IBM® DB2® Universal Database

Developing Enterprise Java Applications Using DB2 Version 8

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

Contents

Developing Enterprise Java™ Applications Using DB2® Version 8

Introduction

DB2 UDB supports all the key Internet standards, making it an ideal database for use on the Web. DB2 UDB supports WebSphere®, Java, and XML technology, which make it easy for you to deploy your e-business applications. DB2 Version 8 also adds self-managing and resource tuning (SMART) database technology to enhance the automation of administration tasks.

DB2 Universal Database™ supports many types of Java programs. It provides driver support for client applications and applets written in Java using JDBC. It also provides support for embedded SQL for Java (SQLJ), Java user-defined functions (UDFs), and Java stored procedures.

This paper discusses the Java application development environment provided by the DB2 UDB Universal Developer's Edition Version 8 (UDE).

Java enablement

DB2 JDBC technology

According to the JDBC specification, there are four types of JDBC driver architectures:

- Type 1 - drivers that implement the JDBC API as a mapping to another data access API, such as Open Database Connectivity (ODBC). Drivers of this type are generally dependent on a native library, which limits their portability. The JDBC-ODBC Bridge driver is an example of a Type 1 driver.
- Type 2 - drivers that are written partly in the Java programming language and partly in native code. The drivers use a native client library specific to the data source to which they connect. Again, because of the native code, their portability is limited.
- Type 3 - drivers that use a pure Java client and communicate with a middleware server using a database-independent protocol. The middleware server then communicates the client's requests to the data source.
- Type 4 - drivers that are pure Java and implement the network protocol for a specific data source. The client connects directly to the data source.

DB2 Version 8 provides Type 2, Type 3 and Type 4 JDBC drivers. DB2 Version 8 Type 2 and Type 3 drivers continue to use the DB2 CLI (Call Level Interface)

interface to communicate to DB2 UDB servers (OS/390®, UNIX®, Windows®, Linux, and OS/400®). The JDBC Type 2 and Type 3 drivers are provided in the `db2java.zip` file located in the `sqllib\java` directory. DB2 Version 8 adds a new DB2 JDBC Universal Driver (Type 4), which uses the Distributed Relational Database Architecture™ (DRDA®) protocol for client/server communications. This new driver is provided in the `db2jcc.jar` file in the `sqllib\java` directory.

You do not need to execute the **usejdbc** script in DB2 Version 8.

DB2 JDBC application driver (Type 2)

The DB2 JDBC application (Type 2) driver (Figure 1) enables Java applications to make calls to DB2 through JDBC. Calls to the JDBC application driver are translated to Java native methods. The Java applications that use this driver must run on a DB2 client, through which JDBC requests flow to the DB2 server. A DB2 Connect™ Version 8 license/installation is required to access DB2 for OS/390 databases.

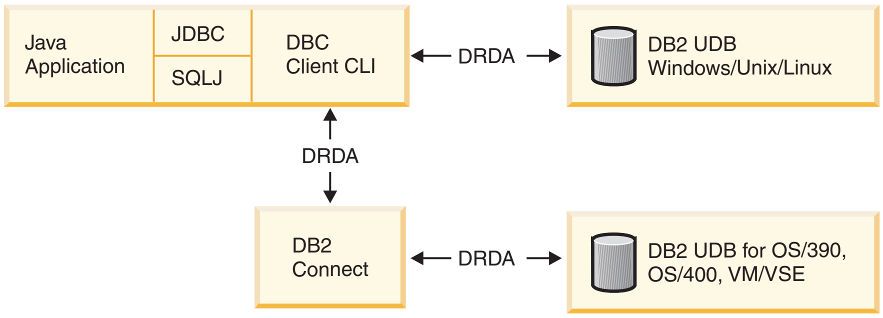


Figure 1. DB2 Java Application Architecture

The DB2 JDBC application (Type 2) driver is included in the **COM.ibm.db2.jdbc.app** package.

JDBC 1 Connections

The implementation classes for establishing a connection to DB2 UDB servers include:

- COM.ibm.db2.jdbc.app.DB2Driver.

JDBC 2 Connections

The implementation classes for establishing a connection to DB2 UDB servers include:

- COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource
- COM.ibm.db2.jdbc.DB2DataSource
- COM.ibm.db2.jdbc.DB2XADataSource.

To configure the application to use the datasource implemented by DB2 JDBC application (Type 2) driver, the following information is required:

- databaseName: the database name
- user: the userid used to connect to the database
- password: the password used to connect to the database

DB2 Thin Client Driver JDBC and SQLJ

DB2 UDB Version 8 provides a Type 3 and a Type 4 "thin" driver for JDBC and SQLJ applications. This figure depicts both driver architectures.

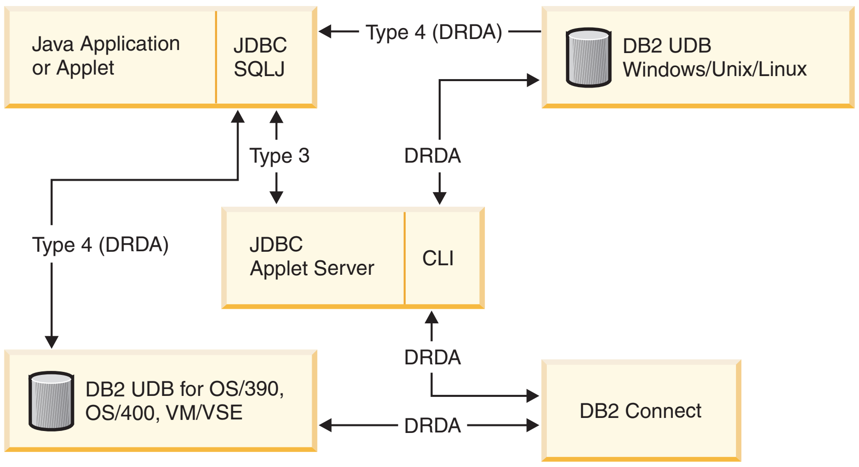

Java Application or Applet
JDBC SQLJ
Type 4 (DRDA)
DB2 UDB Windows/Unix/Linux
Type 3
DRDA
Type 4 (DRDA)
JDBC Applet Server
CLI
DRDA
DB2 UDB for OS/390, OS/400, VM/VSE
DB2 Connect
DRDA

Figure 2. DB2 JDBC/SQLj Thin Driver Architecture

DB2 JDBC Type 3 driver

The DB2 JDBC Type 3 driver, also known as the applet or net driver, consists of a JDBC client and a JDBC server. The DB2 JDBC applet driver can be loaded by the Web browser along with the applet or the applet driver can be used in standalone Java applications. When the applet requests a connection to a DB2 database, the applet driver opens a TCP/IP socket to the DB2 JDBC applet server on the machine where the Web server is running.

After a connection is set up, the applet driver sends each of the subsequent database access requests from the applet to the JDBC server through the TCP/IP connection. The JDBC server then makes corresponding DB2 calls to perform the task. Upon completion, the JDBC server sends the results back to the JDBC client through the connection. The JDBC server process is db2jd.

The DB2 JDBC Type 3 driver is included in the **COM.ibm.db2.net** package.

JDBC 1 Connections

The implementation classes for establishing a connection to DB2 UDB servers include:

- COM.ibm.db2.jdbc.net.DB2Driver.

JDBC 2 Connections

The implementation classes for establishing a connection to DB2 UDB servers include:

- COM.ibm.db2.net.DB2ConnectionPoolDataSource
- COM.ibm.db2.net.DB2DataSource

To configure the application to use the datasource implemented by DB2 JDBC Type 3 driver, the following information is required:

- databaseName: the database name
- serverName: the server name where the JDBC applet server process resides
- portNumber: the port number used by the JDBC applet server process (6789 default)
- user: the userid used to connect to the database
- password: the password used to connect to the database

There is no Java Transaction API (JTA) support provided by the DB2 JDBC Type 3 driver. JTA provides an interface to accomplish distributed (2 phase commit) transactions.

IBM® DB2 JDBC Universal Driver (Type 4 – New in Version 8)

Many new features and enhancements have been made to the JDBC drivers in DB2 version 8. This new IBM JDBC Universal Driver is based on an open distributed protocol, known as Distributed Relational Database Architecture (DRDA) and is compatible with all DB2 server platforms (UNIX, Windows, Linux, z/OS™) with appropriate DRDA Application Server (AS) level support, and prerequisite stored procedures.

Features unique to the new IBM DB2 JDBC Universal Driver include:

- Updateable ResultSet support
- Improved security for DB2 authentication
- Improved Java SQL error information
- Dynamic SQL statement performance improvements
- Programmatic tracing facilities

To use the JDBC Universal Driver, the db2jcc.jar must be included in the Java CLASSPATH environment variable.

DB2 JDBC Universal Driver Type 4 driver is included in the **com.ibm.db2.jcc** package.

JDBC 1 Connections

The DB2 UDB version 8 server will use a TCP/IP connection, specified in the DB2 DBM configuration file (SVCENAME), to communicate with the new DB2 JDBC Universal Driver.

The implementation classes for establishing a connection to DB2 UDB servers include:

- com.ibm.db2.jcc.DB2Driver

Connection objects can be created using the following URL:
jdbc:db2://server:port/database

JDBC 2 Connections

The implementation classes for establishing a connection to DB2 UDB servers include:

- com.ibm.db2.jcc.DB2SimpleDataSource

To configure the application to use the DataSource implemented by DB2 JDBC Universal Driver (Type 4), the following information is required:

- databaseName: the database name
- user: the userid used to connect to the database
- password: the password used to connect to the database
- driverType: the type of the driver used (4) (required)
- serverName: the TCP/IP address or host name for the DRDA server
- portNumber: the TCP/IP port number where the DRDA server listens for connection requests to this data source

Refer to Appendix A for additional DB2 JDBC Universal Driver (Type 4) driver information.

SQLJ support (New Architecture in Version 8)

DB2 SQLJ support enables you to build and run SQLJ applets and applications. These Java programs contain embedded SQL statements that are precompiled and bound to a DB2 UDB database.

The SQLJ standard has three components: a translator, customizer, and a run-time environment. The translator produces Java code based on the embedded SQL statements within a source SQLJ program. A binary representation of the SQL statements is created in a separate serialized profile (.ser file). Static SQL packages are created when the profile is customized using the db2sqljcustomize command. SQLJ applications require the the db2jcc.jar file, and SQLJ program preparation also requires the sqlj.zip file.

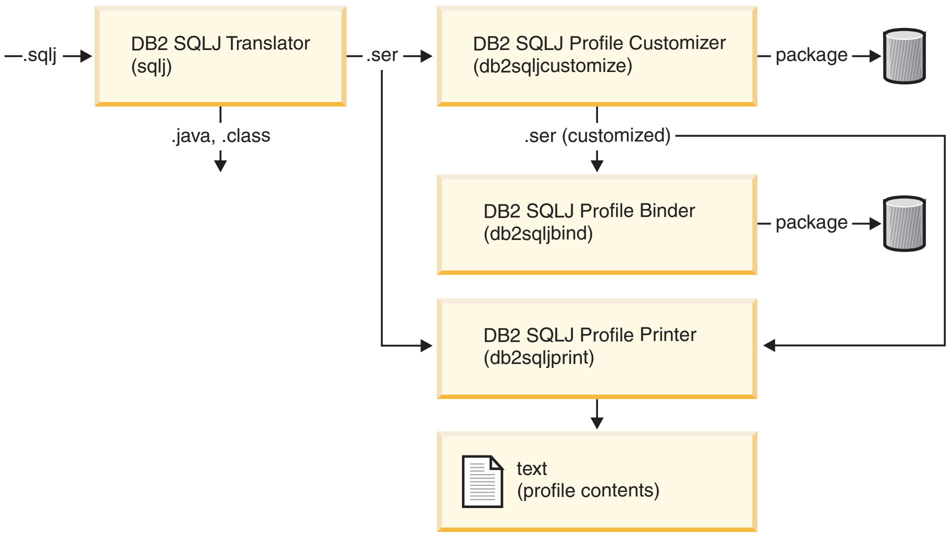

.sqlj
DB2 SQLJ Translator (sqlj)
.java, .class
.ser
DB2 SQLJ Profile Customizer (db2sqljcustomize)
package
.ser (customized)
DB2 SQLJ Profile Binder (db2sqljbind)
package
DB2 SQLJ Profile Printer (db2sqljprint)
text (profile contents)

Figure 3. SQLJ Application Development

SQLJ provides:

- A static package level security model
- A static SQL engine performance (e.g. SELECT xxx INTO :hv1, :hv2)
- Increased development productivity as compared to JDBC, especially if an application is being ported from an existing embedded SQL architecture (C, COBOL, etc.)

DB2 Version 8 provides the following SQLJ utilities, as shown in Figure 3:

- sqlj - IBM SQLJ Translator. It translates an .sqlj source file and creates a serialized profile and a program.
- db2sqljcustomize - customizer and online-checker. It creates a DB2 customization for the serialized profile, optionally online-checks SQL statements that can be dynamically prepared, and optionally (by default) binds the DB2 packages for this program.
- db2sqljbind - standalone binder. It binds a previously customized SQLJ profile to a database.
- db2sqljprint - prints contents of a DB2 customized profile.

The serialized profile that is output from the sqlj translator must be customized before the sqlj program can execute static SQL against DB2 at runtime. Without customization the application will dynamically execute the SQL statements contained in the profile.

Java 2 Platform, Enterprise Edition (J2EE)

Java 2 Platform, Enterprise Edition (J2EE) defines the standard for developing multi-tier enterprise applications. The J2EE platform manages the infrastructure and supports the Web services to enable development of secure, robust and interoperable business applications. J2EE not only takes advantages of many features of the Java 2 Platform, Standard Edition, such as ″Write Once, Run Anywhere″ portability, JDBC API for database access, but also adds full support for Enterprise JavaBeans™ (EJBs) components, Java Servlets, JavaServer Pages (JSPs) and XML technology. With simplicity, portability, scalability and legacy integration, J2EE technology and its component-based model simplify enterprise development and deployment.

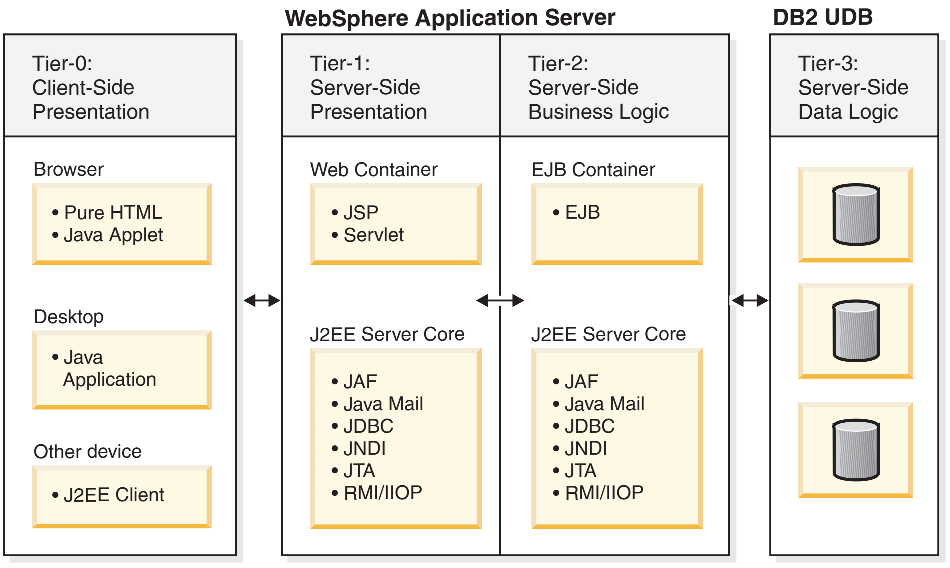


Figure 4. J2EE application model

The J2EE application model divides enterprise applications into three fundamental parts: components, containers, and connectors. Components are the key focus of application developers, while system vendors implement containers and connectors to conceal complexity and promote portability. Containers intercede between clients and components, providing services transparently to both, including transaction support and resource pooling. Container mediation allows many component behaviors to be specified at deployment time, rather than in program code. Connectors sit beneath the J2EE platform, defining a portable service API to plug into existing enterprise vendor offerings. Connectors promote flexibility by enabling a variety of implementations of specific services.

The DB2 JDBC Type 2 driver is J2EE certified for use with WebSphere Application Server, which means it conforms to J2EE specifications.

Java Technology within DB2 UDB Version 8

DB2 Java Stored Procedures and User Defined Functions (UDFs) (new architecture for DB2 Version 8)

You can create stored procedures and SQL language extensions, known as user defined functions, in Java just as you would in other languages. Stored procedures need to be registered within the database using the CREATE PROCEDURE statement. Java based user defined functions can be registered using the CREATE FUNCTION statement. You can then invoke the stored procedure from your application using any supported client API (such as JDBC, ODBC, SQLJ). The Java based user-defined function can be invoked using any SQL statement. The stored procedure or user defined function can be executed within its own address space (FENCED) or within the same address space as DB2 UDB (NOT FENCED).

In DB2 UDB version 8, Java stored procedures and user defined functions can now be registered as THREADSAFE or NOT THREADSAFE. The THREADSAFE option for Java procedures/functions will result in the use of a single JVM (Java Virtual Machine) on the DB2 UDB Server. The default for Java procedures/functions in version 8 is THREADSAFE and therefore less memory is required for concurrent Java stored procedures and performance has also been improved.

JDK/JRE Support

Java Runtime Environment (JRE) 1.3.1 is required for running applications and tools on all platforms , except HP-UX, where JRE 1.4 is required. For application development, Java Development Kit (JDK) is required. JRE/JDK 1.3.1 is required on all platforms, except 64-bit HP-UX and 64-bit Solaris, where 1.4 is required.

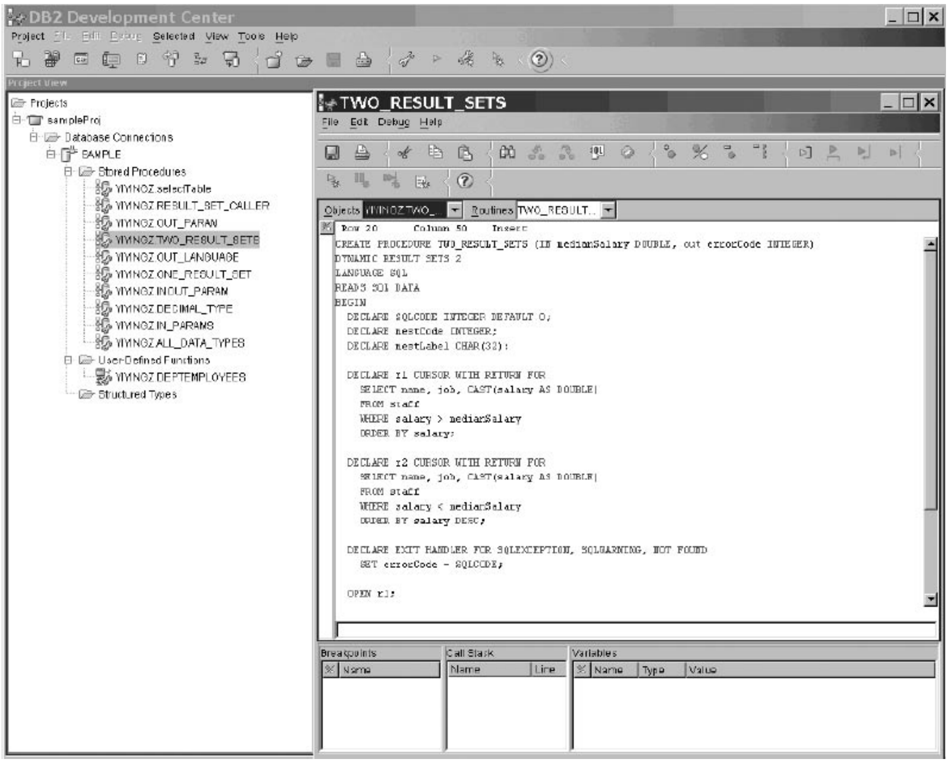
DB2 Java Application Development Environment

The DB2 Development Center, a new feature Version 8 to simplify the task of creating Java stored procedures and UDFs. WebSphere Studio is an integrated development environment (IDE) that enables you to build, test, and deploy Java applications to WebSphere Application Server and DB2 Universal Database. WebSphere Application Server provides a robust deployment environment for e-business applications. Its components let you build and deploy personalized, dynamic Web content quickly and easily. In this section, the Java application tools will be discussed.

DB2 Development Center (New for Version 8)

The DB2 Development Center is a replacement for the Stored Procedure Builder. It offers application developers facilities to build, debug, test and deploy Java stored procedures and user defined functions.

It can also be used to build, test, and deploy table functions that read MQSeries® messages, access OLE DB data sources and extract data from XML documents. The DB2 Development Center provides a standalone environment for developers. It can be used along side a complete Java IDE such as WebSphere studio.



```
Row 20      Column 50     Insert
CREATE PROCEDURE TWO_RESULT_SETS (IN medianSalary DOUBLE, out errorCode INTEGER)
DYNAMIC RESULT SETS 2
LANGUAGE SQL
READS SQL DATA
BEGIN
  DECLARE SQLCODE INTEGER DEFAULT 0;
  DECLARE nestCode INTEGER;
  DECLARE nestLabel CHAR(32);

  DECLARE r1 CURSOR WITH RETURN FOR
    SELECT name, job, CAST(salary AS DOUBLE)
    FROM staff
    WHERE salary > medianSalary
    ORDER BY salary;

  DECLARE r2 CURSOR WITH RETURN FOR
    SELECT name, job, CAST(salary AS DOUBLE)
    FROM staff
    WHERE salary < medianSalary
    ORDER BY salary DESC;

  DECLARE EXIT HANDLER FOR SQLEXCEPTION, SQLWARNING, NOT FOUND
    SET errorCode = SQLCODE;

  OPEN r1;
```

Figure 5. DB2 Development Center

WebSphere Studio

The IBM WebSphere Studio family is a series of products built on a common IBM WebSphere Studio Workbench. The set of WebSphere Studio products is IBM's replacement technology for VisualAge® for Java. This workbench is IBM's implementation of the open-source Eclipse platform for application development tools. Each product in the WebSphere Studio family offers the same integrated development environment (IDE) and a common base of tools, for example for Java and Web development. The difference between these products lie in which plug-in tools are available in each configuration.

WebSphere Studio is a single, comprehensive development environment designed to meet all of your development needs -- from Web interfaces to server-side applications, from individual development to advanced team environments, from Java™ development to application integration. Available in a number of configurations, with extensions from IBM and other vendors, the

WebSphere Studio family enables developers to use a single development environment designed to meet their specific development needs.

WebSphere Studio offers open standards, tool integration, and flexibility, including the ability to tie in existing applications. These are only some of the benefits that the WebSphere Studio product family delivers:

- Open standards: All the products in WebSphere Studio family are built on WebSphere Studio Workbench, which is IBM's implementation of the Eclipse platform. Eclipse is an open-source project that provides a common platform and set of APIs for creating plug-in development tools. For more information about the Eclipse project, see www.eclipse.org.
- Vertical and horizontal integration: Every WebSphere Studio product that is built on the workbench will offer tools that are already integrated, freeing you to focus on building applications rather than on integrating tools.
- Role-based development with consistent look and feel.
- Maximum programming performance.
- Support for J2EE business topologies.

WebSphere Studio Site Developer: WebSphere Studio Site Developer meets the needs of content authors, graphic artists, programmers, and Web developers. This integrated tool makes it easy to collaboratively create, assemble, publish, deploy, and maintain dynamic, interactive Web applications. You can quickly build and test business logic, and enhance presentation artifacts with built-in Web creation tools inside this IDE, before deploying on a production server.

WebSphere Studio Site Developer allows Web developers to create and deploy dynamic e-business applications quickly and efficiently. It also provides full Web services and XML development environments. WebSphere Studio Site Developer seamlessly interacts with many third party tools.

IBM WebSphere Application Server provides packaging support for both Web and J2EE applications. It interacts with either IBM WebSphere Application Server or Apache Tomcat to publish Web applications directly to the application server. WebSphere Studio Site Developer is the most comprehensive Web authoring tool set available for creating compelling, advanced Web sites.

You can develop Web applications that employ the following technologies:

- JavaServer Pages (JSP) - A simple, fast, and consistent way to extend Web server functionality and create dynamic Web content. JSP pages enable rapid development of Web applications that are server and platform-independent.

- Servlets - Server applications that execute within a Web application. WebSphere Studio Site Developer supports the Java Servlet specification.
- Web services - Self-contained, modular applications that can be described, published, located, and invoked over the Internet or within intranets.

WebSphere Studio Application Developer: WebSphere Studio Application Developer includes all the features of WebSphere Studio Site Developer, plus additional plug-in tools for developers of complex, enterprise-wide J2EE applications. In addition to the HTML and JSP development tools mentioned above, WebSphere Studio Application Developer provides industry-leading support for building, testing, and deploying EJB components, plus integration components such as Java Connector Architecture (JCA)-based application adapters. It also optimized for the WebSphere software platform.

WebSphere Studio Application Developer provides profiling and logging tools so that you can detect application performance problems early in the development cycle. You can build and deploy custom application adapters to integrate with back-end systems, thereby increasing productivity by reusing existing resources. Furthermore, the built-in test environment for WebSphere Application Server and advanced tools for code generation help shorten the test cycle.

WebSphere Studio Application Developer is a complete Java, Web, Web services, Enterprise JavaBeans (EJB), and XML development environment. It provides wizards and other tools to enable rapid development of Web services applications. The Web services development tools provided in WebSphere Studio Application Developer are based on open, cross-platform standards: Universal Description Discovery and Integration (UDDI), Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL).

WebSphere Studio Application Developer facilitates the following processes to assist with building and deploying Web services-enabled applications:

- Create or transform. Create Web services from existing artifacts, such as Java beans and XML document.
- Build. Wrap existing artifacts as SOAP and HTTP accessible services and describe them in WSDL. The Web services wizards assist you in generating a SOAP proxy to Web services described in WSDL and in generating bean skeletons to classes.
- Deploy. Deploy Web services in the WebSphere Application Server or Tomcat test environments.
- Develop. Generate sample applications to assist you in creating a Web service client application.
- Test. Test Web services running locally or remotely.

- Publish. Publish Web services to the UDDI business registry, advertising your Web services so that other businesses can access them.
- Discover. Browse the UDDI business registry to locate existing Web services for integration.

WebSphere Studio Application Developer provides a comprehensive XML development environment that includes tools for building Document Type Definitions (DTDs), XML schemas, and XML files. It also supports integration of relational data and XML. You can use the Relational Database (RDB) to XML Mapping Editor to easily map relational data to XML formats. The editor can map columns in one or more relational tables to elements and attributes in an XML document. It can generate a Document Access Definition (DAD) file, which is used by DB2 XML Extender to either compose XML documents from existing DB2 data, or to decompose XML documents into DB2 data.

WebSphere Studio Application Developer contains a relational database environment to create and manipulate the data design for projects. It is an environment for exploring, importing, designing and querying databases. The SQL Query Builder provides a visual interface for creating and executing SQL statements.

WebSphere Studio Application Developer provides testing and publishing tools for testing enterprise applications. It provides a unit test environment for testing JSPs, servlets, and HTML files. It also provides the capability to configure other local or remote servers for integrating testing and debugging of Web and EJB applications. The server can be WebSphere Application Server, Apache Tomcat, or TCP/IP Monitoring Server.

DB2 UDB plug-ins for WebSphere Studio (New for Version 8): The DB2 stored procedure and UDF builder component provides wizards and tools for creating and working with stored procedures and user-defined functions (UDFs) for use with DB2 Universal Database. When you create an application that accesses DB2 data, you can improve your application's performance by incorporating stored procedures and UDFs that are registered with the database server.

You can reduce network traffic and make better use of shared business logic. After you create a routine (stored procedure or UDF) with one of the wizards, you can modify it in the text editor. When you are satisfied, you can build it and register it on the DB2 server. After the routine is on the server, you can execute it and run the SQL statements that are included in the routine. When you run a routine, you can look in the Output view to see relevant information such as messages, parameters (input and output), and result sets that are returned. If you are not satisfied with the results, you can continue to modify and rebuild the routine until it returns the desired results.

Use this component to create the following types of routines:

- SQL and Java stored procedures
- SQL UDFs
- UDFs that read or receive messages from MQSeries message queues

You can:

- Create a new routine using a wizard
- Modify existing routines using the text editor
- Build (register) routines on the DB2 database server
- Run (execute) routines on the DB2 database server
- View result sets, messages, and parameters in the Output view
- Drop routines from the database

WebSphere Application Server

WebSphere Application Server is the foundation of the WebSphere software platform, WebSphere Application Server provides a rich, e-business application deployment environment with a complete set of application services including capabilities for transaction management, security, clustering, performance, availability, connectivity and scalability. It is a compliant Java 2 Platform, Enterprise Edition (J2EE) server.

DB2/WebSphere connection pooling

Connection pooling support is provided with WebSphere Application Server. WebSphere connection pooling is the implementation of the JDBC 2.0 optional package API specification. Connection pooling spreads the connection overhead across several user requests by establishing a pool of connections which servlets can use. After the initial resources are spent to produce the connections in the pool, additional connect/disconnect overhead is insignificant because the existing connections are reused repeatedly.

DB2 maintains a pool of agent process/threads on the DB2 UDB server to satisfy client requests for database resources. WebSphere Application Server (WAS) connection pools are maintained independently from the DB2 server agent pool. Ensure that the allocated WAS connection pools are set in conjunction with the DB2 server agent configuration.

Statement Caching

WebSphere Application Server provides a prepared statement (PreparedStatement objects only) cache. This cache should be configured in conjunction with the DB2 package cache (PCKCACHESZ) for dynamic/static SQL statement processing. Depending on application data access patterns, the WAS prepared statement cache can improve application performance.

Session Persistence

Persistent sessions are essential for using HTTP sessions for failover facility. When an application server receives a request associated with a session ID that it currently does not have in memory, it can obtain the required session state by accessing the external store (database or memory-to-memory). If persistent sessions are not enabled, an application server cannot access session information for HTTP requests that are sent to servers other than the one where the session was originally created.

The WebSphere Application Server (WAS) stores session states in a DB2 database to provide fault tolerance for Web applications. If an application server goes offline, the state of its current sessions is still available in the DB2 database. This enables other application servers to continue processing. The DB2 tablespace page size configured within WAS. The default row size is 4K. WAS will exploit DB2's larger page size (32KB) for persisting session objects. This optimization results in improved application performance.

Enterprise JavaBeans

As shown in Figure 4 on page 7, there are two types of application components known as Enterprise JavaBeans (EJB): session beans and entity beans. Persistent data components are created using entity beans, while application logic components are created using session beans.

Session beans encapsulate temporary data associated with a particular client. Entity beans encapsulate permanent data that is stored in DB2. The persistence service ensures that the data associated with entity beans is properly synchronized with their corresponding data in the data source. To accomplish this task, the persistence service works with the transaction service to insert, update, extract, and remove data from the data source at the appropriate times.

There are two types of entity beans: those with container-managed persistence (CMP) and with bean-managed persistence (BMP). WebSphere Studio provides the ideal environment for creating CMP EJBs. Once the EJBs have been mapped to corresponding DB2 tables/columns, WebSphere Studio generates the required JDBC code. This generated JDBC must be handled by the EJB developer if BMP EJBs are to be used.

Isolation level settings specify various degrees of runtime data integrity provided by the corresponding database.

Table 1. Mapping of isolation levels between J2EE and DB2

J2EE isolation level	DB2 isolation level	Description
Serializable	Repeatable read	Prohibits dirty reads, nonrepeatable reads and phantom reads
Repeatable reads	Read stability	Prohibits dirty reads and nonrepeatable reads, but it allows phantom reads
Read committed	Cursor stability	Prohibits dirty reads, but allows nonrepeatable reads and phantom reads
Read uncommitted	Uncommitted read	Allows dirty reads, nonrepeatable reads and phantom reads

EJB Sample (New for Version 8): There is a new DB2 UDB EJB based sample provided with DB2 Version 8. The sample application includes a Web client which accesses the DB2 sample database (db2sampl) using a JSP/servlet interface to a Session/Entity EJB component. The sample application is contained in the AccessEmployee.ear file, which is available in the <DB2_root>\samples\java\Websphere\ directory on Windows platform and the <DB_instance_home>/sqllib/samples/java/Websphere directory on UNIX.

A full description of the EJB sample application and deployment information is provided in the README file located in the `samples/Java` directory.

DB2 Web services: Web services technology is essentially a new programming paradigm to aid in the development and deployment of loosely coupled applications within a company or across industries.

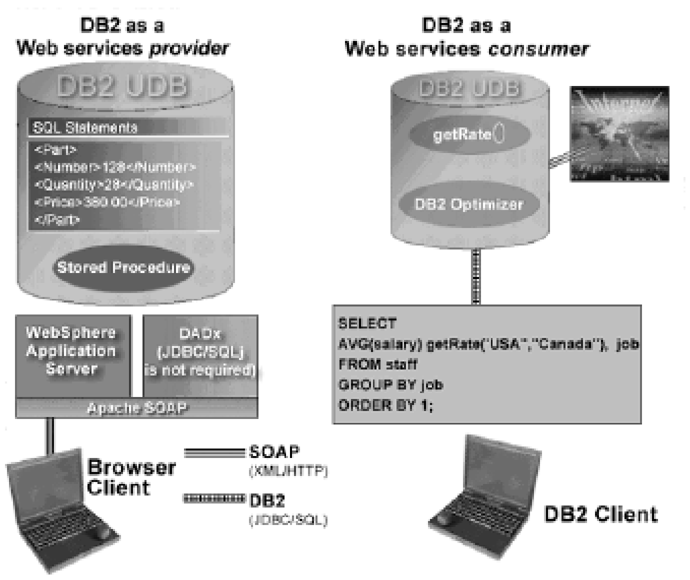


Figure 6. DB2 Web Services Overview

WSDL (Web Services Description Language) defines an XML format for describing network services as a set of endpoints operating on messages that contain either document-oriented or procedure-oriented information.

SOAP (Simple Object Access Protocol) messages can be created or consumed using any programming language. Even the transport/network layer is flexible. The vast majority of initial Web services are being deployed using HTTP protocol over TCP/IP networks, but other options are also available including WebSphere; MQ.

The Web services architecture takes into account that WSDL interfaces to an application component may change, and likely will change, over time. The Universal Description, Discovery and Integration (UDDI) specification helps distributed application developers and implementers solve these application evolution issues. There are public UDDI registries hosted on the Internet by companies such as IBM and Microsoft® (and others). These registries can be used by developers to publicize their application interfaces (as specified by WSDL); alternatively, the registries are used to find other developers' application interfaces. When a WSDL interface has changed, the developers can republish their new interface in the public UDDI registry.

DB2 as a Web Service Provider: Extensions to the Apache SOAP server environment are provided with DB2 UDB version 8. The runtime environment is known as The WORF (Web Object Runtime Framework). DB2 Web Services is an XML technology for simplify the creation of Web services that access relational databases. A DADX file is used to define the DB2 access operations.

WORF provides the runtime support for invoking DADX documents as Web services using SOAP, which is supported by WebSphere Application Server and Apache Jakarta Tomcat. WORF supports generating and storing XML documents by using the XML collection method, and it also allows stored procedures and SQL statements to be exposed as Web service operations. In addition, WORF can generate a test page, WSDL document and XML Schema Definition (XSD) files.

WebSphere Studio provides tools to create DADX files. The XML from the SQL wizard supports the creation of a DADX file from SQL queries, DAD files.

DB2 Integrated Web Services Tutorial

IBM Video Central tutorial gives a sample solution for a company that provides business services to another company over the web. This concept is known as a Business-to-Business application. The IBM Video Central tutorial demonstrates the integration of DB2 and WebSphere. Many technologies are used to design and build a fully functional application, including: IBM's WebSphere Application Server, VisualAge for Java, and the DB2 XML Extender. A set of centralized services for individual retail video stores is accessed using Simple Object Access Protocol (SOAP) (see http://www.w3.org/2000/xp/).

The IBM Video Central tutorial demonstrates the design, development, and implementation of simple data repository (insertion and modification) and query services. The query services use the existing DB2 XML Extender. The first version of the tutorial is provided using the web, and it includes: Java Servlets, JSP (Java Server Pages), and the accompanying tutorial documentation.

DB2 UDB XML Technology

DB2 XML enabled databases (New features for Version 8)

Extensible Markup Language (XML) is the accepted standard technique of data exchange between applications. An XML document is a tagged document consisting of character data and markup tags. The markup tags are definable by the author of the document. A Document Type Definition (DTD) is used to declare the required data elements and attributes for an XML document. XML Schemas can be used to further qualify the data types of the elements and attributes of an XML document. The DB2 XML Extender provides a mechanism for programs to manipulate XML data using SQL extensions.

To extend a DB2 UDB Version 8 database for XML usage simply enable the database using the **dxxadm** command. For example, **dxxadm enable_db sample**

XML documents can be stored in DB2 within a single column or as a collection, using a set of columns. The DB2 XML Extender introduces three new data types: XMLVARCHAR, XMLCLOB, and XMLFILE. The Extender provides UDFs to store, extract and update XML documents stored within a single column. Searching can be performed on the entire XML document or based on structural components using the location path, which uses a subset of the abbreviated syntax defined by the XML Path Language (XPath). Side tables can be used to improve search performance for elements or attributes that are frequently queried.

To facilitate storing XML documents as a set of columns, the DB2 XML Extender provides an administration tool to aid the designer with XML-to-relational database mapping. The Document Access Definition (DAD) is used to maintain the structural and mapping data for the XML documents. The DAD is defined and stored as an XML document, making it simple to manipulate and understand. New stored procedures are available to compose or decompose the document.

New XML features for DB2 Version 8 include:
- XML schema validation
- XML stylesheet (transformation) support
- SQL/XML enhancements including: XMLAGG, XMLATTRIBUTES, XMLELEMENT, and XML2CLOB.

WebSphere Studio - Relational database (RDB) to XML mapping editor (New)

Mapping from XML to relational data is simple using the WebSphere Studio RDB to XML mapping editor. The WebSphere Studio product is a replacement for the previous DB2 XML Extender Wizard used to create document access definition (DAD) files. You can map columns in one or more relational tables to elements and attributes in an XML document. You can generate a DAD script to either compose XML documents from existing DB2 data, or decompose XML documents into DB2 data. You can also create a test harness to test the generated DAD file.

The RDB to XML mapping editor is designed to work in conjunction with DB2 UDB Version 8 XML enabled databases. It simplifies development tasks in the following ways:

- Provides a visual interface to easily define mappings between relational data and XML elements and attributes.
- Automatically generates DAD files.
- Automatically generates a test harness.

MQSeries enablement

A set of MQSeries functions are provided with DB2 Universal Database Version 8 to allow DB2 UDB applications to interact with asynchronous messaging operations. This means that MQSeries support is available to applications written in any programming language supported by DB2 UDB. For simplicity, all examples shown in this section are SQL statements. A WebSphere application can utilize these MQSeries SQL statements.

Messaging styles

MQSeries does not mandate that the messages it transports adhere to a particular structure. XML messages typically have a self-describing message structure. Messages can also be unstructured, requiring user code to parse or construct the message content. Such messages are often semi-structured, that is, they use either byte positions or fixed delimiters to separate the fields within a message.

MQSeries supports three messaging models: datagrams, publish/subscribe (p/s), and request/reply (r/r). Messages sent as datagrams are sent to a single destination with no reply expected. In the p/s model, one or more publishers send a message to a publication service that distributes the message to one or more interested subscribers. Request/reply is similar to datagram, but the sender expects to receive a response. The supplied DB2 MQSeries functions support all three message models.

MQSeries is used in a various ways. Simple datagrams are exchanged to coordinate multiple applications, exchange information, request services, and provide notification of interesting events. The publish/subscribe style is most often used to disseminate real-time information in a timely manner. The request/reply style is generally used as a simple form of pseudo-synchronous remote procedure call (RPC). More complex models can be constructed by combining these basic styles.

DB2 UDB Version 8 provides a new MQSeries Assist wizard within the DB2 Development Center. This wizard creates a table function that reads from an MQSeries queue using the MQSeries UDFs, which are also new in DB2 UDB Version 8. The wizard can treat each MQSeries message as a delimited string or a fixed length column string. The created table function parses the string according to your specifications, and returns each MQSeries message as a row of the table function. The wizard also allows you to create a view on top of the table function and to preview an MQSeries message and the table function result.

DB2 / MQ infrastructure
In a basic configuration, as shown in Figure, an MQSeries server is located on the database server machine along with DB2 Universal Database. The MQSeries functions are available from a DB2 server and provide access to other MQSeries applications. Multiple DB2 clients can concurrently access the MQSeries functions through the database. The MQSeries operations allow DB2 applications to asynchronously communicate with other MQSeries applications. For instance, the new functions provide a simple way for a DB2 application to publish database events to remote MQSeries applications, initiate a workflow through the optional MQSeries Workflow product, or communicate with an existing application package with the optional MQSeries Integrator product.

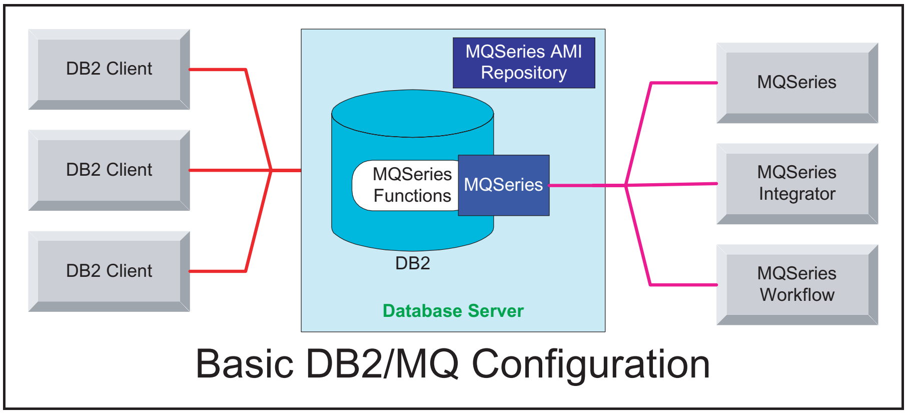


Figure 7. DB2/MQ configuration

When MQSeries support is installed as part of DB2 Universal Database, a configuration script automatically establishes a default configuration that client applications can use with no further administrative action. The default configuration enables application programmers to get started quickly and provides a simpler development interface.

Using the DB2 XML Extender and MQSeries through SQL and XML: A series of SQL User Defined Functions and Stored Procedures are supplied in DB2 enabling a DB2 client application to do the following from a single request:

- To read from MQSeries queues into DB2 XML columns
- To send and publish messages to MQseries queues from DB2 XML columns
- To send XML messages composed from relational data to MQSeries queues
- To decompose (shred) XML messages held in MQSeries queues into relational data

Net Search Extender (New features in Version 8)

The DB2 Net Search Extender uses an indexing technique, known as n-gram index, to provide a new high-speed text search extender. There are many uses for this extender in the area of Web applications, as text fields are commonly queried by end users. Finding relevant documents based on text field indexes can improve Web user satisfaction. Any columns based on CHAR, VARCHAR, or LONG VARCHAR can be indexed using an n-gram index. When the index has been created and activated, searches can be performed using a new stored procedure. Active indexes are stored in shared memory to optimize search performance.

DB2 version 8 Net Search Extender Replaces all existing Text Extenders:

- DB2 Text Information Extender V7.2
- DB2 Text Extender V7.1
- DB2 Net Search Extender V7.2

Summary

The DB2 UDB Universal Developer's Edition Version 8 product delivers all the tools you need to rapidly build and deploy applications. The package includes a full-function integrated development environment, a scalable Web application server, and DB2 UDB features such as the XML Extender. DB2 Universal Database is a scalable, industrial-strength database that will be the data management foundation for your e-business.

Additional information

For additional information, please refer to the following Web sites:

DB2 Universal Database Resources:

- http://www.software.ibm.com/data/developer
- http://www.software.ibm.com/data/db2/java

WebSphere Developer Domain:

- http://www.ibm.com/websphere/developer

Appendix A. IBM DB2 JDBC Universal Driver (Type 4)

Requirements

A Java Runtime Environment at the 1.3.1 level is required.

IBM DB2 JDBC Universal Driver (type 4) requires prerequisite stored procedures and views on the target DB2 UDB server. DB2 UDB version 8 on UNIX, Linux and Windows has the required stored procedures and views. For DB2 on OS/390, the stored procedures must be installed manually.

Information for installing these stored procedures to DB2 for OS/390 Version 6 and Version 7 is available from the http://www.ibm.com/software/data/db2/os390/spb/sprocedure/index.html website.

Since the IBM DB2 JDBC Universal Driver (Type 4) uses TCP/IP for communications the DB2 UDB target server must be configured for TCP/IP.

Determining JDBC Driver Information

You can programmatically determine which JDBC driver has been loaded by your application using the following methods:

- java.sql.DatabaseMetaData.getDriverName()
- java.sql.DatabaseMetaData.getDriverMajorVersion()
- java.sql.DatabaseMetaData.getDriverMinorVersion()
- com.ibm.db2.jcc.DB2DatabaseMetaData.getJCCDriverBuildNumber()

Trace Facilities

DB2 JDBC Universal Driver (type 4) error handling relies on the installation of a prerequisite stored procedures on the target server. DB2 UDB Version 8 servers on UNIX, Linux, and Windows will have the required stored procedures created. DB2 for OS/390 Version 6 and Version 7 must have the stored procedures installed for the DB2 JDBC Universal Driver (type 4) driver to provide additional error messages.

Tracing can be enabled using DataSource properties (logWriter, traceLevel, traceFile). JDBC 1 connections must use the java.sql.DriverManager.setLogWriter() method.

Behavior Differences

The new DB2 JDBC Universal Driver (type 4) removes any dependency on DB2 Call Level Interface (CLI). Therefore, any previous db2cli.ini settings should be considered if you previously used DB2 JDBC drivers.

New Driver Properties

Connection (Performance)

- boolean com.ibm.db2.jcc.DB2BaseDataSource.fullyMaterializeLobData - default is true
- boolean com.ibm.db2.jcc.DB2BaseDataSource.deferPrepares - default is true

Security

int com.ibm.db2.jcc.DB2BaseDataSource.securityMechanism

- final static int com.ibm.db2.jcc.DB2BaseDataSource.USER_ONLY_SECURITY
- final static int com.ibm.db2.jcc.DB2BaseDataSource. CLEAR_TEXT_PASSWORD_SECURITY

- final static int com.ibm.db2.jcc.DB2BaseDataSource.
  ENCRYPTED_PASSWORD_SECURITY
- final static int com.ibm.db2.jcc.DB2BaseDataSource.
  ENCRYPTED_USER_AND_PASSWORD_SECURITY
- final static int com.ibm.db2.jcc.DB2BaseDataSource.KERBEROS_SECURITY

String com.ibm.db2.jcc.DB2BaseDataSource.kerberosServerPrincipal

transient org.ietf.jgss.GSSCredential
com.ibm.db2.jcc.DB2BaseDataSource.gssCredential

void com.ibm.db2.jcc.DB2SimpleDataSource.setPassword (String password)

Data Sources

abstract class com.ibm.db2.jcc.DB2BaseDataSource

class com.ibm.db2.jcc.DB2SimpleDataSource

interface com.ibm.db2.jcc.DB2JccDataSource
- String getJccVersion()

New Methods

DB2 Set Client Information (application identification)
- void com.ibm.db2.jcc.DB2Connection.setDB2ClientUser (String user)
- void com.ibm.db2.jcc.DB2Connection.setDB2ClientWorkstation (String name)
- void void com.ibm.db2.jcc.DB2Connection.setDB2ClientApplicationInformation (String info)
- void com.ibm.db2.jcc.DB2Connection.setDB2ClientAccountingInformation (String info)
- String com.ibm.db2.jcc.DB2Connection.getDB2ClientUser()
- String com.ibm.db2.jcc.DB2Connection.getDB2ClientWorkstation()
- String com.ibm.db2.jcc.DB2Connection.getDB2ClientApplicationInformation()
- String com.ibm.db2.jcc.DB2Connection.getDB2ClientAccountingInformation()

Current Packageset
- void com.ibm.db2.jcc.DB2Connection.setDB2CurrentPackageSet(String packageset)
- String com.ibm.db2.jcc.DB2Connection.getDB2CurrentPackageSet()
- String com.ibm.db2.jcc.DB2BaseDataSource.currentPackageSet

Trace

- transient java.io.PrintWriter com.ibm.db2.jcc.DB2BaseDataSource.logWriter
- void com.ibm.db2.jcc.DB2Connection.setJCCLogWriter (java.io.PrintWriter printWriter)
- void com.ibm.db2.jcc.DB2Connection.setJCCLogWriter (java.io.PrintWriter printWriter, int traceLevel)
- java.io.PrintWriter com.ibm.db2.jcc.DB2Connection.getJCCLogWriter()
- int com.ibm.db2.jcc.DB2BaseDataSource.traceLevel
- String com.ibm.db2.jcc.DB2BaseDataSource.traceFile
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_NONE
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_CONNECTION_CALLS
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_STATEMENT_CALLS
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_RESULT_SET_CALLS
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DRIVER_CONFIGURATION
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_CONNECTS
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DRDA_FLOWS
  - final static int com.ibm.db2.jcc.DB2BaseDataSource. TRACE_RESULT_SET_META_DATA
  - final static int com.ibm.db2.jcc.DB2BaseDataSource. TRACE_PARAMETER_META_DATA
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DIAGNOSTICS
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_SQLJ
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_ALL

Diagnostics

- com.ibm.db2.jcc.DB2Diagnosable
  - DB2Sqlca getSqlca ()
  - java.lang.Throwable com.ibm.db2.jcc.DB2Diagnosable.getThrowable ()
  - void printTrace (java.io.PrintWriter printWriter, String messageHeader)
- com.ibm.db2.jcc.DB2Sqlca
  - String[] getSqlErrmcTokens ()
  - String getSqlErrmc ()
  - String getSqlErrp ()

  - int[] getSqlErrd ()
  - char[] getSqlWarn ()
  - int getSqlCode ()
  - String getSqlState ()
  - String getMessage ()
- com.ibm.db2.jcc.DB2ExceptionFormatter
  - static void printTrace (com.ibm.db2.jcc.DB2Sqlca sqlca, java.io.PrintWriter printWriter, String header)
  - static void printTrace (SQLException sqlca, java.io.PrintWriter printWriter, String header)
  - static void printTrace (java.lang.Throwable sqlca, java.io.PrintWriter printWriter, String header)

com.ibm.db2.jcc.DB2DatabaseMetaData
- int getJCCDriverBuildNumber()

com.ibm.db2.jcc.DB2Driver
- int getJCCBuildNumber()
- int getJCCBuildCertification()
- final static int jccTestBuild
- final static int jccBetaBuild
- final static int jccReleaseBuild
- String[] getJCCCompatibleJREVersions()

JDBC 2 Data Source and JDBC 1 Connection Properties
- int com.ibm.db2.jcc.DB2BaseDataSource.driverType
- String com.ibm.db2.jcc.DB2BaseDataSource.databaseName - specified in URL for JDBC 1 connectivity
- String com.ibm.db2.jcc.DB2BaseDataSource.description
- int com.ibm.db2.jcc.DB2BaseDataSource.portNumber - specified in URL for JDBC 1 connectivity
- String com.ibm.db2.jcc.DB2BaseDataSource.serverName - specified in URL for JDBC 1 connectivity
- String com.ibm.db2.jcc.DB2BaseDataSource.user
- int com.ibm.db2.jcc.DB2BaseDataSource.resultSetHoldability
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.HOLD_CURSORS_OVER_COMMIT
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.CLOSE_CURSORS_AT_COMMIT
- boolean com.ibm.db2.jcc.DB2BaseDataSource.fullyMaterializeLobData

- String com.ibm.db2.jcc.DB2BaseDataSource.currentPackageSet
- String com.ibm.db2.jcc.DB2BaseDataSource.planName
- int com.ibm.db2.jcc.DB2BaseDataSource.securityMechanism
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.USER_ONLY_SECURITY
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.CLEAR_TEXT_PASSWORD_SECURITY
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.ENCRYPTED_PASSWORD_SECURITY
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.ENCRYPTED_USER_AND_PASSWORD_SECURITY
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.KERBEROS_SECURITY
- String com.ibm.db2.jcc.DB2BaseDataSource.kerberosServerPrincipal
- transient org.ietf.jgss.GSSCredential com.ibm.db2.jcc.DB2BaseDataSource.gssCredential
- String com.ibm.db2.jcc.DB2BaseDataSource.traceFile
- int com.ibm.db2.jcc.DB2BaseDataSource.traceLevel
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_NONE
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_CONNECTION_CALLS
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_STATEMENT_CALLS
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_RESULT_SET_CALLS
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DRIVER_CONFIGURATION
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_CONNECTS
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DRDA_FLOWS
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_RESULT_SET_META_DATA
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_PARAMETER_META_DATA
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DIAGNOSTICS
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_SQLJ
  - final static int com.ibm.db2.jcc.DB2BaseDataSource.TRACE_ALL
- transient java.io.PrintWriter com.ibm.db2.jcc.DB2BaseDataSource.logWriter
- boolean com.ibm.db2.jcc.DB2BaseDataSource.deferPrepares

- boolean com.ibm.db2.jcc.DB2BaseDataSource.readOnly
- java.util.Properties com.ibm.db2.jcc.DB2BaseDataSource.getProperties()

Current Restrictions

- User-defined Structured Types
- Connection Pooling
- Arrays
- Distributed Transactions (JTA)
- Row Sets
- Callable Statement Batches

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

Trademarks