The benefits of using IBM Business Process Manager Advanced

SOA, process integration, tools, and more

Stuart C. Jones
Paul Pacholski

June 20, 2012

IBM® Business Process Manager is provided in two major configurations: Business Process Manager Standard and Business Process Manager Advanced. The benefits of using Business Process Manager Standard are fairly well understood, but the incremental benefits of using Business Process Manager Advanced are not as well understood. This article is intended to address that situation

Introduction

In June of 2011, IBM released IBM Business Process Manager V7.5. This new product integrated the best capabilities of WebSphere® Lombardi Edition and WebSphere Process Server.

The objectives of this article are as follows:

- Explain the capabilities and benefits of Business Process Manager Advanced.
- Explain the kinds of integration capabilities that are available with Business Process Manager Advanced and IBM Integration Designer. In particular, show how straightforward it is to build integration capabilities without coding.
- Show how the integration capabilities of Business Process Manager Advanced can supplement and extend Business Process Manager Standard and enable straightforward support of more advanced use cases.

The target audience for this article is one who is familiar with IBM Business Process Manager Standard but has little experience of IBM Business Process Manager Advanced. However, anyone interested in understanding the distinguishing characteristics of IBM Business Process Manager Advanced will benefit.

Process integration challenges

Building business processes that consist of end-user interactions coupled together with a few web services and database calls is a fairly straightforward undertaking. Indeed, if the world consisted strictly of web services and relational databases, we might think that integrating components together with business processes was a done deal. However, there are several concerns that serve to make the integration of processes and the underlying systems that they interact with much more complex:

1. We do not live in a world where all interactions are well-defined – either by WSDL or by SQL. Even where these are available some choose not to use them.
   We do live in a world where there are data sources that do not support WSDL or SQL, where interfaces to systems of record are defined by messaging, TCP/IP, SNA (yes, you still see these on occasion!), proprietary data formats and unusual, and one-off integration mechanisms.
   We also live in a world where, even though there are WSDL interfaces, they may use complex XML schema or may involve more than just the basic WSDL capabilities, such as WS-Security.
2. The interactions with underlying systems are not always successful. Systems might be unavailable when requests are made or fail in flight, they might respond too slowly for the calling business process, the input data might be incorrect, and any number of other failure scenarios.
3. There are cases where the business process requires that the entire process needs to be undone in a controlled manner – a credit check might fail, a risk assessment might be unfavorable, and so on. In these cases, any system interactions that have already completed need to be reversed.
4. Data mapping requirements are often complex; for example, those that require content-dependent mappings, contain lists, or require rules or business logic.
5. Inbound interactions are required; for example, I want to start a process instance when a specific update is made to a particular system.

Why do we care about these cases? Don't all systems these days work most of the time? Don't they have fail-safes and back-ups so that they're always available? And if a system does fail, don't we always have activity and transaction logs so that we can go back and fix any problems? The answer to these questions is "yes, some of the time," which translates to "no, not all of the time." Let's look at some examples that illustrate the issues:

Financial transactions

Let's take a Business Process Manager environment and its connected systems, such as CICS®, IMS™ Rational Application Developer ™ and Oracle® Financials. This system, as a whole, is 99.999% reliable – pretty good by most standards. This is a medium-sized financial institution that processes 10 million transactions per day. Note that this is easily 50 times smaller than the largest institutions.

With 99.999% systems reliability, 0.001% of transactions will have some kind of issue that needs to be addressed. For this mid-sized company that is 100 issues every single working day that need to be taken care of in some way. For larger institutions, we're looking at 50 times that, or 5,000 issues per day!

So even with this fairly reliable system, the sheer volume of transactions being processed means that any problem, even a small one, means a lot of issues to fix every single day. The process integration capabilities available in Business Process Manager Advanced provide a way to either avoid problems altogether or provide an automated way to address problems once they have occurred. In particular, an automated recovery environment allows you to bypass the manual intervention that would otherwise be required.

Duplicating orders

Not every organization is a large financial institution, so let's also take a look at a more down to earth example. Let's say I have a solar powered fan on the roof of my house and the fan fails. It is still under warranty so I call the company that supplied the fan. The company agrees to replace the fan and sends an email requesting shipping details. When the shipping information is received, it will be entered into the shipping system by hand.

After a few days, I call the company to check on the status. I talk to a different person than previously. She makes a few checks and tells me that she is entering my details into the shipping system right now. The same morning, my original contact also enters my details into the shipping system. There are no facilities to check for duplicates.

After two days I receive my replacement fan. After 3 days I receive my second fan. The solar fan company is down one extra fan, plus shipping costs.

Handling failure scenarios with IBM Business Process Manager

The Business Process Manager Standard environment was designed to handle straightforward integration challenges. Specifically:

- While it's possible to build simple interfaces to backend systems out of the box, it can be quite involved to implement anything other than a simple style. It is not impossible, but there are multiple hoops to jump through and the implementation usually involves a good deal of JavaScript™.
- BPM Standard does not support transactions, failure recovery or compensation. Rather, IBM Business Process Manager Standard takes a fairly optimistic view of the processing environment and so does not provide significant out-of-the-box capabilities for addressing issues when they occur.

Fortunately, these types of advanced capabilities are addressed in the IBM Business Process Manager Advanced environment, which provides many connectivity capabilities and adopts a rather more pessimistic view to complement the more optimistic view of IBM Business Process Manager Standard. The Business Process Manager Advanced environment provides capabilities to handle issues when they arise. Combined together, the capabilities provided in Business Process Manager Standard and Business Process Manager Advanced provide a comprehensive solution for enterprise integration requirements.

Note: For most environments, a Business Process Manager Advanced configuration includes all of the capabilities of BPM Standard.

Business Process Manager process integration capabilities

One of the most successful component oriented, distributed computing models of recent years is SOA. SOA has been adopted by most vendors and a very large set of customers. Fully exploiting the services provided by SOA requires an orchestration component that can not only invoke the services in the correct order, but also handle failure, errors and any other challenge that comes along when there are many services on many different platforms in many different locations. The following sections describe the capabilities that are provided by IBM Business Process Manager Advanced to fully exploit an SOA infrastructure.

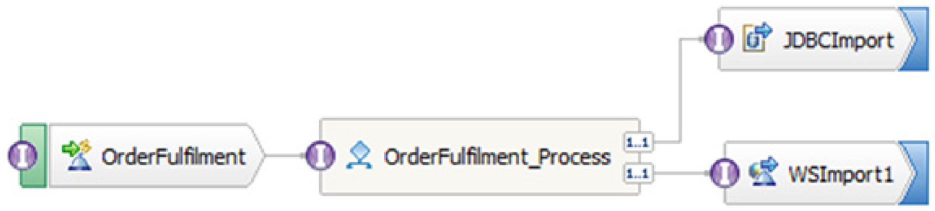
Service Component Architecture (SCA)

Today's operating environments are made up of many different moving parts, or components. Properly joining all of these components together is a major challenge, involving support of multiple connection types, different interfaces, global and discrete transactions, passing of security context and so on. The traditional approach to solving this issue was tightly coupled solutions where a process was tied to a specific WSDL, queue or URL.

SCA eases these challenges by providing a layer of indirection in which a calling component knows the interface its calling out to, but doesn't know or need to know to which implementation it is tied. SCA was part of WebSphere Process Server and is now available in Business Process Manager Advanced.

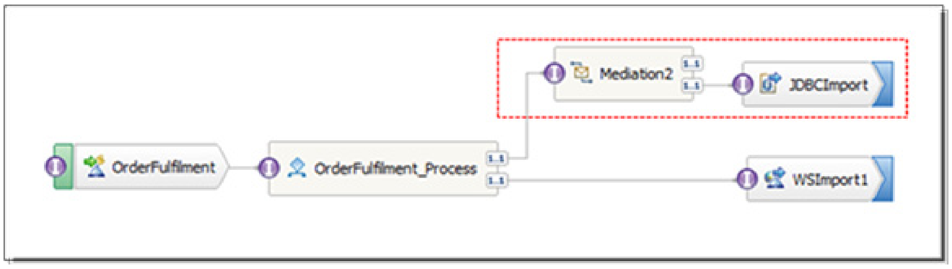
The IBM implementation of SCA provides a graphical representation of how components are associated with each other, as illustrated in Figure 1.

Figure 1. SCA wiring diagram



With SCA, the only concern is with the interface. This means that it is easy to add to or change the diagram without affecting the rest of the picture, as shown in Figure 2.

Figure 2. Easy substitution of components



Note: Figures 1 and 2 represent how components are joined together; they should not be viewed as flow from one side to the other or top to bottom.

Figure 2 shows how the components are connected, so beginning on the left, the required components are invoked in a call/return style, then the flow is exited the same way it was entered. The components in the diagram are SCA components. The simple representation in Figure 2 shows:

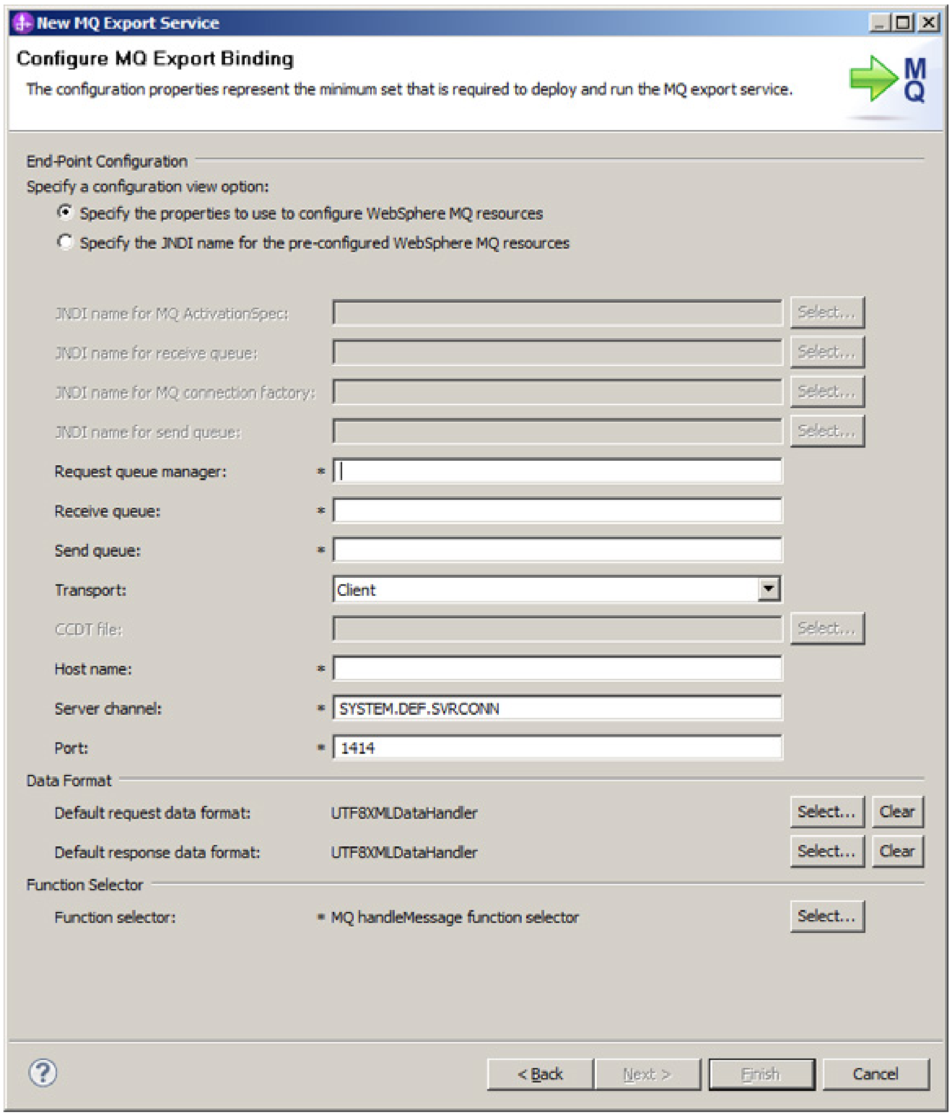
- On the left, a call from a Business Process Manager Standard component (this is an Advanced Integration Service (AIS) Import component).
- In the middle, a BPEL process component and a mediation component.
- On the right, calls to a database (JDBCImport is a call to a database using the JDBC adapter) and a web service call (WSImport1).

The available component types are processes, human tasks, Java™, rule groups, state machines and mediations.

To communicate with an SCA assembly, there is a set of exports and imports, three of which are shown above. These provide support for HTTP, JMS, generic JMS (enables the use of non-J2EE Connector Architecture (JCA) 1.5 Java Message Service (JMS) providers), WebSphere MQ, MQ over JMS, web services (SOAP 1.2, SOAP 1.1, SOAP 1.1 using JAX-RPC, SOAP 1.1/JMS), Enterprise Java Bean (EJB) and business process diagram (BPD) (to invoke a Business Process Manager Standard component), AIS (to enable invocation from a Business Process Manager Standard Component), IBM Case Manager and an optimized, internal SCA binding. Some of the advantages of these exports and imports are as follows:

- No programming is required, which makes it simple to use a target connection style by simply performing some configuration. In the WebSphere MQ example shown in Figure 3, all that is required is to complete the MQ server and queue name information (or use JNDI) and, in some cases, configure the data handler.

Figure 3. WebSphere MQ export binding configuration wizard



Configure MQ Export Binding

The configuration properties represent the minimum set that is required to deploy and run the MQ export service.

End-Point Configuration

Specify a configuration view option:

- [✓ Specify the properties to use to configure WebSphere MQ resources]
- Specify the JNDI name for the pre-configured WebSphere MQ resources

JNDI name for MQ ActivationSpec:	
JNDI name for receive queue:	
JNDI name for MQ connection factory:	
JNDI name for send queue:	
Request queue manager: *	
Receive queue: *	
Send queue: *	
Transport:	Client
CCDT file:	
Host name: *	
Server channel: *	SYSTEM.DEF.SVRCONN
Port: *	1414

Data Format

Default request data format:	UTF8XMLDataHandler
Default response data format:	UTF8XMLDataHandler

Function Selector

Function selector: *	MQ handleMessage function selector

This means connectivity can be implemented rapidly without any coding, and connection/session pooling are all handled under the covers.

- The imports and the exports all interchangeable. This makes it straightforward and non-disruptive, for example, to switch from a web service call to a JMS call.
- You can add components to the diagram without affecting the existing components (though you may have to match up interfaces). This is similar to the way that you build BPDs in Business Process Manager Standard.
- There is built-in support for different standards and capabilities, many of which are not supported in Business Process Manager Standard, such as different levels of SOAP support, web services security and many more.

Adapters

Many application environments provide simple, well documented interfaces to enable other components to access their capabilities. However, some popular applications have complex interfaces and arcane interaction styles. Others make it straightforward to pass information to an application but not so easy to get information out of it. IBM Business Process Manager Advanced provides the following set of inbound and outbound adapters to ease the challenges of accessing some of the more popular application environments:

- CICS (outbound)
- IBM Enterprise Content Management (inbound and outbound)
- IMS (outbound)
- Oracle JD Edwards EnterpriseOne (inbound and outbound)
- Lotus® Domino (inbound and outbound)
- Oracle eBusiness Suite (inbound and outbound)
- PeopleSoft™ (inbound and outbound)
- SAP® (inbound and outbound)
- Siebel® (inbound and outbound)

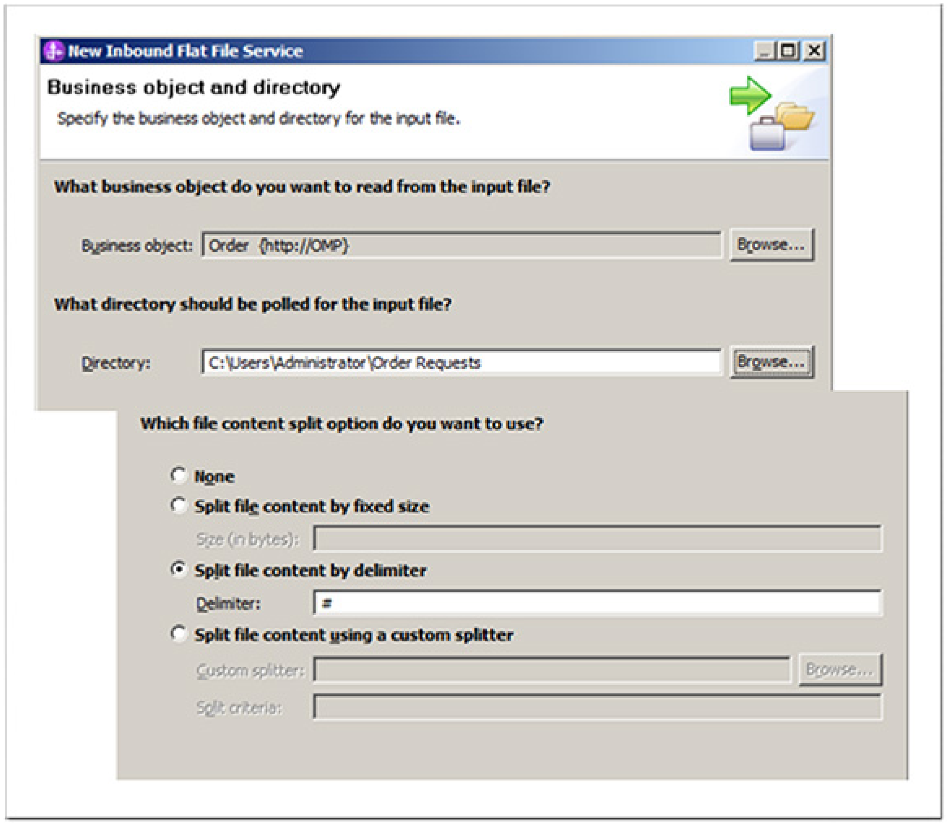
As well as adapters for applications, there are also adapters that help in connecting to the following common technology infrastructures:

- Email (inbound and outbound)
- Flat files (inbound and outbound)
- FTP (inbound and outbound)
- iSeries (inbound and outbound)
- JDBC Databases (inbound and outbound)

These standards-based adapters all conform to the J2C specification and are all available as SCA components, which means that they be integrated with other components using SCA. The JDBC adapter in the SCA diagram in Figure 1 illustrates this. All of the requirements for connecting to the target environment and exchanging data with it are encapsulated into each adapter.

Adapters make it much simpler to connect to resources and applications. For example, Figure 4 shows the required configuration to receive input from a file.

Figure 4. Flat file adapter configuration


New Inbound Flat File Service

Business object and directory

Specify the business object and directory for the input file.

What business object do you want to read from the input file?

Business object: Order {http://OMP}

What directory should be polled for the input file?

Directory: C:\Users\Administrator\Order Requests

Which file content split option do you want to use?

- ( ) None
- ( ) Split file content by fixed size
  - Size (in bytes):
- (•) Split file content by delimiter
  - Delimiter: #
- ( ) Split file content using a custom splitter
  - Custom splitter:
  - Split criteria:

Each adapter will have its own target-specific configuration. The file adapter configuration shown in Figure 4 is probably the most basic.

Integrated ESB

One of the common ways of connecting components is to use the capabilities of an ESB. An ESB can be used in conjunction with an SCA environment, shown as a mediation in Figure 2, or it can be used standalone. An ESB provides the following primary capabilities:

- Connectivity - Integration of different transports; for example, where one system uses JMS messaging and another uses WSDL/HTTP.
- Endpoint virtualization - The ESB is aware of the actual endpoints that should be used rather than requiring the caller to know. This is especially useful when there are choices of endpoints or the endpoints change frequently.
- Data mediation/transformation - A particular system represents data in one way and another represents the same data in a different way.
- Data encapsulation - Minimizes the data exposed to processes. Some systems, especially those that support standard data formats (such as EDI) require large, often sparse data structures as input and output. To avoid the need for a process to handle these data structures, the ESB does it instead.
- Data enrichment - Adding additional data fields to a message.

IBM Business Process Manager Advanced includes WebSphere ESB. WebSphere ESB is a good option for an ESB, but it is not a required component. Many customers will choose to use

WebSphere Message Broker, DataPower XI50/52 or even another vendor's ESB. All of these are valid choices. However, using the embedded WebSphere ESB has several advantages over a separate component:

- Same tooling, same component model, less to learn.
- The ability to use Process Center and its management and governance capabilities.
- Transaction propagation.
- The possibility of co-location when performance requirements demand such a pattern.
- Error handling policies such as time-out and retry count specification.

When faced with process integration challenges, it's tempting to think that Business Process Manager Standard and an ESB are sufficient. This configuration will certainly handle some challenges, most notably connecting to other systems, but there are other aspects of process integration that this combination cannot address. The next section describes these aspects.

Distributed transactions

The word transaction means different things to different people. In the context of this discussion, a transaction is an interaction with a system in which that interaction has ACID (Atomic, Consistent, Isolation, and Durable) properties. As described in the IBM CICS documentation, this means the following:

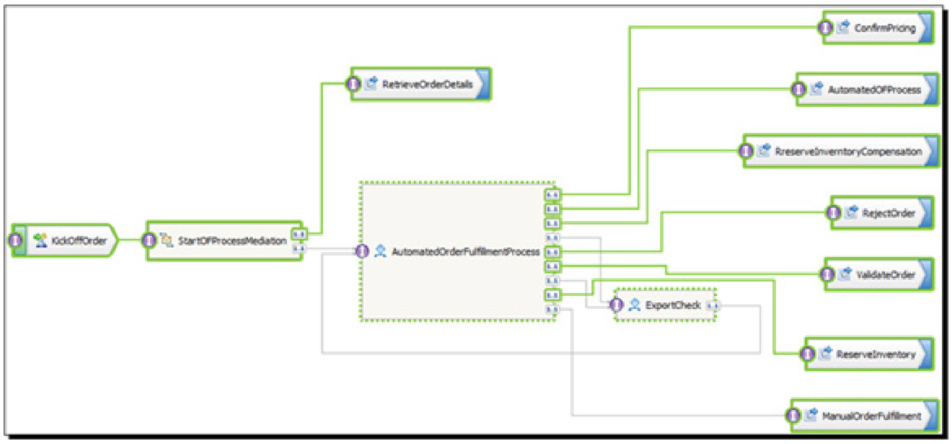
- Atomicity
  All changes to data are performed as if they are a single operation. That is, all the changes are performed or none of them are.
  For example, in an application that transfers funds from one account to another, the atomicity property ensures that, if a debit is made successfully from one account, the corresponding credit is made to the other account. A more romantic example is a wedding. There is no point at which only one of the parties is married – they're either both married or both not (yet) married!
- Consistency
  Data is in a consistent state when a transaction starts and when it ends.
  For example, in an application that transfers funds from one account to another, the consistency property ensures that the total value of funds in both the accounts is the same at the start and end of each transaction.
- Isolation
  The intermediate state of a transaction is invisible to other transactions. As a result, transactions that run concurrently appear to be serialized.
  For example, in an application that transfers funds from one account to another, the isolation property ensures that another transaction sees the transferred funds in one account or the other, but not in both, nor in neither.
- Durability
  After a transaction successfully completes, changes to data persist and are not undone, even in the event of a system failure. For example, in an application that transfers funds from one account to another, the durability property ensures that the changes made to each account will not be reversed.

What does this mean to non-techies? If something is a part of a transaction, then it happens reliably and consistently, and in an automated, recoverable way. This is not as straightforward as it sounds; when all of the systems and services are on the same server things are not so difficult, but consider what happens when the process engine is on one server and the target service is on some other server. Let's make things even more difficult; let's suppose that you have multiple services or applications all on separate servers, some fairly local, some very remote, and you have to have a single ACID transaction that spans them all! Further, if there's a failure in one of these systems, the transaction must be suspended and then recovered when the systems comes back, ensuring that all resources are properly synchronized.

This is the sort of capability that IBM has been supporting since the 1960s. In fact, this capability has been around for so long that many of our clients take it for granted, assuming that these capabilities are available in all systems. Well, they're not. When it comes to transactional integrity IBM rules the roost and – from a BPM viewpoint - IBM Business Process Manager Advanced is where this capability is supported.

In fact, the capabilities in Business Process Manager Advanced go beyond what's described above. The implementer of the process gets to control which systems are involved in which distributed transactions so that when discrete control is needed, it is there to be used. For example, as shown in Figure 5, in IBM Integration Designer, a developer may select how transactions will be handled using a graphical editor (a broken green line indicates that a transaction might be joined or propagated but the system cannot confirm that will happen).

Figure 5. Integration Designer enables setting of transaction quality of service



One last note on transactions. All systems will suffer failures at some point in their lives; the issue really is how well the system deals with the failure. One of the distinguishing characteristics of Business Process Manager Advanced is that it can deal with transaction recovery in an automated style – no human intervention is needed to figure out what went wrong and how to make it right. It's not enough to collect log entries and other information that enables things to be fixed, but the capability to recover automatically, rather than requiring manual intervention that distinguishes Business Process Manager Advanced. Just consider the amount of activity that a system processing 1000 transactions a second generates when one of its component parts fails and the advantages of automated recovery become very clear.

Compensation

As powerful as distributed transactions are, they don't cover all of the bases. There are two main scenarios that cause problems for distributed transactions:

- As suggested above, the ACID properties of transactions require holding onto, or locking, resources until everything in the transaction scope as been carried out, then commit everything at once. This locking means that no other processes can see the data. This is the all or nothing aspect of transactions. This is fine in the world of sub-second interactions, but it doesn't work so well in the world of business processes that can execute over an extended time period - even a minute or so is an eternity for today's processors. For this reason, for long-running processes it's generally not practical to include multiple services in a single transaction and so, in this long-running environment, services are committed on an individual basis. This is a somewhat optimistic model, though a necessary one. It works well if the process runs successfully to completion, but there are some challenges if that's not that case.
- Not all services and systems support the protocols that are necessary for distributed ACID transactions. For example, a file system is not transactional, so if a file (for example, an Excel spreadsheet) is involved in a business process, it cannot be a part of a distributed transaction. There is a second class of this scenario. There are some activities that are both non-transactional and irreversible. The simplest example of this is probably printing a letter that is destined for a customer. Once a print request has been fulfilled, there is no reverse action for the request, whereas for a file, you could consider removing the "offending" file, file insert, and so on.

When these kinds of activities are involved in a business process, transactions don't fulfill all of the requirements and you another set of capabilities called compensation. Compensation provides the ability to undo the actions carried out in a business process in an ordered, automated style. It includes the following capabilities:

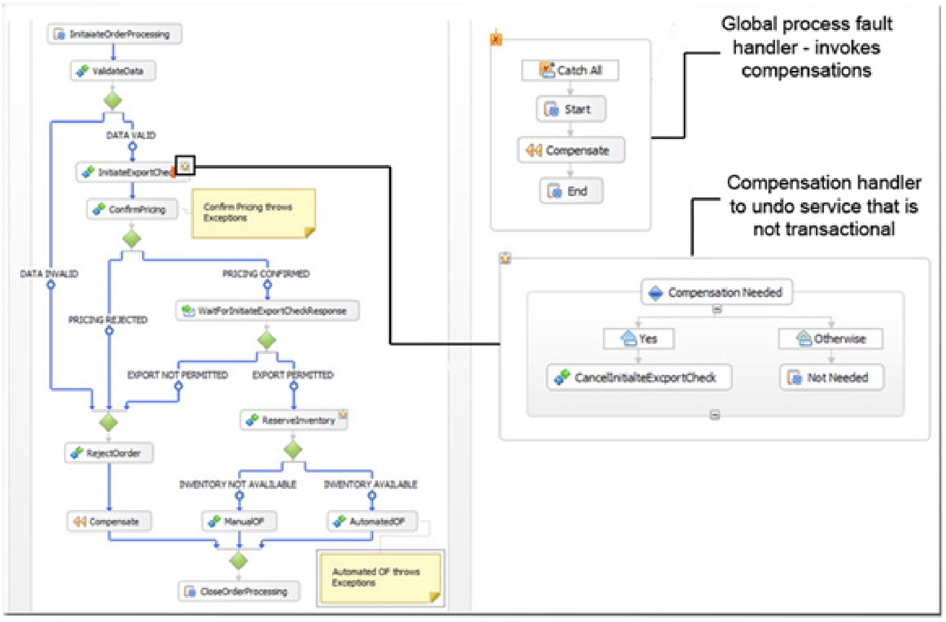
- Undo actions that correspond to all of the do ((forward) actions in a process. Very often, the undo action for a do action is easy to figure out. A delete corresponds to an insert/append for a database. If you added something to a database or file, you can delete it. If you deleted something, you can add it back (though you need to make sure to retain a copy of the deleted data just in case it's required.)
  Sometimes, though it is not so simple. Take the print request describe earlier. There is no "unprint," so instead an undo action might be to print some form of retraction or apology.
  The subtlety to consider here is this: transactions and simple undo logic for a database are technical IT activities, but printing a second letter for a customer is a business-driven activity rather than an IT-driven activity. Further, based on business considerations you might choose to have a much more sophisticated undo action, such as print the apology but if this is a gold customer, also call them or send them a text message.
- A compensation manager watches every instance of every process and notes all do actions and the order in which they were executed. There could be multiple paths in a business process, so this is not as straightforward as it might first appear.

There must be an activity to cause compensation to take place. In some cases, compensation might be driven by an IT-level failure in the process. In other cases, the decision to abandon this process instance (thus driving compensation) might be business-driven, for instance, based on a customer decision.

The compensation capabilities of Business Process Manager enable you to undo the previous actions of a process instance, some of which might have been transactional in nature. In addition, the compensation process is fully automated. You don't need to write reams of code to make this happen, because compensation is a core component of Business Process Manager Advanced. In some cases, you may still need to provide undo actions that are not provided by the service, or when the undo action is non-intuitive, as in the printing example.

Figure 6 shows a (BPEL) process flow on the left and two examples of compensation functions on the right. The top one, labeled "Global process fault handler," is for handling of general, usually unexpected faults. The lower one, labeled "Compensation handler to undo service that is not transactional," is specific to a particular situation, that is, where a non-transactional update has been performed.

Figure 6. Compensation of transactional and non-transactional BPEL activities



Failed events management

There are some kinds of process failures that neither transactions nor compensation can help out with. For example:

- A target service is not available or is failing.
- The process is not capable of handling the business data; for example, a field is unexpectedly out of bounds or is not present at all. This can often impact decision nodes or endpoint services. This situation often happens when a service is updated without the process being aware of it.
- The target for a JMS or WebSphere MQ message is not available. This can occur on inbound messages when the target component is unavailable, or on output when the target JMS destination is unavailable.

Very often, the only available action when these situations occur is to abandon the process and restart it when the endpoint services are available, or with better or more complete data. This is time consuming, repetitive, and expensive. However, it's possible to configure a process to be suspended, rather than terminated, and to issue a failed event in situations like these. The failed event can then be manually processed. Business Process Manager Advanced provides a failed event manager to enable these events to be processed. Depending on the specific event type, events can be deleted, resubmitted or modified, or an internal trace can be activated for further diagnosis.

The failed event manager has a feature called Store and Forward. Using this feature, if one of the components in the SCA assembly model is accessed asynchronously (by some form of messaging), and that component becomes unavailable, a failed event is generated and all subsequent messages destined for that unavailable component are stored inside the process engine database until the target component is available again. Once the component is available the stored messages are then forwarded to the component. The benefit of this is that you don't raise multiple failed events when a component becomes unavailable.

Event sequencing

There are many scenarios in which processing of information in strict order is of paramount importance to the integrity of the environment. For example:

- An accountCreation event is created and sent for processing. This is followed by an accountCredit event. Clearly, the accountCredit event cannot be processed before the accountCreation event. If this does happen, then at best there will be a confusing "unknown account" message and, at worst, a lost credit.
- An erroneous addressUpdate event is created and is followed by a corrected addressUpdate event. If these are processed out of order, then the erroneous event will be the final version.
- Your account has 10 Euros in it. You deposit a check for 100 Euros into your account. You then withdraw 50 Euros. If the withdrawal transaction is processed first, you would get an "insufficient funds" error (and possibly a large fee from your bank).

In order to accommodate challenges like this, Business Process Manager Advanced provides event sequencing. This enables events for the same key (accountNumber, customerId, and so on) to be sequenced, while other activities can be processed in the order of their arrival.

In order to provide this capability, it's necessary to verify that there is no overlapping work already in the system, so this capability is computationally intense. Because of this, it is recommended that you use it sparingly.

Endpoint retry and reselection

One of the core capabilities of Business Process Manager Advanced is the orchestration of endpoint services. One of the challenges with this orchestration is taking care of situations in which an endpoint is unavailable or in which there is a choice of endpoints depending upon cost, speed, capability, and so on. Through the underlying WebSphere ESB component, Business Process Manager Advanced has the ability to dynamically select or reselect an endpoint. This is generally provided in concert with some form of service registry and repository, such as WebSphere Service

Registry and Repository, that has the list of alternate services. The ESB can access this list and then automatically, based on configuration, retry failed service calls, select alternate service endpoints, or choose from a set of service endpoints based on some classification of the set of service endpoints.

This capability prevents needless failure scenarios when services are not available and enables a single ESB mediation to call different services depending on circumstances.

Tools support for SOA integration with IBM Integration Designer

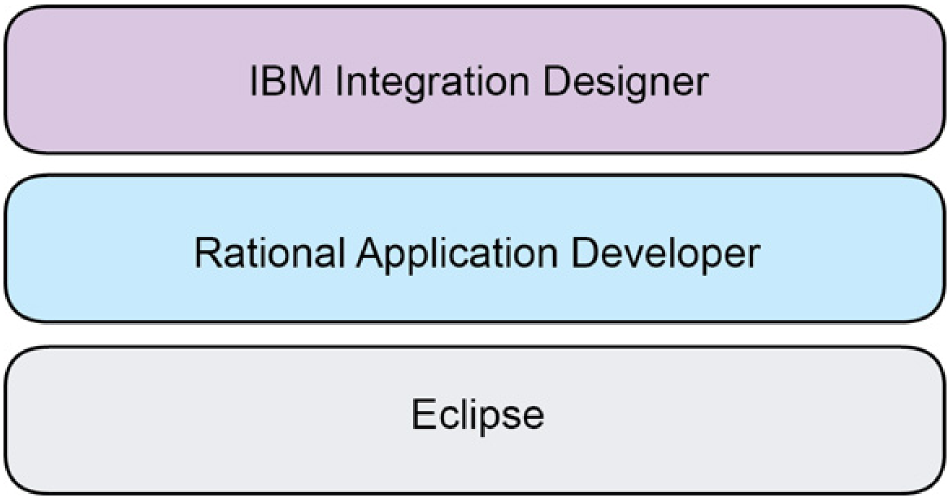
Above and beyond the functional capabilities provided by Business Process Manager Advanced, Business Process Manager Advanced also provides graphical tooling that makes building solutions straightforward for the IT developer. One of the key points about developing solutions in a complex, interconnected IT environment is that solution development needs to be made as straightforward as it can be. IBM Integration Designer provides graphical tools support for all of the capabilities described in previous sections, including all of those provided by IBM Business Process Manager Advanced, WebSphere Adapters (and Adapter Toolkit) and WebSphere ESB. It also supports WebSphere DataPower, which is not discussed in this article. Integration Designer is supported on all Windows® operating systems and some Linux® operating systems, providing a choice of target platform.

Integration Designer has basic features, which are described in the following sections: Eclipse integration, visual tooling, refactoring, testing, and integration.

Eclipse integration

Integration Designer is based on Eclipse and a subset of Rational® Application Developer, as shown in Figure 7.

Figure 7. Integration Developer's foundations: Rational Application Developer and Eclipse



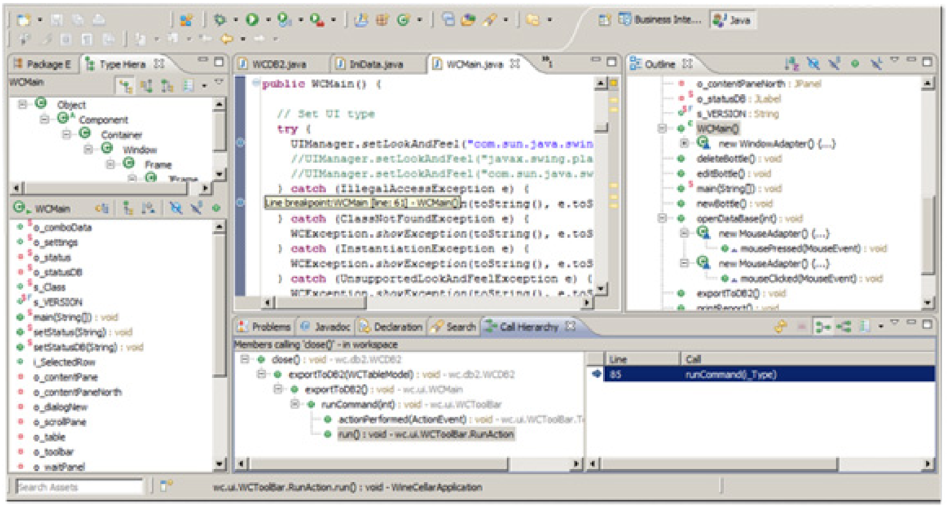
Eclipse is familiar to the J2EE development community and a widely adopted IDE. The Eclipse plug-in architecture enables Integration Designer to leverage third-party tool plug-ins. For example, Eclipse plug-ins are available for most key source code management software, such as CVS and IBM Rational Team Concert.

Visual tooling

Integration Designer inherits graphical tools from Eclipse and Rational Application Developer, and provides specific tools for advanced Business Process Manager requirements, such as SCA and BPEL. Each visual tool is adapted to the particular requirements of the development challenge being addressed, as shown in the Figures 8 and 9,

Figure 8 shows the Java development environment inherited from Eclipse.

Figure 8. Eclipse visual Java development tools

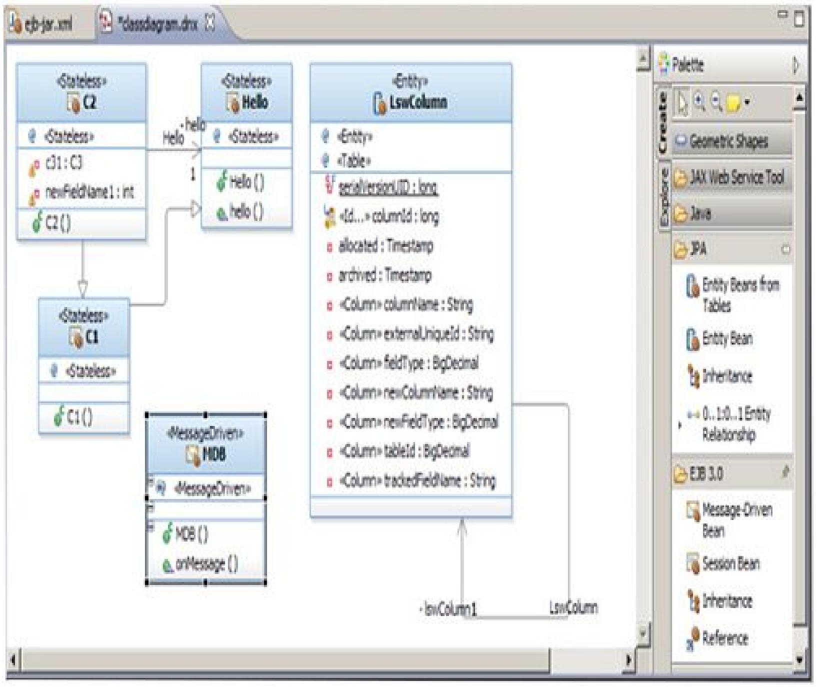


```
public WCMain() {

  // Set UI type
  try {
    UIManager.setLookAndFeel("com.sun.java.swin
    //UIManager.setLookAndFeel("javax.swing.pla
    //UIManager.setLookAndFeel("com.sun.java.sw
  } catch (IllegalAccessException e) {
Line breakpoint:WCMain [line: 61] - WCMain()n(toString(), e.toS
  } catch (ClassNotFoundException e) {
    WCException.showException(toString(), e.toS
  } catch (InstantiationException e) {
    WCException.showException(toString(), e.toS
  } catch (UnsupportedLookAndFeelException e) {
    WCException.showException(toString(), e.toS
```

Figure 9 shows the EJB development environment inherited from Rational Application Developer.

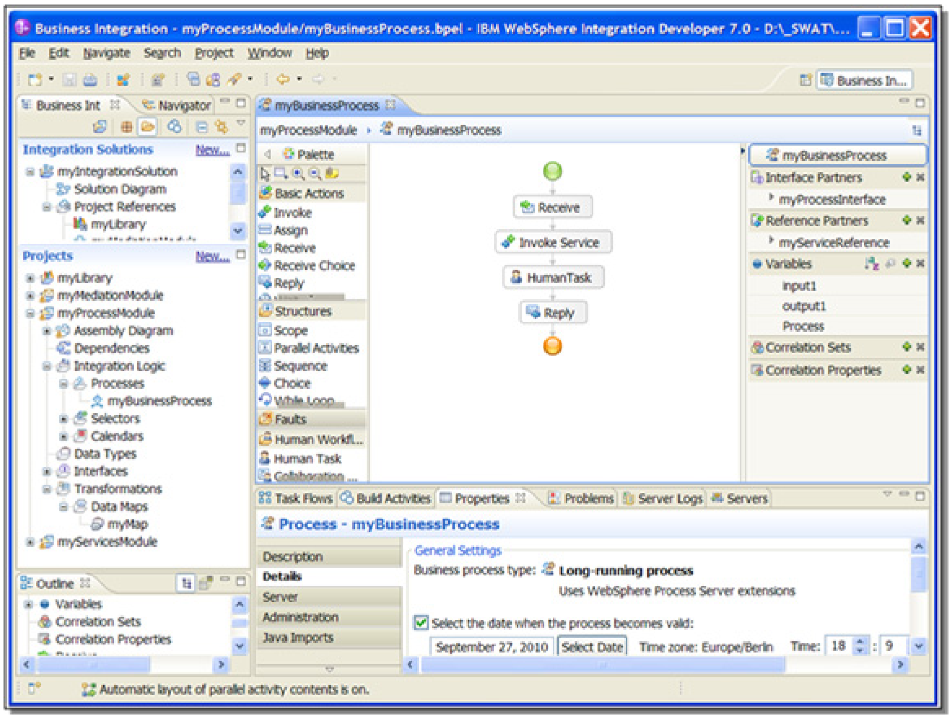
Figure 9. Rational Application Developer visual EJB development tools



Some of the Rational Application Developer tools available in Integration Designer are: XML tools, J2EE tools, web services tools, web development tools, and many others.

Integration Designer provides visual editors for all of the SCA component types mentioned previously. Figure 10 and 11 show examples of these editors.

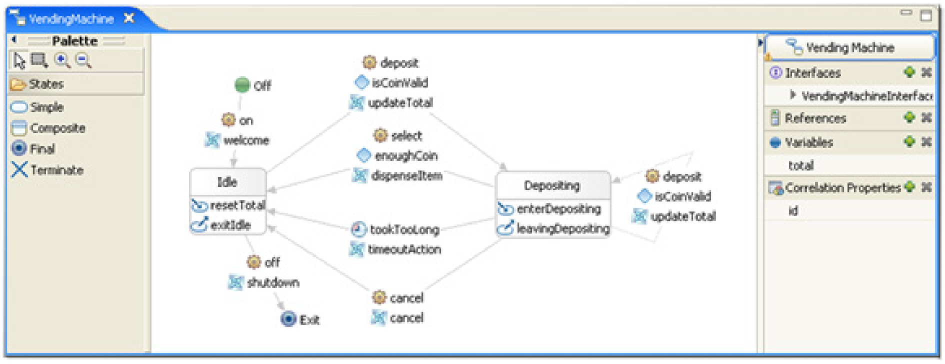
Figure 10. Visual BPEL editor



General Settings
Business process type: Long-running process
Uses WebSphere Process Server extensions
Select the date when the process becomes valid:
September 27, 2010 Select Date Time zone: Europe/Berlin Time: 18 : 9

Automatic layout of parallel activity contents is on.

Figure 11. Visual editor for state machine component implementation

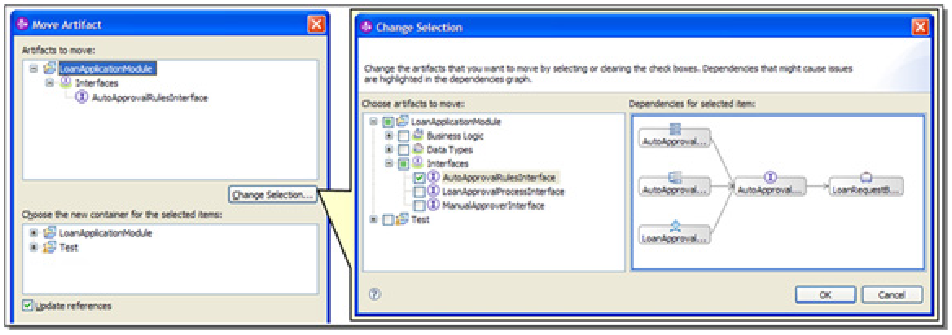


For all of these component types, Integration Designer provides a highly productive, codeless, user-friendly, visual, drag-and-drop development environment that supports the SCA and J2EE programming models underlying the Business Process Manager Advanced capabilities discussed earlier.

Refactoring support

It's a fact of life that all projects change. They do this as they grow into a production implementation and when subsequent rounds of modifications take place. Changes to projects can involve restructuring of business objects, modification of components, and so on. If a business object or component is reused in multiple other components, a simple change can quickly become difficult and error prone. The ability to manage these sorts of changes is known as refactoring, and Integration Designer provides rich refactoring and search facilities to facilitate frequent code and design changes, as shown in Figure 12.

Figure 12. Rich refactoring tools enable rapid and accurate code changes

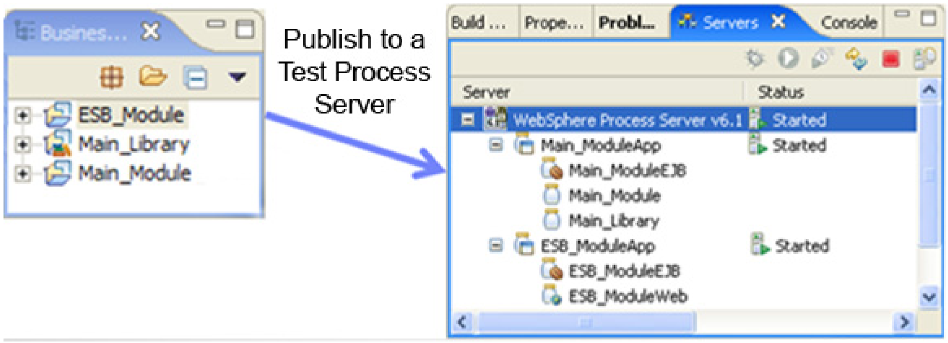


Change the artifacts that you want to move by selecting or clearing the check boxes. Dependencies that might cause issues are highlighted in the dependencies graph.

Component testing

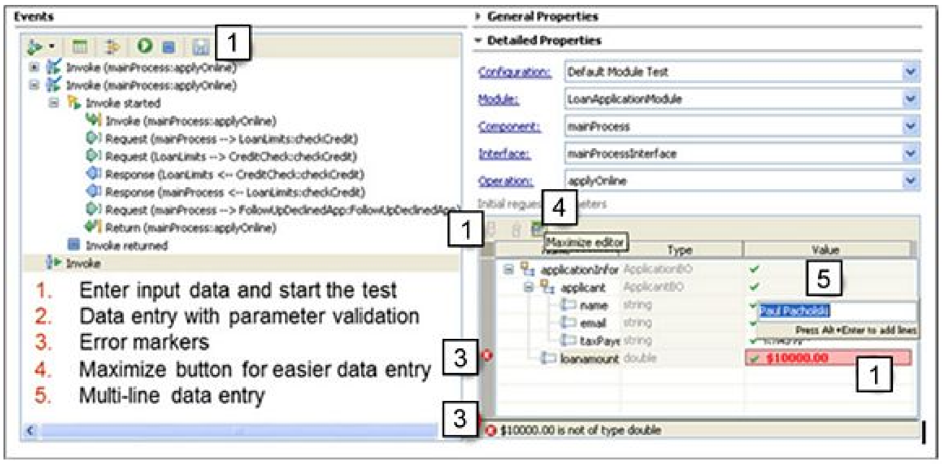
A Process Server instance is built into Integration Designer to facilitate testing of components. Integration Designer provides the capability to test at various levels from individual components to entire applications. The J2EE artifacts generated by the visual editors can be deployed to the test Process Server for testing and debugging, as shown in Figure 13.

Figure 13. Generated J2EE artifacts published to test Process Server



Once the generated code is deployed to the test Process Server, integration developers can use the graphical test client to unit test their integrations. Connected components that are not yet integrated can be scaffolded so that their absence does not affect the component under test. The test client displays the component invocation tree along with the input and output data at each execution step, as shown in Figure 14.

Figure 14. Component test view



Events

- Invoke (mainProcess:applyOnline)
- Invoke (mainProcess:applyOnline)
  - Invoke started
    - Invoke (mainProcess:applyOnline)
    - Request (mainProcess --> LoanLimits:checkCredit)
    - Request (LoanLimits --> CreditCheck:checkCredit)
    - Response (LoanLimits <-- CreditCheck:checkCredit)
    - Response (mainProcess <-- LoanLimits:checkCredit)
    - Request (mainProcess --> FollowUpDeclinedApp:FollowUpDeclinedApp)
    - Return (mainProcess:applyOnline)
  - Invoke returned
- Invoke

1. Enter input data and start the test
2. Data entry with parameter validation
3. Error markers
4. Maximize button for easier data entry
5. Multi-line data entry

General Properties

Detailed Properties

Configuration: Default Module Test
Module: LoanApplicationModule
Component: mainProcess
Interface: mainProcessInterface
Operation: applyOnline

Initial request parameters

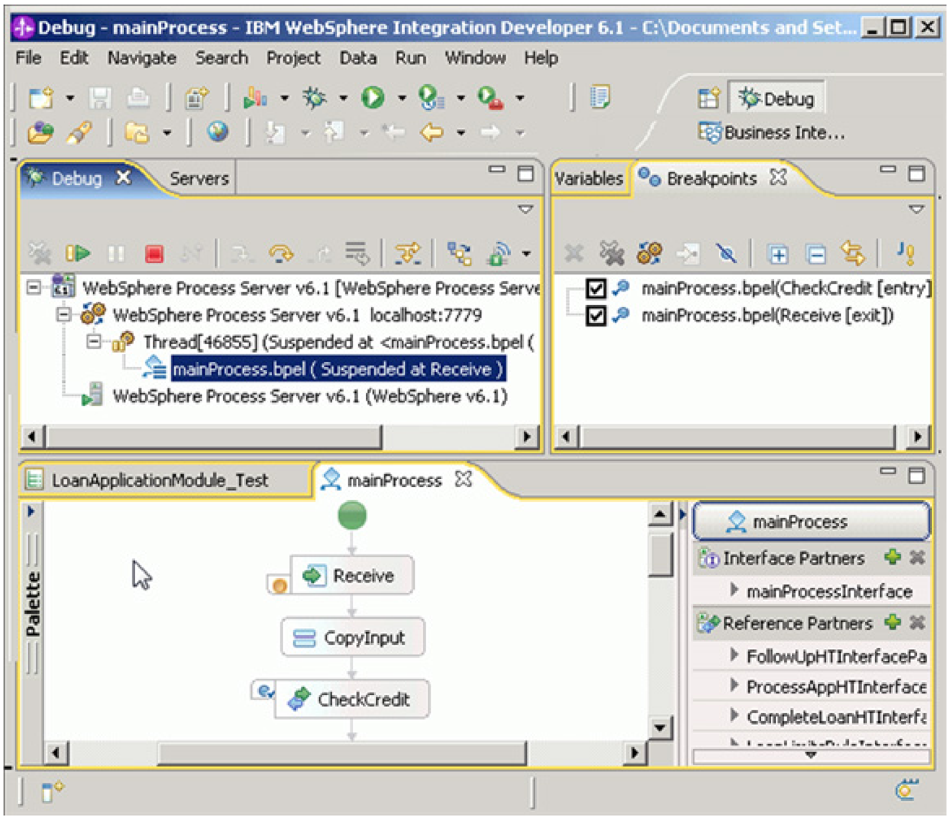
Name	Type	Value
applicationInfor	ApplicationBO	✓
applicant	ApplicantBO	✓
name	string	✓ Paul Pacholski
email	string	✓
taxPaye	string	✓
loanamount	double	✓ $10000.00

Press Alt+Enter to add lines

$10000.00 is not of type double

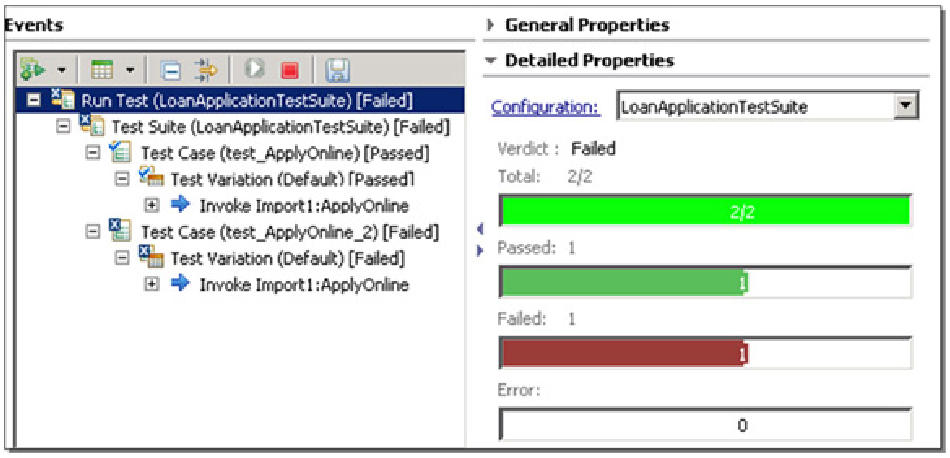
All visual component editors in Integration Designer provide component debuggers. You can set breakpoints for the various artifacts and then use the provided Debug perspective to step through the code, examine and change variables, and resolve issues, as shown in Figure 15.

Figure 15. BPEL editor in the Debug perspective


Debug - mainProcess - IBM WebSphere Integration Developer 6.1 - C:\Documents and Set...
File Edit Navigate Search Project Data Run Window Help
Debug
Business Inte...
Debug
Servers
Variables
Breakpoints
WebSphere Process Server v6.1 [WebSphere Process Serve
WebSphere Process Server v6.1 localhost:7779
Thread[46855] (Suspended at <mainProcess.bpel (
mainProcess.bpel (Suspended at Receive)
WebSphere Process Server v6.1 (WebSphere v6.1)
mainProcess.bpel(CheckCredit [entry]
mainProcess.bpel(Receive [exit])
LoanApplicationModule_Test
mainProcess
Palette
Receive
CopyInput
CheckCredit
mainProcess
Interface Partners
mainProcessInterface
Reference Partners
FollowUpHTInterfacePa
ProcessAppHTInterface
CompleteLoanHTInterfa

Individual test cases can be grouped together in test suites. The test suite editor is used to create both test cases and test suites. When the expected results don't match the output data at runtime, a test case is marked as failed. Test suites can be run either in Integration Designer, from Ant scripts, or from scripts that are a part of server deploy tasks.

Figure 16. Test suites containing test cases automate component testing


Events
Run Test (LoanApplicationTestSuite) [Failed]
Test Suite (LoanApplicationTestSuite) [Failed]
Test Case (test_ApplyOnline) [Passed]
Test Variation (Default) [Passed]
Invoke Import1:ApplyOnline
Test Case (test_ApplyOnline_2) [Failed]
Test Variation (Default) [Failed]
Invoke Import1:ApplyOnline
General Properties
Detailed Properties
Configuration:
LoanApplicationTestSuite
Verdict : Failed
Total: 2/2
2/2
Passed: 1
1
Failed: 1
1
Error:
0

Integration with IBM Process Center and Process Designer

The focus of this article has been the features of IBM Business Process Manager Advanced and the capabilities it provides above and beyond IBM Business Process Manager Standard. However, the two environments need to cooperate and interoperate. All of the Business Process Manager Advanced artifacts are developed using IBM Integration Designer, as described in the previous section. This is an Eclipse-based tool that utilizes local workspaces, unlike IBM Process Designer. At appropriate points (at the discretion of the Integration Designer user), the Integration Designer workspace can be synchronized with the Process Center and all artifacts then become available to Process Designer.

This capability is provided via the Process Center component of the system. Process Center provides the common artifact repository and "join point" for the two environments. This enables artifacts to be brought together and also enables artifacts (and particularly their interfaces) to be shared between Integration Designer and Process Designer. Integration Designer developers can browse for process applications, open them in their workspace and then implement any required process integration services that were specified, but not implemented, in Process Designer. These services are known as Advanced Integration Services (AIS).

IBM Business Process Manager Standard and an ESB?

One frequent question has to do with the notion of pairing IBM Business Process Manager Standard and an ESB. Would doing so provide all of the required process integration capabilities? Although the ESB provides significant connectivity and integration capabilities, the answer is no. An Enterprise Service Bus provides *application* integration capabilities. This addresses some of the process integration capabilities described earlier, but not all. Depending upon the ESB that is selected (not all ESBs are the same!), the following capabilities are available:

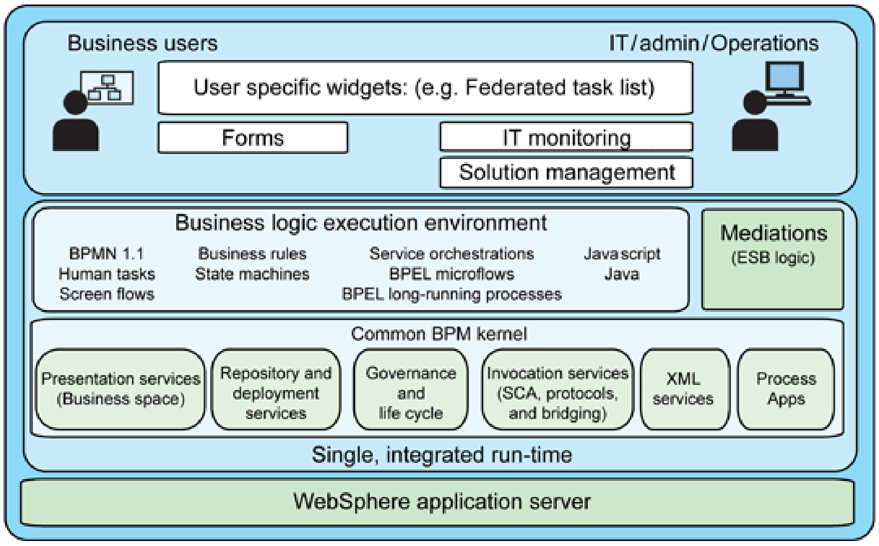
- The basic ESB capabilities mentioned in the Integrated ESB section.
- Some ESBs provide support for adapters.
- ESBs provide different levels of basic transaction support, with the best providing support of distributed transactions.
- Endpoint retry and reselection.

However, there is minimal support for SCA, except with WebSphere ESB, compensation, or failed events. If the full range of process integration capabilities is required, Business Process Manager Advanced is the way to go.

Conclusion

Finally, we will take a look at how these capabilities are implemented in IBM Business Process Manager. Figure 17 shows the primary capabilities of IBM Business Process Manager V7.5.

Figure 17. IBM Business Process Manager capabilities


Business users
IT/admin/Operations
User specific widgets: (e.g. Federated task list)
Forms
IT monitoring
Solution management
Business logic execution environment
BPMN 1.1
Human tasks
Screen flows
Business rules
State machines
Service orchestrations
BPEL microflows
BPEL long-running processes
Javascript
Java
Mediations
(ESB logic)
Common BPM kernel
Presentation services (Business space)
Repository and deployment services
Governance and life cycle
Invocation services (SCA, protocols, and bridging)
XML services
Process Apps
Single, integrated run-time
WebSphere application server

As shown in Figure 17, Business Process Manager Advanced capabilities are scattered across mediations, BPEL microflows and BPEL long-running processes. The SCA environment and the underlying WebSphere Application Server underpin all of these, providing supporting functions.

Before IBM Business Process Manager V7.5, these capabilities were delivered in two separate products, and it was possible but sometimes challenging to join the capabilities. That's now a thing of the past, and it's pretty straightforward to make use of the Business Process Manager Advanced capabilities.

Acknowledgments

The authors would like to thank Neil Kolban, Marc Fasbinder, Scott Simmons and Dave Wakeman for their review and contributions to this article.

Related topics

- IBM Business Process Manager 7.5 documentation
- Web service standards supported in Business Process Manager Advanced
- XML constructs not supported in IBM Business Process Manager Advanced
- developerWorks Middleware: Get the latest technical resources on IBM Business Process Manager and other middleware solutions, including downloads, demos, articles, tutorials, events, webcasts, and more.