

4D in a Java World



4D TECHNOLOGY WHITE PAPER

Java is considered by some to be just a programming language, while others consider it to be a bona fide computing platform – albeit a virtual one. The fact that it is virtual, and not tied to any particular computer hardware or operating system, is a large part of Java's enormous appeal. The promise of "write once, run anywhere" is very alluring to I.T. departments that have sunk millions of dollars into proprietary solutions that were difficult to integrate with other proprietary and even standards-based solutions.

At first glance, 4th Dimension may appear to be a proprietary system, difficult to integrate with the larger world of Information Technology and Java. 4D certainly has several proprietary aspects; however, in exchange for a non-Java, proprietary technology, the 4th Dimension product line achieves a high level of integration, performance, and stability.

4th Dimension, just as with the overwhelming majority of databases, application development tools, and general-purpose software, is not written in Java. Indeed, it would be difficult to imagine a high-performance product like 4D being written in Java. Rather, 4D is written in C/C++, and as such it provides extremely high performance on the Windows and Macintosh operating systems. The 4D programming language is an advanced, high-level 4GL, and as a result is much easier to learn and exploit quickly than Java. 4D can even be extended using plug-ins written in any language (such as C++) to conform to the published 4D API.

There are many reasons an organization might choose to deploy 4th Dimension-based solutions – even in an environment that makes extensive use of Java. 4D has always been known as a tool for very fast application development times. It is also a tool that makes an application very easy to deploy, since the server software serves the data as well as the application itself, via several Web-based mechanisms, or an elegant client/server GUI interface. Applications created with 4D can have a rich Graphical User Interface that look

and behave like a "native" application on Windows or Macintosh. Additionally, 4D applications have a very "snappy" feel to them, as you would expect from something compiled for the target platform's CPU.

THE 4D SUITE OF PRODUCTS

To begin to appreciate all that 4D has to offer, it is first necessary to understand that 4th Dimension is actually an entire suite of products. While there is much more to it, the core of this suite is comprised of:

4th Dimension: A single-user application, used primarily as a development tool for database applications.

4D Server: A high-performance database and application server capable of handling hundreds of concurrent processes.

4D Client: The application used to access 4th Dimension databases and applications on a 4D Server.

4D, 4D Server, and 4D Client all share a common 4GL (4th Generation Language). Along with the 4GL is a world-class debugger that makes the developer's job extremely easy in comparison to many development environments on the market today. The command set and syntax is virtually identical on the Windows and Macintosh platforms.

4D and 4D Server also have a built-in HTTP Server and middleware engine.

Other development tools include:

4D Compiler: A tool for compiling a 4D application's developer-written methods into native machine code for extremely high performance.

4D Insider: A code maintenance and management tool.

4th Dimension includes many plug-ins that extend 4D's enormous repertoire of capabilities. A few such plug-ins are:

4D ODBC: A plug-in that allows for access to any ODBC-compliant database.

4D for Oracle: A plug-in that allows for native access to any Oracle database, 7.1 or higher.

4D Internet Commands: A plug-in that provides high-level access to FTP, POP3 and SMTP, but also supports access to any TCP/IP protocol through low-level commands.

LINKING JAVA WITH 4D

Now that you have a general overview of the 4D product line, you are better equipped to understand how 4th Dimension fits into an organization that makes heavy use of Java.

There are many ways to link 4D with the Java world. The primary thing that 4D and Java will need to share is data. It is crucial that 4D be able to access the same databases that Java can, and vice versa. With its connectivity plug-ins (ODBC, Oracle), 4th Dimension is able to interact with virtually any enterprise database (Oracle, DB2, Sybase, SQL Server) quickly and easily. Java, of course, will use JDBC to connect to these same databases. Since 4D and Java can work with the exact same databases in real time, this establishes an excellent medium for communication with one another.

One of 4th Dimension's advantages is that it has an integrated relational database engine. That makes it extremely easy for 4D to use its own data. But how can the Java world access this data? The answer can be found in 4D Open for Java. This is a 100% Pure Java (certified by Sun Microsystems) set of classes that gives Java applications the ability to work directly with data residing on a 4D Server. It is important to note that 4D Open for Java is not JDBC. Rather, 4D Open for Java uses an API that is similar to the rest of the 4D Open suite (4D Open for 4D, 4D Open for C++ on Windows and Macintosh).

4D Open for Java gives you more than just access to 4D Server's data. As with the rest of the 4D Open suite, it is possible to execute a Stored Procedure (or "server process"). It is also possible to Get and Set the values of variables and arrays for any process that is running on 4D Server. It would be easy to create a simple messaging system that allows Java to send and receive messages with a 4D Server in this fashion – without the overhead of using database records as a means of communicating.

HTTP DATA EXCHANGE

There are still more ways to get 4D and the Java world talking. 4th Dimension and 4D Server feature a built-in HTTP server. This means that any application that "speaks" HTTP can effectively communicate with 4D. HTTP is the protocol (language) used by Web browsers to communicate with Web servers. The technical term for "browser" in the HTTP specification is "User-Agent." This terminology is appropriate, since nowhere is it carved in stone that the application making the request of an HTTP server must be a browser.

Indeed, the use of HTTP as a data transport protocol has been growing. For example, WebDAV (Web Distributed Authoring & Versioning) is a superset of HTTP, as is SOAP (Simple Object Access Protocol). Many CGI and middleware products already support sending and receiving data over HTTP. PERL can do this using the LWP module (libwww-perl). PHP can do this using an "include." This can also be done with Microsoft's Active Server Pages and Microsoft's XML Parser (MSXML3). Macromedia's ColdFusion supports this too, using the CFHTTP tag.

As you would expect, Java can send and receive data over HTTP as well. The standard Java Development Kit provides a Java class named URLConnection. Using this class, it is elementary to open a TCP/IP socket connection to 4D's HTTP server. Once the connection has been established, a Java program can issue any request to 4D. 4D's response can be formatted for easy parsing by the Java application that made the request.

Text parsing is something that most Java developers can do very easily – but they won't be happy to do it, since it involves a fair amount of drudgery. Most Java developers would prefer to use a Java class library that handles this for them. Once again, Sun provides a solution in the form of JAXP, the Java APIs for XML Processing.

Java developers can open an HTTP connection using the URLConnection Java class, and 4D can give a response formatted as XML. How does 4D support responding with XML? XML is simply a document format. Like HTML, it is tag-based and is composed purely of text. 4D's built-in middleware can make ready use of XML "templates" by dynamically replacing special 4D tags embedded in the XML template with values derived from the database engine as the XML response is sent. The only coding required on the 4D side is a query parser to convert the request sent by the Java program into a query against 4D's database.

Writing a Java program to query 4D for data over HTTP and handle the XML-formatted response is a relatively easy programming task. But how can 4D receive XML data sent over HTTP from a Java program? This is somewhat more complicated, but still entirely feasible.

To begin, you will need the 4D Internet Commands – the license for which is included with every copy of 4D. As mentioned earlier, 4D Internet Commands is a plug-in for 4th Dimension that provides low-level access to TCP/IP. A very small amount of 4D coding is required to set up a TCP Listen on a specific port, and to wait for a connection. Once a connection has been established, 4D can receive any data that is sent to it, storing it in a BLOB (Binary Large Object). After the connection has been closed, the BLOB will hold the contents of the XML data.

How will 4D process the XML data? Using 4D's powerful built-in programming language, it is possible to parse through the XML. However, 4D developers don't like text parsing any more than Java developers do. At the present time, 4D does not provide any built-in tools for parsing XML. Fortunately, a third party has stepped in to provide a solution: Expat4D.

Expat4D is a plug-in for 4th Dimension (with versions for both Windows and Macintosh). This product is essentially a "wrapper" for the Expat XML parser. Expat is a stream-oriented parser that is distributed under the GNU General Public License. It is a very fast, powerful parser – so much so that it is the technology used for XML parsing in Netscape's Mozilla project, as well as PERL's XML:Parser.

Using Expat4D is as easy as using Expat itself. If you have received XML data from a Java program over TCP/IP, the XML data is already contained in a BLOB. This BLOB can be fed directly to Expat4D using the `xml_ParseBLOB` function. Like the Expat library, Expat4D is a freeware product. It is available for download on [mitchenall.com](http://www.mitchenall.com):
<http://www.mitchenall.com/products/freeware/4d/expat4d/index.phtml>

REMOTE METHOD INVOCATION

It should be clear to you now that it is possible - and not very difficult - to exchange data between the Java and 4D worlds. You have seen that by using 4D Open for Java, it is possible to execute a 4D Stored Procedure on a 4D Server. Is it possible to execute 4D methods on 4D Client or 4D Stand-Alone? And how can you use 4D to call Java class methods?

Again, the answer to both those questions is "yes." Once again, this functionality is provided by a third party: Ronri Kobo, Inc. (www.ronri-kobo.com) This company provides two plug-ins for 4th Dimension: JbyJ and JExternal.

JbyJ consists of two components: a 4D plug-in, and a set of Java class libraries. On the 4D side, the developer sets up a "JbyJ Server" to handle Java RMI (Remote Method Invocation) calls. Using RMI, a Java program can execute a method on 4D or 4D Client, and data can be returned from the 4D method to the Java program that called it.

JExternal allows 4D to call Java class or object methods. To do so, all a developer needs to do is place the appropriate class files in the 4D plug-ins folder ("Win4DX" or "Mac4DX") along with the JExternal plug-in itself. A one-line command is all it takes to execute the Java code and receive back a result. This opens up enormous possibilities for 4th Dimension developers. For instance, 4D developers can now use any JDBC driver to query the variety of databases for which there is no ODBC support.

4th Dimension and Java are two very different technologies, both architecturally and philosophically. While these differences are many and great, the chasm between these two worlds is one that is easily bridged. 4D can work and behave well in a Java-laden environment, and at the same time 4D can leverage and build on the strengths that the Java platform provides. Working together, each technology can do what it does best, empowering developers to deliver world-class solutions to tackle any business problem.