



IBM Software Group

IBM® WebSphere® Application Server V7 Feature Pack for Service Component Architecture

Creating and deploying an SCA application – simple SCA application



@business on demand.

© 2008 IBM Corporation
Updated December 12, 2008

This presentation will show you how to create and deploy a simple SCA application.

Section

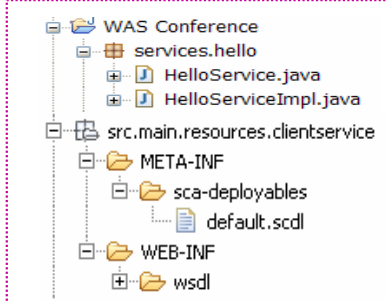
Creating and deploying a sample SCA application



This next section discusses how you go about creating and deploying a simple SCA application.

SCA application packaging

The POJO JAR package



The composite definition

```
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
  xmlns:foo="http://helloservice"
  name="HelloService">
  <component name="HelloService">
  <implementation.java
  class="services.hello.HelloServiceImpl"/>
  </component>
</composite>
```

The Java™ implementation with annotations

```
// This is the service interface
package services.hello;
@Remotable
public interface HelloService {
String hello(String message);
}
```

```
// This is the service implementation
package services.hello;
import org.osoa.sca.annotations.*;

@Service(HelloService.class)
public class HelloServiceImpl
  implements HelloService {
  public String hello(String message) {
  ...
  }}
```

First, look at the packaging to see where things go before creating an application.

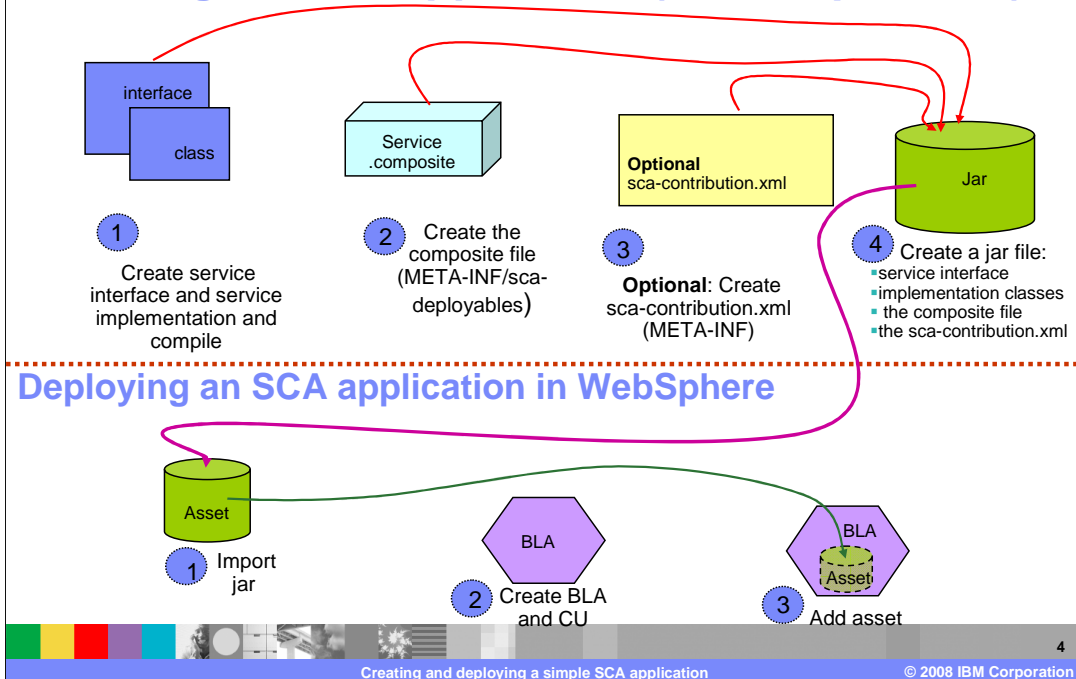
Take a look at the POJO JAR package on the top left side. As you can see, when you create a simple service interface class with its implementation class, these classes will go under a package. The **default.scdl** goes under the META-INF directory and, of course, if there was a wsdl involved, it goes under the WEB-INF directory. The composite files would go in the sca-deployables directory.

Also on that picture you can see a sample composite definition, the interface class and the implementation class. Note the use of the @service annotation that was discussed earlier and the @Remotable.

As a re-cap, the @Remotable annotation on a Java interface indicates that the interface is designed to be used for remote communication. Remotable interfaces are intended to be used for **coarse grained** services. Operations parameters and return values are passed **by-value**.

Note that the SCA Packaging model is not dependent on JEE JAR, zip, or WAR files; therefore it can be deployed directly into WebSphere Application Server.

Creating an SCA application (service provider)



This picture is a summary of the steps needed to create and deploy an SCA application. Under creating an SCA application, you can see the implementation classes, the composite file, contribution file and the .jar file. Under deployment, you can see the asset, business level application and the asset added into the business level application. Each of these steps is discussed in the next couple of slides.

Building an SCA application with default binding (1)

1 a. Build the service interface class

Example: HelloWorldService.java

b. Build the service implementation class

Example: HelloWorldServiceImpl.java

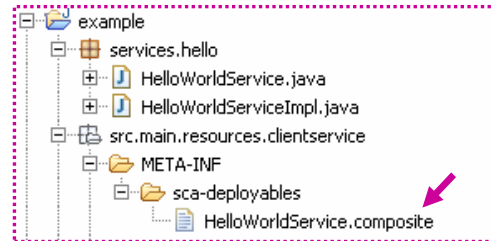
c. Compile these classes

2 Create a composite file
(.composite – example)

HelloWorldService.composite)

to declare the sca composite,
components and services

- place this composite file under META-INF/sca-deployables directory



Now that you have seen the general structure of how files are laid out in SCA, the next step is to create a simple SCA application step by step. The first step is to create the Java classes, the service interface classes and the service implementation classes, and then compile them.

In general, a best practice for application development is to start with WSDL as the interface, run `wsimport` to generate the Java interface. After that you can write the implementation from the generated Java interface and the generated JAXB objects.

Once you have created your Java classes, the next step is to create a composite file to declare the SCA composite, components, and services. The name of the composite file and the directory where it is stored does not matter if you plan on creating the `sca-contribution.xml` file as explained later. Otherwise you need to place this composite file under `META-INF/sca-deployables` directory.

Sample HelloWorldService.composite

```
<composite autowire="false"
  local="true"
  name="HelloWorldService"
  targetNamespace="http://foo"
  xmlns:foo="http://foo"
  xmlns="http://www.oesa.org/xmlns/sca/1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.oesa.org/xmlns/sca/1.0 http://www.oesa.org/xmlns/sca/1.0 ">

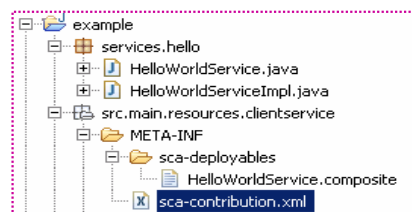
  <component name="HelloWorldServiceComponent">
    <implementation.java class="myapp.HelloWorldServiceImpl"/>
  </component>
</composite>
```



Here is a sample composite file and what it looks like. Note the specified namespace and the component name.

Building an SCA application with default binding (2)

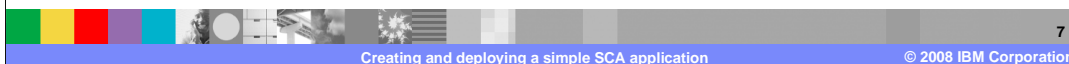
- 3 OPTIONAL: Create the `sca-contribution.xml` under META-INF directory



Sample `sca-contribution.xml`

```
<?xml version="1.0" encoding="ASCII"?>
<contribution xmlns="http://www.osoa.org/xmlns/sca/1.0" xmlns:foo="http://foo">
<deployable composite="foo>HelloWorldService"/>
</contribution>
```

- 4 Create a **jar** containing:
- service interface
 - implementation classes
 - composite file
 - `sca-contribution.xml` file



The third step is creating the **sca-contribution.xml** file under the META-INF directory. It is **optional** to create this file if the composite file is created in the `sca_deployables` directory. You can have composites anywhere in the system, and you can use a contribution file to declare the deployable composites. If you create a contribution file, make sure that the namespace under which the deployable composite is declared is the target namespace defined for that composite in its composite file. In the example shown, HelloWorldService composite is declared under the namespace `http://foo`, which is the target namespace for that composite declared in the HelloWorldService.composite.

The final step is to create a jar containing service interface and implementation classes, the composite file and the `sca-contribution.xml` file. This becomes your server side SCA application.

Now that an application has been created, you will see how you run and look at it.

Building war file (side step)

Client side

- On the client side, create a WAR file with a JSP to locate the helloworld service

Sample of what is in the JSP

```
private HelloWorldService helloWorldService;  
private CompositeContext context;  
  
context = CurrentCompositeContext.getContext();  
helloWorldService = context.getService(HelloWorldService.class,  
"HelloWorldServiceComponent/HelloWorldService");
```

- Install the WAR into the administrative console



On the client side of things, you need to create a WAR file with a simple JSP to locate the HelloWorld Service. This step can be done anytime during the creation or deployment of an SCA application. The example on the slide shows what is inside the JSP. Note the **context.getService** to locate the HelloWorld Service. Note that context.getService is not supported on the Web services binding. Once you have the WAR file in place, you are now ready to deploy the application. You can install the WAR file through the administrative console just like any JEE application.

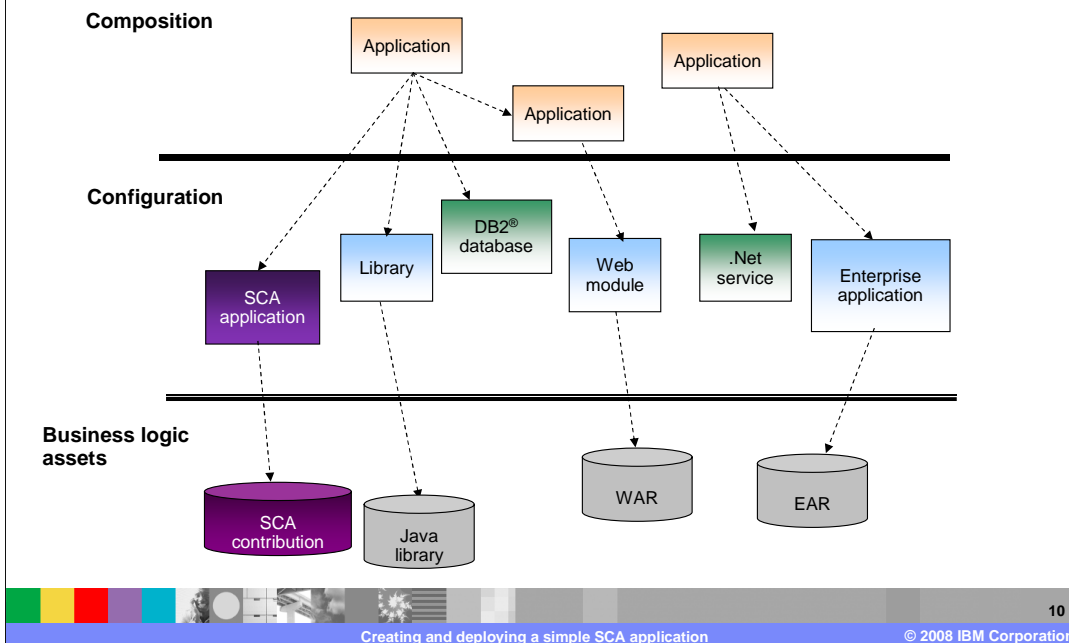
Section

Deploying an application



This section will show you how you deploy an SCA application using the administrative console.

Applications as compositions - BLA



In the administrative console, there is the concept of business level applications. A business level application, or BLA, is a WebSphere configuration artifact that has these characteristics:

- 1) It captures the entire definition of an enterprise level application consisting of WebSphere Application Server, though it may not explicitly manage the life cycle of every constituent part of its definition. It is a composition model for defining an application.
- 2) It does not represent or contain the application binaries. This provides a clean separation between the administration of binaries versus the administration of the application definition.
- 3) It also supports recursive composition. The composition at its lowest level consists of configured instances of application binaries that are configured to run in a specific runtime environment such as WebSphere Application Server.

The above three-tier diagram shows the **composition model** for a business level application.

At the bottom are the assets such as the JAR files, WAR files, EAR files and so on and the **SCA contribution** and the **SCA WAR file**. The next level up is the composition units. The composition units are the deployed assets such as the EJB modules, Web modules, Web service modules and so on. This level also contains the **SCA application**. At the top level are the Business-level applications.

As an example, you can take an EAR file (an asset) and install it into WebSphere. This EAR file gets saved in the WebSphere repository as a composition unit, or CU. This CU can then get added to a BLA.

SCA contribution – Import asset

Integrated Solutions Console Welcome booz Help | Logout

View: All tasks

Assets

Assets help

Use this page to manage assets in the asset repository.

Preferences

Import Delete Update Export

Select	Name	Description
<input type="checkbox"/>	exa1.jar	Business Service example1

Total 1

Field help

For field help information, select a field label or list marker when the help cursor appears.

Page help

[More information about this page](#)

Command Assistance

[View administrative scripting command for last action](#)

1 Import asset

Done Local intranet

Creating and deploying a simple SCA application © 2008 IBM Corporation

Now to continue with the deployment of the SCA application that was created a couple of slides earlier. If the jar you created is called `exa1.jar`, that JAR file needs to be imported as an asset. That is done in the administrative console through the menu option, Applications->ApplicationTypes->Assets and then select "Import."

Administrative console experience - BLAs

Integrated Solutions Console Welcome booz Help Logout

View: All tasks

- Welcome
- Guided Activities
- Servers
- Applications
 - New Application
 - Application Types
 - WebSphere enterprise applications
 - Business-level applications
 - Assets
- Services
- Security
- Environment
- Services
- System administration
- Users and Groups
- Monitoring and Tuning
- Troubleshooting
- Service integration
- UDDI

Business Level Applications

Business Level Applications

Use this panel to manage business level applications.

Preferences

Start Stop New Delete

Select	Name	Description	Status
<input type="checkbox"/>	example1	example1 is a web service	➔

Total 1

Field help
For field help information, select a field label or list marker when the help cursor appears.

Page help
[More information about this page](#)

Command Assistance
[View administrative scripting command for last action](#)

2 Create a business level application

Next you create a business level application from the menu Applications->Application Types->Business level Applications.

Administrative console experience – SCA composite

The screenshot displays the IBM Integrated Solutions Console (ISC) interface for managing a Business Level Application (BLA). The main content area shows the configuration for an application named 'example1'. The 'General Properties' section includes fields for Name (example1), Description (example1), and a detailed description (WebSphere:assetname=exa1.jar,assetversion=1.0). The 'Additional Properties' section contains links for 'Provide HTTP endpoint URL information', 'View SCDL', and 'View Domain'. The 'SCA composite components' section is highlighted with a red box and a callout bubble containing the number '3' and the text 'Add asset to the business level application'. This section contains a table with the following data:

Component Name	Component Implementation
BasicBusinessComponent	com.exe.impl.BusinessServiceVanilla

Below this table, there are sections for 'SCA composite services' and 'SCA composite references'. The 'SCA composite services' section shows a table with 'Service Name' and 'Service: Promote'. The 'SCA composite references' section shows a table with 'Reference Name', 'Reference Promote', and 'Reference Target'.

The bottom status bar of the console displays the text 'Creating and deploying a simple SCA application' and '© 2008 IBM Corporation'.

Next you add the asset under the BLA, which will create the SCA composite components. From the picture, note the SCA composite component created. You can also install the war file from the administrative console so that the service is seen. Once that is done, you are ready to start the SCA application.

This concludes the SCA application creation and deployment process. Bear in mind that this is a very simple case of creating an SCA application. Many other things can come into play like the quality of service when creating SCA applications. For example, you could add authorization policies to composites and configure user roles on the administrative console during deployment. Other presentations look at security and authorization as quality of services.

The next three slides show you a quick snapshot of some of the things you might configure on an SCA application.

IBM Software Group IBM

Administrative console: Intents

Integrated Solutions Console Welcome gregd Help | Logout

View: All tasks

- ▢ Welcome
- ▢ Guided Activities
- ▢ Servers
- ▢ Applications
 - ▢ New Application
 - ▢ Application Types
 - ▢ WebSphere enterprise applications
 - ▢ Business-level applications
 - ▢ Assets
- ▢ Services
- ▢ Resources
- ▢ Security
- ▢ Environment
- ▢ System administration
- ▢ Users and Groups
- ▢ Monitoring and Tuning
- ▢ Troubleshooting
- ▢ Service integration
- ▢ UDDI

Step 1: Set options

Step 2: Map composition unit to a target

Step 3: Map Virtual Host

→ Step 4: Attach policy set

Step 5: Summary

Attach policy set

Specify policy set for the composite defined in this SCA application.

Include default policy sets. Default PolicySets specify common QoS behavior for generic message format. Evaluate whether they provide adequate QoS characteristics for your services before applying Default PolicySets.

Attach ▾ Detach Policy Set Assign service policy set binding Assign reference policy set binding

Select	Name	Intents	Matched policy sets	Attached policy set
<input type="checkbox"/>	SampleComposite			
<input type="checkbox"/>	SampleComponent			
<input type="checkbox"/>	SampleService			
<input type="checkbox"/>	binding.ws	confidentiality,message	WS-I RSP, Username WSSecurity default, WSSecurity default	
<input type="checkbox"/>	sampleReference			
<input type="checkbox"/>	binding.ws	confidentiality,message	WS-I RSP, Username WSSecurity default, WSSecurity default	

Creating and deploying a simple SCA application © 2008 IBM Corporation

As an example, from the administrative console, intents can be specified for your application optionally when adding assets with policy set defined to business level applications.

Security policy example – Map to policySet

Business Level Applications Close page

Create new business level application

Use this page to create a new business level application.

→ Step 1: Attach policyset
 Step 2: Set options
 Step 3: Map composition unit to a target
 Step 4: Summary

Attach policyset

Specify policy set for the composite defined in this SCA application.

Include default policy sets. Default PolicySets specify common QoS behavior for generic message format. Evaluate whether they provide adequate QoS characteristics for your services before applying Default PolicySets.

Attach - Detach	Service/Reference/Binding	Type	Intents	Matched policy sets	Attached policy set
LTPA SecureConversation					
Username RAMP default					
Username WSSecurity default					
WSHTTPS default					
WSTransaction					
RAMP default					
WSReliableMessaging default		Composite			
WSReliableMessaging persistent	intent	Component			
WSSecurity default		Component/Service			
Username SecureConversation					
LTPA WSSecurity default					
WSAddressing default					
LTPA RAMP default					
WSReliableMessaging 1_0					
SSL WSTransaction					
SecureConversation		Component/Service/Web Services Binding	authentication	LTPA SecureConversation, Username RAMP default, Username WSSecurity default, WSHTTPS default, WSTransaction, Username SecureConversation, LTPA WSSecurity default, LTPA RAMP default	

Next Cancel

© 2008 IBM Corporation

This example shows how the intent is mapped to the policy set on the administrative console using the “**Attach**” button.

Security policy example – Service providers

The screenshot shows the IBM Service Providers configuration console. The main content area is titled "Service providers" and shows the configuration for "BusinessService1". The "Policy set attachments" section is highlighted with a red dashed box. The table below shows the attached policy sets and their bindings.

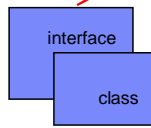
Select	Service/Endpoint/Operation	Attached policy set	Binding
<input type="checkbox"/>	BusinessService1	WSSecurity default	Default
<input type="checkbox"/>	BusinessService1SoapPort	WSSecurity default (inherited)	Default (inherited)

Total 2

You can also see from the service providers the policy set that got attached to the service component.

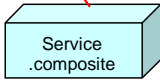
Creating an SCA application (service provider)

Recap!



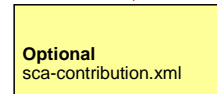
1

1 Create Service interface and service implementation and compile



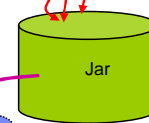
2

2 Create the composite file (META-INF/sca-deployables)



3

3 Optional: Create sca-contribution.xml (META-INF)



4

4 Create a jar file:

- service interface
- implementation classes
- the composite file
- the sca-contribution.xml

Recap!

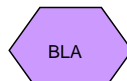
Deploying an SCA application in WebSphere

Recap!



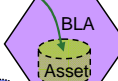
1

1 Import Jar



2

2 Create BLA and CU



3

3 Add asset

Recap!

To summarize the steps one more time: Under creating an SCA application, you create the implementation classes, the composite file, contribution file and the jar. Under Deployment, you import the .jar file as an asset, create a business level application, and add the asset into the business level application.

Section

Summary



In summary, this presentation looked at the creation and deployment of an SCA application.

Summary

- SCA feature pack has implemented the discussed scenarios which will help you understand the features in SCA feature pack
- The shipped samples are a good start at looking at these features in preparation for creating your own SCA applications



The SCA feature pack allows you to choose from three different development approaches. A good starting point to learn about your options is with the samples that are provided with the product.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASv7SCA_CreatingSCApp_createDeployApp.ppt

This module is also available in PDF format at:

._/WASv7SCA_CreatingSCApp_createDeployApp.pdf



You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:
DB2 WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>

EJB, Java, JSP, and all Java-based trademarks and logos 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.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.