Database Binding Component User's Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 821–1069–05
December 2009

091221@23031

Contents

Understanding the Database Binding Component

The topics in this document provide information about Understanding the Database Binding Component.

What You Need to Know

These topics provide information about Database Binding Component.

About Database Binding Component

The Database Binding Component (DB BC) provides a comprehensive solution for configuring and connecting to databases that support Java Database Connectivity (JDBC) from within a Java Business Integration (JBI) environment. Database BC is a JBI component that provides database operations as services. JBI components acting as consumers invoke these Web Services. The Database BC is an implementation in compliance with JBI Specification 1.0.

The Database BC supports the following database artifacts to be exposed as Services.

- Table
- Prepared Statements
- Procedures
- SQL File

Components of Database Binding Component

The Database BC helps users to handle databases with flexibility. It also provide Web Services in conjunction with other OpenESB components.

The following components are part of the Database BC.

- The WSDL from Database wizard, which supports Table, Prepared Statements, Procedure, and SQL File (NetBeans plug-in).
- Custom WSDL extensions for configuring the Web Service (NetBeans plug-in).
- Database Binding Runtime component (JBI runtime component).

Supporting Features in Database Binding Component

The Database BC has all the features of the JDBC Binding Component and subset of SQL Service Engine. The Database BC supports the listed database artifacts that are exposed as Services.

- **Database Objects**
  - Table
  - View
  - Prepared Statements
  - Procedures
  - SQL File
- **Database Operations (DDL)**
  - Table
    - Insert
    - Update
    - Delete
    - Select
    - Database Polling for Inbound
      - PollPostProcessing Delete
      - PollPostProcessing Mark Column
      - PollPostProcessing Copy
      - PollPostProcessing Move
  - Prepared Statements
    - Select
    - Insert
    - Update
    - Delete
  - Procedures
- **Database Connectivity**

  Database connectivity through Java Naming and Directory Interface (JNDI) lookup
- **XSD Creation and Editing**

- Creating XML Schema Definition (XSD) through wizard for
  - Table
  - Prepared Statements
  - Procedure
  - SQL File
- **WSDL Creation and Editing**

  Creating WSDL components through a wizard with relevant binding information, service, and port definitions
  - Table
  - Prepared Statements
  - Procedures
  - SQL File
- **Other Features**
  - Transaction Control (including XA)
  - JDBC WSDL extension validations in the WSDL Editor
  - Fault Handling
- **Databases Supported**
  - **Tier 1 Databases Supported**
  - MySQL
  - Oracle

    Tier 2 Databases Supported
  - SQL Server
  - DB2
  - Sybase
- **Driver Types Supported**

  Drivers are uniquely different in what they do and the type of functions they support.
  - DataDirect 3.7 (Type 4)
  - Derby Network client driver for Derby 10.2
  - Oracle Database
    - Oracle Type 2 driver (thin) for Oracle 9i
    - Oracle Type 2 driver (thin) for Oracle 10G
    - Oracle Type 2 driver (thin) for Oracle 11G

    ojdbc14.jar,ojdbc5.jar: This driver works with Table, Prepared Statements, and Procedures.
  - MySQL Native

mysql-connector-java-5.1.5-bin.jar: This driver works with Table, Prepared Statements, Procedures, and SQL File.

- MySQL DataDirect Driver

  This driver works with Table, Prepared Statements, Procedures, and SQL File.

- Oracle Native

  This driver works with Table, Prepared Statements (supports only during Runtime), Procedures, and SQL File (supports only during Runtime).

- Oracle DataDirect Driver

  This driver works with Table, Prepared Statements, and SQL File.

- SQL SERVER 2005 Native

  sqljdbc.jar: This driver works with Table, Prepared Statements, and SQL File.

- SQL SERVER 2005 DataDirect Driver

  sqljdbc.jar: This driver works with Table, Prepared Statements, and SQL File.

- Sybase Native

  jconn3.jar: This driver works with Table, Prepared Statements, and SQL File.

- DB2 Native

  db2jcc.jar: This driver works with Table, Prepared Statements, Procedures, and SQL File.

Note – With this, both the JAR Files and the NBM Files are installed together.

- **Supported Platforms**
  - Windows
  - Solaris
  - Mac OS X
  - Linux
- **Support for OpenESB**

  NetBeans plug-in compliant with the NetBeans enterprise suite.

- **Systemic Qualities**
  - Application Configuration and Variables

    Provides support for application configuration at deployment time and runtime, that is, after an application has been packaged. Must allow for changing configuration without modifying the packaged application.

  - Logging

    Develop a consistent logging strategy across all runtime components.

- Monitoring and Management

  Provides a shared model for instrumentation, aggregation, and presentation of monitoring data related to performance, activity, and status. Needs to allow for unified monitoring across Sierra. This includes the core platform as well as components.

- Recovery

  XA recovery is a big part of this picture, but it's not everything. All components need to be able to recover gracefully from failure, including failure of other components internally and externally; and dealing with faults or errors in a manner that does not compromise message reliability.

- Dynamic Addressing

  Extend the scope of an application dynamically through dynamic addressing or invocation.

- Retry

  Qualities that can be added to an interaction with an endpoints.

- Throttling

  Qualities that can be added to an interaction with an endpoints.

- Serial Processing and Message Ordering

  Needs explicit support by all components; might benefit from conventions with respect to the status (DONE or ERROR) is sent back (that convention is related to TX and reliable messaging as well).

- Common Fault and Error Handling

  Establishes a common fault or error framework for all components. This ensures consistency in fault behavior and content.

- Transaction Propagation

  Provides a transferable construct for transaction information within the context of a given invocation (parent -> child).

- Configuration Usability

  MBean and WSDL extensibility elements consistency or usability.

Open ESB Packages that Make Up the New Database Binding Component

This runtime component is available as a **sun-database-binding**.

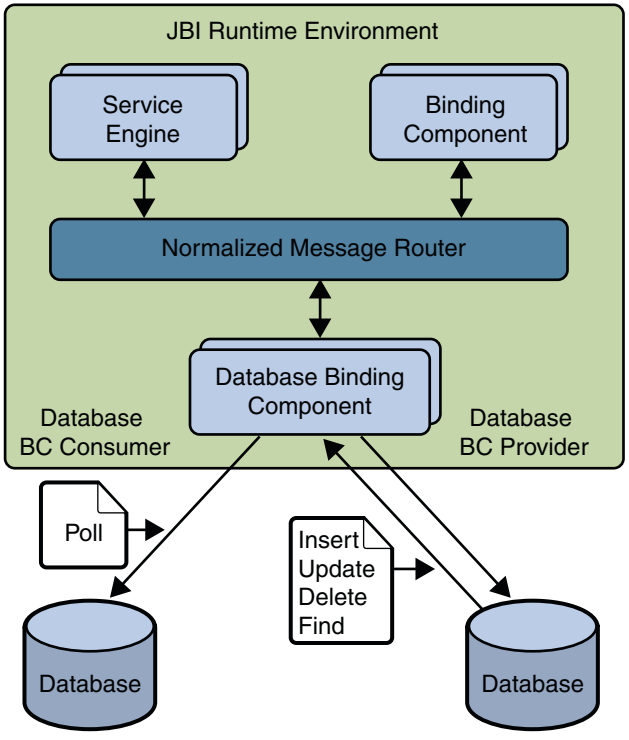
The design-time components still leverage the **org-netbeans-modules-wsdlextensions-jdbc.nbm**. NetBeans Modules help create artifacts for the **sun-database-binding**. The artifacts that the new version of these components generate are deployed to the Database BC instead of the JDBC Binding Component or SQL Service Engine.

Database BC (DB BC) is a Java Business Integration (JBI) runtime component that provides a comprehensive solution for configuring and connecting to databases. Database BC provides data operations as Services. It supports the JDBC from within a JBI environment. Other JBI components invoke these Web Services acting as consumers. Database BC considers both Data Manipulation Language (DML) and Data Definition Language (DDL) operations as Web Services.

The services that the Database BC exposes are actually SQL operations on Table, Prepared Statements, and Procedures. The Database BC supports the following database artifacts to be exposed as Services.

- Table
- Prepared Statements
- Procedures
- SQL File

The Database BC can assume the role of either a JBI consumer (polling inbound requests) or a JBI provider (sending outbound messages).



Once installed, the Database BC can be used to design, deploy, and run the Service Units. The most important part of a Service Unit is the WSDL that describes the Database services. Database BC provides a set of extension elements specific to Database BC for connecting to the Database.

Database Binding Component as Provider

Database BC acts as a provider in case of outbound message flow. Database BC acts as an external service provider when other engines and components 'invoke' it. In this role, when it receives a normalized message as part of the message exchange, it converts and extracts the SQL operation. The SQL operation is then executed on the specified database. In other words, when the Database BC acts as a JBI provider, it extracts the SQL query from a JBI message received from the JBI framework. It then executes the query on a specified database. It converts the reply from the database into a JBI message that other JBI components can service.

Database Binding Component as Consumer

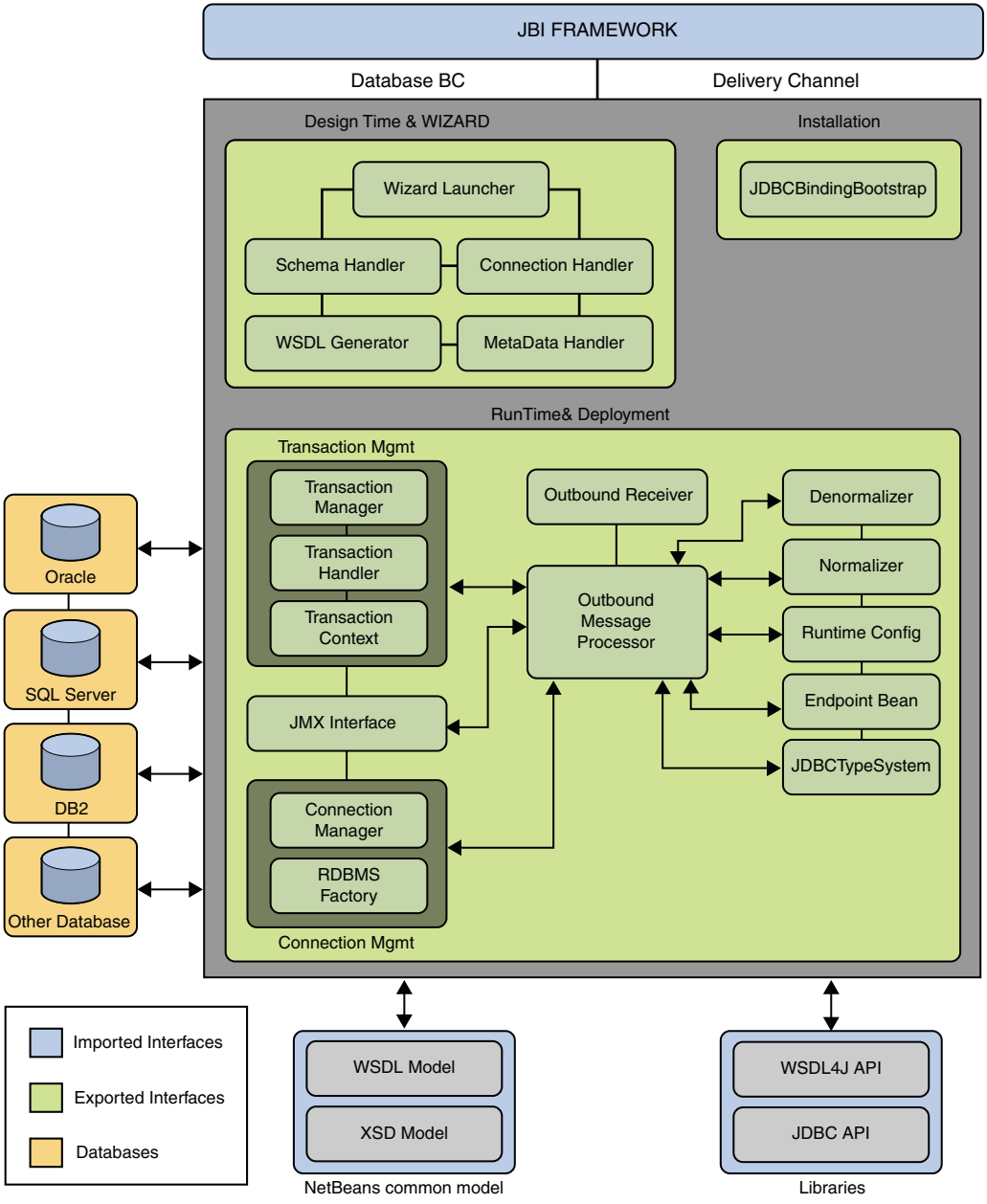
Database BC also acts as a consumer incase of inbound functionality where Database BC polls for records from a particular table, converts them into normalized message, and sends to the Normalized Message Router (NMR). This process is analogous to the inbound connections implemented in CAPS 6. In other words, When the Database BC acts as a JBI consumer, it polls a specified database for updates to a table in the database. When a new record is stored in the table, the database polls for the record for the specified time interval and the Database BC picks up that record, constructs a JBI message, and sends the message to the JBI framework so it can be serviced by other JBI components.

Database Binding Component WSDL Extensibility Elements

The Database BC WSDL extensibility element is a template used to construct an instance of a Database BC WSDL. The Database BC WSDL extensibility elements contains information for constructing the Database BC message. These messages are constructed using the message parts, message formats, properties mapping, and other message related information necessary for the Database BC to properly map message exchanges to Database BC messages and vice versa. The Database BC WSDL extensibility elements also contain information about the database to which it connects.

Functional Architecture of Database Binding Component

The following figure shows the functional architecture of Database Binding Component.



The following table briefly describes the functional operation of each module or wizard.

TABLE 1 Functional Description

Function	Description
Installation Module	Installs, uninstalls, and manages the lifecycle of the Database BC. This module is used to plug the Database BC into the JBI Framework and monitor the lifecycle of the Binding Component such as install, uninstall, start, stop, and so on.
Wizard Module	Assists in interacting with the database. ■ The Database Wizard create/edit interface queries the database to build WSDL from Database Table, Procedures, and Prepared SQL Statements. ■ XSDs are created based on the Table, Procedures, and Prepared Statements in the external data source. ■ Database Operations are added to provide the appropriate database functionality. ■ The wizard assists in interacting with the database using JDBC API specific calls and ensures that appropriate methods are called and the data is formatted appropriately when manipulating the database. ■ The Database Wizard can create XSDs based on any combination of Table, Procedures or Prepared Statements. ■ The Database Wizard uses imported interfaces like the WSDL Editor and XSD Editor.
Wizard Launcher	Launcher the Database Wizard. This interface basically plugged in into the existing WSDL Editor Wizard.
Schema Handler	Is responsible for creating an XML Schema for the corresponding table. The generated schema can be imported into the WSDL
MetaData Handler	Gets the MetaData from the database and displays the data to the user through the wizard. MetaData consists of the user-specific description of the table. MetaData handler gets the MetaData of the TableColumns, Prepared Statements, and Procedures. This data is supplied to the schema generator module to generate the schemas.
WSDL Generator	Generates the WSDL using imported APIs and the Schema Handler.
Database BC Runtime	Provides the functionality for the Database BC at runtime. The Database BC receives the normalized message it gets from the NMR, denormalizes the message, and gathers the required parameters (JNDI name, Operations, and so on) from the message. It processes the parameter, normalizes the output, and sends it back to the NMR.

TABLE 1 Functional Description *(Continued)*

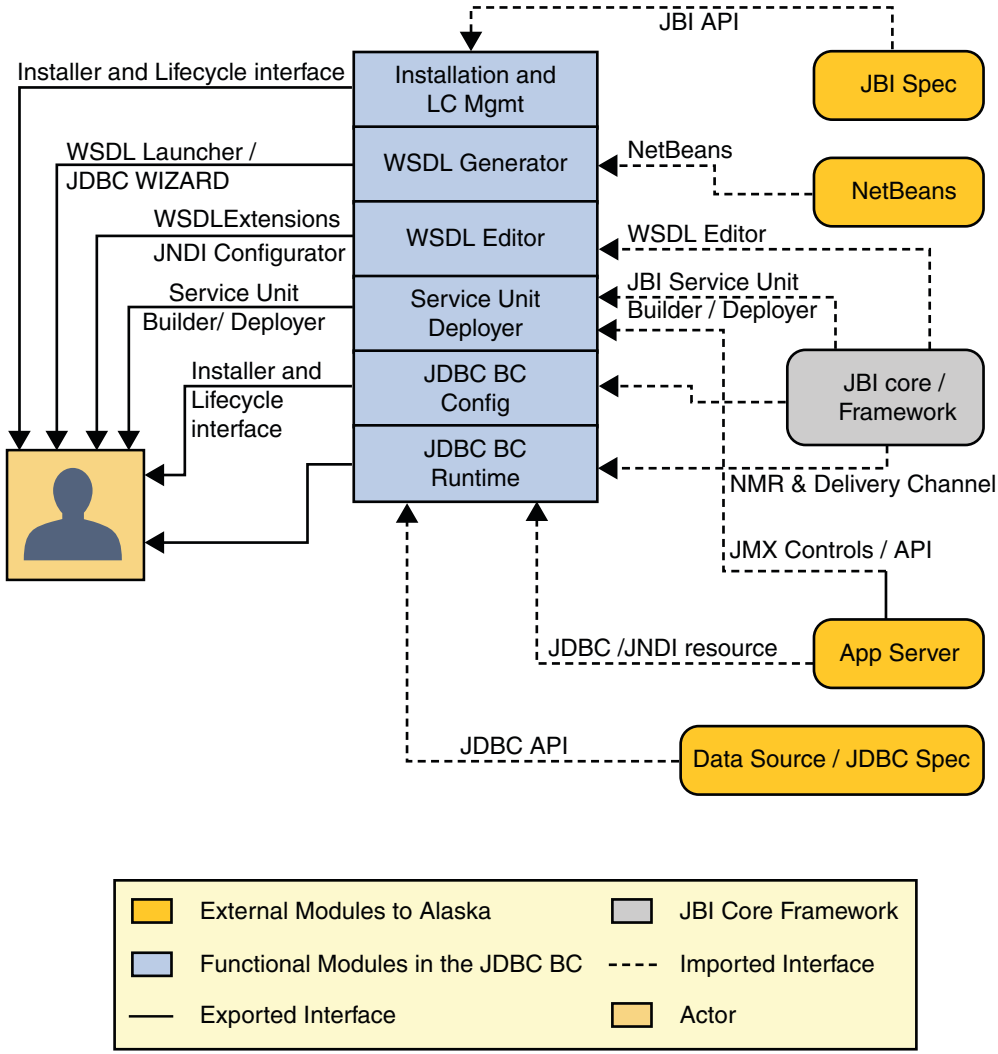
Function	Description
Connection Handler	Provides the functionality to get the connection from different databases. This module uses the JMX API to create a connection pool and the JNDI name to obtain the connection from the data source or Java Naming API. This information is used to create JDBC resource and bind it to the JDBC context of the JNDI tree. The Connection Handler uses two methods to get connected to the database. If the user has already created the JNDI name and wants to establish a connection to the database, then the Connection Handler looks up the JNDI name in the JNDI context and gets the connection. If the given JNDI Name does not exists the Connection Handler binds as a new JDBC context to the JNDI tree during deployment time. In the second method, the Connection Handler establishes a connection using the connection parameters. (driver class name, URL, username, and password).
Transaction Management	The Database BC implements the XAResource interface of JTA to be part of the global transaction and enlists the resource with the Transaction Manager. The Transaction Manager is responsible for starting and ending the XA Transaction. It implements the two-phase commit protocol to support the global transaction.
JMX Interface	Provides a method to bind the Database BC context to the JNDI tree of an application server context.
JBI Framework	Provides administration tools such as install, uninstall, deploy, and undeploy and normalized message router functionality to the Database BC.
NetBeans Common Model	The Database BC uses the WSDL Editor and XSD Editor imported models from the NetBeans as part of the enterprise pack.

Functional Architecture of the JDBC Binding Component — Comparative Study

The following figure depicts the functional architecture of the Database BC including various logical components and external systems. The diagram is centered around the external and internal interfaces provided by the binding component. The term interface is used in a generic sense to mean any piece of information going back and forth between components.

The architecture includes the following:

- Public Private (External to Alaska)
- Project Private (Internal to Alaska but External to JDBC BC)
- Imported Interfaces
- Exported Interfaces
- Core JDBC BC Functional Modules


JBI API
JBI Spec
Installer and Lifecycle interface
Installation and LC Mgmt
WSDL Launcher / JDBC WIZARD
WSDL Generator
NetBeans
NetBeans
WSDLExtensions
JNDI Configurator
WSDL Editor
WSDL Editor
Service Unit Builder/ Deployer
Service Unit Deployer
JBI Service Unit Builder / Deployer
Installer and Lifecycle interface
JDBC BC Config
JBI core / Framework
JDBC BC Runtime
NMR & Delivery Channel
JMX Controls / API
JDBC /JNDI resource
App Server
JDBC API
Data Source / JDBC Spec
External Modules to Alaska
JBI Core Framework
Functional Modules in the JDBC BC
Imported Interface
Exported Interface
Actor