

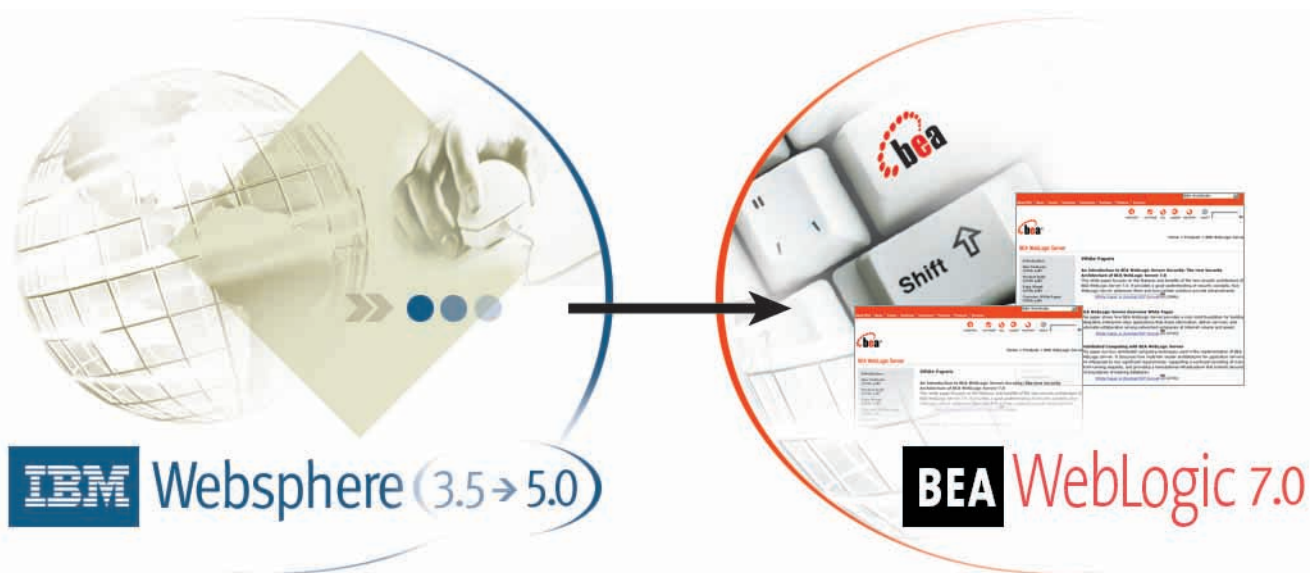
Whitepaper

IBM WebSphere 3.5 to BEA WebLogic 7.0 Migration

The Market

Many organizations have utilized IBM WebSphere 3.5 as their Java 2 Platform, Enterprise Edition (J2EE) application server for a number of projects. As IBM WebSphere 3.5 is now several years old and because it is based upon older, less comprehensive versions of J2EE standards, these organizations are considering whether they should migrate their existing applications to the soon-to-be released IBM WebSphere 5.0 or migrate to the standards- and market-leader, BEA WebLogic Platform version 7.0.

This white paper is provided for technical project managers and software architects who are considering a migration from an existing WebSphere 3.5 application to BEA WebLogic 7.0.



IT Solutions. Fixed Price. Fixed Time.™

www.eforceglobal.com

Benefits of the BEA WebLogic Platform

Many organizations are looking at the BEA WebLogic Platform with interest. It is the broad capabilities on a single, standards-based architecture that appeals to them.

- **WebLogic Server** offers leading support for J2EE standards as well as the scalability, performance, and fault tolerance required by today's mission-critical enterprise solutions.
- **WebLogic Portal** provides a complete, unified, advanced portal framework for cost-effectively web enabling all interactions with employees, customers, vendors, partners, and other stakeholders. With the robust profiling, content integration, personalization, branding, e-commerce, and other related capabilities, WebLogic Portal users can configure and integrate an advanced user experience that would otherwise require the development of significant amounts of custom code with IBM WebSphere.
- **WebLogic Integration** provides a J2EE-based architecture for Enterprise Application Integration (EAI), Business Process Management (BPM) and Business-to-Business Integration (B2Bi). While IBM WebSphere-based organizations must custom code integration or leverage a separate product, such as MQ or CrossWorlds, based upon a fundamentally different, non-standard architecture, organizations using the WebLogic suite are able to build a cohesive enterprise-wide software architecture.
- **WebLogic Workshop** offers an integrated development framework lowering the barrier of entry so that even junior team members can rapidly create, test, and deploy enterprise-class Web Service applications. With the visual development environment of WebLogic Workshop, developers are insulated from many of the complexities of J2EE.
- **WebLogic JRockit** provides a robust foundation for WebLogic Platform 7.0. It is the industry's first Java Virtual Machine designed for mission-critical server-side usage. Advanced garbage collection, choice of threading models and built-in management support, consistent performance and stability for server-side Java applications that is unprecedented. As JRockit is built for the generally lower cost Intel hardware platform including 64-bit Itanium 2 servers, it offers optimal quality at an unprecedented hardware price point.

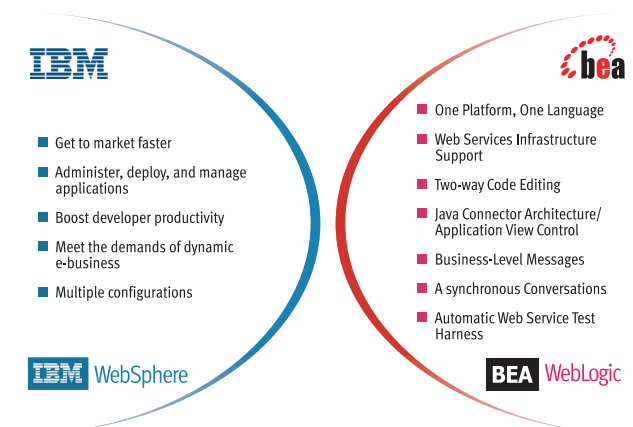
Reasons to Migrate to BEA WebLogic Platform

While BEA WebLogic and IBM WebSphere each offer a foundational J2EE application server for 100% custom-developed applications, it is WebLogic 7.0's robust, standards-based capabilities that are causing organizations with a significant, legacy investment in IBM WebSphere 3.5 to consider what possibilities actually are available to them.

Some organizations are questioning if an enterprise-class J2EE strategy should bear the costs of a 100% custom strategy. Or, is it possible to leverage the advanced capabilities offered by WebLogic Portal, WebLogic Integration, and WebLogic Workshop, eliminating the need to hand-craft the complete solution.

Other organizations that have made both BEA WebLogic- and IBM WebSphere-based investments are interested in the cost saving and quality of service benefits associated with technology consolidation. These organizations simply want to make informed decisions in determining whether to continue to fund a heterogeneous environment, or to consolidate. In addition, they would like to make informed decisions about which way to consolidate and how much it will cost to migrate applications.

Regardless of an organization's situation, all types of organizations using IBM WebSphere may be surprised to learn that a WebSphere 5.0 upgrade is not their only choice. In fact, an IBM WebSphere 3.5 to BEA WebLogic 7.0 upgrade is easier than many organizations realize and, in many cases, it may actually be easier to upgrade to BEA than the next version of IBM WebSphere.



Much of the Migration Concerns Keeping Pace with Maturing J2EE Standards

To understand the relative level of effort of an IBM WebSphere 3.5 to BEA WebLogic 7.0 migration, it is necessary to review the capabilities of WebSphere and look at how each J2EE component can be migrated to BEA WebLogic 7.0.

IBM WebSphere 3.5 is based upon the following versions of the Java and J2EE standards:

- J2SE SDK version 1.2.2
- Servlet 2.2 (as well as a legacy "Compatibility Mode")
- Java Server Pages 1.1
- Enterprise JavaBean (EJB) 1.0 (with some EJB 1.1 support)

- JDBC 2.0 (as well as legacy connection management APIs)
- Java Native Directory Interface (JNDI) 1.2
- RMI/IIOP 1.0
- Java Transaction Service (JTS)/Java Transaction API (JTA) 1.0
- Java Messaging Service (JMS) 1.0

BEA WebLogic Server version 7.0 is based upon the following versions of the Java and J2EE standards:

- J2SE SDK version 1.3.1
- J2EE 1.3
- Servlet 2.3
- Java Server Pages 1.2
- JDBC 2.0
- EJB 2.0
- J2EE Connector Architecture (JCA) 1.0
- JTA 1.0.1
- JMS 1.0.2
- JNDI 1.2
- Java RMI 1.0
- RMI/IIOP 1.0
- Java Authentication and Authorization Service (JAAS) 1.0
- JavaMail 1.1
- Java API for XML Processing (JAXP) 1.1

To be effective, a migration strategy must address each J2EE component. In fact, many of the recommendations provided in this white paper concern keeping pace with maturing J2EE standards. The benefits of keeping up with the standard include increased stability, flexibility, and portability.

In addition to the J2EE related upgrades, specific WebLogic configuration points have been identified.

IBM WebSphere Configuration Export Recommendations

■ Use XMLConfig to export IBM WebSphere Configuration

One of the essential tools that can be utilized in the migration is XMLConfig. XMLConfig is a command-line Java client utility that ships with WebSphere 3.5. XMLConfig provides the ability to import and export a WebSphere configuration as well as administer the server via the command-line or from a programmatic Java client.

In the migration, it will be utilized specifically for exporting WebSphere configuration of JDBC drivers, JDBC data sources, application servers, EJB containers, Servlet engines, web applications, Servlets, JSPs, and security.

JDBC Migration Recommendations

Both IBM WebSphere 3.5 and BEA WebLogic 7.0 support JDBC 2.0. However, the two servers have slightly different configurations for database connectivity.

■ Use XMLConfig to export WebSphere JDBC driver configuration

XMLConfig can be used to export information on each JDBC driver from WebSphere. Key JDBC driver information can be found in the XMLConfig export file, specifically within the jdbc-driver XML tag. This information includes the name of the driver, the name of the JDBC driver implementation class (such as oracle.jdbc.OracleDriver, the URL connection string, an indicator whether or not JTA transactions are enabled, and the file system path of the JDBC zip file.

■ Use XMLConfig to export WebSphere JDBC data source configuration

In addition to the information on the JDBC driver, XMLConfig can also be used to export information on each configured JDBC data source. Key information about each JDBC data source can be found in the XMLConfig export file under the data-source XML tag. For each JDBC data source, this file identifies key configuration information including the database name, the JDBC driver name, the minimum connection pool size, the maximum connection pool size, and connection timeout settings.

■ Check WebSphere's datasources.xml for additional JDBC configuration

In addition to information exported using XMLConfig, some additional JDBC configuration information may be defined in the datasources.xml configuration file. This file may contain additional properties for defining the connection to the JDBC data source.

The above JDBC configuration information from WebSphere can be used to define the corresponding JDBC configuration within WebLogic Server.

■ Define JDBC connection pools in WebLogic's config.xml

In WebLogic Server, most of this type of JDBC configuration information is placed within the config.xml configuration file. Specifically JDBC connectivity configuration is specified within the JDBCConnectionPool XML tag as well as within the JDBCDataSource tag.

The information that is specified within the JDBCConnectionPool tag includes the JDBC driver name, the URL connect string, login and password information, connection pool information, and connection testing information. Connection pool configuration includes the initial size of the connection pool, the maximum size of the connection pool, and the number of connections by which the connection pool increases or decreases. Many of the above values can be defined by the values obtained by XMLConfig as well as the configuration specified within WebSphere's datasources.xml. The additional values illustrate WebLogic's greater robustness in database connection management.

■ Define JDBC data sources in WebLogic's config.xml

The information that is specified within the JDBCDataSource of WebLogic's config.xml file includes whether or not two-phase commits are enabled, the JNDI name of the data source and mapping the underlying connection pool. WebLogic provides greater flexibility JDBC connection information. Specifically the logic naming of JNDI names and the ability to map data sources (virtual) to connection pools (physical) yields easier configuration and greater robustness.

■ Put the JDBC driver in the CLASSPATH in startWebLogic.sh

In addition to the above JDBC configuration specified in the config.xml file, WebLogic requires the JDBC driver to be placed within the application server's CLASSPATH as a part of application server start-up. Typically, this configuration is located in a start-up script such as a startWebLogic.sh script. The JDBC driver identified within WebSphere's XMLConfig jdbc-driver needs to be placed within the CLASSPATH of such a server start-up script.

In addition to the above JDBC 2.0 capabilities, WebSphere 3.5 supports two other methods of obtaining JDBC connections. The first method to obtain a JDBC connection is to utilize the JDBC 1.0 API. The second method is to utilizing WebSphere's proprietary connection pooling mechanism.

■ Upgrade all JDBC connection management code from JDBC 1.0 to JDBC 2.0

In JDBC 1.0, the DriverManager interface was used to obtain a connection. The database driver, user ID, and password for the data source were specified explicitly in the getConnection() request. A single connection (not a connection within a connection pool) would be provided using syntax such as: Connection con = DriverManager.getConnection(driver, userid, password);

In JDBC 2.0, the proper approach is to use JNDI to obtain a connection from the connection pool defined within the application server. While migration of JDBC 1.0-style connection management to JDBC 2.0 is not required for an upgrade from WebSphere 3.5 to WebLogic 7.0, it is highly recommended given the power and flexibility of JDBC connection management in WebLogic.

■ Migrate WebSphere proprietary connection management code to JDBC 2.0

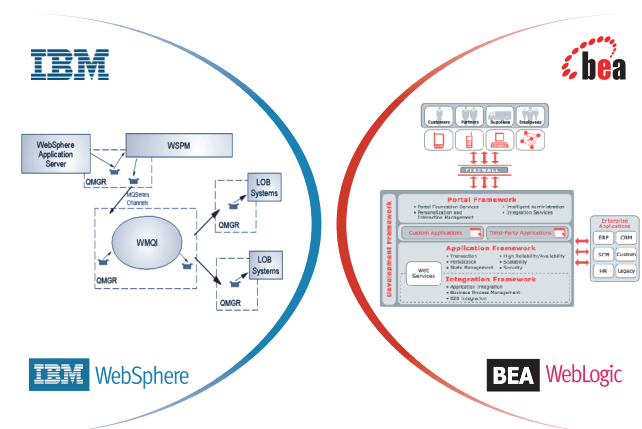
In addition to the JDBC 1.0 API approach, WebSphere provides its own proprietary Connection Manager API. While this API had significant utility prior to the advances of JDBC 2.0, it now represents a proprietary API.

Specifically for this API, WebSphere 3.5 represents a transitional architecture. This API was provided with earlier releases of WebSphere to complement the JDBC 1.0 API. In WebSphere 3.5 although it is still provided, it is deprecated by IBM.

Specifically, one should identify code that imports from the com.ibm.websphere.appserver.cm,

com.ibm.websphere.appserver.cm.factory, com.ibm.ejs.cm.pool, and com.ibm.ejs.cm.portability packages. Even older WebSphere applications may take advantage of com.ibm.servlet.connmgr, com.ibm.db2.jdbc.app.stdeext.javax.sql, and com.ibm.ejs.dbm.jdbcext.

It is recommended that such JDBC 1.0-based and IBM Connection Manager-based JDBC connection management code be migrated to the JDBC 2.0 API. It is additionally worth noting that such a recommendation is not unique to a WebSphere to WebLogic migration but in fact represents a best practice that should be utilized even if an organization were to upgrade from WebSphere 3.5 to WebSphere 5.0.



Server Configuration Recommendations

WebSphere 3.5 supports a notion of application servers. In WebSphere 3.5, an application server represents a container that includes both a Servlet engine as well as an EJB container.

A WebSphere Application Server corresponds to simply what WebLogic refers to as a "server."

■ Use XMLConfig to export WebSphere application server configuration

XMLConfig can be used to export essential configuration information that can be used to establish corresponding configuration information within WebLogic. This information includes user permissions, directory locations, logging output files, thread pool size, and additional information. Specifically this information can be found within the node tag of the XMLConfig export file.

■ Define servers in WebLogic's config.xml

WebLogic servers are configured within the config.xml configuration file, specifically within the server tag. Information configured within WebLogic includes the path to the Java compiler, the server name, the thread count, the IIOp name, logging output files, SSL, security certificates, and more.

EJB Migration and Deployment Recommendations

■ Use XMLConfig to export WebSphere EJB configuration

XMLConfig can be used to export information on each EJB container that has been configured and each EJB that has been deployed.

In addition to XMLConfig, Jetace can also be used to export the binary deployment descriptor information to XML.

EJB container configuration information is within the container node of the XMLConfig export file. The container node contains information on the EJB container's user id, password, configuration of the cache, and the data source.

EJB deployment information is within the ejb node of the XMLConfig export file. This information includes the file system path to the JAR file, the home interface name, user id, password, pool size, JNDI name, and more.

■ Define EJB deployment in J2EE standard and WebLogic deployment descriptors

J2EE version 1.3 and WebLogic 7.0 support a significantly more sophisticated configuration. The information from the XMLConfig export file can be used to provide configuration values for both the J2EE standard ejb-jar.xml configuration file and the weblogic-ejb-jar.xml configuration file. Mapping of data sources to JNDI names goes within the ejb-jar.xml file while EJB cache configuration is placed within weblogic-ejb-jar.xml.

Items that were configured at an EJB container level in WebSphere 3.5, can now be configured at an individual EJB level for improved performance and resource utilization.

■ Migrate key areas of EJB code to the new EJB standard

Several areas of EJB code are handled differently between WebSphere and WebLogic.

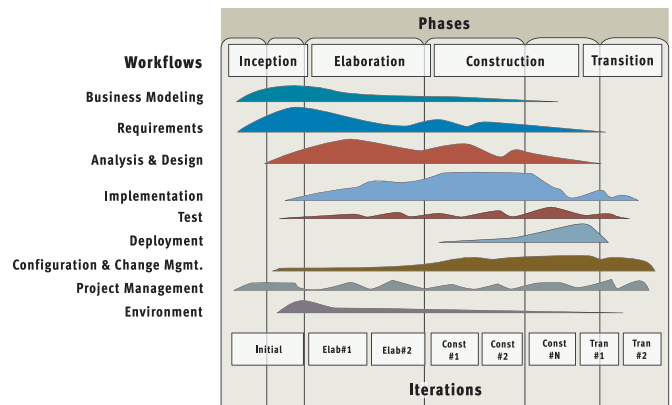
First, WebSphere provides the ability to define certain entity EJB methods as read only. This is done as an optimization to reduce the number of calls that the server must make to an entity EJB's ejbStore() method. In WebLogic, this information can be defined in the deployment descriptor, specifically with is-modified-method-name in weblogic-ejb-jar.xml.

Second, WebSphere provides the capability FinderHelpers to define how CMP entity EJB finder methods operate. In WebLogic, this is specified in the standard deployment descriptor in EJB query language (EJB QL).

Third, if entity EJBs are deployed under the EJB 1.0 specification, there is a difference with the newer EJB specifications that must be considered. In EJB 1.0, ejbCreate() methods returned void. In EJB 1.1 and beyond, the ejbCreate() method returns the type of the primary key and for CMP a null value may simply be returned.

Fourth, if EJBs utilize java.rmi.RemoteException in accord with EJB 1.0, these should typically be replaced with javax.ejb.EJBException.

Fifth, IBM Visual Age offers the ability to create associations between CMP entity EJBs. This should be migrated to container managed relationships (CMR) supported in EJB 2.0.



Web Application Migration and Deployment Recommendations

IBM WebSphere 3.5 supports Servlet 2.2 and JSP 1.1 in addition to a proprietary Compatibility Mode. Depending upon which mode web applications are deployed in IBM WebSphere 3.5, there are different migration paths.

■ Migrate "Compatibility Mode" deployed web applications to J2EE deployment

For web applications that are deployed using the proprietary Compatibility Mode, it is recommended to convert deployment to J2EE standard deployment.

Information on web applications deployed within WebSphere can be obtained from the XMLConfig export file, specifically within the Servlet-engine, web-application, and Servlet tags. For each Servlet engine, this information includes the name of the Servlet engine, maximum number of concurrent connections, logging, and more. For each web application, this information includes the name, description, document root, CLASSPATH, error page, reloading, root URI, shared context information and more. For each Servlet, this information includes the Servlet name, description, class name, initialization, URI path mapping, and more.

With J2EE standard-based deployment in WebLogic 7.0, most of this information is mapped to the J2EE standard deployment descriptor web.xml as well as WebLogic's weblogic.xml. The configuration specified within these files is then placed within the deployable WAR file.

Much of the configuration specified at the Servlet engine level within WebSphere can now be controlled at an individual web application level offering greater flexibility of configuration.

If the web application was deployed in WebSphere using Servlet 2.2 capabilities, the migration will be less effort as most of this standards-based configuration can be reused.

■ Migrate Proprietary Servlet Code to the new J2EE standard

The legacy WebSphere application may have Servlets built using WebSphere Studio. In these cases, the generated Servlets inherit from IBM's proprietary PageListServlet instead of directly from the J2EE standard HttpServlet. It is recommended that these Servlets be changed to be J2EE compliant.

The `com.ibm.servlet` and `com.ibm.webtools.runtime` import statements should be removed and the Servlets be changed to inherit directly from HttpServlet. Additionally, usage of the `setRequestAttribute()` needs to be changed to `request.setAttribute()`. Finally the call to the `HandleError()` method should be removed.

JNDI Migration Recommendations

■ Migrate JNDI initial context code

In WebLogic the JNDI implementation is handled by the `com.ibm.ejs.ns.jndi.CNInitialContextFactory` class. In WebLogic Server, it is handled by the `weblogic.jndi.WLInitialContextFactory` class.

For resolving the initial context, the latter implementation must be used with WebLogic Server.

Conclusions

Many organizations are now evaluating whether they should upgrade from IBM WebSphere 3.5 to IBM WebSphere 5.0, or should they migrate directly to BEA WebLogic 7.0.

A significant part of the WebSphere upgrade involves upgrading in tandem with the maturing J2EE standards. These include:

- JDBC 2.0 offers greatly improved connection management over JDBC 1.0.
- Servlet 2.2 and now Servlet 2.3 offers J2EE standard deployment of web applications.
- EJB 1.1 and now EJB 2.0 offers J2EE standard deployment of EJBs.
- EJB 2.0 offers configuration of container managed relationships.

It is worth noting that upgrading with the J2EE standards is recommended regardless of whether a WebSphere upgrade or a WebLogic migration is

pursued. Keeping up with the standards provides increased robustness, flexibility, and portability, resulting in lower cost of ownership.

As such, many of the recommendations provided in this document apply not only for a WebLogic migration but also for a WebSphere upgrade. As a result, the level of effort associated with an upgrade vs. a migration is actually not that different. Depending upon which specific standard-based and proprietary APIs have been leveraged in the WebSphere 3.5-based system, a WebLogic upgrade may be approximately the same level of effort as a WebSphere upgrade.

In conclusion, the increased power and flexibility of the complete BEA WebLogic Platform, including Portal, Integration, Workshop, and JRockit provides a compelling reason why organizations with an investment in WebSphere, or an investment in WebSphere and WebLogic, should seriously consider standardizing on BEA WebLogic as their enterprise J2EE platform.

About eFORCE

eFORCE specializes in the Fixed Time, Fixed Price implementation of solutions encompassing all areas of the Enterprise Value Chain— Enterprise Portals, CRM, EAI, Business Intelligence, Enterprise Content Management and Enterprise Infrastructure. Combining expertise in business architecture, technical architecture, design, deployment and maintenance, eFORCE delivers production-scale enterprise solutions that result in measurable ROI. eFORCE customers include Global 1000 organizations such as Alcatel, AT&T, Avaya, Baker Hughes, Bank of America, British Telecom, Cisco, DHL, FedEx, Fleet Bank, French Telecom, GE Capital, GE Power, Hilton, HP, Intel, Janssen, Janus, Merck, Mitsubishi, Morgan Stanley, Novartis, Reuters, Shell, Viacom, Visa and Wells Fargo. eFORCE delivers solutions based on best-in-class enabling technologies such as ATG, BEA Systems, BroadVision, Documentum, E.piphany, HP, IBM, Interwoven, MatrixOne, Microsoft, Netegrity, Oracle, Plumtree, SeeBeyond, Siebel Systems, Stellent, Sun Microsystems, TIBCO and webMethods. eFORCE (www.eforceglobal.com) is headquartered in Silicon Valley, has Development Centers in North America, Europe and India, and through its Global Delivery Methodology provides both onshore and offshore full life-cycle implementation – design, development, deployment, maintenance, support.

Contact eFORCE

800.295.1914
510.265.5800
sales@eorceglobal.com

eFORCE Clients

Alcatel	CapitalOne	Fidelity Investments	The Hartford	Kingfisher B&Q	Nortel Networks
American Express	Charles Schwab	France Telecom	Hoover's	Lucent Technologies	Pearson Group
Avaya	Compaq	GE Capital	Intel	MasterCard	Reuters
Bank of America	Credit Suisse	GTE Genuity	Janus	Merrill Lynch	Visa USA
British Telecom	FedEx	H&R Block	Johnson Controls	Mitsubishi	Wells Fargo