



**IBM's Smart SOA[™] Approach
Delivers the Best End-to-End
Transaction Integrity
in the Industry**

IBM
July 1, 2008

Table of Contents

1. EXECUTIVE SUMMARY	3
2. WHERE IS TRANSACTION INTEGRITY REQUIRED?.....	4
3. WHAT IS TRANSACTION INTEGRITY?.....	5
3.1 TRANSACTION INTEGRITY IN AN APPLICATION SERVER.....	5
3.2 TRANSACTION INTEGRITY IN BUSINESS PROCESSES.....	5
3.3 TRANSACTION INTEGRITY IN MESSAGING.....	6
4. IBM DELIVERS COMPREHENSIVE TRANSACTION INTEGRITY	7
4.1 IBM PROVIDES BETTER TRANSACTION INTEGRITY FOR APPLICATION SERVERS	7
4.2 IBM PROVIDES BETTER TRANSACTION INTEGRITY FOR BUSINESS PROCESS MANAGEMENT	11
4.3 IBM PROVIDES BETTER TRANSACTION INTEGRITY FOR MESSAGING.....	17
5. IBM DELIVERS THE BEST END-TO-END TRANSACTION INTEGRITY	19
6. APPENDIX A: PRODUCTS USED FOR THE STUDY.....	20
7. APPENDIX B: ADDITIONAL RESOURCES.....	21

1. Executive Summary

Each day businesses execute billions of business transactions - think of the world's financial exchanges, credit card purchases, or on-line commerce. A failure in these systems could cost a company millions of dollars in lost revenue and lost customer satisfaction. As a result of this risk, it is critical that applications handle these transactions correctly and ensure that enterprise data accessed during business transactions is maintained in a reliable and consistent state, regardless of any system or business failure that may occur. IBM software protects transaction data regardless of any failures that may occur, so we say that IBM software provides **transaction integrity**. Transaction integrity guarantees users that they can rely on critical data that is accessed during transactions by SOA applications.

When evaluating the end-to-end transaction integrity of a SOA platform, all major platform components, including the application server, business processes, and messaging layer must be considered. If a platform fails to provide transaction integrity in any of these components, the integrity of the entire platform can be questioned. IBM leads industry rivals Microsoft, Oracle, and BEA in transaction integrity by providing more robust and more comprehensive integrity across its' platform. The recent Competitive Project Office study of these products concluded that the IBM solution has the following strengths:

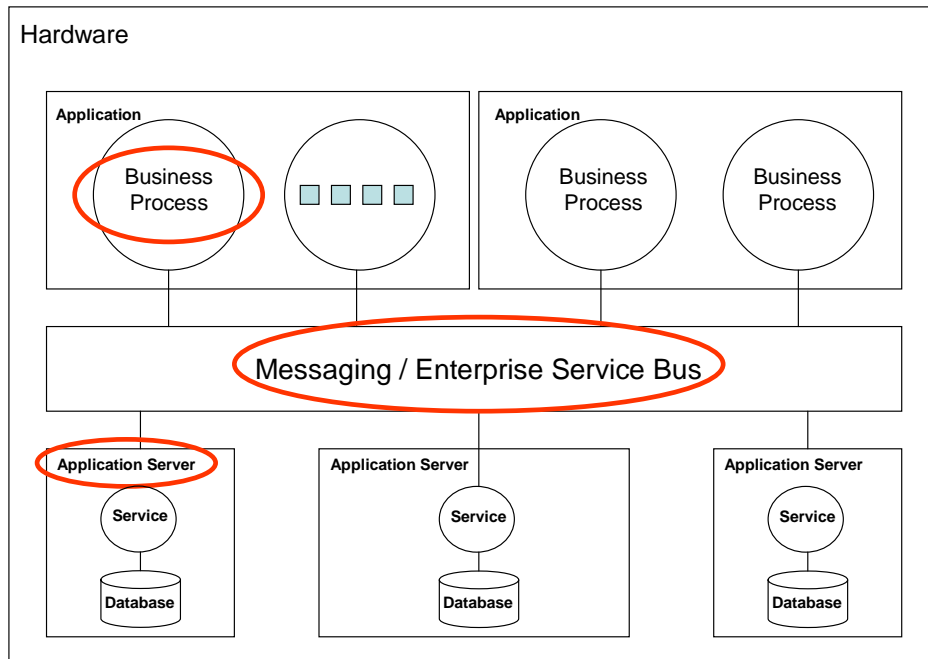
- **IBM provides better transaction integrity for application servers.** Although all application servers can coordinate a transaction that updates multiple database resources, IBM achieves this with less effort than other vendors and demonstrates no integrity problems even when subjected to heavy loads and system outages.
- **IBM provides better transaction integrity for business processes.** IBM's business processes are superior to all others at keeping data consistent within a business process and ensuring that process instances are not lost in the event of failures.
- **IBM provides better transaction integrity for messaging.** IBM ensures that no messages will be lost as they are moved from point to point within a messaging infrastructure. IBM provides this integrity with its WebSphere MQ, which provides greater flexibility and coverage than messaging offerings from Microsoft, Oracle, or BEA.

No matter how quickly or elegantly a developer creates an application, if the application cannot ensure transaction integrity it jeopardizes every application that it touches. The remainder of this paper details how IBM provides comprehensive, end-to-end transaction integrity for application servers, business processes, and enterprise messaging.

2. Where is Transaction Integrity Required?

Transaction integrity requires that every component of an application can recover from any IT or business failure. These failures could take place in the business process, the application server, or the messaging layer (enterprise service bus).

Where is Transaction Integrity Required?



An SOA environment that cannot ensure transaction integrity in each component gives an organization a false sense of security. An application may be able to withstand certain types of failures, but ultimately it will be dangerously exposed to a failure to its Achilles heels. With such a failure, a compromised application can cause a loss of data consistency. While data consistency may sound like a purely technical problem, this consequence of incomplete transaction integrity can have a profound affect on any business.

3. What is Transaction Integrity?

When critical applications fail, the business can lose not only revenue, but it can have a severe negative impact on customer satisfaction and loyalty. In addition, if the applications do not execute reliably, the business exposes itself to increased financial and regulatory risk. Therefore, enterprises must ensure that each layer of their applications can ensure transaction integrity. However delivering transaction integrity requires different qualities for application servers, business processes, and messaging. For each of these components, the Competitive Project Office (CPO) team determined how to define transaction integrity and then how to measure their success at achieving it.

3.1 Transaction Integrity in an Application Server

The application server provides data consistency across the applications it touches by establishing a two-phase commit transaction that guarantees that all resources will be updated in unison. In a two-phase commit transaction, all databases resources in the transaction either are committed or rolled back, ensuring that the databases remain consistent. By providing this transaction integrity, the application server enables an application to recover from failures, such as resource or server outages, while keeping data accurate and consistent.

In order to assess the transaction integrity of an application server, the CPO performed a series of tests to determine how the application server protected data integrity under both normal and adverse conditions. In the study, the application server had to manage thousands of transactions, each of which updated multiple databases. The application server's transaction integrity was tested by generating failures to the application, database resources, the network, and the application server itself. After each test, the databases were inspected to determine whether data integrity was maintained; i.e., the data in each database was correct and remained accurate and consistent with each other. These failures were performed first with a minimal application server load and then again with a high server load in order to determine if the application server could continue to maintain transaction integrity in a much more demanding, more production-like, environment. As the applications were developed and executed for these tests, the CPO also identified development or deployment difficulties that might delay completion of the application or affect some part of the IT environment.

3.2 Transaction Integrity in Business Processes

For enterprise business processes, transaction integrity has different implications depending on whether the process is synchronous or asynchronous. Synchronous processes complete nearly instantaneously, and do not involve events that require the process to wait, such as responses from human participants. For synchronous processes, transaction integrity follows the "ACID" principle – transactions must be Atomic, Consistent, Isolated, and Durable. Generally, ACID qualities are achieved by the process' beginning a transaction and then committing or rolling-back based on whether all steps in the process execute successfully. Such a transaction ensures that the affected data remains in a consistent state. Asynchronous processes can take hours, days, or weeks to execute, and thus require a different idea of transaction integrity. For asynchronous processes, such as those that require interaction with human participants, data consistency cannot be a measure of transaction integrity, as it may take some time until all steps in the process are executed and thus all database changes are complete. As a consequence, transaction integrity for asynchronous processes is the guarantee that process instances or states will not be lost. As long as the process instance remains in the system

despite any application or hardware failures, the enterprise ultimately can reach a consistent state. A SOA platform must support transaction integrity for both synchronous and asynchronous processes in order to provide truly enterprise ready applications.

Another important element of transaction integrity for BPM is the ability for the administrator to monitor processes and retry processes that have failed. In order to retry a failed process, the BPM solution must be able not only to compensate for successfully completed activities, but also be able to retain the process instance with its state and identify it to an administrator. Particularly when the process fails because an activity's target application is not available, the administrator can repair the outage and then retry the process. Such an approach is more reliable than requiring the application that called the process to become aware of the failure and then re-invoke the process later – such awareness could be quite impractical for a long-running process that may have failed days after it originally was invoked. Process monitoring for the BPM solution should allow an administrator to identify failed processes and also provide auditing of completed processes, including their data.

3.3 Transaction Integrity in Messaging

Enterprise Messaging provides transaction integrity by ensuring that messages will be retained as they pass between applications. In a messaging transaction the message server will not remove a message from a queue until it has added it successfully to the next queue in the application process. This transaction protects the message contents in event of an application error, a message server failure, or the unavailability of a target message server.

In order to assess the transaction integrity of an enterprise messaging server, the CPO performed a series of tests to determine how the message server protected messages under a variety of conditions. In the study, the message server had to manage a message flow from a sender to a receiver via several applications, each running on its own message server. The message server's transaction integrity was tested by generating failures to the application, the network, and the message servers. After each test, the message servers were inspected to ensure that transaction integrity was maintained; i.e., messages were not lost or duplicated. As the applications required to execute these tests were developed and executed, the CPO also identified development or deployment difficulties that might delay completion of the application or affect some part of the IT environment.

4. IBM Delivers Comprehensive Transaction Integrity

The results of the Competitive Project Office study clearly leads the CPO team to believe that IBM not only provides better transaction integrity for each platform component that was studied, but it is the best vendor to deliver end-to-end transaction integrity.

4.1 IBM provides better transaction integrity for application servers

In the CPO study, IBM WebSphere Application Server passed all of the tests that exercised transaction integrity. Both with and without a load, WebSphere Application Server coordinated a transaction between two databases and kept them consistent regardless of application, network, and power failures to which it was subjected.

4.1.1 WebSphere Application Server retains transaction integrity even under heavy load

In the study, the application was subjected server to loads of fifty thousand orders in a sixty minute period. With this load testing, the goal was to determine whether the application server would provide transaction integrity in a production-like environment.

Transaction integrity means that it could not tolerate data inconsistencies, including any partial or incomplete orders, any orders completed in one database but not in the other, and data corruption in any table.

During the test, different types of failures were introduced in an attempt to “break” the application and try to cause data consistency problems. The application server was tested with the following three types of failures:

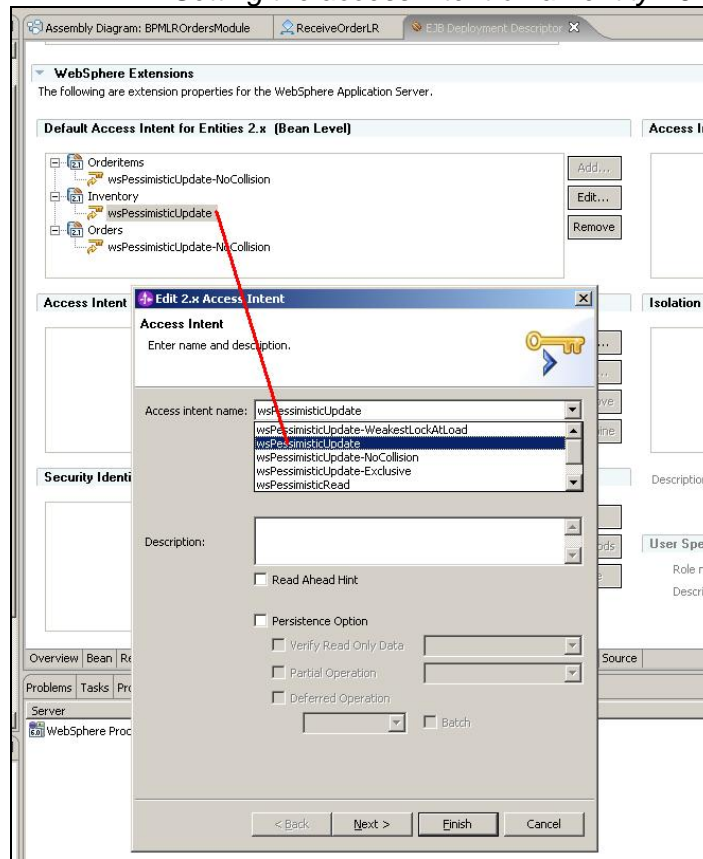
- Application failures – caused the application to throw exceptions
- Network outages - unplugged network cables at the databases and the application server
- Power outages - unplugged the power to the databases and the application server

WebSphere Application Server retained complete data consistency and high performance under heavy load while enduring all of these failures. Other vendors had degraded performance and lost transactions or updated databases inconsistently, resulting in inaccurate and corrupted enterprise data.

4.1.2 WebSphere Application Server builds transaction support into the server

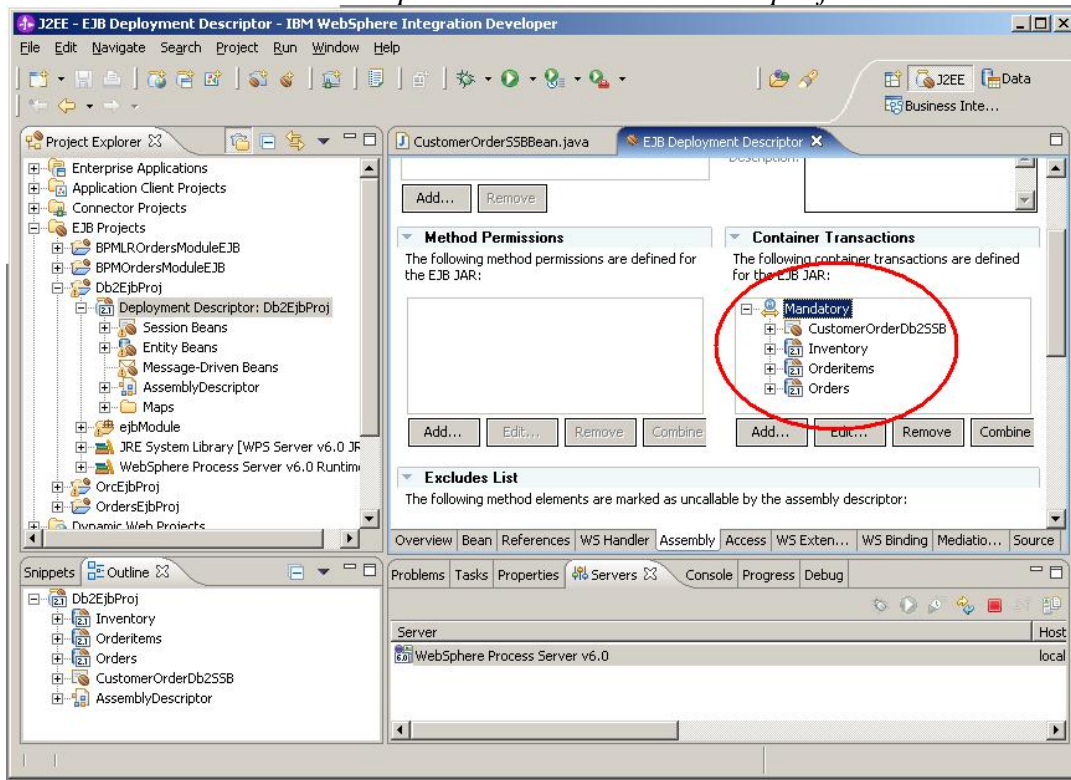
After the developer generates an entity bean, he still must configure Access Intents and Transaction Scope before he can deploy it. Access Intents tell the application server how applications will use the associated database table. For example, in this study it is expected that quantities in the Inventory table will be updated while multiple, concurrent order compete for access. As a result, the Access Intent to “PessimisticUpdate” must be set, which means that the assumption is that an application that reads a record in the table intends to update that record. With the “PessimisticUpdate” setting, the Inventory entity bean will tell the database to lock the row when it is read so that only one order is allowed to update it at a time. For any subsequent request for that row, the database will place it in a lock wait status until the prior request finishes with the row.

Setting the access intent on an entity EJB



In addition to setting the Access Intent, the developer also may set the Transaction Scope for each entity bean. Although the entity bean does not require that the developer set this value, developers consider it a good practice to do so. For the CPO study, the developer set all three entity beans to a Transaction Scope of "Mandatory," which means that the application that invokes the entity beans must pass to them an existing transaction in which they can participate. Unlike the Transaction Scope of "Required," an EJB set to "Mandatory" cannot begin its own transaction if one is not started for it. Like entity beans, session beans may specify a Transaction Scope.

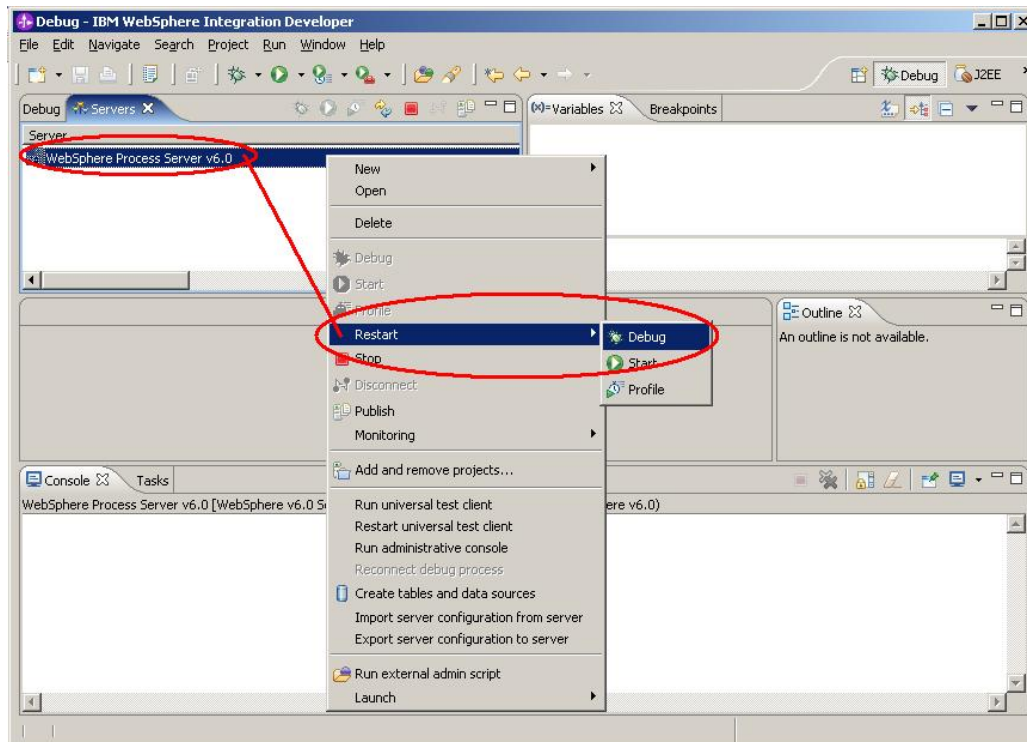
Here is how a developer sets the 'Transaction Scope' for a bean:



Setting the Transaction Scope takes the developer moments through the graphical deployment descriptor editor shown in the above figure. Unlike other vendors' solutions, WebSphere allows the developer to setup exactly the same entity beans on top of DB2 and Oracle. Other vendors require the developer to fine tune the settings for different databases, adding time, effort, and errors to development.

4.1.3 WebSphere Integration Developer makes it easy to test applications

In addition to wizards that reduce the time a developer needs to create application components like EJB's, WebSphere Integration Developer provides a built-in JNDI explorer and EJB testing tool that allow the developer to test EJB's without having to write any test code. WebSphere developer tooling includes a fully functional built-in test server; in fact, the built-in test server is a real production-ready server. In addition, the development tool can be linked directly to yet another test server on a remote machine if so desired. This means the developer can hot deploy his application to the internal test server or an external server or both. In either case the developer has full debug ability of the application, including break points and inspection or alteration of data variables on the fly.



This testing architecture of WebSphere tooling is what makes the developer highly productive. The ability to alter a piece of application code and instantly hot deploy the change saves the developer hours of time during the “alter, deploy, and re-test” cycle, which is required and often used by any application developer. Other vendors do not provide a built-in testing capability, requiring the developer to create a separate test client, adding time and effort.

4.1.4 WebSphere Application Server provides better transaction integrity than BEA

BEA Workshop for WebLogic does not have the ability to choose access intents like WebSphere, and instead makes the developer specify different isolation levels for each of the different databases. This makes development more difficult and time consuming because of the extra research required determining how to handle each database.

Unlike WebSphere, BEA Workshop for WebLogic does not have the ability to introspect database tables and create EJB Entity beans from them. As a result of this deficiency, development time is considerably longer and more difficult. The developer would have to write several lines of code for each column in a given database table (Workshop uses annotations, so the specifications go inside the bean code file itself rather than directly into the deployment descriptor). If a table has a large number of fields, then the developer will spend a significant amount of time doing the initial bean development. Some of these annotations are very difficult to code, especially the Container Managed Relation (CMR) fields, which define the relationships between beans (tables). All of this difficult and time consuming work is done for the developer with IBM's tools.

Finally, BEA Workshop for WebLogic does not have a tool dedicated to EJB testing. This means that the developer will either have to write his own testing harness or he will have to wait until the entire application (in the case of this study, six entity beans, three session beans, and a front end) is complete in order to test it. Writing one's own testing harness will lead to extended development time and effort, while waiting until completion to test results in "big bang" testing, which is the opposite of what most experts believe to be a best practice for software development.

4.1.5 WebSphere Application Server provides better transaction integrity than Microsoft

Although transaction support is a critical function of every application server, Microsoft warns that enabling XA transactions (needed for distributed transaction) can lead to "serious data corruptions" on remote resource providers, and denial-of-service attacks on the server. XA transactions are dangerous in Windows Server because when they are enabled, Microsoft's Distributed Transaction Coordinator (DTC) loads a set of user-specified DLL's into its own process in the Windows kernel, without checking them in any way. Instead, it just warns the user of the potential dangers. Other operating systems have strict rules for adding user-specified code into the kernel. Given the pervasive use of XA transactions in enterprise applications, Windows' implementation of XA transactions opens a huge security and reliability exposure in enterprise systems.

Microsoft's .NET Framework has two different programming models for transactions. If both are used in the same program, the developer must ensure they interoperate correctly; otherwise they will likely interfere with each other and can cause data integrity problems and program failures.

With Microsoft each database system has its own programming language. When an application requires multiple databases, the developer must learn how to use each database system's language correctly for transactions to operate properly in the application. The need to use different languages correctly can increase development time and delay the application's availability.

4.2 IBM provides better transaction integrity for Business Process Management

4.2.1 WebSphere Process Server provides unrivaled transaction support

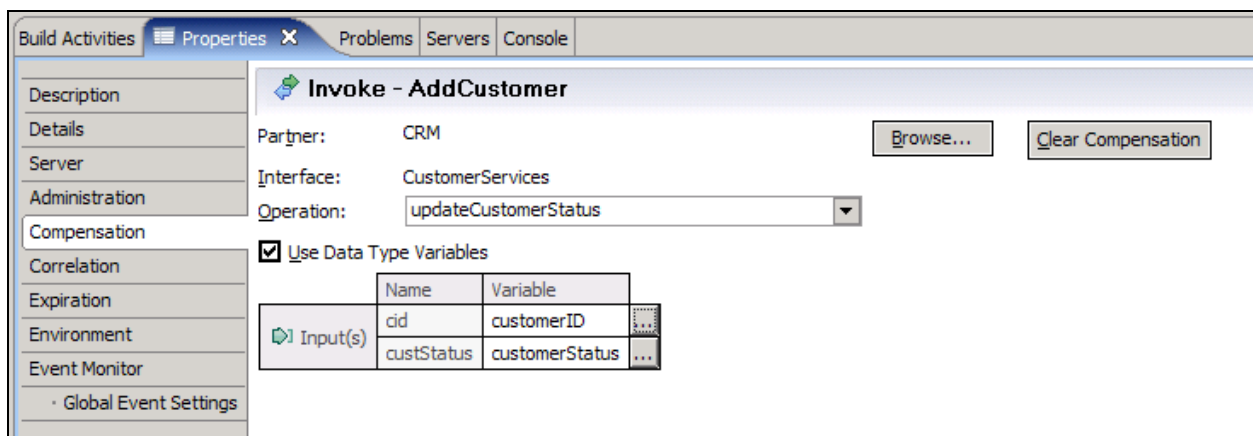
If a process cannot treat a business process as a single unit of work, it cannot ensure that applications that it updates will remain synchronized at all times. If instead the process treats each activity as its own transaction, the process cannot simply rollback the completed activities when a failure occurs: it must perform an "un-do," or compensating, activity for each of them. However, providing compensating activities is not enough to ensure data integrity, as resource updates will be visible to other applications until the compensating activity executes, leading to a potential loss of data integrity. For example, imagine an order process that updates both inventory and order tables. While the process is attempting to update the order table, other processes will be able to access the updated inventory. If the update to the order table eventually fails and the activity is compensated, those other processes would have made decisions based on inaccurate inventory data.

WebSphere Process Server, by comparison, can treat the entire process as a single transaction provided that the process is synchronous; i.e., the process can be executed without interruption. With WebSphere Process Server even a process with interruptions can have transactions within it where consecutive activities are synchronous.

WebSphere Process Server not only can treat short-running processes as a single transaction, but it also can optimize its execution. Unlike long-running processes that complete without notifying the requestor, a short-running process experiencing a failure immediately would return an exception to the requestor. Since the process is short lived, WebSphere Process Server can run it without saving its state, significantly improving its performance.

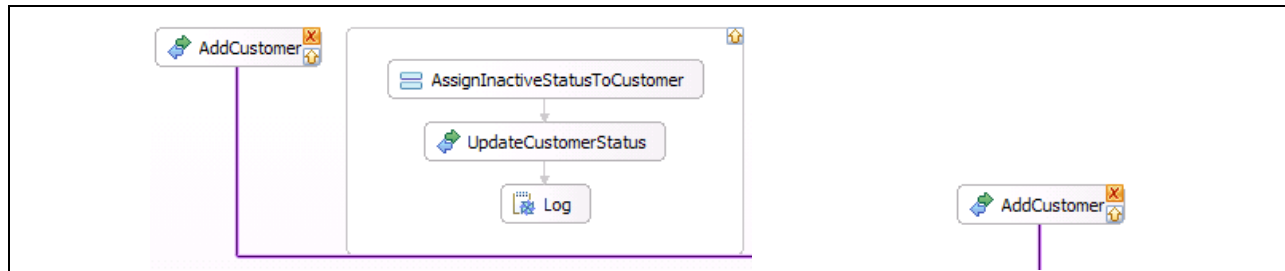
4.2.2 WebSphere Process Server delivers a comprehensive compensation framework

IBM WebSphere Integration Developer (WID) and WebSphere Process Server allow business process developers to easily define fault handling and compensation using a business process designer. The fault handlers and compensation blocks can be associated with the process activities that have transactional boundaries. In a simple case, the compensation can be specified as calls to the process partners in the activity properties without any additional programming effort:



Compensation defined as Invoke activity property

Defining the compensation action as an activity's property allows the business flow diagram to remain clear and uncluttered. In a more complex situation of a long-running process, business compensation might need more than a single step. In this case, it can be defined on an activity using compensation block as illustrated by Figure 2 below. The compensation block code can be collapsed by clicking on the compensation icon, so the developer can keep the diagram uncluttered.



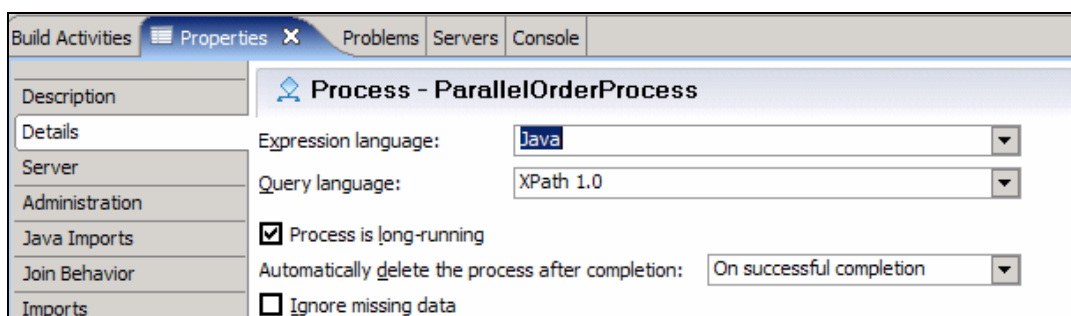
An activity with expanded (on the left) and collapsed (on the right) Compensation block

WebSphere Process Server's long-running processes have default fault handlers that will automatically execute compensations that were defined for all successfully completed process steps. There is no need to code this behavior, it is part of WebSphere Process Server's implementation of the BPEL specification.

4.2.3 WebSphere Process Server effectively can handle failed activities and terminated process instances

With a synchronous process that has a short completion time, it is reasonable to expect a response that indicates whether the desired actions succeeded. In a case when a process is interruptible (e.g., one with human-based activities or with asynchronous invocations of services), a party that calls the process will be notified immediately that the request was received but will not know the final outcome of the process right away. A BPM solution must assure the process requestor that the process eventually will reach completion. If the process fails the BPM solution must retain the process data and allow an administrator to restart the process when the cause of the failure is resolved.

As noted in the previous section, WebSphere Process Server automatically will compensate completed process steps through its default fault handling and its simple compensation framework. However, these capabilities would be insufficient for overall business transaction integrity if there were no means to retain a process instance with its business data for future restart. WebSphere Process Server retains a process instance simply by the developer's setting a property of the process definition. As shown in the figure below, the process developer can set the property "Automatically delete the process after completion" to "Yes", "No" or "On successful completion". If the process is configured to be deleted only on successful completion, its instance will be retained until the cause of the failure is corrected and the instance is restarted by an administrator.



Setting the process level property "Automatically delete the process after completion"

A common situation in a BPM solution is a business process that integrates various systems within an enterprise or across multiple enterprises by invoking components (e.g., Web Services) provided by disparate applications. Clearly, it would be unreasonable to assume that all systems would be available at all times. A truly enterprise ready BPM solution must handle gracefully all system and communication failures. The BPM solution must be able to recognize that a failure was caused by inability to reach an external component which may become available at later time. In such situation, compensating previous successfully completed steps might not be the most effective course of action. WebSphere Process Server intelligently handles a failure to invoke an external service – it stops the failed invoke activity and leaves the process instance in running state, allowing an administrator to resume the process from the point where it was stopped instead of executing compensation, terminating, and restarting the process.

4.2.4 WebSphere Process Server provides process monitoring without development effort

IBM's BPM solution goes even further in providing a complete solution for handling process failures. WebSphere Process Server includes out-of-the-box the means for business process monitoring on both the process instance and activity level, which is critical in order to deliver transaction and data integrity for long-running processes. WebSphere Process Server includes not only comprehensive server administration tools for configuring options, tuning the server, and resolving technical issues, but it also delivers an unrivaled process monitoring tool – BPC Explorer. BPC Explorer has facilities that allow an administrator to monitor each process instance state and drill down on an activity to determine a proper action in case of failed or stopped activity. The BPC Explorer gives an administrator a complete view of all process templates, instances and activities within each instance, with all of the details necessary for taking corrective action in the event of a failed or stopped activity.

Process Instance

Use this page to view information about a process instance and, optionally, to work on the process instance. ⓘ

Process Description

Process Instance Name `_PI:90030118.864c158d.fbffff5.f0e50000`

Description

State `Terminated`

Activity Name	State	Kind	Owner	Activated
startOrder	Finished	Receive		3/6/08 5:52:56 PM
Reply	Finished	Reply		3/6/08 5:52:58 PM
GetOrderXML	Finished	Script		3/6/08 5:52:58 PM
GetCustomerBO	Finished	Script		3/6/08 5:52:59 PM
InitCustomerAndOrderInfo	Finished	Assign		3/6/08 5:52:59 PM
Join	Inactive	Empty		
LogResults	Finished	Script		3/6/08 5:52:59 PM
Wait1	Terminated	Wait		3/6/08 5:52:59 PM
AddCustomer	Failed	Invoke		3/6/08 5:52:59 PM
startCheckInventory	Finished	Invoke		3/6/08 5:53:04 PM
Log2	Finished	Script		3/6/08 5:53:05 PM
ThrowCRMServiceNotAvailable	Failed	Throw		3/6/08 5:53:05 PM
AssignInactiveStatusToOrder	Finished	Assign		3/6/08 5:53:05 PM
Log	Finished	Script		3/6/08 5:53:05 PM
Terminate	Finished	Terminate		3/6/08 5:53:06 PM

Items found: 15 Page 1 of 1 Items per page: 20

Process instance activities view in BPC Explorer

4.2.5 WebSphere Process Server supports the complete process lifecycle

WebSphere Process Server delivers on the SOA promise of supporting changing business condition by providing comprehensive and safe versioning of the business processes. This capability is especially important for long-running processes, which are more likely than short-running processes to run during the lifespan of different process versions. WebSphere Process Server ensures that process instances behave as expected: existing instances continue with the old version and new instances are created with the new version of the process.

Another typical situation is when the process does not change from the business perspective, but requires redeployment because of a code update (e.g., a bug fix). If instances of the updated process are running, WebSphere Process Server would not allow the administrator to stop and deploy the updated process until all instances are completed and deleted from the server. By ensuring that process instances do not change in the middle of their execution, WebSphere Process Server prevents the loss of transaction integrity and possible process instance corruption.

If a BPM solution does not manage process updates like WebSphere Process Server, the application could experience unexpected results and a loss of transaction integrity, as process behaviors could change in the middle of a process. A developer could be compelled to make an additional copy of the process with the desired changes, increasing the development effort for him as well as for those applications that invoke the process.

4.2.6 WebSphere Process Server provides better transaction integrity than Oracle

Unlike WebSphere Process Server, Oracle BPEL Process Manager cannot retry or resume a failed process instance when the underlying problem has been solved. Failed process instances can only be deleted, which compromises transaction integrity. This places much more burden on the Oracle process developer to handle any and all process exceptions so that process instances never fail.

When the Oracle process deployment is overwritten without changing the version number, all running process instances that were started using the overwritten deployment are immediately terminated and placed in a "stale" status. The only available option for "stale" process instances is to delete them, which compromises transaction integrity. When a WebSphere Process Server process deployment is overwritten, existing process instances continue to be processed using the original process deployment, while new process instances use the new process deployment.

In addition, Oracle BPEL Process Manager violates two separate fault handling provisions in the BPEL specification. As a result of these violations, processes can be deployed that handle faults in unexpected ways that can compromise transaction integrity. WebSphere Process Server handles faults in compliance with the BPEL specification.

4.2.7 WebSphere Process Server provides better transaction integrity than BEA

AquaLogic BPM cannot treat a synchronous process as a single unit of work. As a result, in order to restore the enterprise's data when a process activity fails, the developer must code a compensating or "un-do" step for each successfully completed activity. With compensation the process exposes the enterprise to a loss of data integrity.

While WebSphere Process Server can stop a process that experiences a failure so that an administrator can resume it when the underlying problem has been solved, in the event of a failure AquaLogic BPM will compensate completed process steps and then lose the request. Compensating instead of resuming a process under some circumstances can cause a loss of data integrity if compensation fails. To avoid losing failed processes, a BEA developer would have to code compensation logic in order to keep the process data and retry it later.

AquaLogic BPM's compensation framework requires greater development effort than IBM's. While IBM's compensation steps are hidden in the process diagram, AquaLogic BPM's steps clutter the process, making it harder to maintain. In addition, invoking system-based steps in AquaLogic BPM requires the programmer to code in BEA's proprietary PBL language, so creating compensation steps creates extra code that must be maintained.

4.2.8 WebSphere Process Server provides better transaction integrity than Microsoft

BizTalk cannot update business applications or databases as a single unit of work. As a result, in order to restore the enterprise's data when a process activity fails, the developer must code a compensating or "un-do" step for each successfully completed activity. With compensation the process exposes the enterprise to a loss of data integrity, and significantly increases the developer's burden.

BizTalk cannot automatically save and restart failed processes. WebSphere Process Server can retain a failed process so that an administrator can restart (from the beginning) or resume (from the point of failure) when the underlying problem has been solved. This is true whether error handling has been built into the application. BizTalk will allow an administrator to resume from the point of failure – but only if the error wasn't caught and handled by the orchestration. There is no built-in facility for an administrator to restart an orchestration from the beginning.

BizTalk runtime administration is confusing and error-prone. The administration of a running BizTalk installation is provided by the BizTalk Server Administration tool and by Health and Activity Tracking. Neither tool presents an effectively organized view of the status of BizTalk orchestrations and their component parts. Information about the orchestration instances is intermingled with an explosion of information about internal BizTalk messaging. There is no ability to drill down into a problem through successive layers, as there is with WebSphere Process Server administration. This makes it much harder to monitor the health of the system and to diagnose runtime problems.

4.3 IBM provides better transaction integrity for messaging

4.3.1 IBM WebSphere MQ provides outstanding protection against Application Failure.

During a situation where a message is read from a queue by an application and the application is terminated by either an error or unknown condition before the message can be processed and a COMMIT can be performed on the transaction, what happens to that message? Is the message lost, since it has been read from the queue?

In this situation, IBM WebSphere MQ will detect that the application has lost connectivity without performing a COMMIT on the most recent read of the queue. IBM WebSphere MQ will then perform a rollback of that message back on to the queue, preventing data loss, and preserving the transactional integrity of the message.

If that queue has been configured to handle poison messages, and the *back out threshold* is equal to one, IBM WebSphere MQ will remove the message from the queue and place it onto the *back out queue* for examination by the system administrator, while allowing the processing of all other messages contained within the application queue to continue.

4.3.2 IBM WebSphere MQ provides exceptional Service Failure protection.

During a situation where one server is sending messages to another server using store-and-forward technology, what will happen to the messages if the receiving server encounters an unexpected power down situation? Will the message transfer experience message loss?

The architecture of IBM WebSphere MQ during store-and-forward message transfer contains procedures to prevent data loss as transmission queues on client machines pass messages to the destination queue on the determination server. If an interruption of service occurs between the two machines during the transfer of a message, IBM WebSphere MQ handles the interruption without allowing duplicate messages or message loss, thus ensuring transaction integrity.

In this way, IBM WebSphere MQ prevents data loss or duplication of messages during service failure or power interruption.

4.3.3 IBM WebSphere MQ provides excellent protection against Network Failure.

During a situation where one server is sending messages to another server using store-and-forward technology, what will happen to the messages if the receiving server encounters an unexpected network failure? Will the message transfer experience message loss?

The architecture of IBM WebSphere MQ during store-and-forward message transfer contains procedures to prevent data loss as transmission queues on client machines pass messages to the destination queue on the receiving server. If an interruption of service occurs between the two machines during the transfer of a message, IBM WebSphere MQ handles the interruption without allowing duplicate messages or message loss, thus ensuring transaction integrity.

In this way, IBM WebSphere MQ prevents data loss or duplication in the event of Network Failure.

4.3.4 IBM WebSphere MQ provides outstanding handling of Queue Manager unavailable conditions.

During a situation where one server is sending messages to another server using store-and-forward technology, what will happen to the message transfer if the receiving queue manager is unavailable? Will the message transfer experience message loss?

IBM WebSphere MQ has designed its store-and-forward queues to use transmission queues to transfer messages between queue managers. The transmission queue is located on the client queue manager. When the destination queue manager is not available, the client queue manager simply shuts down the channel, but stores all of the messages within the transmission queue. The application is not aware that the messages have not yet been sent to the destination queue manager, because the messages have simply been temporarily placed on the transmission queue. Therefore there is no interruption of service. Once the destination queue manager becomes available, the client queue manager starts sending its message to the destination queue manager. This process is transparent to the application, and preserves transaction integrity.

4.3.5 IBM WebSphere MQ provides better transaction integrity than Oracle

Oracle does not support persistent store-and-forward queues or topics. IBM WebSphere MQ refers to store-and-forward technology as remote queuing, while Oracle refers to store-and-forward technology as propagation. IBM WebSphere MQ supports store-and-forward queue capability for persistent queues, but Oracle does not. However, Oracle does support store-and-forward technology for express queues. The use of persistent queues is the de facto industry standard for store-and-forward technology. Since Oracle does not provide this function, enterprises that use Oracle's store-and-forward technology with express queues are exposed to a great risk of potential message loss.

Oracle AQ defines a topic as a multiple-consumer enabled queue. Oracle AQ only supports persistent messages for topic-based store-and-forward technology. A multi-consumer queue is accessed programmatically as a topic, not a queue. This exposes enterprises using Oracle's store-and-forward technology to additional architectural complexity.

4.3.6 IBM WebSphere MQ provides better transaction integrity than Microsoft

While MSMQ 3.0 clients can transactionally send messages to remote servers, *clients cannot transactionally receive messages from remote servers*. This deficiency causes an enterprise using MSMQ 3.0 to handle this situation in one of three methods. First, the client application can be placed on the server directly, making the queue local; this may violate design requirements. Second, the client can receive messages from the remote server in a non-transactional manner, resulting in potential data loss. Finally, the server can place the messages on a store-and-forward queue transactionally, sending the messages to a local queue on the client; this adds complexity and requires the use of additional resources.

Unlike WebSphere MQ, MSMQ 3.0 does not contain any function to handle "poison messages." A poison message is a message that cannot be handled properly by the application, and is rolled back onto the queue repeatedly, effectively preventing the subsequent messages from being read. This lack of function exposes the vulnerability of applications using MSMQ 3.0 to processing inefficiency, belated or tardy processing of messages, and potential data loss.

Microsoft claims that MSMQ 4.0 supports the transactional reception of messages from remote queues as well as the proper handling of poison messages. However, it is important to note that MSMQ is tied to the level of the Windows® operating system. Therefore if the enterprise chooses to use MSMQ 4.0, it must upgrade all of its clients and servers to either Windows Vista® or Windows Server® 2008. This not only forces the enterprise to incur the expense of acquiring additional licenses, but also incurs the expense of additional regression testing, since the new versions of Windows Vista and Windows Server 2008 contain new security features that have been known to break existing applications. Furthermore, Microsoft has reported that in some rare instances, incompatibilities may occur within environments using both MSMQ 4.0 and previous versions of MSMQ.

5. IBM Delivers the Best End-to-End Transaction Integrity

Transaction integrity is the key to making SOA applications reliable and scalable in enterprise environments. Transaction integrity ensures that an SOA application that integrates services from separate applications can update these applications consistently. Without consistent data, SOA applications become a liability to a business instead of a way to innovate and create business value.

When choosing an SOA platform, enterprises should consider end-to-end transaction integrity as a critical requirement. A customer that is considering IBM and other solutions should compare each solution's ability to deliver the following transaction integrity capabilities:

- Which solution can ensure data consistency when application server failures occur, even under heavy load?
- Which solution makes it easier to create J2EE-standard components that access data and ensure transaction integrity when included in enterprise applications?
- Which solution can treat all of the activities in a short-running business process as a single transaction and thus ensure data consistency?
- Which solution allows a failed business process to be restarted by an administrator to ensure that the request will not be lost?
- Which solution ensures that messages will not be lost and can be delivered regardless of any application, server, or network failure?

Oracle, BEA, and Microsoft struggle to deliver these capabilities that ensure transaction integrity. Their deficiencies result in an application that takes longer to develop and exposes the applications with which it interacts to a greater risk of loss of data consistency. With strengths in application server, business processes, and enterprise messaging, IBM allows SOA applications to access enterprise services with confidence and assured transaction integrity.

6. Appendix A: Products Used for the Study

The following table details the product versions that the Competitive Project Office used for the study. Each was the latest available when the study began.

	IBM	Oracle	BEA	Microsoft
Application Server	WebSphere Application Server 6.0.2	Oracle Application Server 10.1.3.3	WebLogic Server 10.0	Windows Server 2003
Business Process Management	WebSphere Process Server 6.1	Oracle BPEL Process Manager 10.1.3.3	AquaLogic BPM 6.0	BizTalk Server 2006 R2
Enterprise Messaging	WebSphere MQ 6.0	Oracle AQ 11g	WebLogic Server 10.0	MSMQ 3.0

7. Appendix B: Additional Resources

For more detail on each element of the study, please read the following papers:

- “IBM WebSphere Application Server Delivers Transaction Integrity for J2EE Applications”
- “IBM Delivers Outstanding Transaction Integrity for Business Process Management”
- “IBM WebSphere MQ Delivers Better Transaction Integrity for Enterprise Messaging”

The following papers address other areas where IBM delivers outstanding enterprise readiness.

- “IBM's Smart SOA Approach Delivers Enterprise Readiness”
- “IBM's Smart SOA Approach Delivers Enterprise Readiness Better than Microsoft”
- “IBM's Smart SOA Approach Delivers 21st Century Transaction Processing”
- “IBM's Smart SOA Approach Delivers Enterprise Readiness for Information Management”
- “IBM's Smart SOA Approach Delivers Enterprise Readiness for the Enterprise Service Bus”
- “IBM's Smart SOA Approach Delivers Better IT Service Management than Microsoft”
- “IBM's Smart SOA Approach Delivers Enterprise Ready Collaboration”

Please contact your IBM representative for a copy of any of these papers.

©Copyright IBM Corporation 2008

IBM Corporation
Software Group
Route 100
Somers, NY 10589
USA

Produced in the United States
June 2008
All Rights Reserved

IBM, the IBM logo, and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

BEA, AquaLogic BPM Suite, WebLogic, and BEA Workshop are trademarks or registered trademarks of BEA. Other company, product or service names may be trademarks or service marks of others.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both..

Other company, product or service names may be trademarks or service marks of others.

The information contained in this documentation is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this documentation, it is provided "as is" without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this documentation or any other documentation. Nothing contained in this documentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM (or its suppliers or licensors), or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.