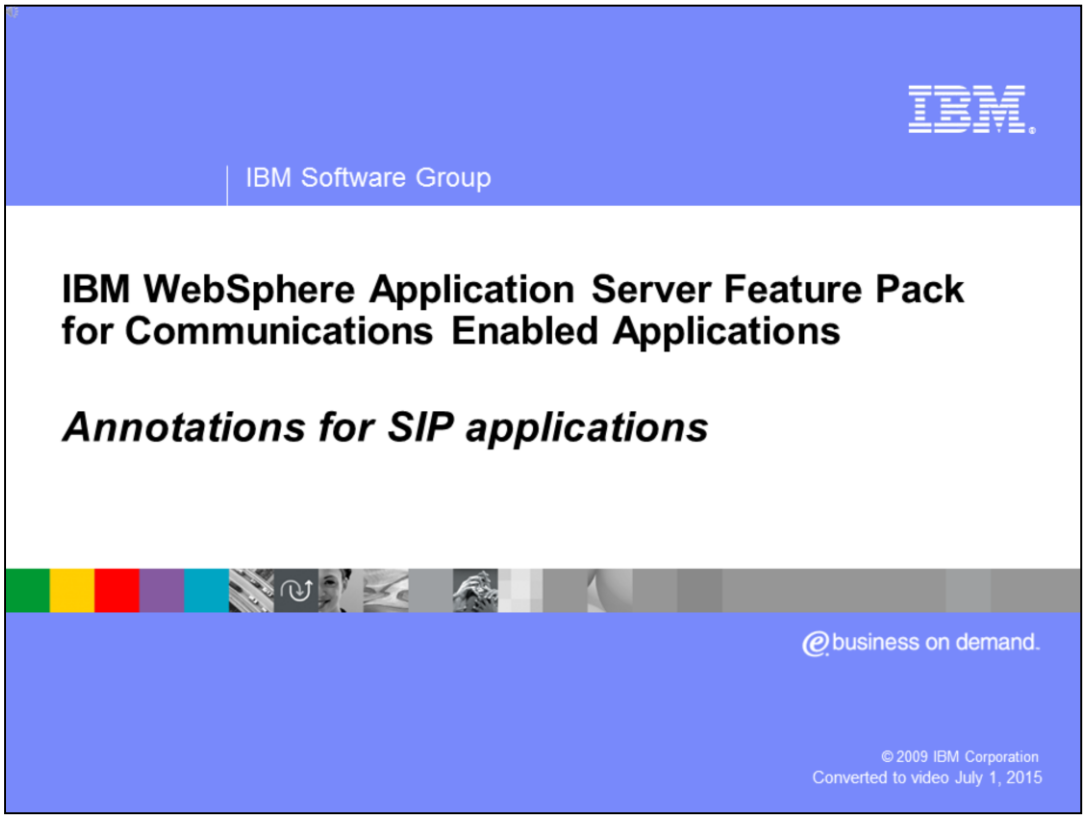
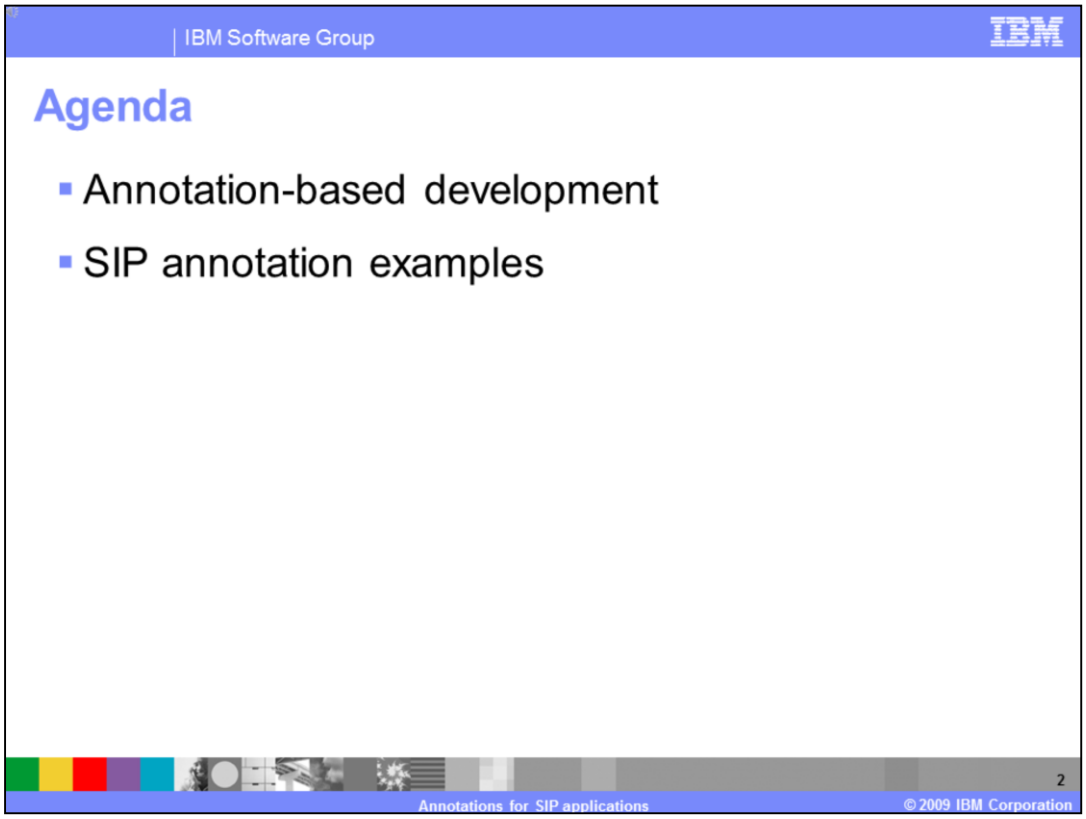
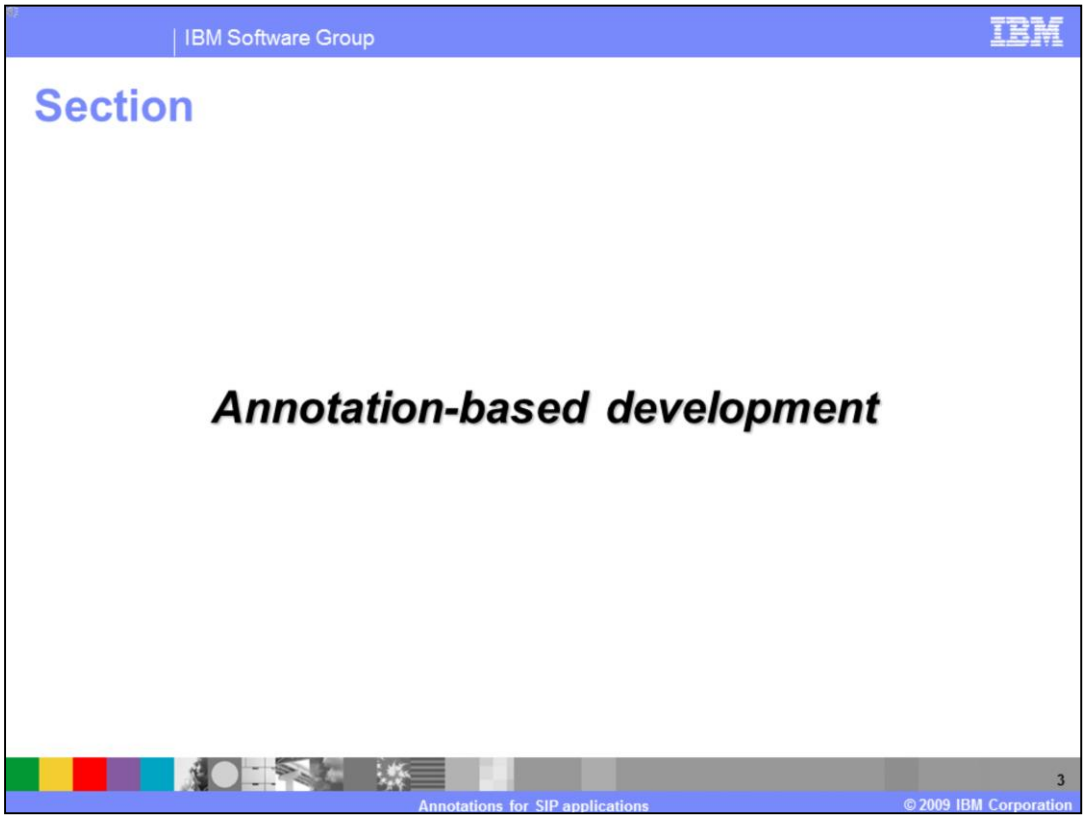


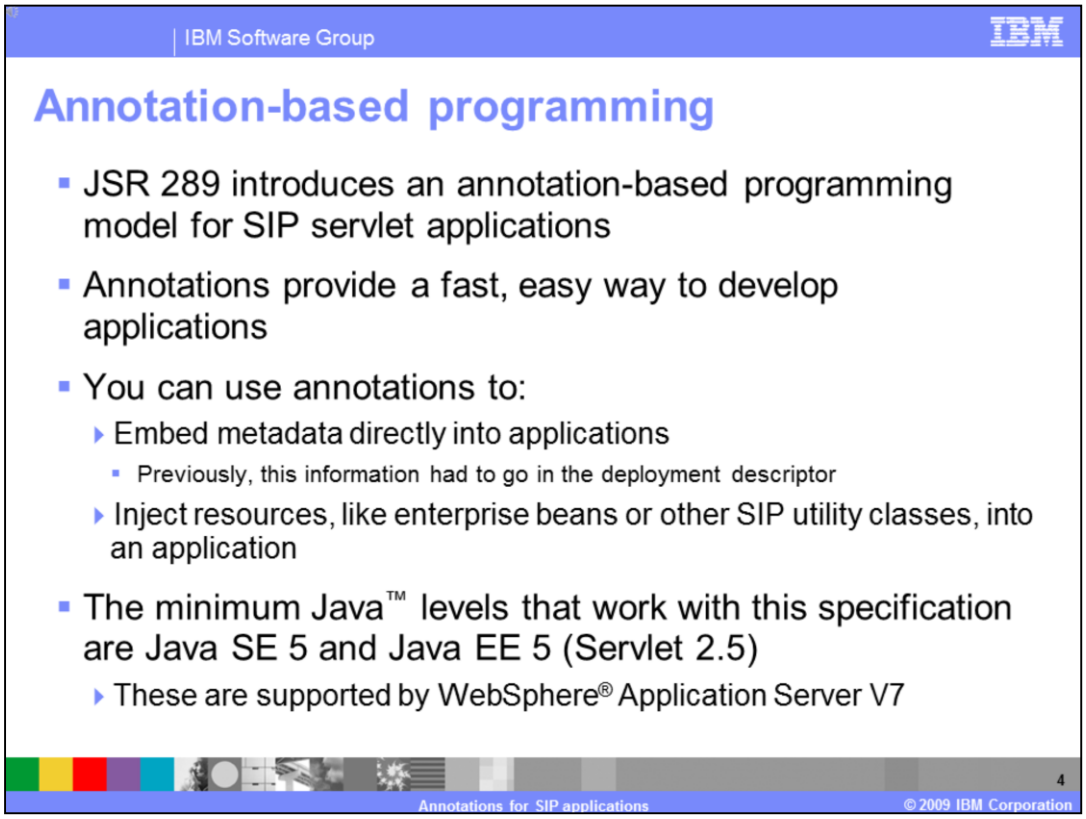
This presentation covers the new annotation-based programming model for SIP applications that was introduced in the SIP servlet 1.1 specification.



The first section of this presentation provides a brief overview of annotation-based development, and the second section takes a look at some examples of the new SIP-related annotations that were added in the JSR 289 SIP servlet 1.1 specification.

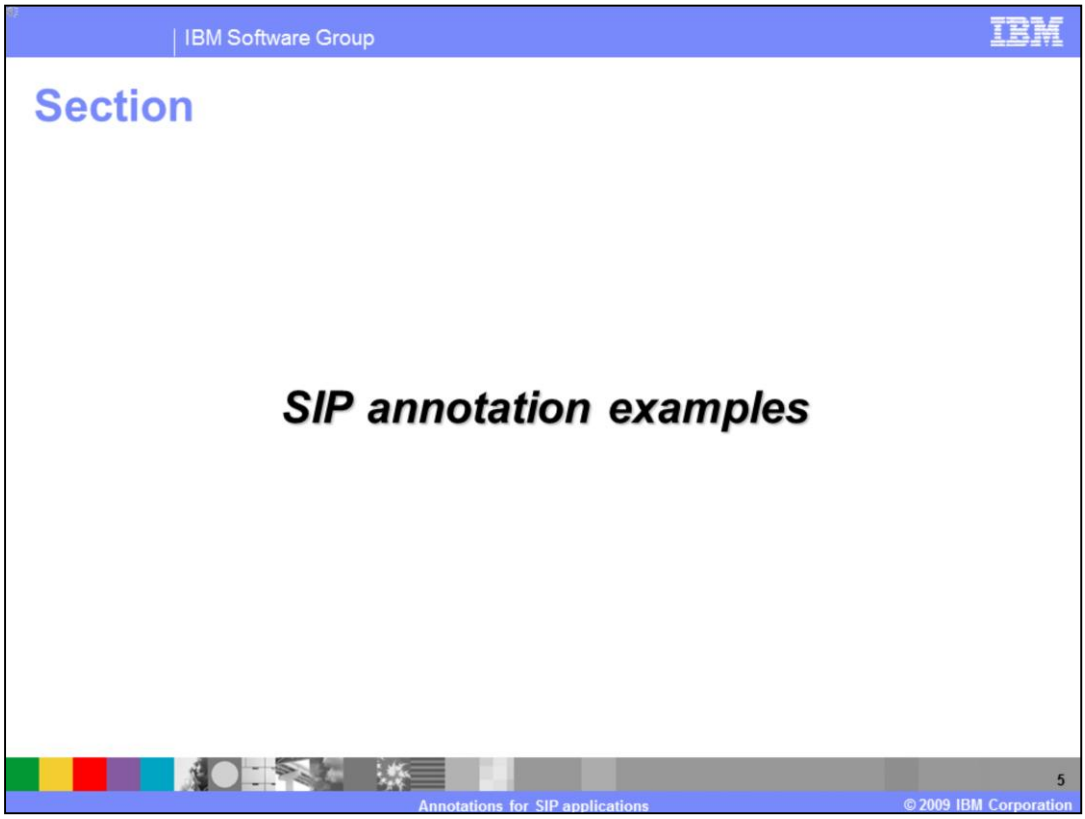


This section provides a brief overview of annotation-based development.

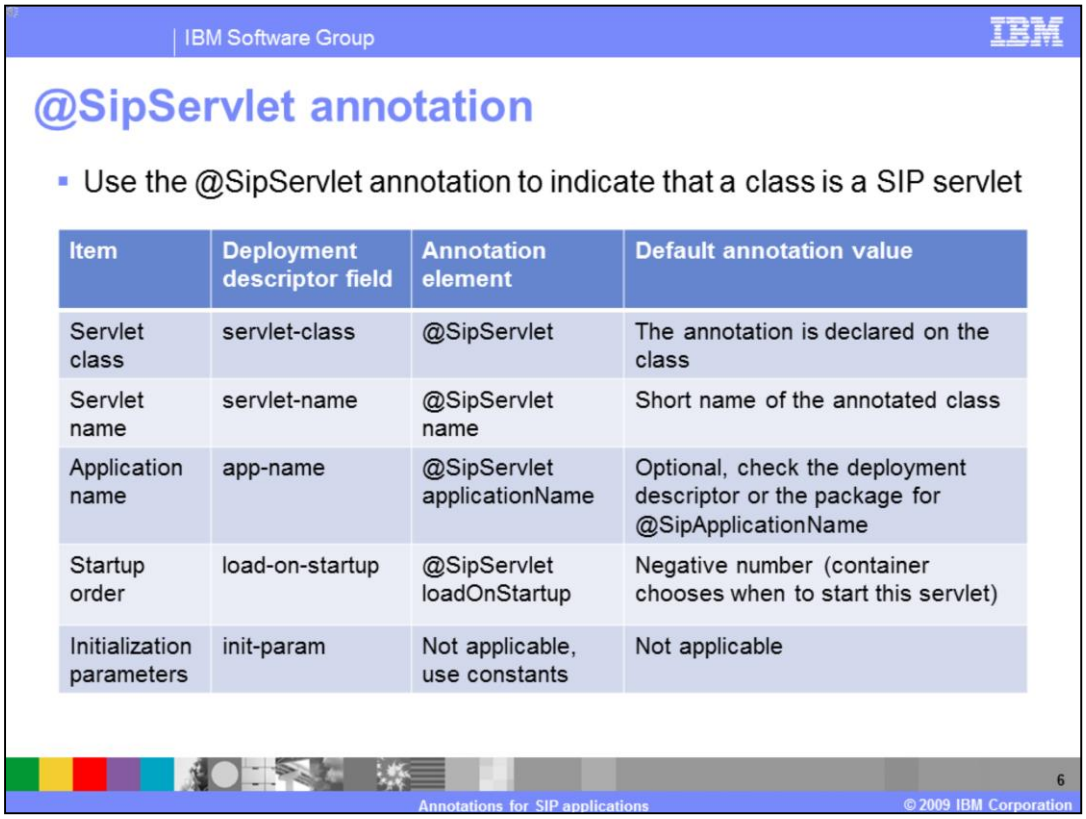


The SIP servlet 1.1 specification introduces an annotation-based programming model for SIP servlet applications, similar to how annotations are used throughout the Java EE 5 specification. Annotations improve the development experience by simplifying the code being created. Annotations allow you to embed metadata directly into applications, rather than having to use deployment descriptors. Deployment descriptors are still an option, and will override settings described in the annotations, but they are not required.

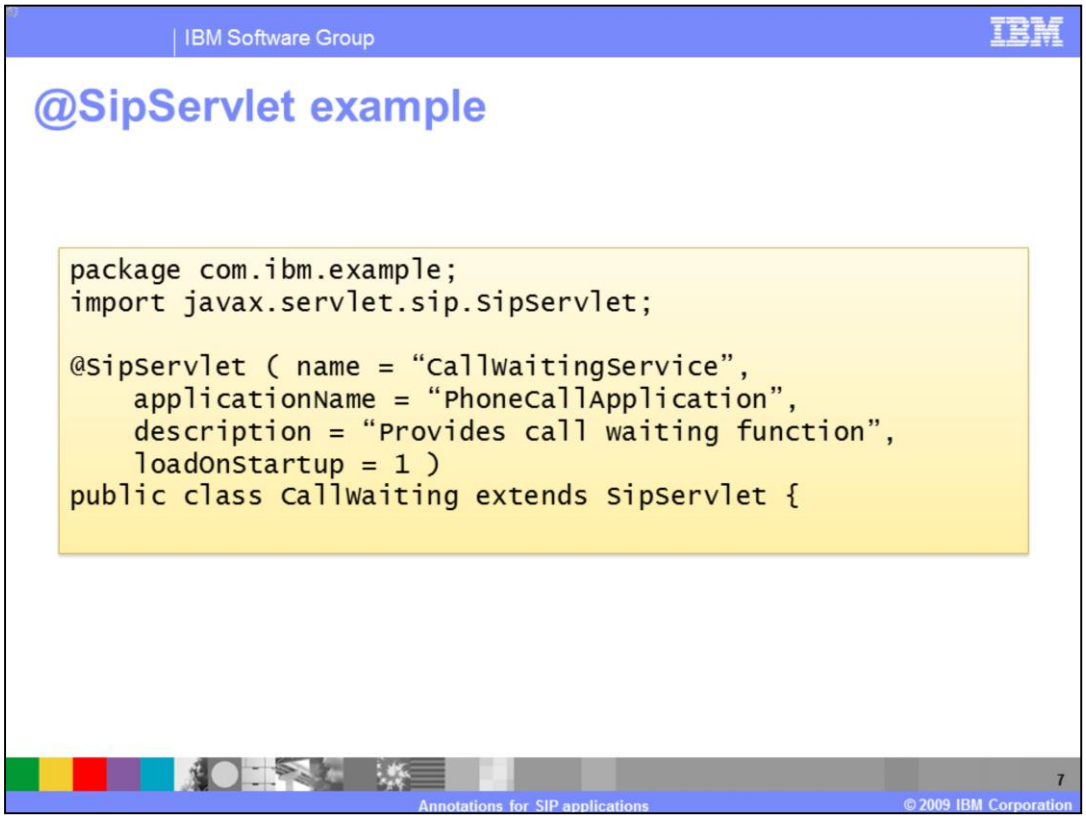
Resource injection is a simplified model for pulling resources, like SIP utility classes or Enterprise JavaBeans, into an enterprise application, and the new SIP servlet annotations in JSR 289 support resource injection. Because of the use of annotations in the SIP servlet 1.1 specification, to use this specification, you need to be using versions of Java that support annotations Java SE 5 and Java EE 5, both of which are supported in WebSphere Application Server Version 7.



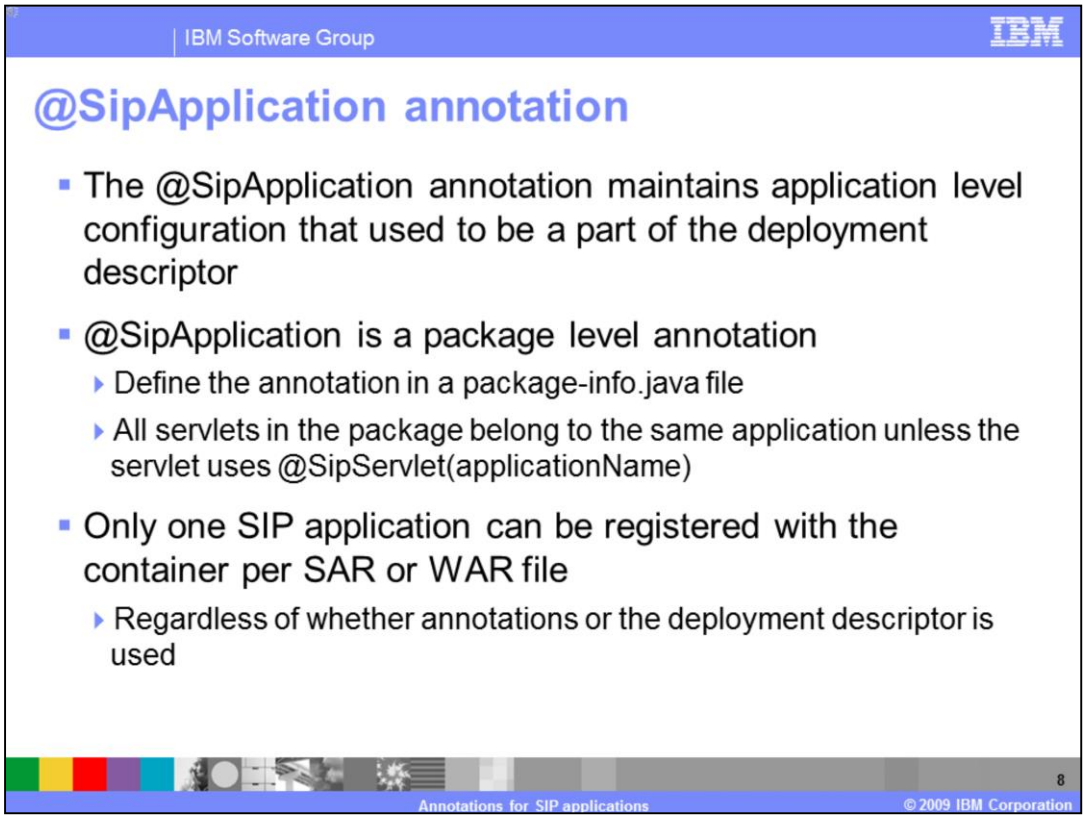
This section describes some of the new annotations introduced as a part of the SIP servlet 1.1 specification.



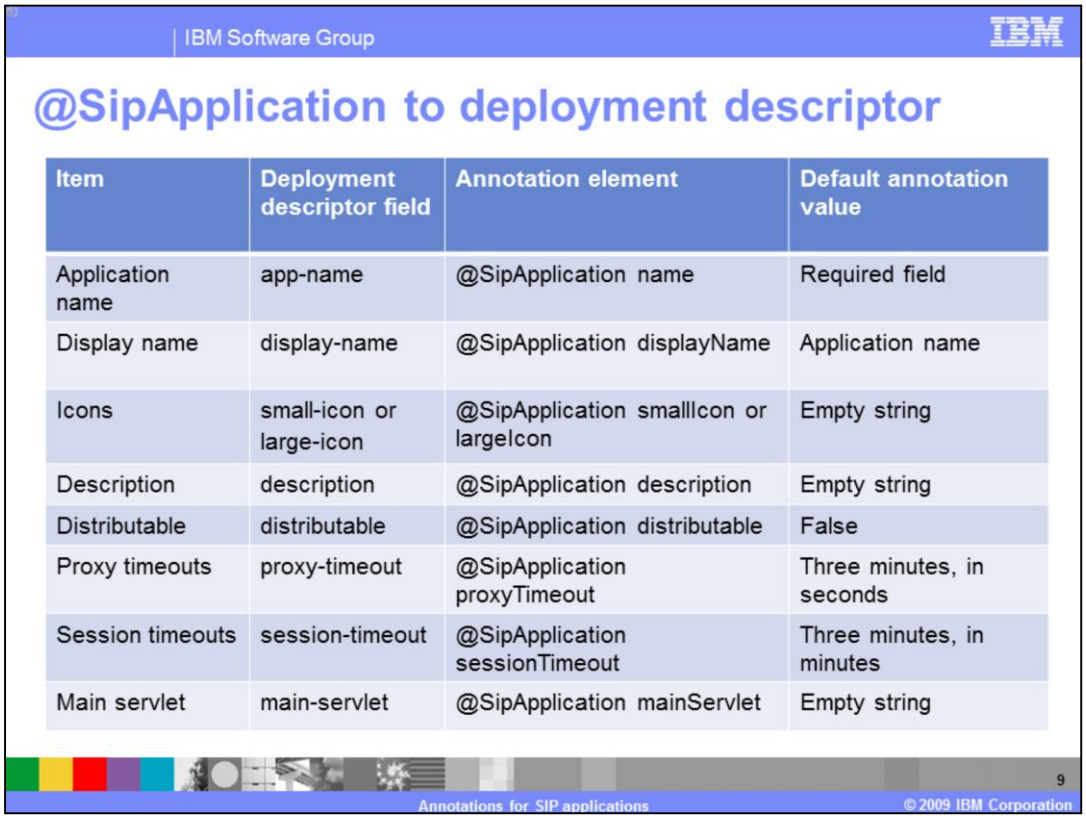
The @SipServlet annotation is used to indicate that a class is a SIP servlet; it allows SipServlet metadata to be declared without having to create a deployment descriptor. Certain values in the deployment descriptor are no longer required since they can now be declared in the source file of the servlet itself. The servlet-class field is not needed since the annotation is declared in the class. The servlet-name field is replaced by the name element on the SipServlet annotation. If no value is provided for the name element, the default behavior is to use the short name of the class. The second element in the SipServlet annotation is the applicationName. This is an optional element; it can also be defined in the deployment descriptor or using a special @SipApplication annotation. The loadOnStartup element defines the starting order of the servlet application within the system. The default value is a negative number, which allows the container to choose when to start the servlet. Finally, the init-parm field is not useful since it can just as easily be a static constant in the source file where the annotation is declared.



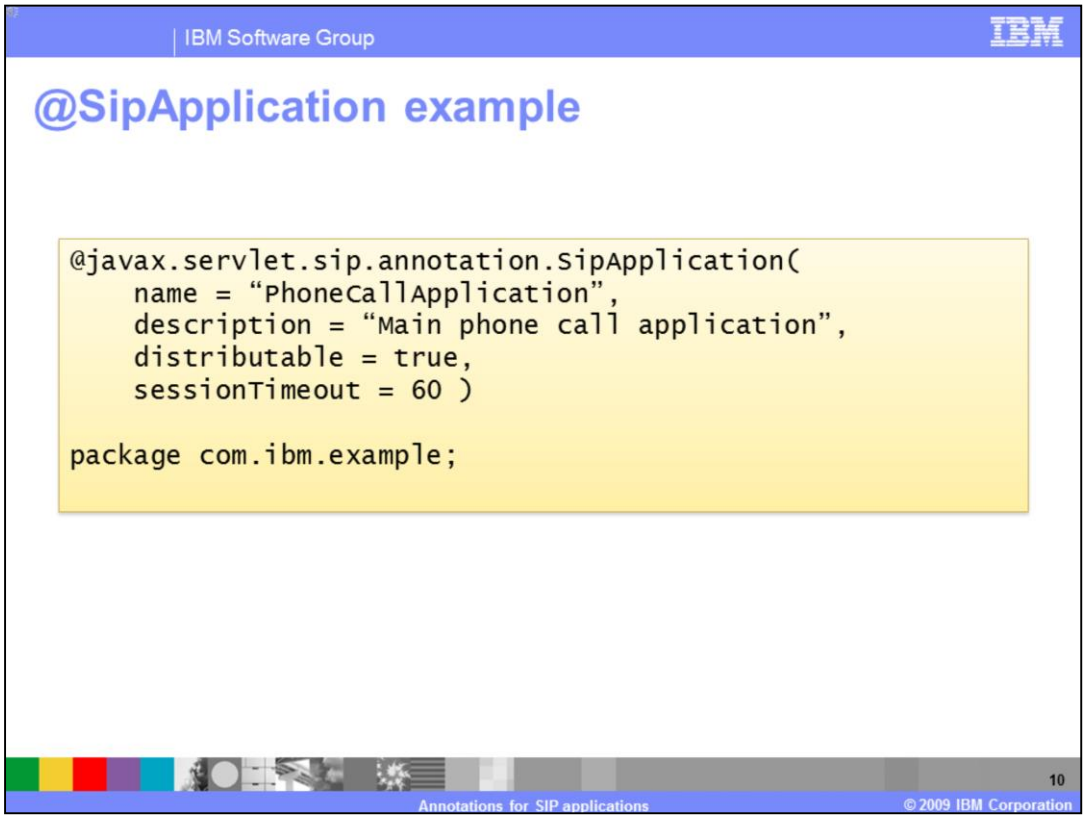
This page shows an example of how to use the @SipServlet annotation to define a SipServlet. This example defines the name of the SipServlet as CallWaitingService. Notice that this is different than what the default name would have been if this element had been omitted from the @SipServlet annotation. This annotation also defines the optional applicationName element. If the applicationName is not set here, it must either be set in a deployment descriptor or using the package level @SipApplication annotation; otherwise, the container treats it as a deployment error. The description element, also optional, can help the consumer of an application understand better what the SipServlet is and what it does. The loadOnStartup element is set to a non-negative number, so the container will use the defined startup weight to start this application. For all servlets with a defined startup weight, the container must start the servlets beginning with the lowest number.



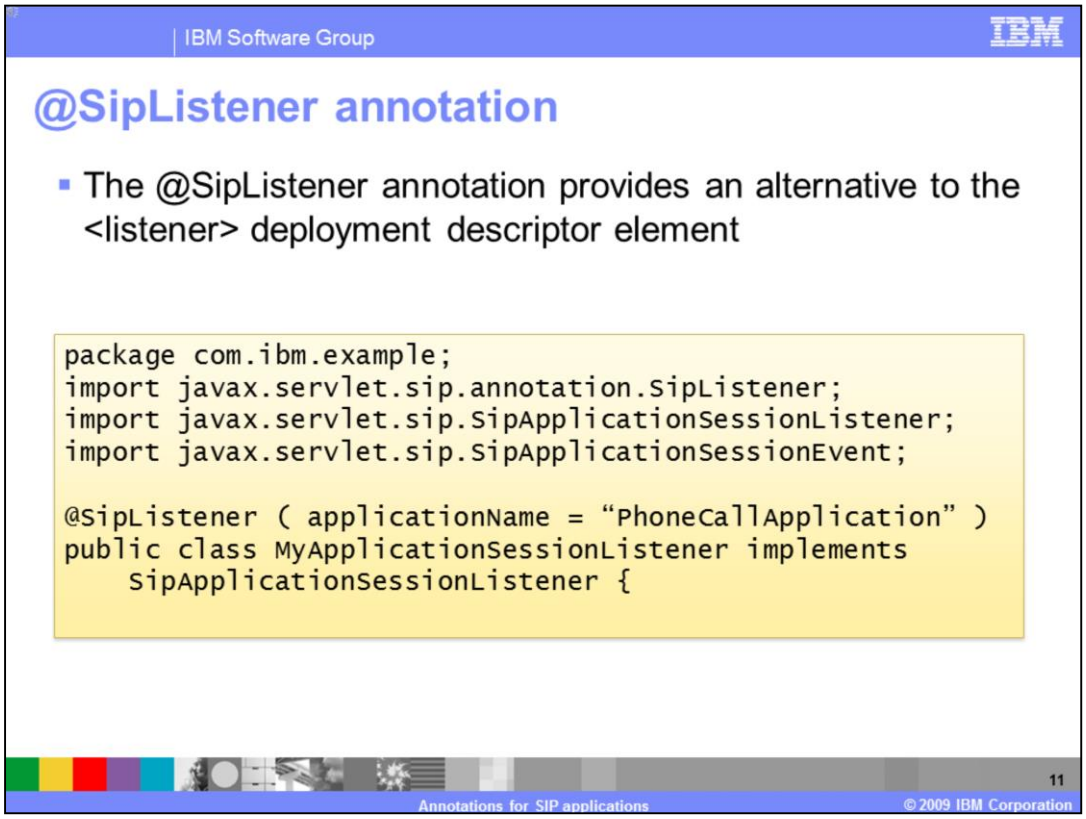
A SIP application is a logical entity that contains a set of servlets and listeners with some common configuration. The @SipApplication annotation is used to define common configuration information about an application. Unlike most other annotations, @SipApplication is a package level annotation that is defined in a special source file, package-info.java. All of the servlets in the package described in package-info.java belong to the same application, unless one of the servlets in the application package overrides the application name using the @SipServlet annotation, as described earlier. Regardless of whether annotations or the deployment descriptor is used, only one SIP application can be registered with the container per SAR or WAR file.



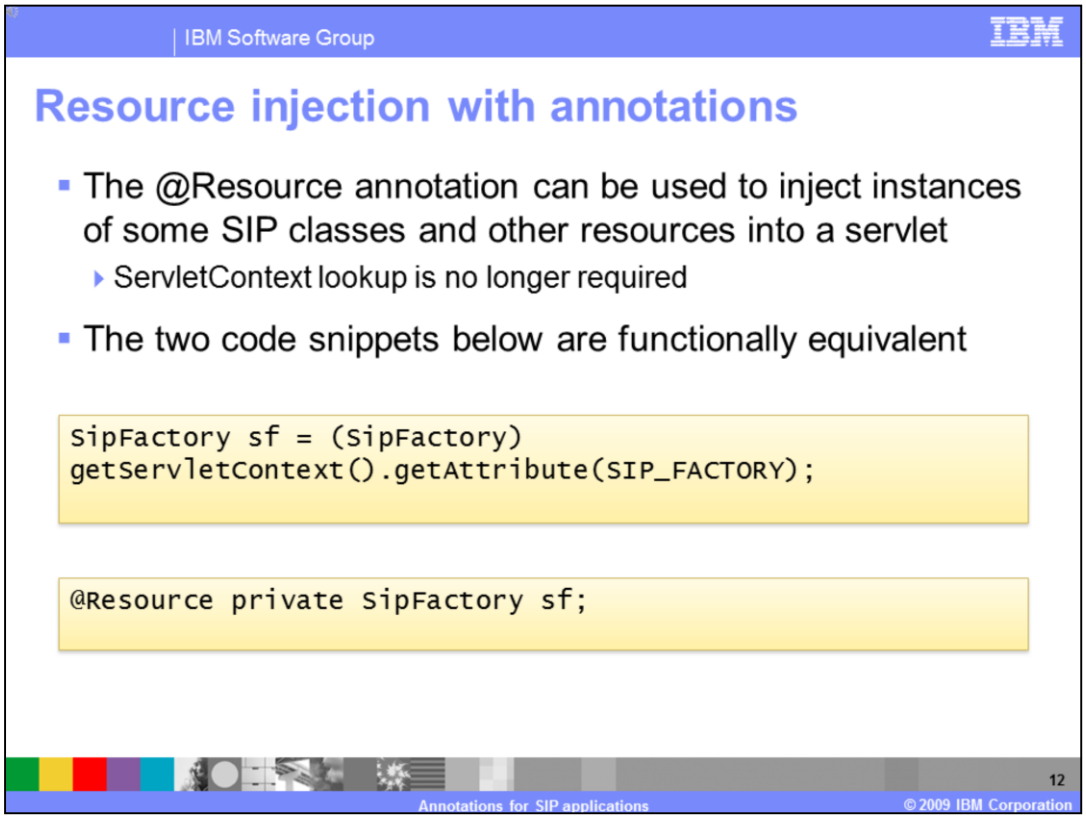
The name element of the @SipApplication annotation is the only required field. It replaces app-name from the deployment descriptor and defines the name that listeners and servlets reference when adding themselves to this logical application. The displayName element is not required, and defaults to the application name; it is equivalent to the display-name field from the deployment descriptor. The optional icon elements can each take a simple string that specifies the location of the image to use for the application's icon, relative to the root path of the archive that contains the class. The description element is also optional and is meant to contain a string that describes the behavior of the application, to help a consumer of the application understand what it does. The distributable element indicates to the container whether this application is developed to properly function in a distributed environment. The default for distributable is false, so the application developer must set this element to true if he wants the application to run in a distributed environment. The proxyTimeout element specifies, in whole seconds, the default timeout for all proxy operations in this application. The container can override this value as a result of its own local policy. The sessionTimeout specifies, in whole minutes, the default session timeout for all application sessions created in this application. Note that this timeout is for application sessions, and not SIP sessions, because the lifetime of the SIP session is tied to that of the parent application session. If this is set to a non-positive number, then the behavior of the sessions is that they never time out. The mainServlet element of the @SipApplication annotation defines which servlet is designated as the main servlet of the application. A main servlet must be defined for any application that contains multiple servlets. When the application router tells the container to call into an application to process a request, the main servlet is invoked for the initial request.



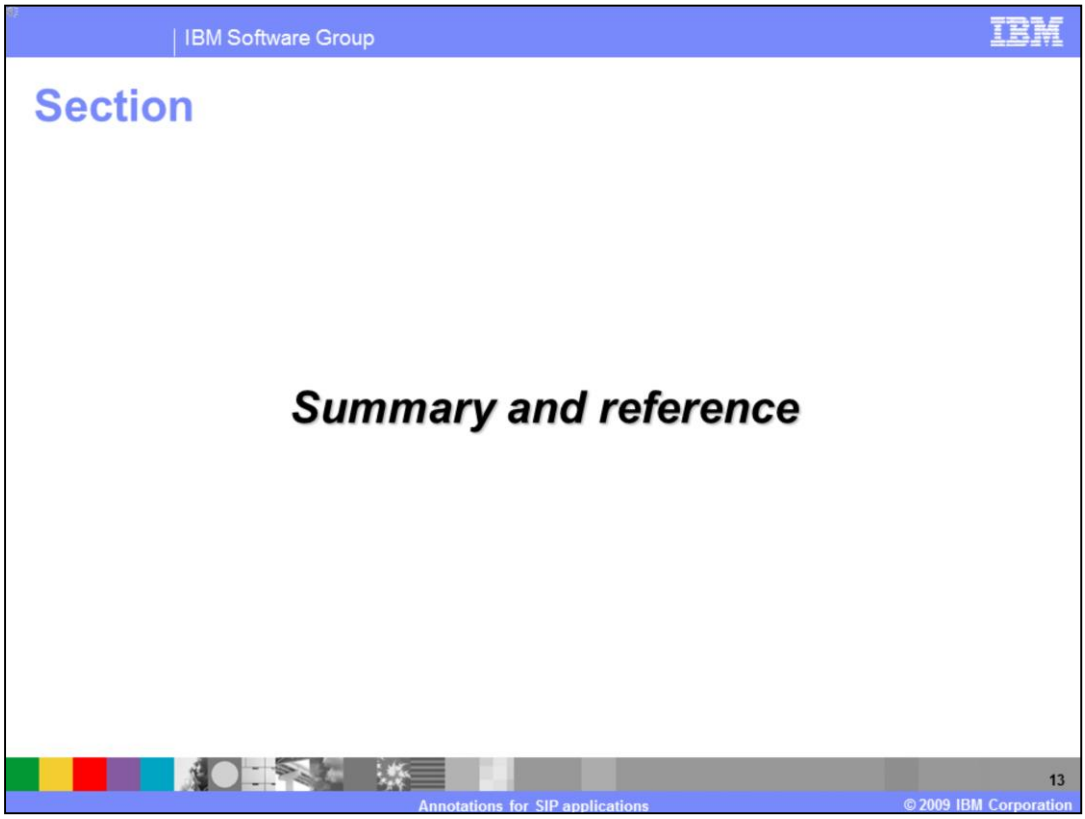
The @SipApplication annotation needs to be defined in a special package-info.java file that is included in the application archive. This example shows an example of the syntax for this annotation. The name element is the only one that is required, all other fields are optional. In this case, the application developer is explicitly overriding the default distributable setting and the default session timeout. This application is then defined to function properly in a distributed environment, and has a session timeout of 60 minutes.



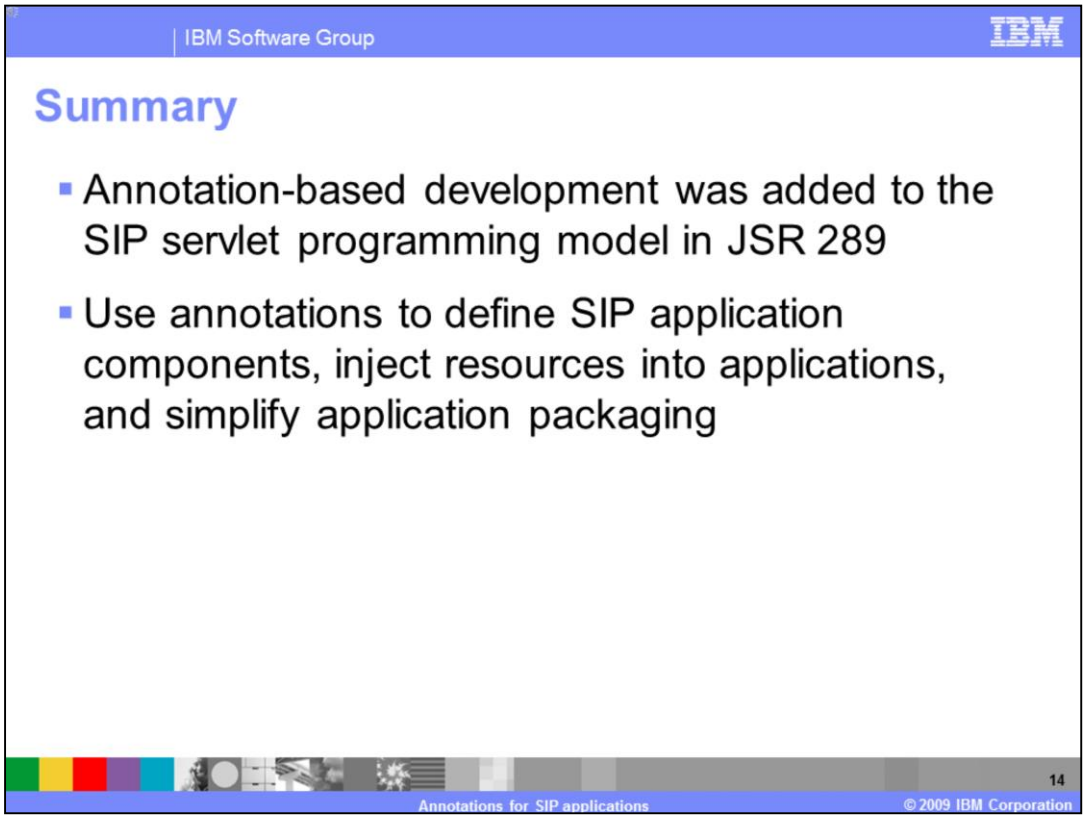
The @SipListener annotation provides an alternative to the listener field in the deployment descriptor. It allows the application developer to specify a listener without declaring it in the deployment descriptor of the application. The listener type is inferred from the interfaces implemented by the target class. The @SipListener annotation does not have any required fields, but can take an optional application name and description. The class annotated by @SipListener must implement at least one of the listener interfaces described in JSR 289.



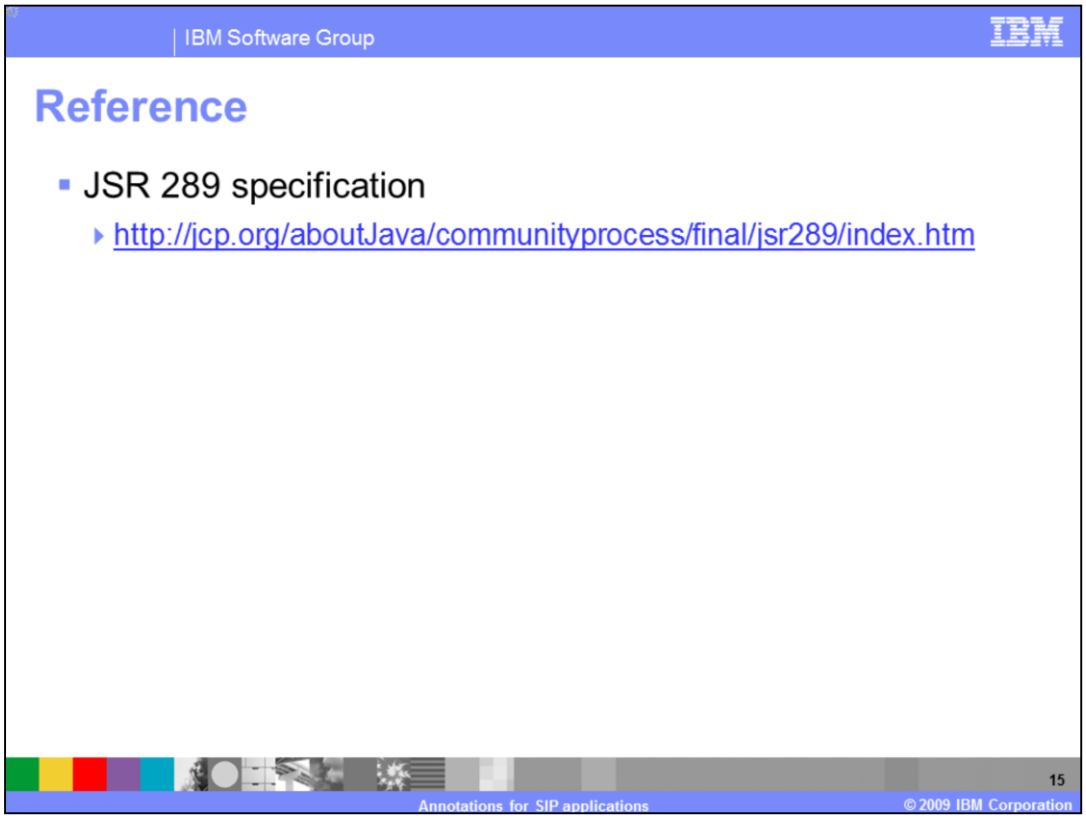
The @Resource annotation can be used to inject an instance of a SipFactory, without having to do ServletContext lookup for the SipFactory from within a servlet. The two code snippets shown on this slide illustrate the previous method of creating a SipFactory instance, and the new method for creating the factory using resource injection with annotations. The two snippets are functionally equivalent. The @Resource annotation can be used to inject an instance of the SipFactory in any SIP servlet or in a Java EE application component deployed on a converged container in the same EAR file. Other resources, like Enterprise JavaBeans, can also be instantiated using resource injection, using the same syntax available in the Java EE 5 specification.



This section contains a summary and reference.



Annotations are a key component of many Java-based programming models, including Java EE 5, and an annotation-based programming model has been added to the SIP servlet specification in Version 1.1 (JSR 289). Annotations help simplify application development by making it easy to define application components, inject resources including SIP resources and other Java resources directly into applications, and simplifying the application packaging model by eliminating the need for deployment descriptors in some cases.



The JSR 289 specification document provides comprehensive information on all of the new annotations that are a part of the SIP servlet 1.1 standard.

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:

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

Enterprise JavaBeans, Java, 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.

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.