table spaces.
tsinfo.sqC	How to get information at the table space level.
Table level - Samples that deal with table objects in DB2.	
tbconstr.sqC	How to work with table constraints.
tbcreate.sqC	How to create, alter and drop tables.
tbident.sqC	How to use identity columns.
tbinfo.sqC	How to get and set information at a table level.
tbintrig.sqC	How to use an 'INSTEAD OF' trigger on a view.
tbmerge.sqC	How to use the MERGE statement.
tbmod.sqC	How to modify information in a table.
tbmove.sqC	How to move a table data.
tbpriv.sqC	How to grant/display/revoke table level privileges.
tbread.sqC	How to read information in a table.
tbreorg.sqC	How to reorganize a table.
tbsavept.sqC	How to use external savepoints. Also demonstrates how to change the default value of a column.
tbsel.sqC	How to select from each of: insert, update, delete.
tbselcreate.db2	How to create the tables for the tbsel program.
tbseldrop.db2	How to drop the tables for the tbsel program.

Table 12. C++ Sample Program Files (continued)

Sample program name	Program description
tbtemp.sqC	How to use a declared temporary table.
tbtrig.sqC	How to use a trigger on a table.
tbunion.sqC	How to insert through a UNION ALL view.
tbxload.sqC	How to simultaneously return data from a select statement and load it into a table.
Data type level - Samples that deal with data types.	
dtformat.sqC	How to use load and import data format extensions.
dtlob.sqC	How to read and write LOB data.
dtstruct.sqC	How to create, use, and drop a hierarchy of structured types and typed tables.
dtudt.sqC	How to create, use, and drop user-defined distinct types.
DB2 function level	
fnuse.sqC	How to use SQL functions.
Stored procedure level - Samples that demonstrate stored procedures.	
spcat	Stored procedure catalog script for the spserver program. This script calls spdrop.db2 and spcreate.db2.
spcreate.db2	CLP script to issue CREATE PROCEDURE statements.
spdrop.db2	CLP script to drop stored procedures from the catalog.
spclient.sqC	Client program used to call the server routines declared in spserver.sqc, spserver.sqC.
spserver.sqC	Stored procedure routines built and run on the server.
UDF level - Samples that demonstrate user-defined functions.	
udfcli.sqC	Client application which calls the user-defined function in udfsrv.c, udfsrv.C.
udfsrv.C	User-defined function ScalarUDF called by udfcli.sqc, udfcli.sqC.
udfemcli.sqC	Client application which calls the embedded SQL user defined function library udfemsrv.
udfemsrv.sqC	Embedded SQL User-defined function library called by udfemcli.
Other	
evm.sqC	How to create and parse file, pipe, and table event monitors.
utilrecov.C	Utilities for the backup, restore and log file samples.
utilsnap.C	Utilities for the snapshot monitor samples.

For the latest samples updates, visit the DB2 application development samples Web page:

```
http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html
```

Related concepts:

- "Build files" on page 97
- "Makefiles" on page 99
- "Error-checking utilities" on page 102
- "Sample files" on page 59

C# samples

Directory: `sqllib\samples\.NET\cs`.

Table 13. C# .NET sample program files

Sample program name	Program description
Database Level - Samples that deal with database objects in DB2.	
`DbAuth.cs`	How to grant/display/revoke authorities at the database level.
`DbDatAdp.cs`	How to use a DB2DataAdapter.
`DbDatMap.cs`	How to set up DataTable and DataColumn mappings.
`DbDsetCn.cs`	How to add existing constraints to a DataSet.
`DbEvent.cs`	How to handle DB2DataAdapter events.
`DbUse.cs`	How to use database objects.
`DbValue.cs`	How to obtain a single value from a database.
Table Level - Samples that deal with table objects in DB2.	
`TbConstr.cs`	How to work with table constraints.
`TbInfo.cs`	How to get and set information at a table level.
`TbPriv.cs`	How to grant/display/revoke table level privileges.
`TbSel.cs`	How to select from each of: insert, update, delete.
`TbTrig.cs`	How to use a trigger on a table.
`TbUse.cs`	How to manipulate table data and connect to/disconnect from a database.
Data Type Level - Samples that deal with data types.	
`DtLob.cs`	How to use the LOB data type.
Stored Procedures - Samples that demonstrate stored procedures.	
`SpCat.db2`	Drops and creates the procedures implemented in SpServer.cs.
`SpClient.cs`	Client program used to call the stored procedures in SpServer.cs.
`SpCreate.db2`	Creates the external procedures implemented in SpServer.cs.
`SpDrop.db2`	Drops the external procedures created in SpCreate.db2 for C#.
`SpReturn.cs`	Client application that calls the stored procedure EMP_DETAILS and obtains its return value.
`SpServer.cs`	C# external code implementation of procedures created in SpCat.db2.
`EmpDetails.db2`	CLP script that creates a stored procedure named EMP_DETAILS.
User-defined functions - Samples that demonstrate UDFs.	
`UdfCat.db2`	Drops and creates the external UDFs implemented in UdfSrv.cs.
`UdfCli.cs`	Client application that calls the user-defined functions in UdfSrv.cs.
`UdfCreate.db2`	Creates the external UDFs implemented in UdfSrv.cs.
`UdfDrop.db2`	Drops the external UDFs created in udfcreate.db2 for C#.
`UdfSrv.cs`	User-defined scalar functions called by UdfCli.cs.
Loosely Coupled Transactions	
`empcat.bat`	Catalogs the stored procedure EMP_DETAILS for the C# client program, SpReturn.
`LCTrans.cs`	Demonstrates loosely coupled transactions.

Table 13. C# .NET sample program files (continued)

Sample program name	Program description
regCOM.bat	Registers COM+ objects for the C# LCTrans program.
RootCOM.cs	This file is used to create a library assembly RootCOM.dll. LCTrans.cs refers to the classes and methods that are defined in this file.
SubCOM.cs	This file is used to create a library assembly SubCOM.dll. LCTrans.cs refers to the classes and methods that are defined in this file.

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Related concepts:

- "Sample files" on page 59

Related tasks:

- "Building C# .NET applications" on page 245
- "Building Common Language Runtime (CLR) .NET routines" on page 251

CLI samples

UNIX directory: `sqllib/samples/cli`. Windows directory: `sqllib\samples\cli`.

Table 14. Sample CLI program files

Sample program name	Program description
Tutorial samples - Programs that demonstrate basic database operations.	
tut_mod.c	How to modify table data.
tut_read.c	How to read tables.
tut_use.c	How to use a database.
Installation image level - Samples that deal with the installation image level of DB2 and CLI.	
ilinfo.c	How to get and set installation level information (such as the version of the CLI driver).
Client level - Samples that deal with the client level of DB2.	
cli_info.c	How to get and set client level information.
clihandl.c	How to allocate and free handles.
clisqlca.c	How to work with SQLCA data.
Instance level - Samples that deal with the instance level of DB2.	
ininfo.c	How to get and set instance level information.
Database level - Samples that deal with database objects in DB2.	
dbcongui.c	How to connect to a database with a Graphical User Interface (GUI).
dbconn.c	How to connect and disconnect from a database.
dbinfo.c	How to get and set information at a database level.
dbmcon.c	How to connect and disconnect from multiple databases.

Table 14. Sample CLI program files (continued)

Sample program name	Program description
dbmconx.c	How to connect and disconnect from multiple databases with embedded SQL.
dbmconx1.h	Header file for dbmconx1.sqc.
dbmconx1.sqc	Embedded SQL file for the dbmconx program.
dbmconx2.h	Header file for dbmconx2.sqc.
dbmconx2.sqc	Embedded SQL file for the dbmconx program.
dbnative.c	How to translate a statement that contains an ODBC escape clause to a data source specific format.
dbuse.c	How to use database objects.
dbusemx.sqc	How to use database objects with embedded SQL.
dbxamon.c	How to show and roll back indoubt transactions.
Table level - Samples that deal with table objects in DB2.	
tbconstr.c	How to work with table constraints.
tbcreate.c	How to create, alter, and drop tables.
tbinfo.c	How to get and set information at a table level.
tbload.c	How to insert data using the CLI LOAD utility.
tbmod.c	How to modify information in a table.
tbread.c	How to read information in a table.
Data type level - Samples that deal with data types.	
dtinfo.c	How to get information about data types.
dtlob.c	How to read and write LOB data.
dtudt.c	How to create, use, and drop user defined distinct types.
Stored procedure level - Samples that demonstrate stored procedures.	
spcat	Stored procedure catalog script for the spserver program. This script calls spdrop.db2 and spcreate.db2.
spcreate.db2	CLP script to issue CREATE PROCEDURE statements.
spdrop.db2	CLP script to drop stored procedures from the catalog.
spclient.c	Client program used to call the server functions declared in spserver.c.
spserver.c	Stored procedure functions built and run on the server.
spclires.c	Client application that demonstrates the difference between SQLMoreResults and SQLNextResults for multiple result sets.
spcall.c	Client program for calling any stored procedure.
UDF level - Samples that demonstrate user defined functions.	
udfcli.c	Client application which calls the user defined function in udfsrv.c.
udfsrv.c	User defined function ScalarUDF called by udfcli.c.
Common utility files	
utilcli.c	Utility functions used in CLI samples.
utilcli.h	Header file for utility functions used in CLI samples.

For the latest samples updates, visit the DB2 application development samples Web page:

http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html

Related concepts:

- "Build files" on page 97
- "Makefiles" on page 99
- "Error-checking utilities" on page 102
- "Sample files" on page 59

Command Line Processor (CLP) samples

UNIX directory: `sqllib/samples/clp`. Windows directory: `sqllib\samples\clp`.

Table 15. Command Line Processor (CLP) sample scripts.

Sample file name	File description
`autocfg.db2`	How to automatically configure database and database manager configuration parameters based on the Performance Configuration wizard's recommendations.
`const.db2`	Creates a table with a CHECK CONSTRAINT clause.
`cte.db2`	Demonstrates a common table expression.
`flt.db2`	Demonstrates a recursive query.
`healthmon.db2`	How to use Table Functions for Health Monitor Snapshot.
`join.db2`	Demonstrates an outer join of tables.
`onlineload.db2`	How to do online loading using the ALLOW READ ACCESS option.
`stock.db2`	Demonstrates the use of triggers.
`testdata.db2`	Uses DB2 built-in functions such as RAND() and TRANSLATE() to populate a table with randomly generated test data.

For the latest samples updates, visit the DB2 application development samples Web page:

http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html

Related concepts:

- "Sample files" on page 59

Related tasks:

- "Calling procedures from the Command Line Processor (CLP)" on page 130
- "Running Command Line Processor (CLP) scripts" on page 129

COBOL samples

UNIX directories. IBM COBOL: `sqllib/samples/cobol`; Micro Focus COBOL: `sqllib/samples/cobol_mf`.

Windows directories. IBM COBOL: `sqllib\samples\cobol`; Micro Focus COBOL: `sqllib\samples\cobol_mf`.

Note: The COBOL samples are not structured in the DB2 level design used for the C, CLI, C++, C#, Java, Perl, PHP, Visual Basic ADO, and Visual Basic .NET samples.

Table 16. COBOL DB2 API sample programs with no embedded SQL

Sample program	Included APIs
checkerr.cbl	• sqlaintp - Get Error Message • sqlogstt - Get SQLSTATE Message
client.cbl	• sqleqryc - Query Client • sqlesetc - Set Client
d_dbconf.cbl	• sqleatin - Attach • sqledtin - Detach • sqlfddb - Get Database Configuration Defaults
d_dbmcon.cbl	• sqleatin - Attach • sqledtin - Detach • sqlfdsys - Get Database Manager Configuration Defaults
db_udcs.cbl	• sqleatin - Attach • sqlecrea - Create Database • sqledrpd - Drop Database
dbcat.cbl	• sqlecadb - Catalog Database • db2DbDirCloseScan - Close Database Directory Scan • db2DbDirGetNextEntry - Get Next Database Directory Entry • db2DbDirOpenScan - Open Database Directory Scan • sqleuncd - Uncatalog Database
dbcmt.cbl	• sqledcgd - Change Database Comment • db2DbDirCloseScan - Close Database Directory Scan • db2DbDirGetNextEntry - Get Next Database Directory Entry • db2DbDirOpenScan - Open Database Directory Scan • sqleisig - Install Signal Handler
dbconf.cbl	• sqleatin - Attach • sqlecrea - Create Database • sqledrpd - Drop Database • sqlfrdb - Reset Database Configuration • sqlfudb - Update Database Configuration • sqlfxdb - Get Database Configuration
dbinst.cbl	• sqleatcp - Attach and Change Password • sqleatin - Attach • sqledtin - Detach • sqlegins - Get Instance
dbmconf.cbl	• sqleatin - Attach • sqledtin - Detach • sqlfrsys - Reset Database Manager Configuration • sqlfusys - Update Database Manager Configuration • sqlfxsys - Get Database Manager Configuration
dbsnap.cbl	• sqleatin - Attach • sqlmonss - Get Snapshot
dbstart.cbl	• sqlepstart - Start Database Manager

Table 16. COBOL DB2 API sample programs with no embedded SQL (continued)

Sample program	Included APIs
dbstop.cbl	• sqlefrce - Force Application • sqlepstp - Stop Database Manager
dcscat.cbl	• sqlegdad - Catalog DCS Database • sqlegdcl - Close DCS Directory Scan • sqlegdel - Uncatalog DCS Database • sqlegdge - Get DCS Directory Entry for Database • sqlegdgt - Get DCS Directory Entries • sqlegdsc - Open DCS Directory Scan
ebcdicdb.cbl	• sqleatin - Attach • sqlecrea - Create Database • sqledrpd - Drop Database
migrate.cbl	• sqlemgdb - Migrate Database
monreset.cbl	• sqleatin - Attach • sqlmrset - Reset Monitor
monsz.cbl	• sqleatin - Attach • sqlmonss - Get Snapshot • sqlmonsz - Estimate Size Required for sqlmonss() Output Buffer
nodecat.cbl	• sqlectnd - Catalog Node • sqlencls - Close Node Directory Scan • sqlengne - Get Next Node Directory Entry • sqlenops - Open Node Directory Scan • sqleuncn - Uncatalog Node
restart.cbl	• sqlerstd - Restart Database
setact.cbl	• sqlesact - Set Accounting String
sws.cbl	• sqleatin - Attach • sqlmon - Get/Update Monitor Switches

Table 17. COBOL DB2 API embedded SQL sample programs

Sample program	Included APIs
dbauth.sqb	• sqluadau - Get Authorizations
dbstat.sqb	• db2Reorg - Reorganize Table • db2Runstats - Run Statistics
expsamp.sqb	• db2Export - Export • sqluimpr - Import
impexp.sqb	• db2Export - Export • sqluimpr - Import
loadqry.sqb	• db2LoadQuery - Load Query
rebind.sqb	• sqlarbnd - Rebind

Table 17. COBOL DB2 API embedded SQL sample programs (continued)

Sample program	Included APIs
tabscont.sqb	• sqlbctcq - Close Tablespace Container Query • sqlbftcq - Fetch Tablespace Container Query • sqlbotcq - Open Tablespace Container Query • sqlbtcq - Tablespace Container Query • sqlefmem - Free Memory
tabspace.sqb	• sqlbctsq - Close Tablespace Query • sqlbftpq - Fetch Tablespace Query • sqlbgtss - Get Tablespace Statistics • sqlbmtsq - Tablespace Query • sqlbotsq - Open Tablespace Query • sqlbstpq - Single Tablespace Query • sqlefmem - Free Memory
tload.sqb	• db2Export - Export • sqluload - Load • sqluvqdp - Quiesce Tablespaces for Table
tspace.sqb	• sqlbctcq - Close Tablespace Container Query • sqlbctsq - Close Tablespace Query • sqlbftcq - Fetch Tablespace Container Query • sqlbftpq - Fetch Tablespace Query • sqlbgtss - Get Tablespace Statistics • sqlbmtsq - Tablespace Query • sqlbotcq - Open Tablespace Container Query • sqlbotsq - Open Tablespace Query • sqlbstpq - Single Tablespace Query • sqlbstsc - Set Tablespace Containers • sqlbtcq - Tablespace Container Query • sqlefmem - Free Memory

Table 18. COBOL Embedded SQL sample programs with No DB2 APIs

Sample program name	Program description
advsql.sqb	Demonstrates the use of advanced SQL expressions like CASE, CAST, and scalar full selects.
cursor.sqb	Demonstrates the use of a cursor using static SQL.
delet.sqb	Demonstrates static SQL to delete items from a database.
dynamic.sqb	Demonstrates the use of a cursor using dynamic SQL.
joinsql.sqb	Demonstrates using advanced SQL join expressions.
lobeval.sqb	Demonstrates the use of LOB locators and defers the evaluation of the actual LOB data.
lobfile.sqb	Demonstrates the use of LOB file handles.
lobloc.sqb	Demonstrates the use of LOB locators.
openftch.sqb	Demonstrates fetching, updating, and deleting rows using static SQL.
static.sqb	Demonstrates static SQL to retrieve information.

Table 18. COBOL Embedded SQL sample programs with No DB2 APIs (continued)

Sample program name	Program description
tabsql.sqb	Demonstrates the use of advanced SQL table expressions.
trigsql.sqb	Demonstrates using advanced SQL triggers and constraints.
updat.sqb	Demonstrates static SQL to update a database.
varinp.sqb	Demonstrates variable input to Embedded Dynamic SQL statement calls using parameter markers.

For the latest samples updates, visit the DB2 application development samples Web page:

http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html

Related concepts:

- "Build files" on page 97
- "Makefiles" on page 99
- "Error-checking utilities" on page 102
- "Sample files" on page 59

Dynamic reconfiguration samples

Directory (AIX and Solaris): sqllib/samples/DLPAR.

Table 19. Dynamic reconfiguration sample scripts

Sample script name	File description
ibm_db2_sln	This Korn shell Dynamic Reconfiguration script (DR script) for AIX facilitates the use of Dynamic Logical Partitioning (DLPAR) capabilities provided with AIX Version 5.2 running on POWER4-based pSeries systems, such as p690 and p670. These capabilities allow the dynamic adding and removing of resources such as central processing units and memory from active logical partitions without requiring a reboot. This DR script can be called during dynamic reconfiguration events to ensure that the events occur without affecting the operation of DB2. The DB2 configuration is dynamically modified to account for the event. Further information can be found in the DR script itself.
IBM,DB2	This perl script for the Solaris Operating Environment is used by dynamic reconfiguration and coordination tools to provide the interface between DB2 and the Reconfiguration and Coordination Manager (RCM). These tools are provided with Solaris 9 or later running on a 3800 series, or later, machine. The correct use of this script will ensure that DB2 continues to operate even in the case of the removal of hardware resources. Further information can be found in the script itself.

The ibm_db2_sln sample is in the sqllib/samples/DLPAR directory on DB2 for AIX, and the IBM,DB2 sample is in the sqllib/samples/DLPAR directory on DB2 for Solaris.

For the latest samples updates, visit the DB2 application development samples Web page:

http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html

Related concepts:

- "Sample files" on page 59

Related tasks:

- "Configuring parameters dynamically" in the *Administration Guide: Performance*

Related reference:

- "AIX supported development software" on page 8
- "Solaris supported development software" on page 17

JDBC samples

UNIX directory: `sqllib/samples/java/jdbc`.

Windows directory: `sqllib\samples\java\jdbc`.

Table 20. JDBC sample program files

Sample program name	Program Description
Tutorial samples - Programs that demonstrate basic database operations.	
`TutMod.java`	How to modify table data.
`TutRead.java`	How to read tables.
Installation image level - Samples that deal with the installation image level of DB2.	
`IlInfo.java`	How to get and set installation level information.
Database level - Samples that deal with database objects in DB2.	
`DbAuth.java`	How to grant/display/revoke authorities at the database level.
`DbConn.java`	How to connect and disconnect from a database.
`DbInfo.java`	How to get and set information at a database level.
`DbMCon.java`	How to connect and disconnect from multiple databases.
`DbNative.java`	How to translate a statement that contains an ODBC escape clause to a data source specific format.
`DbRsHold.java`	How to use result set cursor holdability in Legacy JDBC Type 2 and Universal JDBC drivers. To compile this sample, you need Java Developer Kit 1.4 or above. To run this sample, you need Java Runtime Environment 1.4 or above.
`DbSeq.java`	How to create, alter and drop a sequence in a database.
`DbUse.java`	How to use database objects.
Table level - Samples that deal with table objects in DB2.	
`TbConstr.java`	How to work with table constraints.
`TbCreate.java`	How to create, alter and drop tables.
`TbGenCol.java`	How to use generated columns.
`TbIdent.java`	How to use Identity Columns.
`TbInfo.java`	How to get and set information at a table level.
`TbInTrig.java`	How to use an 'INSTEAD OF' trigger on a view.
`TbMerge.java`	How to use the MERGE statement.
`TbMod.java`	How to modify information in a table.
`TbPriv.java`	How to grant/display/revoke table level privileges.

Table 20. JDBC sample program files (continued)

Sample program name	Program Description
TbRead.java	How to read information in a table.
TbSel.java	How to select from each of: insert, update, delete.
TbTemp.java	How to use Declared Temporary Tables.
TbTrig.java	How to use a trigger on a table.
TbUnion.java	How to insert through a UNION ALL view.
Data type level - Samples that deal with data types.	
DtInfo.java	How to get information about data types.
DtLob.java	How to read and write LOB data.
DtUdt.java	How to create, use, and drop user-defined distinct types.
Applets - Samples that demonstrate applets.	
Applt.java	How to use applets.
Stored procedures - Samples that demonstrate stored procedures.	
spcat	Stored procedure catalog script for the spserver program. This script calls SpDrop.db2 and SpCreate.db2.
SpCreate.db2	CLP script to issue CREATE PROCEDURE statements.
SpDrop.db2	CLP script to drop stored procedures from the catalog.
SpClient.java	Client program used to call the server functions declared in SpServer.java.
SpServer.java	Stored procedure functions built and run on the server.
UDFs - Samples that demonstrate user-defined functions.	
UDFcli.java	Client application which calls the user-defined function library UDFsrv.
UDFsrv.java	User-defined functions called by UDFcli.java.
udfcat	UDF catalog script for the UDFsrv program. This script calls UDFDrop.db2 and UDFCreate.db2.
UDFDrop.db2	CLP script to drop UDFs from the catalog.
UDFCreate.db2	CLP script to issue CREATE PROCEDURE statements.
UDFjcli.java	Client application which calls the user-defined function library UDFjsrv.
UDFjsrv.java	User-defined functions called by UDFjcli.java.
udfjcat	UDF catalog script for the UDFjsrv program. This script calls UDFjDrop.db2 and UDFjCreate.db2.
UDFjDrop.db2	CLP script to drop UDFs from the catalog.
UDFjCreate.db2	CLP script to issue CREATE PROCEDURE statements.
UDFsCreate.db2	How to catalog the UDFs contained in UDFsqlsv.java
UDFsDrop.db2	How to uncatalog the UDFs contained in UDFsqlsv.java
UDFsqlcl.java	Call the UDFs in UDFsqlsv.java
UDFsqlsv.java	User-Defined Functions with SQL statements called by UDFsqlcl.java
Java beans - Samples that demonstrate Java Bean classes.	
CreateEmployee.java	How to create an employee record.

Table 20. JDBC sample program files (continued)

Sample program name	Program Description
`GeneratePayroll.java`	How to generate payroll reports by department.
Other	
`Util.java`	Utilities for JDBC sample programs.

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Related concepts:

- "Java sample programs" on page 107
- "Sample files" on page 59

Related reference:

- "SQLJ samples" on page 80
- "Java WebSphere samples" on page 82
- "Java plug-in samples" on page 83

SQLJ samples

UNIX directory: `sqllib/samples/java/sqlj`.

Windows directory: `sqllib\samples\java\sqlj`.

Table 21. SQLJ sample program files

Sample program name	Program Description
Tutorial samples - Programs that demonstrate basic database operations.	
`TutMod.sqlj`	How to modify table data.
`TutRead.sqlj`	How to read tables.
Database level - Samples that deal with database objects in DB2.	
`DbAuth.sqlj`	How to grant/display/revoke authorities at the database level.
`DbConn.sqlj`	How to connect and disconnect from a database.
`DbMCon.sqlj`	How to connect and disconnect from multiple databases.
`DbUse.sqlj`	How to use database objects.
Table level - Samples that deal with table objects in DB2.	
`TbConstr.sqlj`	How to work with table constraints.
`TbCreate.sqlj`	How to create, alter and drop tables.
`TbIdent.sqlj`	How to use identity columns.
`TbInfo.sqlj`	How to get and set information at a table level.
`TbMod.sqlj`	How to modify information in a table.
`TbPriv.sqlj`	How to grant/display/revoke table level privileges.
`TbRead.sqlj`	How to read information in a table.
`TbSel.sqlj`	How to select from each of: insert, update, delete.
`TbTrig.sqlj`	How to use a trigger on a table.

Table 21. SQLJ sample program files (continued)

Sample program name	Program Description
Data type level - Samples that deal with data types.	
DtUdt.sqlj	How to create, use, and drop user-defined distinct types.
Applet level - Samples that demonstrate applets.	
Applt.sqlj	How to use applets.
Stored procedure level - Samples that demonstrate stored procedures.	
spcat	Stored procedure catalog script for the SpServer program. This script calls SpDrop.db2 and SpCreate.db2.
SpCreate.db2	CLP script to issue CREATE PROCEDURE statements.
SpDrop.db2	CLP script to drop stored procedures from the catalog.
SpClient.sqlj	Client program used to call the server functions declared in SpServer.sqlj.
SpServer.sqlj	Stored procedure functions built and run on the server.
SpIterat.sqlj	Iterator class file for SpServer.sqlj.
UDF level - Samples that demonstrate user-defined functions.	
UDFcli.sqlj	Client application which calls the user-defined function library UDFsrv.
UDFsrv.java	User-defined functions called by UDFcli.
udfcat	UDF catalog script for the UDFsrv program. This script calls UDFDrop.db2 and UDFCreate.db2.
UDFDrop.db2	CLP script to drop UDFs from the catalog.
UDFCreate.db2	CLP script to issue CREATE PROCEDURE statements.
UDFjcli.sqlj	Client application which calls the user-defined function library UDFjsrv.
UDFjsrv.java	User-defined functions called by UDFjcli.
udfjcat	UDF catalog script for the UDFjsrv program. This script calls UDFjDrop.db2 and UDFjCreate.db2.
UDFjDrop.db2	CLP script to drop UDFs from the catalog.
UDFjCreate.db2	CLP script to issue CREATE PROCEDURE statements.
Java beans - Samples that demonstrate Java Bean classes.	
CreateEmployee.sqlj	How to create an employee record.
GeneratePayroll.sqlj	How to generate payroll reports by department.
Data Source - Samples that demonstrate data sources.	
Batch1Demo.sqlj	SQLJ batching -- How SQLJ batching works.
Batch2Demo.sqlj	SQLJ batching - Association of ExecutionContext with BatchContext.
Batch3Demo.sqlj	SQLJ Batching - When do we need to implicitly execute a batch.
BlobClobDemo.sqlj	How to access Blob or Clob fields in DB2 tables.
createRegisterDS.java	Create and Register DataSources as specified by the DataSource property files.
CreateDemoSchema.sqlj	This program creates the schema for the DataSource Demo programs.
DbConnDataSource.sqlj	How to connect to a database using DataSource with the DB2 Universal JDBC driver.

Table 21. SQLJ sample program files (continued)

Sample program name	Program Description
DbConMDataSources.sqlj	How to connect to a database using Multiple DataSources with the DB2 Universal JDBC driver.
ScrollIterDemo.sqlj	How to use Named and Positional Scrollable Iterators in SQLJ.
Other	
Util.sqlj	Utilities for SQLJ sample programs.

For the latest samples updates, visit the DB2 application development samples Web page:

http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html

Related concepts:

- "Java sample programs" on page 107
- "Sample files" on page 59

Related reference:

- "JDBC samples" on page 78
- "Java WebSphere samples" on page 82
- "Java plug-in samples" on page 83

Java WebSphere samples

UNIX directory: sqllib/samples/java/Websphere.

Windows directory: sqllib\samples\java\Websphere.

Table 22. Java WebSphere sample files

Sample program name	Program description
AccessEmployee.ear	This Enterprise ARchive (.EAR) file consists of four modules containing 32 different .class, .JSP and .HTML files. This EAR file, easily deployed using IBM WebSphere Application Server, demonstrates how Java clients can interact with Enterprise Java Beans (EJBs) to access data stored in DB2.

For the latest samples updates, visit the DB2 application development samples Web page:

http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html

Related concepts:

- "Java sample programs" on page 107
- "Sample files" on page 59

Related reference:

- "JDBC samples" on page 78
- "SQLJ samples" on page 80
- "Java plug-in samples" on page 83

Java plug-in samples

UNIX directory: `sqllib/samples/java/plugin`.

Windows directory: `sqllib\samples\java\plugin`.

Table 23. Java Control Center plug-in sample files

Sample program name	Program description
`Example1.java`	How to add a new toolbar button to the Control Center toolbar.
`Example2.java`	How to add new menu actions to Control Center Database objects.
`Example3.java`	How to add new objects under Database objects in the Control Center tree.
`Example3Child.java`	How to add plug-in objects under Database objects in the Control Center tree.
`Example3Folder.java`	How to add new objects under Database objects in the Control Center tree.

Related concepts:

- "Introducing the plug-in architecture for the Control Center" in the *Administration Guide: Implementation*
- "Compiling and running the example plugins" in the *Administration Guide: Implementation*
- "Java sample programs" on page 107
- "Sample files" on page 59
- "Writing plugins as Control Center extensions" in the *Administration Guide: Implementation*

Related tasks:

- "Creating a plugin that adds a toolbar button" in the *Administration Guide: Implementation*
- "Setting attributes for a plugin tree object" in the *Administration Guide: Implementation*

Related reference:

- "JDBC samples" on page 78
- "SQLJ samples" on page 80
- "Java WebSphere samples" on page 82

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Log management user exit samples

UNIX directory: `sqllib/samples/c`. Windows directory: `sqllib\samples\c`.

Note: Instructions for compiling the Log Management User Exit programs are given at the top of each of the source files listed in the following table.

Table 24. Log management user exit sample program files.

Sample file name	File description
db2uext2.ctsm	This is a sample User Exit utilizing Tivoli Storage Manager (TSM) APIs to archive and retrieve database log files. The sample provides an audit trail of calls (stored in a separate file for each option) including a timestamp and parameters received. It also provides a trail of calls in error including a timestamp and an error isolation string for problem determination. These options can be disabled. The file must be renamed db2uext2.c and compiled as a C program. Available on UNIX and Windows operating systems. **Note:** Applications on AIX using the TSM API Client must be built with the xlc_r or xlC_r compiler invocations, not with xlc or xlC, even if the applications are single-threaded. This ensures that the libraries are thread-safe. If you have an application that is compiled with a non-thread-safe library, you can apply fixtest IC21925E or contact your application provider. The fixtest is available on the index.storsys.ibm.com anonymous ftp server.
db2uext2.cdisk	This is a sample User Exit utilizing the system copy command for the particular platform on which it ships. The program archives and retrieves database log files, and provides an audit trail of calls (stored in a separate file for each option) including a timestamp and parameters received. It also provides an error trail of calls in error including a timestamp and an error isolation string for problem determination. These options can be disabled. The file must be renamed db2uext2.c and compiled as a C program. Available on UNIX and Windows operating systems.
db2uext2.ctape	This is a sample User Exit utilizing system tape commands for the particular UNIX platform on which it ships. The program archives and retrieves database log files. All limitations of the system tape commands are limitations of this user exit. The sample provides an audit trail of calls (stored in a separate file for each option) including a timestamp and parameters received. It also provides an error trail of calls in error including a timestamp and an error isolation string for problem determination. These options can be disabled. The file must be renamed db2uext2.c and compiled as a C program. Available on UNIX platforms only.
db2uext2.cxbsa	This is a sample User Exit utilizing XBSA APIs to Archive and Retrieve database log files. This sample provides an audit trail of calls (stored in a separate file for each option) including a timestamp and parameters received. It also provides an error trail of calls in error including a timestamp and an error isolation string for problem determination. These options can be disabled. The file must be renamed db2uext2.c and compiled as a C program. Available on UNIX platforms only.

For the latest samples updates, visit the DB2 application development samples Web page:

```
http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html
```

Related concepts:

- "Managing log files through log archiving" in the *Data Recovery and High Availability Guide and Reference*
- "Sample files" on page 59

Related reference:

- "Tivoli Storage Manager" in the *Data Recovery and High Availability Guide and Reference*

Object Linking and Embedding (OLE) samples

Directories. Visual Basic: `sqllib\samples\ole\msvb`; Visual C++: `sqllib\samples\ole\msvc`.

Table 25. Object Linking and Embedding (OLE) sample programs

Sample program name	Program description
`sales`	Demonstrates rollup queries on a Microsoft Excel sales spreadsheet (implemented in Visual Basic).
`names`	Queries a Lotus Notes address book (implemented in Visual Basic).
`inbox`	Queries Microsoft Exchange inbox e-mail messages through OLE/Messaging (implemented in Visual Basic).
`invoice`	An OLE automation user-defined function that sends Microsoft Word invoice documents as e-mail attachments (implemented in Visual Basic).
`bcounter`	An OLE automation user-defined function demonstrating a scratchpad using instance variables (implemented in Visual Basic).
`ccount`	A counter OLE automation user-defined function (implemented in Visual C++).
`salsrv`	An OLE automation stored procedure that calculates the median salary of the STAFF table of the `sample` database (implemented in Visual Basic).
`salcltvc`	A Visual C++ DB2 CLI sample that calls the Visual Basic stored procedure, `salsrv`.
`salcltvb`	A Visual Basic DB2 CLI sample that calls the Visual Basic stored procedure, `salsrv`.
`salsvado`	An OLE automation stored procedure, implemented in 32-bit Visual Basic and ADO, that demonstrates output parameters by calculating the median salary in newly-created table, STAFF2, and demonstrates result sets by retrieving salaries from the table.
`salclado`	A Visual Basic client that calls the Visual Basic stored procedure, `salsvado`.
`testcli`	An OLE automation embedded SQL client application that calls the stored procedure, `tstsrv` (implemented in Visual C++).
`tstsrv`	An OLE automation stored procedure demonstrating the passing of various types between client and stored procedure (implemented in Visual C++).

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Related concepts:

- "Sample files" on page 59

Related reference:

- "Object Linking and Embedding Database (OLE DB) table function samples" on page 86

Object Linking and Embedding Database (OLE DB) table function samples

Directory: `sqllib\samples\oledb`.

Table 26. Object Linking and Embedding Database (OLE DB) table functions

Sample program name	Program description
`inora.db2`	INTERSOLV Oracle8 OLE DB Provider
`jet.db2`	Microsoft.Jet.OLEDB.3.51 Provider
`jetsrv.db2`	Federated database functionality with Microsoft.Jet.OLEDB.4.0 Provider.
`mapi.db2`	INTERSOLV Connect OLE DB for MAPI
`msdaora.db2`	Microsoft OLE DB Provider for Oracle
`msdasql.db2`	Microsoft OLE DB Provider for ODBC Drivers
`msidxs.db2`	Microsoft OLE DB Index Server Provider
`notes.db2`	INTERSOLV Connect OLE DB for Notes
`sampprov.db2`	Microsoft OLE DB Sample Provider
`sqloledb.db2`	Microsoft OLE DB Provider for SQL Server

Related concepts:

- "Sample files" on page 59

Related reference:

- "Object Linking and Embedding (OLE) samples" on page 85

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Perl Samples

UNIX directory: `sqllib/samples/perl`.

Windows directory: `sqllib\samples\perl`.

Table 27. Perl sample program files

Sample program name	Program description
Database Level - Samples that deal with database objects in DB2.	
`dbauth.pl`	How to grant/display/revoke authorities at the database level.
`dbuse.pl`	How to use a database.
Table Level - Samples that deal with table objects in DB2.	
`tbconstr.pl`	How to create, use, and drop constraints.
`tbinfo.pl`	How to get information about a table at the table level.
`tbpriv.pl`	How to grant, display and revoke privileges on a table.
`tbsel.pl`	How to select from each of: insert, update, delete.
`tbselcreate.pl`	How to create the tables for the tbsel program.

Table 27. Perl sample program files (continued)

Sample program name	Program description
tbseldrop.pl	How to drop the tables for the tbsel program.
tbtrig.pl	How to use a trigger on a table.
tbuse.pl	How to perform basic database operations and connect to/disconnect from a database.
Data Type Level - Samples that deal with data types.	
dtlob.pl	How to use the LOB data type.
Stored Procedures - Samples that demonstrate stored procedures.	
spclient.pl	Client program containing ten functions to call stored procedures.
Other sample files	
DB2SampUtil.pm	Defines common functions like command line argument checking. Also defines functions to prepare and execute an SQL statement, and to roll back if an error occurs.

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Related concepts:

- "Programming Considerations for Perl" in the *Application Development Guide: Programming Client Applications*
- "Sample files" on page 59

Related tasks:

- "Building Perl applications" on page 139

PHP samples

UNIX directory: `sqllib/samples/php`.

Windows directory: `sqllib\samples\php`.

Table 28. PHP sample program files

Sample program name	Program description
Database Level - Samples that deal with database objects in DB2.	
dbauth.php	How to grant/display/revoke authorities at the database level.
dbuse.php	How to use a database.
Table Level - Samples that deal with table objects in DB2.	
tbconstr.php	How to create, use, and drop constraints.
tbinfo.php	How to get information about a table at the table level.
tbpriv.php	How to grant, display and revoke privileges on a table.
tbsel.php	How to select from each of: insert, update, delete.
tbselcreate.db2	How to create the tables for the tbsel program.
tbseldrop.db2	How to drop the tables for the tbsel program.

Table 28. PHP sample program files (continued)

Sample program name	Program description
tbtrig.php	How to use a trigger on a table.
tbuse.php	How to perform basic database operations and connect to/disconnect from a database.
Data Type Level - Samples that deal with data types.	
dtlob.php	How to use the LOB data type.
User-defined functions - Samples that demonstrate user-defined functions.	
udfcli.php	Client program calling a variety of types of user-defined functions.
Other sample files	
util_funcs.php	Defines common functions like command line argument checking. Also defines functions to prepare and execute an SQL statement, and to roll back if an error occurs.

For the latest samples updates, visit the DB2 application development samples Web page:

```
http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html
```

Related concepts:

- "Sample files" on page 59

Related tasks:

- "Building PHP applications" on page 141

REXX samples

AIX directory: `sqllib/samples/rexx`. Windows directory: `sqllib\samples\rexx`.

Table 29. REXX sample program files.

Sample file name	File description
blobfile.cmd	Demonstrates Binary Large OBject (BLOB) manipulation.
chgisl.cmd	Demonstrates the CHANGE ISOLATION LEVEL API.
client.cmd	Demonstrates the SET CLIENT and QUERY CLIENT APIs.
d_dbconf.cmd	Demonstrates the API: GET DATABASE CONFIGURATION DEFAULTS
d_dbmcon.cmd	Demonstrates the API: GET DATABASE MANAGER CONFIGURATION DEFAULTS
db_udcs.cmd	Demonstrates the CREATE DATABASE and DROP DATABASE APIs to simulate the collating behavior of a DB2 for MVS/ESA CCSID 500 (EBCDIC International) collating sequence
dbauth.cmd	Demonstrates the GET AUTHORIZATIONS API

dbcat.cmd	Demonstrates the following APIs:
	`CATALOG DATABASE`
	`CLOSE DATABASE DIRECTORY SCAN`
	`GET NEXT DATABASE DIRECTORY ENTRY`
	`OPEN DATABASE DIRECTORY SCAN`
	`UNCATALOG DATABASE`

Table 29. REXX sample program files. (continued)

Sample file name	File description
dbcmt.cmd	Demonstrates the following APIs: CHANGE DATABASE COMMENT GET ERROR MESSAGE INSTALL SIGNAL HANDLER
dbconf.cmd	Demonstrates the following APIs: CREATE DATABASE DROP DATABASE GET DATABASE CONFIGURATION RESET DATABASE CONFIGURATION UPDATE DATABASE CONFIGURATION
dbinst.cmd	Demonstrates the following APIs: ATTACH TO INSTANCE DETACH FROM INSTANCE GET INSTANCE
dbmconf.cmd	Demonstrates the following APIs: GET DATABASE MANAGER CONFIGURATION RESET DATABASE MANAGER CONFIGURATION UPDATE DATABASE MANAGER CONFIGURATION
dbstart.cmd	Demonstrates the START DATABASE MANAGER API
dbstat.cmd	Demonstrates the following APIs: REORGANIZE TABLE RUN STATISTICS
dbstop.cmd	Demonstrates the following APIs: FORCE USERS STOP DATABASE MANAGER
dcscat.cmd	Demonstrates the following APIs: ADD DCS DIRECTORY ENTRY CLOSE DCS DIRECTORY SCAN GET DCS DIRECTORY ENTRY FOR DATABASE GET DCS DIRECTORY ENTRIES OPEN DCS DIRECTORY SCAN UNCATALOG DCS DIRECTORY ENTRY
dynamic.cmd	Demonstrates the use of a ″CURSOR″ using dynamic SQL
ebcdicdb.cmd	Demonstrates the CREATE DATABASE and DROP DATABASE APIs to simulate the collating behavior of a DB2 for MVS/ESA CCSID 037 (EBCDIC US English) collating sequence
impexp.cmd	Demonstrates the EXPORT and IMPORT APIs
lobeval.cmd	Demonstrates deferring the evaluation of a LOB within a database
lobfile.cmd	Demonstrates the use of LOB file handles
lobloc.cmd	Demonstrates the use of LOB locators
lobval.cmd	Demonstrates the use of LOBs
migrate.cmd	Demonstrates the MIGRATE DATABASE API
nodecat.cmd	Demonstrates the following APIs: CATALOG NODE CLOSE NODE DIRECTORY SCAN GET NEXT NODE DIRECTORY ENTRY OPEN NODE DIRECTORY SCAN UNCATALOG NODE
quitab.cmd	Demonstrates the API: QUIESCE TABLESPACES FOR TABLE

Table 29. REXX sample program files. (continued)

Sample file name	File description

rechist.cmd	Demonstrates the following APIs:
	`CLOSE RECOVERY HISTORY FILE SCAN`
	`GET NEXT RECOVERY HISTORY FILE ENTRY`
	`OPEN RECOVER HISTORY FILE SCAN`
	`PRUNE RECOVERY HISTORY FILE ENTRY`
	`UPDATE RECOVERY HISTORY FILE ENTRY`

restart.cmd	Demonstrates the RESTART DATABASE API
sqlecsrx.cmd	An example of a collating sequence
updat.cmd	Uses dynamic SQL to update a database

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Related concepts:

- "Sample files" on page 59

Related tasks:

- "Building REXX applications on AIX" on page 191
- "Building Object REXX applications on Windows" on page 282

Security plug-in samples

UNIX directory: `sqllib/samples/security/plugins`.

Windows directory: `sqllib\samples\security\plugins`.

Table 30. Security plug-in sample program files

Sample program name	Program description
combined.c	Combined userid/password authentication and group lookup sample.
group_file.c	Simple file-based group management plug-in sample.
gssapi_simple.c	Basic GSS-API authentication plug-in sample (both client and server).
IBMkrb5.c	Source code for the IBM-supplied Kerberos security plug-in for UNIX.

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Related concepts:

- "Sample files" on page 59
- "Security plug-ins" in the *Application Development Guide: Programming Client Applications*

SQL procedure samples

UNIX directory: `sqllib/samples/sqlproc`.

Windows directory: `sqllib\samples\sqlproc`.

Table 31. SQL procedure sample program files

Sample program name	Program description
`basecase.db2`	The UPDATE_SALARY procedure raises the salary of an employee identified by the ″empno″ IN parameter in the ″staff″ table of the ″sample″ database. The procedure determines the raise according to a CASE statement that uses the ″rating″ IN parameter.
`basecase.sqc`	Calls the UPDATE_SALARY procedure.
`baseif.db2`	The UPDATE_SALARY_IF procedure raises the salary of an employee identified by the ″empno″ IN parameter in the ″staff″ table of the ″sample″ database. The procedure determines the raise according to an IF statement that uses the ″rating″ IN parameter.
`baseif.sqc`	Calls the UPDATE_SALARY_IF procedure.
`dynamic.db2`	The CREATE_DEPT_TABLE procedure uses dynamic DDL to create a new table. The name of the table is based on the value of the IN parameter to the procedure.
`dynamic.sqc`	Calls the CREATE_DEPT_TABLE procedure.
`iterate.db2`	The ITERATOR procedure uses a FETCH loop to retrieve data from the ″department″ table. If the value of the ″deptno″ column is not 'D11', modified data is inserted into the ″department″ table. If the value of the ″deptno″ column is 'D11', an ITERATE statement passes the flow of control back to the beginning of the LOOP statement.
`iterate.sqc`	Calls the ITERATOR procedure.
`leave.db2`	The LEAVE_LOOP procedure counts the number of FETCH operations performed in a LOOP statement before the ″not_found″ condition handler invokes a LEAVE statement. The LEAVE statement causes the flow of control to exit the loop and complete the stored procedure.
`leave.sqc`	Calls the LEAVE_LOOP procedure.
`loop.db2`	The LOOP_UNTIL_SPACE procedure counts the number of FETCH operations performed in a LOOP statement until the cursor retrieves a row with a space (' ') value for column ″midinit″. The loop statement causes the flow of control to exit the loop and complete the stored procedure.
`loop.sqc`	Calls the LOOP_UNTIL_SPACE procedure.
`nestcase.db2`	The BUMP_SALARY procedure uses nested CASE statements to raise the salaries of employees in a department identified by the dept IN parameter from the ″staff″ table of the ″sample″ database.
`nestcase.sqc`	Calls the BUMP_SALARY procedure.
`nestif.db2`	The BUMP_SALARY_IF procedure uses nested IF statements to raise the salaries of employees in a department identified by the dept IN parameter from the ″staff″ table of the ″sample″ database.
`nestif.sqc`	Calls the BUMP_SALARY_IF procedure.
`nestedsp.db2`	The OUT_AVERAGE, OUT_MEDIAN, and MAX_SALARY procedures return average, median and max values from the ″staff″ table of the sample database.
`nestedspdrop.db2`	Drops the OUT_AVERAGE, OUT_MEDIAN and MAX_SALARY SQL procedures that are created with the nestedsp.db2 script.
`NestedSP.java`	Calls the OUT_AVERAGE procedure.

Table 31. SQL procedure sample program files (continued)

Sample program name	Program description
repeat.db2	The REPEAT_STMT procedure counts the number of FETCH operations performed in a repeat statement until the cursor can retrieve no more rows. The condition handler causes the flow of control to exit the repeat loop and complete the stored procedure.
repeat.sqc	Calls the REPEAT_STMT procedure.
rsultset.c	Calls the MEDIAN_RESULT_SET procedure, displays the median salary, then displays the result set generated by the SQL procedure. This client is written using the CLI API, which can accept result sets.
rsultset.db2	The MEDIAN_RESULT_SET procedure obtains the median salary of employees in a department identified by the "dept" IN parameter from the "staff" table of the "sample" database. The median value is assigned to the salary OUT parameter and returned to the "rsultset" client. The procedure then opens a WITH RETURN cursor to return a result set of the employees with a salary greater than the median. The procedure returns the result set to the client.
spserver.db2	The SQL procedures in this CLP script demonstrate basic error-handling, nested stored procedure calls, and returning result sets to the client application or the calling application. You can call the procedures using the "spcall" application, in the CLI samples directory. You can also use the "spclient" application, in the C and CPP samples directories, to call the procedures that do not return result sets.
tbfn.db2	Creates the tables and table functions used in the tbfnuse sample. After the tbfnuse script is run, all changes are rolled back and the tables and functions created in this file are dropped.
tbfnuse.db2	Demonstrates the use of table functions created in the tbfnuse sample. At the end of this script, statements are rolled back and the tables and functions created in tbfn.db2 are dropped.
tbsel.sqc	How to select from each of: insert, update, delete. This sample calls an SQL procedure, BUY_COMPANY, created from tbselcreate.db2. BUY_COMPANY contains usage examples of a SELECT from a data change statement.
tbselcreate.db2	How to create the tables and the procedure used in the tbsel program.
tbseldrop.db2	How to drop the tables and the procedure used in the tbsel program.
whiles.db2	The DEPT_MEDIAN procedure obtains the median salary of employees in a department identified by the "dept" IN parameter from the "staff" table of the "sample" database. The median value is assigned to the salary OUT parameter and returned to the "whiles" client. The whiles client then prints the median salary.
whiles.sqc	Calls the DEPT_MEDIAN procedure.

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Related concepts:

- "Sample files" on page 59

Related tasks:

- "Creating SQL procedures" on page 133

Visual Basic samples

Directories. ActiveX Data Objects: `sqllib\samples\VB\ADO`; Microsoft Transaction Server: `sqllib\samples\VB\MTS`; Remote Data Objects: `sqllib\samples\VB\RDO`.

Table 32. Visual Basic ActiveX Data Objects sample program files

Sample program name	Program description
Client level	
`cliExeSQL.bas`	How to execute SQL statements.
`cli_Info.bas`	How to get/set client level information.
Database level	
`dbConn.bas`	How to connect/disconnect from a database.
`dbInfo.bas`	How to get and set information at a database level.
`dbCommit.bas`	How to control autocommit dynamically on the database level.
Data type level	
`dtHier.bas`	How to retrieve hierarchical data.
`dtLob.bas`	How to get LOB data.
Stored procedures	
`spCall.bas`	How to call stored procedures.
User-defined functions	
`udfUse.bas`	How to create and work with UDTs and UDFs.

Table 33. Visual Basic Microsoft Transaction Server sample program files

Sample program name	Program description
`db2com.vbp`	This Visual Basic project demonstrates updating a database using the Microsoft Transaction Server. It creates a server DLL used by the client program, `db2mts.vbp`, and has four class modules: • UpdateNumberColumn.cls • UpdateRow.cls • UpdateStringColumn.cls • VerifyUpdate.cls For this program a temporary table, `DB2MTS`, is created in the `sample` database.
`db2mts.vbp`	This is a Visual Basic project for a client program that uses the Microsoft Transaction Server to call the server DLL created from `db2com.vbp`.
`LCTransTest.vbp`	This Visual Basic project demonstrates a loosely-coupled transaction on a DB2 database. It creates a server DLL used by the client program, main.vbp, and has one class module, `TestClass.cls`. A temporary table, LCTEST, is created in the sample database.
`main.vbp`	This is a Visual Basic project for a client program that uses the Microsoft Transaction Server to call the server DLL created from LCTransTest.vbp.

Table 34. Visual Basic Remote Data Objects sample program files

Sample Program Name	Program Description
Bank.vbp	An RDO program to create and maintain data for bank branches, with the ability to perform transactions on customer accounts. The program can use any database specified by the user as it contains the DDL to create the necessary tables for the application to store data.

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Related concepts:

- "Sample files" on page 59

Related tasks:

- "Building ADO applications with Visual Basic" on page 237
- "Building RDO applications with Visual Basic" on page 243
- "Building loosely-coupled transactions with Visual Basic" on page 240
- "Troubleshooting a Visual Basic loosely-coupled transaction project" on page 242

Related reference:

- "Windows Management Instrumentation samples" on page 96

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Visual Basic .NET samples

Directory: `sqllib\samples\.NET\vb`.

Table 35. Sample Visual Basic .NET program files

Sample program name	Program description
Database level - Samples that deal with database objects in DB2.	
DbAuth.vb	How to grant/display/revoke authorities at the database level.
DbDatAdp.vb	How to use a DB2DataAdapter.
DbDatMap.vb	How to set up DataTable and DataColumn mappings.
DbDsetCn.vb	How to add existing constraints to a DataSet.
DbEvent.vb	How to handle DB2DataAdapter events.
DbUse.vb	How to use database objects.
DbValue.vb	How to obtain a single value from a database.
Table level - Samples that deal with table objects in DB2.	
TbConstr.vb	How to work with table constraints.
TbInfo.vb	How to get and set information at a table level.
TbPriv.vb	How to grant/display/revoke table level privileges.
TbSel.vb	How to select from each of: insert, update, delete.

Table 35. Sample Visual Basic .NET program files (continued)

Sample program name	Program description
TbTrig.vb	How to use a trigger on a table.
TbUse.vb	How to manipulate table data and connect to/disconnect from a database.
Data type level - Samples that deal with data types.	
DtLob.vb	How to use the LOB data type.
Stored procedures - Samples that demonstrate stored procedures.	
SpCat.db2	Drops and creates the procedures implemented in SpServer.vb.
SpClient.vb	Client program used to call the stored procedures in SpServer.vb.
SpCreate.db2	Creates the external procedures implemented in SpServer.vb.
SpDrop.db2	Drops the external procedures created in SpCreate.db2.
SpReturn.vb	Client application that calls the stored procedure EMP_DETAILS and obtains its return value.
SpServer.vb	Visual Basic .NET external code implementation of procedures created in SpCat.db2.
EmpDetails.db2	CLP script that creates a stored procedure named EMP_DETAILS.
User-defined functions - Samples that demonstrate UDFs.	
UdfCat.db2	Drops and creates the external UDFs implemented in UdfSrv.vb.
UdfCli.vb	Client application that calls the user-defined functions in UdfSrv.vb.
UdfCreate.db2	Creates the external UDFs implemented in UdfSrv.vb.
UdfDrop.db2	Drops the external UDFs created in udfcreate.db2.
UdfSrv.vb	User-defined scalar functions called by UdfCli.
Loosely coupled transactions	
empcat.bat	Catalogs the stored procedure EMP_DETAILS for the client program, SpReturn.
LCTrans.vb	Demonstrates loosely coupled transactions.
regCOM.bat	Registers COM+ objects for the LCTrans program.
RootCOM.vb	This file is used to create a library assembly RootCOM.dll. LCTrans.vb refers to the classes and methods that are defined in this file.
SubCOM.vb	This file is used to create a library assembly SubCOM.dll. LCTrans.vb refers to the classes and methods that are defined in this file.

Related concepts:

- "Sample files" on page 59

Related tasks:

- "Building Visual Basic .NET applications" on page 248
- "Building Common Language Runtime (CLR) .NET routines" on page 251

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Visual C++ samples

Directory: `sqllib\samples\VC\ADO`.

Table 36. Visual C++ sample program files

Sample program name	Program description
`BLOBAccess.dsw`	This sample demonstrates highlighting ADO/Blob access using Microsoft Visual C++. It is similar to the Visual Basic sample, `Blob.vbp`. The BLOB sample has two main functions: 1. Read a BLOB from the Sample database and display it to the screen. 2. Read a BLOB from a file and insert it into the database. (Import)
`VarCHAR.dsp`	A Visual C++ program that uses ADO to access VarChar data as textfields. It provides a graphical user interface to allow users to view and update data in the `ORG` table of the `sample` database.

Related concepts:

- "Object Linking and Embedding (OLE) automation with Visual C++" on page 258
- "Sample files" on page 59

Related tasks:

- "Building ADO applications with Visual C++" on page 256

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Windows Management Instrumentation samples

Directory: `sqllib\samples\wmi`.

Table 37. Windows Management Instrumentation sample program files.

Sample file name	File description
`backupdb.vbs`	How to backup a database
`createdb.vbs`	How to create and drop a database.
`listsvr.vbs`	How to enumerate server instances and start/stop a DB2 instance.
`perfmon.mof`	MOF file for perfmon.vbs.
`perfmon.vbs`	How to obtain a DB2 performance counter. Note: you must run "mofcomp perfmon.mof" first.
`regvar.mof`	MOF file for regvar.vbs.
`regvar.vbs`	How to obtain a DB2 registry variable. Note: you must run "mofcomp regvar.mof" first.
`restoredb.vbs`	How to restore a database.
`rollfwddb.vbs`	How to rollforward a database.
`updatedbcfg.vbs`	How to get and update the database configuration.
`updatedbmcfg.vbs`	How to get and update the database manager configuration.

Related concepts:

- "Sample files" on page 59
- "Windows Management Instrumentation (WMI)" on page 237

Related reference:

- "Visual Basic samples" on page 93

For the latest samples updates, visit the DB2 application development samples Web page:

`http://www.ibm.com/software/data/db2/udb/ad/v8/samples.html`

Build files, makefiles, and error-checking utilities

Build files

The files used to demonstrate building sample programs are known as script files on UNIX® and batch files on Windows®. We refer to them, generically, as build files. They contain the recommended compile and link commands for supported platform compilers.

Build files are provided by DB2® for each language on supported platforms where the types of programs they build are available, in the same directory as the sample programs for each language. The following table lists the different types of build files for building different types of programs. These build files, unless otherwise indicated, are for supported languages on all supported platforms. The build files have the `.bat` (batch) extension on Windows, which is not included in the table. There is no extension for UNIX platforms.

Table 38. DB2 build files

Build file	Types of programs built
bldapp	Application programs
bldrtn	Routines (stored procedures and UDFs)
bldsqlj	Java™ SQLJ applications
bldsqljs	Java SQLJ routines (stored procedures and UDFs)
bldmc	C/C++ multi-connection applications
bldmt	C/C++ multi-threaded applications
bldcli	CLI client applications for SQL procedures in the sqlproc samples sub-directory.

Note: The `bldcli` file is the same as the `bldapp` file in the `samples/cli` directory. It was given a different name because the embedded C `bldapp` file is also included in the `samples/sqlproc` directory.

The following table lists the build files by platform and programming language, and the directories where they are located. In the online documentation, the build file names are hot-linked to the source files in HTML. The user can also access the text files in the appropriate samples directories.

Table 39. Build files by language and platform

Platform —> Language	AIX®	HP-UX	Linux	Solaris	Windows
C samples/c	bldapp bldrtn bldmt bldmc	bldapp bldrtn bldmt bldmc	bldapp bldrtn bldmt bldmc	bldapp bldrtn bldmt bldmc	bldapp.bat bldrtn.bat bldmt.bat bldmc.bat
C++ samples/cpp	bldapp bldrtn bldmt bldmc	bldapp bldrtn bldmt bldmc	bldapp bldrtn bldmt bldmc	bldapp bldrtn bldmt bldmc	bldapp.bat bldrtn.bat bldmt.bat bldmc.bat
CLI samples/cli	bldapp bldrtn bldmc	bldapp bldrtn bldmc	bldapp bldrtn bldmc	bldapp bldrtn bldmc	bldapp.bat bldrtn.bat bldmc.bat
SQLJ samples/java/sqlj	bldsqlj bldsqljs	bldsqlj bldsqljs	bldsqlj bldsqljs	bldsqlj bldsqljs	bldsqlj.bat bldsqljs.bat
IBM® COBOL samples/cobol	bldapp bldrtn	n/a	n/a	n/a	bldapp.bat bldrtn.bat
Micro Focus COBOL samples/cobol_mf	bldapp bldrtn	bldapp bldrtn	bldapp bldrtn	bldapp bldrtn	bldapp.bat bldrtn.bat
C# samples\.NET\cs	n/a	n/a	n/a	n/a	bldapp.bat bldrtn.bat
Visual Basic .NET samples\.NET\vb	n/a	n/a	n/a	n/a	bldapp.bat bldrtn.bat

The build files are used in the documentation for building applications and routines because they demonstrate very clearly the compile and link options that DB2 recommends for the supported compilers. There are generally many other compile and link options available, and users are free to experiment with them. See your compiler documentation for all the compile and link options provided. Besides building the sample programs, developers can also build their own programs with the build files. The sample programs can be used as templates that can be modified by users to assist their programming development.

Conveniently, the build files are designed to build a source file with any file name allowed by the compiler. This is unlike the makefiles, where the program names are hardcoded into the file. The makefiles access the build files for compiling and linking the programs they make. The build files use the $1 variable on UNIX and the %1 variable on Windows operating systems to substitute internally for the program name. Incremented numbers for these variable names substitute for other arguments that might be required.

The build files allow for quick and easy experimentation, as each one is suited to a specific kind of program-building, such as stand-alone applications, routines (stored procedures and UDFs) or more specialized program types such as multi-connection or multi-threaded programs. Each type of build file is provided wherever the specific kind of program it is designed for is supported by the compiler.

The object and executable files produced by a build file are automatically overwritten each time a program is built, even if the source file is not modified.

This is not the case when using a makefile. It means a developer can rebuild an existing program without having to delete previous object and executable files, or modifying the source.

The build files contain a default setting for the sample database. If the user is accessing another database, he or she can simply supply another parameter to overwrite the default. If they are using the other database consistently, they could hardcode this database name, replacing `sample`, within the build file itself.

For embedded SQL programs, except when using the IBM COBOL precompiler on Windows, the build files call another file, `embprep`, that contains the precompile and bind steps for embedded SQL programs. These steps might require the optional parameters for user ID and password, depending on where the embedded SQL program is being built.

Except for SQLJ, if a developer is building the program on a server instance where the database is located, then the user ID and password will be common to both, and therefore will not need to be provided. On the other hand, if a developer is in a different instance, such as on a client machine accessing a server database remotely, then these parameters would have to be provided.

The SQLJ build files require user ID and password for the `db2sqljcustomize` customizer, even for accessing a local database. This follows the conventions of the DB2 Universal JDBC driver.

Finally, the build files can be modified by the developer for his or her convenience. Besides changing the database name in the build file (explained above) the developer can easily hardcode other parameters within the file, change compile and link options, or change the default DB2 instance path. The simple, straightforward, and specific nature of the build files makes tailoring them to your needs an easy task.

Related concepts:

- "Makefiles" on page 99
- "Error-checking utilities" on page 102
- "Sample files" on page 59

Related reference:

- "AIX supported development software" on page 8
- "HP-UX supported development software" on page 10
- "Linux supported development software" on page 12
- "Solaris supported development software" on page 17
- "Windows supported development software" on page 19

Makefiles

Each samples directory for a supported compiler includes a makefile for building most of the supplied sample programs within the directory. The makefile calls the build files to compile and link each program. The syntax for the makefiles and the output from their commands differ in some important respects from the build files. However, by using the makefile as the 'front-end' for the build files, the user is able to exploit the makefile's simple and powerful commands:

make <program_name>
Compiles and links the program specified.

make all
Compiles and links all programs listed in the makefile.

make clean
Deletes all intermediate files, such as object files, for all programs listed in the makefile.

make cleanall
Deletes all intermediate and executable files for all programs listed in the makefile.

Java™ does not usually use makefiles, and make executables are not shipped with Java Developer Kits. However, DB2® provides makefiles as an option for the Java samples, in case the user wants the convenience of the make commands. To use the Java makefiles, you must have a make executable available that is normally used with another language compiler.

Here are the makefiles by platform provided by DB2 for the main programming languages/APIs, and the sample directories where they are placed. These are hot-linked in the online documentation, and the sample programs that they build are linked within them. These files can also be accessed in the sample directories.

Table 40. Sample makefiles by platform

Platform —> Language	**AIX®**	**HP-UX**	**Linux**	**Solaris**	**Windows®**
C samples/c	makefile	makefile	makefile	makefile	makefile
C++ samples/cpp	makefile	makefile	makefile	makefile	makefile
CLI samples/cli	makefile	makefile	makefile	makefile	makefile
JDBC samples/java/jdbc	makefile	makefile	makefile	makefile	makefile
SQLJ samples/java/sqlj	makefile	makefile	makefile	makefile	makefile
IBM® COBOL samples/cobol	makefile	n/a	n/a	n/a	makefile
Micro Focus COBOL samples/cobol_mf	makefile	makefile	makefile	makefile	makefile
SQL procedures samples/sqlproc	makefile	makefile	makefile	makefile	makefile
C# samples\.NET\cs	n/a	n/a	n/a	n/a	makefile
Visual Basic .NET samples\.NET\vb	n/a	n/a	n/a	n/a	makefile

Unlike the build files, the makefiles will not overwrite existing intermediate and executable files for programs listed within it. This makes it faster, using the `make all` command, to create executables for some of the files if other files already have executables, as `make all` will just ignore these files. But it also assumes the need

for the `make clean` and `make cleanall` commands, to get rid of existing object and executable files when they are not needed.

The makefiles can be used for program development. Because they require hardcoding the program name within the file itself, you might consider the makefiles less convenient to use then the build files, but if you want the power and convenience of the make commands, this is a route to consider.

The makefiles organize the programs they call under several client and server program categories represented by variables (see the makefiles for details). If you are adding a program to a makefile, make sure you add it to be accessed by the correct variables. For example, a program that can run on any client (local to the server or remote) is placed under the `client_run` variable.

You also need to specify the program name under the `cleanall` variable to be sure that the executable produced can be deleted by the `make cleanall` command. Also, if it is an embedded SQL program, specify the non-embedded SQL file created as a result of precompilation under the `clean` variable so that the `make clean` command (as well as the `make cleanall` command which calls it) will delete the non-embedded SQL file.

In addition, you need to specify the new file with the correct syntax to call the appropriate build file to compile and link the program.

To appreciate where a new file needs to be added to one of the sample makefiles, here are all the places where the embedded SQL program, `dbauth`, is located in the AIX C makefile:

```
#############################################################################
#                  2f - make client_run
#############################################################################

client_run : \
       cli_info clisnap \
       dbauth dbconn dbcreate dbinfo dbmcon \
. . .
#############################################################################
#                  2g - make clean
#############################################################################

clean :
       $(ERASE) *.o
       $(ERASE) *.DEL *.TXT *.MSG
       $(ERASE) dbauth.c dbcfg.c dbconn.c dbmcon.c dbmcon1.c dbmcon2.c
. . .
#############################################################################
#                  2h - make cleanall
#############################################################################

cleanall : \
       clean
       $(ERASE) *.bnd
       $(ERASE) cli_info clisnap
       $(ERASE) dbauth dbcfg dbconn dbcreate dbinfo dbmcon dbmcon1 dbmcon2
. . .
#############################################################################
#                  3b -  regular samples, embedded SQL
#############################################################################

dbauth :
       $(BLDAPP) dbauth $(ALIAS) $(UID) $(PWD)
```

The three variables following the program name in the last line above, ALIAS, UID, and PWD, represent, respectively, the database alias name, the user ID, and the password for the database. These variables are passed to the build file (in this case, the bldapp build file represented by the BLDAPP variable). If the program uses embedded SQL, ALIAS, UID, and PWD are in turn passed to the embprep precompile and bind script, which the build file calls. Before using the makefile, you might need to change the values for these variables. By default, ALIAS is set to the sample database, and UID and PWD have no value set.

UID and PWD, are optional parameters that do not need to be set if the user is already working in the same instance as the server database. However, if this is not the case, for example, if the user is remotely connecting to the server from a client machine, then he or she will need to modify the makefile to give the correct values to the UID and PWD variables in order to access the database.

For multi-connection programs, the C, CLI, and C++ makefiles also have a second database alias, ALIAS2, which by default is set to the sample2 database. The corresponding user ID and password variables, UID2 and PWD2 have no value set. As with the UID and PWD variables, they do not need a value if the second database is accessed locally.

The makefiles also define an ERASE variable to delete files when the make clean and make cleanall commands are called. On UNIX®, this is set to rm -f; on Windows it is set to del.

Related concepts:

- "Build files" on page 97
- "Error-checking utilities" on page 102
- "Sample files" on page 59

Related reference:

- "AIX supported development software" on page 8
- "HP-UX supported development software" on page 10
- "Linux supported development software" on page 12
- "Solaris supported development software" on page 17
- "Windows supported development software" on page 19

Error-checking utilities

The DB2® AD Client provides several utility files. These files have functions for error-checking and printing out error information. Utility files are provided for each language in the samples directory. When used with an application program, the error-checking utility files provide helpful error information, and make debugging a DB2 program much easier. Most of the error-checking utilities use the DB2 APIs GET SQLSTATE MESSAGE (sqlogstt) and GETERROR MESSAGE (sqlaintp) to obtain pertinent SQLSTATE and SQLCA information related to problems encountered in program execution. The DB2 CLI utility file, utilcli.c, does not use these DB2 APIs; instead it uses equivalent DB2 CLI statements. With all the error-checking utilities, descriptive error messages are printed out to allow the developer to quickly understand the problem.

Some DB2 programs, such as routines (stored procedures and user-defined functions), do not need to use the utilities. They are also not necessary for Java™ because the SQLException object will be thrown if an exception occurs.

Here are the error-checking utility files used by DB2-supported compilers for the different programming languages:

Table 41. Error-checking utility files by language

Language	Non-embedded SQL source file	Non-embedded SQL header file	Embedded SQL source file	Embedded SQL header file
C samples/c	utilapi.c	utilapi.h	utilemb.sqc	utilemb.h
C++ samples/cpp	utilapi.C	utilapi.h	utilemb.sqC	utilemb.h
CLI samples/cli	utilcli.c	utilcli.h	n/a	n/a
IBM® COBOL samples/cobol	checkerr.cbl	n/a	n/a	n/a
Micro Focus COBOL samples/cobol_mf	checkerr.cbl	n/a	n/a	n/a

In order to use the utility functions, the utility file must first be compiled, and then its object file linked in during the creation of the target program's executable. Both the `makefile` and build files in the `samples` directories do this for the programs that require the error-checking utilities.

The following example demonstrates how the error-checking utilities are used in DB2 programs. The `utilemb.h` header file defines the `EMB_SQL_CHECK` macro for the functions `SqlInfoPrint()` and `TransRollback()`:

```
/* macro for embedded SQL checking */
#define EMB_SQL_CHECK(MSG_STR)                          \
SqlInfoPrint(MSG_STR, &sqlca, __LINE__, __FILE__);      \
if (sqlca.sqlcode < 0)                                  \
{                                                       \
  TransRollback();                                      \
  return 1;                                             \
}
```

`SqlInfoPrint()` checks the SQLCODE flag. It prints out any available information related to the specific error indicated by this flag. It also points to where the error occurred in the source code. `TransRollback()` allows the utility file to safely rollback a transaction where an error has occurred. It requires embedded SQL statements to connect to the database and execute a rollback. The following is an example of how the C program `dbuse` calls the utility functions by using the macro, supplying the value `"Delete with host variables -- Execute"` for the `MSG_STR` parameter of the `SqlInfoPrint()` function:

```
EXEC SQL DELETE FROM org
  WHERE deptnumb = :hostVar1 AND
        division = :hostVar2;
EMB_SQL_CHECK("Delete with host variables -- Execute");
```

The `EMB_SQL_CHECK` macro ensures that if the `DELETE` statement fails, the transaction will be safely rolled back, and an appropriate error message printed out.

Developers are encouraged to use and expand upon these error-checking utilities when creating their own DB2 programs.

Related concepts:

- "Build files" on page 97

- "Makefiles" on page 99
- "Sample files" on page 59

Part 2. Building and Running Platform-Independent Applications

Chapter 4. Java

This chapter provides detailed information for building Java applets and applications. For the latest DB2 Java application development updates, visit the Web page at:

| `http://www.ibm.com/software/data/db2/udb/ad/v8/java`

Java sample programs

DB2® provides sample programs to demonstrate building and running JDBC programs that exclusively use dynamic SQL, and SQLJ programs that use static SQL. There are separate directories for JDBC and SQLJ samples under the `java` samples directory. Here is the Java™ samples directory structure on UNIX® and Windows® operating systems:

- On UNIX:

  `sqllib/samples/java`
  Contains a README file for Java sample programs in all sub-directories.

  `sqllib/samples/java/jdbc`
  Contains JDBC sample program files.

  `sqllib/samples/java/sqlj`
  Contains SQLJ sample programs.

  `sqllib/samples/java/Websphere`
  Contains Websphere sample programs.

  `sqllib/samples/java/plugin`
  Contains plugin example files for the DB2 Control Center.

  `sqllib/samples/java/plugin/doc`
  Contains javadoc files for the plugin interfaces.

- On Windows:

  `sqllib\samples\java`
  Contains a README file for Java sample programs in all sub-directories.

  `sqllib\samples\java\jdbc`
  Contains JDBC sample programs.

  `sqllib\samples\java\sqlj`
  Contains SQLJ sample programs.

sqllib\samples\java\Websphere
Contains Websphere sample programs.

sqllib\samples\java\plugin
Contains plugin example files for the DB2 Control Center.

sqllib\samples\java\plugin\doc
Contains javadoc files for the plugin interfaces.

The SQLJ samples directory contains build files (scripts on UNIX, batch files on Windows) to build the embedded SQL for Java programs. The JDBC directory does not contain build files because building JDBC programs on the command line using `javac` is so simple that build files are not needed.

Both the JDBC and SQLJ samples directories also contain optional makefiles. Makefiles are not widely used with Java, and the Java Development Kits (JDKs) do not ship with make executable files. DB2 supplies Java sample makefiles in case the user wants the convenience they provide. Each Java `makefile` builds all the supplied sample programs in either the JDBC or SQLJ samples directory. You can use a make program, such as gnumake, that's used with another language compiler.

There are two SQLJ build files provided: `bldsqlj` on UNIX, or `bldsqlj.bat` on Windows, which builds SQLJ applets and applications, and `bldsqljs` on UNIX, or `bldsqljs.bat` on Windows, which builds SQLJ routines (stored procedures and user-defined functions).

Related tasks:
- "Setting up the Java environment" on page 26
- "Building JDBC applets" on page 110
- "Building JDBC applications" on page 111
- "Building JDBC routines" on page 112
- "Building SQLJ applets" on page 115
- "Building SQLJ applications" on page 117
- "Building SQLJ routines" on page 121
- "Building SQLJ programs" on page 114

Related reference:
- "JDBC samples" on page 78
- "SQLJ samples" on page 80
- "Java WebSphere samples" on page 82
- "Java plug-in samples" on page 83

Java applet considerations

DB2® databases can be accessed by using Java™ applets. Please keep the following points in mind when using them:

1. If you are using the now deprecated type 3 driver (also known as the "net" driver), it is essential that the `db2java.zip` file used by the Java applet be at the same FixPak level as the JDBC applet server. Under normal circumstances, `db2java.zip` is loaded from the Web Server where the JDBC applet server is running. This ensures a match. If, however, your configuration has the Java applet loading `db2java.zip` from a different location, a mismatch can occur. Matching FixPak levels between the two files is strictly enforced at connection

time. If a mismatch is detected, the connection is rejected, and the client receives one of the following exceptions:

- If db2java.zip is at DB2 Version 7 FixPak 2 or later:

```
COM.ibm.db2.jdbc.DB2Exception: [IBM][JDBC Driver]
CLI0621E  Unsupported JDBC server configuration.
```

- If db2java.zip is prior to DB2 Version 7 FixPak 2:

```
COM.ibm.db2.jdbc.DB2Exception: [IBM][JDBC Driver]
CLI0601E  Invalid statement handle or statement is closed.
SQLSTATE=S1000
```

If a mismatch occurs, the JDBC applet server logs one of the following messages in the `jdbcerr.log` file:

- If the JDBC applet server is at DB2 Version 7 FixPak 2 or later:

```
jdbcFSQLConnect: JDBC Applet Server and client (db2java.zip)
versions do not match. Unable to proceed with connection., einfo= -111
```

- If the JDBC applet server is prior to DB2 Version 7 FixPak 2:

```
jdbcServiceConnection(): Invalid Request Received., einfo= 0
```

Note: The `db2JDBCVersion.java` sample file in `sqllib\samples\java` (Windows), or in `sqllib/samples/java` (UNIX) should not be used with DB2 Version 8. This program can be used with DB2 Version 7 to check which version of the DB2 JDBC driver is currently in use, and whether the JDBC environment is correctly set up for it.

Users are strongly recommended to migrate their applets to the DB2 Universal JDBC driver.

2. For a larger JDBC or SQLJ applet that consists of several Java classes, you might choose to package all its classes in a single JAR file. For an SQLJ applet, you would also have to package its serialized profiles along with its classes. If you choose to do this, add your JAR file into the `archive` parameter in the "applet" tag. For details, see the Java Developer Kit Version 1.3 documentation.

   For SQLJ applets, some browsers do not yet have support for loading a serialized object from a resource file associated with the applet. For example, you will get the following error message when trying to load the supplied sample applet `Applt` in those browsers:

```
java.lang.ClassNotFoundException: Applt_SJProfile0
```

   As a workaround, there is a utility which converts a serialized profile into a profile stored in Java class format. The utility is a Java class called `sqlj.runtime.profile.util.SerProfileToClass`. It takes a serialized profile resource file as input and produces a Java class containing the profile as output. Your profile can be converted using one of the following commands:

```
profconv Applt_SJProfile0.ser
```

   or

```
java sqlj.runtime.profile.util.SerProfileToClass Applt_SJProfile0.ser
```

   The class `Applt_SJProfile0.class` is created as a result. Replace all profiles in `.ser` format used by the applet with profiles in `.class` format, and the problem should go away.

3. You can place the file `db2java.zip` or `db2jcc.jar`, or both, into a directory that is shared by several applets that might be loaded from your Web site. `db2java.zip` is for applets using the JDBC type 3 driver; `db2jcc.jar` is for applets using the DB2 Universal JDBC driver or for any SQLJ applet. These files are in the `sqllib\java` directory on Windows® operating systems, and in the `sqllib/java` directory on UNIX®. You might need to add a `codebase` parameter into the "applet" tag in the HTML file to identify the directory. For details, see the Java Developer Kit Version 1.3 documentation.
4. Since DB2 Version 5.2, signal handling has been added to the JDBC applet server (listener), `db2jd`, to make it more robust. As a result, one cannot use the CTRL-C command to kill `db2jd`. Therefore, the only way to terminate the listener is to kill the process by using `kill -9` (for Unix) or the Task Manager (for Windows).
5. For information on running DB2 Java applets on a Web server, specifically the Domino™ Go Webserver, visit:

   ```
   http://www.ibm.com/software/data/db2/db2lotus/gojava.htm
   ```

Related tasks:

- "Setting up the Java environment" on page 26
- "Building JDBC applets" on page 110
- "Building SQLJ applets" on page 115

JDBC

Building JDBC applets

`Applt` demonstrates a dynamic SQL Java applet to access a DB2 database.

Procedure:

You can use the, now deprecated, type 3 driver (also known as the "net" driver), or the universal JDBC driver, which is installed with DB2 Java Enablement. Sections for connecting with both these drivers are presented below. It is strongly recommended that you migrate your applets to the universal JDBC driver.

To build and run the JDBC applet, `Applt`, by commands entered at the command line, either ensure that a web server is installed and running on your DB2 machine (server or client), or use the applet viewer that comes with the Java Development Kit by entering the following command in the working directory of your client machine:

```
appletviewer Applt.html
```

Connecting with the Type 3 ("net") Driver

To connect with the type 3 driver, first modify the `Applt.html` file according to the instructions in the file. Then, start the JDBC applet server on the TCP/IP port specified in `Applt.html`. For example, if in `Applt.html`, you specified `param name=port value='6789'`, then you would enter:

```
db2jstrt 6789
```

Make sure the JDBC port number in the connection string is the recommended default, "6789". Only change this if you are sure the number does not conflict with another port number. Do not use the database port number, "50000".

Connecting with the universal JDBC driver

To connect with the universal JDBC driver, modify the `Applt.html` file according to the instructions in the file. For the TCP/IP port number, you can use the database port number, "50000".

Building the applet

1. Compile `Applt.java` to produce the file `Applt.class` with this command:
   ```
   javac Applt.java
   ```
2. Ensure that your working directory is accessible by your web browser. If it is not, copy `Applt.class` and `Applt.html` into a directory that is accessible.
3. If using a type 3 driver, copy `sqllib\java\db2java.zip` on Windows, or `sqllib/java/db2java.zip` on UNIX, into the same directory as `Applt.class` and `Applt.html`.

   If using the universal JDBC driver, copy `sqllib\java\db2jcc.jar` on Windows or `sqllib/java/db2jcc.jar` on UNIX, into the same directory as `Applt.class` and `Applt.html`.
4. On your client machine, start your web browser (which must support Java 1.3) and load `Applt.html`.

You can also use the Java `makefile` to build this program.

Related concepts:

- "Java applet considerations" on page 108

Related tasks:

- "Building JDBC applications" on page 111
- "Building JDBC routines" on page 112
- "Building SQLJ applets" on page 115

Related reference:

- "JDBC samples" on page 78

Related samples:

- "Applt.java -- A Java applet that use JDBC applet driver to access a database (JDBC)"

Building JDBC applications

`DbInfo` demonstrates a dynamic SQL Java application accessing a DB2 database.

Procedure:

To build and run this application by commands entered at the command line:

1. Compile `DbInfo.java` to produce the file `DbInfo.class` with this command:
   ```
   javac DbInfo.java
   ```
2. Run the java interpreter on the application with this command:
   ```
   java DbInfo
   ```

You can also use the Java `makefile` to build this program.

Note: If you are running a Java application on Unix in a 64-bit DB2 instance but the Java Developer Kit is 32-bit, you must change the DB2 library path before running the application. For example on AIX:

- If using bash or Korn shell:

```
export LIBPATH=$HOME/sqllib/lib32
```

- If using C shell:

```
setenv LIBPATH $HOME/sqllib/lib32
```

Related tasks:

- "Building JDBC applets" on page 110
- "Building JDBC routines" on page 112
- "Building SQLJ applications" on page 117

Related reference:

- "JDBC samples" on page 78

Related samples:

- "DbInfo.java -- How to get/set info in a database (JDBC)"

Building JDBC routines

DB2 provides sample programs demonstrating JDBC routines (stored procedures and user-defined functions) in the `samples/java/jdbc` directory on UNIX, and the `samples\java\jdbc` directory on Windows. Routines are compiled and stored on a server. When called by a client application, they access the server database and return information to the client application.

Procedure:

The following examples show you how to build routines comprising:

- stored procedures
- user-defined functions without SQL statements
- user-defined functions with SQL statements

Stored Procedures

`SpServer` demonstrates dynamic SQL PARAMETER STYLE JAVA stored procedures.

To build and run this program on the server from the command line:

1. Compile `SpServer.java` to produce the file `SpServer.class` with this command:

   ```
   javac SpServer.java
   ```

2. Copy `SpServer.class` to the `sqllib\function` directory on Windows operating systems, or to the `sqllib/function` directory on UNIX.
3. Next, catalog the routines by running the `spcat` script on the server. Enter:

   ```
   spcat
   ```

   This script connects to the sample database, uncatalogs the routines if they were previously cataloged by calling `SpDrop.db2`, then catalogs them by calling `SpCreate.db2`, and finally disconnects from the database. You can also run the `SpDrop.db2` and `SpCreate.db2` scripts individually.

4. Then, stop and restart the database to allow the new class file to be recognized. If necessary, set the file mode for the class file to ″read″ so it is readable by the fenced user.
5. Compile and run the `SpClient` client application to access the stored procedure class.

User-defined functions without SQL statements

`UDFsrv` is a user-defined function library that does not contain SQL statements. DB2 provides both a JDBC client application, `UDFcli`, and an SQLJ client application, `UDFcli`, that can access the `UDFsrv` library.

To build and run the UDF program on the server from the command line:

1. Compile `UDFsrv.java` to produce the file `UDFsrv.class` with this command:
   ```
   javac UDFsrv.java
   ```
2. Copy `UDFsrv.class` to the `sqllib\function` directory on Windows operating systems, or to the `sqllib/function` directory on UNIX.
3. To access the `UDFsrv` library, you can use either JDBC or SQLJ client applications. Both versions of the client program contain the `CREATE FUNCTION` SQL statement that you use to register the UDFs contained in `UDFsrv` with the database, and also contain SQL statements that make use of the UDFs, once they have been registered.

User-defined functions with SQL statements

`UDFsqlsv` is a user-defined function library that contains SQL statements. DB2 provides a JDBC client application, `UDFsqlcl`, to access the `UDFsqlsv` library.

To build and run the UDF program on the server from the command line:

1. Compile `UDFsqlsv.java` to produce the file `UDFsqlsv.class` with this command:
   ```
   javac UDFsqlsv.java
   ```
2. Copy `UDFsqlsv.class` to the `sqllib\function` directory on Windows operating systems, or to the `sqllib/function` directory on UNIX.
3. To access the `UDFsqlsv` library, use the client program, `UDFsqlcl`, which contains the `CREATE FUNCTION` SQL statement that you use to register the UDFs contained in `UDFsqlsv` with the database. The client program also contains SQL statements that make use of the UDFs, once they have been registered.

You can also use the Java makefile to build the above programs.

Related tasks:

- “Building JDBC applets” on page 110
- “Building JDBC applications” on page 111
- “Building SQLJ routines” on page 121

Related reference:

- “JDBC samples” on page 78

Related samples:

- “spcat -- To catalog SQLj stored procedures on UNIX”
- “SpClient.java -- Call a variety of types of stored procedures from SpServer.java (JDBC)”

- "SpCreate.db2 -- How to catalog the stored procedures contained in SpServer.java "
- "SpDrop.db2 -- How to uncatalog the stored procedures contained in SpServer.java"
- "SpServer.java -- Provide a variety of types of stored procedures to be called from (JDBC)"
- "UDFcli.java -- Call the UDFs in UDFsrv.java (JDBC)"
- "UDFCreate.db2 -- How to catalog the Java UDFs contained in UDFsrv.java "
- "UDFDrop.db2 -- How to uncatalog the Java UDFs contained in UDFsrv.java "
- "UDFsCreate.db2 -- How to catalog the UDFs contained in UDFsqlsv.java "
- "UDFsDrop.db2 -- How to uncatalog the UDFs contained in UDFsqlsv.java "
- "UDFsqlcl.java -- Call the UDFs in UDFsqlsv.java (JDBC)"
- "UDFsqlsv.java -- Provide UDFs to be called by UDFsqlcl.java (JDBC)"
- "UDFsrv.java -- Provide UDFs to be called by UDFcli.java (JDBC)"

SQLJ

Building SQLJ programs

DB2 supplies build files to build the SQLJ sample programs. For applets and applications, you can use the `bldsqlj` script on UNIX or the `bldsqlj.bat` batch file on Windows. For routines (stored procedures and user-defined functions), you can use the `bldsqljs` script on UNIX, or the `bldsqljs.bat` batch file on Windows.

The SQLJ translator shipped with DB2 calls the Java compiler to compile the translated `.java` files into `.class` files. Therefore, the build files use the **sqlj** command to do both.

Notes:

1. In previous versions of DB2, the `db2profc` command used a URL of the form `-url=jdbc:db2:dbname` where `dbname` was the locally cataloged database alias. The new `db2sqljcustomize` command follows the conventions for the DB2 universal JDBC driver: `-url jdbc:db2://hostname:portnumber/dbname` where `hostname` is the name of the DB2 server, `portnumber` is the TCP/IP listener port number of the DB2 server and `dbname` is the database alias cataloged on the DB2 Server. This means the DB2 Server must be configured for TCP/IP connections.
2. SQLJ programs translated with previous versions of the sqlj command must be retranslated with the DB2 Version 8 sqlj command, and customized with the `db2sqljcustomize` command.
3. The DB2 SQLJ profile printer, `db2sqljprint`, prints the contents of a DB2 profile in plain text.

Procedure:

To build the different types of DB2 SQLJ programs, see the following:

- "Building SQLJ applets" on page 115
- "Building SQLJ applications" on page 117
- "Building SQLJ routines" on page 121

Related concepts:

- "Java sample programs" on page 107
- "Java applet considerations" on page 108

Related tasks:
- "Building SQLJ applets" on page 115
- "Building SQLJ applications" on page 117
- "Building SQLJ routines" on page 121

Related reference:
- "db2sqljcustomize - DB2 SQLJ Profile Customizer Command" in the *Command Reference*
- "db2sqljprint - DB2 SQLJ Profile Printer Command" in the *Command Reference*
- "SQLJ samples" on page 80
- "sqlj - DB2 SQLJ Translator Command" in the *Command Reference*

Building SQLJ applets

The following steps show how to build the `Applt` sample that demonstrates an SQLJ applet accessing a DB2 database. These steps use the build file, `bldsqlj` (UNIX), or `bldsqlj.bat` (Windows), which contains commands to build either an SQLJ applet or application.

The build file takes up to six parameters: $1, $2, $3, $4, $5, and $6 on UNIX, and %1, %2, %3, %4, %5, and %6 on Windows. The first parameter specifies the name of your program. The second parameter specifies the user ID for the database instance, the third parameter specifies the password. The fourth parameter specifies the server name. The fifth parameter specifies the port number. And the sixth parameter specifies the database name. For all but the first parameter, program name, default values can be used. See the build file for details about using default parameter values.

Procedure:

You can use the, now deprecated, type 3 driver (also known as the "net" driver), or the universal JDBC driver, which is installed with DB2 Java Enablement. Sections for connecting with both these drivers are presented below. It is strongly recommended that you migrate your applets to the universal JDBC driver.

To run this applet, either ensure that a web server is installed and running on your DB2 machine (server or client), or you can use the applet viewer that comes with the Java Development Kit by entering the following command in the working directory of your client machine:

```
appletviewer Applt.html
```

Connecting with the Type 3 ("net") Driver

To connect with the type 3 driver, first modify the `Applt.html` file according to the instructions in the file. Then, start the JDBC applet server on the TCP/IP port specified in `Applt.html`. For example, if in `Applt.html`, you specified `param name=port value='6789'`, then you would enter:

```
db2jstrt 6789
```

Make sure the JDBC port number in the connection string is the recommended default, "6789". Only change this if you are sure the number does not conflict with another port number. Do not use the database port number, "50000".

Connecting with the universal JDBC driver

To connect with the universal JDBC driver, modify the `Applt.html` file according to the instructions in the file. For the TCP/IP port number, you should use the database port number, "50000".

Building the Applet

1. Build the applet with this command:

   ```
   bldsqlj Applt <userid> <password> <server_name> <port_number> <db_name>
   ```

   where all parameters except the program name can have default values, as explained in the build file.
2. Ensure that your working directory is accessible by your web browser, or by appletviewer if you are using it. If your directory is not accessible, copy the following files into a directory that is accessible:

   ```
   Applt.html                      Applt.class
   Applt_Cursor1.class             Applt_Cursor2.class
   Applt_SJProfileKeys.class       Applt_SJProfile0.ser
   ```
3. If using a type 3 driver, copy `sqllib\java\db2jcc.jar` and `sqllib\java\db2java.zip` on Windows, or `sqllib/java/db2jcc.jar` and `sqllib/java/db2java.zip` on UNIX, into the same directory as `Applt.class` and `Applt.html`.

   If using the universal JDBC driver, copy `sqllib\java\db2jcc.jar` on Windows or `sqllib/java/db2jcc.jar` on UNIX, into the same directory as `Applt.class` and `Applt.html`.
4. On your client machine, start your web browser (which must support Java Developer Kit 1.3), or appletviewer, and load `Applt.html`.

You can also use the Java `makefile` to build this program.

Related concepts:

- "Java applet considerations" on page 108

Related tasks:

- "Building JDBC applets" on page 110
- "Building SQLJ applications" on page 117
- "Building SQLJ routines" on page 121

Related reference:

- "SQLJ application and applet options for UNIX" on page 119
- "SQLJ application and applet options for Windows" on page 121
- "SQLJ samples" on page 80

Related samples:

- "Applt.sqlj -- An SQLJ applet that uses a JDBC applet driver to access a database (SQLj)"
- "bldsqlj.bat -- Builds a Java embedded SQL (SQLJ) application or applet on Windows"

- "bldsqlj -- Builds Java embedded SQL (SQLJ) applications and applets on UNIX"

Building SQLJ applications

The following steps show how to build the `TbMod` sample that demonstrates an SQLJ application accessing a DB2 database. These steps use the build file, `bldsqlj` (UNIX), or `bldsqlj.bat` (Windows), which contains commands to build either an SQLJ applet or application.

The build file takes up to six parameters: $1, $2, $3, $4, $5, and $6 on UNIX, and %1, %2, %3, %4, %5, and %6 on Windows. The first parameter specifies the name of your program. The second parameter specifies the user ID for the database instance, the third parameter specifies the password. The fourth parameter specifies the server name. The fifth parameter specifies the port number. And the sixth parameter specifies the database name. For all but the first parameter, program name, default values can be used. See the build file for details about using default parameter values.

Procedure:

To build `TbMod` with the build file, `bldsqlj` (UNIX) or `bldsqlj.bat` (Windows), enter this command:

```
bldsqlj TbMod <userid> <password> <server_name> <port_number> <db_name>
```

where all parameters except the program name can have default values, as explained in the build file.

Run the Java interpreter on the application with this command:

```
java TbMod
```

You can also use the Java `makefile` to build this program.

Note: If you are running a Java application on Unix in a 64-bit DB2 instance but the Java Developer Kit is 32-bit, you must change the DB2 library path before running the application. For example on AIX:

- If using bash or Korn shell:

  ```
  export LIBPATH=$HOME/sqllib/lib32
  ```

- If using C shell:

  ```
  setenv LIBPATH $HOME/sqllib/lib32
  ```

Related tasks:

- "Building JDBC applications" on page 111
- "Building SQLJ applets" on page 115
- "Building SQLJ routines" on page 121

Related reference:

- "SQLJ application and applet options for UNIX" on page 119
- "SQLJ application and applet options for Windows" on page 121
- "SQLJ samples" on page 80

Related samples:

- "bldsqlj.bat -- Builds a Java embedded SQL (SQLJ) application or applet on Windows"

- "bldsqlj -- Builds Java embedded SQL (SQLJ) applications and applets on UNIX"
- "TbMod.sqlj -- How to modify table data (SQLj)"

UNIX build script for SQLJ applications and applets

```
#! /bin/sh
# SCRIPT: bldsqlj
# Builds Java embedded SQL (SQLJ) applications and applets on UNIX
# Usage: bldsqlj prog_name  (requires hardcoding user ID and password)
#        bldsqlj prog_name userid password
#        bldsqlj prog_name userid password server_name
#        bldsqlj prog_name userid password server_name port_number
#        bldsqlj prog_name userid password server_name port_number db_name
#
#        Defaults:
#          userid      = $USER variable requires updating if used
#          password    = $PSWD variable requires updating if used
#          server_name = $SERVER variable set to local hostname
#          port_number = $PORTNUM variable set to 50000
#          db_name     = $DB variable set to "sample"

# To hardcode user ID (USER) and password (PSWD)
# Replace "NULL" with the correct values in quotes
USER="NULL"
PSWD="NULL"
# You can replace the defaults for each of the following
# with a new value. Note that the PORTNUM number cannot
# be one already used by another process.
SERVER=`hostname`
PORTNUM=50000
DB="sample"

# Translate and compile the SQLJ source file
# and bind the package to the database.
 if (( [ $# -eq 1 ]  && [ $USER != "NULL" ] && [ $PSWD != "NULL" ] ) || \
    ( [ $# -ge 3 ]  && [ $# -le 6 ] ) )
 then
    # Remove .sqlj extension
    progname=${1%.sqlj}

    sqlj "${progname}.sqlj"

    if [ $# -eq 1 ]
    then
       db2sqljcustomize -url jdbc:db2://$SERVER:$PORTNUM/$DB \
       -user $USER -password $PSWD "${progname}_SJProfile0"
    elif [ $# -eq  3 ]
    then
       db2sqljcustomize -url jdbc:db2://$SERVER:$PORTNUM/$DB -user $2 -password $3 \
       "${progname}_SJProfile0"
    elif [ $# -eq 4 ]
    then
       db2sqljcustomize -url jdbc:db2://$4:$PORTNUM/$DB -user $2 -password $3 \
       "${progname}_SJProfile0"
    elif [ $# -eq 5 ]
    then
       db2sqljcustomize -url jdbc:db2://$4:$5/$DB -user $2 -password $3 \
       "${progname}_SJProfile0"
    else
       db2sqljcustomize -url jdbc:db2://$4:$5/$6 -user $2 -password $3 \
       "${progname}_SJProfile0"
    fi
 else
    echo 'Usage: bldsqlj prog_name (requires hardcoding user ID and password)'
    echo '       bldsqlj prog_name userid password'
    echo '       bldsqlj prog_name userid password server_name'
```

```
    echo '       bldsqlj prog_name userid password server_name port_number'
    echo '       bldsqlj prog_name userid password server_name port_number db_name'
    echo ''
    echo '       Defaults:'
    echo '         userid      = '$USER
    echo '         password    = '$PSWD
    echo '         server_name = '$SERVER
    echo '         port_number = '$PORTNUM
    echo '         db_name     = '$DB
fi
```

SQLJ application and applet options for UNIX

The following table contains the SQLJ translator and customizer options used in the `bldsqlj` build script on UNIX. These are the options DB2 recommends that you use to build SQLJ applications and applets on UNIX platforms.

Translator and customizer options for **bldsqlj**
sqlj The SQLJ translator (also compiles the program).
"${progname}.sqlj" The SQLJ source file. The `progname=${1%.sqlj}` command removes the extension if it was included in the input file name, so when the extension is added back again, it is not duplicated.
db2sqljcustomize The DB2 for Java profile customizer.
-url Specifies a JDBC URL for establishing a database connection, such as `jdbc:db2://servername:50000/sample`.
-user Specifies a user ID.
-password Specifies a password.
"${progname}_SJProfile0" Specifies a serialized profile for the program.

Related tasks:

- "Building SQLJ applets" on page 115
- "Building SQLJ applications" on page 117

Related reference:

- "SQLJ routine options for UNIX" on page 124

Related samples:

- "bldsqlj -- Builds Java embedded SQL (SQLJ) applications and applets on UNIX"

Windows batch file for SQLJ applications and applets

```
@echo off
rem BATCH FILE: bldsqlj.bat
rem Builds a Java embedded SQL (SQLJ) application or applet on Windows

rem To add defaults for user ID (USER) and password (PSWD)
rem Uncomment the following and add the appropriate values
rem set USR=
rem set PSWD=

rem You can replace the defaults for each of the following
```

```
rem with a new value. Note that the PORTNUM number cannot be
rem one already used by another process.
set SERVER=%COMPUTERNAME%
set PORTNUM=50000
set DB=sample

goto start
:usage
echo Usage: bldsqlj prog_name (requires hardcoding user ID and password)
echo        bldsqlj prog_name userid password
echo        bldsqlj prog_name userid password server_name
echo        bldsqlj prog_name userid password server_name port_number
echo        bldsqlj prog_name userid password server_name port_number db_name
echo.
echo        Defaults:
echo          userid      = %USR%
echo          password    = %PSWD%
echo          server_name = %SERVER%
echo          port_number = %PORTNUM%
echo          db_name     = %DB%
goto exit

:start
rem Translate and compile the SQLJ source file
rem and bind the package to the database.
if "%1" == "" goto usage
if "%2" == "" goto case1
if "%3" == "" goto usage
if "%4" == "" goto case3
if "%5" == "" goto case4
if "%6" == "" goto case5
if "%7" == "" goto case6
goto usage

:case1
   if "%USR%" == "" goto usage
   if "%PSWD%" == "" goto usage
   if "%SERVER%" == "" goto nohostname
   sqlj %1.sqlj
   db2sqljcustomize -url jdbc:db2://%SERVER%:%PORTNUM%/%DB% -user %USR%
      -password %PSWD% %1_SJProfile0
   goto continue

:case3
   if "%SERVER%" == "" goto nohostname
   sqlj %1.sqlj
   db2sqljcustomize -url jdbc:db2://%SERVER%:%PORTNUM%/%DB% -user %2
      -password %3 %1_SJProfile0
   goto continue

:case4
   sqlj %1.sqlj
   db2sqljcustomize -url jdbc:db2://%4:%PORTNUM%/%DB% -user %2
      -password %3 %1_SJProfile0
   goto continue

:case5
   sqlj %1.sqlj
   db2sqljcustomize -url jdbc:db2://%4:%5/%DB% -user %2
      -password %3 %1_SJProfile0
   goto continue

:case6
   sqlj %1.sqlj
   db2sqljcustomize -url jdbc:db2://%4:%5/%6 -user %2
      -password %3 %1_SJProfile0
   goto continue
```

```
:continue
goto exit

:nohostname
echo Server name (hostname) could not be determined.
echo.
goto usage

:exit
@echo on
```

SQLJ application and applet options for Windows

The following table contains the SQLJ translator and customizer options used in the `bldsqlj.bat` batch file on Windows operating systems. These are the options DB2 recommends that you use to build SQLJ applications and applets on Windows.

Translator and customizer options for `bldsqlj.bat`	
sqlj	The SQLJ translator (also compiles the program).
%1.sqlj	The SQLJ source file.
db2sqljcustomize	The DB2 for Java profile customizer.
-url	Specifies a JDBC URL for establishing a database connection, such as `jdbc:db2://servername:50000/sample`.
-user	Specifies a user ID.
-password	Specifies a password.
%1_SJProfile0	Specifies a serialized profile for the program.

Related tasks:

- "Building SQLJ applets" on page 115
- "Building SQLJ applications" on page 117

Related reference:

- "SQLJ routine options for Windows" on page 126

Related samples:

- "bldsqlj.bat -- Builds a Java embedded SQL (SQLJ) application or applet on Windows"

Building SQLJ routines

DB2 provides sample programs demonstrating SQLJ routines (stored procedures and user-defined functions) in the `samples/java/sqlj` directory on UNIX, and the `samples\java\sqlj` directory on Windows. Routines are compiled and stored on a server. When called by a client application, they access the server database and return information to the client application.

In the same directory, DB2 also supplies the build file, `bldsqljs` (UNIX), or `bldsqljs.bat` (Windows), which contains commands to build routines.

The build file takes up to six parameters: $1, $2, $3, $4, $5, and $6 on UNIX, and %1, %2, %3, %4, %5, and %6 on Windows. The first parameter specifies the name of your program. The second parameter specifies the user ID for the database instance, the third parameter specifies the password. The fourth parameter specifies the server name. The fifth parameter specifies the port number. And the sixth parameter specifies the database name. For all but the first parameter, program name, default values can be used. See the build file for details about using default parameter values.

Procedure:

The following example shows you how to build a class file with stored procedures.

`SpServer` demonstrates PARAMETER STYLE JAVA stored procedures using the JDBC application driver to access a DB2 database.

To build this stored procedure class with the build file, `bldsqljs` (UNIX) or `bldsqljs.bat` (Windows):

1. Enter the following command:

   ```
   bldsqljs SpServer <userid> <password> <server_name> \
     <port_number> <db_name>
   ```

   where all parameters except the program name can have default values, as explained in the build file.
2. Next, catalog the routines by running the `spcat` script on the server. Enter:

   ```
   spcat
   ```

   This script connects to the sample database, uncatalogs the routines if they were previously cataloged by calling `SpDrop.db2`, then catalogs them by calling `SpCreate.db2`, and finally disconnects from the database. You can also run the `SpDrop.db2` and `SpCreate.db2` scripts individually.
3. Then, stop and restart the database to allow the new class file to be recognized. If necessary, set the file mode for the class file to ″read″ so it is readable by the fenced user.
4. Build and run the `SpClient` client application to call the stored procedures. You can build `SpClient` with the application build file, `bldsqlj` (UNIX) or `bldsqlj.bat` (Windows).

You can also use the Java `makefile` to build the above programs.

Related tasks:

- “Building JDBC routines” on page 112
- “Building SQLJ applets” on page 115
- “Building SQLJ applications” on page 117

Related reference:

- “SQLJ routine options for UNIX” on page 124
- “SQLJ routine options for Windows” on page 126
- “SQLJ samples” on page 80

Related samples:

- "bldsqljs.bat -- Builds a Java embedded SQL (SQLJ) stored procedure on Windows"
- "bldsqljs -- Builds Java embedded SQL (SQLJ) stored procedures on UNIX"
- "spcat -- To catalog SQLj stored procedures on UNIX"
- "SpClient.sqlj -- Call a variety of types of stored procedures from SpServer.sqlj (SQLj)"
- "SpCreate.db2 -- How to catalog the stored procedures contained in SpServer.sqlj "
- "SpDrop.db2 -- How to uncatalog the stored procedures contained in SpServer.sqlj"
- "SpIterat.sqlj -- Iterator class file for SpServer.sqlj (SQLj)"
- "SpServer.sqlj -- Provide a variety of types of stored procedures to be called from (SQLj)"

UNIX build script for SQLJ routines

```
#! /bin/sh
# SCRIPT: bldsqljs
# Builds Java embedded SQL (SQLJ) stored procedures on UNIX
# Usage: bldsqljs prog_name  (requires hardcoding user ID and password)
#        bldsqljs prog_name userid password
#        bldsqljs prog_name userid password server_name
#        bldsqljs prog_name userid password server_name port_number
#        bldsqljs prog_name userid password server_name port_number db_name
#
#        Defaults:
#          userid      = $USER variable requires updating if used
#          password    = $PSWD variable requires updating if used
#          server_name = $SERVER variable set to local hostname
#          port_number = $PORTNUM variable set to 50000
#          db_name     = $DB variable set to "sample"

# To hardcode user ID (USER) and password (PSWD)
# Replace "NULL" with the correct values in quotes
USER="NULL"
PSWD="NULL"
# You can replace the defaults for each of the following
# with a new value. Note that the PORTNUM number cannot
# be one already used by another process.
SERVER=`hostname`
PORTNUM=50000
DB="sample"

# Translate and compile the SQLJ source file
# and bind the package to the database.
 if (( [ $# -eq 1 ]  && [ $USER != "NULL" ] && [ $PSWD != "NULL" ] ) \
    || ( [ $# -ge 3 ] && [ $# -le 6 ] ))
 then
    # Remove .sqlj extension
    progname=${1%.sqlj}

    sqlj "${progname}.sqlj"

    if [ $# -eq 1 ]
    then
       db2sqljcustomize -url jdbc:db2://$SERVER:$PORTNUM/$DB \
       -user $USER -password $PSWD "${progname}_SJProfile0"
    elif [ $# -eq  3 ]
    then
       db2sqljcustomize -url jdbc:db2://$SERVER:$PORTNUM/$DB -user $2 \
       -password $3 "${progname}_SJProfile0"
```

```
   elif [ $# -eq 4 ]
   then
      db2sqljcustomize -url jdbc:db2://$4:$PORTNUM/$DB -user $2 -password $3 \
      "${progname}_SJProfile0"
   elif [ $# -eq 5 ]
   then
      db2sqljcustomize -url jdbc:db2://$4:$5/$DB -user $2 -password $3 \
      "${progname}_SJProfile0"
   else
      db2sqljcustomize -url jdbc:db2://$4:$5/$6 -user $2 -password $3 \
      "${progname}_SJProfile0"
   fi

   # Copy the *.class and *.ser files to the 'function' directory.
   rm -f "$DB2PATH/function/${progname}*.class"
   rm -f "$DB2PATH/function/${progname}*.ser"
   cp "${progname}*.class" "$DB2PATH/function"
   cp "${progname}*.ser" "$DB2PATH/function"

else
   echo 'Usage: bldsqljs prog_name (requires hardcoding user ID and password)'
   echo '       bldsqljs prog_name userid password'
   echo '       bldsqljs prog_name userid password server_name'
   echo '       bldsqljs prog_name userid password server_name port_number'
   echo '       bldsqljs prog_name userid password server_name port_number db_name'
   echo ''
   echo '       Defaults:'
   echo '         userid      = '$USER
   echo '         password    = '$PSWD
   echo '         server_name = '$SERVER
   echo '         port_number = '$PORTNUM
   echo '         db_name     = '$DB
fi
```

SQLJ routine options for UNIX

The following table contains the SQLJ translator and customizer options used in the `bldsqljs` build script on UNIX. These are the options DB2 recommends that you use to build SQLJ routines (stored procedures and user-defined functions) on UNIX platforms.

Translator and customizer options for **bldsqljs**
sqlj The SQLJ translator (also compiles the program).
"${progname}.sqlj" The SQLJ source file. The `progname=${1%.sqlj}` command removes the extension if it was included in the input file name, so when the extension is added back again, it is not duplicated.
db2sqljcustomize The DB2 for Java profile customizer.
-url Specifies a JDBC URL for establishing a database connection, such as `jdbc:db2://servername:50000/sample`.
-user Specifies a user ID.
-password Specifies a password.
"${progname}_SJProfile0" Specifies a serialized profile for the program.

Related tasks:

- "Building SQLJ routines" on page 121

Related reference:

- "SQLJ application and applet options for UNIX" on page 119

Related samples:

- "bldsqljs -- Builds Java embedded SQL (SQLJ) stored procedures on UNIX"

Windows batch file for SQLJ routines

```
@echo off
rem BATCH FILE: bldsqljs.bat
rem Builds a Java embedded SQL (SQLJ) stored procedure on Windows

rem To add defaults for user ID (USR) and password (PSWD)
rem Uncomment the following and add the appropriate values
rem set USR=
rem set PSWD=
rem You can replace the defaults for each of the following
rem with a new value. Note that the PORTNUM number cannot be
rem one already used by another process.
set SERVER=%COMPUTERNAME%
set PORTNUM=50000
set DB=sample

goto start
:usage
echo Usage: bldsqljs prog_name (requires hardcoding user ID and password)
echo        bldsqljs prog_name userid password
echo        bldsqljs prog_name userid password server_name
echo        bldsqljs prog_name userid password server_name port_number
echo        bldsqljs prog_name userid password server_name port_number db_name
echo.
echo        Defaults:
echo          userid      = %USR%
echo          password    = %PSWD%
echo          server_name = %SERVER%
echo          port_number = %PORTNUM%
echo          db_name     = %DB%
goto exit

:start
rem Translate and compile the SQLJ source file
rem and bind the package to the database.
if "%DB2PATH%" == "" goto nodb2cmd
if "%1" == "" goto usage
if "%2" == "" goto case1
if "%3" == "" goto usage
if "%4" == "" goto case3
if "%5" == "" goto case4
if "%6" == "" goto case5
if "%7" == "" goto case6
goto usage

:case1
   if "%USR%" == "" goto usage
   if "%PSWD%" == "" goto usage
   if "%SERVER%" == "" goto nohostname
   sqlj %1.sqlj
   db2sqljcustomize -url jdbc:db2://%SERVER%:%PORTNUM%/%DB% -user %USR%
      -password %PSWD% %1_SJProfile0
   goto continue

:case3
   if "%SERVER%" == "" goto nohostname
```

```
   sqlj %1.sqlj
   db2sqljcustomize -url jdbc:db2://%SERVER%:%PORTNUM%/%DB% -user %2
      -password %3 %1_SJProfile0
   goto continue

:case4
   sqlj %1.sqlj
   db2sqljcustomize -url jdbc:db2://%4:%PORTNUM%/%DB% -user %2
     -password %3 %1_SJProfile0
   goto continue

:case5
   sqlj %1.sqlj
   db2sqljcustomize -url jdbc:db2://%4:%5/%DB% -user %2
     -password %3 %1_SJProfile0
   goto continue

:case6
   sqlj %1.sqlj
   db2sqljcustomize -url jdbc:db2://%4:%5/%6 -user %2
     -password %3 %1_SJProfile0
   goto continue

:continue
rem Copy the *.class and *.ser files to the 'function' directory.
copy %1*.class %DB2PATH%\function\
copy %1*.ser %DB2PATH%\function\
goto exit

:nodb2cmd
echo DB2 command line environment not initialized. Please run db2cmd and try again.
goto exit

:nohostname
echo Server name (hostname) could not be determined.
echo.
goto usage

:exit
@echo on
```

SQLJ routine options for Windows

The following table contains the SQLJ translator and customizer options used in the `bldsqljs.bat` batch file on Windows operating systems. These are the options DB2 recommends that you use to build SQLJ routines (stored procedures and user-defined functions).

Translator and customizer options for **bldsqljs.bat**
sqlj The SQLJ translator (also compiles the program).
%1.sqlj The SQLJ source file.
db2sqljcustomize The DB2 for Java profile customizer.
-url Specifies a JDBC URL for establishing a database connection, such as `jdbc:db2://servername:50000/sample`.
-user Specifies a user ID.
-password Specifies a password.
%1_SJProfile0 Specifies a serialized profile for the program.

Related tasks:

- "Building SQLJ routines" on page 121

Related reference:

- "SQLJ application and applet options for Windows" on page 121

Related samples:

- "bldsqljs.bat -- Builds a Java embedded SQL (SQLJ) stored procedure on Windows"

Chapter 5. The Command Line Processor

This chapter provides detailed information for using the Command Line Processor for accessing a DB2 database, by running CLP scripts, and by calling DB2 stored procedures on the command line with the CALL statement.

SQL procedures are also coded in CLP scripts. Information on creating SQL procedures is in Chapter 6, "SQL procedures," on page 133.

For the latest DB2 application development updates, visit the Web page at:

```
http://www.ibm.com/software/data/db2/udb/ad
```

Running Command Line Processor (CLP) scripts

The CLP is directly accessible from the command line of a DB2 instance. It can be run interactively by entering "db2" on the command line. It can also be run in non-interactive mode by entering DB2 commands, or DB2 scripts containing DB2 commands, preceded by the keyword "db2". The examples used here will be in non-interactive mode.

CLP scripts code SQL statements directly rather than embed them in a host language so the user only needs to know SQL to program a CLP script. And with CLP scripts there is no need to compile and link source files before running the script.

DB2 provides CLP scripts for creating and running SQL statements against the `sample` database. These are located in the `sqllib/samples/clp` directory on UNIX, and the `sqllib\samples\clp` directory on Windows. The README file in the directory describes the programs and how to run them.

Procedure:

The CLP script, `cte.db2`, defines two common table expressions, `PAYLEVEL` and `PAYBYED` which are accessed by a select statement.

To run this script:

1. first connect to the sample database:

   ```
   db2 connect to sample
   ```

2. then enter the following command:

   ```
   db2 -vf cte.db2 -t
   ```

   The "v" is the verbose flag which is not required, but provides detailed output so is recommended.

The result is displayed on the screen by default. It gives the SQL common table expression and the output from it:

```
WITH PAYLEVEL AS (SELECT EMPNO, YEAR(HIREDATE) AS HIREYEAR, EDLEVEL,
SALARY+BONUS+COMM AS TOTAL_PAY FROM EMPLOYEE WHERE EDLEVEL > 16 ),
PAYBYED (EDUC_LEVEL, YEAR_OF_HIRE, AVG_TOTAL_PAY) AS (SELECT EDLEVEL,
HIREYEAR, AVG(TOTAL_PAY) FROM PAYLEVEL GROUP BY EDLEVEL, HIREYEAR )
SELECT EMPNO, EDLEVEL, YEAR_OF_HIRE, TOTAL_PAY, AVG_TOTAL_PAY FROM
PAYLEVEL, PAYBYED WHERE EDLEVEL=EDUC_LEVEL AND HIREYEAR = YEAR_OF_HIRE
AND TOTAL_PAY < AVG_TOTAL_PAY

EMPNO  EDLEVEL YEAR_OF_HIRE TOTAL_PAY     AVG_TOTAL_PAY
------ ------- ------------ ------------- --------------------------------
000210      17         1979      20132.00      25896.500000000000000000000

  1 record(s) selected.
```

Related concepts:

- "Command Line Processor (CLP)" in the *Command Reference*

Related tasks:

- "Calling procedures from the Command Line Processor (CLP)" on page 130

Related reference:

- "Command Line Processor (CLP) samples" on page 73
- "db2 - Command Line Processor Invocation Command" in the *Command Reference*

Related samples:

- "cte.db2 -- How to create a COMMON TABLE EXPRESSION "

Calling procedures from the Command Line Processor (CLP)

You can call stored procedures by using the CALL statement from the DB2 command line processor interface. The stored procedure being called must be defined in the DB2 system catalog tables.

Procedure:

To call the stored procedure, first connect to the database:

```
db2 connect to sample user userid using password
```

where *userid* and *password* are the user ID and password of the instance where the sample database is located.

To use the CALL statement, enter the stored procedure name plus any IN or INOUT parameter values, as well as '?' as a place-holder for each OUT parameter value.

The parameters for a stored procedure are given in the CREATE PROCEDURE statement for the stored procedure in the program source file.

SQL procedure example

For information on creating an SQL procedure, see Creating SQL procedures.

In the whiles.db2 file, the CREATE PROCEDURE statement for the DEPT_MEDIAN procedure signature is as follows:

```
CREATE PROCEDURE DEPT_MEDIAN
(IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
```

To invoke this procedure, use the CALL statement in which you must specify the procedure name and appropriate parameter arguments, which in this case are the value for the IN parameter, and a question mark, '?', for the value of the OUT parameter. The procedure's SELECT statement uses the `deptNumber` value on the DEPT column of the STAFF table, so to get meaningful output the IN parameter needs to be a valid value from the DEPT column; for example, the value "51":

```
db2 call dept_median (51, ?)
```

Note: On UNIX platforms the parentheses have special meaning to the command shell, so they must be preceded with a "\" character or surrounded with quotes, as follows:

```
db2 "call dept_median (51, ?)"
```

You do not use quotes if you are using the interactive mode of the command line processor.

After running the above command, you should receive this result:

```
Value of output parameters
--------------------------
Parameter Name  : MEDIANSALARY
Parameter Value : +1.76545000000000E+004

Return Status = 0
```

C stored procedure example

You can also call stored procedures created from supported host languages with the Command Line Processor. In the `samples/c` directory on UNIX, and the `samples\c` directory on Windows, DB2 provides files for creating stored procedures. The `spserver` shared library contains a number of stored procedures that can be created from the source file, `spserver.sqc`. The `spcreate.db2` file catalogs the stored procedures.

In the `spcreate.db2` file, the `CREATE PROCEDURE` statement for the `MAIN_EXAMPLE` procedure begins:

```
CREATE PROCEDURE MAIN_EXAMPLE (IN job CHAR(8),
                               OUT salary DOUBLE,
                               OUT errorcode INTEGER)
```

To call this stored procedure, you need to put in a CHAR value for the IN parameter, `job`, and a question mark, '?', for each of the OUT parameters. The procedure's SELECT statement uses the `job` value on the JOB column of the EMPLOYEE table, so to get meaningful output the IN parameter needs to be a valid value from the JOB column. The C sample program, `spclient`, that calls the stored procedure, uses `'DESIGNER'` for the `JOB` value. We can do the same, as follows:

```
db2 "call MAIN_EXAMPLE ('DESIGNER', ?, ?)"
```

After running the above command, you should receive this result:

```
Value of output parameters
--------------------------
Parameter Name  : SALARY
Parameter Value : +2.37312500000000E+004

Parameter Name  : ERRORCODE
Parameter Value : 0

Return Status = 0
```

An ERRORCODE of zero indicates a successful result.

Comparing with the spclient program, notice that spclient has formatted the result in decimal for easier viewing:

```
CALL stored procedure named MAIN_EXAMPLE
Stored procedure returned successfully
Average salary for job DESIGNER =  23731.25
```

Related tasks:

- "Creating SQL procedures" on page 133
- "Calling SQL procedures with client applications" on page 134
- "Calling procedures from triggers or SQL routines" in the *Application Development Guide: Programming Server Applications*
- "Calling procedures from applications or external routines" in the *Application Development Guide: Programming Server Applications*

Related samples:

- "spclient.sqc -- Call various stored procedures (C)"
- "spcreate.db2 -- How to catalog the stored procedures contained in spserver.sqc (C)"
- "spserver.sqc -- Definition of various types of stored procedures (C)"
- "whiles.db2 -- To create the DEPT_MEDIAN SQL procedure "
- "whiles.sqc -- To call the DEPT_MEDIAN SQL procedure"

Chapter 6. SQL procedures

Creating SQL procedures

The DB2 Command Line Processor scripts (those ending with the `.db2` extension), in the `sqllib/samples/sqlproc` directory on UNIX and the `sqllib\samples\sqlproc` directory on Windows, execute CREATE PROCEDURE statements to create stored procedures on the server. Each CLP script has a corresponding client application file of the same name, with an extension denoting its language and application interface: `.sqc` (for C embedded SQL), `.c` (for DB2 CLI), or `.java` (for JDBC).

Notes:

1. Beginning with DB2 Version 8.2, the creation of SQL procedures does not require a C or C++ compiler on the server, so no C or C++ compiler setup is required. When an SQL procedure is created, its procedural statements are converted to a native representation that is stored in the database catalogs, as is done with other SQL statements. When an SQL procedure is called, this representation is loaded from the catalogs and is executed by the DB2 engine.
2. CALL is an SQL statement in DB2 Version 8. This means that you can no longer create procedures in any order. The compiler checks for the existence of invoked procedures at compilation time and raises an error with SQLCODE -440 if the procedure is not found.

Procedure:

Before running a CREATE PROCEDURE CLP script, connect to the sample database with the command:

```
db2 connect to sample user userid using password
```

where *userid* and *password* are the user ID and password of the instance where the `sample` database is located.

To execute the CREATE PROCEDURE statement contained in the `rsultset.db2` script file, enter the following command:

```
db2 -td@ -vf rsultset.db2
```

Now, the SQL procedure is ready to be called.

Related tasks:

- "Customizing precompile and bind options for SQL procedures" on page 135
- "Rebinding SQL procedures" on page 137

Related samples:

- "rsultset.db2 -- To register and create the MEDIAN_RESULT_SET SQL procedure"

Calling SQL procedures with client applications

After you have created an SQL procedure, as explained in "Creating SQL procedures" on page 133, you can call the SQL procedure by building and running a client application. DB2 supplies sample client programs in `sqllib/samples/sqlproc` (UNIX), and in `sqllib\samples\sqlproc` (Windows). There are client source files for DB2 CLI, C embedded SQL, and JDBC. DB2 supplies command line processor scripts in the UNIX directory, and batch files in the Windows directory, to call the SQL procedures..

Procedure:

Depending on the application interface you are using, you can build and run a sample client program to call SQL procedures by following these examples:

DB2 CLI

To build the DB2 CLI client application, `rsultset`, from the source file `rsultset.c`, enter:

```
bldcli rsultset
```

This command creates the executable file, `rsultset` on UNIX, and `rsultset.exe` on Windows.

To call the SQL procedure, run the sample client application by entering the executable file name, the name of the database to which you are connecting, and the user ID and password of the database instance:

```
rsultset database userid password
```

C embedded SQL

To build the embedded SQL client application, `basecase`, from the source file `basecase.sqc`, enter the script file name, the executable name, the database to which you are connecting, and the user ID and password of the database instance:

```
bldapp basecase database userid password
```

The result is an executable file, `basecase` on UNIX, and `basecase.exe` on Windows.

To call the SQL procedure, run the sample client application by entering:

```
basecase database userid password
```

JDBC To build the JDBC client application, `NestedSP`, from the source file `NestedSP.java`, compile the source file:

```
javac NestedSP.java
```

The result is the class file, `NestedSP.class`.

To call the SQL procedure, run the java interpreter on the application:

```
java NestedSP database userid password
```

Related tasks:

- "Creating SQL procedures" on page 133
- "Calling procedures from the Command Line Processor (CLP)" on page 130
- "Rebinding SQL procedures" on page 137

Related samples:

- "basecase.sqc -- To call the UPDATE_SALARY SQL procedure"
- "NestedSP.java -- Client application for invoking nested stored procedures "
- "rsultset.c -- To call the MEDIAN_RESULT_SET SQL procedure"

Customizing precompile and bind options for SQL procedures

Procedure:

The precompile and bind options for SQL procedures can be customized by setting the instance-wide DB2 registry variable, DB2_SQLROUTINE_PREPOPTS with the command:

```
db2set DB2_SQLROUTINE_PREPOPTS=<options>
```

Only the following options are allowed:

```
BLOCKING {UNAMBIG | ALL | NO}
DATETIME {DEF | USA | EUR | ISO | JIS | LOC}
DEGREE {1 | degree-of-parallelism | ANY}
DYNAMICRULES {BIND | RUN}
EXPLAIN {NO | YES | ALL}
EXPLSNAP {NO | YES | ALL}
FEDERATED {NO | YES}
INSERT {DEF | BUF}
ISOLATION {CS |RR |UR |RS |NC}
QUERYOPT optimization-level
VALIDATE {RUN | BIND}
```

These options can be changed at the procedure level with the SET_ROUTINE_OPTS stored procedure. The values of the options set for the creation of SQL procedures in the current session can be obtained with the GET_ROUTINE_OPTS function.

Example.

The SQL procedures used in this example will be defined in CLP scripts (given below). These scripts are not in the sqlproc samples directory, but you can easily create these files by cutting-and-pasting the CREATE procedure statements into your own files.

The examples use a table named "expenses", which you can create in the sample database as follows:

```
db2 connect to sample
db2 CREATE TABLE expenses(amount DOUBLE, date DATE)
db2 connect reset
```

To begin, we specify the use of ISO format for dates as an instance-wide setting:

```
db2set DB2_SQLROUTINE_PREPOPTS="DATETIME ISO"
db2stop
db2start
```

Stopping and restarting DB2 is necessary for the change to take affect.

Then connect to the database:

```
db2 connect to sample
```

The first procedure is defined in CLP script maxamount.db2 as follows:

```
CREATE PROCEDURE maxamount(OUT maxamnt DOUBLE)
BEGIN
  SELECT max(amount) INTO maxamnt FROM expenses;
END @
```

It will be created with options DATETIME ISO and ISOLATION UR:

```
db2 "CALL SET_ROUTINE_OPTS(GET_ROUTINE_OPTS() || ' ISOLATION UR')"
db2 -td@ -vf maxamount.db2
```

The next procedure is defined in CLP script fullamount.db2 as follows:

```
CREATE PROCEDURE fullamount(OUT fullamnt DOUBLE)
BEGIN
  SELECT sum(amount) INTO fullamnt FROM expenses;
END @
```

It will be created with option ISOLATION CS (note that we are not using the instance-wide DATETIME ISO setting in this case):

```
CALL SET_ROUTINE_OPTS('ISOLATION CS')
db2 -td@ -vf fullamount.db2
```

The last procedure in our example is defined in CLP script perday.db2 as follows:

```
CREATE PROCEDURE perday()
BEGIN
  DECLARE cur1 CURSOR WITH RETURN FOR
    SELECT date, sum(amount)
    FROM expenses
    GROUP BY date;

    OPEN cur1;
END @
```

The last SET_ROUTINE_OPTS call uses the NULL value as the argument. This restores the global setting specified in the DB2_SQLROUTINE_PREPOPTS registry, so the last procedure will be created with option DATETIME ISO:

```
CALL SET_ROUTINE_OPTS(NULL)
db2 -td@ -vf perday.db2
```

Related tasks:

- "Backing up and restoring SQL procedures created prior to DB2 8.2" on page 136
- "Creating SQL procedures" on page 133
- "Calling procedures from the Command Line Processor (CLP)" on page 130
- "Calling SQL procedures with client applications" on page 134
- "Rebinding SQL procedures" on page 137

Related reference:

- "PRECOMPILE Command" in the *Command Reference*

Backing up and restoring SQL procedures created prior to DB2 8.2

In DB2 Version 8.2, SQL procedures are first class database objects, in the sense that they are managed completely inside the database, in a similar way to triggers and views. Therefore, they require no special consideration during backup and restore. Procedures created prior to Version 8.2 do require some special consideration, explained below, as they involve the creation of DLLs (dynamic linked libraries) in the file system. When an SQL procedure is created in a version

of DB2 prior to V8.2, the generated shared dynamic linked library (DLL) is kept in the database catalog, along with the source text, package, and related files. Therefore, all this information is saved when you perform a database backup.

Procedure:

At database recovery time, all SQL procedure executables on the filesystem that belong to the database being recovered will be removed. If the index creation configuration parameter, `indexrec`, is set to RESTART, all SQL procedure executables will be extracted from the catalog table and put back on the filesystem at next connect time. Otherwise, the SQL executables will be extracted on first execution of the SQL procedures.

The executables will be put back in the following directory:

UNIX $HOME/sqllib/function/routine/sqlproc/<*database_name*>

Windows
sqllib\function\routine\sqlproc\<*database_name*>

where <*database_name*> represents the database with which the SQL procedures were created.

If the first attempt to connect to a database after a restore operation returns:

```
SQL2048N An error occurred while accessing object "SQL PROCEDURE FILES".
Reason code: "7".
```

Stop DB2 with `db2stop`, and restart with `db2start`.

Related tasks:

- "Customizing precompile and bind options for SQL procedures" on page 135
- "Creating SQL procedures" on page 133
- "Calling procedures from the Command Line Processor (CLP)" on page 130
- "Calling SQL procedures with client applications" on page 134
- "Rebinding SQL procedures" on page 137

Rebinding SQL procedures

Procedure:

To rebind the package corresponding to an SQL procedure, call the `SYSPROC.REBIND_ROUTINE_PACKAGE` built-in stored procedure.

For example, if an SQL procedure named `MYSCHEMA.MYPROC` exists in the database, its package can be rebound from the command line processor (CLP) by issuing the following command:

```
CALL SYSPROC.REBIND_ROUTINE_PACKAGE('P', 'MYSCHEMA.MYPROC', 'CONSERVATIVE')
```

where `'P'` indicates that `'MYSCHEMA.MYPROC'` is a procedure name. A value of `'SP'` for the first parameter would indicate that `'MYSCHEMA.MYPROC'` is a specific procedure name. `'CONSERVATIVE'` indicates that conservative rebinding semantics should be applied. See the REBIND command in the related links below for more details on conservative rebinding.

Related tasks:

- "Customizing precompile and bind options for SQL procedures" on page 135
- "Backing up and restoring SQL procedures created prior to DB2 8.2" on page 136
- "Creating SQL procedures" on page 133
- "Calling procedures from the Command Line Processor (CLP)" on page 130
- "Calling SQL procedures with client applications" on page 134

Related reference:

- "REBIND Command" in the *Command Reference*

Chapter 7. Perl

This chapter provides detailed information for building Perl programs to access a DB2 database.

For the latest DB2 application development updates, visit the Web page at:

`http://www.ibm.com/software/data/db2/udb/ad`

Building Perl applications

DB2 supports database access for client applications written in Perl. At the time of printing, Release 0.76 of the DB2 UDB driver (DBD::DB2) for the Perl Database Interface (Perl DBI) Version 0.93 or later is available for AIX, HP-UX, Linux, Solaris and Windows. For information on how to obtain the latest driver, visit:

`http://www.ibm.com/software/data/db2/perl`

DB2 provides Perl sample programs located on UNIX in the `sqllib/samples/perl` directory, and on Windows in the `sqllib\samples\perl` directory.

Setup for 64-bit Environments on UNIX

On UNIX, if you are not using a 64-bit version of Perl, the same process should be followed as is recommended for running 32-bit applications in 64-bit environments in "Migrating applications from 32-bit to 64-bit environments" on page 51 except the applications do not have to be rebound. If using a wrapper program you would substitute `perl $1` for `$1`.

Note: In the following examples, if using a wrapper program for UNIX 64-bit environments, instead of `perl <program_name>`, substitute `<wrapper_name> <program_name>`.

Procedure:

To run the perl interpreter on a DB2 Perl program on the command line, enter the interpreter name and the program name (including extension):

- If connecting locally on the server:

  ```
  perl dbauth.pl
  ```

- If connecting from a remote client:

  ```
  perl dbauth.pl sample <userid> <password>
  ```

Some programs require support files to be run. The `tbsel` sample program requires several tables created by the `tbselcreate.db2` CLP script. The `tbselinit` script (UNIX), or the `tbselinit.bat` batch file (Windows), first calls `tbseldrop.db2` to drop the tables if they exist, and then calls `tbselcreate.db2` to create them. So to run the program, you would enter the following commands:

- If connecting locally on the server:

  ```
  tbselinit
  perl tbsel.pl
  ```

- If connecting from a remote client:

```
tbselinit
perl tbsel.pl sample <userid> <password>
```

Note: For a remote client, you need to modify the connect statement in the `tbselinit` or `tbselinit.bat` file to hardcode your user ID and password:
`db2 connect to sample user <userid> using <password>`

Calling routines

DB2 client applications can access routines (stored procedures and user-defined functions) that are created by supported host languages or by SQL procedures. For example, the sample program `spclient.pl` can access the SQL procedures `spserver` shared library, if it exists in the database.

Note: To build a host language routine, you must have the appropriate compiler set up on the server. SQL procedures do not require a compiler. The shared library can only be built on the server, and not from a remote client.

To demonstrate calling SQL procedures, go to the `samples/sqlproc` directory (UNIX) or the `samples\sqlproc` directory (Windows) on the server, and run the following commands to create and catalog the SQL procedures in the `spserver` library:

```
db2 connect to sample
db2 -td@ -vf spserver.db2
```

Next, come back to the `perl` samples directory (this can be on a remote client machine), and run the Perl interpreter on the client program to access the `spserver` shared library:

- If connecting locally on the server, enter the following:

  ```
  perl spclient
  ```

- If connecting from a remote client, enter the following:

  ```
  perl spclient sample <userid> <password>
  ```

Related concepts:

- "Programming Considerations for Perl" in the *Application Development Guide: Programming Client Applications*
- "Perl DBI" in the *Application Development Guide: Programming Client Applications*

Related tasks:

- "Migrating applications from 32-bit to 64-bit environments" on page 51

Related reference:

- "Perl Samples" on page 86

Chapter 8. PHP

This chapter provides detailed information for building PHP programs to access a DB2 database.

For the latest DB2 application development updates, visit the Web page at:

```
http://www.ibm.com/software/data/db2/udb/ad
```

Building PHP applications

DB2 supports database access for client applications written in PHP. PHP is a server-side, HTML-embedded, cross-platform scripting language. It supports DB2 access using the Unified-ODBC access method, in which the user-level PHP communicates to DB2 using ODBC calls. Unlike standard ODBC, with the Unified-ODBC method, communication is directly to the DB2 CLI layer, not through the ODBC layer. For more information about using PHP with DB2, search the DB2 support site:

```
http://www.ibm.com/software/data/db2/udb/winos2unix/support
```

At the time of printing, the latest version is PHP 4.3.4. You can download the latest version of PHP from:

```
http://www.php.net
```

After downloading the tar file, compile it with the option `--with-ibm-db2=<DIR>`, where `<DIR>` is determined by the platform path:

- On UNIX, the option would be: `--with-ibm-db2=$HOME/sqllib`
- On Windows, the option would be `--with-ibm-db2=%DB2PATH%`

After untarring the `php-4.3.4.tar` file (or the file of a later PHP version, if available), the php-4.3.4 directory contains the `php.ini-dist` file which has the default settings for new PHP installations. The file has to be copied into the installation path with the file name changed to `php.ini`. The following commands assume a typical installation path:

- On UNIX:

```
cd ../php-4.x.y
cp php.ini-dist /usr/local/lib/php.ini
```

- On Windows:

```
cd ..\php-4.x.y
copy php.ini-dist C:\Windows\php.ini
```

Note: A different path can be specified during install by using the following option:

```
--with-config-file-path=<path>
```

If installing using RPM on Linux, the default path for `php.ini` is `/etc`. For RPM on Linux, create a new file, `.odbc.ini`, under the home directory and include the following to configure ODBC with DB2:

```
[ODBC Data Sources]
Sample          = <description>

[Sample]
Driver          = $HOME/sqllib/lib/libdb2.so
Description     = <description>
Host            =  localhost
UserName        = <user>
Password        =  <password>
Database        =  sample
```

If installing using InstallShield on Windows, add the IBM DB2 ODBC driver and configure it for the sample database using Data Sources (ODBC) in administrative tools.

See the INSTALL file for more information.

You can edit the `php.ini` file to set PHP options. DB2 recommends that you set the following options for running the DB2 sample programs:

```
track_errors = On
register_globals = On
register_argc_argv = On
max_execution_time = 60
odbc.defaultlrl = 100000
```

Notes:

1. `track_errors = On`: allows errors to be tracked.
2. `register_globals = On`: if `register_globals = Off`, then you should use `$_SERVER['argc']` and `$_SERVER['argv'][0]` instead of `$argc` and `$argv` respectively.
3. `register_argc_argv = On`: allows command line arguments.
4. `max_execution_time = 60`: some scripts may take longer to run so the default value needs to be changed.
5. `odbc.defaultlrl = 100000`: this is in bytes, large enough to hold the BLOB data in the samples.

DB2 provides PHP sample programs located on UNIX in the `sqllib/samples/php` directory, and on Windows in the `sqllib\samples\php` folder.

Setup for 64-bit Environments on UNIX

On UNIX, if you are not using a 64-bit version of PHP, the same process should be followed as is recommended for running 32-bit applications in 64-bit environments in "Migrating applications from 32-bit to 64-bit environments" on page 51 except the applications do not have to be rebound. If using the wrapper program you would substitute `php $1` for `$1`.

Note: In the following examples, if using a wrapper program for UNIX 64-bit environments, instead of `php <program_name>`, substitute `<wrapper_name> <program_name>`.

Procedure:

To run the php interpreter on a DB2 PHP source file on the command line, enter the interpreter name and the source file name (including extension):

- If connecting locally on the server:

```
php dbauth.php
```

- If connecting from a remote client:

```
php dbauth.php sample <userid> <password>
```

Some programs require support files to be run. The `tbsel` sample program requires several tables created by the `tbselcreate.db2` CLP script. The `tbselinit` script on UNIX, or the `tbselinit.bat` batch file on Windows, first calls `tbseldrop.db2` to drop the tables if they exist, and then calls `tbselcreate.db2` to create them. So to run the program, you would enter the following commands:

- If connecting locally on the server:

```
tbselinit
php tbsel.php
```

- If connecting from a remote client:

```
tbselinit
php tbsel.php sample <userid> <password>
```

Note: For a remote client, you need to modify the connect statement in the `tbselinit` or `tbselinit.bat` file to hardcode your user ID and password: `db2 connect to sample user <userid> using <password>`

Calling user-defined functions

DB2 client applications can access user-defined functions that are created by supported host languages. For example, the sample program `udfcli.php` can access the C user-defined function `udfsrv` shared library, if it exists in the database.

Note: To build a host language user-defined function shared library, you must have the appropriate compiler set up on the server. The shared library can only be built on the server, and not from a remote client. PHP does not support client programs calling stored procedures.

Assuming a C compiler is set up on the server, demonstrate calling user-defined functions by going to the `samples/c` directory (UNIX) or the `samples\c` directory (Windows) on the server, and run the following command to create the `udfsrv` library in the database:

```
bldrtn udfsrv
```

Next, come back to the `php` samples directory (this can be on a remote client machine), and run the php interpreter on the client program to access the `udfsrv` shared library:

- If connecting locally on the server, enter the following:

```
php udfcli.php
```

- If connecting from a remote client, enter the following:

```
php udfcli.php sample <userid> <password>
```

Related tasks:

- "Migrating applications from 32-bit to 64-bit environments" on page 51

Related reference:

- "PHP samples" on page 87

Part 3. Building and Running Platform-Specific Applications

Chapter 9. UNIX

This chapter describes the common steps to build applications and routines for supported UNIX operating systems. The platform-specific details, such as compiler options, are provided in the platform chapters that follow this one.

Information for building C/C++ multi-threaded applications are contained in the platform chapters as each operating system requires specific building requirements for these programs.

Building UNIX C applications

DB2 provides build scripts for compiling and linking C embedded SQL and DB2 API programs. These are located in the `sqllib/samples/c` directory, along with sample programs that can be built with these files.

The build file, `bldapp`, contains the commands to build a DB2 application program.

The first parameter, `$1`, specifies the name of your source file. This is the only required parameter, and the only one needed for DB2 API programs that do not contain embedded SQL. Building embedded SQL programs requires a connection to the database so three optional parameters are also provided: the second parameter, `$2`, specifies the name of the database to which you want to connect; the third parameter, `$3`, specifies the user ID for the database, and `$4` specifies the password.

For an embedded SQL program, `bldapp` passes the parameters to the precompile and bind script, `embprep`. If no database name is supplied, the default `sample` database is used. The user ID and password parameters are only needed if the instance where the program is built is different from the instance where the database is located.

Procedure:

The following examples show you how to build and run DB2 API and embedded SQL applications.

To build the DB2 API non-embedded SQL sample program, `cli_info`, from the source file `cli_info.c`, enter:

```
bldapp cli_info
```

The result is an executable file, `cli_info`.

To run the executable file, enter the executable name:

```
cli_info
```

Building and Running Embedded SQL Applications

There are three ways to build the embedded SQL application, `tbmod`, from the source file `tbmod.sqc`:

1. If connecting to the sample database on the same instance, enter:
   ```
   bldapp tbmod
   ```
2. If connecting to another database on the same instance, also enter the database name:
   ```
   bldapp tbmod database
   ```
3. If connecting to a database on another instance, also enter the user ID and password of the database instance:
   ```
   bldapp tbmod database userid password
   ```

The result is an executable file, `tbmod`.

There are three ways to run this embedded SQL application:

1. If accessing the `sample` database on the same instance, simply enter the executable name:
   ```
   tbmod
   ```
2. If accessing another database on the same instance, enter the executable name and the database name:
   ```
   tbmod database
   ```
3. If accessing a database on another instance, enter the executable name, database name, and user ID and password of the database instance:
   ```
   tbmod database userid password
   ```

Related concepts:

- "Build files" on page 97

Related tasks:

- "Building UNIX C routines" on page 151

Related reference:

- "AIX C application compile and link options" on page 168
- "HP-UX C application compile and link options" on page 194
- "Linux C application compile and link options" on page 208
- "Solaris C application compile and link options" on page 222

Related samples:

- "bldapp -- Builds AIX C application programs (C)"
- "bldapp -- Builds HP-UX C applications (C)"
- "bldapp -- Builds Linux C applications (C)"
- "bldapp -- Builds Solaris C applications (C)"
- "cli_info.c -- Set and get information at the client level (C)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"
- "tbmod.sqc -- How to modify table data (C)"

Building UNIX C multi-connection applications

DB2 provides build scripts for compiling and linking C embedded SQL and DB2 API programs. These are located in the sqllib/samples/c directory, along with sample programs that can be built with these files.

The build file, bldmc, contains the commands to build a DB2 multi-connection program, requiring two databases. The compile and link options are the same as those used in bldapp.

The first parameter, $1, specifies the name of your source file. The second parameter, $2, specifies the name of the first database to which you want to connect. The third parameter, $3, specifies the second database to which you want to connect. These are all required parameters.

Note: The makefile hardcodes default values of "sample" and "sample2" for the database names ($2 and $3, respectively) so if you are using the makefile, and accept these defaults, you only have to specify the program name (the $1 parameter). If you are using the bldmc script, you must specify all three parameters.

Optional parameters are not required for a local connection, but are required for connecting to a server from a remote client. These are: $4 and $5 to specify the user ID and password, respectively, for the first database; and $6 and $7 to specify the user ID and password, respectively, for the second database.

Procedure:

For the multi-connection sample program, dbmcon, you require two databases. If the sample database is not yet created, you can create it by entering db2sampl on the command line. The second database, here called sample2, can be created with one of the following commands:

If creating the database locally:

```
db2 create db sample2
```

If creating the database remotely:

```
db2 attach to <node_name>
db2 create db sample2
db2 detach
db2 catalog db sample2 as sample2 at node <node_name>
```

where <node_name> is the node where the database resides.

Multi-connection also requires that the TCP/IP listener is running. To ensure it is, do the following:

1. Set the environment variable DB2COMM to TCP/IP as follows:

   ```
   db2set DB2COMM=TCPIP
   ```

2. Update the database manager configuration file with the TCP/IP service name as specified in the services file:

   ```
   db2 update dbm cfg using SVCENAME <TCP/IP service name>
   ```

   Each instance has a TCP/IP service name listed in the services file. Ask your system administrator if you cannot locate it or do not have the file permission to change the services file.

3. Stop and restart the database manager in order for these changes to take effect:

```
db2stop
db2start
```

The dbmcon program consists of five files:

dbmcon.sqc
Main source file for connecting to both databases.

dbmcon1.sqc
Source file for creating a package bound to the first database.

dbmcon1.h
Header file for dbmcon1.sqc included in dbmcon.sqc for accessing the SQL statements for creating and dropping a table to be bound to the first database.

dbmcon2.sqc
Source file for creating a package bound to the second database.

dbmcon2.h
Header file for dbmcon2.sqc included in dbmcon.sqc for accessing the SQL statements for creating and dropping a table to be bound to the second database.

To build the multi-connection sample program, dbmcon, enter:

```
bldmc dbmcon sample sample2
```

The result is an executable file, dbmcon.

To run the executable file, enter the executable name:

```
dbmcon
```

The program demonstrates a one-phase commit to two databases.

Related concepts:

- "Build files" on page 97

Related tasks:

- "Building UNIX C applications" on page 147

Related reference:

- "svcename - TCP/IP service name configuration parameter" in the *Administration Guide: Performance*
- "AIX C application compile and link options" on page 168
- "HP-UX C application compile and link options" on page 194
- "Linux C application compile and link options" on page 208
- "Solaris C application compile and link options" on page 222

Related samples:

- "bldmc -- Builds AIX C multi-connection applications (C)"
- "bldmc -- Builds HP-UX C multi-connection applications (C)"
- "bldmc -- Builds Linux C multi-connection applications (C)"
- "bldmc -- Builds Solaris C multi-connection applications (C)"
- "dbmcon.sqc -- How to use multiple databases (C)"

- "dbmcon1.h -- Function declarations for the source file, dbmcon1.sqc (C)"
- "dbmcon1.sqc -- Functions used in the multiple databases program dbmcon.sqc (C)"
- "dbmcon2.h -- Function declarations for the source file, dbmcon2.sqc (C)"
- "dbmcon2.sqc -- Functions used in the multiple databases program dbmcon.sqc (C)"

Building UNIX C routines

DB2 provides build scripts for compiling and linking C programs. These are located in the `sqllib/samples/c` directory, along with sample programs that can be built with these files.

The script, `bldrtn`, contains the commands to build routines (stored procedures and user-defined functions). The script compiles the routines into a shared library that can be loaded by the database manager and called by a client application.

The first parameter, `$1`, specifies the name of your source file. The second parameter, `$2`, specifies the name of the database to which you want to connect.

The database parameter is optional. If no database name is supplied, the program uses the default `sample` database. And since the stored procedure must be built on the same instance where the database resides, there are no parameters for user ID and password.

Procedure:

The following examples show you how to build routine shared libraries with:

- stored procedures
- non-embedded SQL user-defined functions (UDFs)
- embedded SQL user-defined functions (UDFs)

Stored Procedure Shared Library

To build the sample program `spserver` from the source file `spserver.sqc`:

1. If connecting to the `sample` database, enter the build script name and program name:

   ```
   bldrtn spserver
   ```

   If connecting to another database, also enter the database name:

   ```
   bldrtn spserver database
   ```

   The script copies the shared library to the server in the path `sqllib/function`.
2. Next, catalog the routines by running the `spcat` script on the server:

   ```
   spcat
   ```

   This script connects to the sample database, uncatalogs the routines if they were previously cataloged by calling `spdrop.db2`, then catalogs them by calling `spcreate.db2`, and finally disconnects from the database. You can also call the `spdrop.db2` and `spcreate.db2` scripts individually.
3. Then, if this is not the first time the stored procedure is built, stop and restart the database to ensure the new version of the shared library is recognized. You can do this by entering `db2stop` followed by `db2start` on the command line.

Once you build the shared library, spserver, you can build the client application, spclient, that accesses the shared library.

You can build spclient by using the script, bldapp.

To call the stored procedures in the shared library, run the sample client application by entering:

spclient *database userid password*

where

database
Is the name of the database to which you want to connect. The name could be sample, or its alias, or another database name.

userid Is a valid user ID.

password
Is a valid password for the user ID.

The client application accesses the shared library, spserver, and executes a number of stored procedure functions on the server database. The output is returned to the client application.

Non-embedded SQL UDF Shared Library

To build the user-defined function program, udfsrv, from the source file udfsrv.c, enter the build script name and program name:

```
bldrtn udfsrv
```

The script copies the UDF to the sqllib/function directory.

Once you build udfsrv, you can build the client application, udfcli, that calls it. DB2 CLI and embedded SQL versions of this program are provided. You can build the DB2 CLI udfcli client program from the source file udfcli.c, in sqllib/samples/cli, using the script, bldapp.

You can build the embedded SQL udfcli client program from the source file udfcli.sqc, in sqllib/samples/c, using the script, bldapp.

To call the UDFs in the shared library, run the client application by entering:

udfcli *database userid password*

where

database
Is the name of the database to which you want to connect. The name could be sample, or its alias, or another database name.

userid Is a valid user ID.

password
Is a valid password for the user ID.

The client application accesses the shared library, udfsrv, and executes the user-defined functions on the server database. The output is returned to the client application.

Embedded SQL UDF Shared Library

To build the embedded SQL user-defined function program, `udfemsrv`, from the source file `udfemsrv.sqc`, if connecting to the `sample` database, enter the build script name and program name:

```
bldrtn udfemsrv
```

If connecting to another database, also enter the database name:

```
bldrtn udfemsrv database
```

The script copies the UDF to the `sqllib/function` directory.

Once you build `udfemsrv`, you can build the client application, `udfemcli`, that calls it. You can build the `udfemcli` client program from the source file `udfemcli.sqc`, in `sqllib/samples/c`, using the script, `bldapp`.

To call the UDFs in the shared library, run the client application by entering:

`udfemcli` *database userid password*

where

database
Is the name of the database to which you want to connect. The name could be `sample`, or its alias, or another database name.

userid Is a valid user ID.

password
Is a valid password for the user ID.

The client application accesses the shared library, `udfemsrv`, and executes the user-defined functions on the server database. The output is returned to the client application.

Related concepts:

- "Build files" on page 97

Related tasks:

- "Building UNIX C applications" on page 147

Related reference:

- "AIX C routine compile and link options" on page 170
- "HP-UX C routine compile and link options" on page 196
- "Linux C routine compile and link options" on page 210
- "Solaris C routine compile and link options" on page 224

Related samples:

- "bldrtn -- Builds AIX C routines (stored procedures and UDFs) (C)"
- "bldrtn -- Builds HP-UX C routines (stored procedures and UDFs) (C)"
- "bldrtn -- Builds Linux C routines (stored procedures or UDFs) (C)"
- "bldrtn -- Builds Solaris C routines (stored procedures or UDFs) (C)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

- "spclient.sqc -- Call various stored procedures (C)"
- "spserver.sqc -- Definition of various types of stored procedures (C)"
- "udfcli.sqc -- Call a variety of types of user-defined functions (C)"
- "udfemcli.sqc -- Call a variety of types of embedded SQL user-defined functions. (C)"
- "udfemsrv.sqc -- Call a variety of types of embedded SQL user-defined functions. (C)"
- "udfsrv.c -- Defines a variety of types of user-defined functions (C)"

Building UNIX C++ applications

DB2 provides build scripts for compiling and linking C++ embedded SQL and DB2 API programs. These are located in the `sqllib/samples/cpp` directory, along with sample programs that can be built with these files.

The build file, `bldapp` contains the commands to build DB2 API and embedded SQL applications.

The first parameter, `$1`, specifies the name of your source file. This is the only required parameter, and the only one needed for DB2 API programs that do not contain embedded SQL. Building embedded SQL programs requires a connection to the database so three optional parameters are also provided: the second parameter, `$2`, specifies the name of the database to which you want to connect; the third parameter, `$3`, specifies the user ID for the database, and `$4` specifies the password.

For an embedded SQL program, `bldapp` passes the parameters to the precompile and bind script, `embprep`. If no database name is supplied, the default `sample` database is used. The user ID and password parameters are only needed if the instance where the program is built is different from the instance where the database is located.

Procedure:

The following examples show you how to build and run DB2 API and embedded SQL applications.

To build the non-embedded SQL sample program `cli_info` from the source file `cli_info.C`, enter:

```
bldapp cli_info
```

The result is an executable file, `cli_info`. You can run the executable file against the `sample` database by entering:

```
cli_info
```

Building and Running Embedded SQL Applications

There are three ways to build the embedded SQL application, `tbmod`, from the source file `tbmod.sqC`:

1. If connecting to the sample database on the same instance, enter:

   ```
   bldapp tbmod
   ```

2. If connecting to another database on the same instance, also enter the database name:

```
bldapp tbmod database
```

3. If connecting to a database on another instance, also enter the user ID and password of the database instance:

```
bldapp tbmod database userid password
```

The result is an executable file, `tbmod`.

There are three ways to run this embedded SQL application:

1. If accessing the `sample` database on the same instance, simply enter the executable name:

```
tbmod
```

2. If accessing another database on the same instance, enter the executable name and the database name:

```
tbmod database
```

3. If accessing a database on another instance, enter the executable name, database name, and user ID and password of the database instance:

```
tbmod database userid password
```

Related concepts:

- "Build files" on page 97

Related tasks:

- "Building UNIX C++ routines" on page 158

Related reference:

- "AIX C++ application compile and link options" on page 173
- "HP-UX C++ application compile and link options" on page 199
- "Linux C++ application compile and link options" on page 213
- "Solaris C++ application compile and link options" on page 227

Related samples:

- "bldapp -- Builds AIX C++ applications (C++)"
- "bldapp -- Builds HP-UX C++ applications (C++)"
- "bldapp -- Builds Linux C++ applications (C++)"
- "bldapp -- Builds Solaris C++ applications (C++)"
- "cli_info.C -- Set and get information at the client level (C++)"
- "tbmod.sqC -- How to modify table data (C++)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

Building UNIX C++ multi-connection applications

DB2 provides build scripts for compiling and linking C++ embedded SQL and DB2 API programs. These are located in the `sqllib/samples/cpp` directory, along with sample programs that can be built with these files.

The build file, `bldmc`, contains the commands to build a DB2 multi-connection program, requiring two databases. The compile and link options are the same as those used in `bldapp`.

The first parameter, $1, specifies the name of your source file. The second parameter, $2, specifies the name of the first database to which you want to connect. The third parameter, $3, specifies the second database to which you want to connect. These are all required parameters.

Note: The makefile hardcodes default values of ″sample″ and ″sample2″ for the database names ($2 and $3, respectively) so if you are using the makefile, and accept these defaults, you only have to specify the program name (the $1 parameter). If you are using the bldmc script, you must specify all three parameters.

Optional parameters are not required for a local connection, but are required for connecting to a server from a remote client. These are: $4 and $5 to specify the user ID and password, respectively, for the first database; and $6 and $7 to specify the user ID and password, respectively, for the second database.

Procedure:

For the multi-connection sample program, dbmcon, you require two databases. If the sample database is not yet created, you can create it by entering db2sampl on the command line. The second database, here called sample2, can be created with one of the following commands:

If creating the database locally:

```
db2 create db sample2
```

If creating the database remotely:

```
db2 attach to <node_name>
db2 create db sample2
db2 detach
db2 catalog db sample2 as sample2 at node <node_name>
```

where <node_name> is the node where the database resides.

Multi-connection also requires that the TCP/IP listener is running. To ensure it is, do the following:

1. Set the environment variable DB2COMM to TCP/IP as follows:

   ```
   db2set DB2COMM=TCPIP
   ```

2. Update the database manager configuration file with the TCP/IP service name as specified in the services file:

   ```
   db2 update dbm cfg using SVCENAME <TCP/IP service name>
   ```

   Each instance has a TCP/IP service name listed in the services file. Ask your system administrator if you cannot locate it or do not have the file permission to change the services file.

3. Stop and restart the database manager in order for these changes to take effect:

   ```
   db2stop
   db2start
   ```

The dbmcon program consists of five files:

dbmcon.sqC
Main source file for connecting to both databases.

dbmcon1.sqC
Source file for creating a package bound to the first database.

dbmcon1.h
Header file for dbmcon1.sqC included in dbmcon.sqC for accessing the SQL statements for creating and dropping a table to be bound to the first database.

dbmcon2.sqC
Source file for creating a package bound to the second database.

dbmcon2.h
Header file for dbmcon2.sqC included in dbmcon.sqC for accessing the SQL statements for creating and dropping a table to be bound to the second database.

To build the multi-connection sample program, dbmcon, enter:

```
bldmc dbmcon sample sample2
```

The result is an executable file, dbmcon.

To run the executable file, enter the executable name:

```
dbmcon
```

The program demonstrates a one-phase commit to two databases.

Related concepts:

- "Build files" on page 97

Related tasks:

- "Building UNIX C++ applications" on page 154

Related reference:

- "svcename - TCP/IP service name configuration parameter" in the *Administration Guide: Performance*
- "AIX C++ application compile and link options" on page 173
- "HP-UX C++ application compile and link options" on page 199
- "Linux C++ application compile and link options" on page 213
- "Solaris C++ application compile and link options" on page 227

Related samples:

- "bldmc -- Builds AIX C++ multi-connection applications (C++)"
- "bldmc -- Builds HP-UX C++ multi-connection applications (C++)"
- "bldmc -- Builds Linux C++ multi-connection applications (C++)"
- "bldmc -- Builds Solaris C++ multi-connection applications (C++)"
- "dbmcon.sqC -- How to use multiple databases (C++)"
- "dbmcon1.h -- Class declaration for the source file, dbmcon1.sqC (C++)"
- "dbmcon1.sqC -- Functions used in the multiple databases program dbmcon.sqC (C++)"
- "dbmcon2.h -- Class declaration for the source file, dbmcon2.sqC (C++)"
- "dbmcon2.sqC -- Functions used in the multiple databases program dbmcon.sqC (C++)"

Building UNIX C++ routines

DB2 provides build scripts for compiling and linking C++ programs. These are located in the sqllib/samples/cpp directory, along with sample programs that can be built with these files.

The script file bldrtn contains the commands to build routines. The script file compiles the routines into a shared library that can be loaded by the database manager and called by a client application.

The first parameter, $1, specifies the name of your source file. The second parameter, $2, specifies the name of the database to which you want to connect.

The database parameter is optional. If no database name is supplied, the program uses the default sample database. And since the stored procedure must be built on the same instance where the database resides, there are no parameters for user ID and password.

Procedure:

The following examples show you how to build routine shared libraries with:

- stored procedures
- non-embedded SQL user-defined functions (UDFs)
- embedded SQL user-defined functions (UDFs)

Stored Procedure Shared Library

To build the sample program spserver from the source file spserver.sqC:

1. If connecting to the sample database, enter the build script name and program name:

   ```
   bldrtn spserver
   ```

   If connecting to another database, also enter the database name:

   ```
   bldrtn spserver database
   ```

   The script file copies the shared library to the server in the path sqllib/function.
2. Next, catalog the routines by running the spcat script on the server:

   ```
   spcat
   ```

   This script connects to the sample database, uncatalogs the routines if they were previously cataloged by calling spdrop.db2, then catalogs them by calling spcreate.db2, and finally disconnects from the database. You can also call the spdrop.db2 and spcreate.db2 scripts individually.
3. Then, if this is not the first time the stored procedure is built, stop and restart the database to ensure the new version of the shared library is recognized. You can do this by entering db2stop followed by db2start on the command line.

Once you build the shared library, spserver, you can build the client application, spclient, that accesses the shared library. You can build spclient by using the script file, bldapp.

To call the stored procedures in the shared library, run the sample client application by entering:

`spclient` *database userid password*

where

database
Is the name of the database to which you want to connect. The name could be `sample`, or its alias, or another database name.

userid Is a valid user ID.

password
Is a valid password for the user ID.

The client application accesses the shared library, `spserver`, and executes a number of stored procedure functions on the server database. The output is returned to the client application.

Non-embedded SQL UDF Shared Library

To build the user-defined function program, `udfsrv`, from the source file `udfsrv.C`, enter the build script name and program name:

```
bldrtn udfsrv
```

The script file copies the UDF to the `sqllib/function` directory.

If necessary, set the file mode for the UDF so the database manager can access it.

Once you build `udfsrv`, you can build the client application, `udfcli`, that calls it. You can build `udfcli` from the source file `udfcli.sqC` using the script file, `bldapp`.

To call the UDFs in the shared library, run the client application by entering:

`udfcli` *database userid password*

where

database
Is the name of the database to which you want to connect. The name could be `sample`, or its alias, or another database name.

userid Is a valid user ID.

password
Is a valid password for the user ID.

The client application accesses the shared library, `udfsrv`, and executes the user-defined functions on the server database. The output is returned to the client application.

Embedded SQL UDF Shared Library

To build the embedded SQL user-defined function program, `udfemsrv` from the source file `udfemsrv.sqC`, if connecting to the `sample` database, enter the build script name and program name:

```
bldrtn udfemsrv
```

If connecting to another database, also enter the database name:

```
bldrtn udfemsrv database
```

The script file copies the UDF to the `sqllib/function` directory.

Once you build `udfemsrv`, you can build the client application, `udfemcli`, that calls it. You can build `udfemcli` from the source file `udfemcli.sqC` using the script file `bldapp`.

To call the UDFs in the shared library, run the client application by entering:

`udfemcli` *database userid password*

where

database
Is the name of the database to which you want to connect. The name could be `sample`, or its alias, or another database name.

userid Is a valid user ID.

password
Is a valid password for the user ID.

The client application accesses the shared library, `udfemsrv`, and executes the user-defined functions on the server database. The output is returned to the client application.

Related concepts:

- "Build files" on page 97

Related tasks:

- "Building UNIX C++ applications" on page 154

Related reference:

- "AIX C++ routine compile and link options" on page 174
- "HP-UX C++ routine compile and link options" on page 201
- "Linux C++ routine compile and link options" on page 215
- "Solaris C++ routine compile and link options" on page 229

Related samples:

- "bldrtn -- Builds AIX C++ routines (stored procedures and UDFs) (C++)"
- "bldrtn -- Builds HP-UX C++ routines (stored procedures and UDFs) (C++)"
- "bldrtn -- Builds Linux C++ routines (stored procedures and UDFs) (C++)"
- "bldrtn -- Builds Solaris C++ routines (stored procedures or UDFs) (C++)"
- "spclient.sqC -- Call various stored procedures (C++)"
- "spserver.sqC -- Definition of various types of stored procedures (C++)"
- "udfcli.sqC -- Call a variety of types of user-defined functions (C++)"
- "udfemcli.sqC -- Call a variety of types of embedded SQL user-defined functions. (C++)"
- "udfemsrv.sqC -- Call a variety of types of embedded SQL user-defined functions. (C++)"
- "udfsrv.C -- Defines a variety of types of user-defined functions (C++)"

- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

Building UNIX Micro Focus COBOL applications

DB2 provides build scripts for compiling and linking Micro Focus COBOL embedded SQL and DB2 API programs. These are located in the `sqllib/samples/cobol_mf` directory, along with sample programs that can be built with these files.

The build file, `bldapp` contains the commands to build a DB2 application program.

The first parameter, `$1`, specifies the name of your source file. This is the only required parameter for programs that do not contain embedded SQL. Building embedded SQL programs requires a connection to the database so three optional parameters are also provided: the second parameter, `$2`, specifies the name of the database to which you want to connect; the third parameter, `$3`, specifies the user ID for the database, and `$4` specifies the password.

For an embedded SQL program, `bldapp` passes the parameters to the precompile and bind script, `embprep`. If no database name is supplied, the default `sample` database is used. The user ID and password parameters are only needed if the instance where the program is built is different from the instance where the database is located.

Procedure:

To build the non-embedded SQL sample program, `client`, from the source file `client.cbl`, enter:

```
bldapp client
```

The result is an executable file `client`. You can run the executable file against the `sample` database by entering:

```
client
```

Building and Running Embedded SQL Applications

There are three ways to build the embedded SQL application, `updat`, from the source file `updat.sqb`:

1. If connecting to the sample database on the same instance, enter:
   ```
   bldapp updat
   ```
2. If connecting to another database on the same instance, also enter the database name:
   ```
   bldapp updat database
   ```
3. If connecting to a database on another instance, also enter the user ID and password of the database instance:
   ```
   bldapp updat database userid password
   ```

The result is an executable file, `updat`.

There are three ways to run this embedded SQL application:

1. If accessing the `sample` database on the same instance, simply enter the executable name:

```
updat
```

2. If accessing another database on the same instance, enter the executable name and the database name:
   ```
   updat database
   ```
3. If accessing a database on another instance, enter the executable name, database name, and user ID and password of the database instance:
   ```
   updat database userid password
   ```

Related tasks:

- "Building UNIX Micro Focus COBOL routines" on page 162

Related reference:

- "AIX Micro Focus COBOL application compile and link options" on page 188
- "HP-UX Micro Focus COBOL application compile and link options" on page 204
- "Solaris Micro Focus COBOL application compile and link options" on page 232
- "Linux Micro Focus COBOL application compile and link options" on page 219

Related samples:

- "bldapp -- Builds AIX Micro Focus COBOL applications"
- "bldapp -- Builds HP-UX Micro Focus COBOL applications"
- "bldapp -- Builds Linux Micro Focus COBOL applications"
- "bldapp -- Builds Solaris Micro Focus COBOL applications"
- "client.cbl -- How to set and query a client (MF COBOL)"
- "updat.sqb -- How to update, delete and insert table data (MF COBOL)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

Building UNIX Micro Focus COBOL routines

DB2 provides build scripts for compiling and linking Micro Focus COBOL embedded SQL and DB2 API programs. These are located in the `sqllib/samples/cobol_mf` directory, along with sample programs that can be built with these files.

The script, `bldrtn`, contains the commands to build routines (stored procedures). The script compiles the routine source file into a shared library that can be called by a client application.

The first parameter, `$1`, specifies the name of your source file. The script uses the source file name for the shared library name. The second parameter, `$2`, specifies the name of the database to which you want to connect. Since the shared library must be built in the same instance where the database resides, there are no parameters for user ID and password.

Only the first parameter, source file name, is required. Database name is optional. If no database name is supplied, the program uses the default `sample` database.

Solaris-specific settings

Before building Micro Focus routines on Solaris, run the following commands:

```
db2stop
db2set DB2LIBPATH=$LD_LIBRARY_PATH
db2set DB2ENVLIST="COBDIR LD_LIBRARY_PATH"
db2set
db2start
```

Ensure that `db2stop` stops the database. The last `db2set` command is issued to check your settings: make sure `DB2LIBPATH` and `DB2ENVLIST` are set correctly.

Procedure:

To build the sample program `outsrv` from the source file `outsrv.sqb`, if connecting to the sample database, enter:

```
bldrtn outsrv
```

If connecting to another database, also enter the database name:

```
bldrtn outsrv database
```

The script file copies the shared library to the server in the path `sqllib/function`.

Once you build the stored procedure `outsrv`, you can build the client application `outcli` that calls it. You can build `outcli` using the script file, `bldapp`.

To call the stored procedure, run the sample client application by entering:

```
outcli database userid password
```

where

database
: Is the name of the database to which you want to connect. The name could be `sample`, or its alias, or another name.

userid Is a valid user ID.

password
: Is a valid password for the user ID.

The client application accesses the shared library, `outsrv`, and executes the stored procedure function of the same name on the server database. The output is then returned to the client application.

Related tasks:

- "Building UNIX Micro Focus COBOL applications" on page 161

Related reference:

- "AIX Micro Focus COBOL routine compile and link options" on page 190
- "HP-UX Micro Focus COBOL routine compile and link options" on page 206
- "Solaris Micro Focus COBOL routine compile and link options" on page 234
- "Linux Micro Focus COBOL routine compile and link options" on page 220

Related samples:

- "bldrtn -- Builds AIX Micro Focus COBOL routines (stored procedures)"
- "bldrtn -- Builds HP-UX Micro Focus COBOL routines (stored procedures)"
- "bldrtn -- Builds Linux Micro Focus COBOL routines (stored procedures)"
- "bldrtn -- Builds Solaris Micro Focus COBOL routines (stored procedures)"

- "outcli.sqb -- Call stored procedures using the SQLDA structure (MF COBOL)"
- "outsrv.sqb -- Demonstrates stored procedures using the SQLDA structure (MF COBOL)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

Chapter 10. AIX

This chapter provides detailed information for building applications on AIX. For the latest DB2 application development updates for AIX, visit the Web page at:

```
http://www.ibm.com/software/data/db2/udb/ad
```

Important considerations

This section gives AIX-specific information for building DB2 applications on various supported compilers.

AIX export files for routines

External routines are compiled on the server, and stored and executed in shared libraries on the server. These shared libraries are created when you compile the routines.

On AIX®, you must either provide an export file which specifies which global functions in the library are callable from outside it, or you can create AIX shared objects and libraries with 'export all' behaviour using the compiler, as in the following example:

```
xlC -qmkshrobj -q32 -g dlsource.C -o libtestit.so
```

The DB2® samples use an export file. This file must include the names of all routines in the library. Other UNIX® platforms simply export all global functions in the library. This is an example of an AIX export file:

```
#! spserver export file
outlanguage
```

The export file `spserver.exp` lists the stored procedure `outlanguage`. The linker uses `spserver.exp` to create the shared library `spserver` that contains the `outlanguage` stored procedure.

The AIX linker documentation has additional information on export files.

Related concepts:

- "AIX routines and the CREATE Statement" on page 166

Related tasks:

- "Replacing an AIX shared library" on page 167

AIX routines and the CREATE Statement

The following explains the relationship between compiling and linking your routine and the information you provide in the EXTERNAL NAME clause of the CREATE statement.

When you compile and link your program, you can identify external functions by using an export file specified with the `-bE:` option.

Suppose that the library `myrtns` contains three routines: `modify`, `remove`, and `add`. You identify `modify` as the default entry point, by putting it as the first entry in the export file that is linked in in the link step. The `remove` and `add` functions are indicated as additional exportable functions by also including them in an export file.

In the link step, you specify:

```
-bE:myrtns.exp
```

which identifies the export file `myrtns.exp`.

The export file looks like this:

```
modify
remove
add
```

Finally, your EXTERNAL NAME clauses for the routines, which are implemented with the `modify`, `remove`, and `add` functions, are coded as follows:

```
EXTERNAL NAME '/u/mydir/routines/myrtns!modify'
```

and

```
EXTERNAL NAME '/u/mydir/routines/myrtns!remove'
```

and

```
EXTERNAL NAME '/u/mydir/routines/myrtns!add'
```

Note: The default path used will be `sqllib/function`. This means if the EXTERNAL NAME clause is specified as follows:

```
EXTERNAL NAME 'myrtns!modify'
```

DB2® will attempt to load `myrtns` from `sqllib/function`.

Related concepts:

- "AIX export files for routines" on page 165

Related tasks:

- "Replacing an AIX shared library" on page 167

Replacing an AIX shared library

Procedure:

After a shared library is built, it is typically copied into a directory from which DB2 will access it. When attempting to replace a routine shared library, you should either run `/usr/sbin/slibclean` to flush the AIX shared library cache, or remove the library from the target directory and then copy the library from the source directory to the target directory. Otherwise, the copy operation might fail because AIX keeps a cache of referenced libraries and does not allow the library to be overwritten.

Related concepts:

- "AIX export files for routines" on page 165
- "AIX routines and the CREATE Statement" on page 166

Considerations for installing COBOL on AIX

Because of the way AIX® loads routines and resolves library references within them, there are requirements on how COBOL should be installed. These requirements become a factor when a COBOL program loads a shared library (routine) at run time.

When a routine is loaded, the chain of libraries it refers to must also be loaded. When AIX searches for a library only indirectly referenced by your program, it must use the path compiled into the library that referenced it when it was built by the language provider (IBM COBOL or Micro Focus COBOL). This path might very well not be the same path in which the compiler was installed. If the library in the chain cannot be found, the routine load will fail, and you will receive SQLCODE -444.

To ensure this does not happen, install the compiler wherever you want, then create symbolic links of all language libraries from the install directory into `/usr/lib` (a directory that is almost always searched when a library needs to be loaded). You could link the libraries into `sqllib/function` (the routine directory), but this only works for one database instance; `/usr/lib` works for everyone on the machine.

Related tasks:

- "Setting up the UNIX application development environment" on page 31
- "Configuring the IBM COBOL compiler on AIX" on page 181
- "Configuring the Micro Focus COBOL compiler on AIX" on page 187

IBM C

Building information for DB2 CLI applications and routines is in the *CLI Guide and Reference*.

For information on building C applications on supported UNIX operating systems, see "Building UNIX C applications" on page 147. For information on building C routines on supported UNIX operating systems, see "Building UNIX C routines" on page 151.

Build script for C applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds AIX C application programs
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set lib32 for 32-bit programs, lib for 64-bit,
# and set extra compile flag for 64-bit programs.
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $bitwidth = "\"32\"" ]; then
  LIB=lib32
  EXTRA_CFLAG=
else
  LIB=lib
  EXTRA_CFLAG=-q64
fi

# If an embedded SQL program, precompile and bind it.
# Note: some .sqc files contain no SQL but link in
# utilemb.sqc, so if you get this warning, ignore it:
# SQL0053W  No SQL statements were found in the program.
if [ -f $1".sqc" ]
then
  ./embprep $1 $2 $3 $4
  # Compile the utilemb.c error-checking utility.
  xlc $EXTRA_CFLAG -I$DB2PATH/include -c utilemb.c
else
  # Compile the utilapi.c error-checking utility.
  xlc $EXTRA_CFLAG -I$DB2PATH/include -c utilapi.c
fi

# Compile the program.
xlc $EXTRA_CFLAG -I$DB2PATH/include -c $1.c

if [ -f $1".sqc" ]
then
  # Link the program with utilemb.o
  xlc $EXTRA_CFLAG -o $1 $1.o utilemb.o -ldb2 -L$DB2PATH/$LIB
else
  # Link the program with utilapi.o
  xlc $EXTRA_CFLAG -o $1 $1.o utilapi.o -ldb2 -L$DB2PATH/$LIB
fi
```

AIX C application compile and link options

The following are the compile and link options recommended by DB2 for building C embedded SQL and DB2 API applications with the AIX IBM C compiler, as demonstrated in the `bldapp` build script.

Compile and Link Options for bldapp
Compile Options:
xlc The IBM C compiler.
$EXTRA_CFLAG Contains "-q64" for an instance where 64-bit support is enabled; otherwise, it contains no value.
-I$DB2PATH/include Specify the location of the DB2 include files. For example: `$HOME/sqllib/include`.
-c Perform compile only; no link. Compile and link are separate steps.
Link Options:
xlc Use the compiler as a front end for the linker.
$EXTRA_CFLAG Contains "-q64" for an instance where 64-bit support is enabled; otherwise, it contains no value.
-o $1 Specify the executable program.
$1.o Specify the program object file.
utilemb.o If an embedded SQL program, include the embedded SQL utility object file for error checking.
utilapi.o If not an embedded SQL program, include the DB2 API utility object file for error checking.
-ldb2 Link to the DB2 library.
-L$DB2PATH/$LIB Specify the location of the DB2 runtime shared libraries. For example: `$HOME/sqllib/$LIB`. If you do not specify the `-L` option, the compiler assumes the following path: `/usr/lib:/lib`.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX C applications" on page 147

Related reference:

- "AIX C routine compile and link options" on page 170

Related samples:

- "bldapp -- Builds AIX C application programs (C)"

Build script for C routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds AIX C routines (stored procedures and UDFs)
# Usage: bldrtn <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set lib32 for 32-bit programs, lib for 64-bit,
# and set extra compile flag for 64-bit programs.
```

```
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $bitwidth = "\"32\"" ]; then
  LIB=lib32
  EXTRA_CFLAG=
else
  LIB=lib
  EXTRA_CFLAG=-q64
fi

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqc" ]
then
  ./embprep $1 $2
fi

# Compile the program.
xlc_r $EXTRA_CFLAG -I$DB2PATH/include -c $1.c

# Link the program using the export file $1.exp,
xlc_r $EXTRA_CFLAG -qmkshrobj -o $1 $1.o -ldb2 -L$DB2PATH/$LIB -bE:$1.exp

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

AIX C routine compile and link options

The following are the compile and link options recommended by DB2 for building C routines (stored procedures and user-defined functions) with the AIX IBM C compiler, as demonstrated in the `bldrtn` build script.

Compile and link options for bldrtn	
Compile options:	
xlc_r	Use the multi-threaded version of the IBM C compiler, needed as the routines can run in the same process as other routines (THREADSAFE) or in the engine itself (NOT FENCED).
$EXTRA_CFLAG	Contains "-q64" for an instance where 64-bit support is enabled; otherwise, it contains no value.
-I$DB2PATH/include	Specify the location of the DB2 include files. For example: `$HOME/sqllib/include`.
-c	Perform compile only; no link. Compile and link are separate steps.

Compile and link options for bldrtn
Link options:
xlc_r Use the multi-threaded version of the compiler as a front end for the linker.
$EXTRA_CFLAG Contains "-q64" for an instance where 64-bit support is enabled; otherwise, it contains no value.
-qmkshrobj Create the shared library.
-o $1 Specify the output file name.
$1.o Specify the object file.
-ldb2 Link with the DB2 library.
-L$DB2PATH/$LIB Specify the location of the DB2 runtime shared libraries. For example: `$HOME/sqllib/$LIB`. If you do not specify the `-L` option, the compiler assumes the following path: `/usr/lib:/lib`.
-bE:$1.exp Specify an export file. The export file contains a list of the routines.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX C routines" on page 151

Related reference:

- "AIX C application compile and link options" on page 168

Related samples:

- "bldrtn -- Builds AIX C routines (stored procedures and UDFs) (C)"

Building C multi-threaded applications on AIX

C multi-threaded applications on AIX need to be compiled and linked with the `xlc_r` compiler instead of the `xlc` compiler or, for C++, with the `xlC_r` compiler instead of the `xlC` compiler. The `_r` versions set the appropriate preprocessor defines for multi-threaded compilation, and supply the appropriate threaded library names to the linker.

Additional information about compiler and link flag settings using the multi-threaded compiler front ends can be obtained from your compiler documentation.

The script file `bldmt`, in `sqllib/samples/c`, contains the commands to build an embedded SQL multi-threaded program. Besides the `xlc_r` compiler and the absence of a utility file linked in, the compile and link options are the same as those used for the embedded SQL script file, `bldapp`.

Procedure:

To build the multi-threaded sample program, `dbthrds`, from the source file `dbthrds.sqc`, enter:

```
bldmt dbthrds
```

The result is an executable file, dbthrds. To run the executable file against the sample database, enter the executable name:

```
dbthrds
```

Related concepts:

- "Build files" on page 97

Related tasks:

- "Building UNIX C applications" on page 147

Related reference:

- "AIX C application compile and link options" on page 168
- "C samples" on page 64

Related samples:

- "bldmt -- Builds AIX C multi-threaded applications (C)"
- "dbthrds.sqc -- How to use multiple context APIs on UNIX (C)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

VisualAge C++

For information on building C++ applications on supported UNIX operating systems, see "Building UNIX C++ applications" on page 154. For information on building C++ routines on supported UNIX operating systems, see "Building UNIX C++ routines" on page 158.

Build script for C++ applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds AIX C++ applications
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set lib32 for 32-bit programs, lib for 64-bit,
# and set extra compile flag for 64-bit programs.
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $bitwidth = "\"32\"" ]; then
  LIB=lib32
  EXTRA_CFLAG=
else
  LIB=lib
  EXTRA_CFLAG=-q64
fi

# If an embedded SQL program, precompile and bind it.
# Note: some .sqC files contain no SQL but link in
# utilemb.sqC, so if you get this warning, ignore it:
# SQL0053W  No SQL statements were found in the program.
if [ -f $1".sqC" ]
then
  ./embprep $1 $2 $3 $4
  # Compile the utilemb.C error-checking utility.
  xlC $EXTRA_CFLAG -qstaticinline -I$DB2PATH/include -c utilemb.C
else
```

```
  # Compile the utilapi.C error-checking utility.
  xlC $EXTRA_CFLAG -qstaticinline -I$DB2PATH/include -c utilapi.C
fi

# Compile the program.
xlC $EXTRA_CFLAG -qstaticinline -I$DB2PATH/include -c $1.C

if [ -f $1".sqC" ]
then
  # Link the program with utilemb.o
  xlC $EXTRA_CFLAG -o $1 $1.o utilemb.o -ldb2 -L$DB2PATH/$LIB
else
  # Link the program with utilapi.o
  xlC $EXTRA_CFLAG -o $1 $1.o utilapi.o -ldb2 -L$DB2PATH/$LIB
fi
```

AIX C++ application compile and link options

The following are the compile and link options recommended by DB2 for building C++ embedded SQL and DB2 API applications with the AIX IBM VisualAge C++ compiler, as demonstrated in the `bldapp` build script.

Compile and link options for bldapp
Compile options: **xlC** The VisualAge C++ compiler. **EXTRA_CFLAG** Contains "-q64" for an instance where 64-bit support is enabled; otherwise, it contains no value. **-qstaticinline** Gives non-inlined inline functions internal linkage so there are no warnings at link time if the function exists in multiple object files. **-I$DB2PATH/include** Specify the location of the DB2 include files. For example: `$HOME/sqllib/include`. **-c** Perform compile only; no link. Compile and link are separate steps.
Link options: **xlC** Use the compiler as a front end for the linker. **EXTRA_CFLAG** Contains "-q64" for an instance where 64-bit support is enabled; otherwise, it contains no value. **-o $1** Specify the executable program. **$1.o** Specify the program object file. **utilapi.o** Include the API utility object file for non-embedded SQL programs. **utilemb.o** Include the embedded SQL utility object file for embedded SQL programs. **-ldb2** Link with the DB2 library. **-L$DB2PATH/$LIB** Specify the location of the DB2 runtime shared libraries. For example: `$HOME/sqllib/$LIB`. If you do not specify the `-L` option, the compiler assumes the following path `/usr/lib:/lib`. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX C++ applications" on page 154
- "Building C++ embedded SQL applications with configuration files" on page 178
- "Building C++ DB2 API applications with configuration files" on page 177

Related reference:

- "AIX C++ routine compile and link options" on page 174

Related samples:

- "bldapp -- Builds AIX C++ applications (C++)"

Build script for C++ routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds AIX C++ routines (stored procedures and UDFs)
# Usage: bldrtn <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set lib32 for 32-bit programs, lib for 64-bit,
# and set extra compile flag for 64-bit programs.
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $bitwidth = "\"32\"" ]; then
  LIB=lib32
  EXTRA_CFLAG=
else
  LIB=lib
  EXTRA_CFLAG=-q64
fi

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqC" ]
then
  ./embprep $1 $2
fi

# Compile the program.
xlC_r $EXTRA_CFLAG -qstaticinline -I$DB2PATH/include -c $1.C

# Link using export file $1.exp, creating shared library $1
xlC_r $EXTRA_CFLAG -qmkshrobj -o $1 $1.o -L$DB2PATH/$LIB -ldb2 -bE:$1.exp

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

AIX C++ routine compile and link options

The following are the compile and link options recommended by DB2 for building C++ routines (stored procedures and user-defined functions) with the AIX VisualAge C++ compiler, as demonstrated in the `bldrtn` build script.

Compile and link options for bldrtn
Compile options: **xlC_r** The multi-threaded version of the IBM VisualAge C++ compiler, needed as the routines can run in the same process as other routines (THREADSAFE) or in the engine itself (NOT FENCED). **$EXTRA_CFLAG** Contains "-q64" for an instance where 64-bit support is enabled; otherwise, it contains no value. **-qstaticinline** Gives non-inlined inline functions internal linkage so there are no warnings at link time if the function exists in multiple object files. **-I$DB2PATH/include** Specify the location of the DB2 include files. For example: `$HOME/sqllib/include`. **-c** Perform compile only; no link. Compile and link are separate steps.
Link options: **xlC_r** Use the multi-threaded version of the compiler as a front-end for the linker. **$EXTRA_CFLAG** Contains "-q64" for an instance where 64-bit support is enabled; otherwise, it contains no value. **-qmkshrobj** Create a shared library. **-o $1** Specify the output as a shared library file. **$1.o** Specify the program object file. **-L$DB2PATH/$LIB** Specify the location of the DB2 runtime shared libraries. For example: `$HOME/sqllib/$LIB`. If you do not specify the `-L` option, the compiler assumes the following path: `/usr/lib:/lib`. **-ldb2** Link with the DB2 library. **-bE:$1.exp** Specify an export file. The export file contains a list of the routines. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX C routines" on page 151
- "Building C++ stored procedures with configuration files" on page 178
- "Building C++ user-defined functions with configuration files" on page 180

Related reference:

- "AIX C++ application compile and link options" on page 173

Related samples:

- "bldrtn -- Builds AIX C++ routines (stored procedures and UDFs) (C++)"

Building C++ multi-threaded applications on AIX

C++ multi-threaded applications on AIX need to be compiled and linked with the `xlC_r` compiler instead of the `xlC` compiler or, for C, with the `xlc_r` compiler

instead of the xlc compiler. The _r versions set the appropriate preprocessor defines for multi-threaded compilation and supply the appropriate threaded library names to the linker.

Additional information about compiler and link flag settings using the multi-threaded compiler front ends can be obtained from your compiler documentation.

The script, bldmt, contains the commands to build multi-threaded applications. Besides the xlC_r compiler, discussed above, and no utility file linked in, the compile and link options are the same as those used in the embedded SQL script file, bldapp.

Procedure:

To build the multi-threaded sample program, dbthrds, from the source file dbthrds.sqC, enter:

```
bldmt dbthrds
```

The result is an executable file, dbthrds. To run the executable file against the sample database, enter the executable name:

```
dbthrds
```

Related concepts:

- "Build files" on page 97

Related tasks:

- "Building UNIX C++ applications" on page 154

Related reference:

- "AIX C++ application compile and link options" on page 173
- "C samples" on page 64

Related samples:

- "bldmt -- Builds AIX C++ multi-threaded applications (C++)"
- "dbthrds.sqC -- How to use multiple context APIs on UNIX (C++)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

VisualAge C++ configuration files

Note: Building information for CLI applications and routines is in the *CLI Guide and Reference*.

Building VisualAge C++ programs with configuration files

VisualAge C++ Version 5.0 has both an incremental compiler and a batch mode compiler. While the batch mode compiler uses makefiles and build files, the incremental compiler uses configuration files instead. See the documentation that comes with VisualAge C++ Version 5.0 to learn more about this.

DB2 provides configuration files for the different types of DB2 programs you can build with the VisualAge C++ compiler.

Procedure:

To use a DB2 configuration file, you first set an environment variable to the program name you wish to compile. Then you compile the program with a command supplied by VisualAge C++. Here are the topics describing how you can use the configuration files provided by DB2 to compile different types of programs:

- Building C++ Embedded SQL Applications with Configuration Files
- Building C++ DB2 API Applications with Configuration Files
- Building C++ Stored Procedures with Configuration Files
- Building C++ User-defined Functions with Configuration Files

Related tasks:

- "Building C++ embedded SQL applications with configuration files" on page 178
- "Building C++ DB2 API applications with configuration files" on page 177
- "Building C++ stored procedures with configuration files" on page 178
- "Building C++ user-defined functions with configuration files" on page 180
- "Building UNIX C applications" on page 147
- "Building UNIX C routines" on page 151
- "Building UNIX C++ applications" on page 154
- "Building UNIX C++ routines" on page 158

Building C++ DB2 API applications with configuration files

The configuration file, `api.icc`, in `sqllib/samples/c` and in `sqllib/samples/cpp`, allows you to build DB2 API programs in C or C++ on AIX.

Procedure:

To use the configuration file to build the DB2 API sample program `cli_info` from the source file `cli_info.c`, do the following:

1. Set the API environment variable to the program name by entering:
   - For bash or Korn shell:
     ```
     export API=cli_info
     ```
   - For C shell:
     ```
     setenv API cli_info
     ```
2. If you have an `api.ics` file in your working directory, produced by building a different program with the `api.icc` file, delete the `api.ics` file with this command:
   ```
   rm api.ics
   ```
   An existing `api.ics` file produced for the same program you are going to build again does not have to be deleted.
3. Compile the sample program by entering:
   ```
   vacbld api.icc
   ```
   Note: The `vacbld` command is provided by VisualAge C++.

The result is an executable file, `cli_info`. You can run the program by entering the executable name:

```
cli_info
```

Related tasks:

- "Building C++ embedded SQL applications with configuration files" on page 178
- "Building C++ stored procedures with configuration files" on page 178
- "Building C++ user-defined functions with configuration files" on page 180

Building C++ embedded SQL applications with configuration files

The configuration file, `emb.icc`, in `sqllib/samples/c` and `sqllib/samples/cpp`, allows you to build DB2 embedded SQL applications in C and C++ on AIX.

Procedure:

To use the configuration file to build the embedded SQL application `tbmod` from the source file `tbmod.sqc`, do the following:

1. Set the EMB environment variable to the program name by entering:
   - For bash or Korn shell:
     ```
     export EMB=tbmod
     ```
   - For C shell:
     ```
     setenv EMB tbmod
     ```
2. If you have an `emb.ics` file in your working directory, produced by building a different program with the `emb.icc` file, delete the `emb.ics` file with this command:
   ```
   rm emb.ics
   ```
   An existing `emb.ics` file produced for the same program you are going to build again does not have to be deleted.
3. Compile the sample program by entering:
   ```
   vacbld emb.icc
   ```
   Note: The `vacbld` command is provided by VisualAge C++.

The result is an executable file, `tbmod`. You can run the program by entering the executable name:

```
tbmod
```

Related tasks:

- "Building C++ DB2 API applications with configuration files" on page 177
- "Building C++ stored procedures with configuration files" on page 178
- "Building C++ user-defined functions with configuration files" on page 180

Building C++ stored procedures with configuration files

The configuration file, `stp.icc`, in `sqllib/samples/c` and `sqllib/samples/cpp`, allows you to build DB2 embedded SQL stored procedures in C and C++ on AIX.

Procedure:

To use the configuration file to build the embedded SQL stored procedure shared library `spserver` from the source file `spserver.sqc`, do the following:

1. Set the STP environment variable to the program name by entering:

- For bash or Korn shell:

  ```
  export STP=spserver
  ```

- For C shell:

  ```
  setenv STP spserver
  ```

2. If you have an `stp.ics` file in your working directory, produced by building a different program with the `stp.icc` file, delete the `stp.ics` file with this command:

   ```
   rm stp.ics
   ```

   An existing `stp.ics` file produced for the same program you are going to build again does not have to be deleted.
3. Compile the sample program by entering:

   ```
   vacbld stp.icc
   ```

   Note: The `vacbld` command is provided by VisualAge C++.

The stored procedure shared library is copied to the server in the path `sqllib/function`.

Next, catalog the stored procedures in the shared library by running the `spcat` script on the server:

```
spcat
```

This script connects to the sample database, uncatalogs the stored procedures if they were previously cataloged by calling `spdrop.db2`, then catalogs them by calling `spcreate.db2`, and finally disconnects from the database. You can also call the `spdrop.db2` and `spcreate.db2` scripts individually.

Then, stop and restart the database to allow the new shared library to be recognized. If necessary, set the file mode for the shared library so the DB2 instance can access it.

Once you build the stored procedure shared library, `spserver`, you can build the client application, `spclient`, that calls the stored procedures in it. You can build `spclient` using the configuration file, `emb.icc`.

To call the stored procedures, run the sample client application by entering:

`spclient` *database userid password*

where

database
: Is the name of the database to which you want to connect. The name could be `sample`, or its remote alias, or some other name.

userid Is a valid user ID.

password
: Is a valid password.

The client application accesses the shared library, `spserver`, and executes a number of stored procedure functions on the server database. The output is returned to the client application.

Related tasks:

- "Building C++ embedded SQL applications with configuration files" on page 178
- "Building C++ DB2 API applications with configuration files" on page 177
- "Building C++ user-defined functions with configuration files" on page 180

Building C++ user-defined functions with configuration files

The configuration file, `udf.icc`, in `sqllib/samples/c` and `sqllib/samples/cpp`, allows you to build user-defined functions in C and C++ on AIX.

Procedure:

To use the configuration file to build the user-defined function program `udfsrv` from the source file `udfsrv.c`, do the following:

1. Set the UDF environment variable to the program name by entering:
   - For bash or Korn shell:
     ```
     export UDF=udfsrv
     ```
   - For C shell:
     ```
     setenv UDF udfsrv
     ```
2. If you have a `udf.ics` file in your working directory, produced by building a different program with the `udf.icc` file, delete the `udf.ics` file with this command:
   ```
   rm udf.ics
   ```
   An existing `udf.ics` file produced for the same program you are going to build again does not have to be deleted.
3. Compile the sample program by entering:
   ```
   vacbld udf.icc
   ```
   Note: The `vacbld` command is provided by VisualAge C++.

The UDF library is copied to the server in the path `sqllib/function`.

If necessary, set the file mode for the user-defined function so the DB2 instance can run it.

Once you build `udfsrv`, you can build the client application, `udfcli`, that calls it. DB2 CLI and embedded SQL versions of this program are provided.

You can build the DB2 CLI `udfcli` program from the source file `udfcli.c`, in `sqllib/samples/cli`, by using the configuration file `cli.icc`.

You can build the embedded SQL `udfcli` program from the source file `udfcli.sqc`, in `sqllib/samples/c`, by using the configuration file `emb.icc`.

To call the UDF, run the sample calling application by entering the executable name:

```
udfcli
```

The calling application calls the `ScalarUDF` function from the `udfsrv` library.

Related tasks:

- "Building C++ embedded SQL applications with configuration files" on page 178
- "Building C++ DB2 API applications with configuration files" on page 177

- "Building C++ stored procedures with configuration files" on page 178

IBM COBOL Set for AIX

Configuring the IBM COBOL compiler on AIX

The following are steps you need to take if you develop applications that contain embedded SQL and DB2 API calls, and you are using the IBM COBOL Set for AIX compiler.

Procedure:

- When you precompile your application using the command line processor command `db2 prep`, use the `target ibmcob` option.
- Do not use tab characters in your source files.
- You can use the `PROCESS` and `CBL` keywords in the first line of your source files to set compile options.
- If your application contains only embedded SQL, but no DB2 API calls, you do not need to use the `pgmname(mixed)` compile option. If you use DB2 API calls, you must use the `pgmname(mixed)` compile option.
- If you are using the ″System/390 host data type support″ feature of the IBM COBOL Set for AIX compiler, the DB2 include files for your applications are in the following directory:

  ```
  $HOME/sqllib/include/cobol_i
  ```

  If you are building DB2 sample programs using the script files provided, the include file path specified in the script files must be changed to point to the `cobol_i` directory and not the `cobol_a` directory.

  If you are NOT using the ″System/390 host data type support″ feature of the IBM COBOL Set for AIX compiler, or you are using an earlier version of this compiler, then the DB2 include files for your applications are in the following directory:

  ```
  $HOME/sqllib/include/cobol_a
  ```

  Specify COPY file names to include the `.cbl` extension as follows:

  ```
  COPY "sql.cbl".
  ```

Related concepts:

- "Considerations for installing COBOL on AIX" on page 167

Related tasks:

- "Setting up the UNIX application development environment" on page 31
- "Building IBM COBOL applications on AIX" on page 182
- "Building IBM COBOL routines on AIX" on page 184

Related reference:

- "AIX IBM COBOL application compile and link options" on page 183
- "AIX IBM COBOL routine compile and link options" on page 186

Building IBM COBOL applications on AIX

DB2 provides build scripts for compiling and linking IBM COBOL embedded SQL and DB2 API programs. These are located in the `sqllib/samples/cobol` directory, along with sample programs that can be built with these files.

The build file, `bldapp` contains the commands to build a DB2 application program.

The first parameter, `$1`, specifies the name of your source file. This is the only required parameter for programs that do not contain embedded SQL. Building embedded SQL programs requires a connection to the database so three optional parameters are also provided: the second parameter, `$2`, specifies the name of the database to which you want to connect; the third parameter, `$3`, specifies the user ID for the database, and `$4` specifies the password.

For an embedded SQL program, `bldapp` passes the parameters to the precompile and bind script, `embprep`. If no database name is supplied, the default `sample` database is used. The user ID and password parameters are only needed if the instance where the program is built is different from the instance where the database is located.

Procedure:

To build the non-embedded SQL sample program `client` from the source file `client.cbl`, enter:

```
bldapp client
```

The result is an executable file `client`. You can run the executable file against the `sample` database by entering:

```
client
```

Building and running embedded SQL applications

There are three ways to build the embedded SQL application, `updat`, from the source file `updat.sqb`:

1. If connecting to the sample database on the same instance, enter:
   ```
   bldapp updat
   ```
2. If connecting to another database on the same instance, also enter the database name:
   ```
   bldapp updat database
   ```
3. If connecting to a database on another instance, also enter the user ID and password of the database instance:
   ```
   bldapp updat database userid password
   ```

The result is an executable file, `updat`.

There are three ways to run this embedded SQL application:

1. If accessing the `sample` database on the same instance, simply enter the executable name:
   ```
   updat
   ```
2. If accessing another database on the same instance, enter the executable name and the database name:
   ```
   updat database
   ```

3. If accessing a database on another instance, enter the executable name, database name, and user ID and password of the database instance:

   ```
   updat database userid password
   ```

Related concepts:

- "Build files" on page 97

Related tasks:

- "Building IBM COBOL routines on AIX" on page 184

Related reference:

- "AIX IBM COBOL application compile and link options" on page 183
- "COBOL samples" on page 73

Related samples:

- "bldapp -- Builds AIX COBOL applications"
- "client.cbl -- How to set and query a client (IBM COBOL)"
- "embprep -- To prep and bind a COBOL embedded SQL sample on AIX"
- "updat.sqb -- How to update, delete and insert table data (IBM COBOL)"

Build script for IBM COBOL applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds AIX COBOL applications
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqb" ]
then
  ./embprep $1 $2 $3 $4
fi

# Compile the checkerr.cbl error checking utility.
cob2 -qpgmname\(mixed\) -qlib -I$DB2PATH/include/cobol_a \
     -c checkerr.cbl

# Compile the program.
cob2 -qpgmname\(mixed\) -qlib -I$DB2PATH/include/cobol_a \
     -c $1.cbl

# Link the program.
cob2 -o $1 $1.o checkerr.o -L$DB2PATH/lib -ldb2
```

AIX IBM COBOL application compile and link options

The following are the compile and link options recommended by DB2 for building COBOL embedded SQL and DB2 API applications with the IBM AIX COBOL Set compiler, as demonstrated in the `bldapp` build script.

Compile and link options for bldapp	
Compile options:	
cob2	The IBM COBOL Set compiler.
-qpgmname\(mixed\)	Instructs the compiler to permit CALLs to library entry points with mixed-case names.
-qlib	Instructs the compiler to process COPY statements.
-I$DB2PATH/include/cobol_a	Specify the location of the DB2 include files. For example: `$HOME/sqllib/include/cobol_a`.
-c	Perform compile only; no link. Compile and link are separate steps.
Link options:	
cob2	Use the compiler as a front end for the linker.
-o $1	Specify the executable program.
$1.o	Specify the program object file.
checkerr.o	Include the utility object file for error-checking.
-L$DB2PATH/lib	Specify the location of the DB2 runtime shared libraries. For example: `$HOME/sqllib/lib`. If you do not specify the `-L` option, the compiler assumes the following path: `/usr/lib:/lib`.
-ldb2	Link with the database manager library.
Refer to your compiler documentation for additional compiler options.	

Related tasks:

- "Building IBM COBOL applications on AIX" on page 182

Related reference:

- "AIX IBM COBOL routine compile and link options" on page 186

Related samples:

- "bldapp -- Builds AIX COBOL applications"

Building IBM COBOL routines on AIX

DB2 provides build scripts for compiling and linking COBOL embedded SQL and DB2 API programs. These are located in the `sqllib/samples/cobol` directory, along with sample programs that can be built with these files.

The script, `bldrtn`, in `sqllib/samples/cobol`, contains the commands to build routines (stored procedures). The script compiles the routines into a shared library that can be called by a client application.

The first parameter, `$1`, specifies the name of your source file. The second parameter, `$2`, specifies the name of the database to which you want to connect. Since the shared library must be built on the same instance where the database resides, there are no parameters for user ID and password.

Only the first parameter, source file name, is required. The script uses the source file name, $1, for the shared library name. Database name is optional. If no database name is supplied, the program uses the default `sample` database.

Procedure:

To build the sample program `outsrv` from the source file `outsrv.sqb`, connecting to the sample database, enter:

```
bldrtn outsrv
```

If connecting to another database, also include the database name:

```
bldrtn outsrv database
```

The script file copies the shared library to the server in the path `sqllib/function`.

Once you build the routine shared library, `outsrv`, you can build the client application, `outcli`, that calls the routine within the library. You can build `outcli` using the script file `bldapp`.

To call the routine, run the sample client application by entering:

```
outcli database userid password
```

where

database
Is the name of the database to which you want to connect. The name could be `sample`, or its remote alias, or some other name.

userid Is a valid user ID.

password
Is a valid password for the user ID.

The client application accesses the shared library, `outsrv`, and executes the routine of the same name on the server database, and then returns the output to the client application.

Related concepts:

- "Build files" on page 97

Related tasks:

- "Building IBM COBOL applications on AIX" on page 182

Related reference:

- "AIX IBM COBOL routine compile and link options" on page 186
- "COBOL samples" on page 73

Related samples:

- "bldrtn -- Builds AIX COBOL routines (stored procedures)"
- "embprep -- To prep and bind a COBOL embedded SQL sample on AIX"
- "outcli.sqb -- Call stored procedures using the SQLDA structure (IBM COBOL)"
- "outsrv.sqb -- Demonstrates stored procedures using the SQLDA structure (IBM COBOL)"

Build script for IBM COBOL routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds AIX COBOL routines (stored procedures)
# Usage:  bldrtn <program_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Precompile and bind the program.
./embprep $1 $2

# Compile the checkerr.cbl error checking utility.
cob2 -qpgmname\(mixed\) -qlib -I$DB2PATH/include/cobol_a \
     -c checkerr.cbl

# Compile the program.
cob2 -qpgmname\(mixed\) -qlib -c -I$DB2PATH/include/cobol_a $1.cbl

# Link the program creating shared library $1 with export file $1.exp
cob2 -o $1 $1.o checkerr.o -bnoentry -bE:$1.exp \
     -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function directory of the DB2 instance.
# This assumes the user has write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

AIX IBM COBOL routine compile and link options

The following are the compile and link options recommended by DB2 for building COBOL routines (stored procedures) with the IBM COBOL Set compiler on AIX, as demonstrated in the `bldrtn` build script.

Compile and Link Options for bldrtn
Compile Options:
cob2 The IBM COBOL Set compiler.
-qpgmname\(mixed\) Instructs the compiler to permit CALLs to library entry points with mixed-case names.
-qlib Instructs the compiler to process COPY statements.
-c Perform compile only; no link. Compile and link are separate steps.
-I$DB2PATH/include/cobol_a Specify the location of the DB2 include files. For example: `$HOME/sqllib/include/cobol_a`.

Compile and Link Options for bldrtn	
Link Options:	
cob2	Use the compiler to link edit.
-o $1	Specify the output as a shared library file.
$1.o	Specify the stored procedure object file.
checkerr.o	Include the utility object file for error-checking.
-bnoentry	Do not specify the default entry point to the shared library.
-bE:$1.exp	Specify an export file. The export file contains a list of the stored procedures.
-L$DB2PATH/lib	Specify the location of the DB2 runtime shared libraries. For example: `$HOME/sqllib/lib`. If you do not specify the `-L` option, the compiler assumes the following path: `/usr/lib:/lib`.
-ldb2	Link with the database manager library.
Refer to your compiler documentation for additional compiler options.	

Related tasks:

- "Building IBM COBOL routines on AIX" on page 184

Related reference:

- "AIX IBM COBOL application compile and link options" on page 183

Related samples:

- "bldrtn -- Builds AIX COBOL routines (stored procedures)"

Micro Focus COBOL

For information on building Micro Focus COBOL applications on supported UNIX operating systems, see "Building UNIX Micro Focus COBOL applications" on page 161. For information on building Micro Focus COBOL routines on supported UNIX operating systems, see "Building UNIX Micro Focus COBOL routines" on page 162.

Configuring the Micro Focus COBOL compiler on AIX

Do the following if you develop applications that contain embedded SQL and DB2 API calls with the Micro Focus COBOL compiler.

Procedure:

- When you precompile your application using the command line processor command `db2 prep`, use the `target mfcob` option.
- You must include the DB2 COBOL COPY file directory in the Micro Focus COBOL environment variable COBCPY. The COBCPY environment variable specifies the location of the COPY files. The DB2 COPY files for Micro Focus COBOL reside in `sqllib/include/cobol_mf` under the database instance directory.

  To include the directory, enter:

– On bash or Korn shell:

```
export COBCPY=$COBCPY:$HOME/sqllib/include/cobol_mf
```

– On C shell:

```
setenv COBCPY $COBCPY:$HOME/sqllib/include/cobol_mf
```

Note: You might want to set COBCPY in the `.profile` or `.login` file.

Related concepts:

- "Considerations for installing COBOL on AIX" on page 167

Related tasks:

- "Setting up the UNIX application development environment" on page 31
- "Building UNIX Micro Focus COBOL applications" on page 161
- "Building UNIX Micro Focus COBOL routines" on page 162

Related reference:

- "AIX Micro Focus COBOL application compile and link options" on page 188
- "AIX Micro Focus COBOL routine compile and link options" on page 190

Build script for Micro Focus COBOL applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds AIX Micro Focus COBOL applications
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set COBCPY to include the DB2 COPY files directory.
COBCPY=$COBCPY:$DB2PATH/include/cobol_mf

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqb" ]
then
  ./embprep $1 $2 $3 $4
fi

# Compile the checkerr.cbl error checking utility.
cob -c -x checkerr.cbl

# Compile the program.
cob -c -x $1.cbl

# Link the program.
cob -x -o $1 $1.o checkerr.o -L$DB2PATH/lib -ldb2 -ldb2gmf
```

AIX Micro Focus COBOL application compile and link options

The following are the compile and link options recommended by DB2 for building COBOL embedded SQL and DB2 API applications with the Micro Focus COBOL compiler on AIX, as demonstrated in the `bldapp` build script.

Compile and link options for bldapp	
Compile options:	
cob	The COBOL compiler.
-c	Perform compile only; no link.
-x	When used with -c, produces an object file.
Link Options:	
cob	Use the compiler as a front end for the linker.
-x	Produces an executable program.
-o $1	Specify the executable program.
$1.o	Specify the program object file.
-L$DB2PATH/lib	Specify the location of the DB2 runtime shared libraries. For example: $HOME/sqllib/lib. If you do not specify the -L option, the compiler assumes the following path: /usr/lib:/lib.
-ldb2	Link to the DB2 library.
-ldb2gmf	Link to the DB2 exception-handler library for Micro Focus COBOL.
Refer to your compiler documentation for additional compiler options.	

Related tasks:

- "Building UNIX Micro Focus COBOL applications" on page 161

Related reference:

- "AIX Micro Focus COBOL routine compile and link options" on page 190

Related samples:

- "bldapp -- Builds AIX Micro Focus COBOL applications"

Build script for Micro Focus COBOL routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds AIX Micro Focus COBOL routines (stored procedures)
# Usage: bldrtn <program_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set COBCPY to include the DB2 COPY files directory.
COBCPY=$COBCPY:$DB2PATH/include/cobol_mf

# Precompile and bind the program.
./embprep $1 $2

# Compile the program.
cob -c -x $1.cbl

# Link the program.
cob -x -o $1 $1.o -Q -bnoentry \
  -Q -bI:$DB2PATH/lib/db2g.imp -L$DB2PATH/lib -ldb2 -ldb2gmf
```

```
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

AIX Micro Focus COBOL routine compile and link options

The following are the compile and link options recommended by DB2 for building COBOL routines (stored procedures) with the Micro Focus COBOL compiler on AIX, as demonstrated in the `bldrtn` build script.

Compile and link options for bldrtn	
Compile options:	
cob	The COBOL compiler.
-c	Perform compile only; no link. Compile and link are separate steps.
-x	Compile to an object module when used with the `-c` option.
Link options:	
cob	Use the compiler as a front-end for the linker.
-x	Produce a shared library.
-o $1	Specify the executable program.
$1.o	Specify the program object file.
-Q -bnoentry	Do not specify the default entry point to the shared library.
-Q -bI:$DB2PATH/lib/db2g.imp	Provides a list of entry points to the DB2 application library.
-L$DB2PATH/lib	Specify the location of the DB2 runtime shared libraries. For example: `$HOME/sqllib/lib`. If you do not specify the `-L` option, the compiler assumes the following path: `/usr/lib:/lib`.
-ldb2	Link to the DB2 library.
-ldb2gmf	Link to the DB2 exception-handler library for Micro Focus COBOL.
Refer to your compiler documentation for additional compiler options.	

Related tasks:

- "Building UNIX Micro Focus COBOL routines" on page 162

Related reference:

- "AIX Micro Focus COBOL application compile and link options" on page 188

Related samples:

- "bldrtn -- Builds AIX Micro Focus COBOL routines (stored procedures)"

REXX

Building REXX applications on AIX

The following shows you how to build REXX applications on AIX. DB2 for AIX supports classic REXX as well as Object REXX. Object REXX is an object-oriented version of the REXX language. Object-oriented extensions have been added to classic REXX, but its existing functions and instructions have not changed. The Object REXX interpreter is an enhanced version of its predecessor, with additional support for:

- Classes, objects, and methods
- Messaging and polymorphism
- Single and multiple inheritance

Object REXX is fully compatible with classic REXX. In this section, whenever REXX is used, all versions of REXX are inferred, including Object REXX.

You do not precompile or bind REXX programs.

Procedure:

To run DB2 REXX/SQL programs on AIX, you must set the `LIBPATH` environment variable to include `lib` under the DB2 install directory.

For bash or Korn shell, enter:

```
export LIBPATH=$LIBPATH:/lib:/usr/lib:/usr/opt/db2_08_01/lib
```

For C shell, enter:

```
setenv LIBPATH $LIBPATH:/lib:/usr/lib:/usr/opt/db2_08_01/lib
```

On AIX, your application file can have any file extension. You can run your application using either of the following two methods:

1. At the shell command prompt, enter `rexx` *name* where *name* is the name of your REXX program (including an extension, if one exists).
2. If the first line of your REXX program contains a "magic number", (#!), and identifies the directory where the REXX/6000 interpreter resides, you can run your REXX program by entering its name at the shell command prompt. For example, if the REXX/6000 interpreter file is in the `/usr/bin` directory, include the following as the very first line of your REXX program:

   ```
   #! /usr/bin/rexx
   ```

   Then, make the program executable by entering the following command at the shell command prompt:

   ```
   chmod +x name
   ```

   Run your REXX program by entering its file name at the shell command prompt.

REXX sample programs are in the directory `sqllib/samples/rexx`. To run the sample REXX program `updat.cmd`, enter:

```
updat.cmd
```

Related tasks:

- "Setting up the UNIX application development environment" on page 31

Related reference:
- "REXX samples" on page 88

Chapter 11. HP-UX

This chapter provides detailed information for building DB2 applications on HP-UX. For the latest DB2 application development updates for HP-UX, visit the DB2 application development Web page at:

`http://www.ibm.com/software/data/db2/udb/ad`

HP-UX C

Building information for DB2 CLI Applications and routines is in the *CLI Guide and Reference*.

For information on building C applications on supported UNIX operating systems, see "Building UNIX C applications" on page 147. For information on building C routines on supported UNIX operating systems, see "Building UNIX C routines" on page 151.

Build script for C applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds HP-UX C applications
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Determine the HP platform and set correct compile/link options
hpplat=`uname -m`
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $hpplat = "ia64" ]; then
    if [ $bitwidth = "\"64\"" ]; then
      EXTRA_CFLAG="+DD64"
      LIB="lib"
    else
      EXTRA_CFLAG="+DD32"
      LIB="lib32"
    fi
else
    if [ $bitwidth = "\"64\"" ]; then
      EXTRA_CFLAG="+DA2.0W"
      LIB="lib"
    else
      EXTRA_CFLAG=
```

```
      LIB="lib32"
    fi
fi

# The runtime path is recommended for all applications.
# If you need to use SHLIB_PATH or LD_LIBRARY_PATH, unset
# the RUNTIME variable by commenting out the following line.
RUNTIME=true

if [ "$RUNTIME" != "" ]
then
  EXTRA_LFLAG="-Wl,+b$DB2PATH/$LIB"
else
  EXTRA_LFLAG=""
fi

# If an embedded SQL program, precompile and bind it.
# Note: some .sqc files contain no SQL but link in
# utilemb.sqc, so if you get this warning, ignore it:
# SQL0053W  No SQL statements were found in the program.
if [ -f $1".sqc" ]
then
  ./embprep $1 $2 $3 $4
  # Compile the utilemb.c error-checking utility.
  cc  $EXTRA_CFLAG  -Ae -I$DB2PATH/include -c utilemb.c
else
  # Compile the utilapi.c error-checking utility.
  cc  $EXTRA_CFLAG  -Ae -I$DB2PATH/include -c utilapi.c
fi

# Compile the program.
cc  $EXTRA_CFLAG  -Ae -I$DB2PATH/include -c $1.c

if [ -f $1".sqc" ]
then
  # Link the program with utilemb.o.
  cc  $EXTRA_CFLAG  -o $1 $1.o utilemb.o $EXTRA_LFLAG -L$DB2PATH/$LIB -ldb2
else
  # Link the program with utilapi.o.
  cc  $EXTRA_CFLAG  -o $1 $1.o utilapi.o $EXTRA_LFLAG -L$DB2PATH/$LIB -ldb2
fi
```

HP-UX C application compile and link options

The following are the compile and link options recommended by DB2 for building C embedded SQL and DB2 API applications with the HP-UX C compiler, as demonstrated in the `bldapp` build script.

Compile and link options for bldapp
Compile options: **cc** The C compiler. **$EXTRA_CFLAG** If the HP-UX platform is IA64 and 64-bit support is enabled, this flag contains the value **+DD64**; if 32-bit support is enabled, it contains the value **+DD32**. If the HP-UX platform is PA-RISC and 64-bit support is enabled, it contains the value **+DA2.0W**. For 32-bit support on a PA-RISC platform, this flag contains no value. **+DD64** Must be used to generate 64-bit code for HP-UX on IA64. **+DD32** Must be used to generate 32-bit code for HP-UX on IA64. **+DA2.0W** Must be used to generate 64-bit code for HP-UX on PA-RISC. **-Ae** Enables HP ANSI extended mode. **-I$DB2PATH/include** Specifies the location of the DB2 include files. **-c** Perform compile only; no link. Compile and link are separate steps.
Link options: **cc** Use the compiler as a front end to the linker. **$EXTRA_CFLAG** If the HP-UX platform is IA64 and 64-bit support is enabled, this flag contains the value **+DD64**; if 32-bit support is enabled, it contains the value **+DD32**. If the HP-UX platform is PA-RISC and 64-bit support is enabled, it contains the value **+DA2.0W**. For 32-bit support on a PA-RISC platform, this flag contains no value. **+DD64** Must be used to generate 64-bit code for HP-UX on IA64. **+DD32** Must be used to generate 32-bit code for HP-UX on IA64. **+DA2.0W** Must be used to generate 64-bit code for HP-UX on PA-RISC. **-o $1** Specify the executable. **$1.o** Specify the program object file. **utilemb.o** If an embedded SQL program, include the embedded SQL utility object file for error checking. **utilapi.o** If a non-embedded SQL program, include the DB2 API utility object file for error checking. **$EXTRA_LFLAG** Specify the runtime path. If set, for 32-bit it contains the value `-Wl,+b$HOME/sqllib/lib32`, and for 64-bit: `-Wl,+b$HOME/sqllib/lib`. If not set, it contains no value. **-L$DB2PATH/$LIB** Specify the location of the DB2 runtime shared libraries. For 32-bit: `$HOME/sqllib/lib32`; for 64-bit: `$HOME/sqllib/lib`. **-ldb2** Link with the DB2 library. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX C applications" on page 147

Related samples:

- "bldapp -- Builds HP-UX C applications (C)"

Build script for C routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds HP-UX C routines (stored procedures and UDFs)
# Usage: bldrtn <prog_name> [ <db_name> ]
```

```
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Determine the HP platform and set correct compile/link options
hpplat=`uname -m`
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $hpplat = "ia64" ]; then
    if [ $bitwidth = "\"64\"" ]; then
      EXTRA_CFLAG="+DD64"
      LIB="lib"
    else
      EXTRA_CFLAG="+DD32"
      LIB="lib32"
    fi
else
    if [ $bitwidth = "\"64\"" ]; then
      EXTRA_CFLAG="+DA2.0W"
      LIB="lib"
    else
      EXTRA_CFLAG=
      LIB="lib32"
    fi
fi

# The runtime path is recommended for all applications.
# If you need to use SHLIB_PATH or LD_LIBRARY_PATH, unset
# the RUNTIME variable by commenting out the following line.
RUNTIME=true

if [ "$RUNTIME" != "" ]
then
  EXTRA_LFLAG="+b$DB2PATH/$LIB"
else
  EXTRA_LFLAG=""
fi

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqc" ]
then
  ./embprep $1 $2
fi

# Compile the program.
cc $EXTRA_CFLAG  +u1 +z -Ae -I$DB2PATH/include \
  -D_POSIX_C_SOURCE=199506L -c $1.c

# Link the program to create a shared library
ld -b -o $1 $1.o $EXTRA_LFLAG -L$DB2PATH/$LIB -ldb2 -lpthread

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

HP-UX C routine compile and link options

The following are the compile and link options recommended by DB2 for building C routines (stored procedures and user-defined functions) with the HP-UX C compiler, as demonstrated in the `bldrtn` build script.

Compile and link options for bldrtn
Compile options: **cc** The C compiler. **$EXTRA_CFLAG** If the HP-UX platform is IA64 and 64-bit support is enabled, this flag contains the value **+DD64**; if 32-bit support is enabled, it contains the value **+DD32**. If the HP-UX platform is PA-RISC and 64-bit support is enabled, it contains the value **+DA2.0W**. For 32-bit support on a PA-RISC platform, this flag contains no value. **+DD64** Must be used to generate 64-bit code for HP-UX on IA64. **+DD32** Must be used to generate 32-bit code for HP-UX on IA64. **+DA2.0W** Must be used to generate 64-bit code for HP-UX on PA-RISC. **+u1** Allow unaligned data access. Use only if your application uses unaligned data. **+z** Generate position-independent code. **-Ae** Enables HP ANSI extended mode. **-I$DB2PATH/include** Specify the location of the DB2 include files. For example: `-I$DB2PATH/include`. **-D_POSIX_C_SOURCE=199506L** POSIX thread library option that ensures _REENTRANT is defined, needed as the routines can run in the same process as other routines (THREADSAFE) or in the engine itself (NOT FENCED). **-c** Perform compile only; no link. Compile and link are separate steps.
Link options: **ld** Use the linker to link. **-b** Create a shared library rather than a normal executable. **-o $1** Specify the output as a shared library file. **$1.o** Specify the program object file. **$EXTRA_LFLAG** Specify the runtime path. If set, for 32-bit it contains the value `+b$HOME/sqllib/lib32`, and for 64-bit: `+b$HOME/sqllib/lib`. If not set, it contains no value. **-L$DB2PATH/$LIB** Specify the location of the DB2 runtime shared libraries. For 32-bit: `$HOME/sqllib/lib32`; for 64-bit: `$HOME/sqllib/lib`. **-ldb2** Link with the DB2 library. **-lpthread** Link with the POSIX thread library. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX C routines" on page 151

Related samples:

- "bldrtn -- Builds HP-UX C routines (stored procedures and UDFs) (C)"

Building C multi-threaded applications on HP-UX

HP-UX provides a POSIX thread library and a DCE thread library. Only multi-threaded applications using the POSIX thread library are supported by DB2.

Multi-threaded applications on HP-UX need to have _REENTRANT defined for their compilation. The HP-UX documentation recommends compiling with -D_POSIX_C_SOURCE=199506L. This will also ensure _REENTRANT is defined. Applications also need to be linked with `-lpthread`.

The script file, bldmt contains the commands to build multi-threaded applications. Besides the options specified above, the compile and link options are the same as those used in the embedded SQL script file, bldapp.

Procedure:

To build the sample program, dbthrds, from the source file dbthrds.sqc, enter:

```
bldmt dbthrds
```

The result is an executable file, dbthrds. To run the executable file against the sample database, enter the executable name:

```
dbthrds
```

Related concepts:

- "Build files" on page 97

Related reference:

- "C samples" on page 64
- "HP-UX C application compile and link options" on page 194

Related samples:

- "bldmt -- Builds HP-UX C multi-threaded applications (C)"
- "dbthrds.sqc -- How to use multiple context APIs on UNIX (C)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

HP-UX C++

For information on building C++ applications on supported UNIX operating systems, see "Building UNIX C++ applications" on page 154. For information on building C++ routines on supported UNIX operating systems, see "Building UNIX C++ routines" on page 158.

Build script for C++ applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds HP-UX C++ applications
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Determine the HP platform and set correct compile/link options
hpplat=`uname -m`
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $hpplat = "ia64" ]; then
    if [ $bitwidth = "\"64\"" ]; then
      EXTRA_CFLAG="+DD64 -AA"
      LIB="lib"
    else
      EXTRA_CFLAG="+DD32 -AA"
      LIB="lib32"
    fi
else
    if [ $bitwidth = "\"64\"" ]; then
      EXTRA_CFLAG="+DA2.0W"
```

```
      LIB="lib"
    else
      EXTRA_CFLAG=
      LIB="lib32"
    fi
fi

# The runtime path is recommended for all applications.
# If you need to use SHLIB_PATH or LD_LIBRARY_PATH, unset
# the RUNTIME variable by commenting out the following line.
RUNTIME=true

if [ "$RUNTIME" != "" ]
then
  EXTRA_LFLAG="-Wl,+b$DB2PATH/$LIB"
else
  EXTRA_LFLAG=""
fi

# If an embedded SQL program, precompile and bind it.
# Note: some .sqC files contain no SQL but link in
# utilemb.sqC, so if you get this warning, ignore it:
# SQL0053W  No SQL statements were found in the program.
if [ -f $1".sqC" ]
then
  ./embprep $1 $2 $3 $4
  # Compile the utilemb.C error-checking utility.
  aCC $EXTRA_CFLAG -ext -I$DB2PATH/include -c utilemb.C
else
  # Compile the utilapi.C error-checking utility.
  aCC $EXTRA_CFLAG -ext -I$DB2PATH/include -c utilapi.C
fi

# Compile the program.
aCC $EXTRA_CFLAG -ext -I$DB2PATH/include -c $1.C

if [ -f $1".sqC" ]
then
  # Link the program with utilemb.o.
  aCC $EXTRA_CFLAG -o $1 $1.o utilemb.o $EXTRA_LFLAG -L$DB2PATH/$LIB -ldb2
else
  # Link the program with utilapi.o.
  aCC $EXTRA_CFLAG -o $1 $1.o utilapi.o $EXTRA_LFLAG -L$DB2PATH/$LIB -ldb2
fi
```

HP-UX C++ application compile and link options

The following are the compile and link options recommended by DB2 for building C++ embedded SQL and DB2 API applications with the HP-UX C++ compiler, as demonstrated in the `bldapp` build script.

Compile and link options for bldapp
Compile options: **aCC** The HP aC++ compiler. **$EXTRA_CFLAG** If the HP-UX platform is IA64 and 64-bit support is enabled, this flag contains the value **+DD64 -AA**; if 32-bit support is enabled, it contains the value **+DD32 -AA**. If the HP-UX platform is PA-RISC and 64-bit support is enabled, it contains the value **+DA2.0W**. For 32-bit support on a PA-RISC platform, this flag contains no value. **+DD64** Must be used to generate 64-bit code for HP-UX on IA64. **+DD32** Must be used to generate 32-bit code for HP-UX on IA64. **-AA** Allows ANSI C++ standard features such as namespace std and the C++ standard library on IA64. **+DA2.0W** Must be used to generate 64-bit code for HP-UX on PA-RISC. **-ext** Allows various C++ extensions including "long long" support. **-I$DB2PATH/include** Specifies the location of the DB2 include files. For example: `$HOME/sqllib/include` **-c** Perform compile only; no link. Compile and link are separate steps.
Link options: **aCC** Use the HP aC++ compiler as a front end for the linker. **$EXTRA_CFLAG** If the HP-UX platform is IA64 and 64-bit support is enabled, this flag contains the value **+DD64 -AA**; if 32-bit support is enabled, it contains the value **+DD32 -AA**. If the HP-UX platform is PA-RISC and 64-bit support is enabled, it contains the value **+DA2.0W**. For 32-bit support on a PA-RISC platform, this flag contains no value. **+DD64** Must be used to generate 64-bit code for HP-UX on IA64. **+DD32** Must be used to generate 32-bit code for HP-UX on IA64. **-AA** Allows ANSI C++ standard features such as namespace std and the C++ standard library on IA64. **+DA2.0W** Must be used to generate 64-bit code for HP-UX on PA-RISC. **-o $1** Specify the executable. **$1.o** Specify the program object file. **utilemb.o** If an embedded SQL program, include the embedded SQL utility object file for error checking. **utilapi.o** If a non-embedded SQL program, include the DB2 API utility object file for error checking. **$EXTRA_LFLAG** Specify the runtime path. If set, for 32-bit it contains the value `-Wl,+b$HOME/sqllib/lib32`, and for 64-bit: `-Wl,+b$HOME/sqllib/lib`. If not set, it contains no value. **-L$DB2PATH/$LIB** Specify the location of the DB2 runtime shared libraries. For 32-bit: `$HOME/sqllib/lib32`; for 64-bit: `$HOME/sqllib/lib`. **-ldb2** Link with the DB2 library. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX C++ applications" on page 154

Related samples:

- "bldapp -- Builds HP-UX C++ applications (C++)"

Build script for C++ routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds HP-UX C++ routines (stored procedures and UDFs)
# Usage: bldrtn <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Determine the HP platform and set correct compile/link options
hpplat=`uname -m`
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $hpplat = "ia64" ]; then
    if [ $bitwidth = "\"64\"" ]; then
      EXTRA_CFLAG="+DD64 -AA"
      LIB="lib"
    else
      EXTRA_CFLAG="+DD32 -AA"
      LIB="lib32"
    fi
else
    if [ $bitwidth = "\"64\"" ]; then
      EXTRA_CFLAG="+DA2.0W"
      LIB="lib"
    else
      EXTRA_CFLAG=
      LIB="lib32"
    fi
fi

# The runtime path is recommended for all applications.
# If you need to use SHLIB_PATH or LD_LIBRARY_PATH, unset
# the RUNTIME variable by commenting out the following line.
RUNTIME=true

if [ "$RUNTIME" != "" ]
then
  EXTRA_LFLAG="-Wl,+b$DB2PATH/$LIB"
else
  EXTRA_LFLAG=""
fi

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqC" ]
then
  ./embprep $1 $2
fi

# Compile the program. First ensure it is coded with extern "C".
aCC  $EXTRA_CFLAG +u1 +z -ext -mt -I$DB2PATH/include -c $1.C

# Link the program to create a shared library.
aCC  $EXTRA_CFLAG -mt -b -o $1 $1.o $EXTRA_LFLAG -L$DB2PATH/$LIB -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

HP-UX C++ routine compile and link options

The following are the compile and link options recommended by DB2 for building C++ routines (stored procedures and user-defined functions) with the HP-UX C++ compiler, as demonstrated in the `bldrtn` build script.

Compile and link options for bldrtn
Compile options: **aCC** The HP aC++ compiler. **$EXTRA_CFLAG** If the HP-UX platform is IA64 and 64-bit support is enabled, this flag contains the value **+DD64 -AA**; if 32-bit support is enabled, it contains the value **+DD32 -AA**. If the HP-UX platform is PA-RISC and 64-bit support is enabled, it contains the value **+DA2.0W**. For 32-bit support on a PA-RISC platform, this flag contains no value. **+DD64** Must be used to generate 64-bit code for HP-UX on IA64. **+DD32** Must be used to generate 32-bit code for HP-UX on IA64. **-AA** Allows ANSI C++ standard features such as namespace std and the C++ standard library on IA64. **+DA2.0W** Must be used to generate 64-bit code for HP-UX on PA-RISC. **+u1** Allows unaligned data access. **+z** Generate position-independent code. **-ext** Allow various C++ extensions including ″long long″ support. **-mt** Allows threads support for the HP aC++ compiler, needed as the routines can run in the same process as other routines (THREADSAFE) or in the engine itself (NOT FENCED). **-I$DB2PATH/include** Specify the location of the DB2 include files. For example: `$DB2PATH/include` **-c** Perform compile only; no link. Compile and link are separate steps.
Link options: **aCC** Use the HP aC++ compiler as a front end for the linker. **$EXTRA_CFLAG** If the HP-UX platform is IA64 and 64-bit support is enabled, this flag contains the value **+DD64 -AA**; if 32-bit support is enabled, it contains the value **+DD32 -AA**. If the HP-UX platform is PA-RISC and 64-bit support is enabled, it contains the value **+DA2.0W**. For 32-bit support on a PA-RISC platform, this flag contains no value. **+DD64** Must be used to generate 64-bit code for HP-UX on IA64. **+DD32** Must be used to generate 32-bit code for HP-UX on IA64. **-AA** Allows ANSI C++ standard features such as namespace std and the C++ standard library on IA64. **+DA2.0W** Must be used to generate 64-bit code for HP-UX on PA-RISC. **-mt** Allows threads support for the HP aC++ compiler, needed as the routines can run in the same process as other routines (THREADSAFE) or in the engine itself (NOT FENCED). **-b** Create a shared library rather than a normal executable. **-o $1** Specify the executable. **$1.o** Specify the program object file. **$EXTRA_LFLAG** Specify the runtime path. If set, for 32-bit it contains the value `-Wl,+b$HOME/sqllib/lib32`, and for 64-bit: `-Wl,+b$HOME/sqllib/lib`. If not set, it contains no value. **-L$DB2PATH/$LIB** Specify the location of the DB2 runtime shared libraries. For 32-bit: `$HOME/sqllib/lib32`; for 64-bit: `$HOME/sqllib/lib`. **-ldb2** Link with the DB2 library. Refer to your compiler documentation for additional compiler options.

Related tasks:

- “Building UNIX C++ routines” on page 158

Related samples:

- "bldrtn -- Builds HP-UX C++ routines (stored procedures and UDFs) (C++)"

Building C++ multi-threaded applications on HP-UX

HP-UX provides a POSIX thread library and a DCE thread library. Only multi-threaded applications using the POSIX thread library are supported by DB2 on HP-UX.

For the HP-UX C++ compiler, `-mt` must be used for multi-threaded applications in both the compile and link steps.

The script, `bldmt`, contains the commands to build multi-threaded applications. Besides the options specified above, the compile and link options are the same as those used in the embedded SQL script file, `bldapp`.

Procedure:

To build the sample program, `dbthrds`, from the source file `dbthrds.sqC`, enter:

```
bldmt dbthrds
```

The result is an executable file, `dbthrds`. To run the executable file against the `sample` database, enter the executable name:

```
dbthrds
```

Related concepts:

- "Build files" on page 97

Related reference:

- "C samples" on page 64
- "HP-UX C++ application compile and link options" on page 199

Related samples:

- "bldmt -- Builds HP-UX C++ multi-threaded applications (C++)"
- "dbthrds.sqC -- How to use multiple context APIs on UNIX (C++)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

Micro Focus COBOL

For information on building Micro Focus COBOL applications on supported UNIX operating systems, see "Building UNIX Micro Focus COBOL applications" on page 161. For information on building Micro Focus COBOL routines on supported UNIX operating systems, see "Building UNIX Micro Focus COBOL routines" on page 162.

Configuring the Micro Focus COBOL compiler on HP-UX

If you develop applications that contain embedded SQL and DB2 API calls, and you are using the Micro Focus COBOL compiler, there are several points to keep in mind.

Procedure:

- When you precompile your application using the command line processor command `db2 prep`, use the `target mfcob` option.
- You must include the DB2 COBOL COPY file directory in the Micro Focus COBOL environment variable COBCPY. The COBCPY environment variable specifies the location of COPY files. The DB2 COPY files for Micro Focus COBOL reside in `sqllib/include/cobol_mf` under the database instance directory.

  To include the directory,

  – on bash or Korn shell, enter:

  ```
  export COBCPY=$COBCPY:$HOME/sqllib/include/cobol_mf
  ```

  – on C shell, enter:

  ```
  setenv COBCPY ${COBCPY}:${HOME}/sqllib/include/cobol_mf
  ```

  Note: You might want to set COBCPY in the `.profile` or `.login` file.

Related tasks:

- "Building UNIX Micro Focus COBOL applications" on page 161
- "Building UNIX Micro Focus COBOL routines" on page 162

Related reference:

- "HP-UX Micro Focus COBOL application compile and link options" on page 204
- "HP-UX Micro Focus COBOL routine compile and link options" on page 206

Build script for Micro Focus COBOL applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds HP-UX Micro Focus COBOL applications
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set COBCPY to include the DB2 COPY files directory.
COBCPY=$COBCPY:$DB2PATH/include/cobol_mf

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqb" ]
then
  ./embprep $1 $2 $3 $4
fi

# Compile the checkerr.cbl error checking utility.
cob -cx checkerr.cbl

# Compile the program.
cob -cx $1.cbl

# Link the program.
cob -x $1.o checkerr.o -L$DB2PATH/lib -ldb2 -ldb2gmf
```

HP-UX Micro Focus COBOL application compile and link options

The following are the compile and link options recommended by DB2 for building COBOL embedded SQL and DB2 API applications with the Micro Focus COBOL compiler on HP-UX, as demonstrated in the `bldapp` build script.

Compile and link options for bldapp
Compile options: **cob** The Micro Focus COBOL compiler. **-cx** Compile to object module.
Link options: **cob** Use the compiler as a front end for the linker. **-x** Specify an executable program. **$1.o** Include the program object file. **checkerr.o** Include the utility object file for error checking. **-L$DB2PATH/lib** Specify the location of the DB2 runtime shared libraries. **-ldb2** Link to the DB2 library. **-ldb2gmf** Link to the DB2 exception-handler library for Micro Focus COBOL. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX Micro Focus COBOL applications" on page 161

Related samples:

- "bldapp -- Builds HP-UX Micro Focus COBOL applications"

Build script for Micro Focus COBOL routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds HP-UX Micro Focus COBOL routines (stored procedures)
# Usage: bldrtn <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set COBCPY to include the DB2 COPY files directory.
COBCPY=$COBCPY:$DB2PATH/include/cobol_mf

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqb" ]
then
  embprep $1 $2
fi

# Compile the program.
cob +z -cx $1.cbl

# Link the program.
ld -b -o $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2gmf \
   -L$COBDIR/coblib -lcobol -lcrtn

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

HP-UX Micro Focus COBOL routine compile and link options

The following are the compile and link options recommended by DB2 for building COBOL routines (stored procedures) with the Micro Focus COBOL compiler on HP-UX, as demonstrated in the `bldrtn` build script.

Compile and link options for bldrtn
Compile options: **cob** The COBOL compiler. **+z** Generate position-independent code. **-cx** Compile to object module.
Link options: **ld** Use the linker to link. **-b** Create a shared library rather than a normal executable file. **-o $1** Specify the executable. **$1.o** Include the program object file. **-L$DB2PATH/lib** Specify the location of the DB2 runtime shared libraries. **-ldb2** Link to the DB2 shared library. **-ldb2gmf** Link to the DB2 exception-handler library for Micro Focus COBOL. **-L$COBDIR/coblib** Specify the location of the COBOL runtime libraries. **-lcobol** Link to the COBOL library. **-lcrtn** Link to the crtn library. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX Micro Focus COBOL routines" on page 162

Related samples:

- "bldrtn -- Builds HP-UX Micro Focus COBOL routines (stored procedures)"

Chapter 12. Linux

This chapter provides detailed information for building applications on Linux. For the latest DB2 application development updates for Linux, visit the Web page at:

`http://www.ibm.com/software/data/db2/udb/ad`

Linux C

Building information for DB2 CLI Applications and routines is in the *CLI Guide and Reference*.

For information on building C applications on supported UNIX operating systems, see "Building UNIX C applications" on page 147. For information on building C routines on supported UNIX operating systems, see "Building UNIX C routines" on page 151.

Build script for C applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds Linux C applications
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Determine if we are running with 32-bit, and
# if we are running with 32-bit on Linux AMD64
LIB="lib"
EXTRA_C_FLAGS=""
HARDWAREPLAT=`uname -m`
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $bitwidth = "\"32\"" ]; then
   LIB="lib32"
   if [ "$HARDWAREPLAT" = "x86_64" ]; then
       EXTRA_C_FLAGS="-m32"
   fi
fi

# The runtime path is recommended for all applications.
# If you need to use LD_LIBRARY_PATH, unset the RUNTIME
# variable by commenting out the following line.
RUNTIME=true
```

```
if [ "$RUNTIME" != "" ]
then
  EXTRA_LFLAG="-Wl,-rpath,$DB2PATH/$LIB"
else
  EXTRA_LFLAG=""
fi

# If an embedded SQL program, precompile and bind it.
# Note: some .sqc files contain no SQL but link in
# utilemb.sqc, so if you get this warning, ignore it:
# SQL0053W  No SQL statements were found in the program.
if [ -f $1".sqc" ]
then
  ./embprep $1 $2 $3 $4
  # Compile the utilemb.c error-checking utility.
  gcc $EXTRA_C_FLAGS -I$DB2PATH/include -c utilemb.c
else
  # Compile the utilapi.c error-checking utility.
  gcc $EXTRA_C_FLAGS -I$DB2PATH/include -c utilapi.c
fi

# Compile the program.
gcc $EXTRA_C_FLAGS -I$DB2PATH/include -c $1.c

if [ -f $1".sqc" ]
then
  # Link the program with utilemb.o.
  gcc $EXTRA_C_FLAGS -o $1 $1.o utilemb.o $EXTRA_LFLAG \
    -L$DB2PATH/$LIB -ldb2
else
  # Link the program with utilapi.o.
  gcc $EXTRA_C_FLAGS -o $1 $1.o utilapi.o $EXTRA_LFLAG \
    -L$DB2PATH/$LIB -ldb2
fi
```

Linux C application compile and link options

The following are the compile and link options recommended by DB2 for building C embedded SQL and DB2 API applications with the Linux C compiler, as demonstrated in the `bldapp` build script.

Compile and link options for bldapp	
Compile options:	
gcc	The GNU/Linux C compiler.
$EXTRA_C_FLAGS	Contains ″-m32″ for 32-bit support on AMD64; otherwise it contains no value.
-I$DB2PATH/include	Specify the location of the DB2 include files.
-c	Perform compile only; no link. This script file has separate compile and link steps.

Compile and link options for bldapp
Link options:
gcc Use the compiler as a front end for the linker.
$EXTRA_C_FLAGS Contains ″-m32″ for 32-bit support on AMD64; otherwise it contains no value.
-o $1 Specify the executable.
$1.o Specify the object file.
utilemb.o If an embedded SQL program, include the embedded SQL utility object file for error checking.
utilapi.o If a non-embedded SQL program, include the DB2 API utility object file for error checking.
$EXTRA_LFLAG If 'RUNTIME=true' is uncommented, for 32-bit it contains the value ″-Wl,-rpath,$DB2PATH/lib32″, and for 64-bit it contains the value ″-Wl,-rpath,$DB2PATH/lib″. Otherwise, it contains no value.
-L$DB2PATH/$LIB Specify the location of the DB2 static and shared libraries at link-time. For example, for 32-bit: `$HOME/sqllib/lib32`, and for 64-bit: `$HOME/sqllib/lib`. If you do not specify the `-L` option, `/usr/lib:/lib` is assumed.
-ldb2 Link with the DB2 library.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX C applications" on page 147

Related samples:

- "bldapp -- Builds Linux C applications (C)"

Build script for C routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds Linux C routines (stored procedures or UDFs)
# Usage: bldrtn <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Determine if we are running with 32-bit, and
# if we are running with 32-bit on Linux AMD64
LIB="lib"
EXTRA_C_FLAGS=""
HARDWAREPLAT=`uname -m`
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $bitwidth = "\"32\"" ]; then
   LIB="lib32"
   if [ "$HARDWAREPLAT" = "x86_64" ]; then
       EXTRA_C_FLAGS="-m32"
   fi
fi

# Set the runtime path.
```

```
EXTRA_LFLAG="-Wl,-rpath,$DB2PATH/$LIB"

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqc" ]
then
  ./embprep $1 $2
fi

# Compile the program.
gcc $EXTRA_C_FLAGS -fpic -I$DB2PATH/include -c $1.c -D_REENTRANT

# Link the program and create a shared library
gcc $EXTRA_C_FLAGS -shared -o $1 $1.o $EXTRA_LFLAG -L$DB2PATH/$LIB -ldb2 -lpthread

# Copy the shared library to the function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

Linux C routine compile and link options

The following are the compile and link options recommended by DB2 for building C routines (stored procedures and user-defined functions) with the Linux C compiler, as demonstrated in the `bldrtn` build script.

Compile and Link Options for bldrtn	
Compile options:	
gcc	The GNU/Linux C compiler.
$EXTRA_C_FLAGS	Contains ″-m32″ for 32-bit support on AMD64; otherwise it contains no value.
-fpic	Generate position independent code.
-I$DB2PATH/include	Specify the location of the DB2 include files.
-c	Perform compile only; no link. This script file has separate compile and link steps.
-D_REENTRANT	Defines _REENTRANT, needed as the routines can run in the same process as other routines (THREADSAFE) or in the engine itself (NOT FENCED).

Compile and Link Options for bldrtn
Link options:
gcc Use the compiler as a front end for the linker.
$EXTRA_CFLAG Contains ″-m32″ for 32-bit support on AMD64; otherwise it contains no value.
-shared Generate a shared library.
-o $1 Specify the executable.
$1.o Include the program object file.
$EXTRA_LFLAG Specify the location of the DB2 shared libraries at run-time. For 32-bit it contains the value ″-Wl,-rpath,$DB2PATH/lib32″. For 64-bit it contains the value ″-Wl,-rpath,$DB2PATH/lib″.
-L$DB2PATH/$LIB Specify the location of the DB2 static and shared libraries at link-time. For example, for 32-bit: `$HOME/sqllib/lib32`, and for 64-bit: `$HOME/sqllib/lib`. If you do not specify the `-L` option, `/usr/lib:/lib` is assumed.
-ldb2 Link with the DB2 library.
-lpthread Link with the POSIX thread library.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- “Building UNIX C routines” on page 151

Related samples:

- “bldrtn -- Builds Linux C routines (stored procedures or UDFs) (C)”

Building C multi-threaded applications on Linux

Multi-threaded applications using Linux C need to be compiled with `-D_REENTRANT` and linked with `-lpthread`.

The script, `bldmt`, contains the commands to build multi-threaded applications. Besides the options specified above, the compile and link options are the same as those used in the embedded SQL script file, `bldapp`.

Procedure:

To build the sample program, `dbthrds`, from the source file `dbthrds.sqc`, enter:

```
bldmt dbthrds
```

The result is an executable file, `dbthrds`. To run the executable file against the `sample` database, enter:

```
dbthrds
```

Related concepts:

- “Build files” on page 97

Related reference:

- "C samples" on page 64
- "Linux C application compile and link options" on page 208

Related samples:

- "bldmt -- Builds Linux C multi-threaded applications (C)"
- "dbthrds.sqc -- How to use multiple context APIs on UNIX (C)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

Linux C++

For information on building C++ applications on supported UNIX operating systems, see "Building UNIX C++ applications" on page 154. For information on building C++ routines on supported UNIX operating systems, see "Building UNIX C++ routines" on page 158.

Build script for C++ applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds Linux C++ applications
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Determine if we are running with 32-bit, and
# if we are running with 32-bit on Linux AMD64
LIB="lib"
EXTRA_C_FLAGS=""
HARDWAREPLAT=`uname -m`
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $bitwidth = "\"32\"" ]; then
   LIB="lib32"
   if [ "$HARDWAREPLAT" = "x86_64" ]; then
       EXTRA_C_FLAGS="-m32"
   fi
fi

# The runtime path is recommended for all applications.
# If you need to use LD_LIBRARY_PATH, unset the RUNTIME
# variable by commenting out the following line.
RUNTIME=true

if [ "$RUNTIME" != "" ]
then
  EXTRA_LFLAG="-Wl,-rpath,$DB2PATH/$LIB"
else
  EXTRA_LFLAG=""
fi

# If an embedded SQL program, precompile and bind it.
# Note: some .sqC files contain no SQL but link in
# utilemb.sqC, so if you get this warning, ignore it:
# SQL0053W  No SQL statements were found in the program.
if [ -f $1".sqC" ]
then
  ./embprep $1 $2 $3 $4
  # Compile the utilemb.C error-checking utility.
  g++ $EXTRA_C_FLAGS -I$DB2PATH/include -c utilemb.C
```

```
else
  # Compile the utilapi.C error-checking utility.
  g++ $EXTRA_C_FLAGS -I$DB2PATH/include -c utilapi.C
fi

# Compile the program.
g++ $EXTRA_C_FLAGS -I$DB2PATH/include -c $1.C

if [ -f $1".sqC" ]
then
  # Link the program with utilemb.o
  g++ $EXTRA_C_FLAGS -o $1 $1.o utilemb.o $EXTRA_LFLAG -L$DB2PATH/$LIB -ldb2
else
  # Link the program with utilapi.o
  g++ $EXTRA_C_FLAGS -o $1 $1.o utilapi.o $EXTRA_LFLAG -L$DB2PATH/$LIB -ldb2
fi
```

Linux C++ application compile and link options

The following are the compile and link options recommended by DB2 for building C++ embedded SQL and DB2 API applications with the Linux C++ compiler, as demonstrated in the `bldapp` build script.

Compile and link options for bldapp	
Compile options:	
g++	The GNU/Linux C++ compiler.
$EXTRA_C_FLAGS	Contains ″-m32″ for 32-bit support on AMD64; otherwise it contains no value.
-I$DB2PATH/include	Specify the location of the DB2 include files.
-c	Perform compile only; no link. This script file has separate compile and link steps.

Compile and link options for bldapp
Link options:
g++ Use the compiler as a front end for the linker.
$EXTRA_C_FLAGS Contains ″-m32″ for 32-bit support on AMD64; otherwise it contains no value.
-o $1 Specify the executable.
$1.o Include the program object file.
utilemb.o If an embedded SQL program, include the embedded SQL utility object file for error checking.
utilapi.o If a non-embedded SQL program, include the DB2 API utility object file for error checking.
$EXTRA_LFLAG If 'RUNTIME=true' is uncommented, for 32-bit it contains the value ″-Wl,-rpath,$DB2PATH/lib32″, and for 64-bit it contains the value ″-Wl,-rpath,$DB2PATH/lib″. Otherwise, it contains no value.
-L$DB2PATH/$LIB Specify the location of the DB2 static and shared libraries at link-time. For example, for 32-bit: `$HOME/sqllib/lib32`, and for 64-bit: `$HOME/sqllib/lib`. If you do not specify the `-L` option, `/usr/lib:/lib` is assumed.
-ldb2 Link with the DB2 library.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX C++ applications" on page 154

Related samples:

- "bldapp -- Builds Linux C++ applications (C++)"

Build script for C++ routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds Linux C++ routines (stored procedures and UDFs)
# Usage: bldrtn <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Determine if we are running with 32-bit, and
# if we are running with 32-bit on Linux AMD64
LIB="lib"
EXTRA_C_FLAGS=""
HARDWAREPLAT=`uname -m`
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $bitwidth = "\"32\"" ]; then
   LIB="lib32"
   if [ "$HARDWAREPLAT" = "x86_64" ]; then
       EXTRA_C_FLAGS="-m32"
   fi
fi

# Set the runtime path.
```

```
EXTRA_LFLAG="-Wl,-rpath,$DB2PATH/$LIB"

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqC" ]
then
  ./embprep $1 $2
fi

# Compile the program.
g++ $EXTRA_C_FLAGS -fpic -I$DB2PATH/include -c $1.C -D_REENTRANT

# Link the program and create a shared library.
g++ $EXTRA_C_FLAGS -shared -o $1 $1.o $EXTRA_LFLAG -L$DB2PATH/$LIB -ldb2 -lpthread

# Copy the shared library to the function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

Linux C++ routine compile and link options

These are the compile and link options recommended by DB2 for building C++ routines (stored procedures and user-defined functions) with the Linux C++ compiler, as demonstrated in the `bldrtn` build script.

Compile and link options for bldrtn	
Compile options:	
g++	The GNU/Linux C++ compiler.
$EXTRA_C_FLAGS	Contains ″-m32″ for 32-bit support on AMD64; otherwise it contains no value.
-fpic	Generate position independent code.
-I$DB2PATH/include	Specify the location of the DB2 include files.
-c	Perform compile only; no link. This script file has separate compile and link steps.
-D_REENTRANT	Defines _REENTRANT, needed as the routines can run in the same process as other routines (THREADSAFE) or in the engine itself (NOT FENCED).

Compile and link options for bldrtn
Link options:
g++ Use the compiler as a front end for the linker.
$EXTRA_C_FLAGS Contains ″-m32″ for 32-bit support on AMD64; otherwise it contains no value.
-shared Generate a shared library.
-o $1 Specify the executable.
$1.o Include the program object file.
$EXTRA_LFLAG Specify the location of the DB2 shared libraries at run-time. For 32-bit it contains the value ″-Wl,-rpath,$DB2PATH/lib32″. For 64-bit it contains the value ″-Wl,-rpath,$DB2PATH/lib″.
-L$DB2PATH/$LIB Specify the location of the DB2 static and shared libraries at link-time. For example, for 32-bit: `$HOME/sqllib/lib32`, and for 64-bit: `$HOME/sqllib/lib`. If you do not specify the `-L` option, `/usr/lib:/lib` is assumed.
-ldb2 Link with the DB2 library.
-lpthread Link with the POSIX thread library.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- “Building UNIX C++ routines” on page 158

Related samples:

- “bldrtn -- Builds Linux C++ routines (stored procedures and UDFs) (C++)”

Building C++ multi-threaded applications on Linux

Multi-threaded applications using Linux C++ need to be compiled with `-D_REENTRANT` and linked with `-lpthread`.

The script script, `bldmt`, contains the commands to build an embedded SQL multi-threaded program. Besides the options specified above, the compile and link options are the same as those used in the embedded SQL script file, `bldapp`.

Procedure:

To build the sample program, `dbthrds`, from the source file `dbthrds.sqC`, enter:

```
bldmt dbthrds
```

The result is an executable file, `dbthrds`. To run the executable file against the `sample` database, enter:

```
dbthrds
```

Related concepts:

- “Build files” on page 97

Related reference:

- "C samples" on page 64
- "Linux C++ application compile and link options" on page 213

Related samples:

- "bldmt -- Builds Linux C++ multi-threaded applications (C++)"
- "dbthrds.sqC -- How to use multiple context APIs on UNIX (C++)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

Micro Focus COBOL

Micro Focus COBOL is supported on Linux only on the following architectures:

- **Linux on Intel x86 (32-bit)**
- **Linux on s/390**

For information on building Micro Focus COBOL applications on supported UNIX operating systems, see "Building UNIX Micro Focus COBOL applications" on page 161. For information on building Micro Focus COBOL routines on supported UNIX operating systems, see "Building UNIX Micro Focus COBOL routines" on page 162.

Configuring the Micro Focus COBOL compiler on Linux

Procedure:

To run Micro Focus COBOL routines, the Linux run-time linker must be able to access certain COBOL shared libraries, and DB2 must be able to load these libraries. Since the program that does this loading runs with setuid privileges, it will only look for the dependent libraries in `/usr/lib`.

Create symbolic links to `/usr/lib` for the COBOL shared libraries. This must be done as root. The simplest way to do this is to link all COBOL library files from `$COBDIR/lib` to `/usr/lib`:

```
ln -s $COBDIR/lib/libcob* /usr/lib
```

where `$COBDIR` is where Micro Focus COBOL is installed, usually `/opt/lib/mfcobol`.

Here are the commands to link each individual file (assuming Micro Focus COBOL is installed in `/opt/lib/mfcobol`):

```
ln -s /opt/lib/mfcobol/lib/libcobrts.so /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobrts_t.so /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobrts.so.2 /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobrts_t.so.2 /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobcrtn.so /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobcrtn.so.2 /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobmisc.so /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobmisc_t.so /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobmisc.so.2 /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobmisc_t.so.2 /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobscreen.so /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobscreen.so.2 /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobtrace.so /usr/lib
```

```
ln -s /opt/lib/mfcobol/lib/libcobtrace_t.so /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobtrace.so.2 /usr/lib
ln -s /opt/lib/mfcobol/lib/libcobtrace_t.so.2 /usr/lib
```

The following need to be done on each DB2 instance:

- When you precompile your application using the command line processor command `db2 prep`, use the `target mfcob` option.
- You must include the DB2 COBOL COPY file directory in the Micro Focus COBOL environment variable COBCPY. The COBCPY environment variable specifies the location of the COPY files. The DB2 COPY files for Micro Focus COBOL reside in `sqllib/include/cobol_mf` under the database instance directory.

  To include the directory, enter:

  – On bash or Korn shell:

  ```
  export COBCPY=$HOME/sqllib/include/cobol_mf:$COBDIR/cpylib
  ```

  – On C shell:

  ```
  setenv COBCPY $HOME/sqllib/include/cobol_mf:$COBDIR/cpylib
  ```

- Update the environment variable:

  – On bash or Korn shell:

  ```
  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HOME/sqllib/lib:$COBDIR/lib
  ```

  – On C shell:

  ```
  setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:$HOME/sqllib/lib:$COBDIR/lib
  ```

- Set the DB2 Environment List:

  ```
  db2set DB2ENVLIST="COBDIR LD_LIBRARY_PATH"
  ```

Note: You might want to set COBCPY, COBDIR, and LD_LIBRARY_PATH in the `.bashrc`, `.kshrc` (depending on shell being used), `.bash_profile`, `.profile` (depending on shell being used), or in the `.login`. .

Related tasks:

- "Building UNIX Micro Focus COBOL applications" on page 161
- "Building UNIX Micro Focus COBOL routines" on page 162

Related reference:

- "Linux Micro Focus COBOL application compile and link options" on page 219
- "Linux Micro Focus COBOL routine compile and link options" on page 220

Build script for Micro Focus COBOL applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds Linux Micro Focus COBOL applications
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set COBCPY to include the DB2 COPY files directory.
COBCPY=$COBCPY:$DB2PATH/include/cobol_mf

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqb" ]
then
  ./embprep $1 $2 $3 $4
```

```
fi

# Compile the checkerr.cbl error checking utility.
cob -cx checkerr.cbl

# Compile the program.
cob -cx $1.cbl

# Link the program.
cob -x -o $1 $1.o checkerr.o -L$DB2PATH/lib -ldb2 -ldb2gmf
```

Linux Micro Focus COBOL application compile and link options

The following are the compile and link options recommended by DB2 for building COBOL embedded SQL and DB2 API applications with the Micro Focus COBOL compiler on Linux, as demonstrated in the `bldapp` build script.

Compile and link options for bldapp
Compile options: **cob** The Micro Focus COBOL compiler. **-cx** Compile to object module.
Link options: **cob** Use the compiler as a front end for the linker. **-x** Specify an executable program. **-o $1** Include the executable. **$1.o** Include the program object file. **checkerr.o** Include the utility object file for error checking. **-L$DB2PATH/lib** Specify the location of the DB2 runtime shared libraries. **-ldb2** Link to the DB2 library. **-ldb2gmf** Link to the DB2 exception-handler library for Micro Focus COBOL. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX Micro Focus COBOL applications" on page 161
- "Configuring the Micro Focus COBOL compiler on Linux" on page 217

Related samples:

- "bldapp -- Builds Linux Micro Focus COBOL applications"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

Build script for Micro Focus COBOL routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds Linux Micro Focus COBOL routines (stored procedures)
# Usage: bldrtn <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
```

```
# Set COBCPY to include the DB2 COPY files directory.
COBCPY=$COBCPY:$DB2PATH/include/cobol_mf

# Precompile and bind the program.
./embprep $1 $2

# Compile the program.
cob -cx $1.cbl

# Link the program.
cob -x -o $1 $1.o -Q -G -L$DB2PATH/lib -ldb2 -ldb2gmf

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

Linux Micro Focus COBOL routine compile and link options

The following are the compile and link options recommended by DB2 for building COBOL routines (stored procedures) with the Micro Focus COBOL compiler on Linux, as demonstrated in the `bldrtn` build script.

Compile and link options for bldrtn
Compile options: **cob** The COBOL compiler. **-cx** Compile to object module.
Link options: **cob** Use the compiler as a front end for the linker. **-x** Specify an executable program. **-o $1** Specify the executable. **$1.o** Include the program object file. **-Q -G** Generate a shared library. **-L$DB2PATH/lib** Specify the location of the DB2 runtime shared libraries. **-ldb2** Link to the DB2 library. **-ldb2gmf** Link to the DB2 exception-handler library for Micro Focus COBOL. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX Micro Focus COBOL routines" on page 162
- "Configuring the Micro Focus COBOL compiler on Linux" on page 217

Related samples:

- "bldrtn -- Builds Linux Micro Focus COBOL routines (stored procedures)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

Chapter 13. Solaris

This chapter provides detailed information for building applications in the Solaris operating environment. For the latest DB2 application development updates for Solaris, visit the Web page at:

```
http://www.ibm.com/software/data/db2/udb/ad
```

Solaris C

Building information for DB2 CLI Applications and routines is in the *CLI Guide and Reference*.

For information on building C applications on supported UNIX operating systems, see "Building UNIX C applications" on page 147. For information on building C routines on supported UNIX operating systems, see "Building UNIX C routines" on page 151.

Build script for C applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds Solaris C applications
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set compile and link flags for 32-bit and 64-bit programs.
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $bitwidth = "\"64\"" ];
then
  CFLAG_ARCH=v9
  LIB=lib
else
  CFLAG_ARCH=v8plusa
  LIB=lib32
fi

# Set the runtime path.
# LD_LIBRARY_PATH will be followed instead of the runtime path unless
# you unset LD_LIBRARY_PATH first to allow the runtime path to be used.
EXTRA_LFLAG="-R$DB2PATH/$LIB"

# If an embedded SQL program, precompile and bind it.
```

```
# Note: some .sqc files contain no SQL but link in
# utilemb.sqc, so if you get this warning, ignore it:
# SQL0053W  No SQL statements were found in the program.
if [ -f $1".sqc" ]
then
  ./embprep $1 $2 $3 $4
  # Compile the utilemb.c error-checking utility.
  cc -xarch=$CFLAG_ARCH -I$DB2PATH/include -c utilemb.c
else
  # Compile the utilapi.c error-checking utility.
  cc -xarch=$CFLAG_ARCH -I$DB2PATH/include -c utilapi.c
fi

# Compile the program.
cc -xarch=$CFLAG_ARCH -I$DB2PATH/include -c $1.c

if [ -f $1".sqc" ]
then
  # Link the program with utilemb.o
  cc -xarch=$CFLAG_ARCH -mt -o $1 $1.o utilemb.o \
    -L$DB2PATH/$LIB $EXTRA_LFLAG -ldb2
else
  # Link the program with utilapi.o
  cc -xarch=$CFLAG_ARCH -mt -o $1 $1.o utilapi.o \
    -L$DB2PATH/$LIB $EXTRA_LFLAG -ldb2
fi
```

Solaris C application compile and link options

These are the compile and link options recommended by DB2 for building C embedded SQL and DB2 API applications with the Forte C compiler, as demonstrated in the `bldapp` build script.

Compile and link options for bldapp	
Compile options:	
cc	The C compiler.
-xarch=$CFLAG_ARCH	This option ensures that the compiler will produce valid executables when linking with `libdb2.so`. The value for $CFLAG_ARCH is set to either "v8plusa" for 32-bit, or "v9" for 64-bit.
-I$DB2PATH/include	Specify the location of the DB2 include files. For example: `$HOME/sqllib/include`
-c	Perform compile only; no link. This script has separate compile and link steps.

Compile and link options for bldapp
Link options:
cc Use the compiler as a front end for the linker.
-xarch=$CFLAG_ARCH This option ensures that the compiler will produce valid executables when linking with `libdb2.so`. The value for $CFLAG_ARCH is set to either ″v8plusa″ for 32-bit, or ″v9″ for 64-bit.
-mt Link in multi-thread support. Needed for linking with libdb2. **Note:** If POSIX threads are used, DB2 applications also have to link with `-lpthread`, whether or not they are threaded.
-o $1 Specify the executable.
$1.o Include the program object file.
utilemb.o If an embedded SQL program, include the embedded SQL utility object file for error checking.
utilapi.o If not an embedded SQL program, include the DB2 API utility object file for error checking.
-L$DB2PATH/$LIB Specify the location of the DB2 static and shared libraries at link-time. For example, for 32-bit: `$HOME/sqllib/lib32`, and for 64-bit: `$HOME/sqllib/lib`.
$EXTRA_LFLAG Specify the location of the DB2 shared libraries at run-time. For 32-bit it contains the value ″-R$DB2PATH/lib32″, and for 64-bit it contains the value ″-R$DB2PATH/lib″.
-ldb2 Link with the DB2 library.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- “Building UNIX C applications” on page 147

Related samples:

- “bldapp -- Builds Solaris C applications (C)”

Build script for C routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds Solaris C routines (stored procedures or UDFs)
# Usage: bldrtn <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set compile and link flags for 32-bit and 64-bit programs.
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $bitwidth = "\"64\"" ];
then
  CFLAG_ARCH=v9
  LIB=lib
else
  CFLAG_ARCH=v8plusa
  LIB=lib32
```

```
fi

# Set the runtime path.
# LD_LIBRARY_PATH will be followed instead of the runtime path unless
# you unset LD_LIBRARY_PATH first to allow the runtime path to be used.
EXTRA_LFLAG="-R$DB2PATH/$LIB"

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqc" ]
then
  ./embprep $1 $2
fi

# Compile the program.
cc -xarch=$CFLAG_ARCH -mt -DUSE_UI_THREADS -Kpic \
  -I$DB2PATH/include -c $1.c

# Link the program and create a shared library
cc -xarch=$CFLAG_ARCH -mt -G -o $1 $1.o -L$DB2PATH/$LIB \
  $EXTRA_LFLAG -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

Solaris C routine compile and link options

These are the compile and link options recommended by DB2 for building C routines (stored procedures and user-defined functions) with the Forte C compiler, as demonstrated in the `bldrtn` build script.

Compile and link options for bldrtn	
Compile options:	
cc	The C compiler.
-xarch=$CFLAG_ARCH	This option ensures that the compiler will produce valid executables when linking with `libdb2.so`. The value for $CFLAG_ARCH is set to either "v8plusa" for 32-bit, or "v9" for 64-bit.
-mt	Allow multi-threaded support, needed as the routines can run in the same process as other routines (THREADSAFE) or in the engine itself (NOT FENCED).
-DUSE_UI_THREADS	Allows Sun's "UNIX International" threads APIs.
-Kpic	Generate position-independent code for shared libraries.
-I$DB2PATH/include	Specify the location of the DB2 include files.
-c	Perform compile only; no link. This script has separate compile and link steps.

Compile and link options for bldrtn
Link options:
cc Use the compiler as a front end for the linker.
-xarch=$CFLAG_ARCH This option ensures that the compiler will produce valid executables when linking with `libdb2.so`. The value for $CFLAG_ARCH is set to either ″v8plusa″ for 32-bit, or ″v9″ for 64-bit.
-mt This is required because the DB2 library is linked with `-mt`.
-G Generate a shared library.
-o $1 Specify the executable.
$1.o Include the program object file.
-L$DB2PATH/$LIB Specify the location of the DB2 static and shared libraries at link-time. For example, for 32-bit: `$HOME/sqllib/lib32`, and for 64-bit: `$HOME/sqllib/lib`.
$EXTRA_LFLAG Specify the location of the DB2 shared libraries at run-time. For 32-bit it contains the value ″-R$DB2PATH/lib32″, and for 64-bit it contains the value ″-R$DB2PATH/lib″.
-ldb2 Link with the DB2 library.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- “Building UNIX C routines” on page 151

Related samples:

- “bldrtn -- Builds Solaris C routines (stored procedures or UDFs) (C)”

Building C multi-threaded applications on Solaris

Multi-threaded applications using SUN and POSIX thread libraries are supported by DB2. The default is Sun threads. Multi-threaded applications using Forte C on Solaris need to be compiled and linked with `-mt`. This will pass `-D_REENTRANT` to the preprocessor, and `-lthread` to the linker. You also need to specify the compile define `-DUSE_UI_THREADS`, to use Sun’s ″Unix International″ threads APIs.

Note: If you want to use POSIX threads, you have to add the compiler option `-D_POSIX_PTHREAD_SEMANTICS`, which allows POSIX variants of functions such as `getpwnam_r()`, and also adds the link option `-lpthread`. If you are using the `bldmt` script provided, you also have to delete the `-DUSE_UI_THREADS` define.

The script, `bldmt` contains the commands to build a multi-threaded application. Besides the options specified above, the compile and link options are the same as those used in the embedded SQL script file, `bldapp`.

Procedure:

To build the sample program, `dbthrds`, from the source file `dbthrds.sqc`, enter:

```
bldmt dbthrds
```

The result is an executable file, `dbthrds`. To run the executable file against the `sample` database, enter:

```
dbthrds
```

Note: For multi-threaded programs with a fair number of connections, the kernel parameters `semsys:seminfo_semume` and `shmsys:shminfo_shmseg` might have to be set beyond their default values. Please see the related link below on the db2osconf utility to obtain recommendations on the values to set for these parameters.

Related concepts:

- "Build files" on page 97

Related reference:

- "C samples" on page 64
- "Solaris C application compile and link options" on page 222
- "db2osconf - Utility for Kernel Parameter Values Command" in the *Command Reference*

Related samples:

- "bldmt -- Builds Solaris C multi-threaded applications (C)"
- "dbthrds.sqc -- How to use multiple context APIs on UNIX (C)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

Solaris C++

For information on building C++ applications on supported UNIX operating systems, see "Building UNIX C++ applications" on page 154. For information on building C++ routines on supported UNIX operating systems, see "Building UNIX C++ routines" on page 158.

Build script for C++ applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds Solaris C++ applications
# Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set compile and link flags for 32-bit and 64-bit programs.
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $bitwidth = "\"64\"" ];
then
  CFLAG_ARCH=v9
  LIB=lib
else
  CFLAG_ARCH=v8plusa
  LIB=lib32
fi

# Set the runtime path.
# LD_LIBRARY_PATH will be followed instead of the runtime path unless
# you unset LD_LIBRARY_PATH first to allow the runtime path to be used.
EXTRA_LFLAG="-R$DB2PATH/$LIB"
```

```
# If an embedded SQL program, precompile and bind it.
# Note: some .sqC files contain no SQL but link in
# utilemb.sqC, so if you get this warning, ignore it:
# SQL0053W  No SQL statements were found in the program.
if [ -f $1".sqC" ]
then
  ./embprep $1 $2 $3 $4
  # Compile the utilemb.C error-checking utility.
  CC -xarch=$CFLAG_ARCH -I$DB2PATH/include -c utilemb.C
else
  # Compile the utilapi.C error-checking utility.
  CC -xarch=$CFLAG_ARCH -I$DB2PATH/include -c utilapi.C
fi

# Compile the program.
CC -xarch=$CFLAG_ARCH -I$DB2PATH/include -c $1.C

if [ -f $1".sqC" ]
then
  # Link the program with utilemb.o
  CC -xarch=$CFLAG_ARCH -mt -o $1 $1.o utilemb.o \
    -L$DB2PATH/$LIB $EXTRA_LFLAG -ldb2
else
  # Link the program with utilapi.o
  CC -xarch=$CFLAG_ARCH -mt -o $1 $1.o utilapi.o \
    -L$DB2PATH/$LIB $EXTRA_LFLAG -ldb2
fi
```

Solaris C++ application compile and link options

These are the compile and link options recommended by DB2 for building C++ embedded SQL and DB2 API applications with the Forte C++ compiler, as demonstrated in the `bldapp` build script.

Compile and link options for bldapp	
Compile options:	
CC	The C++ compiler.
-xarch=$CFLAG_ARCH	This option ensures that the compiler will produce valid executables when linking with `libdb2.so`. The value for $CFLAG_ARCH is set to either ″v8plusa″ for 32-bit, or ″v9″ for 64-bit.
-I$DB2PATH/include	Specify the location of the DB2 include files. For example: `$HOME/sqllib/include`
-c	Perform compile only; no link. This script has separate compile and link steps.

Compile and link options for bldapp	
Link options:	
CC	Use the compiler as a front end for the linker.
-xarch=$CFLAG_ARCH	This option ensures that the compiler will produce valid executables when linking with `libdb2.so`. The value for $CFLAG_ARCH is set to either "v8plusa" for 32-bit, or "v9" for 64-bit.
-mt	Link in multi-thread support. Needed for linking with libdb2. **Note:** If POSIX threads are used, DB2 applications also have to link with `-lpthread`, whether or not they are threaded.
-o $1	Specify the executable.
$1.o	Include the program object file.
utilemb.o	If an embedded SQL program, include the embedded SQL utility object file for error checking.
utilapi.o	If a non-embedded SQL program, include the DB2 API utility object file for error checking.
-L$DB2PATH/$LIB	Specify the location of the DB2 static and shared libraries at link-time. For example, for 32-bit: `$HOME/sqllib/lib32`, and for 64-bit: `$HOME/sqllib/lib`.
$EXTRA_LFLAG	Specify the location of the DB2 shared libraries at run-time. For 32-bit it contains the value "-R$DB2PATH/lib32", and for 64-bit it contains the value "-R$DB2PATH/lib".
-ldb2	Link with the DB2 library.
Refer to your compiler documentation for additional compiler options.	

Related tasks:

- "Building UNIX C++ applications" on page 154

Related samples:

- "bldapp -- Builds Solaris C++ applications (C++)"

Build script for C++ routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds Solaris C++ routines (stored procedures or UDFs)
# Usage: bldrtn <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set compile and link flags for 32-bit and 64-bit programs.
bitwidth=`LANG=C db2level | awk '/bits/{print $5}'`
if [ $bitwidth = "\"64\"" ];
then
  CFLAG_ARCH=v9
  LIB=lib
else
  CFLAG_ARCH=v8plusa
  LIB=lib32
```

```
fi

# Set the runtime path.
# LD_LIBRARY_PATH will be followed instead of the runtime path unless
# you unset LD_LIBRARY_PATH first to allow the runtime path to be used.
EXTRA_LFLAG="-R$DB2PATH/$LIB"

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqC" ]
then
  ./embprep $1 $2
fi

# Compile the program.
CC -xarch=$CFLAG_ARCH -mt -DUSE_UI_THREADS -Kpic \
  -I$DB2PATH/include -c $1.C

# Link the program and create a shared library
CC -xarch=$CFLAG_ARCH -mt -G -o $1 $1.o -L$DB2PATH/$LIB \
  $EXTRA_LFLAG -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

Solaris C++ routine compile and link options

These are the compile and link options recommended by DB2 for building C++ routines (stored procedures and user-defined functions) with the Forte C++ compiler, as demonstrated in the `bldrtn` build script.

Compile and link options for bldrtn	
Compile options:	
CC	The C++ compiler.
-xarch=$CFLAG_ARCH	This option ensures that the compiler will produce valid executables when linking with `libdb2.so`. The value for $CFLAG_ARCH is set to either "v8plusa" for 32-bit, or "v9" for 64-bit.
-mt	Allow multi-threaded support, needed as the routines can run in the same process as other routines (THREADSAFE) or in the engine itself (NOT FENCED).
-DUSE_UI_THREADS	Allows Sun's "UNIX International" threads APIs.
-Kpic	Generate position-independent code for shared libraries.
-I$DB2PATH/include	Specify the location of the DB2 include files.
-c	Perform compile only; no link. This script has separate compile and link steps.

Compile and link options for bldrtn
Link options:
CC Use the compiler as a front end for the linker.
-xarch=$CFLAG_ARCH This option ensures that the compiler will produce valid executables when linking with `libdb2.so`. The value for $CFLAG_ARCH is set to either ″v8plusa″ for 32-bit, or ″v9″ for 64-bit.
-mt This is required because the DB2 library is linked with `-mt`.
-G Generate a shared library.
-o $1 Specify the executable.
$1.o Include the program object file.
-L$DB2PATH/$LIB Specify the location of the DB2 static and shared libraries at link-time. For example, for 32-bit: `$HOME/sqllib/lib32`, and for 64-bit: `$HOME/sqllib/lib`.
$EXTRA_LFLAG Specify the location of the DB2 shared libraries at run-time. For 32-bit it contains the value ″-R$DB2PATH/lib32″, and for 64-bit it contains the value ″-R$DB2PATH/lib″.
-ldb2 Link with the DB2 library.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- “Building UNIX C++ routines” on page 158

Related samples:

- “bldrtn -- Builds Solaris C++ routines (stored procedures or UDFs) (C++)”

Building C++ multi-threaded applications on Solaris

Multi-threaded applications using SUN and POSIX thread libraries are supported by DB2. The default is Sun threads. Multi-threaded applications using Forte C++ on Solaris need to be compiled and linked with `-mt`. This will pass `-D_REENTRANT` to the preprocessor, and `-lthread` to the linker. You also need to specify the compile define `-DUSE_UI_THREADS`, to use Sun’s ″Unix International″ threads APIs.

Note: If you want to use POSIX threads, you have to add the compiler option `-D_POSIX_PTHREAD_SEMANTICS`, which allows POSIX variants of functions such as `getpwnam_r()`, and also adds the link option `-lpthread`. If you are using the `bldmt` script provided, you also have to delete the `-DUSE_UI_THREADS` define.

The script, `bldmt`, contains the commands to build a multi-threaded application. Besides the options specified above, the compile and link options are the same as those used in the embedded SQL script file, `bldapp`.

Procedure:

To build the sample program, `dbthrds`, from the source file `dbthrds.sqC`, enter:

```
bldmt dbthrds
```

The result is an executable file, `dbthrds`. To run the executable file against the `sample` database, enter:

```
dbthrds
```

Note: For multi-threaded programs with a fair number of connections, the kernel parameters `semsys:seminfo_semume` and `shmsys:shminfo_shmseg` might have to be set beyond their default values. Please see the related link below on the db2osconf utility to obtain recommendations on the values to set for these parameters.

Related concepts:

- "Build files" on page 97

Related reference:

- "C samples" on page 64
- "Solaris C++ application compile and link options" on page 227
- "db2osconf - Utility for Kernel Parameter Values Command" in the *Command Reference*

Related samples:

- "bldmt -- Builds Solaris C++ multi-threaded applications (C++)"
- "dbthrds.sqC -- How to use multiple context APIs on UNIX (C++)"
- "embprep -- To prep and bind C/C++ and Micro Focus COBOL embedded SQL programs (C)"

Micro Focus COBOL

For information on building Micro Focus COBOL applications on supported UNIX operating systems, see "Building UNIX Micro Focus COBOL applications" on page 161. For information on building Micro Focus COBOL routines on supported UNIX operating systems, see "Building UNIX Micro Focus COBOL routines" on page 162.

Configuring the Micro Focus COBOL compiler on Solaris

If you develop applications that contain embedded SQL and DB2 API calls, and you are using the Micro Focus COBOL compiler, these are points you have to keep in mind.

Procedure:

- When you precompile your application using the command line processor command `db2 prep`, use the `target mfcob` option.
- You must include the DB2 COBOL COPY file directory in the Micro Focus COBOL environment variable COBCPY. The COBCPY environment variable specifies the location of COPY files. The DB2 COPY files for Micro Focus COBOL reside in `sqllib/include/cobol_mf` under the database instance directory.

  To include the directory, enter:

  – On bash or Korn shells:

  ```
  export COBCPY=$COBCPY:$HOME/sqllib/include/cobol_mf
  ```

  – On C shell:

  ```
  setenv COBCPY $COBCPY:$HOME/sqllib/include/cobol_mf
  ```

Note: You might want to set COBCPY in the `.profile` file.

Related tasks:

- "Building UNIX Micro Focus COBOL applications" on page 161
- "Building UNIX Micro Focus COBOL routines" on page 162

Related reference:

- "Solaris Micro Focus COBOL application compile and link options" on page 232
- "Solaris Micro Focus COBOL routine compile and link options" on page 234

Build script for Micro Focus COBOL applications

```
#! /bin/sh
# SCRIPT: bldapp
# Builds Solaris Micro Focus COBOL applications
# Usage: bldapp [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set COBCPY to include the DB2 COPY files directory.
COBCPY=$COBCPY:$DB2PATH/include/cobol_mf

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqb" ]
then
  ./embprep $1 $2 $3 $4
fi

# Compile the checkerr.cbl error-checking utility.
cob -cx checkerr.cbl

# Compile the program.
cob -cx $1.cbl

# Link the program.
cob -x $1.o checkerr.o -L$DB2PATH/lib -ldb2 -ldb2gmf
```

Solaris Micro Focus COBOL application compile and link options

The following are the compile and link options recommended by DB2 for building COBOL embedded SQL and DB2 API applications with the Micro Focus COBOL compiler on Solaris, as demonstrated in the `bldapp` build script.

Compile and link options for bldapp	
Compile options:	
cob	The Micro Focus COBOL compiler.
-cx	Compile to object module.

Compile and link options for bldapp
Link options:
cob Use the compiler as a front end for the linker.
-x Specify an executable program.
$1.o Include the program object file.
checkerr.o Include the utility object file for error-checking.
-L$DB2PATH/lib Specify the location of the DB2 static and shared libraries at link-time. For example: `$HOME/sqllib/lib`.
-ldb2 Link with the DB2 library.
-ldb2gmf Link with the DB2 exception-handler library for Micro Focus COBOL.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX Micro Focus COBOL applications" on page 161

Related samples:

- "bldapp -- Builds Solaris Micro Focus COBOL applications"

Build script for Micro Focus COBOL routines

```
#! /bin/sh
# SCRIPT: bldrtn
# Builds Solaris Micro Focus COBOL routines (stored procedures)
# Usage: bldrtn <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Set COBCPY to include the DB2 COPY files directory.
COBCPY=$COBCPY:$DB2PATH/include/cobol_mf

# If an embedded SQL program, precompile and bind it.
if [ -f $1".sqb" ]
then
  ./embprep $1 $2
fi

# Compile the program.
cob -cx $1.cbl

# Link the program.
cob -yo $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2gmf

# Copy the shared library to the sqllib/function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

Solaris Micro Focus COBOL routine compile and link options

The following are the compile and link options recommended by DB2 for building COBOL routines (stored procedures) with the Micro Focus COBOL compiler on Solaris, as demonstrated in the `bldrtn` build script.

Compile and link options for bldrtn
Compile options:
cob The COBOL compiler.
-cx Compile to object module.
Link options:
cob Use the compiler as a front-end for the linker.
-y Create a self-contained standalone shared library.
-o $1 Specify the executable program.
$1.o Specify the program object file.
-L$DB2PATH/lib Specify the location of the DB2 runtime shared libraries. For example: `$HOME/sqllib/lib`. If you do not specify the `-L` option, the compiler assumes the following path: `/usr/lib:/lib`.
-ldb2 Link to the DB2 library.
-ldb2gmf Link to the DB2 exception-handler library for Micro Focus COBOL.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building UNIX Micro Focus COBOL routines" on page 162

Related samples:

- "bldrtn -- Builds Solaris Micro Focus COBOL routines (stored procedures)"

Chapter 14. Windows

This chapter provides detailed information for building applications on Windows operating systems. For the latest DB2 application development updates for Windows, visit the Web page at:

```
http://www.ibm.com/software/data/db2/udb/ad
```

WCHARTYPE CONVERT precompile option

The WCHARTYPE precompile option determines whether to handle graphic data in either multi-byte format or wide-character format using the `wchar_t` data type.

For DB2® for Windows® operating systems, the WCHARTYPE CONVERT option is supported for applications compiled with the Microsoft® Visual C++ compiler. However, do not use the CONVERT option with this compiler if your application inserts data into a DB2 database in a code page that is different from the database code page. DB2 normally performs a code page conversion in this situation; however, the Microsoft C run-time environment does not handle substitution characters for certain double byte characters. This could result in run time conversion errors.

The default option for WCHARTYPE is NOCONVERT. With the NOCONVERT option, no implicit character conversion occurs between application and the database manager. Data in a graphic host variable is sent to and received from the database manager as unaltered Double Byte Character Set (DBCS) characters.

If you need to convert your graphic data to multi-byte format from wide-character format, use the `wcstombs()` function. For example:

```
wchar_t widechar[200];
wchar_t mb[200];
wcstombs((char *)mb,widechar,200);

EXEC SQL INSERT INTO TABLENAME VALUES(:mb);
```

Similarly, you can use the `mbstowcs()` function to convert from multi-byte to wide-character format.

Do not issue a `setlocale()` call from your application if your application is statically bound to the C run-time libraries, as this might lead to C run-time conversion errors. Using `setlocale()` is not a problem if your application is dynamically bound to the C run-time library. This is also the case for routines (stored procedures and user-defined functions).

Related concepts:

- "wchar_t and sqldbchar Data Types in C and C++" in the *Application Development Guide: Programming Client Applications*
- "WCHARTYPE Precompiler Option in C and C++" in the *Application Development Guide: Programming Client Applications*
- "Graphic host variables in C/C++ routines" in the *Application Development Guide: Programming Server Applications*

Related reference:

- "PRECOMPILE Command" in the *Command Reference*

Object Linking and Embedding Database (OLE DB) table functions

DB2® supports OLE DB table functions. For these functions, there is no application building needed besides creating the `CREATE FUNCTION` DDL. OLE DB table function sample files are provided by DB2 in the `sqllib\samples\oledb` directory. These are Command Line Processor (CLP) files. They can be built with the following steps:

1. db2 connect to database_name
2. db2 -t -v -f *file_name*.db2
3. db2 terminate

where `database_name` is the database you are connecting to, and *file_name* is the name of the CLP file, with extension `.db2`.

These commands must be done in a DB2 Command Window.

Related concepts:

- "OLE DB Table Functions" in the *Application Development Guide: Programming Client Applications*
- "OLE DB user-defined table functions" in the *Application Development Guide: Programming Server Applications*

Related reference:

- "Object Linking and Embedding Database (OLE DB) table function samples" on page 86

Windows Management Instrumentation (WMI)

Windows® Management Instrumentation (WMI) is a key component of Microsoft®'s Windows management services. WMI provides a consistent and richly descriptive model of the configuration, status, and operational aspects of applications and the system.

The DB2® WMI provider allows WMI applications to monitor DB2 server services, enumerate and create databases, configure operational settings and perform database backup, restore, and roll-forward operations.

DB2 provides WMI sample files for the Visual Basic Scripting language located in the `sqllib\samples\wmi` directory. Before running the sample programs, ensure that the DB2 WMI Provider is registered by running the following commands:

```
mofcomp  %DB2PATH%\bin\db2wmi.mof
regsvr32 %DB2PATH%\bin\db2wmi.dll
```

where `%DB2PATH%` is the path where DB2 is installed.

Use the `cscript` command to run the Visual Basic Script samples. For example, to run the `listsvr` sample script, enter:

```
cscript listsvr.vbs
```

Related concepts:

- "Introduction to Windows Management Instrumentation (WMI)" in the *Administration Guide: Implementation*
- "DB2 Universal Database integration with Windows Management Instrumentation" in the *Administration Guide: Implementation*

Related reference:

- "Windows Management Instrumentation samples" on page 96

Microsoft Visual Basic

Building ADO applications with Visual Basic

ActiveX Data Objects (ADO) allow you to write an application to access and manipulate data in a database server through an OLE DB provider. The primary benefits of ADO are high speed, ease of use, low memory overhead, and a small disk footprint.

Visual Basic ADO sample programs are located in the `sqllib\samples\VB\ADO` directory.

Note: To run the DB2 ADO samples, these versions or later of the following components are recommended:

1. Visual Basic 6.0 Professional Edition
2. Microsoft Data Access 2.7 SDK (optionally installed with DB2 Version 8)

3. Visual Basic Service pack 5 from http://msdn.microsoft.com/vstudio/sp/vs6sp5/vbfixes.asp.
4. The latest Visual Studio Service Pack from http://msdn.microsoft.com/vstudio/.

Procedure:

You can use either of two ODBC-compliant providers:

- IBM OLE DB provider for DB2
- Microsoft OLE DB provider for ODBC

Using the IBM OLE DB provider for DB2

DB2 Version 8.2 Clients on Windows operating systems will optionally install IBMDADB2, the IBM OLE DB 2.0-compliant provider for DB2. The provider exposes interfaces for consumers who want to access data in a DB2 database. The IBM OLE DB provider for DB2 supports the following ADO application types:

- Microsoft Active Server Pages (ASP)
- Microsoft Visual Studio C++ and Visual Basic applications
- Microsoft Visual Interdev

For details on these types of applications, refer to the ADO documentation.

To access a DB2 server using the IBM OLE DB provider for DB2, the Visual Basic application should specify the PROVIDER keyword in the ADO connection string as follows:

```
Dim c1 As ADODB.Connection
Dim c1str As String
c1str = "Provider=ibmdadb2; DSN=db2alias; UID=userid; PWD=password"
c1.Open c1str
...
```

where `db2alias` is the alias for the DB2 database which is cataloged in the DB2 database directory.

Note: When using the IBM OLE DB provider for DB2, you do not need to perform the ODBC catalog step for the datasource. This step is required when you are using the OLE DB provider for ODBC.

Using the Microsoft OLE DB provider for ODBC

To use ADO with the Microsoft OLE DB provider and Visual Basic, you need to establish a reference to the ADO type library. Do the following:

1. Select ″References″ from the Project menu
2. Check the box for ″Microsoft ActiveX Data Objects <version_number> Library″
3. Click ″OK″.

where <version_number> is the current version the ADO library.

Once this is done, ADO objects, methods, and properties will be accessible through the VBA Object Browser and the IDE Editor.

Establish a connection:

```
Dim db As Connection
Set db = New Connection
```

Set client-side cursors supplied by the local cursor library:

```
db.CursorLocation = adUseClient
```

and set the provider so ADO will use the Microsoft ODBC Driver.

Accessing the sample database with ADO

A full Visual Basic program includes forms and other graphical elements, and you need to view it inside the Visual Basic environment. Here are Visual Basic commands as part of a program to access the DB2 `sample` database, after you have connected to the database with either the IBM OLE DB provider or the Microsoft OLE DB provider, as discussed above.

Open the `sample` database without specifying a user ID or password; that is, use the current user:

```
db.Open "SAMPLE"
```

Create a record set:

```
Set adoPrimaryRS = New Recordset
```

Use a select statement to fill the record set:

```
adoPrimaryRS.Open "select EMPNO,LASTNAME,FIRSTNME,MIDINIT,EDLEVEL,JOB
from EMPLOYEE Order by EMPNO", db
```

From this point, the programmer can use the ADO methods to access the data such as moving to the next record set:

```
adoPrimaryRS.MoveNext
```

Deleting the current record in the record set:

```
adoPrimaryRS.Delete
```

As well, the programmer can do the following to access an individual field:

```
Dim Text1 as String
Text1 = adoPrimaryRS!LASTNAME
```

Related concepts:

- "Purpose of the IBM OLE DB Provider for DB2" in the *Application Development Guide: Programming Client Applications*
- "Application Types Supported by the IBM OLE DB Provider for DB2" in the *Application Development Guide: Programming Client Applications*
- "Connections to Data Sources with Visual Basic ADO Applications" in the *Application Development Guide: Programming Client Applications*
- "OLE DB Services Automatically Enabled by IBM OLE DB Provider" in the *Application Development Guide: Programming Client Applications*
- "Large Object Manipulation with the IBM OLE DB Provider" in the *Application Development Guide: Programming Client Applications*
- "MTS and COM+ Distributed Transaction Support and the IBM OLE DB Provider" in the *Application Development Guide: Programming Client Applications*
- "IBM OLE DB Provider Restrictions" in the *Application Development Guide: Programming Client Applications*
- "ActiveX Data Objects and Remote Data Objects" in the *Application Development Guide: Programming Client Applications*

Related reference:

- "IBM OLE DB Provider Support for OLE DB Components and Interfaces" in the *Application Development Guide: Programming Client Applications*
- "IBM OLE DB Provider support for OLE DB properties" in the *Application Development Guide: Programming Client Applications*
- "IBM OLE DB Provider Support for ADO Methods and Properties" in the *Application Development Guide: Programming Client Applications*
- "Visual Basic samples" on page 93

Building loosely-coupled transactions with Visual Basic

XA provides two ways by which application threads of control can participate in a single XA global transaction: tightly-coupled and loosely-coupled. The sample project, `LCTransTest`, demonstrates XA loosely-coupled transactions. The sample files are located in the `sqllib\samples\VB\MTS` directory.

Procedure:

To build and run the loosely-coupled transactions sample, follow these steps:

1. **Build the LCTransTest.vbp project**
   a. Open the "LCTransTest.vbp" project by double clicking it.
   b. If you received an error message: "Unable to set the version compatible component X:\...\LCTransTest.dll", click "OK" to continue.
   c. Compile the project. Go to "File" -> "Make LCTransTest.dll", then click "OK".
   d. To fix the version incompatibility problem, right click on the "LCTransTest (LCTransTest.vbp)" project located on the upper right panel. Then choose "LCTransTest Properties". On the "LCTransTest - Project Properties" window. Click on the "Component" tab. Under the "Version Compatibility" Section, select "Binary Compatibility".
   e. Save the project. Go to "File" -> "Save Project".
   f. Close the project.
2. **Build the Main.vbp project**
   a. Open the "Main.vbp" project by double clicking it.
   b. It is likely that you will receive the warning message: "Could not create reference: X:\...\LCTransTest.dll", click "OK" to continue.
   c. Go to "Project" -> "References" (on the tool bar).
   d. On the "References - main.vbp" window, make sure the "Microsoft ActiveX Data Objects 2.7 Library" box is checked. Go to "Browse...". Find the LCTransTest.dll you generated in Step 1 and click "Open" to add this reference. Click "OK" in "References - main.vbp" window.
   e. Compile the project. Go to "File" -> "Make main.exe", then click "OK".
   f. Save the project. Go to "File" -> "Save Project".
   g. Close the project.
3. **Other settings**
   a. On Windows, go to "Start" -> "Settings" -> "Control Panel" -> "Administration Tools" -> "Component Services".
   b. On the "Component Services" window, expand "Component Services" on the left panel until you see "COM+ Applications".
   c. Right click on "COM+ Applications", select "New" -> "Application".

d. On the pop-up window, ″COM Application Install Wizard″, click ″Next″.
e. Select ″Create an empty application″.
f. Enter ″LCTransTest″ as the name of the new application. Keep ″Activation type″ as ″Server application″. Click ″Next″.
g. Click ″Next″, then ″Finish″.
h. Expand ″LCTransTest″, right click on ″Components″. Go to ″New″ -> ″Components″ -> choose ″Import components that are already registered″ _> click on ″LCTransTest.TestClass″ -> ″Next″ -> ″Finish″.
i. Expand ″Components″. Right click on ″LCTransTest.TestClass″. Go to ″Properties″. Under the ″Transaction″ tab, check ″Required″ for ″Transaction support″
j. Restart the Microsoft Distributed Transaction Coordinator by right clicking on ″Component Services″ -> ″Computers″ -> ″My Computer″. Choose ″Stop MS DTC″. Wait until it stopped, then right click on ″My Computer″ -> ″Start MS DTC″ to restart DTC.

4. **To run the sample in debug mode**
   a. Open LCTransTest.vbp.
   b. In LCTransTest, ensure that ″project properties*″, under the ″Debugging″ tab has ″Wait for components to be created″ checked. (It should be by default.)
   c. Put a break point on the line ″con1.Open connString″ (by putting your cursor on that line, and pressing F9).
   d. Press F5 (or select ″Start″ under the ″Run″ pull-down menu). When this dll gets loaded, and this method runs, the debugger will stop execution at that break point.
   e. Open main.vbp.
   f. In the main.vbp, set up the command line arguments. In Project properties, under the ″Make″ tab, in the ″Command Line arguments″ text box, type the following:

      ```
      provider=ibmdadb2;dsn=<dbname>;uid=<userid>;pwd=<password> <filename>
      ```

      where <dbname> is the name of a database you have, and <filename> is the name of a file (including the path) that will contain output information. For instance, C:\lctoutput.txt. Then click OK.
   g. Within Visual Basic, you can use the ″Debug ->Step Into″ <F8> or ″Debug->Step Over″ <shift + F8> to run the executable one line of code at a time.
   h. When you get to the line that calls ″transTest.RunTest″ in the main executable, and try to step over it, the other Visual Basic window (the LCTransTest project that you have open) will come to the front, and you'll be stopped at the breakpoint you put there. Then you can use ″Step Into″ or ″Step Over″ to progress through the RunTest method one line of code at a time.

Related tasks:

- "Building ADO applications with Visual Basic" on page 237
- "Troubleshooting a Visual Basic loosely-coupled transaction project" on page 242

Related reference:

- "Visual Basic samples" on page 93

Troubleshooting a Visual Basic loosely-coupled transaction project

A feature of the Visual Basic integrated environment is that each time you compile a data-linked library (dll), Visual Basic creates a new globally unique identifier (GUID) for it, and registers it in the Windows registry. After the project dll is built, it is registered with the distributed transaction coordinator (DTC) using the GUID. If the project dll is rebuilt, Visual Basic gives it a new GUID, and also registers it in the Windows registry. Therefore, the GUID that the DTC is using to refer to the project dll is now out of date, and there are two different entries in the windows registry for the project.

Procedure:

To avoid this problem, after building the project for the first time, modify the project properties:

1. Under the Component tab, select ″Binary Compatibility″, and then use the ″...″ button to find the whole path for the dll you just built.
2. Then click on the ″OK″ button, and save the project immediately.

Now, every time you recompile the project dll, it will keep the same GUID. However, if you change the interface to the dll (such as by adding or removing methods, or changing the parameters of existing methods) then you have to use a new GUID.

If there are already multiple GUID entries in the windows registry, do the following:

1. Search for the project name with the registry editor.
2. Remove all occurrences of the project name other than what is found in the Visual Basic recent project list.

The connection information, including user ID and password, must be identical in the DTC and in the Visual Basic application in order for the loosely-coupled transaction to occur. Normally, the public methods of a dll are used to directly access it. However, when the DTC is involved, it encapsulates the dll, and intercepts all incoming calls to the methods, and outgoing results from those methods. In this way, the DTC can tell when the database activity in one of those methods should be loosely-coupled with other database activity of the same object or with database activity of a different object.

To avoid any problem this might cause, do the following:

1. In the Application properties in the DTC under the ″Identity″ tab, select ″This user:″.
2. Use the Browse button to find the ID of the user who will run this project.
3. Use the same user ID and password in the project connection string.

If you use the same user ID and password in the Visual Basic application connection string as the one you used to log onto the computer, you do not have to take this additional step.

To ensure the loosely-coupled transaction project is working properly, you can do the following:

1. Look at the output file that the executable creates, and confirm that the database updates are happening.

2. Examine a cli trace for ENLIST_IN_DTC.

If either of these tests fail, then the dll is not registered properly with the DTC, and loosely-coupled transactions are not occurring.

Related tasks:

- "Building ADO applications with Visual Basic" on page 237
- "Building loosely-coupled transactions with Visual Basic" on page 240

Related reference:

- "Visual Basic samples" on page 93

Building RDO applications with Visual Basic

Remote Data Objects (RDO) provide an information model for accessing remote data sources through ODBC. RDO offers a set of objects that make it easy to connect to a database, execute queries and stored procedures, manipulate results, and commit changes to the server. It is specifically designed to access remote ODBC relational data sources, and makes it easier to use ODBC without complex application code, and is a primary means of accessing a relational database that is exposed with an ODBC driver. RDO implements a thin code layer over the Open Database Connectivity (ODBC) API and driver manager that establishes connections, creates result sets and cursors, and executes complex procedures using minimal workstation resources.

DB2 provides Visual Basic RDO sample programs in the `sqllib\samples\VB` directory.

Procedure:

To use RDO with Microsoft Visual Basic, you need to establish a reference to your Visual Basic project. Do the following:

1. Select "References" from the Project menu
2. Check the box for "Microsoft Remote Data Object <Version Number>"
3. Click "OK".

where <version_number> is the current RDO version.

A full Visual Basic program includes forms and other graphical elements, and you need to view it inside the Visual Basic environment. Here are Visual Basic commands as part of a DB2 program that connects to the `sample` database, opens a record set that selects all the columns from the `EMPLOYEE` table, and then displays the employee names in a message window, one by one:

```
Dim rdoEn As rdoEngine
Dim rdoEv As rdoEnvironment
Dim rdoCn As rdoConnection
Dim Cnct$
Dim rdoRS As rdoResultset
Dim SQLQueryDB As String
```

Assign the connection string:

```
Cnct$ = "DSN=SAMPLE;UID=;PWD=;"
```

Set the RDO environment:

```
Set rdoEn = rdoEngine
Set rdoEv = rdoEn.rdoEnvironments(0)
```

Connect to the database:

```
Set rdoCn = rdoEv.OpenConnection("", , , Cnct$)
```

Assign the SELECT statement for the record set:

```
SQLQueryDB = "SELECT * FROM EMPLOYEE"
```

Open the record set and execute the query:

```
Set rdoRS = rdoCn.OpenResultset(SQLQueryDB)
```

While not at the end of the record set, display Message Box with LASTNAME, FIRSTNME from table, one employee at a time:

```
While Not rdoRS.EOF
MsgBox rdoRS!LASTNAME & ", " & rdoRS!FIRSTNME
```

Move to the next row in the record set:

```
rdoRS.MoveNext
Wend
```

Close the program:

```
rdoRS.Close
rdoCn.Close
rdoEv.Close
```

Related concepts:

- "ActiveX Data Objects and Remote Data Objects" in the *Application Development Guide: Programming Client Applications*

Related reference:

- "Visual Basic samples" on page 93

Object Linking and Embedding (OLE) automation with Visual Basic

You can implement OLE automation UDFs and stored procedures in any language, as OLE is language independent. You do this by exposing methods of OLE automation servers, and registering the methods as UDFs with DB2®. Application development environments which support the development of OLE automation servers include certain versions of the following: Microsoft® Visual Basic, Microsoft Visual C++, Microsoft Visual J++, Microsoft FoxPro, Borland Delphi, Powersoft PowerBuilder, and Micro Focus COBOL. Also, Java™ beans objects that are wrapped properly for OLE, for example with Microsoft Visual J++, can be accessed via OLE automation.

You need to refer to the documentation of the appropriate application development environment for further information on developing OLE automation servers.

OLE automation UDFs and stored procedures

Microsoft Visual Basic supports the creation of OLE automation servers. A new kind of object is created in Visual Basic by adding a class module to the Visual Basic project. Methods are created by adding public sub-procedures to the class module. These public procedures can be registered to DB2 as OLE automation UDFs and stored procedures. For further information on creating and building

OLE servers, refer to the Microsoft Visual Basic manual, *Creating OLE Servers, Microsoft Corporation, 1995*, and to the OLE samples provided by Microsoft Visual Basic.

DB2 provides self-containing samples of OLE automation UDFs and stored procedures in Microsoft Visual Basic, located in the directory `sqllib\samples\ole\msvb`. For information on building and running the OLE automation UDF and stored procedure samples, please see the `README` file in `sqllib\samples\ole`.

Related concepts:

- "OLE automation routine design" in the *Application Development Guide: Programming Server Applications*
- "OLE automation routines in BASIC and C++" in the *Application Development Guide: Programming Server Applications*

Related reference:

- "Object Linking and Embedding (OLE) samples" on page 85

.NET

Building C# .NET applications

DB2 provides a batch file, bldapp.bat, for compiling and linking DB2 C# .NET applications, located in the sqllib\samples\.NET\cs directory, along with sample programs that can be built with this file. The batch file takes one parameter, %1, for the name of the source file to be compiled (without the .cs extension).

Procedure:

To build the program, `DbAuth`, from the source file, `DbAuth.cs`, enter:

```
bldapp DbAuth
```

To ensure you have the parameters you need when you run the executable, you can specify different combinations of parameters depending on the number entered:

1. No parameters. Enter just the program name:
   ```
   DbAuth
   ```
2. One parameter. Enter the program name plus the database alias:
   ```
   DbAuth <db_alias>
   ```
3. Two parameters. Enter the program name plus user ID and password:
   ```
   DbAuth <userid> <passwd>
   ```
4. Three parameters. Enter the program name plus the database alias, user ID, and password:
   ```
   DbAuth <db_alias> <userid> <passwd>
   ```
5. Four parameters. Enter the program name plus server name, port number, user ID, and password:
   ```
   DbAuth <server> <portnum> <userid> <passwd>
   ```
6. Five parameters. Enter the program name plus database alias, server name, port number, user ID, and password:
   ```
   DbAuth <db_alias> <server> <portnum> <userid> <passwd>
   ```

To build and run the LCTrans sample program, you need to follow more detailed instructions given in the source file, LCTrans.cs.

Related tasks:

- "Building Common Language Runtime (CLR) .NET routines" on page 251

Related reference:

- "C# samples" on page 70
- "C# .NET application compile and link options" on page 247

Related samples:

- "bldapp.bat -- Builds C# applications on Windows"
- "DbAuth.cs -- How to Grant, display and revoke privileges on database"
- "LCTrans.cs -- Demonstrates loosely coupled transactions (CSNET)"

Batch file for C# .NET applications

```
@echo off
rem BATCH FILE: bldapp.bat
rem Builds C# applications on Windows
rem Usage: bldapp prog_name

rem Default compiler is set to Microsoft C# Compiler
rem To use a different compiler, comment out 'set BLDCOMP=csc'
rem and write 'set BLDCOMP=x' where x is the the required compiler
set BLDCOMP=csc

rem When using the .NET Framework Version 1.0 point to netf10
rem set VERSION=netf10\
rem When using the .NET Framework Version 1.1 point to netf11
set VERSION=netf11\

if exist %1.cs goto build

:build
if "%1"=="LCTrans" goto LCTransbuild
if "%1"=="SubCOM" goto SubCOMbuild
if "%1"=="RootCOM" goto RootCOMbuild
%BLDCOMP% %1.cs /r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll
goto exit

:RootCOMbuild
%BLDCOMP% /out:RootCOM.dll /target:library %1.cs /r:System.EnterpriseServices.dll
  /r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll /r:System.Data.dll /r:System.dll
  /r:SubCOM.dll
goto exit

:SubCOMbuild
%BLDCOMP% /out:SubCOM.dll /target:library %1.cs /r:System.EnterpriseServices.dll
  /r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll /r:System.Data.dll /r:System.Xml.dll
  /r:System.dll
goto exit

:LCTransbuild
%BLDCOMP% %1.cs /r:System.EnterpriseServices.dll
  /r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll /r:System.Data.dll /r:System.dll
  /r:SubCOM.dll /r:RootCOM.dll
goto exit

:exit
@echo on
```

C# .NET application compile and link options

The following are the compile and link options recommended by DB2 for building C# applications on Windows with the Microsoft C# compiler, as demonstrated in the `bldapp.bat` batch file.

Compile and link options for bldapp
Compile and link options for standalone C# applications: **%BLDCOMP%** Variable for the compiler. The default is `csc`, the Microsoft C# compiler. **/r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll** Reference the DB2 data link library for the .NET framework version you are using. **%VERSION%** There are two supported versions of the .NET framework for applications. DB2 has a data link library for each in separate sub-directories. For .NET Framework Version 1.0, `%VERSION%` points to the `netf10\` sub-directory; For .NET Framework Version 1.1, `%VERSION%` points to the `netf11\` sub-directory.

Compile and link options for bldapp
Compile and link options for the loosely-coupled sample program, LCTrans: **%BLDCOMP%** Variable for the compiler. The default is csc, the Microsoft C# compiler. **/out:RootCOM.dll** Output the RootCOM data link library, used by the LCTrans application, from the RootCOM.cs source file, **/out:SubCOM.dll** Output the SubCOM data link library, used by the LCTrans application, from the SubCOM.cs source file, **/target:library %1.cs** Create the data link library from the input source file (RootCOM.cs or SubCOM.cs). **/r:System.EnterpriseServices.dll** Reference the Microsoft Windows System EnterpriseServices data link library. **/r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll** Reference the DB2 data link library for the .NET framework version you are using. **%VERSION%** There are two supported versions of the .NET framework for applications. DB2 has a data link library for each in separate sub-directories. For .NET Framework Version 1.0, %VERSION% points to the netf10 sub-directory; For .NET Framework Version 1.1, %VERSION% points to the netf11 sub-directory. **/r:System.Data.dll** Reference the Microsoft Windows System Data data link library. **/r:System.dll** Reference the Microsoft Windows System data link library. **/r:System.Xml.dll** Reference the Microsoft Windows System XML data link library (for SubCOM.cs). **/r:SubCOM.dll** Reference the SubCOM data link library (for RootCOM.cs and LCTrans.cs). **/r:RootCOM.dll** Reference the RootCOM data link library (for LCTrans.cs). Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building C# .NET applications" on page 245

Related samples:

- "bldapp.bat -- Builds C# applications on Windows"

Building Visual Basic .NET applications

DB2 provides a batch file, bldapp.bat, for compiling and linking DB2 Visual Basic .NET applications, located in the sqllib\samples\.NET\vb directory, along with sample programs that can be built with this file. The batch file takes one parameter, %1, for the name of the source file to be compiled (without the .vb extension).

Procedure:

To build the program, `DbAuth`, from the source file, `DbAuth.vb`, enter:

```
bldapp DbAuth
```

To ensure you have the parameters you need when you run the executable, you can specify different combinations of parameters depending on the number entered:

1. No parameters. Enter just the program name:
   ```
   DbAuth
   ```
2. One parameter. Enter the program name plus the database alias:
   ```
   DbAuth <db_alias>
   ```
3. Two parameters. Enter the program name plus user ID and password:
   ```
   DbAuth <userid> <passwd>
   ```
4. Three parameters. Enter the program name plus the database alias, user ID, and password:
   ```
   DbAuth <db_alias> <userid> <passwd>
   ```
5. Four parameters. Enter the program name plus server name, port number, user ID, and password:
   ```
   DbAuth <server> <portnum> <userid> <passwd>
   ```
6. Five parameters. Enter the program name plus database alias, server name, port number, user ID, and password:
   ```
   DbAuth <db_alias> <server> <portnum> <userid> <passwd>
   ```

To build and run the `LCTrans` sample program, you need to follow more detailed instructions given in the source file, `LCTrans.vb`.

Related tasks:

- "Building Common Language Runtime (CLR) .NET routines" on page 251

Related reference:

- "Visual Basic .NET samples" on page 94
- "Visual Basic .NET application compile and link options" on page 250

Related samples:

- "bldapp.bat -- Builds Visual Basic .Net applications on Windows"
- "DbAuth.vb -- How to Grant, display and revoke privileges on database"
- "LCTrans.vb -- Demonstrates loosely coupled transactions"

Batch file for Visual Basic .NET applications

```
@echo off
rem BATCH FILE: bldapp.bat
rem Builds Visual Basic .Net applications on Windows
rem Usage: bldapp prog_name [ db_name [ userid password ]]

rem Default compiler is set to Microsoft Visual Basic .NET Compiler
rem To use a different compiler, comment out 'set BLDCOMP=vbc'
rem and write 'set BLDCOMP=x' where x is the the required compiler
set BLDCOMP=vbc

rem When using the .NET Framework Version 1.0 point to netf10
rem set VERSION=netf10\
rem When using the .NET Framework Version 1.1 point to netf11
```

```
set VERSION=netf11\

if exist %1.vb goto build

:build
if "%1"=="LCTrans" goto LCTransbuild
if "%1"=="SubCOM" goto SubCOMbuild
if "%1"=="RootCOM" goto RootCOMbuild
%BLDCOMP% %1.vb /r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll /r:System.dll
  /r:System.Data.dll /r:System.Xml.dll
goto exit

:RootCOMbuild
%BLDCOMP% %1.vb /r:System.EnterpriseServices.dll
  /r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll /target:library /r:System.Data.dll
  /r:System.dll /r:SubCOM.dll /out:RootCOM.dll
goto exit

:SubCOMbuild
%BLDCOMP% %1.vb /r:System.EnterpriseServices.dll
  /r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll /target:library /r:System.Data.dll
  /r:System.Xml.dll /r:System.dll /out:SubCOM.dll
goto exit

:LCTransbuild
%BLDCOMP% %1.vb /r:System.EnterpriseServices.dll
  /r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll /r:System.Data.dll /r:System.dll
  /r:SubCOM.dll /r:RootCOM.dll
goto exit

:exit
@echo on
```

Visual Basic .NET application compile and link options

The following are the compile and link options recommended by DB2 for building Visual Basic .NET applications on Windows with the Microsoft Visual Basic .NET compiler, as demonstrated in the `bldapp.bat` batch file.

Compile and link options for bldapp
Compile and link options for standalone VB .NET applications: **%BLDCOMP%** Variable for the compiler. The default is `vbc`, the Microsoft Visual Basic .NET compiler. **/r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll** Reference the DB2 data link library for the .NET framework version you are using. **%VERSION%** There are two supported versions of the .NET framework for applications. DB2 has a data link library for each in separate sub-directories. For .NET Framework Version 1.0, `%VERSION%` points to the `netf10\` sub-directory; For .NET Framework Version 1.1, `%VERSION%` points to the `netf11\` sub-directory.

Compile and link options for bldapp
Compile and link options for the loosely-coupled sample program, LCTrans: **%BLDCOMP%** Variable for the compiler. The default is vbc, the Microsoft Visual Basic .NET compiler. **/out:RootCOM.dll** Output the RootCOM data link library, used by the LCTrans application, from the RootCOM.vb source file, **/out:SubCOM.dll** Output the SubCOM data link library, used by the LCTrans application, from the SubCOM.vb source file, **/target:library %1.cs** Create the data link library from the input source file (RootCOM.vb or SubCOM.vb). **/r:System.EnterpriseServices.dll** Reference the Microsoft Windows System EnterpriseServices data link library. **/r:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll** Reference the DB2 data link library for the .NET framework version you are using. **%VERSION%** There are two supported versions of the .NET framework for applications. DB2 has a data link library for each in separate sub-directories. For .NET Framework Version 1.0, %VERSION% points to the netf10 sub-directory; For .NET Framework Version 1.1, %VERSION% points to the netf11 sub-directory. **/r:System.Data.dll** Reference the Microsoft Windows System Data data link library. **/r:System.dll** Reference the Microsoft Windows System data link library. **/r:System.Xml.dll** Reference the Microsoft Windows System XML data link library (for SubCOM.vb). **/r:SubCOM.dll** Reference the SubCOM data link library (for RootCOM.vb and LCTrans.vb). **/r:RootCOM.dll** Reference the RootCOM data link library (for LCTrans.vb). Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building Visual Basic .NET applications" on page 248

Related samples:

- "bldapp.bat -- Builds Visual Basic .Net applications on Windows"

Building Common Language Runtime (CLR) .NET routines

DB2 provides batch files for compiling and linking DB2 .NET programs. These are located in the sqllib\samples\.NET\cs and sqllib\samples\.NET\vb directories, along with sample programs that can be built with these files.

The batch file `bldrtn.bat` contains the commands to build CLR routines (stored procedures and user-defined functions). The batch file builds a .NET assembly DLL on the server. It takes two parameters, represented inside the batch file by the variables `%1` and `%2`.

The first parameter, `%1`, specifies the name of your source file. The batch file uses the source file name for the assembly DLL name. The second parameter, `%2`, specifies the name of the database to which you want to connect. Since the assembly DLL must be built on the same instance where the database resides, there are no parameters for user ID and password.

Only the first parameter, the source file name, is required. Database name is optional. If no database name is supplied, the program uses the default `sample` database.

Prerequisites:

The database server must be running a Windows operating system installed with Microsoft .NET Framework Version 1.1 (both .NET Framework 1.0 and .NET Framework 1.1 are supported for client applications). The .NET Framework is independently available or as part of the Microsoft .NET Framework 1.1 Software Development Kit.

The following versions of DB2 must be installed:

Server:
DB2 8.2 or later

Client:
DB2 7.2 or later

Authority must be granted to execute the CREATE statement for the routine. For the privileges required to execute the CREATE statement, see the CREATE statement for the routine type: CREATE PROCEDURE or CREATE FUNCTION.

Procedure:

The following examples show you how to build routine assembly DLLs with stored procedures and user-defined functions.

Stored procedure assembly DLL

To build the `SpServer` assembly DLL from the VB .NET source file, `SpServer.vb`, or the C# source file, `SpServer.cs`:

1. Enter the batch file name and program name (without the extension):
   ```
   bldrtn SpServer
   ```
   If connecting to a different database than the default `sample` database, also enter the database name:
   ```
   bldrtn SpServer database
   ```
   The batch file copies the assembly DLL, `SpServer.dll`, to the `sqllib\function` directory.
2. Next, catalog the routines by running the `spcat` script on the server:
   ```
   SpCat
   ```

This script connects to the sample database, uncatalogs the routines if they were previously cataloged by calling SpDrop.db2, then catalogs them by calling SpCreate.db2, and finally disconnects from the database. You can also call the SpDrop.db2 and SpCreate.db2 scripts individually.

3. Then, unless this is the first time the assembly DLL was built, stop and restart the database to allow the new version of the assembly DLL to be recognized. If necessary, set the file mode for the assembly DLL so the DB2 instance can access it.

Once you build the assembly DLL, SpServer, you can build the client application SpClient that calls it.

You can build SpClient by using the batch file, bldapp.bat.

To ensure you have the parameters you need when you run the executable, you can specify different combinations of parameters, instead of accepting the defaults, depending on the number of parameters entered:

1. No parameters. Enter just the program name (for calling locally on the server instance):
   ```
   SpClient
   ```
2. One parameter. Enter the program name plus the database alias (for calling a different database than the sample database locally on the server instance):
   ```
   SpClient <db_alias>
   ```
3. Three parameters. Enter the program name plus the database alias, user ID, and password (for calling from a remote client):
   ```
   SpClient <db_alias> <userid> <passwd>
   ```
4. Five parameters. Enter the program name plus database alias, server name, port number, user ID, and password (for calling from a remote client):
   ```
   SpClient <db_alias> <server> <portnum> <userid> <passwd>
   ```

The client application accesses the assembly DLL, SpServer, and executes a number of routines on the server database. The output is returned to the client application.

User-defined function assembly DLL

To build the user-defined function assembly DLL UDFsrv from the VB .NET source file UDFsrv.vb, or the C# source file, UDFsrv.cs, enter:

```
bldrtn UDFsrv
```

If connecting to a different database than the default sample database, also enter the database name:

```
bldrtn UDFsrv database
```

The batch file copies the user-defined function assembly DLL, UDFsrv.dll, to the sqllib\function directory.

Once you build UDFsrv, you can build the client application, udfcli, that calls it.

You can build UDFcli by using the batch file, bldapp.bat.

To ensure you have the parameters you need when you run the executable, you can specify different combinations of parameters, instead of accepting the defaults, depending on the number of parameters entered:

1. No parameters. Enter just the program name (for calling locally on the server instance):
   ```
   UDFcli
   ```
2. One parameter. Enter the program name plus the database alias (for calling a different database than the `sample` database locally on the server instance):
   ```
   UDFcli <db_alias>
   ```
3. Three parameters. Enter the program name plus the database alias, user ID, and password (for calling from a remote client):
   ```
   UDFcli <db_alias> <userid> <passwd>
   ```
4. Five parameters. Enter the program name plus database alias, server name, port number, user ID, and password (for calling from a remote client):
   ```
   UDFcli <db_alias> <server> <portnum> <userid> <passwd>
   ```

The calling application calls the `ScalarUDF` function from the `udfsrv` assembly DLL.

Related concepts:

- "Common language runtime (CLR) routines" in the *Application Development Guide: Programming Server Applications*

Related samples:

- "bldrtn.bat -- Builds C# routines (stored procedures and UDFs)"
- "SpServer.cs -- C# external code implementation of procedures created in spcat.db2"
- "SpClient.cs -- Call different types of stored procedures from SpServer.java"
- "UDFcli.cs -- Client application that calls the user-defined functions "
- "UDFsrv.cs -- User-defined scalar functions called by udfcli.cs"
- "bldrtn.bat -- Builds Visual Basic .NET routines (stored procedures and UDFs)"
- "SpServer.vb -- VB.NET implementation of procedures created in SpCat.db2"
- "SpClient.vb -- Call different types of stored procedures from SpServer.java"
- "UDFcli.vb -- Client application that calls the user-defined functions "
- "UDFsrv.vb -- User-defined scalar functions called by udfcli.vb "

Batch file for C# .NET routines

```
@echo off
rem BATCH FILE: bldrtn.bat
rem Builds C# routines (stored procedures and UDFs)
rem Usage: bldrtn prog_name

rem When using the .NET Framework Version 1.1 point to netf11
set VERSION=netf11\

rem Compile the program.
csc /out:%1.dll /target:library /debug /lib:"%DB2PATH%"\bin\netf11\
   /reference:"%DB2PATH%"\bin\%VERSION%IBM.Data.DB2.dll %1.cs

if exist "%DB2PATH%\function\%1.dll" goto delete else goto copydll

:delete
del "%DB2PATH%\function\%1.dll"
goto copydll

:copydll
```

```
rem Copy the routine assembly data link library to the 'function' directory
copy "%1.dll" "%DB2PATH%\function"

@echo on
```

Batch file for Visual Basic .NET routines

```
@echo off
rem BATCH FILE: bldrtn.bat
rem Builds Visual Basic .NET routines (stored procedures and UDFs)
rem Usage: bldrtn prog_name

rem Compile the program.
vbc %1.vb /out:%1.dll /target:library /debug /libpath:"%DB2PATH%"\bin\netf11\
  /reference:IBM.Data.DB2.dll /reference:System.dll /reference:System.Data.dll

if exist "%DB2PATH%\function\%1.dll" goto delete else goto copydll

:delete
del "%DB2PATH%\function\%1.dll"
goto copydll

:copydll
rem Copy the routine assembly data link library to the 'function' directory
copy "%1.dll" "%DB2PATH%\function"

@echo on
```

CLR .NET routine compile and link options

The following are the compile and link options recommended by DB2 for building Common Language Runtime (CLR) .NET routines on Windows with either the Microsoft Visual Basic .NET compiler or the Microsoft C# compiler, as demonstrated in the `samples\.NET\cs\bldrtn.bat` and `samples\.NET\vb\bldrtn.bat` batch files.

Compile and link options for bldrtn
Compile and link options using the Microsoft C# compiler:
csc The Microsoft C# compiler.
/out:%1.dll /target:library Output the data link library as a stored procedure assembly dll.
/debug Use the debugger.
/lib: "%DB2PATH%"\bin\netf11 Use the library path for .NET Framework Version 1.1.
/reference:IBM.Data.DB2.dll Use the DB2 data link library for .NET Framework Version 1.1.
Refer to your compiler documentation for additional compiler options.

Compile and link options for bldrtn
Compile and link options using the Microsoft Visual Basic .NET compiler:
vbc The Microsoft Visual Basic .NET compiler.
/out:%1.dll /target:library Output the data link library as a stored procedure assembly dll.
/debug Use the debugger.
/libpath:"%DB2PATH%"\bin\netf11 Use the library path for .NET Framework Version 1.1.
/reference:IBM.Data.DB2.dll Use the DB2 data link library for .NET Framework Version 1.1.
/reference:System.dll Reference the Microsoft Windows System data link library.
/reference:System.Data.dll Reference the Microsoft Windows System Data data link library.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building Common Language Runtime (CLR) .NET routines" on page 251

Related samples:

- "bldrtn.bat -- Builds C# routines (stored procedures and UDFs)"
- "bldrtn.bat -- Builds Visual Basic .NET routines (stored procedures and UDFs)"

Microsoft Visual C++

This section discusses building applications with ActiveX Data Objects (ADO), Object Linking and Embedding (OLE), as well as embedded SQL and DB2 APIs.

Building information for DB2 CLI applications and routines is in the *CLI Guide and Reference*.

Building ADO applications with Visual C++

ActiveX Data Objects (ADO) allow you to write an application to access and manipulate data in a database server through an OLE DB provider. The primary benefits of ADO are high speed, ease of use, low memory overhead, and a small disk footprint.

DB2 provides Visual C++ ADO sample programs in the `sqllib\samples\VC` directory.

Procedure:

You can use either of two ODBC-compliant providers:

- IBM OLE DB provider for DB2
- Microsoft OLE DB provider for ODBC

Using the IBM OLE DB provider for DB2

DB2 Version 8.2 Clients on Windows operating systems will optionally install IBMDADB2, the IBM OLE DB 2.0-compliant provider for DB2. The provider exposes interfaces for consumers who want to access data in a DB2 database. The IBM OLE DB provider for DB2 supports the following ADO application types:

- Microsoft Active Server Pages (ASP)
- Microsoft Visual Studio C++ and Visual Basic applications
- Microsoft Visual Interdev

For details on these types of applications, refer to the ADO documentation.

Using the Microsoft OLE DB provider for ODBC

DB2 ADO programs using the Microsoft OLE DB provider and Visual C++ can be compiled the same as regular C++ programs, once you make the following change.

To have your C++ source program run as an ADO program, you can put the following import statement at the top of your source program file:

```
#import "C:\program files\common files\system\ado\msado<VERSION NUMBER>.dll" \
 no_namespace \
 rename( "EOF", "adoEOF")
```

where <VERSION NUMBER> is the version number of the ADO library.

When the program is compiled, the user will need to verify that the msado<VERSION NUMBER>.dll is in the path specified. An alternative is to add C:\program files\common files\system\ado to the environment variable LIBPATH, and then use this shorter import statement in your source file:

```
#import <msado<VERSION NUMBER>.dll> \
 no_namespace \
 rename( "EOF", "adoEOF")
```

This is the method used in the DB2 sample program, BLOBAccess.dsp.

With this IMPORT statement, your DB2 program will have access to the ADO library. You can now compile your Visual C++ program as you would any other program. If you are also using another programming interface, such as DB2 APIs, or DB2 CLI, refer to the appropriate topic for additional information on building your program.

Related concepts:

- "Purpose of the IBM OLE DB Provider for DB2" in the *Application Development Guide: Programming Client Applications*
- "Application Types Supported by the IBM OLE DB Provider for DB2" in the *Application Development Guide: Programming Client Applications*
- "Compilation and Linking of C/C++ Applications and the IBM OLE DB Provider" in the *Application Development Guide: Programming Client Applications*
- "Connections to Data Sources in C/C++ Applications using the IBM OLE DB Provider" in the *Application Development Guide: Programming Client Applications*
- "OLE DB Services Automatically Enabled by IBM OLE DB Provider" in the *Application Development Guide: Programming Client Applications*
- "Large Object Manipulation with the IBM OLE DB Provider" in the *Application Development Guide: Programming Client Applications*
- "IBM OLE DB Provider Restrictions" in the *Application Development Guide: Programming Client Applications*

Related reference:

- "Data Type Mappings between DB2 and OLE DB" in the *Application Development Guide: Programming Client Applications*
- "Data Conversion for Setting Data from OLE DB Types to DB2 Types" in the *Application Development Guide: Programming Client Applications*
- "Data Conversion for Setting Data from DB2 Types to OLE DB Types" in the *Application Development Guide: Programming Client Applications*
- "IBM OLE DB Provider Support for OLE DB Components and Interfaces" in the *Application Development Guide: Programming Client Applications*
- "IBM OLE DB Provider support for OLE DB properties" in the *Application Development Guide: Programming Client Applications*
- "IBM OLE DB Provider Support for ADO Methods and Properties" in the *Application Development Guide: Programming Client Applications*
- "Visual C++ samples" on page 96

Object Linking and Embedding (OLE) automation with Visual C++

You can implement OLE automation UDFs and stored procedures in any language, as OLE is language independent. You do this by exposing methods of OLE automation servers, and registering the methods as UDFs with DB2®. Application development environments which support the development of OLE automation servers include certain versions of the following: Microsoft® Visual Basic, Microsoft Visual C++, Microsoft Visual J++, Microsoft FoxPro, Borland Delphi, Powersoft PowerBuilder, and Micro Focus COBOL. Also, Java™ beans objects that are wrapped properly for OLE, for example with Microsoft Visual J++, can be accessed via OLE automation.

You need to refer to the documentation of the appropriate application development environment for further information on developing OLE automation servers.

OLE automation UDFs and stored procedures

Microsoft Visual C++ supports the creation of OLE automation servers. Servers can be implemented using Microsoft Foundation Classes and the Microsoft Foundation Class application wizard, or implemented as Win32 applications. Servers can be DLLs or EXEs. Refer to the Microsoft Visual C++ documentation and to the OLE samples provided by Microsoft Visual C++ for further information.

DB2 provides self-containing samples of OLE automation UDFs and stored procedures in Microsoft Visual C++, located in the directory `sqllib\samples\ole\msvc`. For information on building and running the OLE automation UDF and stored procedure samples, please see the `README` file in `sqllib\samples\ole`.

Related concepts:

- "OLE automation routine design" in the *Application Development Guide: Programming Server Applications*
- "OLE automation routines in BASIC and C++" in the *Application Development Guide: Programming Server Applications*

Related reference:

- "Object Linking and Embedding (OLE) samples" on page 85

Building C/C++ applications on Windows

DB2 provides batch files for compiling and linking DB2 API and embedded SQL C/C++ programs. These are located in the `sqllib\samples\c` and `sqllib\samples\cpp` directories, along with sample programs that can be built with these files.

The batch file, `bldapp.bat`, contains the commands to build DB2 API and embedded SQL programs. It takes up to four parameters, represented inside the batch file by the variables `%1`, `%2`, `%3`, and `%4`.

The first parameter, `%1`, specifies the name of your source file. This is the only required parameter for programs that do not contain embedded SQL. Building embedded SQL programs requires a connection to the database so three additional parameters are also provided: the second parameter, `%2`, specifies the name of the database to which you want to connect; the third parameter, `%3`, specifies the user ID for the database, and `%4` specifies the password.

For an embedded SQL program, `bldapp` passes the parameters to the precompile and bind file, `embprep.bat`. If no database name is supplied, the default `sample` database is used. The user ID and password parameters are only needed if the instance where the program is built is different from the instance where the database is located.

Procedure:

The following examples show you how to build and run DB2 API and embedded SQL applications.

To build the DB2 API non-embedded SQL sample program, `cli_info`, from either the source file `cli_info.c`, in `sqllib\samples\c`, or from the source file `cli_info.cxx`, in `sqllib\samples\cpp`, enter:

```
bldapp cli_info
```

The result is an executable file, `cli_info.exe`. You can run the executable file by entering the executable name (without the extension) on the command line:

```
cli_info
```

Building and running embedded SQL applications

There are three ways to build the embedded SQL application, `tbmod`, from the C source file `tbmod.sqc` in `sqllib\samples\c`, or from the C++ source file `tbmod.sqx` in `sqllib\samples\cpp`:

1. If connecting to the sample database on the same instance, enter:
   ```
   bldapp tbmod
   ```
2. If connecting to another database on the same instance, also enter the database name:
   ```
   bldapp tbmod database
   ```
3. If connecting to a database on another instance, also enter the user ID and password of the database instance:
   ```
   bldapp tbmod database userid password
   ```

The result is an executable file `tbmod.exe`.

There are three ways to run this embedded SQL application:

1. If accessing the `sample` database on the same instance, simply enter the executable name:
   ```
   tbmod
   ```
2. If accessing another database on the same instance, enter the executable name and the database name:
   ```
   tbmod database
   ```
3. If accessing a database on another instance, enter the executable name, database name, and user ID and password of the database instance:
   ```
   tbmod database userid password
   ```

Building and running multi-threaded applications

C/C++ multi-threaded applications on Windows need to be compiled with either the `-MT` or `-MD` options. The `-MT` option will link using the static library `LIBCMT.LIB`, and `-MD` will link using the dynamic library `MSVCRT.LIB`. The binary linked with `-MD` will be smaller but dependent on `MSVCRT.DLL`, while the binary linked with `-MT` will be larger but will be self-contained with respect to the runtime.

The batch file `bldmt.bat` uses the `-MT` option to build a multi-threaded program. All other compile and link options are the same as those used by the batch file `bldapp.bat` to build regular standalone applications.

To build the multi-threaded sample program, `dbthrds`, from either the `samples\c\dbthrds.sqc` or `samples\cpp\dbthrds.sqx` source file, enter:
```
bldmt dbthrds
```

The result is an executable file, `dbthrds.exe`.

There are three ways to run this multi-threaded application:

1. If accessing the `sample` database on the same instance, simply enter the executable name (without the extension):
   ```
   dbthrds
   ```
2. If accessing another database on the same instance, enter the executable name and the database name:
   ```
   dbthrds database
   ```
3. If accessing a database on another instance, enter the executable name, database name, and user ID and password of the database instance:
   ```
   dbthrds database userid password
   ```

Related reference:

- "Windows C/C++ application compile and link options" on page 262

Related samples:

- "bldapp.bat -- Builds C applications on Windows"
- "bldmt.bat -- Builds C multi-threaded applications on Windows"
- "cli_info.c -- Set and get information at the client level (C)"
- "dbthrds.sqc -- How to use multiple context APIs on Windows (C)"
- "embprep.bat -- Prep and binds a C/C++ or Micro Focus COBOL embedded SQL program on Windows"
- "tbmod.sqc -- How to modify table data (C)"

- "bldapp.bat -- Builds C++ applications on Windows"
- "bldmt.bat -- Builds C++ multi-threaded applications on Windows"
- "cli_info.C -- Set and get information at the client level (C++)"
- "dbthrds.sqC -- How to use multiple context APIs on Windows (C++)"
- "tbmod.sqC -- How to modify table data (C++)"

Batch file for C/C++ applications

```
@echo off
rem BATCH FILE: bldapp.bat
rem Builds C/C++ applications on Windows
rem Usage: bldapp prog_name [ db_name [ userid password ]]

rem Default compiler is set to Microsoft Visual C++
rem To use a different compiler, comment out 'set BLDCOMP=cl'
rem and uncomment 'set BLDCOMP=icl' or 'set BLDCOMP=ecl'
rem Microsoft C/C++ Compiler
set BLDCOMP=cl

rem Intel C++ Compiler for 32-bit applications
rem set BLDCOMP=icl

rem Intel C++ Compiler for Itanium 64-bit applications
rem set BLDCOMP=ecl

if exist "%1.sqx" goto embedded
if exist "%1.sqc" goto embedded
goto non_embedded

:embedded
rem Precompile and bind the program.
rem Note: some .sqc/.sqx files contain no SQL but link in
rem utilemb.sqc/.sqx, so if you get this warning, ignore it:
rem SQL0053W  No SQL statements were found in the program.
call embprep %1 %2 %3 %4
rem  Compile the program.
if exist "%1.cxx" goto cpp_emb
%BLDCOMP% -Zi -Od -c -W2 -DWIN32 %1.c utilemb.c
goto link_embedded
:cpp_emb
%BLDCOMP% -Zi -Od -c -W2 -DWIN32 %1.cxx utilemb.cxx
rem Link the program.
:link_embedded
link -debug -out:%1.exe %1.obj utilemb.obj db2api.lib
goto exit

:non_embedded
rem  Compile the program.
if exist "%1.cxx" goto cpp_non
%BLDCOMP% -Zi -Od -c -W2 -DWIN32 %1.c utilapi.c
goto link_non_embedded
:cpp_non
%BLDCOMP% -Zi -Od -c -W2 -DWIN32 %1.cxx utilapi.cxx
rem Link the program.
:link_non_embedded
link -debug -out:%1.exe %1.obj utilapi.obj db2api.lib
:exit
@echo on
```

Windows C/C++ application compile and link options

The following are the compile and link options recommended by DB2 for building C/C++ embedded SQL and DB2 API applications on Windows with the Microsoft Visual C++ compiler, as demonstrated in the `bldapp.bat` batch file.

Compile and link options for bldapp
Compile options:
%BLDCOMP% Variable for the compiler. The default is `cl`, the Microsoft Visual C++ compiler. It can be also set to `icl`, the Intel C++ Compiler for 32-bit applications, or `ecl`, the Intel C++ Compiler for Itanium 64-bit applications.
-Zi Enable debugging information
-Od Disable optimizations. It is easier to use a debugger with optimization off.
-c Perform compile only; no link. The batch file has separate compile and link steps.
-W2 Output warning, error, and severe and unrecoverable error messages.
-DWIN32 Compiler option necessary for Windows operating systems.
Link options:
link Use the linker to link.
-debug Include debugging information.
-out:%1.exe Specify a filename
%1.obj Include the object file
utilemb.obj If an embedded SQL program, include the embedded SQL utility object file for error checking.
utilapi.obj If not an embedded SQL program, include the DB2 API utility object file for error checking.
db2api.lib Link with the DB2 library.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building C/C++ applications on Windows" on page 259

Related samples:

- "bldapp.bat -- Builds C applications on Windows"
- "bldapp.bat -- Builds C++ applications on Windows"

Building C/C++ routines on Windows

DB2 provides batch files for compiling and linking DB2 API and embedded SQL programs in C and C++. These are located in the `sqllib\samples\c` and `sqllib\samples\cpp` directories, along with sample programs that can be built with these files.

The batch file bldrtn.bat contains the commands to build embedded SQL routines (stored procedures and user-defined functions). The batch file builds a DLL on the server. It takes two parameters, represented inside the batch file by the variables %1 and %2.

The first parameter, %1, specifies the name of your source file. The batch file uses the source file name for the DLL name. The second parameter, %2, specifies the name of the database to which you want to connect. Since the DLL must be built on the same instance where the database resides, there are no parameters for user ID and password.

Only the first parameter, the source file name, is required. Database name is optional. If no database name is supplied, the program uses the default sample database.

Procedure:

The following examples show you how to build routine DLLs with:

- stored procedures
- non-embedded SQL user-defined functions (UDFs)
- embedded SQL user-defined functions (UDFs)

Stored procedure DLL

To build the spserver DLL from either the C source file, spserver.sqc, or the C++ source file, spserver.sqx:

1. Enter the batch file name and program name:

   ```
   bldrtn spserver
   ```

   If connecting to another database, also enter the database name:

   ```
   bldrtn spserver database
   ```

   The batch file uses the module definition file spserver.def, contained in the same directory as the sample programs, to build the DLL. The batch file copies the DLL, spserver.dll, to the server in the path sqllib\function.
2. Next, catalog the routines by running the spcat script on the server:

   ```
   spcat
   ```

   This script connects to the sample database, uncatalogs the routines if they were previously cataloged by calling spdrop.db2, then catalogs them by calling spcreate.db2, and finally disconnects from the database. You can also call the spdrop.db2 and spcreate.db2 scripts individually.
3. Then, stop and restart the database to allow the new DLL to be recognized. If necessary, set the file mode for the DLL so the DB2 instance can access it.

Once you build the DLL, spserver, you can build the client application spclient that calls it.

You can build spclient by using the batch file, bldapp.bat.

To call the DLL, run the sample client application by entering:

spclient *database userid password*

where

database
Is the name of the database to which you want to connect. The name could be `sample`, or its alias, or another database name.

userid Is a valid user ID.

password
Is a valid password for the user ID.

The client application accesses the DLL, `spserver`, and executes a number of routines on the server database. The output is returned to the client application.

Non-embedded SQL UDF DLL

To build the user-defined function `udfsrv` from the source file `udfsrv.c`, enter:

```
bldrtn udfsrv
```

The batch file uses the module definition file, `udfsrv.def`, contained in the same directory as the sample program files, to build the user-defined function DLL. The batch file copies the user-defined function DLL, `udfsrv.dll`, to the server in the path `sqllib\function`.

Once you build `udfsrv`, you can build the client application, `udfcli`, that calls it. DB2 CLI, as well as embedded SQL C and C++ versions of this program are provided.

You can build the DB2 CLI `udfcli` program from the `udfcli.c` source file in `sqllib\samples\cli` using the batch file `bldapp`.

You can build the embedded SQL C `udfcli` program from the `udfcli.sqc` source file in `sqllib\samples\c` using the batch file `bldapp`.

You can build the embedded SQL C++ `udfcli` program from the `udfcli.sqx` source file in `sqllib\samples\cpp` using the batch file `bldapp`.

To run the UDF, enter:

```
udfcli
```

The calling application calls the `ScalarUDF` function from the `udfsrv` DLL.

Embedded SQL UDF DLL

To build the embedded SQL user-defined function library `udfemsrv` from the C source file `udfemsrv.sqc` in `sqllib\samples\c`, or from the C++ source file `udfemsrv.sqx` in `sqllib\samples\cpp`, enter:

```
bldrtn udfemsrv
```

If connecting to another database, also enter the database name:

```
bldrtn udfemsrv database
```

The batch file uses the module definition file, `udfemsrv.def`, contained in the same directory as the sample programs, to build the user-defined function DLL. The batch file copies the user-defined function DLL, `udfemsrv.dll`, to the server in the path `sqllib\function`.

Once you build `udfemsrv`, you can build the client application, `udfemcli`, that calls it. You can build `udfemcli` from the C source file `udfemcli.sqc` in `sqllib\samples\c`, or from the C++ source file `udfemcli.sqx` in `sqllib\samples\cpp` using the batch file `bldapp`.

To run the UDF, enter:

```
udfemcli
```

The calling application calls the UDFs in the `udfemsrv` DLL.

Related reference:

- "Windows C/C++ routine compile and link options" on page 266

Related samples:

- "bldrtn.bat -- Builds C routines (stored procedures and UDFs) on Windows"
- "embprep.bat -- Prep and binds a C/C++ or Micro Focus COBOL embedded SQL program on Windows"
- "spclient.sqc -- Call various stored procedures (C)"
- "spserver.sqc -- Definition of various types of stored procedures (C)"
- "udfcli.sqc -- Call a variety of types of user-defined functions (C)"
- "udfemcli.sqc -- Call a variety of types of embedded SQL user-defined functions. (C)"
- "udfemsrv.sqc -- Call a variety of types of embedded SQL user-defined functions. (C)"
- "udfsrv.c -- Defines a variety of types of user-defined functions (C)"
- "bldrtn.bat -- Builds C++ routines (stored procedures and UDFs) on Windows"
- "spclient.sqC -- Call various stored procedures (C++)"
- "spserver.sqC -- Definition of various types of stored procedures (C++)"
- "udfcli.sqC -- Call a variety of types of user-defined functions (C++)"
- "udfemcli.sqC -- Call a variety of types of embedded SQL user-defined functions. (C++)"
- "udfemsrv.sqC -- Call a variety of types of embedded SQL user-defined functions. (C++)"
- "udfsrv.C -- Defines a variety of types of user-defined functions (C++)"

Batch file for C/C++ routines

```
@echo off
rem BATCH FILE: bldrtn.bat
rem Builds C/C++ routines (stored procedures and UDFs) on Windows
rem Usage: bldrtn prog_name [ db_name ]

rem Default compiler is set to Microsoft Visual C++
rem To use a different compiler, comment out 'set BLDCOMP=cl'
rem and uncomment 'set BLDCOMP=icl' or 'set BLDCOMP=ecl'
rem Microsoft C/C++ Compiler
set BLDCOMP=cl

rem Intel C++ Compiler for 32-bit applications
rem set BLDCOMP=icl

rem Intel C++ Compiler for Itanium 64-bit applications
rem set BLDCOMP=ecl

if exist "%1.sqc" goto embedded
```

```
if exist "%1.sqx" goto embedded
goto compile

:embedded
rem Precompile and bind the program.
call embprep %1 %2

:compile
rem Compile the program.
if exist "%1.cxx" goto cpp
%BLDCOMP% -Zi -Od -c -W2 -DWIN32 -MD %1.c
goto link_step
:cpp
%BLDCOMP% -Zi -Od -c -W2 -DWIN32 -MD %1.cxx

:link_step
rem Link the program.
link -debug -out:%1.dll -dll %1.obj db2api.lib -def:%1.def

rem Copy the routine DLL to the 'function' directory
copy %1.dll "%DB2PATH%\function"
@echo on
```

Windows C/C++ routine compile and link options

The following are the compile and link options recommended by DB2 for building C/C++ routines (stored procedures and user-defined functions) on Windows with the Microsoft Visual C++ compiler, as demonstrated in the `bldrtn.bat` batch file.

Compile and link options for bldrtn
Compile options: **%BLDCOMP%** Variable for the compiler. The default is `cl`, the Microsoft Visual C++ compiler. It can be also set to `icl`, the Intel C++ Compiler for 32-bit applications, or `ecl`, the Intel C++ Compiler for Itanium 64-bit applications. **-Zi** Enable debugging information **-Od** Disable optimization. **-c** Perform compile only; no link. Compile and link are separate steps. **-W2** Output warning, error, and severe and unrecoverable error messages. **-DWIN32** Compiler option necessary for Windows operating systems. **-MD** Create a multithreaded DLL, using MSVCRT.LIB
Link options: **link** Use the linker to link. **-debug** Include debugging information. **-out:%1.dll** Build a .DLL file. **%1.obj** Include the object file. **db2api.lib** Link with the DB2 library. **-def:%1.def** Module definition file. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building C/C++ routines on Windows" on page 262

Related samples:

- "bldrtn.bat -- Builds C routines (stored procedures and UDFs) on Windows"
- "bldrtn.bat -- Builds C++ routines (stored procedures and UDFs) on Windows"

Building C/C++ multi-connection applications on Windows

DB2 provides batch files for compiling and linking C and C++ embedded SQL and DB2 API programs. These are located in the `sqllib\samples\c` and `sqllib\samples\cpp` directories, along with sample programs that can be built with these files.

The batch file, `bldmc.bat`, contains the commands to build a DB2 multi-connection program, requiring two databases. The compile and link options are the same as those used in the `bldapp.bat` file.

The first parameter, `%1`, specifies the name of your source file. The second parameter, `%2`, specifies the name of the first database to which you want to connect. The third parameter, `%3`, specifies the second database to which you want to connect. These are all required parameters.

Note: The makefile hardcodes default values of "sample" and "sample2" for the database names (`%2` and `%3`, respectively) so if you are using the makefile, and accept these defaults, you only have to specify the program name (the `%1` parameter). If you are using the `bldmc.bat` script, you must specify all three parameters.

Optional parameters are not required for a local connection, but are required for connecting to a server from a remote client. These are: `%4` and `%5` to specify the user ID and password, respectively, for the first database; and `%6` and `%7` to specify the user ID and password, respectively, for the second database.

Procedure:

For the multi-connection sample program, `dbmcon.exe`, you require two databases. If the `sample` database is not yet created, you can create it by entering `db2sampl` on the command line of a DB2 command window. The second database, here called `sample2`, can be created with one of the following commands:

If creating the database locally:

```
db2 create db sample2
```

If creating the database remotely:

```
db2 attach to <node_name>
db2 create db sample2
db2 detach
db2 catalog db sample2 as sample2 at node <node_name>
```

where `<node_name>` is the node where the database resides.

Multi-connection also requires that the TCP/IP listener is running. To ensure it is, do the following:

1. Set the environment variable DB2COMM to TCP/IP as follows:
   ```
   db2set DB2COMM=TCPIP
   ```
2. Update the database manager configuration file with the TCP/IP service name as specified in the services file:
   ```
   db2 update dbm cfg using SVCENAME <TCP/IP service name>
   ```
   Each instance has a TCP/IP service name listed in the services file. Ask your system administrator if you cannot locate it or do not have the file permission to change the services file.
3. Stop and restart the database manager in order for these changes to take effect:
   ```
   db2stop
   db2start
   ```

The `dbmcon.exe` program is created from five files in either the `samples\c` or `samples\cpp` directories:

`dbmcon.sqc` or `dbmcon.sqx`
Main source file for connecting to both databases.

`dbmcon1.sqc` or `dbmcon1.sqx`
Source file for creating a package bound to the first database.

`dbmcon1.h`
Header file for `dbmcon1.sqc` or `dbmcon1.sqx` included in the main source file, `dbmcon.sqc` or `dbmcon.sqx`, for accessing the SQL statements for creating and dropping a table bound to the first database.

`dbmcon2.sqc` or `dbmcon2.sqx`
Source file for creating a package bound to the second database.

`dbmcon2.h`
Header file for `dbmcon2.sqc` or `dbmcon2.sqx` included in the main source file, `dbmcon.sqc` or `dbmcon.sqx`, for accessing the SQL statements for creating and dropping a table bound to the second database.

To build the multi-connection sample program, `dbmcon.exe`, enter:

```
bldmc dbmcon sample sample2
```

The result is an executable file, `dbmcon.exe`.

To run the executable file, enter the executable name, without the extension:

```
dbmcon
```

The program demonstrates a one-phase commit to two databases.

Related concepts:

- "Build files" on page 97

Related reference:

- "svcename - TCP/IP service name configuration parameter" in the *Administration Guide: Performance*
- "Windows C/C++ application compile and link options" on page 262

Related samples:

- "bldmc.bat -- Builds C multi-connection application on Windows"
- "dbmcon.sqc -- How to use multiple databases (C)"
- "dbmcon1.h -- Function declarations for the source file, dbmcon1.sqc (C)"
- "dbmcon1.sqc -- Functions used in the multiple databases program dbmcon.sqc (C)"
- "dbmcon2.h -- Function declarations for the source file, dbmcon2.sqc (C)"
- "dbmcon2.sqc -- Functions used in the multiple databases program dbmcon.sqc (C)"
- "bldmc.bat -- Builds C++ multi-connection application on Windows"
- "dbmcon.sqC -- How to use multiple databases (C++)"
- "dbmcon1.h -- Class declaration for the source file, dbmcon1.sqC (C++)"
- "dbmcon1.sqC -- Functions used in the multiple databases program dbmcon.sqC (C++)"
- "dbmcon2.h -- Class declaration for the source file, dbmcon2.sqC (C++)"
- "dbmcon2.sqC -- Functions used in the multiple databases program dbmcon.sqC (C++)"

IBM VisualAge COBOL

Configuring the IBM COBOL compiler on Windows

If you develop applications that contain embedded SQL and DB2 API calls, and you are using the IBM VisualAge COBOL compiler, there are several points to keep in mind.

Procedure:

- When you precompile your application with the DB2 precompiler, and use the command line processor command `db2 prep`, use the `target ibmcob` option.
- Do not use tab characters in your source files.
- Use the `PROCESS` and `CBL` keywords in your source files to set compile options. Place the keywords in columns 8 to 72 only.
- If your application contains only embedded SQL, but no DB2 API calls, you do not need to use the `pgmname(mixed)` compile option. If you use DB2 API calls, you must use the `pgmname(mixed)` compile option.
- If you are using the "System/390 host data type support" feature of the IBM VisualAge COBOL compiler, the DB2 include files for your applications are in the following directory:

  ```
  %DB2PATH%\include\cobol_i
  ```

  If you are building DB2 sample programs using the batch files provided, the include file path specified in the batch files must be changed to point to the `cobol_i` directory and not the `cobol_a` directory.

  If you are NOT using the "System/390 host data type support" feature of the IBM VisualAge COBOL compiler, or you are using an earlier version of this compiler, then the DB2 include files for your applications are in the following directory:

  ```
  %DB2PATH%\include\cobol_a
  ```

  The `cobol_a` directory is the default.
- Specify COPY file names to include the `.cbl` extension as follows:

  ```
  COPY "sql.cbl".
  ```

Related tasks:

- "Building IBM COBOL applications on Windows" on page 270
- "Building IBM COBOL routines on Windows" on page 273

Related reference:

- "Windows IBM COBOL application compile and link options" on page 272
- "Windows IBM COBOL routine compile and link options" on page 276

Building IBM COBOL applications on Windows

DB2 provides batch files for compiling and linking DB2 API and embedded SQL programs. These are located in the `sqllib\samples\cobol` directory, along with sample programs that can be built with these files.

DB2 supports two precompilers for building IBM COBOL applications on Windows, the DB2 precompiler and the IBM COBOL precompiler. The default is the DB2 precompiler. The IBM COBOL precompiler can be selected by uncommenting the appropriate line in the batch file you are using. Precompilation with IBM COBOL is done by the compiler itself, using specific precompile options.

The batch file, `bldapp.bat`, contains the commands to build a DB2 application program. It takes up to four parameters, represented inside the batch file by the variables `%1`, `%2`, `%3`, and `%4`.

The first parameter, `%1`, specifies the name of your source file. This is the only required parameter for programs that do not contain embedded SQL. Building embedded SQL programs requires a connection to the database so three optional parameters are also provided: the second parameter, `%2`, specifies the name of the database to which you want to connect; the third parameter, `%3`, specifies the user ID for the database, and `%4` specifies the password.

For an embedded SQL program using the default DB2 precompiler, `bldapp.bat` passes the parameters to the precompile and bind file, `embprep.bat`.

For an embedded SQL program using the IBM COBOL precompiler, `bldapp.bat` copies the `.sqb` source file to a `.cbl` source file. The compiler performs the precompile on the `.cbl` source file with specific precompile options.

For either precompiler, if no database name is supplied, the default `sample` database is used. The user ID and password parameters are only needed if the instance where the program is built is different from the instance where the database is located.

Procedure:

The following examples show you how to build and run DB2 API and embedded SQL applications.

To build the non-embedded SQL sample program `client` from the source file `client.cbl`, enter:

```
bldapp client
```

The result is an executable file `client.exe`. You can run the executable file against the `sample` database by entering the executable name (without the extension):

```
client
```

Building and running embedded SQL applications

There are three ways to build the embedded SQL application, `updat`, from the source file `updat.sqb`:

1. If connecting to the sample database on the same instance, enter:
   ```
   bldapp updat
   ```
2. If connecting to another database on the same instance, also enter the database name:
   ```
   bldapp updat database
   ```
3. If connecting to a database on another instance, also enter the user ID and password of the database instance:
   ```
   bldapp updat database userid password
   ```

The result is an executable file, `updat`.

There are three ways to run this embedded SQL application:

1. If accessing the `sample` database on the same instance, simply enter the executable name:
   ```
   updat
   ```
2. If accessing another database on the same instance, enter the executable name and the database name:
   ```
   updat database
   ```
3. If accessing a database on another instance, enter the executable name, database name, and user ID and password of the database instance:
   ```
   updat database userid password
   ```

Related concepts:

- "Build files" on page 97

Related reference:

- "Windows IBM COBOL application compile and link options" on page 272
- "COBOL samples" on page 73

Related samples:

- "bldapp.bat -- Builds Windows VisualAge COBOL applications"
- "client.cbl -- How to set and query a client (IBM COBOL)"
- "embprep.bat -- To prep and bind a COBOL embedded SQL program on Windows"
- "updat.sqb -- How to update, delete and insert table data (IBM COBOL)"

Batch file for IBM COBOL applications

```
@echo off
rem BATCH FILE: bldapp.bat
rem Builds Windows VisualAge COBOL applications
rem Usage: bldapp prog_name [ db_name [ userid password ]]

set IBMCOB_PRECOMP=
set EXTRA_COMPFLAG=

rem To use the IBM COBOL precompiler, uncomment the following line.
rem set IBMCOB_PRECOMP=true
```

```
rem If using the IBM COBOL precompiler
if "%IBMCOB_PRECOMP%" == "true" goto IBMCOB_precompile_step

rem Using the default DB2 precompiler,
rem If an embedded SQL program, precompile and bind it.
if not exist "%1.sqb" goto compile_step
call embprep %1 %2 %3 %4
goto compile_step

:IBMCOB_precompile_step
rem Using the IBM COBOL precompiler,
rem Copy the <prog_name>.sqb file to <prog_name>.cbl.
if exist "%1.sqb" cp -f %1.sqb %1.cbl
rem Assign input parameters to the EXTRA_COMPFLAG variable
if "%1" == "" goto error
if "%2" == "" goto case1
if "%3" == "" goto case2
if "%4" == "" goto error
goto case3

:case1
  set EXTRA_COMPFLAG=-q"SQL('database sample CALL_RESOLUTION DEFERRED')"
  goto compile_step
:case2
  set EXTRA_COMPFLAG=-q"SQL('database %2 CALL_RESOLUTION DEFERRED')"
  goto compile_step
:case3
  set EXTRA_COMPFLAG=-q"SQL('database %2 user %3 using %4
    CALL_RESOLUTION DEFERRED')"
  goto compile_step

:compile_step
rem Compile the error-checking utility.
cob2 -qpgmname(mixed) -c -qlib -I"%DB2PATH%\include\cobol_a" checkerr.cbl

rem Compile the program.
cob2 -qpgmname(mixed) -c -qlib -I"%DB2PATH%\include\cobol_a" %1.cbl
  %EXTRA_COMPFLAG%

rem Link the program.
cob2 %1.obj checkerr.obj db2api.lib
goto exit

:error
echo Usage: bldapp prog_name [ db_name [ userid password ]]

:exit
@echo on
```

Windows IBM COBOL application compile and link options

The following are the compile and link options recommended by DB2 for building COBOL embedded SQL and DB2 API applications on Windows with the IBM VisualAge COBOL compiler, as demonstrated in the `bldapp.bat` batch file.

Compile and link options for bldapp
Compile options: **cob2** The IBM VisualAge COBOL compiler. **-qpgmname(mixed)** Instructs the compiler to permit CALLs to library entry points with mixed-case names. **-c** Perform compile only; no link. Compile and link are separate steps. **-qlib** Instructs the compiler to process COPY statements. **-I***path* Specify the location of the DB2 include files. For example: `-I"%DB2PATH%\include\cobol_a"`. **%EXTRA_COMPFLAG%** If `"set IBMCOB_PRECOMP=true"` is uncommented, the IBM COBOL precompile options are used with one of the following formulations, depending on the input parameters: **-q"SQL('database sample CALL_RESOLUTION DEFERRED')"** precompile using the default sample database, and defer call resolution. **-q"SQL('database %2 CALL_RESOLUTION DEFERRED')"** precompile using a database specified by the user, and defer call resolution. **-q"SQL('database %2 user %3 using %4 CALL_RESOLUTION DEFERRED')"** precompile using a database, user ID, and password specified by the user, and defer call resolution. This is the format for remote client access.
Link options: **cob2** Use the compiler as a front-end for the linker **%1.obj** Include the program object file. **checkerr.obj** Include the error-checking utility object file. **db2api.lib** Link with the DB2 library. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building IBM COBOL applications on Windows" on page 270

Related samples:

- "bldapp.bat -- Builds Windows VisualAge COBOL applications"

Building IBM COBOL routines on Windows

DB2 provides batch files for compiling and linking DB2 API and embedded SQL programs in IBM COBOL. These are located in the `sqllib\samples\cobol` directory, along with sample programs that can be built with these files.

DB2 supports two precompilers for building IBM COBOL applications on Windows, the DB2 precompiler and the IBM COBOL precompiler. The default is the DB2 precompiler. The IBM COBOL precompiler can be selected by uncommenting the appropriate line in the batch file you are using. Precompilation with IBM COBOL is done by the compiler itself, using specific precompile options.

The batch file, bldrtn.bat, contains the commands to build embedded SQL routines (stored procedures). The batch file compiles the routines into a DLL on the server. It takes two parameters, represented inside the batch file by the variables %1 and %2.

The first parameter, %1, specifies the name of your source file. The batch file uses the source file name, %1, for the DLL name. The second parameter, %2, specifies the name of the database to which you want to connect. Since the stored procedure must be built on the same instance where the database resides, there are no parameters for user ID and password.

Only the first parameter, source file name, is required. Database name is optional. If no database name is supplied, the program uses the default sample database.

If using the default DB2 precompiler, bldrtn.bat passes the parameters to the precompile and bind file, embprep.bat.

If using the IBM COBOL precompiler, bldrtn.bat copies the .sqb source file to a .cbl source file. The compiler performs the precompile on the .cbl source file with specific precompile options.

Procedure:

To build the sample program outsrv from the source file outsrv.sqb, connecting to the sample database, enter:

```
bldrtn outsrv
```

If connecting to another database, also include the database name:

```
bldrtn outsrv database
```

The batch file copies the DLL to the server in the path sqllib\function.

Once you build the DLL outsrv, you can build the client application outcli that calls the routine within the DLL (which has the same name as the DLL). You can build outcli using the batch file bldapp.bat.

To call the outsrv routine, run the sample client application by entering:

```
outcli database userid password
```

where

database
Is the name of the database to which you want to connect. The name could be sample, or its remote alias, or some other name.

userid Is a valid user ID.

password
Is a valid password for the user ID.

The client application accesses the DLL, outsrv, and executes the routine of the same name on the server database, and then returns the output to the client application.

Related concepts:

• "Build files" on page 97

Related reference:

- "Windows IBM COBOL routine compile and link options" on page 276
- "COBOL samples" on page 73

Related samples:

- "bldrtn.bat -- Builds Windows VisualAge COBOL routines (stored procedures)"
- "embprep.bat -- To prep and bind a COBOL embedded SQL program on Windows"
- "outcli.sqb -- Call stored procedures using the SQLDA structure (IBM COBOL)"
- "outsrv.sqb -- Demonstrates stored procedures using the SQLDA structure (IBM COBOL)"

Batch file for IBM COBOL routines

```
@echo off
rem BATCH FILE: bldrtn.bat
rem Builds Windows VisualAge COBOL routines (stored procedures)
rem Usage: bldrtn prog_name [ db_name ]

set IBMCOB_PRECOMP=
set EXTRA_COMPFLAG=

rem To use the IBM COBOL precompiler, uncomment the following line.
rem set IBMCOB_PRECOMP=true

rem If using the IBM COBOL precompiler
if "%IBMCOB_PRECOMP%" == "true" goto IBMCOB_precompile_step

rem Using the default DB2 precompiler,
rem Precompile and bind the program.
call embprep %1 %2
goto compile_step

:IBMCOB_precompile_step
rem Using the IBM COBOL precompiler,
rem Copy the <prog_name>.sqb file to <prog_name>.cbl.
if exist "%1.sqb" cp -f %1.sqb %1.cbl
rem Assign input parameters to the EXTRA_COMPFLAG variable
if "%1" == "" goto error
if "%2" == "" goto case1

set EXTRA_COMPFLAG=-q"SQL('database %2 CALL_RESOLUTION DEFERRED')"
goto compile_step

:case1
  set EXTRA_COMPFLAG=-q"SQL('database sample CALL_RESOLUTION DEFERRED')"
  goto compile_step

:compile_step
rem  Compile the stored procedure.
cob2 -qpgmname(mixed) -c -qlib -I"%DB2PATH%\include\cobol_a" %1.cbl %EXTRA_COMPFLAG%

rem  Link the stored procedure and create a shared library.
ilib /nol /gi:%1 %1.obj
ilink /free /nol /dll db2api.lib %1.exp %1.obj iwzrwin3.obj

rem Copy stored procedure to the %DB2PATH%\function directory.
copy %1.dll "%DB2PATH%\function"
goto exit

:error
```

```
echo Usage: bldrtn prog_name [ db_name ]

:exit
@echo on
```

Windows IBM COBOL routine compile and link options

The following are the compile and link options recommended by DB2 for building COBOL routines (stored procedures and user-defined functions) on Windows with the IBM VisualAge COBOL compiler, as demonstrated in the `bldrtn.bat` batch file.

Compile and link options for bldrtn
Compile options: **cob2** The IBM VisualAge COBOL compiler. **-qpgmname(mixed)** Instructs the compiler to permit CALLs to library entry points with mixed-case names. **-c** Perform compile only; no link. This batch file has separate compile and link steps. **-qlib** Instructs the compiler to process COPY statements. **-I***path* Specify the location of the DB2 include files. For example: `-I"%DB2PATH%\include\cobol_a"`. **%EXTRA_COMPFLAG%** If `"set IBMCOB_PRECOMP=true"` is uncommented, the IBM COBOL precompile options are used with one of the following formulations, depending on the input parameters: **-q"SQL('database sample CALL_RESOLUTION DEFERRED')"** precompile using the default sample database, and defer call resolution. **-q"SQL('database %2 CALL_RESOLUTION DEFERRED')"** precompile using a database specified by the user, and defer call resolution.
Link options: **ilink** Use the IBM VisualAge COBOL linker. **/free** Free format. **/nol** No logo. **/dll** Create the DLL with the source program name. **db2api.lib** Link with the DB2 library. **%1.exp** Include the export file. **%1.obj** Include the program object file. **iwzrwin3.obj** Include the object file provided by IBM VisualAge COBOL. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building IBM COBOL routines on Windows" on page 273

Related samples:

- "bldrtn.bat -- Builds Windows VisualAge COBOL routines (stored procedures)"

Micro Focus COBOL

Configuring the Micro Focus COBOL compiler on Windows

If you develop applications that contain embedded SQL and DB2 API calls, and you are using the Micro Focus compiler, there are several points to keep in mind.

Procedure:

- When you precompile your application using the command line processor command `db2 prep`, use the `target mfcob` option.
- Ensure the LIB environment variable points to `%DB2PATH%\lib` like this:

  ```
  set LIB="%DB2PATH%\lib;%LIB%"
  ```

- The DB2 COPY files for Micro Focus COBOL reside in `%DB2PATH%\include\cobol_mf`. Set the `COBCPY` environment variable to include the directory like this:

  ```
  set COBCPY="%DB2PATH%\include\cobol_mf;%COBCPY%"
  ```

You must make calls to all DB2 application programming interfaces using calling convention 74. The DB2 COBOL precompiler automatically inserts a CALL-CONVENTION clause in a SPECIAL-NAMES paragraph. If the SPECIAL-NAMES paragraph does not exist, the DB2 COBOL precompiler creates it, as follows:

```
Identification Division
Program-ID. "static".
special-names.
    call-convention 74 is DB2API.
```

Also, the precompiler automatically places the symbol DB2API, which is used to identify the calling convention, after the ″call″ keyword whenever a DB2 API is called. This occurs, for instance, whenever the precompiler generates a DB2 API run-time call from an embedded SQL statement.

If calls to DB2 APIs are made in an application which is not precompiled, you should manually create a SPECIAL-NAMES paragraph in the application, similar to that given above. If you are calling a DB2 API directly, then you will need to manually add the DB2API symbol after the ″call″ keyword.

Related tasks:

- “Building Micro Focus COBOL applications on Windows” on page 277
- “Building Micro Focus COBOL routines on Windows” on page 280

Related reference:

- “Windows Micro Focus COBOL application compile and link options” on page 279
- “Windows Micro Focus COBOL routine compile and link options” on page 282

Building Micro Focus COBOL applications on Windows

DB2 provides batch files for compiling and linking DB2 API and embedded SQL programs. These are located in the `sqllib\samples\cobol_mf` directory, along with sample programs that can be built with these files.

The batch file bldapp.bat contains the commands to build a DB2 application program. It takes up to four parameters, represented inside the batch file by the variables %1, %2, %3, and %4.

The first parameter, %1, specifies the name of your source file. This is the only required parameter for programs that do not contain embedded SQL. Building embedded SQL programs requires a connection to the database so three optional parameters are also provided: the second parameter, %2, specifies the name of the database to which you want to connect; the third parameter, %3, specifies the user ID for the database, and %4 specifies the password.

For an embedded SQL program, bldapp passes the parameters to the precompile and bind batch file, embprep.bat. If no database name is supplied, the default sample database is used. The user ID and password parameters are only needed if the instance where the program is built is different from the instance where the database is located.

Procedure:

The following examples show you how to build and run DB2 API and embedded SQL applications.

To build the non-embedded SQL sample program, client, from the source file client.cbl, enter:

```
bldapp client
```

The result is an executable file client.exe. You can run the executable file against the sample database by entering the executable name (without the extension):

```
client
```

Building and Running Embedded SQL Applications

There are three ways to build the embedded SQL application, updat, from the source file updat.sqb:

1. If connecting to the sample database on the same instance, enter:
   ```
   bldapp updat
   ```
2. If connecting to another database on the same instance, also enter the database name:
   ```
   bldapp updat database
   ```
3. If connecting to a database on another instance, also enter the user ID and password of the database instance:
   ```
   bldapp updat database userid password
   ```

The result is an executable file, updat.exe.

There are three ways to run this embedded SQL application:

1. If accessing the sample database on the same instance, simply enter the executable name (without the extension):
   ```
   updat
   ```
2. If accessing another database on the same instance, enter the executable name and the database name:
   ```
   updat database
   ```

3. If accessing a database on another instance, enter the executable name, database name, and user ID and password of the database instance:

   ```
   updat database userid password
   ```

Related concepts:

- "Build files" on page 97

Related reference:

- "Windows Micro Focus COBOL application compile and link options" on page 279
- "COBOL samples" on page 73

Related samples:

- "bldapp.bat -- Builds Windows Micro Focus Cobol applications"
- "client.cbl -- How to set and query a client (MF COBOL)"
- "updat.sqb -- How to update, delete and insert table data (MF COBOL)"
- "embprep.bat -- Prep and binds a C/C++ or Micro Focus COBOL embedded SQL program on Windows"

Batch file for Micro Focus COBOL applications

```
@echo off
rem BATCH FILE: bldapp.bat
rem Builds Windows Micro Focus Cobol applications
rem Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

rem If an embedded SQL program, precompile and bind it.
if not exist "%1.sqb" goto compile_step
call embprep %1 %2 %3 %4

:compile_step
rem Compile the error-checking utility.
cobol checkerr.cbl;

rem  Compile the program.
cobol %1.cbl;

rem  Link the program.
cbllink -l %1.obj checkerr.obj db2api.lib
@echo on
```

Windows Micro Focus COBOL application compile and link options

The following are the compile and link options recommended by DB2 for building COBOL embedded SQL and DB2 API applications on Windows with the Micro Focus COBOL compiler, as demonstrated in the `bldapp.bat` batch file.

Compile and link options for bldapp	
Compile option:	
cobol	The Micro Focus COBOL compiler.

Compile and link options for bldapp
Link options: **cbllink** Use the linker to link edit. **-l** Link with the lcobol library. **checkerr.obj** Link with the error-checking utility object file. **db2api.lib** Link with the DB2 API library. Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building Micro Focus COBOL applications on Windows" on page 277

Related samples:

- "bldapp.bat -- Builds Windows Micro Focus Cobol applications"

Building Micro Focus COBOL routines on Windows

DB2 provides batch files for compiling and linking DB2 API and embedded SQL programs in Micro Focus COBOL. These are located in the `sqllib\samples\cobol_mf` directory, along with sample programs that can be built with these files.

The batch file `bldrtn.bat` contains the commands to build embedded SQL routines (stored procedures). The batch file compiles the routines into a DLL on the server. The batch file takes two parameters, represented inside the batch file by the variables `%1` and `%2`.

The first parameter, `%1`, specifies the name of your source file. The batch file uses the source file name, `%1`, for the DLL name. The second parameter, `%2`, specifies the name of the database to which you want to connect. Since the stored procedure must be built on the same instance where the database resides, there are no parameters for user ID and password.

Only the first parameter, source file name, is required. Database name is optional. If no database name is supplied, the program uses the default `sample` database.

Procedure:

To build the sample program `outsrv` from the source file `outsrv.sqb`, if connecting to the sample database, enter:

```
bldrtn outsrv
```

If connecting to another database, also enter the database name:

```
bldrtn outsrv database
```

The script file copies the DLL to the server in the path `sqllib/function`.

Once you build the DLL, `outsrv`, you can build the client application, `outcli`, that calls the routine within the DLL (which has the same name as the DLL). You can build `outcli` using the batch file, `bldapp.bat`.

To call the `outsrv` routine, run the sample client application by entering:

```
outcli database userid password
```

where

database
Is the name of the database to which you want to connect. The name could be `sample`, or its alias, or another name.

userid Is a valid user ID.

password
Is a valid password for the user ID.

The client application accesses the DLL, `outsrv`, which executes the routine of the same name on the server database. The output is then returned to the client application.

Related concepts:

- "Build files" on page 97

Related reference:

- "Windows Micro Focus COBOL routine compile and link options" on page 282
- "COBOL samples" on page 73

Related samples:

- "bldrtn.bat -- Builds Windows Micro Focus Cobol routines (stored procedures)"
- "outcli.sqb -- Call stored procedures using the SQLDA structure (MF COBOL)"
- "outsrv.sqb -- Demonstrates stored procedures using the SQLDA structure (MF COBOL)"
- "embprep.bat -- Prep and binds a C/C++ or Micro Focus COBOL embedded SQL program on Windows"

Batch file for Micro Focus COBOL routines

```
@echo off
rem BATCH FILE: bldrtn.bat
rem Builds Windows Micro Focus Cobol routines (stored procedures)
rem Usage: bldsrv <prog_name> [ <db_name> ]

rem Precompile and bind the program.
call embprep %1 %2

rem  Compile the stored procedure.
cobol %1.cbl /case;

rem  Link the stored procedure and create a shared library.
cbllink /d %1.obj db2api.lib

rem Copy the stored procedure to the %DB2PATH%\function directory.
copy %1.dll "%DB2PATH%\function"
@echo on
```

Windows Micro Focus COBOL routine compile and link options

The following are the compile and link options recommended by DB2 for building COBOL routines (stored procedures and user-defined functions) on Windows with the Micro Focus COBOL compiler, as demonstrated in the `bldrtn.bat` batch file.

Compile and link options for bldrtn
Compile options:
cobol The Micro Focus COBOL compiler.
/case Prevent external symbols being converted to uppercase.
Link options:
cbllink Use the Micro Focus COBOL linker to link edit.
/d Create a .dll file.
db2api.lib Link with the DB2 API library.
Refer to your compiler documentation for additional compiler options.

Related tasks:

- "Building Micro Focus COBOL routines on Windows" on page 280

Related samples:

- "bldrtn.bat -- Builds Windows Micro Focus Cobol routines (stored procedures)"

Object REXX

Building Object REXX applications on Windows

Object REXX is an object-oriented version of the REXX language. Object-oriented extensions have been added to classic REXX, but its existing functions and instructions have not changed. The Object REXX interpreter is an enhanced version of its predecessor, with additional support for:

- Classes, objects, and methods
- Messaging and polymorphism
- Single and multiple inheritance

Object REXX is fully compatible with classic REXX. In this section, whenever REXX is used, all versions of REXX are inferred, including Object REXX.

You do not precompile or bind REXX programs.

On Windows, REXX programs are not required to start with a comment. However, for portability reasons you are recommended to start each REXX program with a comment that begins in the first column of the first line. This will allow the program to be distinguished from a batch command on other platforms:

```
/* Any comment will do. */
```

REXX sample programs can be found in the directory `sqllib\samples\rexx`.

Procedure:

To run the sample REXX program `updat`, enter:

```
rexx updat.cmd
```

Related concepts:

- "Programming Considerations for REXX" in the *Application Development Guide: Programming Client Applications*

Related reference:

- "REXX samples" on page 88

Part 4. Appendixes

Appendix A. DB2 Universal Database technical information

DB2 documentation and help

DB2® technical information is available through the following tools and methods:

- DB2 Information Center
  - Topics
  - Help for DB2 tools
  - Sample programs
  - Tutorials
- Downloadable PDF files, PDF files on CD, and printed books
  - Guides
  - Reference manuals
- Command line help
  - Command help
  - Message help
  - SQL state help
- Installed source code
  - Sample programs

You can access additional DB2 Universal Database™ technical information such as technotes, white papers, and Redbooks™ online at ibm.com®. Access the DB2 Information Management software library site at www.ibm.com/software/data/pubs/.

DB2 documentation updates

IBM® may periodically make documentation FixPaks and other documentation updates to the DB2 Information Center available. If you access the DB2 Information Center at http://publib.boulder.ibm.com/infocenter/db2help/, you will always be viewing the most up-to-date information. If you have installed the DB2 Information Center locally, then you need to install any updates manually before you can view them. Documentation updates allow you to update the information that you installed from the *DB2 Information Center CD* when new information becomes available.

The Information Center is updated more frequently than either the PDF or the hardcopy books. To get the most current DB2 technical information, install the documentation updates as they become available or go to the DB2 Information Center at the www.ibm.com site.

Related concepts:

- "CLI sample programs" in the *CLI Guide and Reference, Volume 1*
- "Java sample programs" on page 107
- "DB2 Information Center" on page 288

Related tasks:

- "Invoking contextual help from a DB2 tool" on page 305

- "Updating the DB2 Information Center installed on your computer or intranet server" on page 297
- "Invoking message help from the command line processor" on page 306
- "Invoking command help from the command line processor" on page 307
- "Invoking SQL state help from the command line processor" on page 307

Related reference:

- "DB2 PDF and printed documentation" on page 299

DB2 Information Center

The DB2® Information Center gives you access to all of the information you need to take full advantage of DB2 family products, including DB2 Universal Database™, DB2 Connect™, DB2 Information Integrator and DB2 Query Patroller™. The DB2 Information Center also contains information for major DB2 features and components including replication, data warehousing, and the DB2 extenders.

The DB2 Information Center has the following features if you view it in Mozilla 1.0 or later or Microsoft® Internet Explorer 5.5 or later. Some features require you to enable support for JavaScript™:

Flexible installation options

You can choose to view the DB2 documentation using the option that best meets your needs:

- To effortlessly ensure that your documentation is always up to date, you can access all of your documentation directly from the DB2 Information Center hosted on the IBM® Web site at http://publib.boulder.ibm.com/infocenter/db2help/
- To minimize your update efforts and keep your network traffic within your intranet, you can install the DB2 documentation on a single server on your intranet
- To maximize your flexibility and reduce your dependence on network connections, you can install the DB2 documentation on your own computer

Search

You can search all of the topics in the DB2 Information Center by entering a search term in the **Search** text field. You can retrieve exact matches by enclosing terms in quotation marks, and you can refine your search with wildcard operators (*, ?) and Boolean operators (AND, NOT, OR).

Task-oriented table of contents

You can locate topics in the DB2 documentation from a single table of contents. The table of contents is organized primarily by the kind of tasks you may want to perform, but also includes entries for product overviews, goals, reference information, an index, and a glossary.

- Product overviews describe the relationship between the available products in the DB2 family, the features offered by each of those products, and up to date release information for each of these products.
- Goal categories such as installing, administering, and developing include topics that enable you to quickly complete tasks and develop a deeper understanding of the background information for completing those tasks.

- Reference topics provide detailed information about a subject, including statement and command syntax, message help, and configuration parameters.

Show current topic in table of contents
You can show where the current topic fits into the table of contents by clicking the **Refresh / Show Current Topic** button in the table of contents frame or by clicking the **Show in Table of Contents** button in the content frame. This feature is helpful if you have followed several links to related topics in several files or arrived at a topic from search results.

Index You can access all of the documentation from the index. The index is organized in alphabetical order by index term.

Glossary
You can use the glossary to look up definitions of terms used in the DB2 documentation. The glossary is organized in alphabetical order by glossary term.

Integrated localized information
The DB2 Information Center displays information in the preferred language set in your browser preferences. If a topic is not available in your preferred language, the DB2 Information Center displays the English version of that topic.

For iSeries™ technical information, refer to the IBM eServer™ iSeries information center at www.ibm.com/eserver/iseries/infocenter/.

Related concepts:

- "DB2 Information Center installation scenarios" on page 289

Related tasks:

- "Updating the DB2 Information Center installed on your computer or intranet server" on page 297
- "Displaying topics in your preferred language in the DB2 Information Center" on page 298
- "Invoking the DB2 Information Center" on page 296
- "Installing the DB2 Information Center using the DB2 Setup wizard (UNIX)" on page 292
- "Installing the DB2 Information Center using the DB2 Setup wizard (Windows)" on page 294

DB2 Information Center installation scenarios

Different working environments can pose different requirements for how to access DB2® information. The DB2 Information Center can be accessed on the IBM® Web site, on a server on your organization's network, or on a version installed on your computer. In all three cases, the documentation is contained in the DB2 Information Center, which is an architected web of topic-based information that you view with a browser. By default, DB2 products access the DB2 Information Center on the IBM Web site. However, if you want to access the DB2 Information Center on an intranet server or on your own computer, you must install the DB2 Information Center using the DB2 Information Center CD found in your product Media Pack. Refer to the summary of options for accessing DB2 documentation which follows, along with the three installation scenarios, to help determine which

method of accessing the DB2 Information Center works best for you and your work environment, and what installation issues you might need to consider.

Summary of options for accessing DB2 documentation:

The following table provides recommendations on which options are possible in your work environment for accessing the DB2 product documentation in the DB2 Information Center.

Internet access	Intranet access	Recommendation
Yes	Yes	Access the DB2 Information Center on the IBM Web site, or access the DB2 Information Center installed on an intranet server.
Yes	No	Access the DB2 Information Center on the IBM Web site.
No	Yes	Access the DB2 Information Center installed on an intranet server.
No	No	Access the DB2 Information Center on a local computer.

Scenario: Accessing the DB2 Information Center on your computer:

Tsu-Chen owns a factory in a small town that does not have a local ISP to provide him with Internet access. He purchased DB2 Universal Database™ to manage his inventory, his product orders, his banking account information, and his business expenses. Never having used a DB2 product before, Tsu-Chen needs to learn how to do so from the DB2 product documentation.

After installing DB2 Universal Database on his computer using the typical installation option, Tsu-Chen tries to access the DB2 documentation. However, his browser gives him an error message that the page he tried to open cannot be found. Tsu-Chen checks the installation manual for his DB2 product and discovers that he has to install the DB2 Information Center if he wants to access DB2 documentation on his computer. He finds the *DB2 Information Center CD* in the media pack and installs it.

From the application launcher for his operating system, Tsu-Chen now has access to the DB2 Information Center and can learn how to use his DB2 product to increase the success of his business.

Scenario: Accessing the DB2 Information Center on the IBM Web site:

Colin is an information technology consultant with a training firm. He specializes in database technology and SQL and gives seminars on these subjects to businesses all over North America using DB2 Universal Database. Part of Colin's seminars includes using DB2 documentation as a teaching tool. For example, while teaching courses on SQL, Colin uses the DB2 documentation on SQL as a way to teach basic and advanced syntax for database queries.

Most of the businesses at which Colin teaches have Internet access. This situation influenced Colin's decision to configure his mobile computer to access the DB2 Information Center on the IBM Web site when he installed the latest version of DB2 Universal Database. This configuration allows Colin to have online access to the latest DB2 documentation during his seminars.

However, sometimes while travelling Colin does not have Internet access. This posed a problem for him, especially when he needed to access to DB2 documentation to prepare for seminars. To avoid situations like this, Colin installed a copy of the DB2 Information Center on his mobile computer.

Colin enjoys the flexibility of always having a copy of DB2 documentation at his disposal. Using the **db2set** command, he can easily configure the registry variables on his mobile computer to access the DB2 Information Center on either the IBM Web site, or his mobile computer, depending on his situation.

Scenario: Accessing the DB2 Information Center on an intranet server:

Eva works as a senior database administrator for a life insurance company. Her administration responsibilities include installing and configuring the latest version of DB2 Universal Database on the company's UNIX® database servers. Her company recently informed its employees that, for security reasons, it would not provide them with Internet access at work. Because her company has a networked environment, Eva decides to install a copy of the DB2 Information Center on an intranet server so that all employees in the company who use the company's data warehouse on a regular basis (sales representatives, sales managers, and business analysts) have access to DB2 documentation.

Eva instructs her database team to install the latest version of DB2 Universal Database on all of the employee's computers using a response file, to ensure that each computer is configured to access the DB2 Information Center using the host name and the port number of the intranet server.

However, through a misunderstanding Migual, a junior database administrator on Eva's team, installs a copy of the DB2 Information Center on several of the employee computers, rather than configuring DB2 Universal Database to access the DB2 Information Center on the intranet server. To correct this situation Eva tells Migual to use the **db2set** command to change the DB2 Information Center registry variables (DB2_DOCHOST for the host name, and DB2_DOCPORT for the port number) on each of these computers. Now all of the appropriate computers on the network have access to the DB2 Information Center, and employees can find answers to their DB2 questions in the DB2 documentation.

Related concepts:

- "DB2 Information Center" on page 288

Related tasks:

- "Updating the DB2 Information Center installed on your computer or intranet server" on page 297
- "Installing the DB2 Information Center using the DB2 Setup wizard (UNIX)" on page 292
- "Installing the DB2 Information Center using the DB2 Setup wizard (Windows)" on page 294
- "Setting the location for accessing the DB2 Information Center: Common GUI help"

Related reference:

- "db2set - DB2 Profile Registry Command" in the *Command Reference*

Installing the DB2 Information Center using the DB2 Setup wizard (UNIX)

DB2 product documentation can be accessed in three ways: on the IBM Web site, on an intranet server, or on a version installed on your computer. By default, DB2 products access DB2 documentation on the IBM Web site. If you want to access the DB2 documentation on an intranet server or on your own computer, you must install the documentation from the *DB2 Information Center CD*. Using the DB2 Setup wizard, you can define your installation preferences and install the DB2 Information Center on a computer that uses a UNIX operating system.

Prerequisites:

This section lists the hardware, operating system, software, and communication requirements for installing the DB2 Information Center on UNIX computers.

- **Hardware requirements**

  You require one of the following processors:
  - PowerPC (AIX)
  - HP 9000 (HP-UX)
  - Intel 32–bit (Linux)
  - Solaris UltraSPARC computers (Solaris Operating Environment)
- **Operating system requirements**

  You require one of the following operating systems:
  - IBM AIX 5.1 (on PowerPC)
  - HP-UX 11i (on HP 9000)
  - Red Hat Linux 8.0 (on Intel 32–bit)
  - SuSE Linux 8.1 (on Intel 32–bit)
  - Sun Solaris Version 8 (on Solaris Operating Environment UltraSPARC computers)

  Note: The DB2 Information Center runs on a subset of the UNIX operating systems on which DB2 clients are supported. It is therefore recommended that you either access the DB2 Information Center from the IBM Web site, or that you install and access the DB2 Information Center on an intranet server.
- **Software requirements**
  - The following browser is supported:
    - Mozilla Version 1.0 or greater
- The DB2 Setup wizard is a graphical installer. You must have an implementation of the X Window System software capable of rendering a graphical user interface for the DB2 Setup wizard to run on your computer. Before you can run the DB2 Setup wizard you must ensure that you have properly exported your display. For example, enter the following command at the command prompt:

  ```
  export DISPLAY=9.26.163.144:0.
  ```
- **Communication requirements**
  - TCP/IP

Procedure:

To install the DB2 Information Center using the DB2 Setup wizard:

1. Log on to the system.
2. Insert and mount the DB2 Information Center product CD on your system.
3. Change to the directory where the CD is mounted by entering the following command:

   ```
   cd /cd
   ```

   where */cd* represents the mount point of the CD.
4. Enter the **./db2setup** command to start the DB2 Setup wizard.
5. The IBM DB2 Setup Launchpad opens. To proceed directly to the installation of the DB2 Information Center, click **Install Product**. Online help is available to guide you through the remaining steps. To invoke the online help, click **Help**. You can click **Cancel** at any time to end the installation.
6. On the **Select the product you would like to install** page, click **Next**.
7. Click **Next** on the **Welcome to the DB2 Setup wizard** page. The DB2 Setup wizard will guide you through the program setup process.
8. To proceed with the installation, you must accept the license agreement. On the **License Agreement** page, select **I accept the terms in the license agreement** and click **Next**.
9. Select **Install DB2 Information Center on this computer** on the **Select the installation action** page. If you want to use a response file to install the DB2 Information Center on this or other computers at a later time, select **Save your settings in a response file**. Click **Next**.
10. Select the languages in which the DB2 Information Center will be installed on **Select the languages to install** page. Click **Next**.
11. Configure the DB2 Information Center for incoming communication on the **Specify the DB2 Information Center port** page. Click **Next** to continue the installation.
12. Review the installation choices you have made in the **Start copying files** page. To change any settings, click **Back**. Click **Install** to copy the DB2 Information Center files onto your computer.

You can also install the DB2 Information Center using a response file.

The installation logs db2setup.his, db2setup.log, and db2setup.err are located, by default, in the /tmp directory.

The db2setup.log file captures all DB2 product installation information, including errors. The db2setup.his file records all DB2 product installations on your computer. DB2 appends the db2setup.log file to the db2setup.his file. The db2setup.err file captures any error output that is returned by Java, for example, exceptions and trap information.

When the installation is complete, the DB2 Information Center will be installed in one of the following directories, depending upon your UNIX operating system:

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris Operating Environment: /opt/IBM/db2/V8.1

Related concepts:

- "DB2 Information Center" on page 288
- "DB2 Information Center installation scenarios" on page 289

Related tasks:

- "Installing DB2 using a response file (UNIX)" in the *Installation and Configuration Supplement*
- "Updating the DB2 Information Center installed on your computer or intranet server" on page 297
- "Displaying topics in your preferred language in the DB2 Information Center" on page 298
- "Invoking the DB2 Information Center" on page 296
- "Installing the DB2 Information Center using the DB2 Setup wizard (Windows)" on page 294

Installing the DB2 Information Center using the DB2 Setup wizard (Windows)

DB2 product documentation can be accessed in three ways: on the IBM Web site, on an intranet server, or on a version installed on your computer. By default, DB2 products access DB2 documentation on the IBM Web site. If you want to access the DB2 documentation on an intranet server or on your own computer, you must install the DB2 documentation from the *DB2 Information Center CD*. Using the DB2 Setup wizard, you can define your installation preferences and install the DB2 Information Center on a computer that uses a Windows operating system.

Prerequisites:

This section lists the hardware, operating system, software, and communication requirements for installing the DB2 Information Center on Windows.

- **Hardware requirements**

  You require one of the following processors:

  – 32-bit computers: a Pentium or Pentium compatible CPU
- **Operating system requirements**

  You require one of the following operating systems:

  – Windows 2000

  – Windows XP

  Note: The DB2 Information Center runs on a subset of the Windows operating systems on which DB2 clients are supported. It is therefore recommended that you either access the DB2 Information Center on the IBM Web site, or that you install and access the DB2 Information Center on an intranet server.
- **Software requirements**

  – The following browsers are supported:

    - Mozilla 1.0 or greater

    - Internet Explorer Version 5.5 or 6.0 (Version 6.0 for Windows XP)
- **Communication requirements**

  – TCP/IP

Restrictions:

- You require an account with administrative privileges to install the DB2 Information Center.

Procedure:

To install the DB2 Information Center using the DB2 Setup wizard:

1. Log on to the system with the account that you have defined for the DB2 Information Center installation.
2. Insert the CD into the drive. If enabled, the auto-run feature starts the IBM DB2 Setup Launchpad.
3. The DB2 Setup wizard determines the system language and launches the setup program for that language. If you want to run the setup program in a language other than English, or the setup program fails to auto-start, you can start the DB2 Setup wizard manually.

   To start the DB2 Setup wizard manually:

   a. Click **Start** and select **Run**.
   b. In the **Open** field, type the following command:

      ```
      x:\setup.exe /i 2-letter language identifier
      ```

      where *x:* represents your CD drive, and *2-letter language identifier* represents the language in which the setup program will be run.
   c. Click **OK**.
4. The IBM DB2 Setup Launchpad opens. To proceed directly to the installation of the DB2 Information Center, click **Install Product**. Online help is available to guide you through the remaining steps. To invoke the online help, click **Help**. You can click **Cancel** at any time to end the installation.
5. On the **Select the product you would like to install** page, click **Next**.
6. Click **Next** on the **Welcome to the DB2 Setup wizard** page. The DB2 Setup wizard will guide you through the program setup process.
7. To proceed with the installation, you must accept the license agreement. On the **License Agreement** page, select **I accept the terms in the license agreement** and click **Next**.
8. Select **Install DB2 Information Center on this computer** on the **Select the installation action** page. If you want to use a response file to install the DB2 Information Center on this or other computers at a later time, select **Save your settings in a response file**. Click **Next**.
9. Select the languages in which the DB2 Information Center will be installed on **Select the languages to install** page. Click **Next**.
10. Configure the DB2 Information Center for incoming communication on the **Specify the DB2 Information Center port** page. Click **Next** to continue the installation.
11. Review the installation choices you have made in the **Start copying files** page. To change any settings, click **Back**. Click **Install** to copy the DB2 Information Center files onto your computer.

You can install the DB2 Information Center using a response file. You can also use the **db2rspgn** command to generate a response file based on an existing installation.

For information on errors encountered during installation, see the db2.log and db2wi.log files located in the 'My Documents'\DB2LOG\ directory. The location of the 'My Documents' directory will depend on the settings on your computer.

The db2wi.log file captures the most recent DB2 installation information. The db2.log captures the history of DB2 product installations.

Related concepts:

- "DB2 Information Center" on page 288
- "DB2 Information Center installation scenarios" on page 289

Related tasks:

- "Installing a DB2 product using a response file (Windows)" in the *Installation and Configuration Supplement*
- "Updating the DB2 Information Center installed on your computer or intranet server" on page 297
- "Displaying topics in your preferred language in the DB2 Information Center" on page 298
- "Invoking the DB2 Information Center" on page 296
- "Installing the DB2 Information Center using the DB2 Setup wizard (UNIX)" on page 292

Related reference:

- "db2rspgn - Response File Generator Command (Windows)" in the *Command Reference*

Invoking the DB2 Information Center

The DB2 Information Center gives you access to all of the information that you need to use DB2 products for Linux, UNIX, and Windows operating systems such as DB2 Universal Database, DB2 Connect, DB2 Information Integrator, and DB2 Query Patroller.

You can invoke the DB2 Information Center from one of the following places:

- Computers on which a DB2 UDB client or server is installed
- An intranet server or local computer on which the DB2 Information Center installed
- The IBM Web site

Prerequisites:

Before you invoke the DB2 Information Center:

- *Optional*: Configure your browser to display topics in your preferred language
- *Optional*: Configure your DB2 client to use the DB2 Information Center installed on your computer or intranet server

Procedure:

To invoke the DB2 Information Center on a computer on which a DB2 UDB client or server is installed:

- From the Start Menu (Windows operating system): Click **Start → Programs → IBM DB2 → Information → Information Center**.
- From the command line prompt:
  - For Linux and UNIX operating systems, issue the **db2icdocs** command.
  - For the Windows operating system, issue the **db2icdocs.exe** command.

To open the DB2 Information Center installed on an intranet server or local computer in a Web browser:

- Open the Web page at http://<host-name>:<port-number>/, where <host-name> represents the host name and <port-number> represents the port number on which the DB2 Information Center is available.

To open the DB2 Information Center on the IBM Web site in a Web browser:

- Open the Web page at publib.boulder.ibm.com/infocenter/db2help/.

Related concepts:

- "DB2 Information Center" on page 288
- "DB2 Information Center installation scenarios" on page 289

Related tasks:

- "Displaying topics in your preferred language in the DB2 Information Center" on page 298
- "Invoking contextual help from a DB2 tool" on page 305
- "Updating the DB2 Information Center installed on your computer or intranet server" on page 297
- "Invoking command help from the command line processor" on page 307
- "Setting the location for accessing the DB2 Information Center: Common GUI help"

Related reference:

- "HELP Command" in the *Command Reference*

Updating the DB2 Information Center installed on your computer or intranet server

The DB2 Information Center available from http://publib.boulder.ibm.com/infocenter/db2help/ will be periodically updated with new or changed documentation. IBM may also make DB2 Information Center updates available to download and install on your computer or intranet server. Updating the DB2 Information Center does not update DB2 client or server products.

Prerequisites:

You must have access to a computer that is connected to the Internet.

Procedure:

To update the DB2 Information Center installed on your computer or intranet server:

1. Open the DB2 Information Center hosted on the IBM Web site at: http://publib.boulder.ibm.com/infocenter/db2help/
2. In the Downloads section of the welcome page under the Service and Support heading, click the **DB2 Universal Database documentation** link.
3. Determine if the version of your DB2 Information Center is out of date by comparing the latest refreshed documentation image level to the documentation level you have installed. The documentation level you have installed is listed on the DB2 Information Center welcome page.

4. If a more recent version of the DB2 Information Center is available, download the latest refreshed *DB2 Information Center* image applicable to your operating system.
5. To install the refreshed *DB2 Information Center* image, follow the instructions provided on the Web page.

Related concepts:

- "DB2 Information Center installation scenarios" on page 289

Related tasks:

- "Invoking the DB2 Information Center" on page 296
- "Installing the DB2 Information Center using the DB2 Setup wizard (UNIX)" on page 292
- "Installing the DB2 Information Center using the DB2 Setup wizard (Windows)" on page 294

Displaying topics in your preferred language in the DB2 Information Center

The DB2 Information Center attempts to display topics in the language specified in your browser preferences. If a topic has not been translated into your preferred language, the DB2 Information Center displays the topic in English.

Procedure:

To display topics in your preferred language in the Internet Explorer browser:

1. In Internet Explorer, click the **Tools** —> **Internet Options** —> **Languages...** button. The Language Preferences window opens.
2. Ensure your preferred language is specified as the first entry in the list of languages.
   - To add a new language to the list, click the **Add...** button.

     Note: Adding a language does not guarantee that the computer has the fonts required to display the topics in the preferred language.
   - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
3. Refresh the page to display the DB2 Information Center in your preferred language.

To display topics in your preferred language in the Mozilla browser:

1. In Mozilla, select the **Edit** —> **Preferences** —> **Languages** button. The Languages panel is displayed in the Preferences window.
2. Ensure your preferred language is specified as the first entry in the list of languages.
   - To add a new language to the list, click the **Add...** button to select a language from the Add Languages window.
   - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
3. Refresh the page to display the DB2 Information Center in your preferred language.

Related concepts:

- "DB2 Information Center" on page 288

DB2 PDF and printed documentation

The following tables provide official book names, form numbers, and PDF file names. To order hardcopy books, you must know the official book name. To print a PDF file, you must know the PDF file name.

The DB2 documentation is categorized by the following headings:

- Core DB2 information
- Administration information
- Application development information
- Business intelligence information
- DB2 Connect information
- Getting started information
- Tutorial information
- Optional component information
- Release notes

The following tables describe, for each book in the DB2 library, the information needed to order the hard copy, or to print or view the PDF for that book. A full description of each of the books in the DB2 library is available from the IBM Publications Center at www.ibm.com/shop/publications/order

Core DB2 information

The information in these books is fundamental to all DB2 users; you will find this information useful whether you are a programmer, a database administrator, or someone who works with DB2 Connect, DB2 Warehouse Manager, or other DB2 products.

Table 42. Core DB2 information

Name	Form Number	PDF File Name
IBM DB2 Universal Database Command Reference	SC09-4828	db2n0x81
IBM DB2 Universal Database Glossary	No form number	db2t0x81
IBM DB2 Universal Database Message Reference, Volume 1	GC09-4840, not available in hardcopy	db2m1x81
IBM DB2 Universal Database Message Reference, Volume 2	GC09-4841, not available in hardcopy	db2m2x81
IBM DB2 Universal Database What's New	SC09-4848	db2q0x81

Administration information

The information in these books covers those topics required to effectively design, implement, and maintain DB2 databases, data warehouses, and federated systems.

Table 43. Administration information

Name	Form number	PDF file name
IBM DB2 Universal Database Administration Guide: Planning	SC09-4822	db2d1x81
IBM DB2 Universal Database Administration Guide: Implementation	SC09-4820	db2d2x81
IBM DB2 Universal Database Administration Guide: Performance	SC09-4821	db2d3x81
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0x81
IBM DB2 Universal Database Data Movement Utilities Guide and Reference	SC09-4830	db2dmx81
IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference	SC09-4831	db2hax81
IBM DB2 Universal Database Data Warehouse Center Administration Guide	SC27-1123	db2ddx81
IBM DB2 Universal Database SQL Reference, Volume 1	SC09-4844	db2s1x81
IBM DB2 Universal Database SQL Reference, Volume 2	SC09-4845	db2s2x81
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847	db2f0x81

Application development information

The information in these books is of special interest to application developers or programmers working with DB2 Universal Database (DB2 UDB). You will find information about supported languages and compilers, as well as the documentation required to access DB2 UDB using the various supported programming interfaces, such as embedded SQL, ODBC, JDBC, SQLJ, and CLI. If you are using the DB2 Information Center, you can also access HTML versions of the source code for the sample programs.

Table 44. Application development information

Name	Form number	PDF file name
IBM DB2 Universal Database Application Development Guide: Building and Running Applications	SC09-4825	db2axx81
IBM DB2 Universal Database Application Development Guide: Programming Client Applications	SC09-4826	db2a1x81
IBM DB2 Universal Database Application Development Guide: Programming Server Applications	SC09-4827	db2a2x81

Table 44. Application development information (continued)

Name	Form number	PDF file name
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1x81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2x81
IBM DB2 Universal Database Data Warehouse Center Application Integration Guide	SC27-1124	db2adx81
IBM DB2 XML Extender Administration and Programming	SC27-1234	db2sxx81

Business intelligence information

The information in these books describes how to use components that enhance the data warehousing and analytical capabilities of DB2 Universal Database.

Table 45. Business intelligence information

Name	Form number	PDF file name
IBM DB2 Warehouse Manager Standard Edition Information Catalog Center Administration Guide	SC27-1125	db2dix81
IBM DB2 Warehouse Manager Standard Edition Installation Guide	GC27-1122	db2idx81
IBM DB2 Warehouse Manager Standard Edition Managing ETI Solution Conversion Programs with DB2 Warehouse Manager	SC18-7727	iwhe1mstx80

DB2 Connect information

The information in this category describes how to access data on mainframe and midrange servers using DB2 Connect Enterprise Edition or DB2 Connect Personal Edition.

Table 46. DB2 Connect information

Name	Form number	PDF file name
IBM Connectivity Supplement	No form number	db2h1x81
IBM DB2 Connect Quick Beginnings for DB2 Connect Enterprise Edition	GC09-4833	db2c6x81
IBM DB2 Connect Quick Beginnings for DB2 Connect Personal Edition	GC09-4834	db2c1x81
IBM DB2 Connect User's Guide	SC09-4835	db2c0x81

Getting started information

The information in this category is useful when you are installing and configuring servers, clients, and other DB2 products.

Table 47. Getting started information

Name	Form number	PDF file name
IBM DB2 Universal Database Quick Beginnings for DB2 Clients	GC09-4832, not available in hardcopy	db2itx81
IBM DB2 Universal Database Quick Beginnings for DB2 Servers	GC09-4836	db2isx81
IBM DB2 Universal Database Quick Beginnings for DB2 Personal Edition	GC09-4838	db2i1x81
IBM DB2 Universal Database Installation and Configuration Supplement	GC09-4837, not available in hardcopy	db2iyx81
IBM DB2 Universal Database Quick Beginnings for DB2 Data Links Manager	GC09-4829	db2z6x81

Tutorial information

Tutorial information introduces DB2 features and teaches how to perform various tasks.

Table 48. Tutorial information

Name	Form number	PDF file name
Business Intelligence Tutorial: Introduction to the Data Warehouse	No form number	db2tux81
Business Intelligence Tutorial: Extended Lessons in Data Warehousing	No form number	db2tax81
Information Catalog Center Tutorial	No form number	db2aix81
Video Central for e-business Tutorial	No form number	db2twx81
Visual Explain Tutorial	No form number	db2tvx81

Optional component information

The information in this category describes how to work with optional DB2 components.

Table 49. Optional component information

Name	Form number	PDF file name
IBM DB2 Cube Views Guide and Reference	SC18–7298	db2aax81

Table 49. Optional component information (continued)

Name	Form number	PDF file name
IBM DB2 Query Patroller Guide: Installation, Administration and Usage Guide	GC09–7658	db2dwx81
IBM DB2 Spatial Extender and Geodetic Extender User's Guide and Reference	SC27-1226	db2sbx81
IBM DB2 Universal Database Data Links Manager Administration Guide and Reference	SC27-1221	db2z0x82
DB2 Net Search Extender Administration and User's Guide **Note:** HTML for this document is *not* installed from the HTML documentation CD.	SH12-6740	N/A

Release notes

The release notes provide additional information specific to your product's release and FixPak level. The release notes also provide summaries of the documentation updates incorporated in each release, update, and FixPak.

Table 50. Release notes

Name	Form number	PDF file name
DB2 Release Notes	See note.	See note.
DB2 Installation Notes	Available on product CD-ROM only.	Not available.

Note: The Release Notes are available in:

- XHTML and Text format, on the product CDs
- PDF format, on the PDF Documentation CD

In addition the portions of the Release Notes that discuss *Known Problems and Workarounds* and *Incompatibilities Between Releases* also appear in the DB2 Information Center.

To view the Release Notes in text format on UNIX-based platforms, see the Release.Notes file. This file is located in the DB2DIR/Readme/%L directory, where *%L* represents the locale name and DB2DIR represents:

- For AIX operating systems: /usr/opt/db2_08_01
- For all other UNIX-based operating systems: /opt/IBM/db2/V8.1

Related concepts:

- "DB2 documentation and help" on page 287

Related tasks:

- "Printing DB2 books from PDF files" on page 304
- "Ordering printed DB2 books" on page 304
- "Invoking contextual help from a DB2 tool" on page 305

Printing DB2 books from PDF files

You can print DB2 books from the PDF files on the *DB2 PDF Documentation* CD. Using Adobe Acrobat Reader, you can print either the entire book or a specific range of pages.

Prerequisites:

Ensure that you have Adobe Acrobat Reader installed. If you need to install Adobe Acrobat Reader, it is available from the Adobe Web site at www.adobe.com

Procedure:

To print a DB2 book from a PDF file:

1. Insert the *DB2 PDF Documentation* CD. On UNIX operating systems, mount the DB2 PDF Documentation CD. Refer to your *Quick Beginnings* book for details on how to mount a CD on UNIX operating systems.
2. Open index.htm. The file opens in a browser window.
3. Click on the title of the PDF you want to see. The PDF will open in Acrobat Reader.
4. Select **File → Print** to print any portions of the book that you want.

Related concepts:

- "DB2 Information Center" on page 288

Related tasks:

- "Mounting the CD-ROM (AIX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the CD-ROM (HP-UX)" in the *Quick Beginnings for DB2 Servers*
- "Mounting the CD-ROM (Linux)" in the *Quick Beginnings for DB2 Servers*
- "Ordering printed DB2 books" on page 304
- "Mounting the CD-ROM (Solaris Operating Environment)" in the *Quick Beginnings for DB2 Servers*

Related reference:

- "DB2 PDF and printed documentation" on page 299

Ordering printed DB2 books

If you prefer to use hardcopy books, you can order them in one of three ways.

Procedure:

Printed books can be ordered in some countries or regions. Check the IBM Publications website for your country or region to see if this service is available in your country or region. When the publications are available for ordering, you can:

- Contact your IBM authorized dealer or marketing representative. To find a local IBM representative, check the IBM Worldwide Directory of Contacts at www.ibm.com/planetwide
- Phone `1-800-879-2755` in the United States or `1-800-IBM-4YOU` in Canada.

- Visit the IBM Publications Center at http://www.ibm.com/shop/publications/order. The ability to order books from the IBM Publications Center may not be available in all countries.

At the time the DB2 product becomes available, the printed books are the same as those that are available in PDF format on the *DB2 PDF Documentation CD*. Content in the printed books that appears in the *DB2 Information Center CD* is also the same. However, there is some additional content available in DB2 Information Center CD that does not appear anywhere in the PDF books (for example, SQL Administration routines and HTML samples). Not all books available on the DB2 PDF Documentation CD are available for ordering in hardcopy.

Note: The DB2 Information Center is updated more frequently than either the PDF or the hardcopy books; install documentation updates as they become available or refer to the DB2 Information Center at http://publib.boulder.ibm.com/infocenter/db2help/ to get the most current information.

Related tasks:

- "Printing DB2 books from PDF files" on page 304

Related reference:

- "DB2 PDF and printed documentation" on page 299

Invoking contextual help from a DB2 tool

Contextual help provides information about the tasks or controls that are associated with a particular window, notebook, wizard, or advisor. Contextual help is available from DB2 administration and development tools that have graphical user interfaces. There are two types of contextual help:

- Help accessed through the **Help** button that is located on each window or notebook
- Infopops, which are pop-up information windows displayed when the mouse cursor is placed over a field or control, or when a field or control is selected in a window, notebook, wizard, or advisor and F1 is pressed.

The **Help** button gives you access to overview, prerequisite, and task information. The infopops describe the individual fields and controls.

Procedure:

To invoke contextual help:

- For window and notebook help, start one of the DB2 tools, then open any window or notebook. Click the **Help** button at the bottom right corner of the window or notebook to invoke the contextual help.

  You can also access the contextual help from the **Help** menu item at the top of each of the DB2 tools centers.

  Within wizards and advisors, click on the Task Overview link on the first page to view contextual help.
- For infopop help about individual controls on a window or notebook, click the control, then click **F1**. Pop-up information containing details about the control is displayed in a yellow window.

Note: To display infopops simply by holding the mouse cursor over a field or control, select the **Automatically display infopops** check box on the **Documentation** page of the Tool Settings notebook.

Similar to infopops, diagnosis pop-up information is another form of context-sensitive help; they contain data entry rules. Diagnosis pop-up information is displayed in a purple window that appears when data that is not valid or that is insufficient is entered. Diagnosis pop-up information can appear for:

- Compulsory fields.
- Fields whose data follows a precise format, such as a date field.

Related tasks:

- "Invoking the DB2 Information Center" on page 296
- "Invoking message help from the command line processor" on page 306
- "Invoking command help from the command line processor" on page 307
- "Invoking SQL state help from the command line processor" on page 307
- "Access to the DB2 Information Center: Concepts help"
- "How to use the DB2 UDB help: Common GUI help"
- "Setting the location for accessing the DB2 Information Center: Common GUI help"
- "Setting up access to DB2 contextual help and documentation: Common GUI help"

Invoking message help from the command line processor

Message help describes the cause of a message and describes any action you should take in response to the error.

Procedure:

To invoke message help, open the command line processor and enter:

```
? XXXnnnnn
```

where *XXXnnnnn* represents a valid message identifier.

For example, `? SQL30081` displays help about the SQL30081 message.

Related concepts:

- "Introduction to messages" in the *Message Reference Volume 1*

Related reference:

- "db2 - Command Line Processor Invocation Command" in the *Command Reference*

Invoking command help from the command line processor

Command help explains the syntax of commands in the command line processor.

Procedure:

To invoke command help, open the command line processor and enter:

```
? command
```

where *command* represents a keyword or the entire command.

For example, `? catalog` displays help for all of the CATALOG commands, while `? catalog database` displays help only for the CATALOG DATABASE command.

Related tasks:

- "Invoking contextual help from a DB2 tool" on page 305
- "Invoking the DB2 Information Center" on page 296
- "Invoking message help from the command line processor" on page 306
- "Invoking SQL state help from the command line processor" on page 307

Related reference:

- "db2 - Command Line Processor Invocation Command" in the *Command Reference*

Invoking SQL state help from the command line processor

DB2 Univerrsal Database returns an SQLSTATE value for conditions that could be the result of an SQL statement. SQLSTATE help explains the meanings of SQL states and SQL state class codes.

Procedure:

To invoke SQL state help, open the command line processor and enter:

```
? sqlstate or ? class code
```

where *sqlstate* represents a valid five-digit SQL state and *class code* represents the first two digits of the SQL state.

For example, `? 08003` displays help for the 08003 SQL state, and `? 08` displays help for the 08 class code.

Related tasks:

- "Invoking the DB2 Information Center" on page 296
- "Invoking message help from the command line processor" on page 306
- "Invoking command help from the command line processor" on page 307

DB2 tutorials

The DB2® tutorials help you learn about various aspects of DB2 Universal Database. The tutorials provide lessons with step-by-step instructions in the areas of developing applications, tuning SQL query performance, working with data warehouses, managing metadata, and developing Web services using DB2.

Before you begin:

You can view the XHTML versions of the tutorials from the Information Center at http://publib.boulder.ibm.com/infocenter/db2help/.

Some tutorial lessons use sample data or code. See each tutorial for a description of any prerequisites for its specific tasks.

DB2 Universal Database tutorials:

Click on a tutorial title in the following list to view that tutorial.

Business Intelligence Tutorial: Introduction to the Data Warehouse Center
Perform introductory data warehousing tasks using the Data Warehouse Center.

Business Intelligence Tutorial: Extended Lessons in Data Warehousing
Perform advanced data warehousing tasks using the Data Warehouse Center.

Information Catalog Center Tutorial
Create and manage an information catalog to locate and use metadata using the Information Catalog Center.

Visual Explain Tutorial
Analyze, optimize, and tune SQL statements for better performance using Visual Explain.

DB2 troubleshooting information

A wide variety of troubleshooting and problem determination information is available to assist you in using DB2® products.

DB2 documentation
Troubleshooting information can be found throughout the DB2 Information Center, as well as throughout the PDF books that make up the DB2 library. You can refer to the "Support and troubleshooting" branch of the DB2 Information Center navigation tree (in the left pane of your browser window) to see a complete listing of the DB2 troubleshooting documentation.

DB2 Technical Support Web site
Refer to the DB2 Technical Support Web site if you are experiencing problems and want help finding possible causes and solutions. The Technical Support site has links to the latest DB2 publications, TechNotes, Authorized Program Analysis Reports (APARs), FixPaks and the latest listing of internal DB2 error codes, and other resources. You can search through this knowledge base to find possible solutions to your problems.

Access the DB2 Technical Support Web site at http://www.ibm.com/software/data/db2/udb/winos2unix/support

DB2 Problem Determination Tutorial Series
Refer to the DB2 Problem Determination Tutorial Series Web site to find information on how to quickly identify and resolve problems you might encounter while working with DB2 products. One tutorial introduces you to the DB2 problem determination facilities and tools available, and helps you decide when to use them. Other tutorials deal with related topics, such

as ″Database Engine Problem Determination″, ″Performance Problem Determination″, and ″Application Problem Determination″.

See the full set of DB2 problem determination tutorials on the DB2 Technical Support site at http://www.ibm.com/software/data/support/pdm/db2tutorials.html

Related concepts:

- "DB2 Information Center" on page 288
- "Introduction to problem determination - DB2 Technical Support tutorial" in the *Troubleshooting Guide*

Accessibility

Accessibility features help users with physical disabilities, such as restricted mobility or limited vision, to use software products successfully. The following list specifies the major accessibility features in DB2® Version 8 products:

- All DB2 functionality is available using the keyboard for navigation instead of the mouse. For more information, see "Keyboard input and navigation."
- You can customize the size and color of the fonts on DB2 user interfaces. For more information, see "Accessible display."
- DB2 products support accessibility applications that use the Java™ Accessibility API. For more information, see "Compatibility with assistive technologies" on page 310.
- DB2 documentation is provided in an accessible format. For more information, see "Accessible documentation" on page 310.

Keyboard input and navigation

Keyboard input

You can operate the DB2 tools using only the keyboard. You can use keys or key combinations to perform operations that can also be done using a mouse. Standard operating system keystrokes are used for standard operating system operations.

For more information about using keys or key combinations to perform operations, see Keyboard shortcuts and accelerators: Common GUI help.

Keyboard navigation

You can navigate the DB2 tools user interface using keys or key combinations.

For more information about using keys or key combinations to navigate the DB2 Tools, see Keyboard shortcuts and accelerators: Common GUI help.

Keyboard focus

In UNIX® operating systems, the area of the active window where your keystrokes will have an effect is highlighted.

Accessible display

The DB2 tools have features that improve accessibility for users with low vision or other visual impairments. These accessibility enhancements include support for customizable font properties.

Font settings

You can select the color, size, and font for the text in menus and dialog windows, using the Tools Settings notebook.

For more information about specifying font settings, see Changing the fonts for menus and text: Common GUI help.

Non-dependence on color

You do not need to distinguish between colors in order to use any of the functions in this product.

Compatibility with assistive technologies

The DB2 tools interfaces support the Java Accessibility API, which enables you to use screen readers and other assistive technologies with DB2 products.

Accessible documentation

Documentation for DB2 is provided in XHTML 1.0 format, which is viewable in most Web browsers. XHTML allows you to view documentation according to the display preferences set in your browser. It also allows you to use screen readers and other assistive technologies.

Syntax diagrams are provided in dotted decimal format. This format is available only if you are accessing the online documentation using a screen-reader.

Related concepts:

- "Dotted decimal syntax diagrams" on page 310

Related tasks:

- "Keyboard shortcuts and accelerators: Common GUI help"
- "Changing the fonts for menus and text: Common GUI help"

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the Information Center using a screen reader.

In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- ? means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.
- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP will be applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.
- * means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one

data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

- + means a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times; that is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Related concepts:

- "Accessibility" on page 309

Related tasks:

- "Keyboard shortcuts and accelerators: Common GUI help"

Related reference:

- "How to read the syntax diagrams" in the *SQL Reference, Volume 2*

Common Criteria certification of DB2 Universal Database products

DB2 Universal Database is being evaluated for certification under the Common Criteria at evaluation assurance level 4 (EAL4). For more information about Common Criteria, see the Common Criteria web site at: http://niap.nist.gov/cc-scheme/.

Appendix B. Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country/region or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country/region where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product, and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs, in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. *_enter the year or years_*. All rights reserved.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both, and have been used in at least one of the documents in the DB2 UDB documentation library.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Information Integrator
DB2 Query Patroller
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the documents in the DB2 UDB documentation library:

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Index

Special characters

Numerics

A

B

C

D

E

F

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Contacting IBM

In the United States, call one of the following numbers to contact IBM:

- 1-800-IBM-SERV (1-800-426-7378) for customer service
- 1-888-426-4343 to learn about available service options
- 1-800-IBM-4YOU (426-4968) for DB2 marketing and sales

In Canada, call one of the following numbers to contact IBM:

- 1-800-IBM-SERV (1-800-426-7378) for customer service
- 1-800-465-9600 to learn about available service options
- 1-800-IBM-4YOU (1-800-426-4968) for DB2 marketing and sales

To locate an IBM office in your country or region, check IBM's Directory of Worldwide Contacts on the web at http://www.ibm.com/planetwide

Product information

Information regarding DB2 Universal Database products is available by telephone or by the World Wide Web at http://www.ibm.com/software/data/db2/udb

This site contains the latest information on the technical library, ordering books, product downloads, newsgroups, FixPaks, news, and links to web resources.

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-IBM-CALL (1-800-426-2255) to order products or to obtain general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, go to the IBM Worldwide page at www.ibm.com/planetwide

Part Number: CT2TUNA

Printed in Ireland.

(1P) P/N: CT2TUNA

SC09-4825-01

Spine information:

IBM® DB2 Universal Database™ Building and Running Applications Version 8.2