Glossary of Terms

This list is grouped by topic, not alphabetically.

BUSINESS OBJECT. An object that is modeled after a business concept, such as a person, place, event, or process. Business objects represent real world things such as employees, products, invoices, or payments. To remain competitive, modern day enterprises need information systems that serve and adapt to their complex needs. Applications designed from the ground up (not hacked) using the business object model are better suited to meet the requirements of rapidly evolving businesses.

LOCAL OBJECTS. Local business objects are objects that reside on the same virtual machine as the client application or applet. Local objects are used to implement a single user or two-tier scenario.

DISTRIBUTED OBJECTS. Distributed objects reside on a different virtual machine than the client application or applet. BSF gives you the ability to easily create true three-tier distributed objects.

SCALABLE BUSINESS OBJECTS. Objects that can be deployed in a variety of configurations from one-tier to n-tier.

BSF allows the creation of local objects, which can run in a one-tier or two-tier configuration. In one-tier deployment an application's user interface, business objects, and database tables all reside in the same machine. An example would be a single user PC using a Microsoft Access database on its local hard drive. In a two-tier deployment the user interface and business objects reside on the same machine, but the database itself resides on a SQL server. This is the traditional approach for creating client/server applications.

BSF also allows the creation of distributed objects in which the application can be deployed in a three-tier or n-tier configurations.

THREE-TIER ARCHITECTURE. A software architecture that divides the presentation (user interface), application logic, and data storage into three distinct layers. Each layer or tier usually resides on a different virtual machine. The presentation layer only communicates with the application or middle layer, which contains the business objects. The middle layer handles the applications processing logic and in turn communicates with the data access layer, such as SQL server. A three-tier application allows the implementation of thin clients and is much more flexible and easy to maintain than a two or one-tier. For example, the data storage layer can be substituted completely without having to change a single line of code at the client side (presentation layer).

THIN CLIENT. A client application or applet that is very small. The ideal thin client contains only user interface code and communicates with a middle layer residing on another machine that implements the business logic. One major advantage of thin clients is that they can run on the new network computers such us the NC, or the NetPC, or other devices such as mobile phones, PDA's, or TV boxes that support a Java virtual machine.

SUBSET/SUPERSET BUSINESS OBJECTS. Usually a business object has 1:1 relation with a database table. VBSF supports the ability to define business objects that do not have a 1:1 relation with a database table. A subset or "projection" object represents only a portion of the fields in a table which is useful for tables with many columns. A superset or "view" object has attributes which represent fields in two or more tables which is useful for decision support applications. Another important feature is the ability to save multiple classes to the same database table. This allows the mapping of all classes in an inheritance tree to a single table, using a filter column to distinguish between subclasses.

RELATIONAL DATABASES. A database organized as a collection of tables where each table is made up of columns and rows. Tables are related to each other using the same value on a link field referred to as the key or foreign key field. Relational databases represent a mature technology that has been around for many years and is in widespread use.

OBJECT DATABASE. A database that stores objects directly. An object in the database references another object without the need for linking fields. Object databases are very powerful when dealing with complex object relationships and with complex data types such us audio or video. Many relational database vendors are in the process marrying relational and object technologies by providing better support for such types.

JAVA OBJECT/ RELATIONAL DATABASE IMPEDANCE MISMATCH. Java objects cannot be directly saved to or retrieved from relational databases. Java developers working on applications that access relational databases are forced to write massive conversion routines using two different languages, SQL and Java. They also have to worry about the database driver's API (e.g. JDBC). VBSF solves the complex issues involved in the translation. Java business objects can be easily

saved and retrieved by simply calling dbUpdate() and get() methods. It supports loading and saving collections of objects as well as complex queries. See the white paper Mapping Objects To Relational Databases by Scott W. Ambler for a detailed explanation of these issues.

JDBC. JDBC (Java Database Connectivity) is a programming level interface based on the X/Open SQL Call Level Interface. It is very similar to Microsoft's Open Database Connectivity (ODBC), and in fact can be implemented on top of ODBC using the JDBC-ODBC bridge. JDBC is part of the core specification under the Java 1.1 API. Many database vendors are introducing native JDBC drivers. For more information on JDBC see http://www.javasoft.com/jdbc.

RDO. Microsoft's RDO (Remote Data Objects) are a set of objects that make up an information model for accessing remote data sources through ODBC. RDO objects are COM objects that run only on Microsoft's implementation of the Java virtual machine. RDO support is included with Visual J++.

ADO. ADO (ActiveX Data Objects) is Microsoft's newest specification for database access. It introduces the concept of a data provider. Microsoft includes an ODBC Provider so that, similar to RDO, it becomes a thin layer on top of ODBC. ADO objects are COM objects that run only on Microsoft's implementation of the Java Virtual Machine. Eventhough ADO support is not bundled with Visual J++, it is included in the Microsoft Java SDK 2.0 or above, which can downloaded from Microsoft at http://www.microsoft.com/java. For more Information on ADO see http://www.microsoft.com/ado.

JDO. (Java Data Objects) A specification developed by Sun Microsystems under the Java Community Process that provides transparent persistence of Java objects to a data store. The specification allows the use of a standard API to perform persistence operations.

RMI. Remote Method Invocation is Java's specification for distributed systems. It allows Java objects to invoke methods on other Java objects residing in different virtual machine address spaces. RMI is part of the core Java specification under the Java 1.1API. BSF builds on top of RMI when implementing distributed objects.

OBJECT PERSISTENCE. After program execution the state of a Java object (the value of its attributes) is lost unless saved to permanent storage. When an object maintains its state between program executions it is said to be persistent. Some common ways to implement object persistence under Java include text or binary files, object databases, and relational databases. Persistence to text or binary files is generally not acceptable for most business applications. Object databases provide direct persistence for Java objects, but are usually more expensive, geared toward complex data types, and not as widely used or supported as relational databases. In our opinion, currently the ideal avenue to implement business object persistence for robust commercial applications is via relational databases.

LEGACY DATABASE INTEGRATION. One of the primary needs for an object relational framework is the existence of legacy data. New object oriented applications that access data located in legacy databases are constantly being developed. VBSF is designed so it can seamlessly integrate with virtually any type of existing relational database that can be accessed via JDBC, RDO, or ADO drivers. Explicit support for major products such as MS SQL Server, Oracle, Sybase, Informix, DB2, CA-OpenIngres, InterBase, Solid, and Access, as well as flexible mapping options to support most other products, is built in. In the event that a particular database that cannot be integrated with BSF, let us know, and there is an excellent chance we will add the necessary support.
