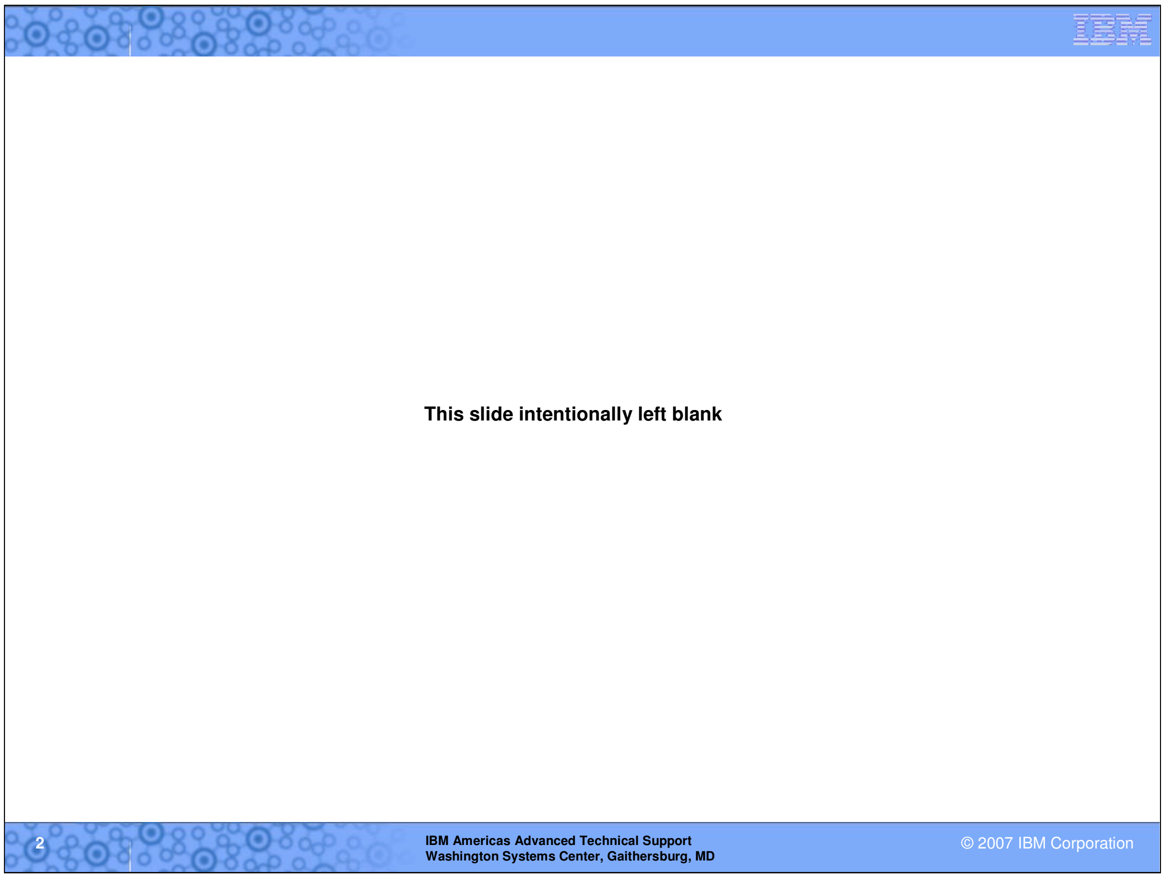
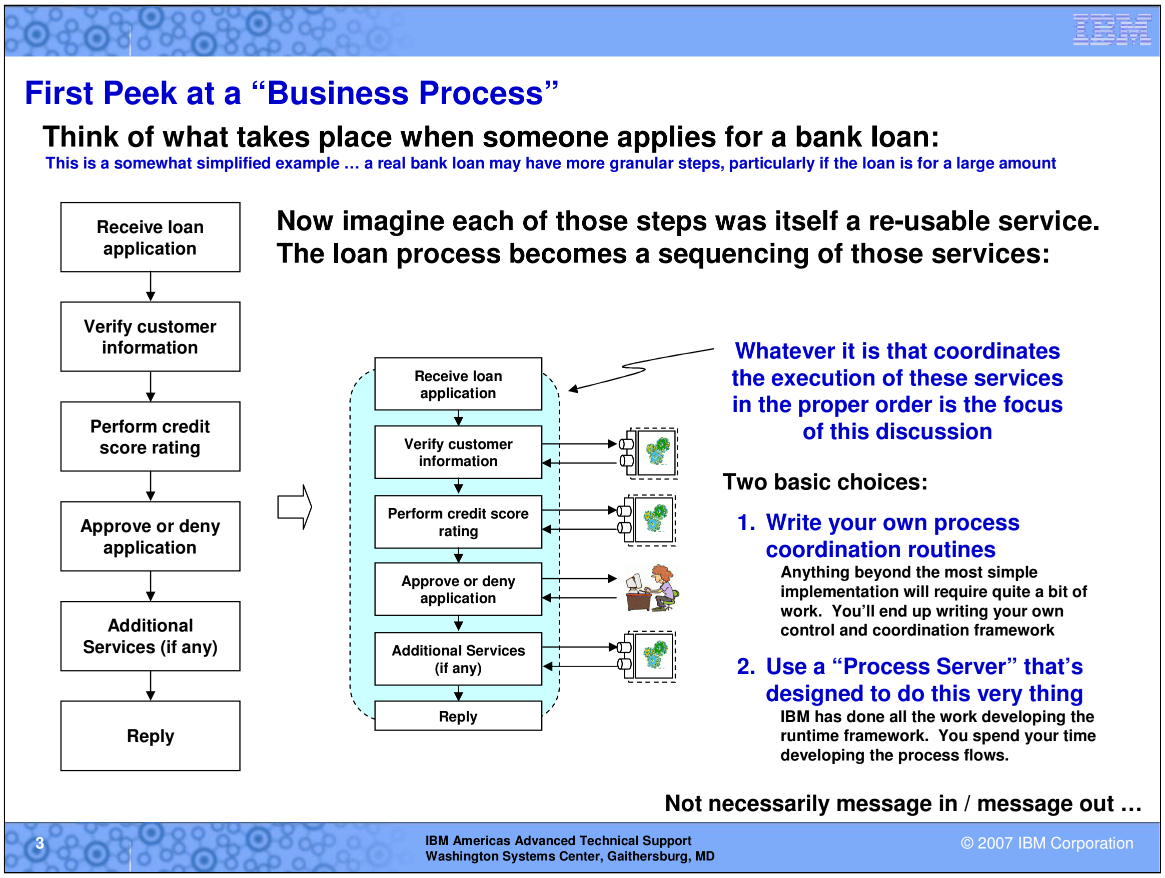
Business Process Management

WebSphere Process Server

Don Bagwell
IBM Washington Systems Center
`dbagwell@us.ibm.com`



This slide intentionally left blank



First Peek at a "Business Process"

Think of what takes place when someone applies for a bank loan:

This is a somewhat simplified example … a real bank loan may have more granular steps, particularly if the loan is for a large amount

Receive loan application → Verify customer information → Perform credit score rating → Approve or deny application → Additional Services (if any) → Reply

Now imagine each of those steps was itself a re-usable service. The loan process becomes a sequencing of those services:

Receive loan application → Verify customer information → Perform credit score rating → Approve or deny application → Additional Services (if any) → Reply

Whatever it is that coordinates the execution of these services in the proper order is the focus of this discussion

Two basic choices:

1. **Write your own process coordination routines**
   Anything beyond the most simple implementation will require quite a bit of work. You'll end up writing your own control and coordination framework
2. **Use a "Process Server" that's designed to do this very thing**
   IBM has done all the work developing the runtime framework. You spend your time developing the process flows.

Not necessarily message in / message out …

3

IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD

© 2007 IBM Corporation

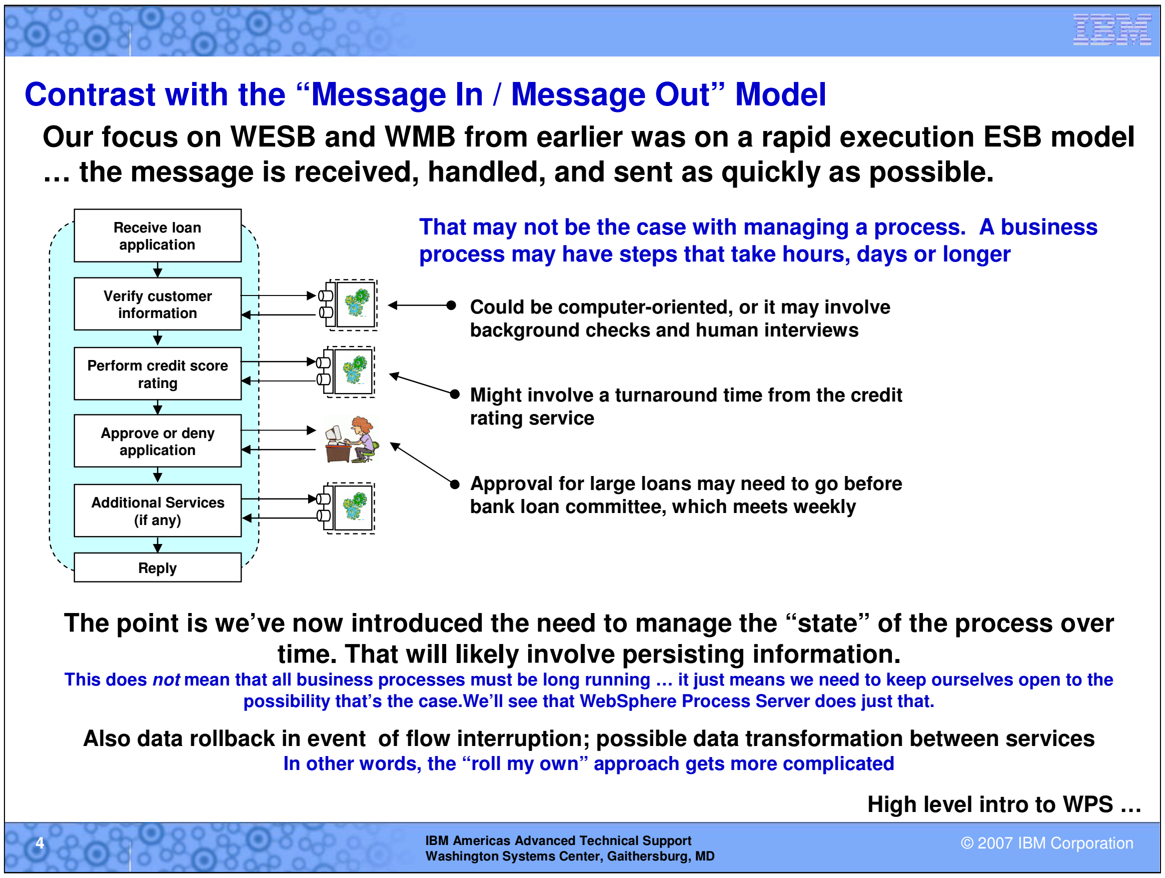
The first thing we'll do is try to define a "business process" and put it in context with all the "services" stuff we've been talking about so far. The term "process" is a bit broad, so rather than offer a definition we'll use an example. Imagine applying for a bank loan. The steps involved might be something like what's shown in the chart -- application received, the information is verified, a credit check is made, a decision is made, perhaps some kind of upsell is offered (a credit card, or something like that), and finally the reply is made.

Now imagine that each one of those steps is itself a kind of reusable service. (And if you can picture that then you've got a good handle on the concept of SOA.) The loan application "process" is then the execution of the steps, from start to finish, resulting in the approval of the loan or the rejection of the application.

So far so good. Now, *something* has to coordinate the execution of that flow. In the old days it would have been a person with a clipboard and a checklist. But in our day we're looking at having that coordination done by a computer-based application of some sort. Given that, we have two basic choices -- write our own coordination and control program, something that invokes the first service and makes sure it's complete, then the second and so forth. Or use someone else's implementation of a "process server" which does all that work, with us having only to program the process server for our loan application's specifics.

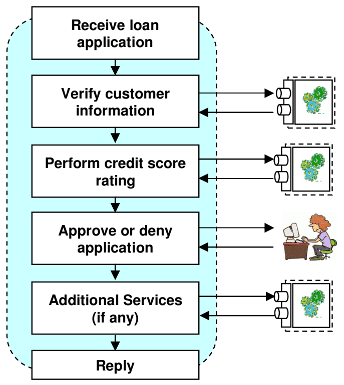
Obviously what we're getting at here is the second one … because that's what WebSphere Process Server is … it's a programming framework that controls the flow of service execution so a bigger "process" is executed in the proper order.

We'd like to now contrast this with the "message in / message out" model we've shown already with WESB and WMB. We do this because it highlights why a "process server" is something more than what the ESB offers.



Contrast with the "Message In / Message Out" Model

Our focus on WESB and WMB from earlier was on a rapid execution ESB model … the message is received, handled, and sent as quickly as possible.


Receive loan application
Verify customer information
Perform credit score rating
Approve or deny application
Additional Services (if any)
Reply

That may not be the case with managing a process. A business process may have steps that take hours, days or longer

- **Could be computer-oriented, or it may involve background checks and human interviews**
- **Might involve a turnaround time from the credit rating service**
- **Approval for large loans may need to go before bank loan committee, which meets weekly**

The point is we've now introduced the need to manage the "state" of the process over time. That will likely involve persisting information.

This does *not* mean that all business processes must be long running … it just means we need to keep ourselves open to the possibility that's the case.We'll see that WebSphere Process Server does just that.

Also data rollback in event of flow interruption; possible data transformation between services

In other words, the "roll my own" approach gets more complicated

High level intro to WPS …

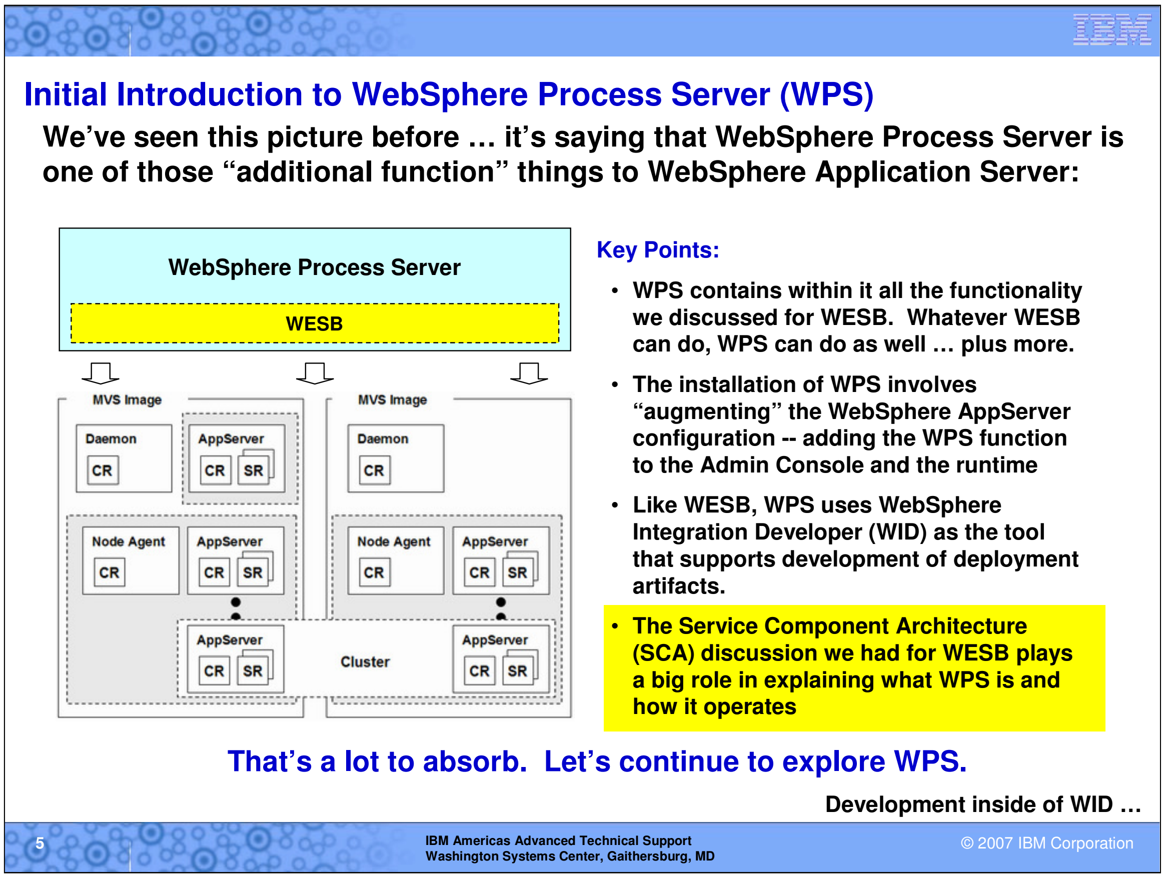
4 IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD © 2007 IBM Corporation

Much of our discussion during the WESB and WMB unit was around a rather rapid "message in / message out" model. The faster the better, frankly, as the ESB is ideally a fast throughput "device" within the SOA environment.

But when we start thinking about coordinating the execution of a string of services, things take on a different color. The key difference is that some of the services that are invoked as part of the process flow may take some time -- from a few seconds to days or perhaps weeks. That is particularly true if the "service" invoked is really something that person does, such as conduct personal interviews or visit with the customer to see with their own eyes what's being claimed in the application. Therefore, we can't necessarily assume the whole process will take seconds or milliseconds.

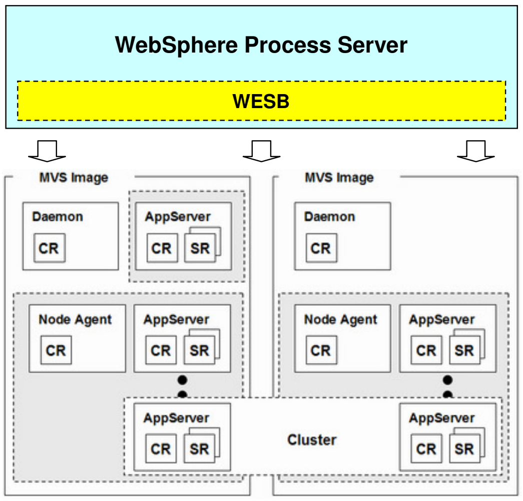
That introduces some complexity to the picture. It means that whatever it is that's coordinating the flow of the service execution in the process needs to be able to manage the "state" of things. And holding things in memory is likely not viable because servers are taken down periodically for things like maintenance. Further, the failure of a service in the middle of the process may mean "rolling back" the updates done in the earlier steps … and in that sense then a "process" takes on a kind of "transaction" feel. The bottom line is that the "roll your own" approach gets more complicated.

What we're leading up to here is, of course, that WebSphere Process Server does all this. That's what it's designed to do.



Initial Introduction to WebSphere Process Server (WPS)

We've seen this picture before … it's saying that WebSphere Process Server is one of those "additional function" things to WebSphere Application Server:



Key Points:

- **WPS contains within it all the functionality we discussed for WESB. Whatever WESB can do, WPS can do as well … plus more.**
- **The installation of WPS involves "augmenting" the WebSphere AppServer configuration -- adding the WPS function to the Admin Console and the runtime**
- **Like WESB, WPS uses WebSphere Integration Developer (WID) as the tool that supports development of deployment artifacts.**
- **The Service Component Architecture (SCA) discussion we had for WESB plays a big role in explaining what WPS is and how it operates**

That's a lot to absorb. Let's continue to explore WPS.

Development inside of WID …

5 IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD © 2007 IBM Corporation

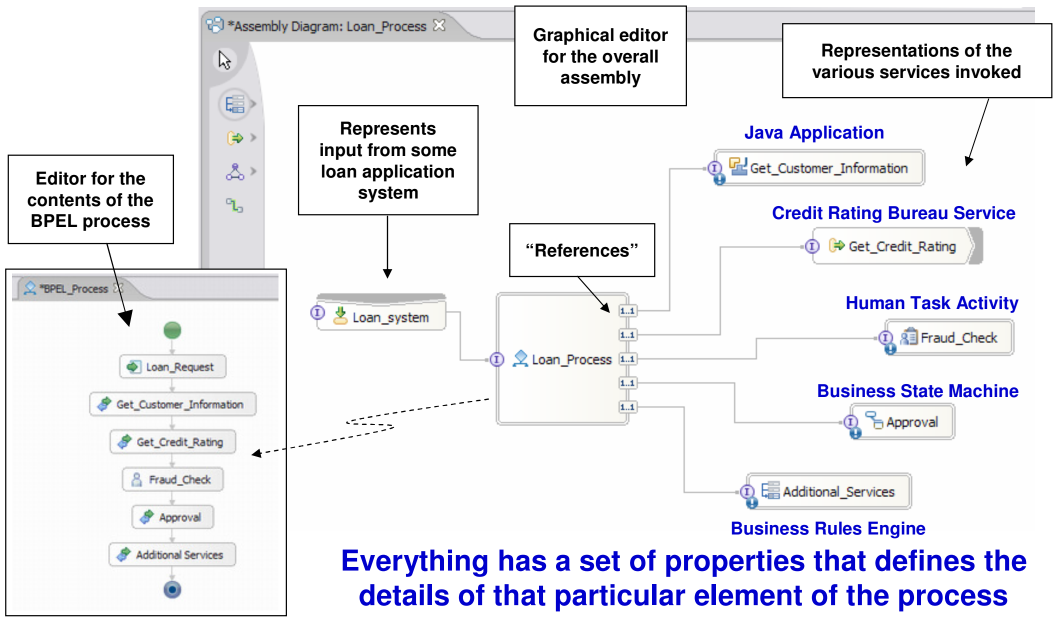
We've seen this picture before … it's a high-level representation of WebSphere Process Server. WPS is function added to an existing WebSphere Application Server configuration. In this sense, it's just like what we saw for WESB. (And it's like what we're seeing more of -- WAS being used as a "foundation" for extended functionality.) This chart is making four key points:

- It's not necessarily obvious, but it is important … WebSphere Process Server comes with the WESB product packaged within it. WPS relies upon the existence of WESB under the covers. The important thing here is that all of the capabilities of WESB we discussed ealier are present in WPS, and more.
- The installation process -- we'll cover it a bit more later -- really involves running some batch jobs and shell scripts to update the existing configuration so it knows about WPS and can use WPS. That involves the creation of links to the WPS code, as well as the "augmentation" (updating) of the configuration profile so it has the WPS function represented on the Admin Console.
- Development of the deployable artifacts for WPS is done using the WebSphere Integration Developer (WID) tool. That's just like WESB required. Again, we're making the point that WPS is really a super-set of WESB. And the tooling is the same.
- Finally, all of the discussion we had about Service Component Architecture (SCA) and Service Data Objects (SDO) comes into play with WPS as well. That makes sense -- WPS is a superset of WESB and WPS includes WESB inside of itself. The development of a WPS process is really the development of a set of SCA components that you string together to form the definition of the process. That's what runs inside of WPS. WPS has the SCA runtime support as part of its implementation.

At this point the picture may not be all that clear. It is a lot to absorb. So let's continue, slowly, and build on all this.

An Example of a Process Inside of WID

Purpose of this is to give you a sense for the creation of a process flow:



Everything has a set of properties that defines the details of that particular element of the process

Putting it into perspective …

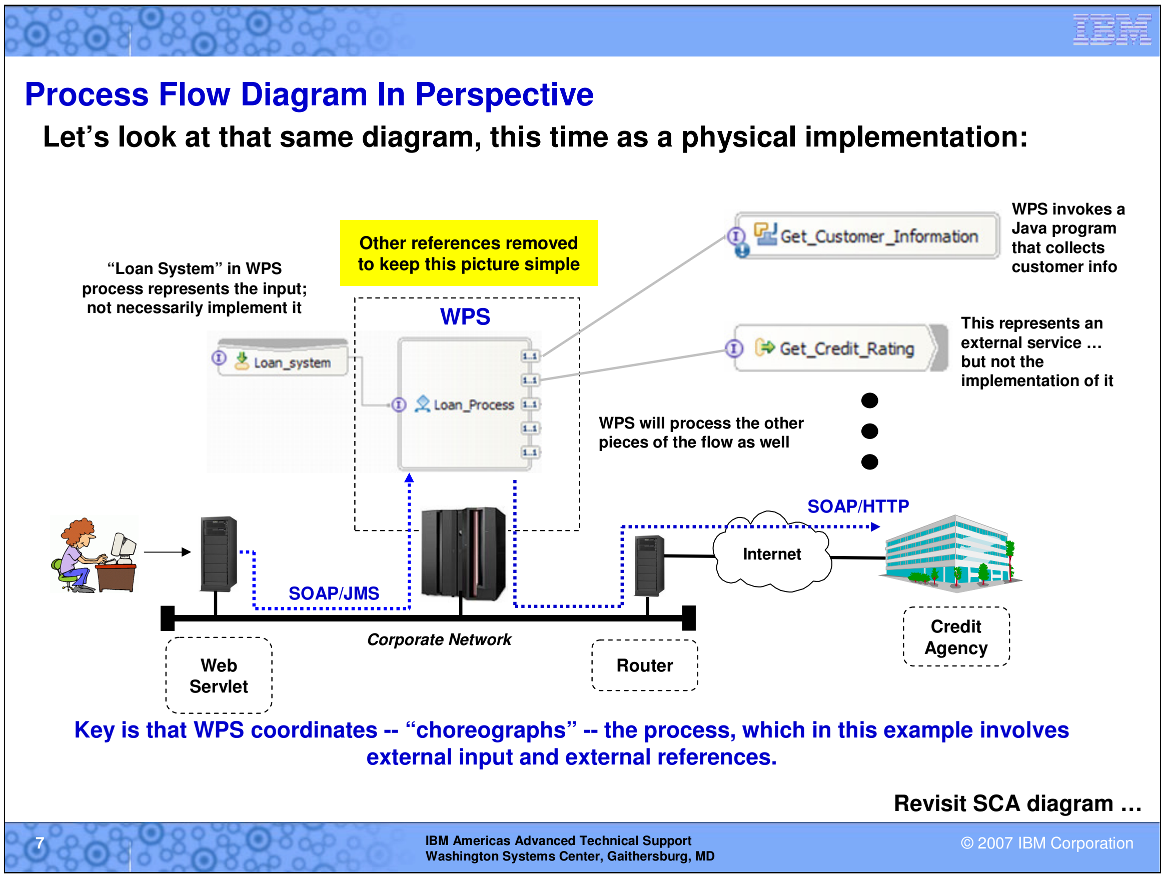
6 IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD
© 2007 IBM Corporation

At this point we need to wade into the topic of developing the process programs that run inside of WPS. And the topic can get deep really fast, particularly for those who do not come from a graphical programming background. So the purpose of this initial chart is to give you a *sense* for what's involve. It's nowhere near a complete picture.

Take note of the following:

- This is primarily a graphical development environment. The purpose of this is to represent the process flow as a graphical picture, and from that generate the code and deployment artifacts. Each of the elements in the picture has a rich set of properties that defines precisely what it does.
- The lines that connect the graphical elements are "wires" … that's the same thing as we discussed for the SCA model. That's because this is SCA.
- The box in the middle is an SCA component who's implementation is that of a "BPEL Process." BPEL stands for "Business Process Execution Language." That means inside of that box there's further graphical detail that tells the story of the flow. A different editor is used to create that, and the items inside the BPEL process are known as "activities." We have a few charts later explaining the various activities available when creating a BPEL process.
- The other things on the chart are representation of different pieces of the BPEL process. One is a Java program that will be invoked by the process; another is s web service call to a credit service provider; a third is a human task. The point is those provide the functional detail behind the flow that's defined inside the BPEL process.
- We see a component that represents the input to the process. This defines what invokes the process flow.

Let's take another crack at putting this into perspective.



Process Flow Diagram In Perspective

Let's look at that same diagram, this time as a physical implementation:

"Loan System" in WPS process represents the input; not necessarily implement it
Other references removed to keep this picture simple
WPS
Loan_system
Loan_Process
Get_Customer_Information
WPS invokes a Java program that collects customer info
Get_Credit_Rating
This represents an external service … but not the implementation of it
WPS will process the other pieces of the flow as well
SOAP/HTTP
SOAP/JMS
Internet
Corporate Network
Web Servlet
Router
Credit Agency

Key is that WPS coordinates -- "choreographs" -- the process, which in this example involves external input and external references.

Revisit SCA diagram …

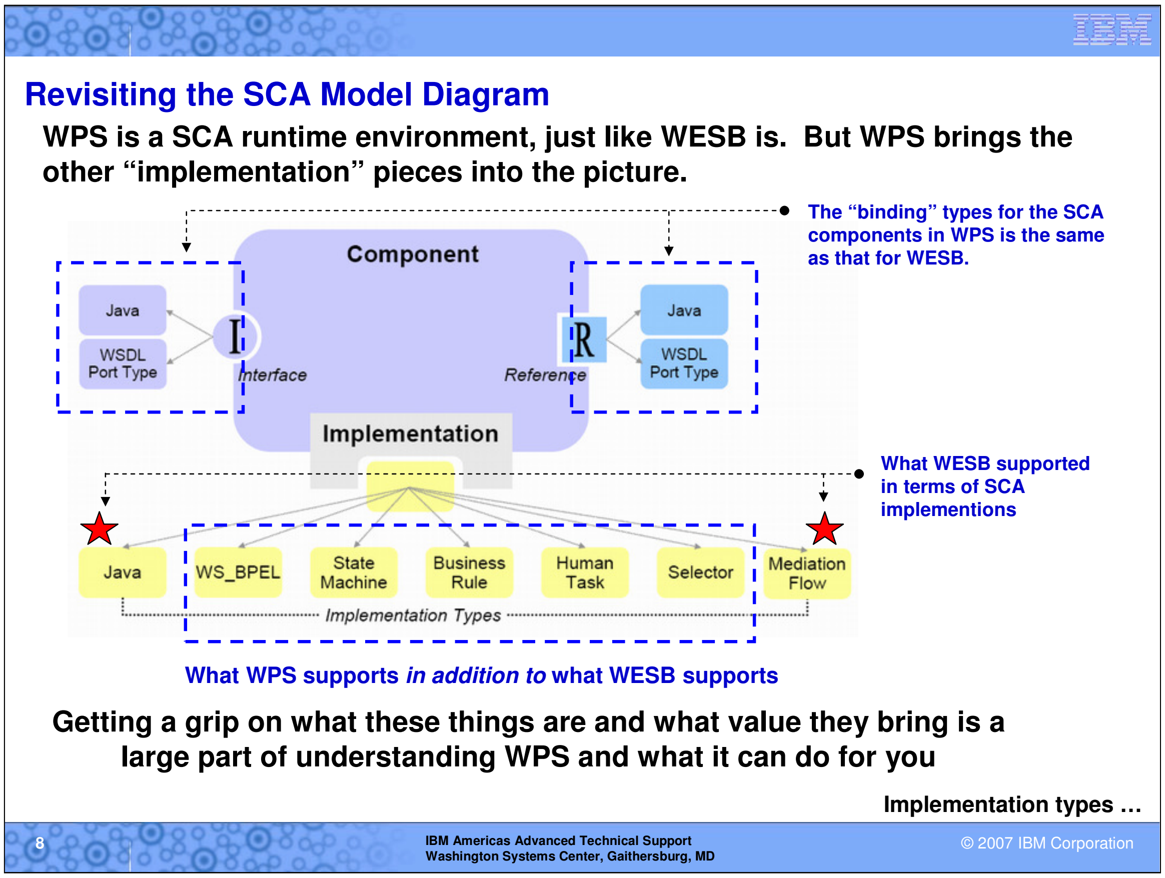
7 IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD © 2007 IBM Corporation

This picture takes a portion of the previous chart's flow and maps it onto a physical layout. We've reduced the number of components from the flow to keep the picture from getting too complicated.

At the heart of this is the BPEL process component. That's what defines the flow of things that gets done as part of of this process flow. In front of that is what's called an "export" -- a funny name given it's an *input* to the process, but that's what it's called -- which represents what initiates the process flow execution. On the right side is the invocation of a Java program to get customer information and the "import" (again, funny name) of a web service to get a credit rating for the customer.

Now let's map that to the physical:

- Imagine the "Loan_system" component was really a representation of a servlet-based system running on some Linux box out front of the z/OS system. Customers connect to that and supply the basic loan application information in the servlet. The servlet then flows the application over JMS to WPS, which is running on z/OS. That invoke the process flow. (By the way, something called a "correlator" is what separates John Smith's application process flow from Mary Johnson's -- think of a unique customer number or something like that.)
- WPS takes over and runs through the process flow as defined in the BPEL component. The first thing is to invoke the Java program to get the customer information. The actual Java program does not necessarily have to be part of the WPS process -- it can be an existing Java program that's now a reusable service. In that case, what's drawn in WID is really just a representation of the Java program with information about how to invoke the program.
- The next thing is invocation of the external web service to get the credit rating. The component drawn in WID to represent this provides information about where that's located and what's involved with invoking it (the WSDL file, either static or dynamically discovered). A SOAP message flows over HTTP and out of the corporate network to the service bureau. Upon return the flow picks up where it left off.



Revisiting the SCA Model Diagram

WPS is a SCA runtime environment, just like WESB is. But WPS brings the other "implementation" pieces into the picture.

Component
Java
WSDL Port Type
Interface
Reference
Java
WSDL Port Type
Implementation
Java
WS_BPEL
State Machine
Business Rule
Human Task
Selector
Mediation Flow
Implementation Types

The "binding" types for the SCA components in WPS is the same as that for WESB.

What WESB supported in terms of SCA implementions

What WPS supports *in addition to* what WESB supports

Getting a grip on what these things are and what value they bring is a large part of understanding WPS and what it can do for you

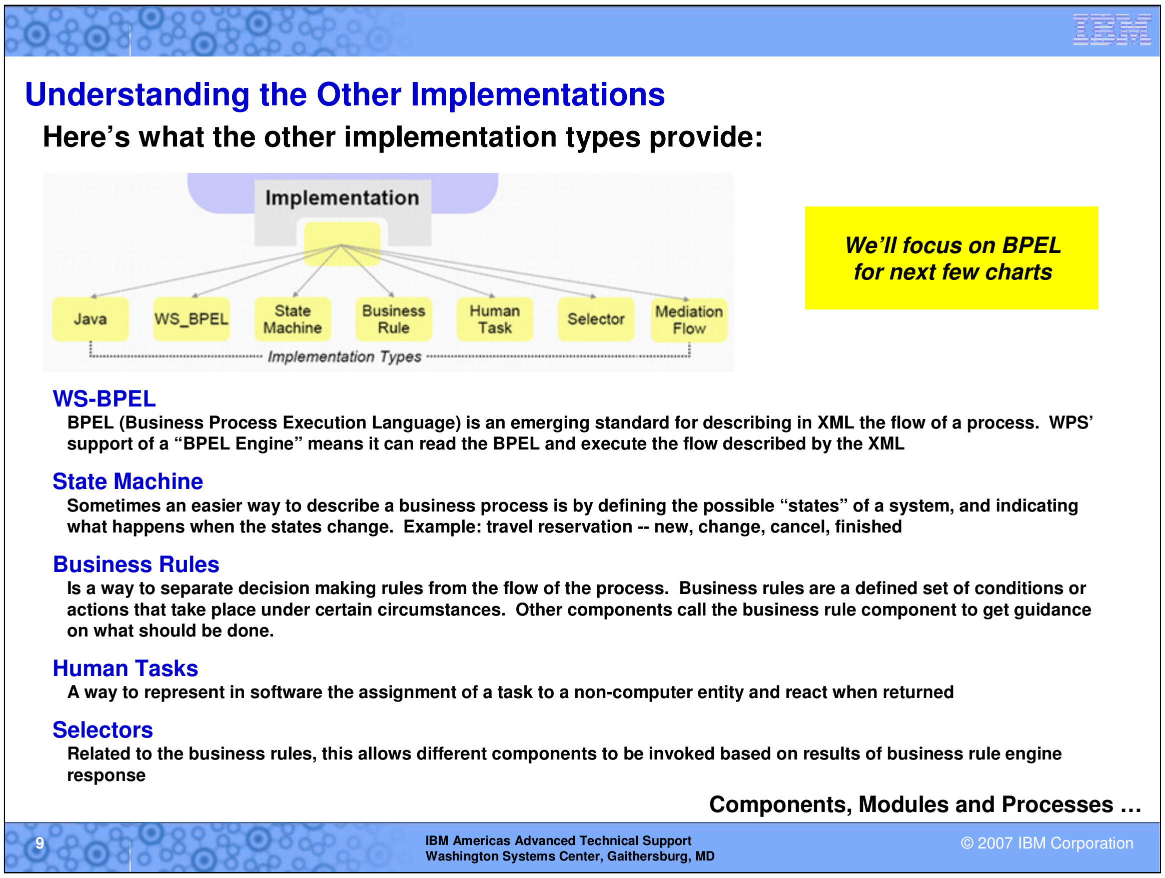
Implementation types …

8

IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD

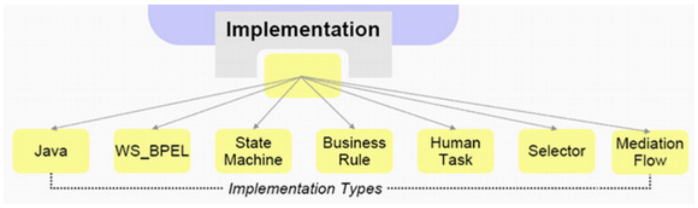
© 2007 IBM Corporation

This is the SCA diagram we saw back when we discussed the WESB product. We're showing it again here because WPS is also a SCA-compatible runtime environment. The same concepts talked about before apply here. What's different is the types of component implementations that WPS supports as compared to what WESB supported. WESB supported only two -- Java and Mediation Flow. WPS supports those and the five others shown bordered by the blue dashed box. In our earlier picture of the WID process we actually saw these things, but we didn't really highlight them much. Now we'll take a slightly closer look.



Understanding the Other Implementations

Here's what the other implementation types provide:



We'll focus on BPEL for next few charts

WS-BPEL

BPEL (Business Process Execution Language) is an emerging standard for describing in XML the flow of a process. WPS' support of a "BPEL Engine" means it can read the BPEL and execute the flow described by the XML

State Machine

Sometimes an easier way to describe a business process is by defining the possible "states" of a system, and indicating what happens when the states change. Example: travel reservation -- new, change, cancel, finished

Business Rules

Is a way to separate decision making rules from the flow of the process. Business rules are a defined set of conditions or actions that take place under certain circumstances. Other components call the business rule component to get guidance on what should be done.

Human Tasks

A way to represent in software the assignment of a task to a non-computer entity and react when returned

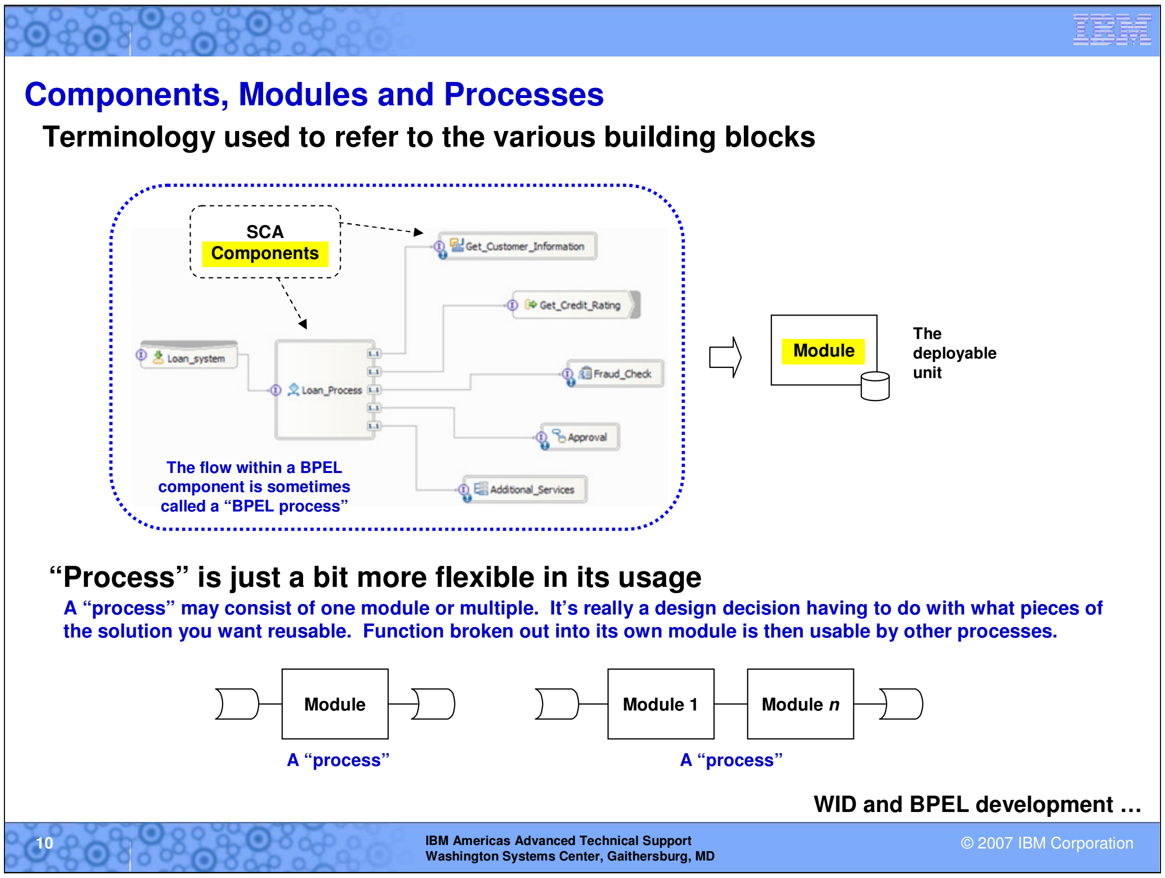
Selectors

Related to the business rules, this allows different components to be invoked based on results of business rule engine response

Components, Modules and Processes …

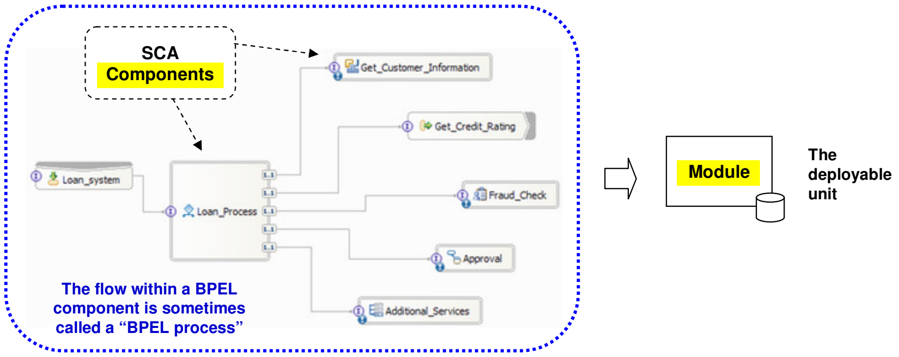
9 IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD © 2007 IBM Corporation

The other SCA component implementation types supported by WPS are described in a bit more detail on this chart.



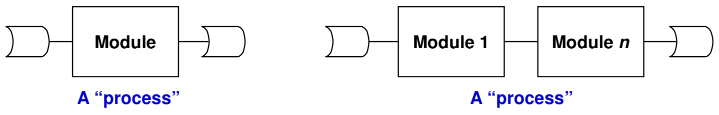
Components, Modules and Processes

Terminology used to refer to the various building blocks



"Process" is just a bit more flexible in its usage

A "process" may consist of one module or multiple. It's really a design decision having to do with what pieces of the solution you want reusable. Function broken out into its own module is then usable by other processes.



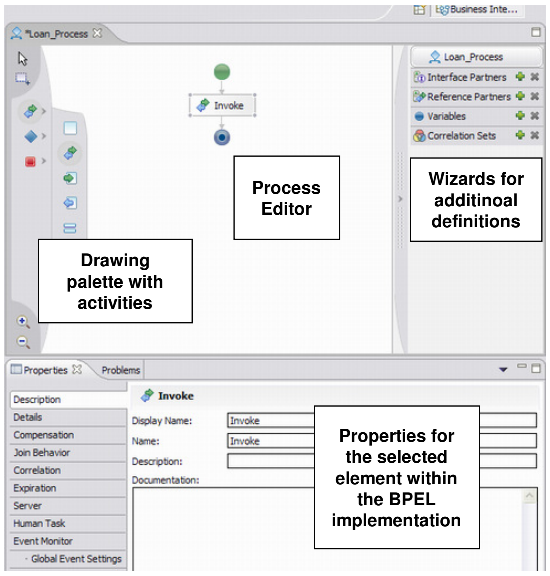
WID and BPEL development …

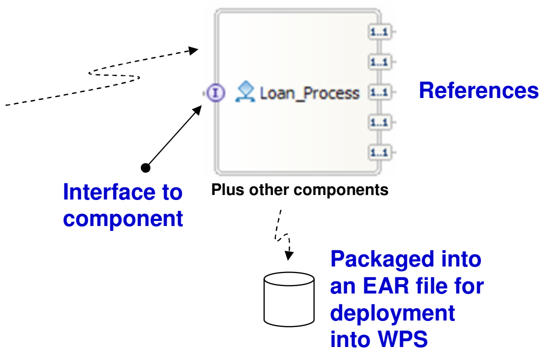
Within this SCA arena we have several pieces of terminology floating around. The key ones we'll focus on here are: "component," "module" and "process."

- **Component** -- this is the basic building block of Service Component Architecture. A component will take the form of one of the implentation types -- Java, BPEL, human task, mediation flow, etc. Most likely anything you build in WID for deployment in WPS will involve more than one component, but it is possible to build a single component process. The example process we provided before had seven components.
- **Module** -- a module is a packaging concept, and it represents the deployable unit into the WPS runtime. A module will have between 1 and n components inside of it. How much or little goes into a module is largely a function of how granular you wish your artifacts to be. In this sense it's a lot like how much function you package into a Java class. You *could* build a monster class that does everything, but then it's not as granular and reusable. Or you can build smaller classes that represent the more reusable boundaries of the function you're coding. The same concept applies with these modules.
- **Process** -- a process is a collection of modules that are stitched together to form the business flow that is to be executed. It may contain one module or many, depending on how you designed your modules.

WID and Creation of BPEL Process

A BPEL implementation is a "component". Inside the component is the flow described by BPEL. The WID "Process Editor" is used to define the flow:





Key Point -- WID has graphical editors and wizards to define the contents of a component as well as the relationship of component to component

Again … there's a learning curve to this

BPEL basic activities …

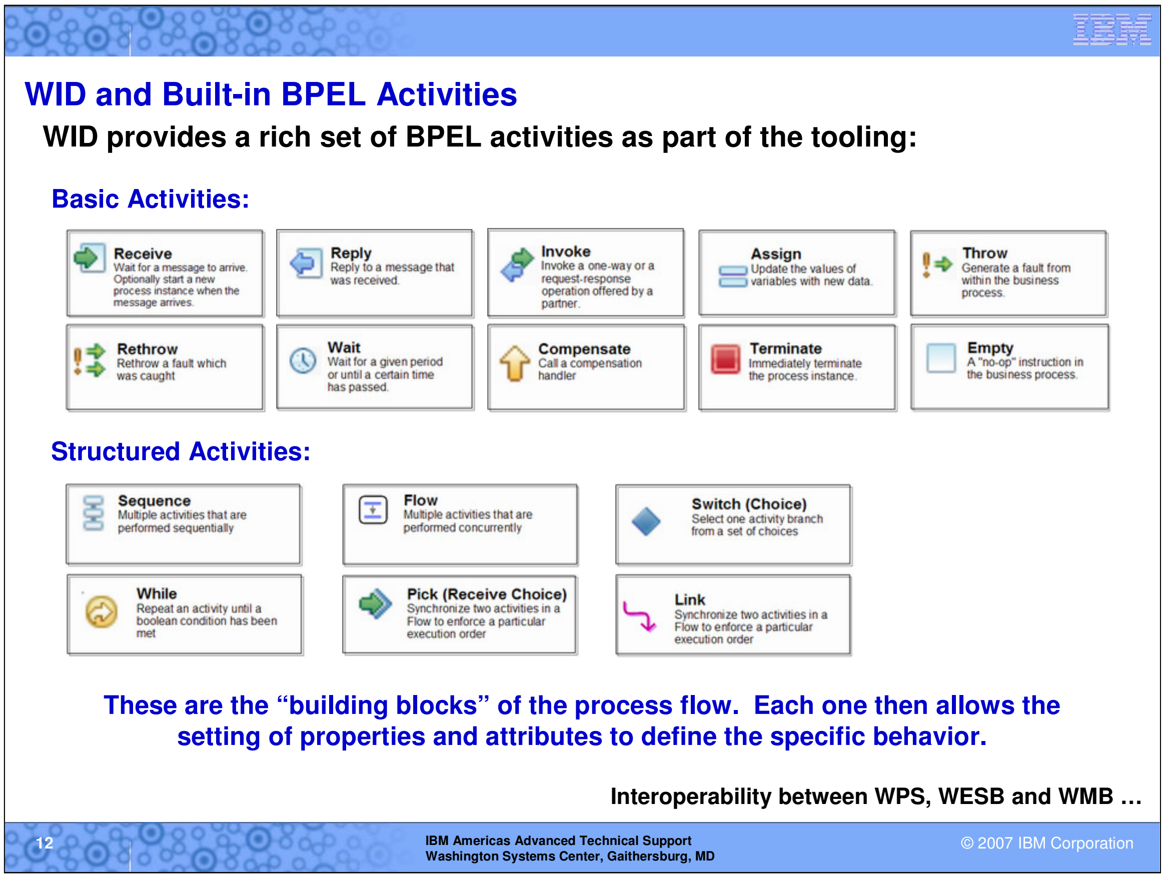
11 IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD © 2007 IBM Corporation

We're going to turn our focus now and concentrate on the construction of the BPEL component. We do that because the BPEL component has within it a kind of flow. Once again, WID provides the graphical environment to develop the internals of the BPEL component, as well as the broader construction of the modules.

The process is like what you've seen with other graphical environments -- a drawing palette provides access to the things you'll use to construct your BPEL flow. Those things are called "activities" and we'll see what activities are provided in WID. To the right is a toolbar with various wizards to do additional definitions, and down below are the properties for any selected activity.

The process is like what we saw in the Broker Toolkit -- draw a picture, set properties, export to an EAR and deploy.

You won't master this in a day … it takes some time to become proficient.



WID and Built-in BPEL Activities

WID provides a rich set of BPEL activities as part of the tooling:

Basic Activities:

Receive Wait for a message to arrive. Optionally start a new process instance when the message arrives.	**Reply** Reply to a message that was received.	**Invoke** Invoke a one-way or a request-response operation offered by a partner.	**Assign** Update the values of variables with new data.	**Throw** Generate a fault from within the business process.
Rethrow Rethrow a fault which was caught	**Wait** Wait for a given period or until a certain time has passed.	**Compensate** Call a compensation handler	**Terminate** Immediately terminate the process instance.	**Empty** A "no-op" instruction in the business process.

Structured Activities:

Sequence Multiple activities that are performed sequentially	**Flow** Multiple activities that are performed concurrently	**Switch (Choice)** Select one activity branch from a set of choices
While Repeat an activity until a boolean condition has been met	**Pick (Receive Choice)** Synchronize two activities in a Flow to enforce a particular execution order	**Link** Synchronize two activities in a Flow to enforce a particular execution order

These are the “building blocks” of the process flow. Each one then allows the setting of properties and attributes to define the specific behavior.

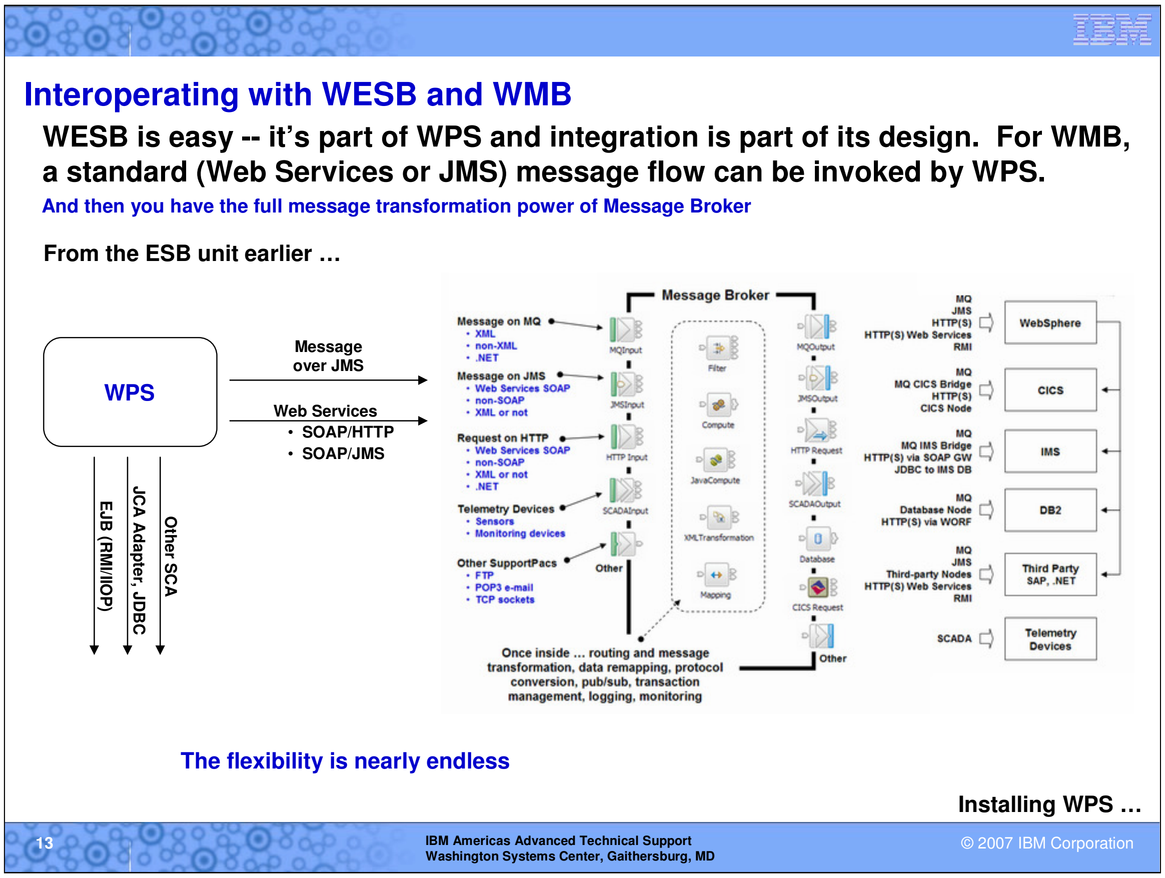
Interoperability between WPS, WESB and WMB …

12

IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD

© 2007 IBM Corporation

WebSphere Integration Developer provides a series of “activities” that allow you to construct your BPEL process definition using provided “building blocks.” This chart shows you the things that are available for your use. The purpose of this chart is not to go through each one, but rather to give you a sense that the tooling does provide built-in function. You draw out your process flow and then set the properties and attributes of each to define the specific behavior you want.

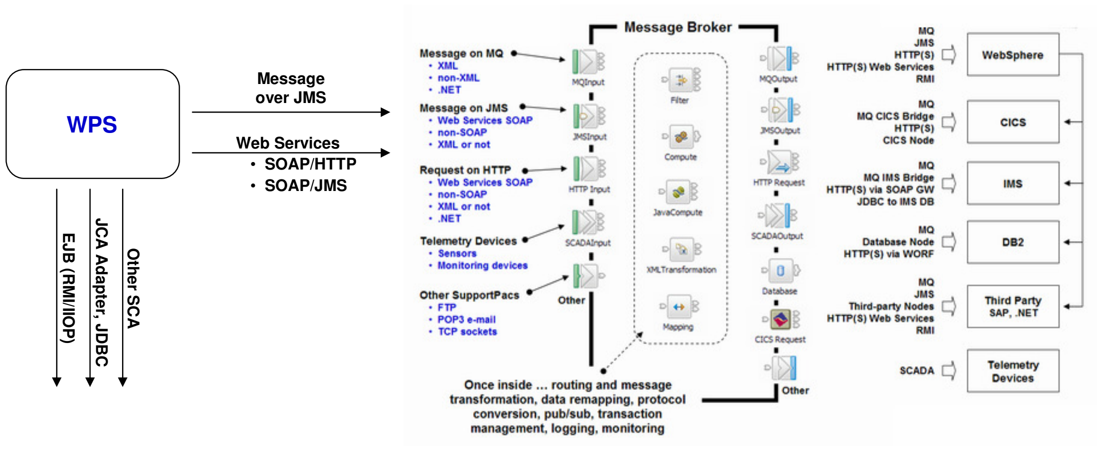


Interoperating with WESB and WMB

WESB is easy -- it's part of WPS and integration is part of its design. For WMB, a standard (Web Services or JMS) message flow can be invoked by WPS.

And then you have the full message transformation power of Message Broker

From the ESB unit earlier …



The flexibility is nearly endless

Installing WPS …

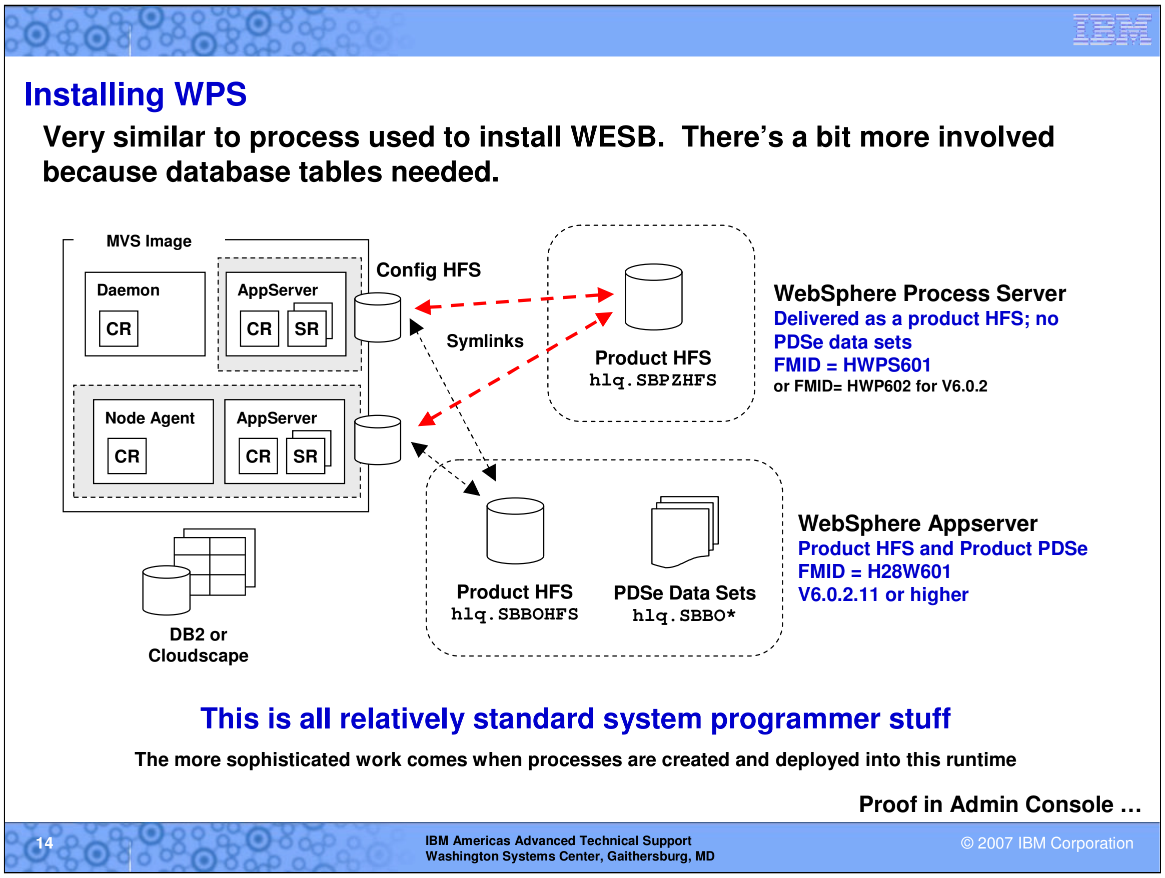
13 IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD
© 2007 IBM Corporation

Rather than go much deeper into the programming aspects of WID, let's now turn our focus back to the broader architectural issues. Specifically, the interoperability of WPS and the ESB products, WESB and WMB.

WESB is relatively simple … WPS is a superset of WESB, and therefore it makes direct use of WESB as part of its functionality. So interoperating with the copy of WESB that's part of the WPS implementation is straight forward. But it can also interoperate with another WESB implementation by using the standard JMS or SOAP interfaces.

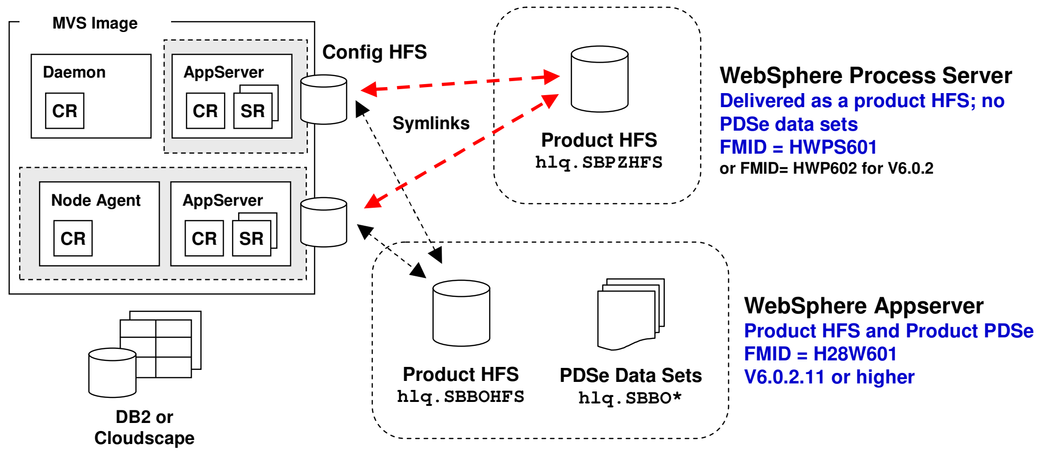
The same holds for WMB. Provided the message flow is defined with a standard interface (JMS or SOAP) then WPS can drive it. (Or "import" the service into the process flow.) When we do that, then the power of WMB comes into the picture and we can get at pretty much any backend system we may desire. WPS has the ability to interface with other services, such as an EJB, a JCA adapter or JDBC, or another SCA module.

You can now get a sense that the options are very flexibile … there are a lot of different ways to structure this, depending on what you're trying to accomplish.



Installing WPS

Very similar to process used to install WESB. There's a bit more involved because database tables needed.



This is all relatively standard system programmer stuff

The more sophisticated work comes when processes are created and deployed into this runtime

Proof in Admin Console …

14 IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD © 2007 IBM Corporation

The installation of WPS is a lot like WESB -- it first involves doing the SMP/E install of the WPS product, which results in an HFS being created and mounted but no PDSe data sets. From there, the process involves running some utilities that create symbolic links in the WebSphere configuration HFS so the new WPS files are accessible to WebSphere.

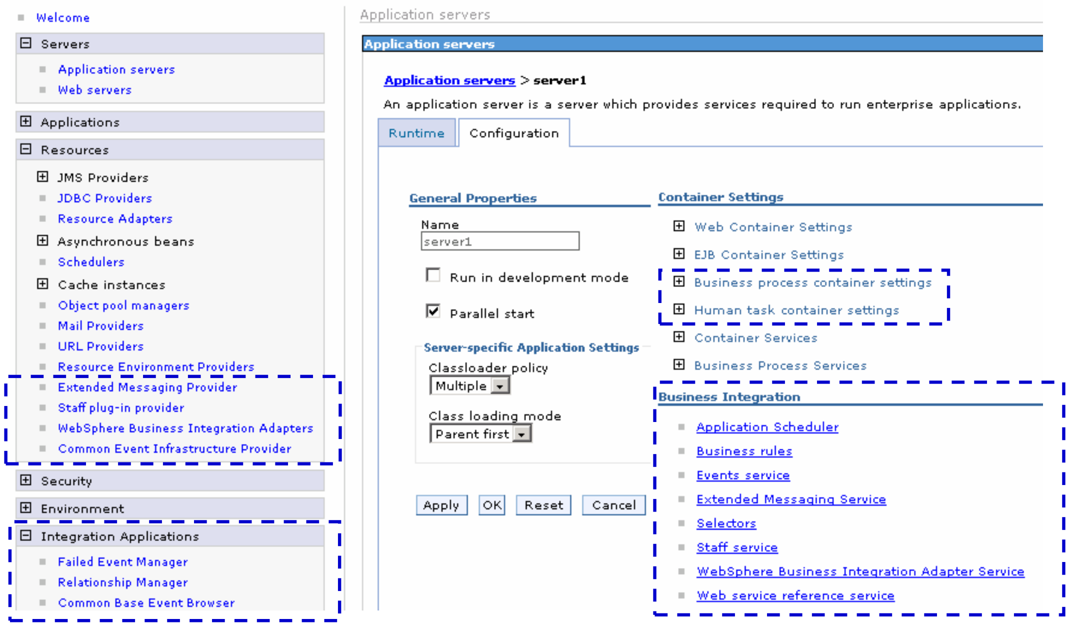
In addition, scripts are provided to create the database tables -- either in Cloudscape (test or development) or DB2 (production).

Overall the process is not difficult. It does require some careful attention, but overall it's common system programmer and database administrator stuff. The more sophisticated stuff takes place when the processes are created and deployed and monitored.

The ZWPS6 workshop is designed to provide you the details you need to install and configure the product.

Proof in Admin Console that Augmentation Succeeded

The Admin Console will show WPS-specific function:

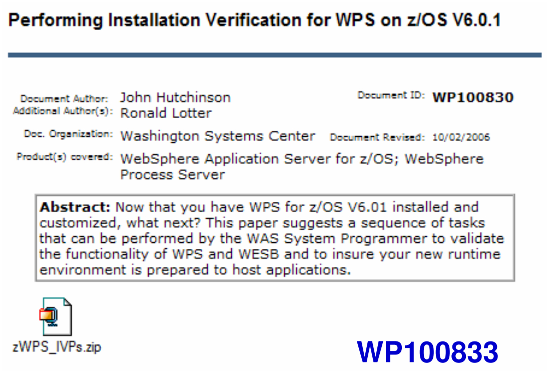


Supporting Techdocs …

One validation mechanism is to check the WebSphere Admin Console … it'll show evidence of it having been updated with WPS stuff. Seeing these things is not proof-positive that everything will work, but it is one indication that the augmentation took place. For more validation routines, there are some techdocs out there.

Techdocs To Support Installation and Verification

ibm.com/support/techdocs



Performing Installation Verification for WPS on z/OS V6.0.1

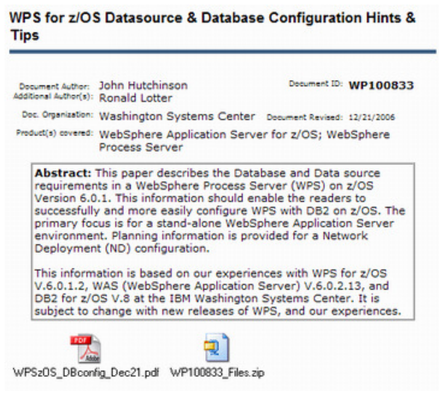
Document Author: John Hutchinson
Additional Author(s): Ronald Lotter
Document ID: **WP100830**
Doc. Organization: Washington Systems Center
Document Revised: 10/02/2006
Product(s) covered: WebSphere Application Server for z/OS; WebSphere Process Server

Abstract: Now that you have WPS for z/OS V6.01 installed and customized, what next? This paper suggests a sequence of tasks that can be performed by the WAS System Programmer to validate the functionality of WPS and WESB and to insure your new runtime environment is prepared to host applications.

zWPS_IVPs.zip

WP100830

Installation Verification



WPS for z/OS Datasource & Database Configuration Hints & Tips

Document Author: John Hutchinson
Additional Author(s): Ronald Lotter
Document ID: **WP100833**
Doc. Organization: Washington Systems Center
Document Revised: 12/21/2006
Product(s) covered: WebSphere Application Server for z/OS; WebSphere Process Server

Abstract: This paper describes the Database and Data source requirements in a WebSphere Process Server (WPS) on z/OS Version 6.0.1. This information should enable the readers to successfully and more easily configure WPS with DB2 on z/OS. The primary focus is for a stand-alone WebSphere Application Server environment. Planning information is provided for a Network Deployment (ND) configuration.

This information is based on our experiences with WPS for z/OS V.6.0.1.2, WAS (WebSphere Application Server) V.6.0.2.13, and DB2 for z/OS V.8 at the IBM Washington Systems Center. It is subject to change with new releases of WPS, and our experiences.

WPSzOS_DBconfig_Dec21.pdf WP100833_Files.zip

WP100833

Database configuration and data source definition setup

As well as …

PRS2232 -- "Shell script tool to fix WPS for z/OS 6.0.1.2 customized variable settings"
PRS2378 -- "WebSphere Process Server v6.0.1 for z/OS Base AppServer Config. Lab"
PRS2520 -- "WPS/WESB V.6.0.x Configuration Samples and Downloads"
WP100878 -- "WPS/WESB v6.0.1 for z/OS: ND Configuration Lab"

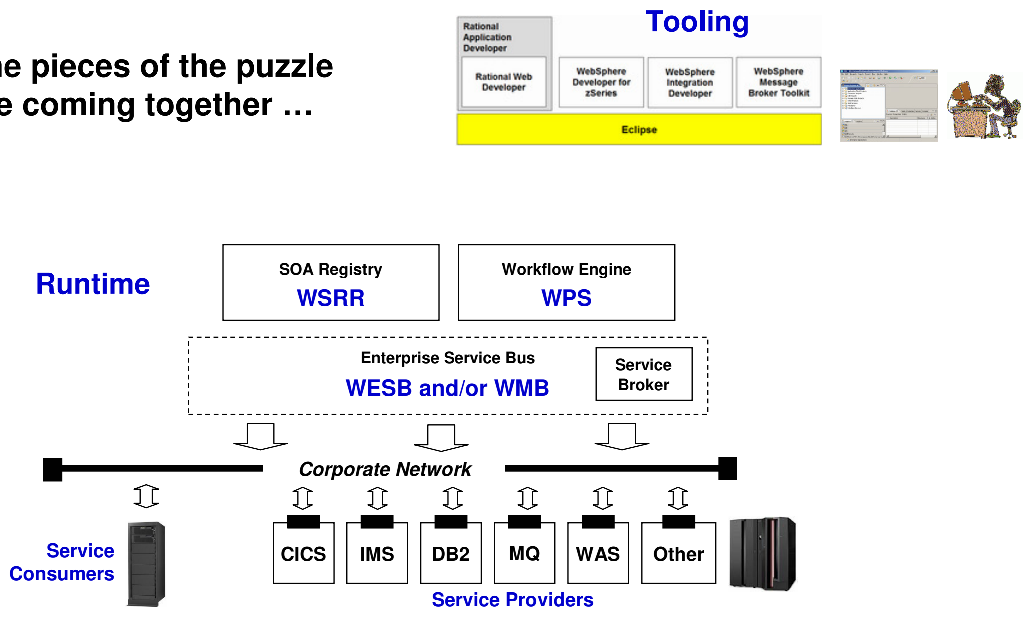
Our original picture, again …

16 IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD
© 2007 IBM Corporation

These techdocs help clarify the process of setting up the environment and validating that things are running as they should run. If you are involved in the installation and configuration of WPS, you should consult these documents to assist your effort.

Our Original Picture … Once Again

The pieces of the puzzle are coming together …



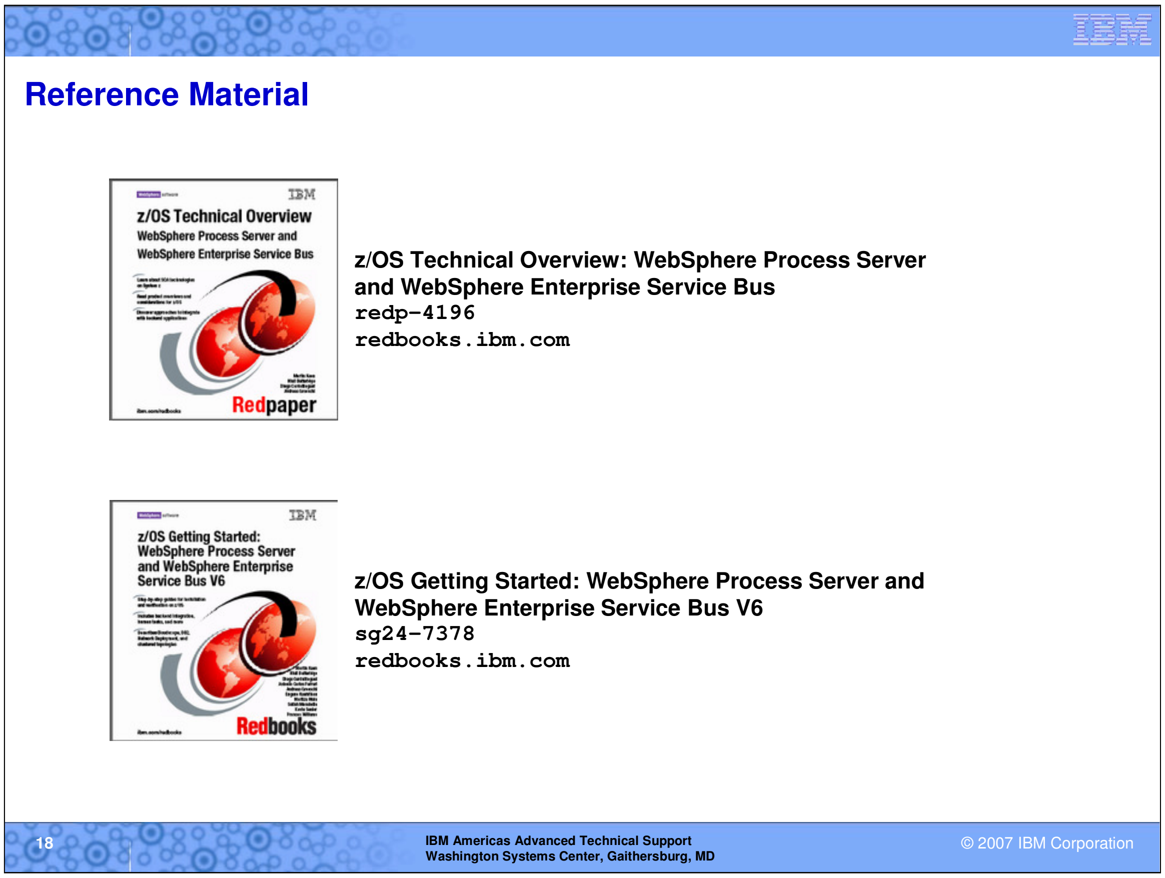
Reference …

Here's our original picture, this time re-swizzled just a big. It shows the same service consumers and service providers, as well as the corporate network at before. The ESB is shown as middleware function mapped onto this network in the form of WESB and/or WMB.

Note: recall that if WPS is in place, WESB is present as well.

On top of that sits the workflow engine in the form of WPS, and it provides the ability to coordinate the execution of processes, which is more than a message-in/message-out model. And providing a registry and repository for the whole thing is WSRR, which is implemented in WebSphere Application Server and is programmatically accessible by WESB, WMB and WPS (and other things).

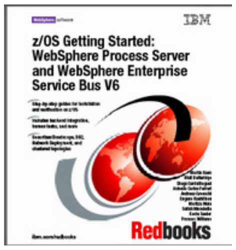
The tooling is all Eclipse based … graphical, consistent, powerful.



Reference Material

z/OS Technical Overview
WebSphere Process Server and WebSphere Enterprise Service Bus
Redpaper

z/OS Technical Overview: WebSphere Process Server and WebSphere Enterprise Service Bus
`redp-4196`
`redbooks.ibm.com`


z/OS Getting Started: WebSphere Process Server and WebSphere Enterprise Service Bus V6
Redbooks

z/OS Getting Started: WebSphere Process Server and WebSphere Enterprise Service Bus V6
`sg24-7378`
`redbooks.ibm.com`

18

IBM Americas Advanced Technical Support
Washington Systems Center, Gaithersburg, MD

© 2007 IBM Corporation

Some good reading material.

End of Unit