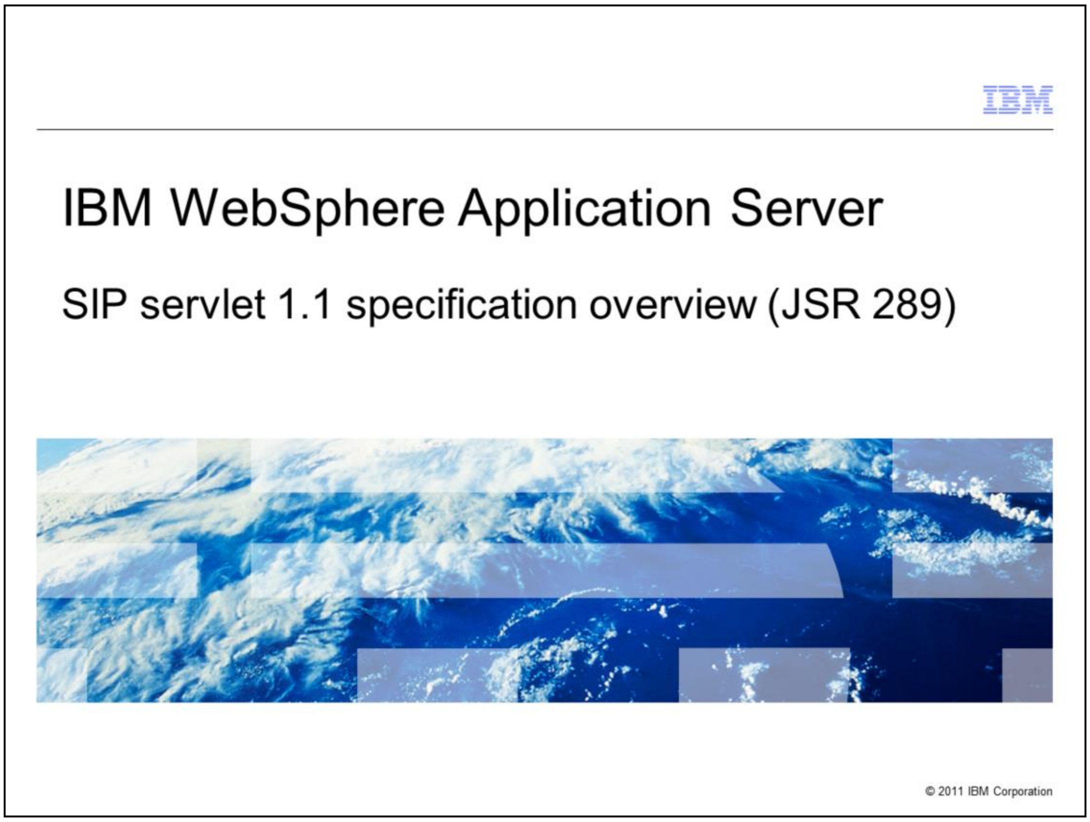
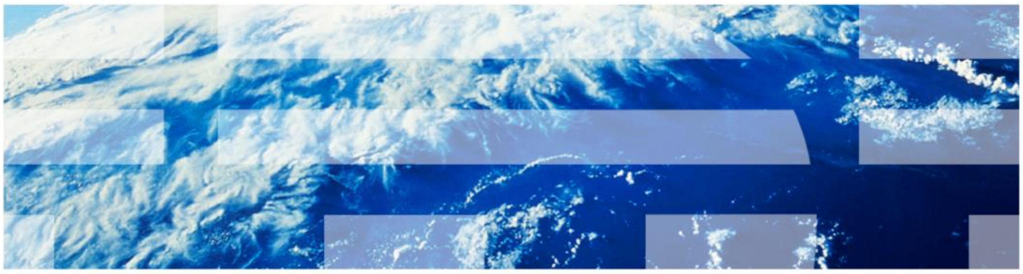
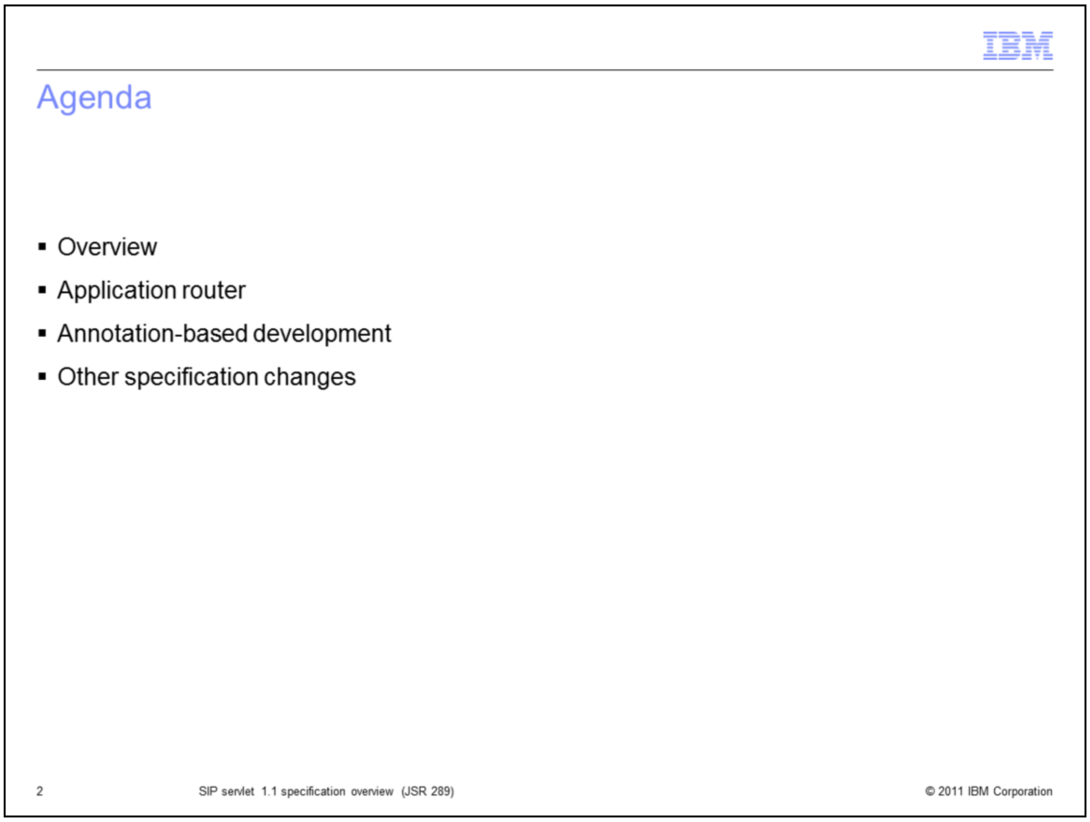


IBM WebSphere Application Server

SIP servlet 1.1 specification overview (JSR 289)



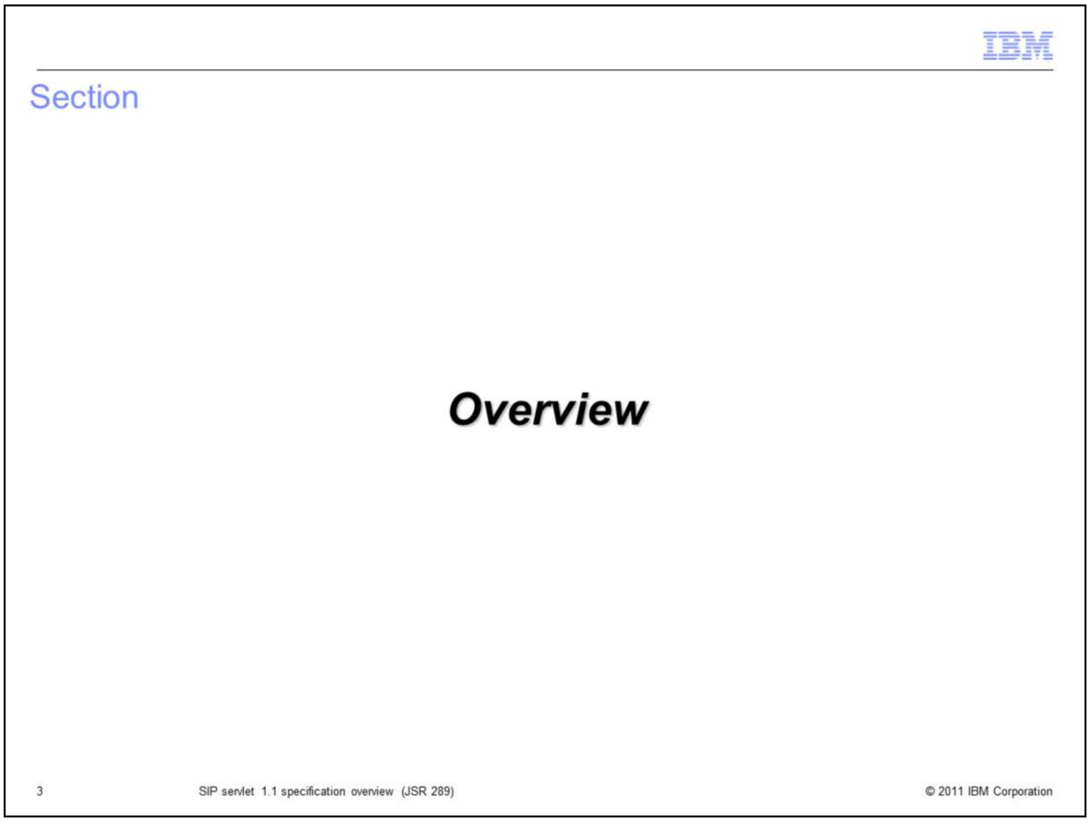
This presentation provides an overview of new features in the JSR 289 specification for SIP servlet 1.1. IBM WebSphere Application Server is compliant with the JSR 289 specification.


Agenda

- Overview
- Application router
- Annotation-based development
- Other specification changes

2 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

This presentation begins by providing an overview of the goals of the SIP servlet 1.1 specification, and then discusses some of the new key features included in the specification, starting with the application router. The application router provides a flexible mechanism for grouping SIP application components together to provide end-to-end services. The programming model for SIP servlet 1.1, like many other current Java specifications, includes annotations to speed up development and simplify application structure and packaging. Other updates in the JSR 289 specification include improved support for converged applications that contain SIP components and other Java EE components, and a B2buaHelper API that simplifies the process for developing the common B2BUA model for SIP applications.

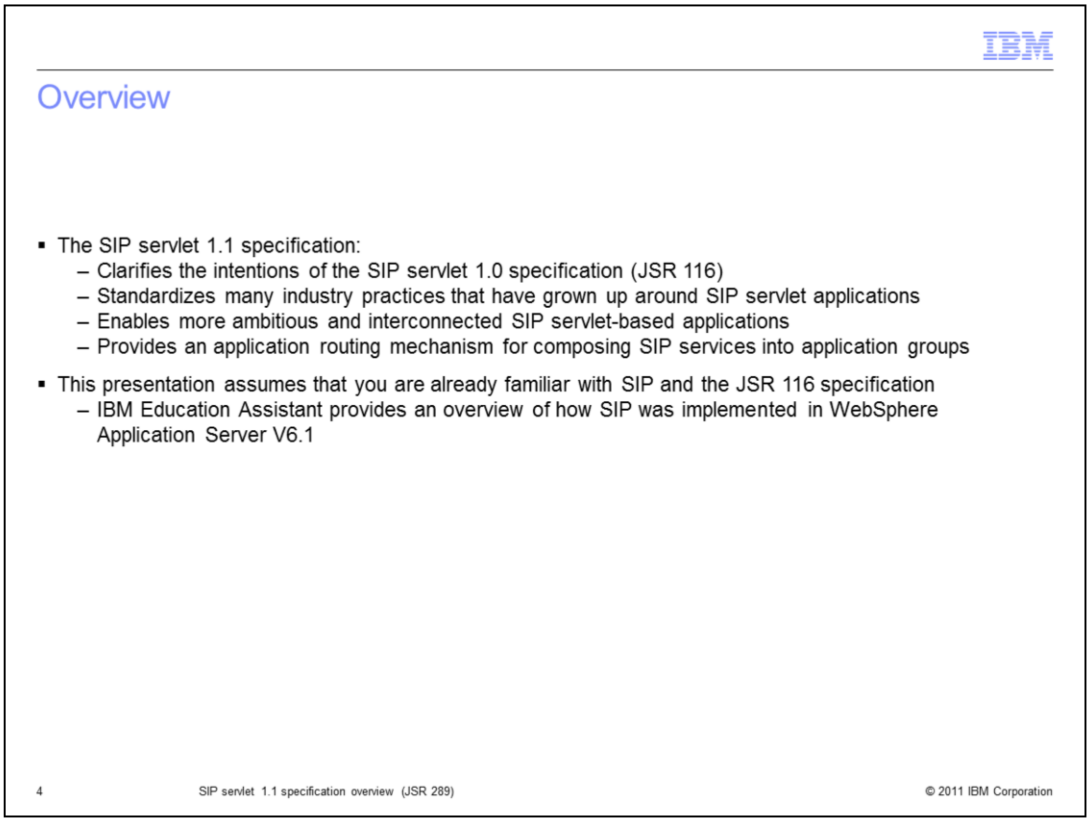


Section

Overview

3 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

This section provides an overview of the goals of the SIP servlet 1.1 specification.



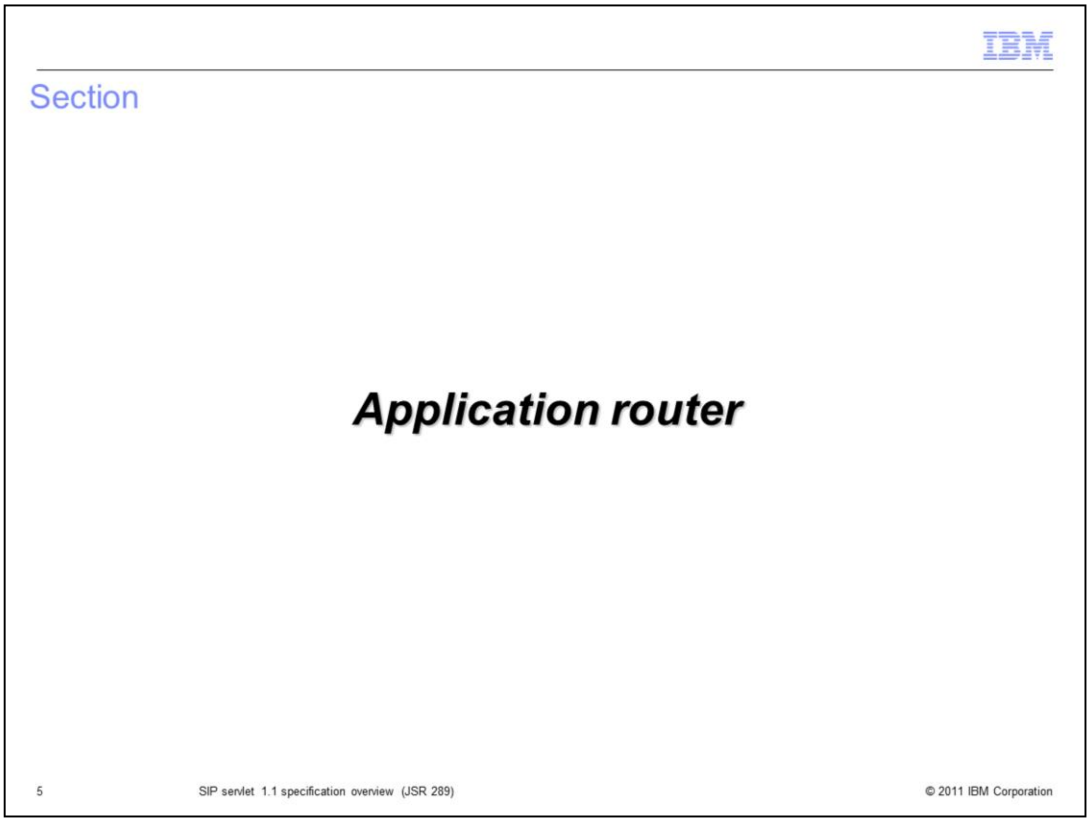
Overview

- The SIP servlet 1.1 specification:
  - Clarifies the intentions of the SIP servlet 1.0 specification (JSR 116)
  - Standardizes many industry practices that have grown up around SIP servlet applications
  - Enables more ambitious and interconnected SIP servlet-based applications
  - Provides an application routing mechanism for composing SIP services into application groups
- This presentation assumes that you are already familiar with SIP and the JSR 116 specification
  - IBM Education Assistant provides an overview of how SIP was implemented in WebSphere Application Server V6.1

4 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

The SIP servlet 1.1 specification builds on the previous 1.0 specification by providing several important new features. In the initial specification, there were several behaviors that were not clearly defined. These intended behaviors are more carefully described in the current specification. Many of industry's best practices grew up around SIP servlet applications beyond what was covered in the 1.0 specification, and the 1.1 specification attempts to codify many of these best practices. The SIP servlet 1.1 specification also enables developers to create more ambitious and interconnected SIP servlet-based applications, including applications that incorporate both SIP components and other Java EE components, like HTTP servlets and Enterprise JavaBeans.

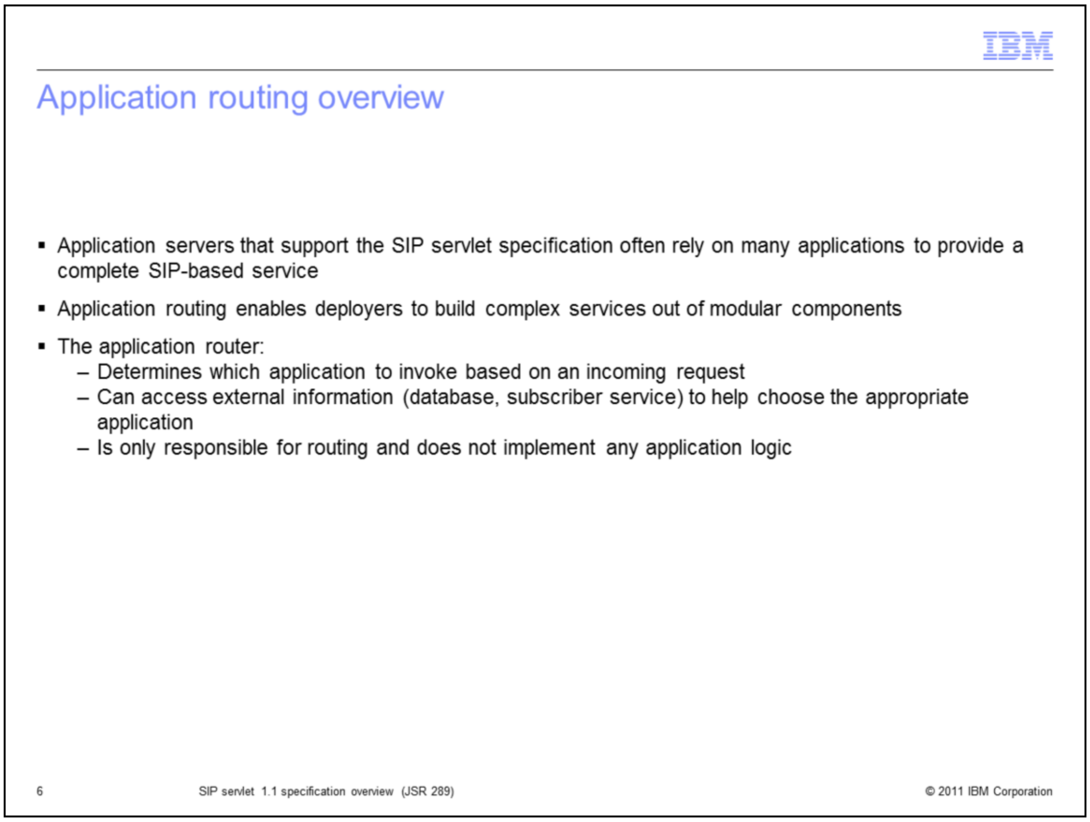
This presentation does not cover the session initiation protocol (SIP) or the SIP servlet 1.0 specification and assumes that the student already has a basic understanding of SIP. For those interested in a SIP refresher, see the Reference section at the end of this module for a link to SIP overview presentation that describes how the SIP container was initially implemented in the application server.



Section

Application router

This section of the presentation provides an overview of the new application router component that is a part of the JSR 289 specification.



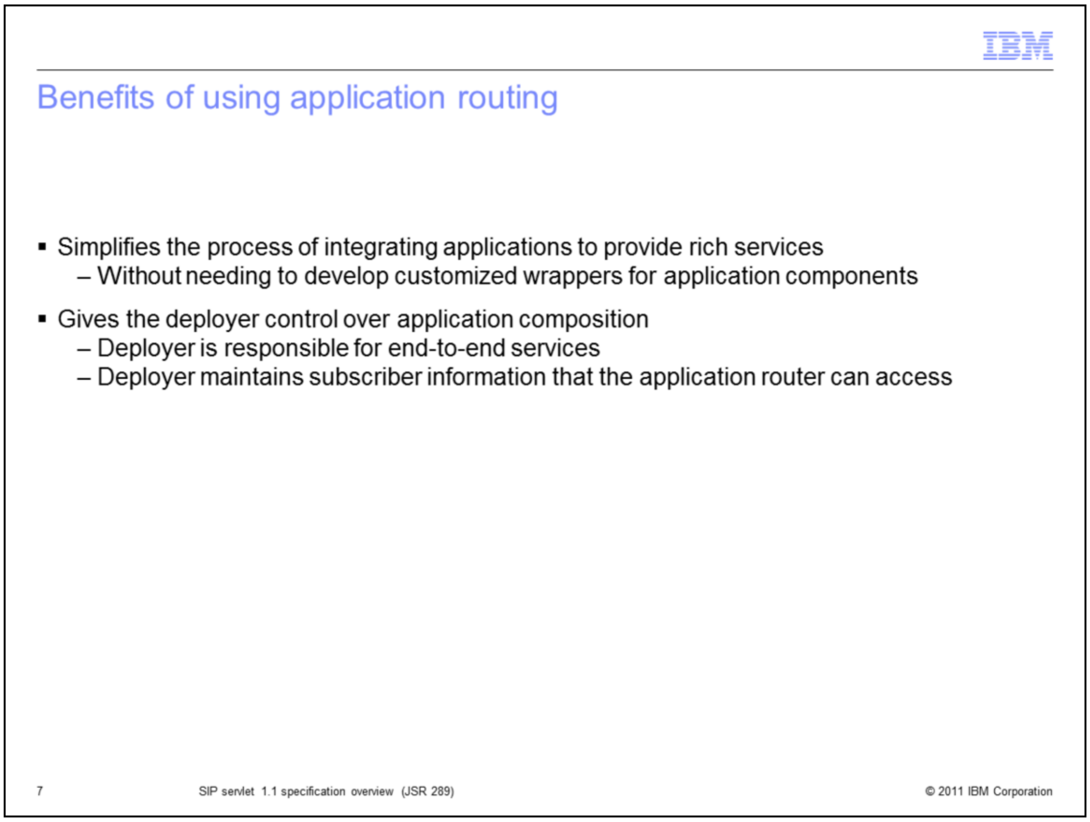
Application routing overview

- Application servers that support the SIP servlet specification often rely on many applications to provide a complete SIP-based service
- Application routing enables deployers to build complex services out of modular components
- The application router:
  - Determines which application to invoke based on an incoming request
  - Can access external information (database, subscriber service) to help choose the appropriate application
  - Is only responsible for routing and does not implement any application logic

6 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

SIP servlet application servers are typically provisioned with many different applications. Each application provides specific functionality, but, by invoking multiple applications to service a call, the deployer can build a complex and complete service. This modular and compositional approach makes it easier for application developers to develop new applications and for the deployer to combine applications from different sources and manage feature interaction. A typical example from traditional telephony is a call-screening application and a call-forwarding application. If the application server receives an incoming INVITE destined to a callee who subscribes to both services, both applications should be invoked.

The application router is a separate component, outside of the SIP container. The container receives initial requests, calls the application router to determine which application to invoke, and then the container calls that application. Once the container has called into an application, that application calls into the appropriate servlet to handle the request, based on the application’s configuration – for example, using mappings defined in the application’s deployment descriptor. By default, WebSphere Application Server uses application start-up weights to define the routing order. The JSR 289 specification also defines a Default Application Router (DAR) properties file format and a custom application router application format to describe application routing.

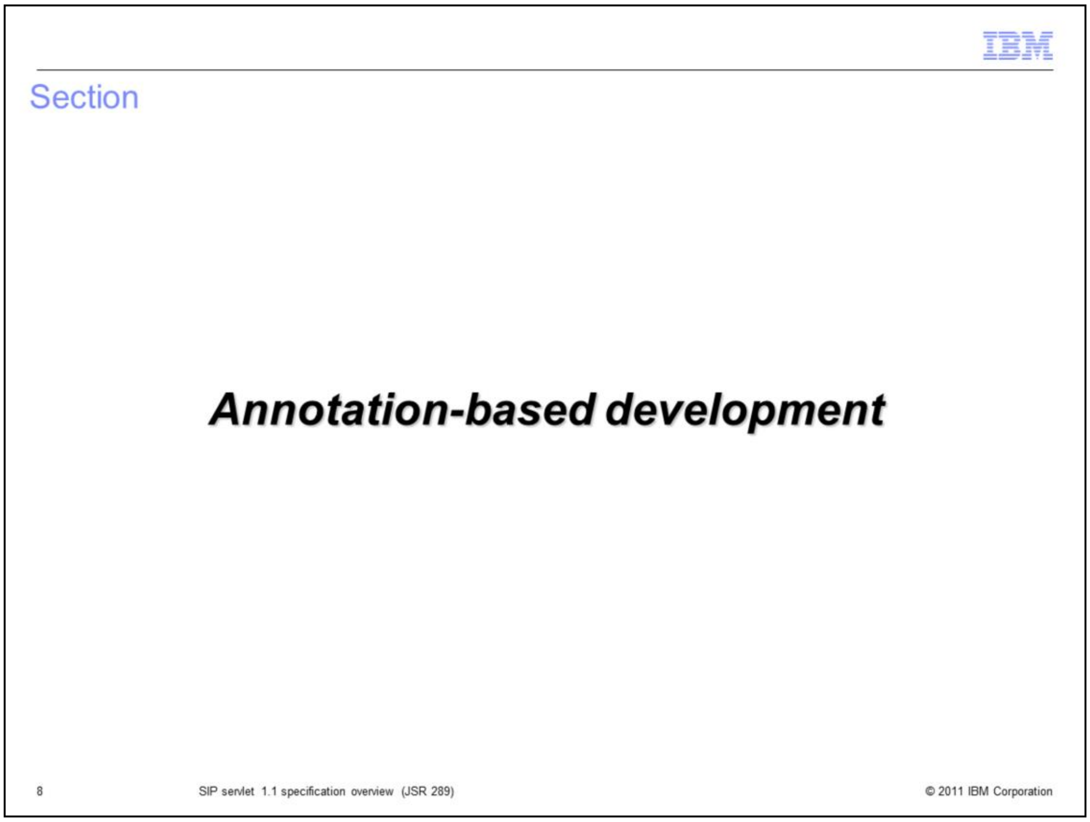


Benefits of using application routing

- Simplifies the process of integrating applications to provide rich services
  - Without needing to develop customized wrappers for application components
- Gives the deployer control over application composition
  - Deployer is responsible for end-to-end services
  - Deployer maintains subscriber information that the application router can access

7 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

The application router makes it easier to buy a vendor application and invoke its services, without having to write custom wrapper code. This gives the deployer control over how the services behave, rather than leaving integration decisions in the hands of the application developer. Say, for example, you provide telephone service to a large number of subscribers, and a law enforcement agency comes to you with a call tracing and monitoring application that you need to run on a specific subset of your subscribers. Previously, this application was invoked for all subscribers and had to include logic to run only on the required subscribers, or you needed to write an application wrapper to determine whether to invoke the application for a particular user. Now, under the SIP servlet 1.1 specification, all of the logic for determining which users require which application services can be moved outside the scope of the application itself and into the application router.

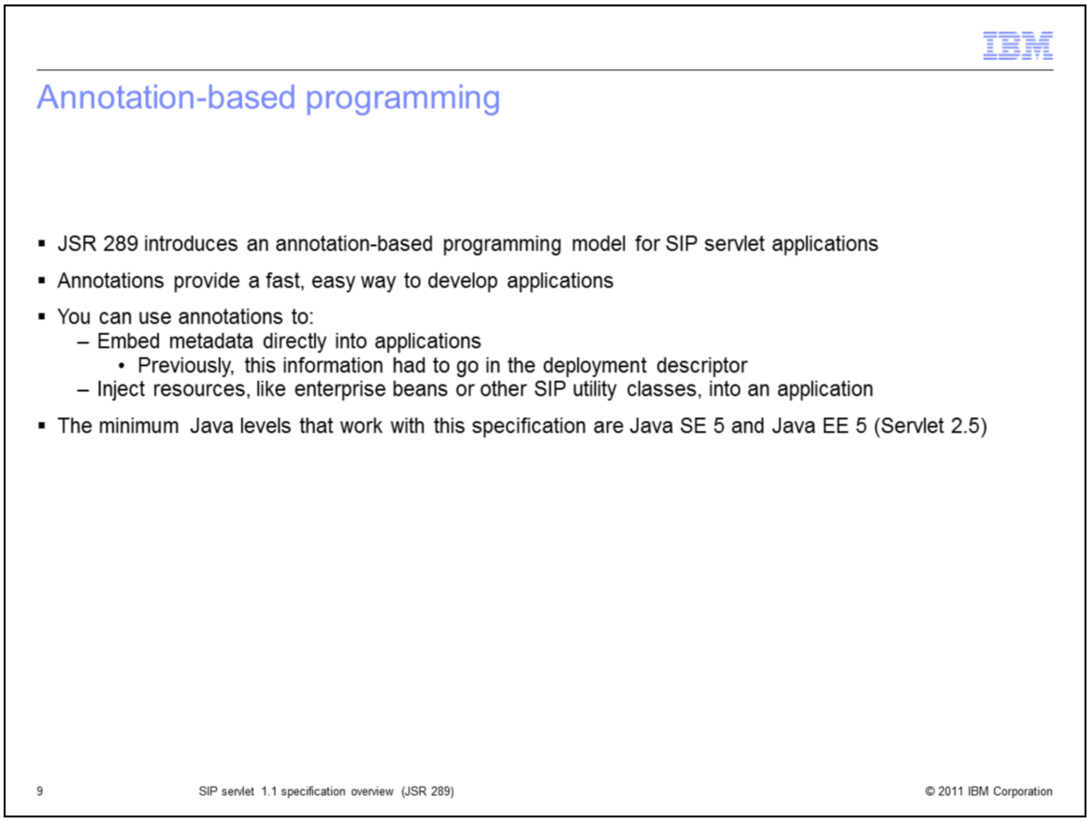


Section

Annotation-based development

8 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

This section provides an overview of the annotation-based programming model introduced in the SIP servlet 1.1 specification.



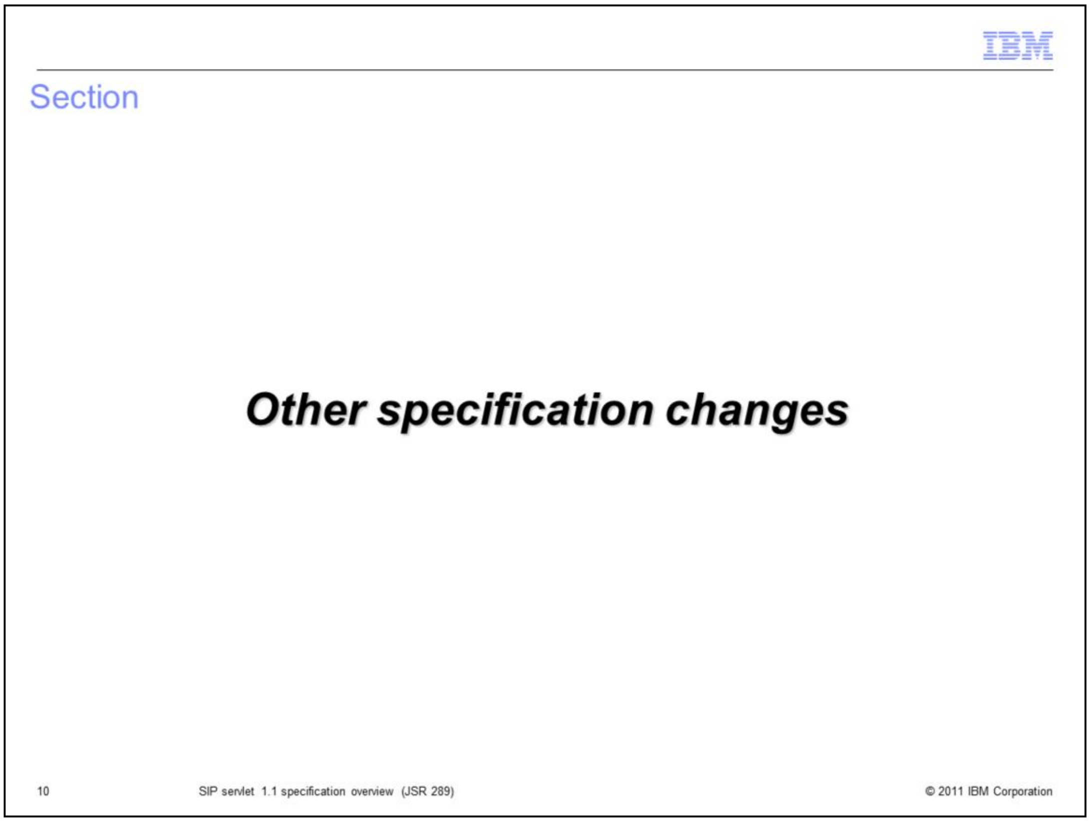
Annotation-based programming

- JSR 289 introduces an annotation-based programming model for SIP servlet applications
- Annotations provide a fast, easy way to develop applications
- You can use annotations to:
  - Embed metadata directly into applications
    - Previously, this information had to go in the deployment descriptor
  - Inject resources, like enterprise beans or other SIP utility classes, into an application
- The minimum Java levels that work with this specification are Java SE 5 and Java EE 5 (Servlet 2.5)

9 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

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.

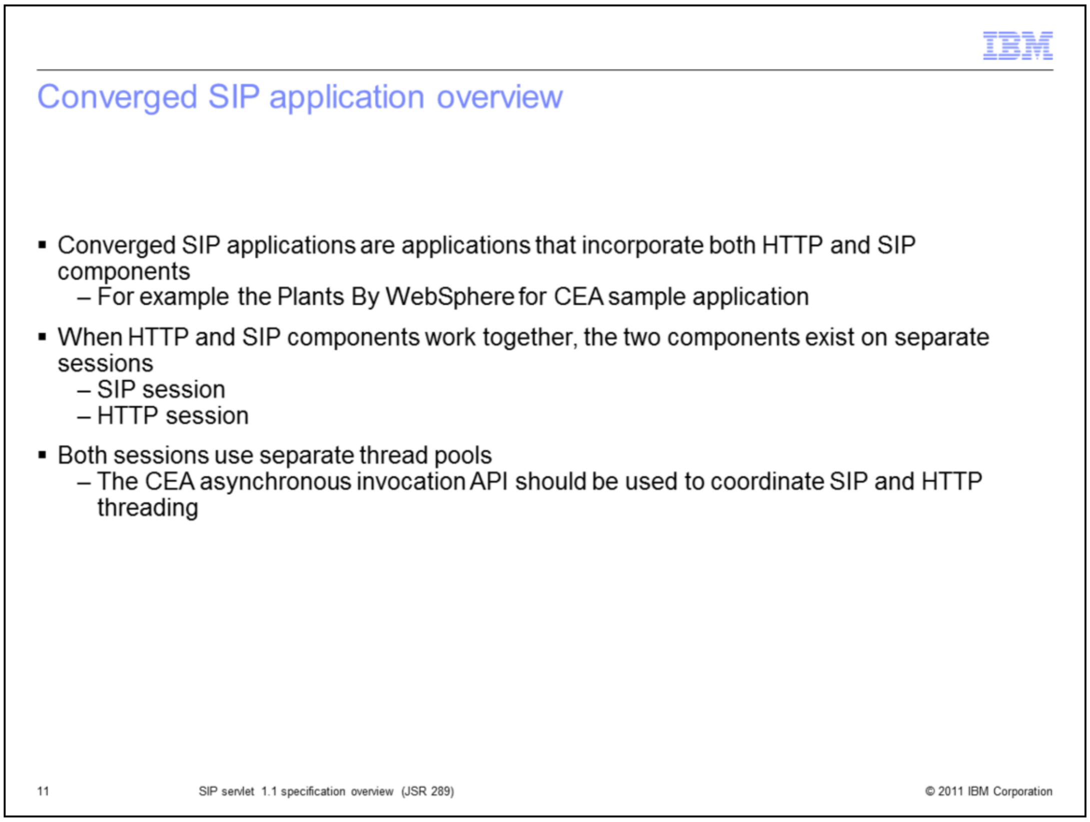
Resource injection is a simplified model for pulling resources, like SIP utility classes or Enterprise JavaBeans, into an enterprise application, and the new SIP servlet annotations in JSR 289 support resource injection. Because of the use of annotations in the SIP servlet 1.1 specification, you must use, at minimum, Java SE 5 or Java EE 5.



Section

Other specification changes

The last section of this presentation covers other JSR 289 updates, including improved support for converged applications and the B2buaHelper APIs.

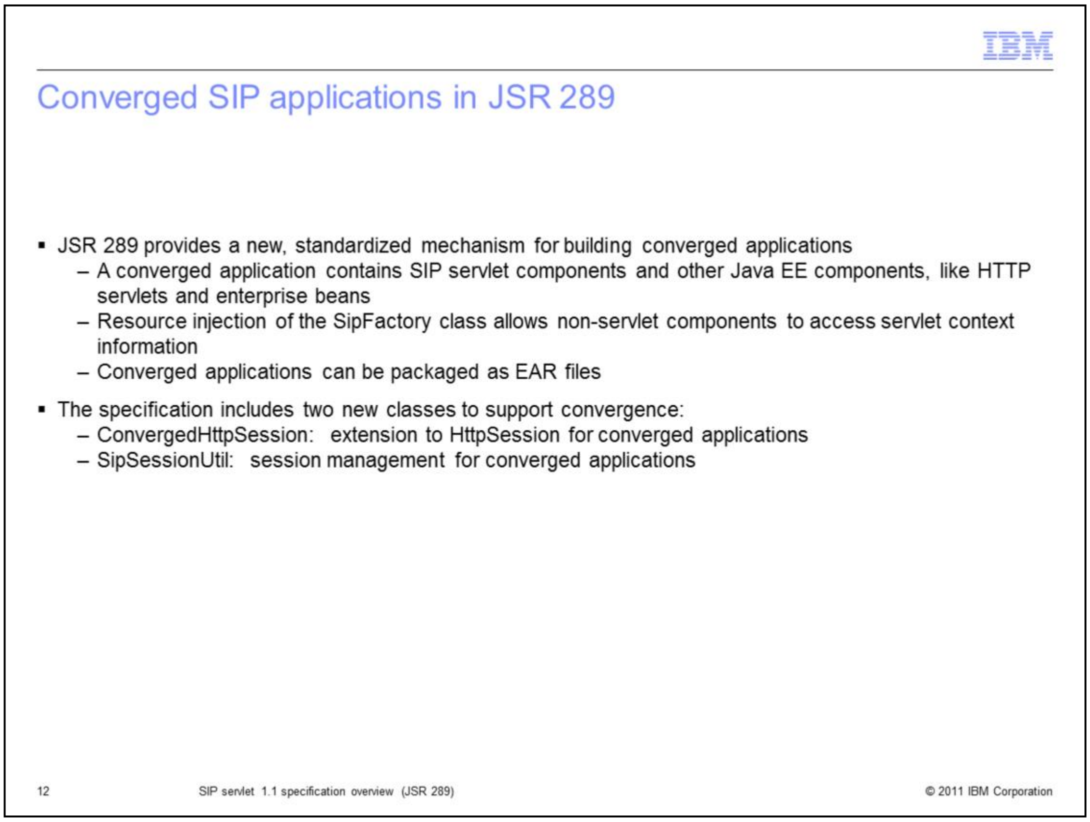

Converged SIP application overview

- Converged SIP applications are applications that incorporate both HTTP and SIP components
  - For example the Plants By WebSphere for CEA sample application
- When HTTP and SIP components work together, the two components exist on separate sessions
  - SIP session
  - HTTP session
- Both sessions use separate thread pools
  - The CEA asynchronous invocation API should be used to coordinate SIP and HTTP threading

11 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

A converged SIP application is an enterprise application that involves some SIP components and some HTTP components. An example of a converged application is the Plants By WebSphere application provided in the Communications Enabled Applications (CEA) samples package the you can download from the WebSphere Application Server Samples site. It has a web based component, which is the sample plants store application, and SIP related components included with the click to call and call notification widgets. The widgets initiate SIP activities while the Plants by WebSphere application initiates SIP related activities by using an HTTP session and Web related pages that host the SIP widgets.

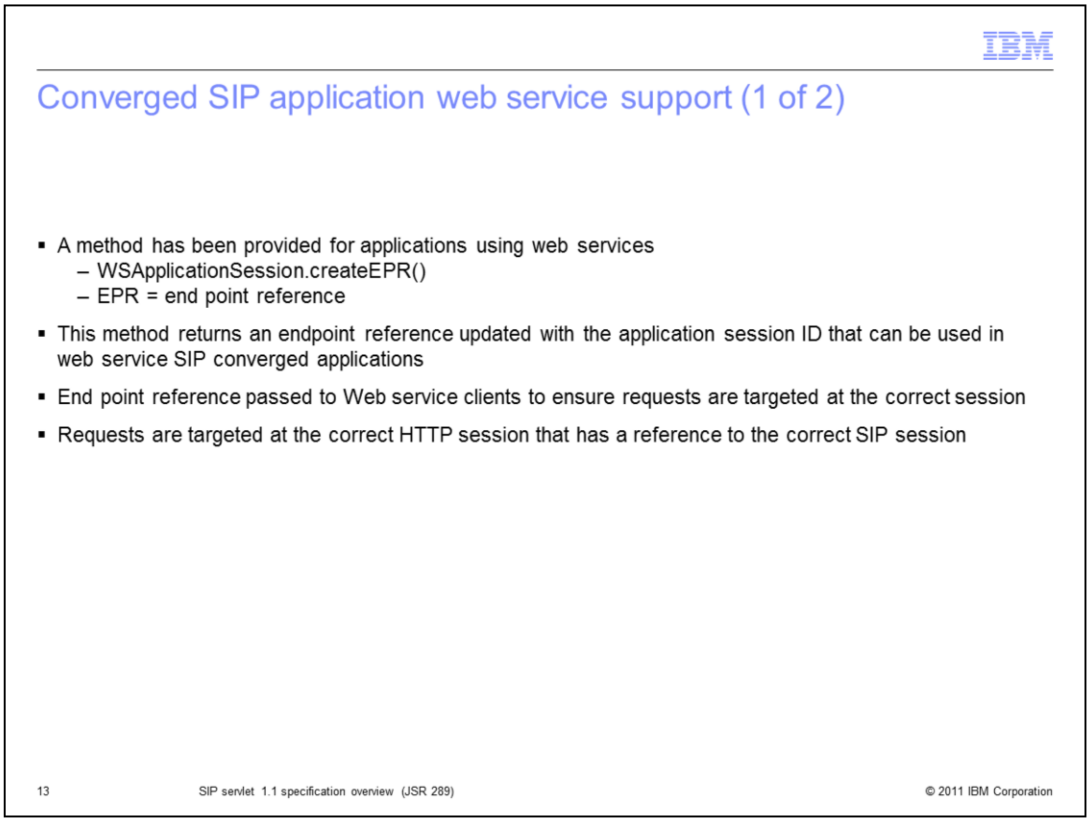
The Web container and the SIP container in IBM WebSphere Application Server each have their own notion of session and affinity. The IBM WebSphere Application Server SIP container also has its own notion of a session and affinity. These two components must come together and cooperate if they are both used in the same application, for example the Plants by WebSphere sample application.


Converged SIP applications in JSR 289

- JSR 289 provides a new, standardized mechanism for building converged applications
  - A converged application contains SIP servlet components and other Java EE components, like HTTP servlets and enterprise beans
  - Resource injection of the SipFactory class allows non-servlet components to access servlet context information
  - Converged applications can be packaged as EAR files
- The specification includes two new classes to support convergence:
  - ConvergedHttpSession: extension to HttpSession for converged applications
  - SipSessionUtil: session management for converged applications

12 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

Converged applications contain both SIP servlet components and other Java EE components, like HTTP servlets and Enterprise JavaBeans. The ability to use annotations for resource injection rather than relying purely on ServletContext lookup allows non-servlet components to access information from the SipFactory. IBM supported application convergence in WebSphere Application Server V6.1 using proprietary APIs, and now this convergence model has become the standard in JSR 289. The IBM APIs are still supported, but the recommendation is to move to the new standardized APIs that are a part of the SIP servlet 1.1 specification. The two new classes to support convergence in JSR 289 are the ConvergedHttpSession, which is an extension to HttpSession for converged applications, and the SipSessionUtil class, which provides session management capability for converged applications.

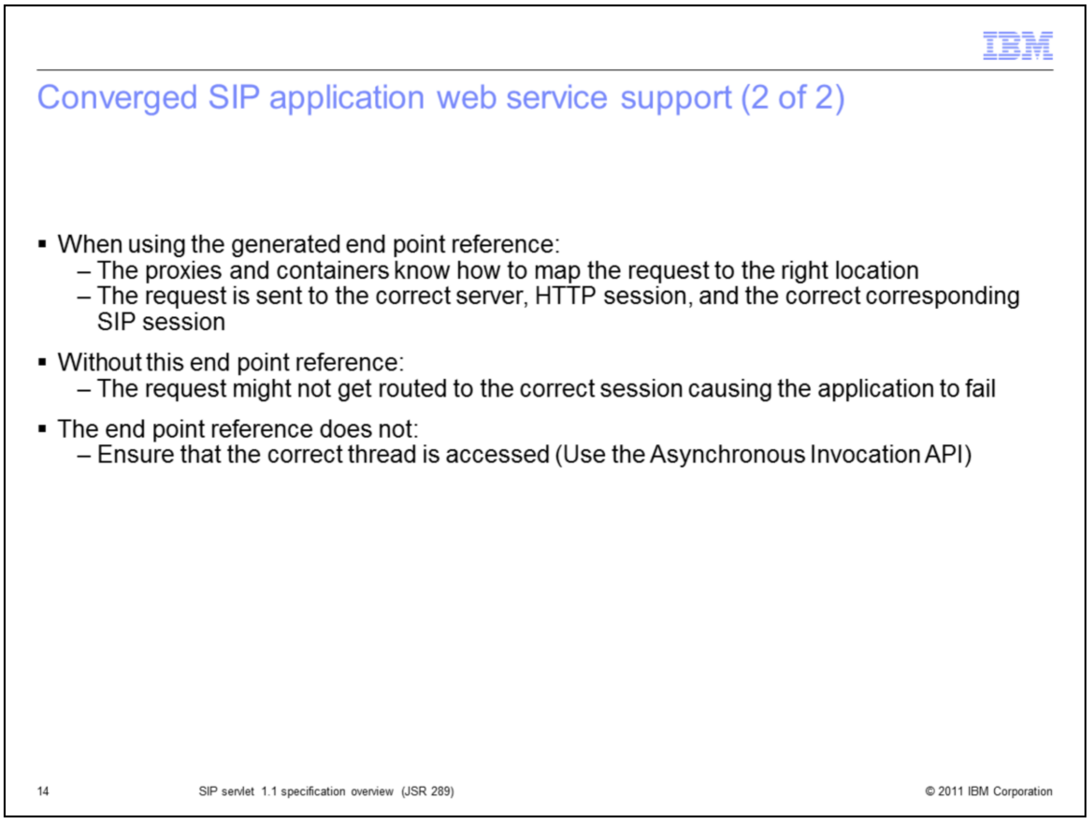


Converged SIP application web service support (1 of 2)

- A method has been provided for applications using web services
  - WSApplicationSession.createEPR()
  - EPR = end point reference
- This method returns an endpoint reference updated with the application session ID that can be used in web service SIP converged applications
- End point reference passed to Web service clients to ensure requests are targeted at the correct session
- Requests are targeted at the correct HTTP session that has a reference to the correct SIP session

13 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

SIP requests and HTTP requests associated with the same user instance need to go to the same place. Proxies performing routing will not know where to send requests from web service clients to a converged application after a CEA web service session is established. After a session is established Web service client calls in a converged SIP application can land on the wrong server, and the application can fail. The new method createEPR on the WSApplicationSession class solves this problem. Web service applications that call this method are returned an end point reference. Web service clients calling the application will then use the end point reference to correspond with the service hosted on the application server. When clients use the end point reference, it will ensure that the calls made to the service are routed to the appropriate HTTP and SIP sessions. Without the end point reference, there is no guarantee that the Web service client calls will make it to the correct HTTP and SIP sessions after the session is established.

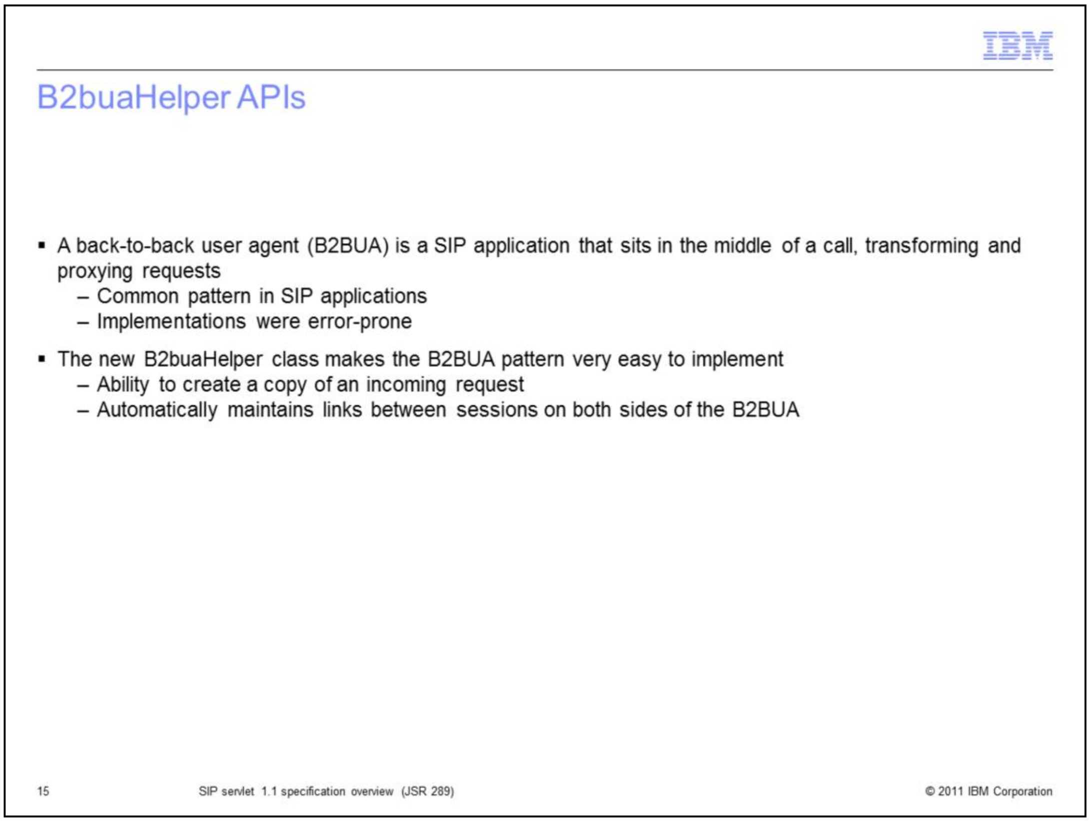


Converged SIP application web service support (2 of 2)

- When using the generated end point reference:
  - The proxies and containers know how to map the request to the right location
  - The request is sent to the correct server, HTTP session, and the correct corresponding SIP session
- Without this end point reference:
  - The request might not get routed to the correct session causing the application to fail
- The end point reference does not:
  - Ensure that the correct thread is accessed (Use the Asynchronous Invocation API)

14 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

web service clients that use the end point reference generated by the WSApplicationSession.createEPR() method will send the request to the correct server and the correct HTTP and SIP sessions. Server proxies and containers know how to map the request from the end point reference to the correct location. The end point reference contains an affinity key used by proxies to determine the server and HTTP session to target the incoming request for the web service. One important thing to note is that this feature does not help with threading. Application developers must still pay close attention to threading when working with a converged application to ensure that the HTTP and SIP sessions are not accessing common objects simultaneously and therefore corrupting object states. The Asynchronous Invocation API should be used in converged SIP applications to ensure that common objects are accessed correctly by the HTTP and SIP threads.

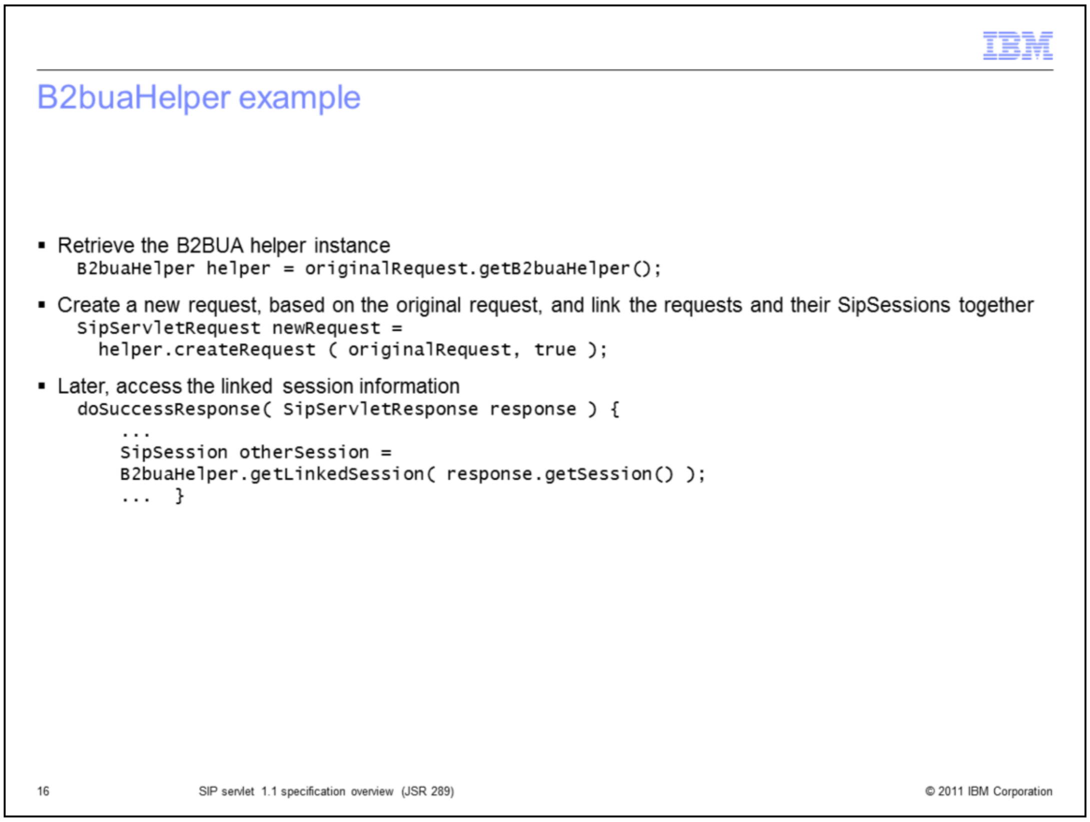


B2buaHelper APIs

- A back-to-back user agent (B2BUA) is a SIP application that sits in the middle of a call, transforming and proxying requests
  - Common pattern in SIP applications
  - Implementations were error-prone
- The new B2buaHelper class makes the B2BUA pattern very easy to implement
  - Ability to create a copy of an incoming request
  - Automatically maintains links between sessions on both sides of the B2BUA

15 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

A back-to-back user agent, or B2BUA, is a common pattern in SIP applications. The B2BUA inserts itself into the path of the request by taking in the request, then acting as a user agent server, or UAS, to perform some operation or transformation on the request, and then acting as a user agent client, or UAC, and sending the request on. Previously, the B2BUA had to clone many requests and responses passing through it and make sure that the requests and responses got mapped appropriately back and forth across the call. Implementations of the request mappings were often complicated and error prone. The new B2buaHelper class makes the B2BUA pattern very easy to implement by providing a mechanism to create a copy of an incoming request and automatically maintaining links between sessions on both sides of the call.



B2buaHelper example

- Retrieve the B2BUA helper instance

```
B2buaHelper helper = originalRequest.getB2buaHelper();
```

- Create a new request, based on the original request, and link the requests and their SipSessions together

```
SipServletRequest newRequest =
  helper.createRequest ( originalRequest, true );
```

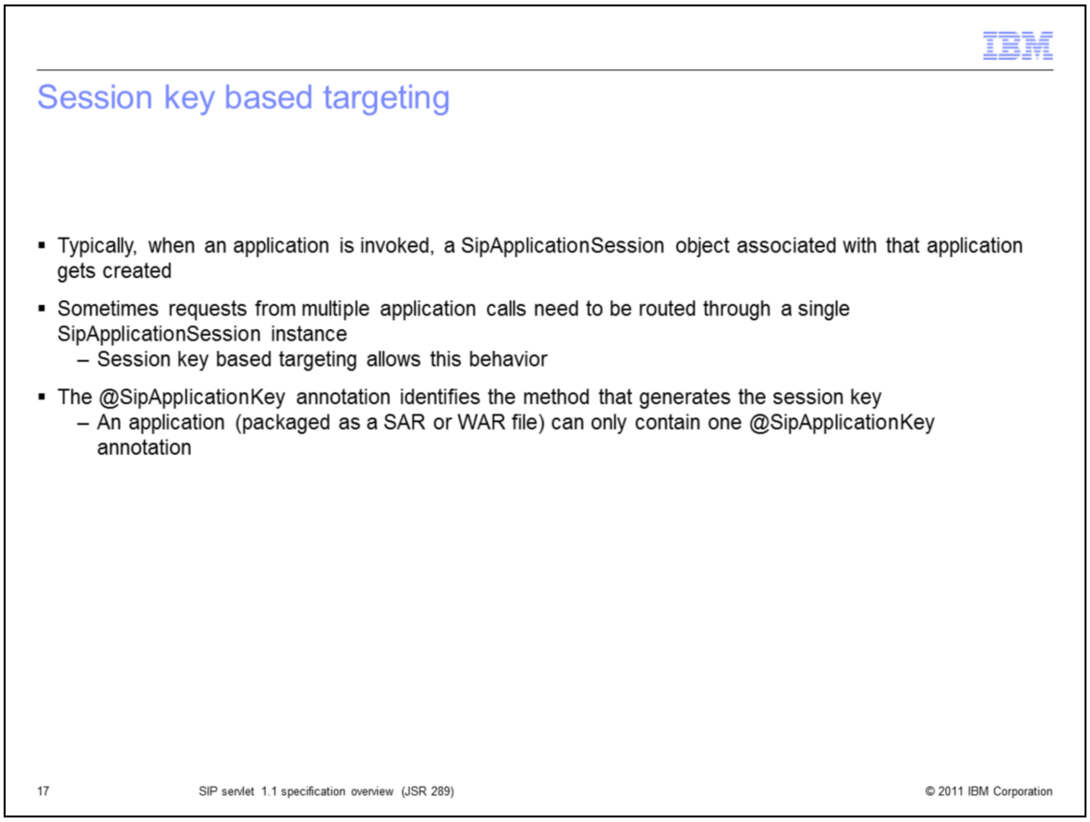
- Later, access the linked session information

```
doSuccessResponse( SipServletResponse response ) {
    ...
    SipSession otherSession =
    B2buaHelper.getLinkedSession( response.getSession() );
    ...  }
```

16 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

The B2buaHelper class instance can be retrieved from a SipServletRequest by invoking the getB2buaHelper() method on it. By making that method call, that indicates to the container that the application is acting as a B2BUA. From that point on, any user agent operation is permitted by the application, but the application can no longer act as a Proxy.

When an application receives an initial request for which it wants to act as a B2BUA, it can invoke the createRequest() method on the B2buaHelper class. This method returns a request that is identical to the one provided as an argument, with the appropriate header fields copied across. By passing in the second argument to the createRequest method as true, the SipSessions are linked together for the original and new SipServletRequests. By linking the sessions together, you might be able to navigate from one to the other. One common function of a B2BUA is to forward requests and responses from one SipSession to another, after performing some transformation or application of business logic. Using linked sessions under the B2buaHelper API, as shown here, simplifies that pattern.

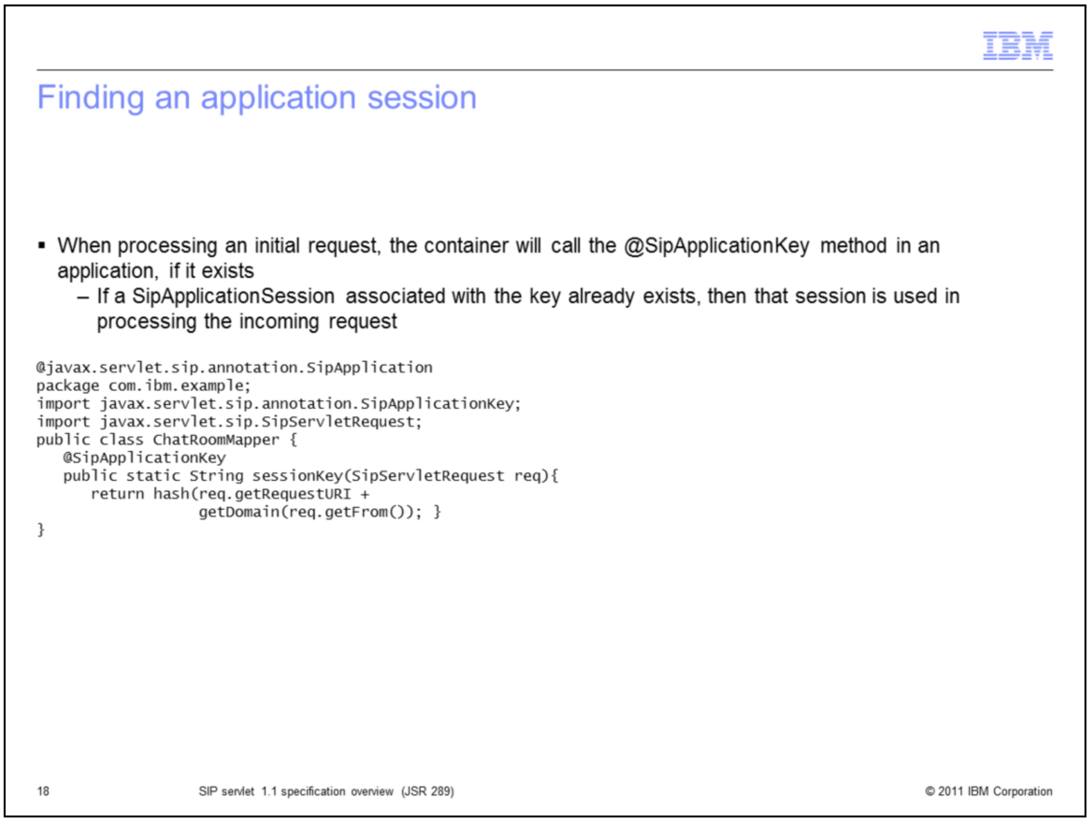


Session key based targeting

- Typically, when an application is invoked, a SipApplicationSession object associated with that application gets created
- Sometimes requests from multiple application calls need to be routed through a single SipApplicationSession instance
  - Session key based targeting allows this behavior
- The @SipApplicationKey annotation identifies the method that generates the session key
  - An application (packaged as a SAR or WAR file) can only contain one @SipApplicationKey annotation

17 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

Typically, when an application is invoked, a new SipAplicationSession object gets created and associated with that application. However, sometimes it is required to route all requests for a subscriber, application, or some other combination of factors to a single SipApplicationSession instance. For example, consider a call waiting application. Say that Alice subscribes to the call waiting service, and she's on the telephone, talking with Bob. During the call, Alice's mother tries to call her, so the call waiting application is invoked to handle the request. The call waiting application should have a way to indicate its need to associate with the existing SipApplicationSession for Alice's current call. It's possible to create such an association using a SipApplicationKey. For an application to use session key based targeting, it needs to have one method identified by the @SipApplicationKey annotation that it responsible for generating the session key. Each SipApplicationSession can only be referred to by a single key.

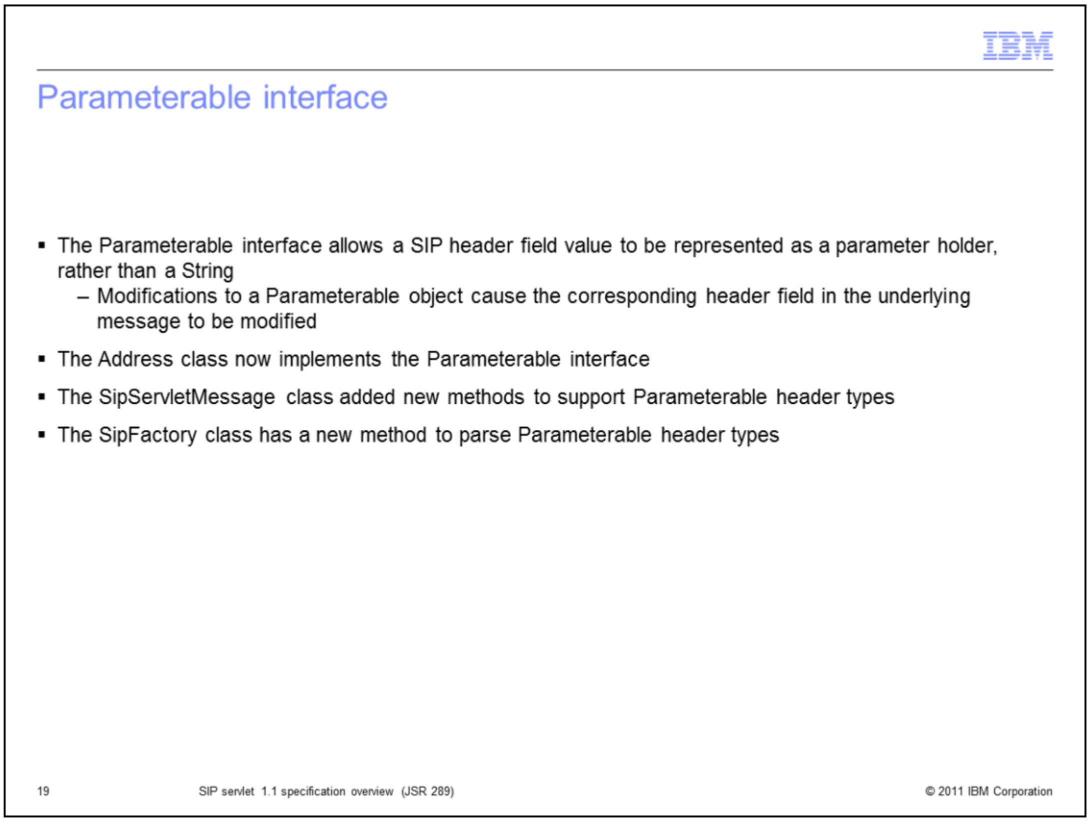

Finding an application session

- When processing an initial request, the container will call the @SipApplicationKey method in an application, if it exists
  - If a SipApplicationSession associated with the key already exists, then that session is used in processing the incoming request

```
@javax.servlet.sip.annotation.SipApplication
package com.ibm.example;
import javax.servlet.sip.annotation.SipApplicationKey;
import javax.servlet.sip.SipServletRequest;
public class ChatRoomMapper {
   @SipApplicationKey
   public static String sessionKey(SipServletRequest req){
      return hash(req.getRequestURI +
                  getDomain(req.getFrom()); }
}
```

18 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

When processing an initial request, the container will call the @SipApplicationKey method in an application, if such a method exists. This method takes as a parameter the incoming SipServletRequest, which is used to generate the key. The example here shows a method that has been defined to create an application session key. The method must be a public static method, returning a String, and it cannot modify the incoming SipServletRequest. If the container finds an application session already associated with a particular key, then that session is used in processing the incoming request.

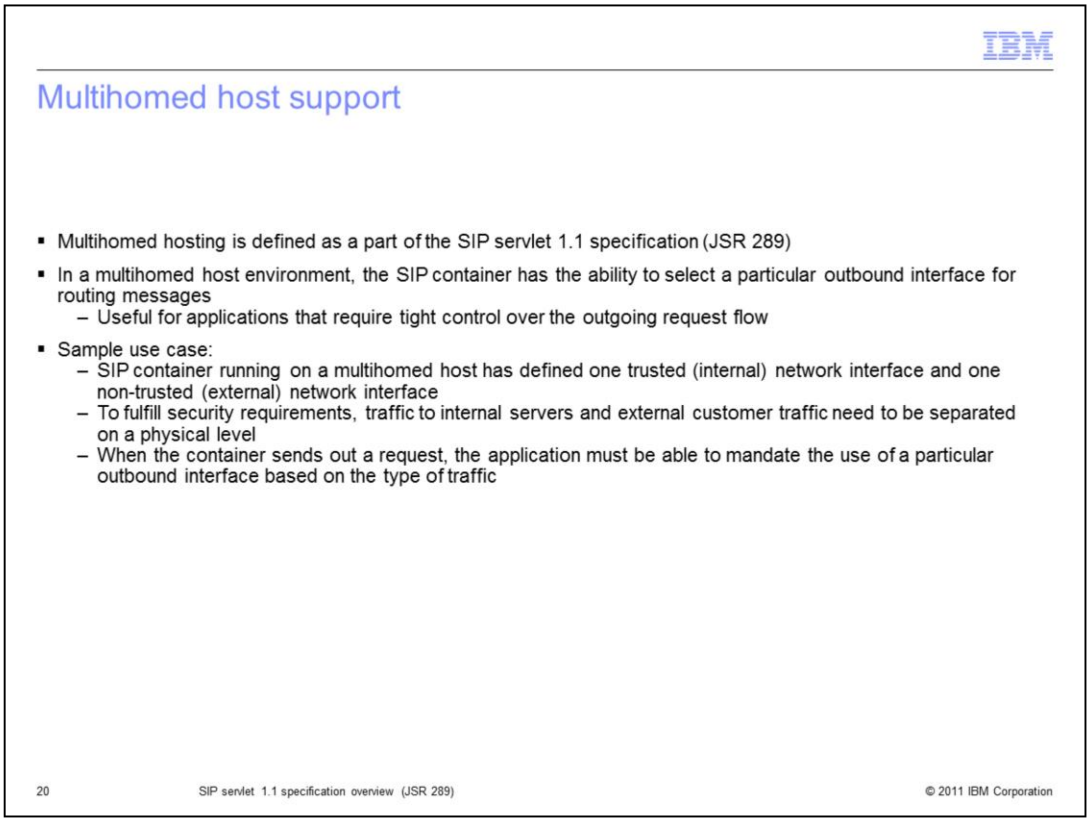


Parameterable interface

- The Parameterable interface allows a SIP header field value to be represented as a parameter holder, rather than a String
  - Modifications to a Parameterable object cause the corresponding header field in the underlying message to be modified
- The Address class now implements the Parameterable interface
- The SipServletMessage class added new methods to support Parameterable header types
- The SipFactory class has a new method to parse Parameterable header types

19 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

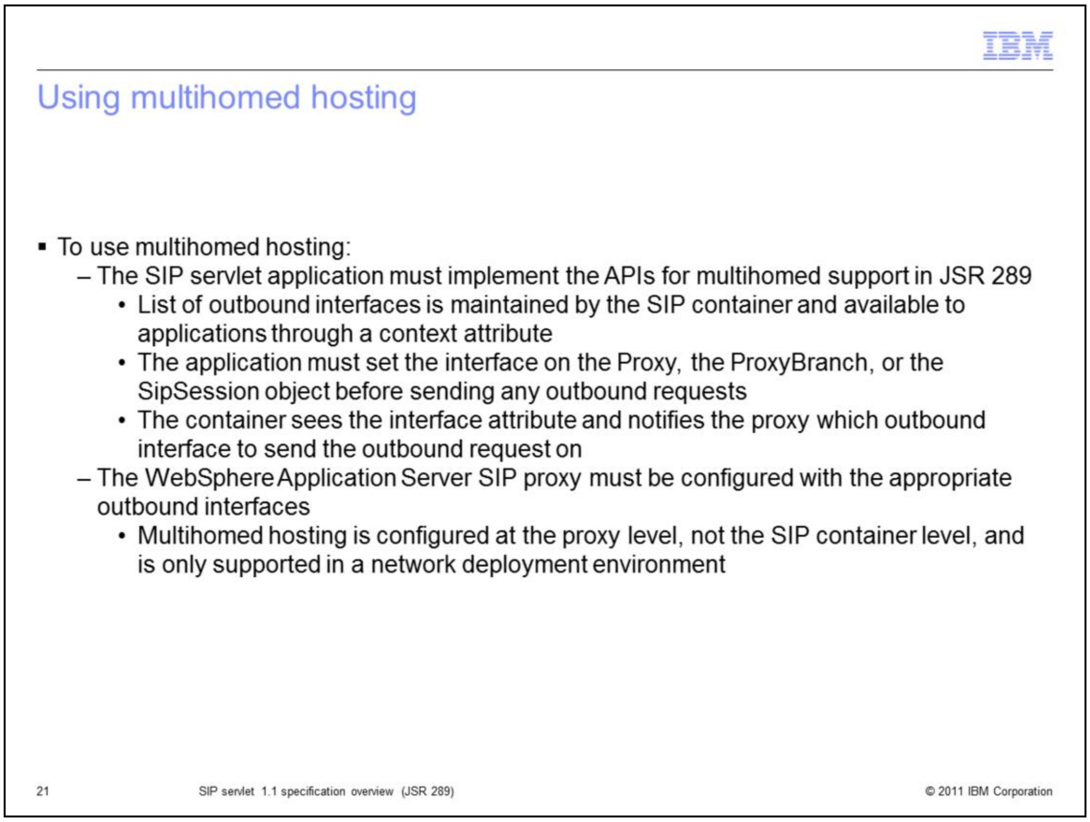
The Parameterable interface allows a SIP header field value to be represented as a parameter, rather than as a String. Having the ability to access Parameterable fields in a parsed form is more convenient and allows for better performance than accessing those header fields directly as Strings. Modifying a Parameterable object causes the corresponding header field in the underlying object to be modified. The Address class now implements the Parameterable interface, and the SipServletMessage and SipFactory classes have new methods to support Parameterable types.


Multihomed host support

- Multihomed hosting is defined as a part of the SIP servlet 1.1 specification (JSR 289)
- In a multihomed host environment, the SIP container has the ability to select a particular outbound interface for routing messages
  - Useful for applications that require tight control over the outgoing request flow
- Sample use case:
  - SIP container running on a multihomed host has defined one trusted (internal) network interface and one non-trusted (external) network interface
  - To fulfill security requirements, traffic to internal servers and external customer traffic need to be separated on a physical level
  - When the container sends out a request, the application must be able to mandate the use of a particular outbound interface based on the type of traffic

20 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

Multihomed hosting is defined as a part of the SIP servlet 1.1 specification, JSR 289. In a multihomed host environment, the SIP container has the ability to select a particular outbound interface for routing messages. This is useful for applications that require tight control over the outgoing request flow. For example, consider a topology in which the SIP container running on a multihomed host has defined one trusted network interface and one non-trusted network interface. The trusted interface is for the internal network, and the non-trusted interface is for the external, or customer-facing, network. To fulfill security requirements, traffic to internal servers must be separated on a physical level from external customer traffic. In this context, when the SIP container sends out a request, the application must be able to mandate the use of a particular outbound interface based on the type of traffic. Using the new multihomed hosting APIs, the application can be written to do just that.

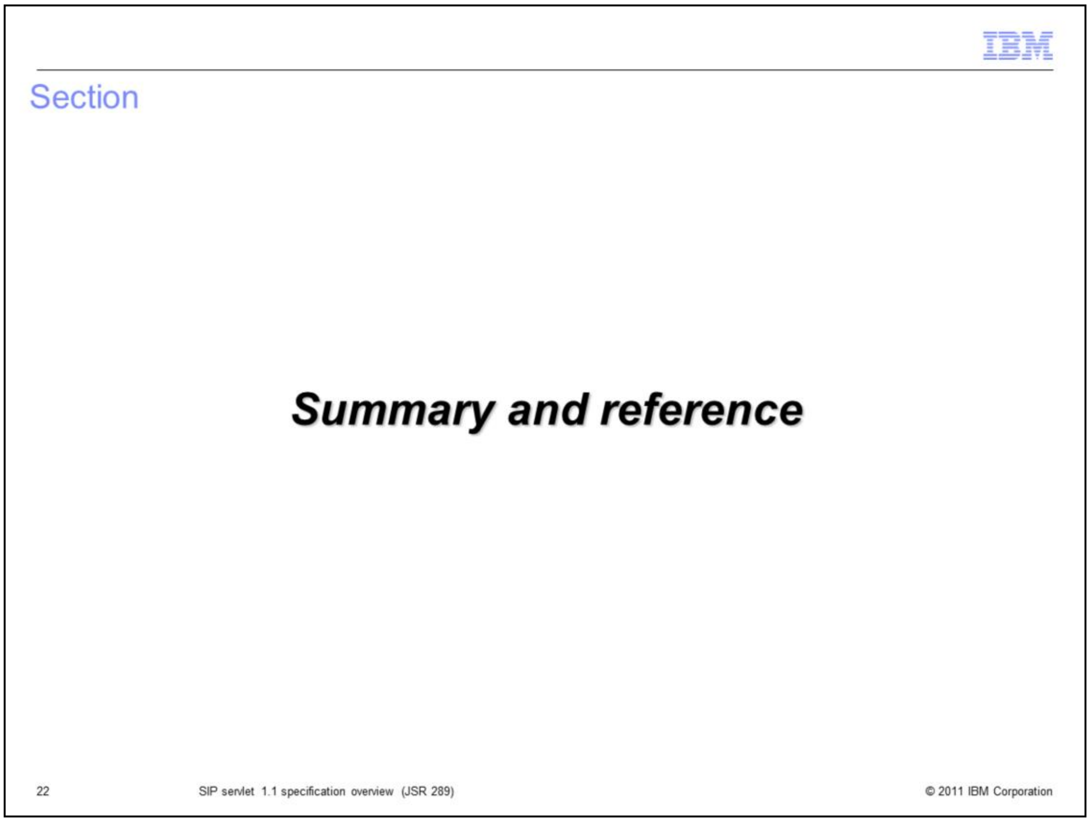


Using multihomed hosting

- To use multihomed hosting:
  - The SIP servlet application must implement the APIs for multihomed support in JSR 289
    - List of outbound interfaces is maintained by the SIP container and available to applications through a context attribute
    - The application must set the interface on the Proxy, the ProxyBranch, or the SipSession object before sending any outbound requests
    - The container sees the interface attribute and notifies the proxy which outbound interface to send the outbound request on
  - The WebSphere Application Server SIP proxy must be configured with the appropriate outbound interfaces
    - Multihomed hosting is configured at the proxy level, not the SIP container level, and is only supported in a network deployment environment

21 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

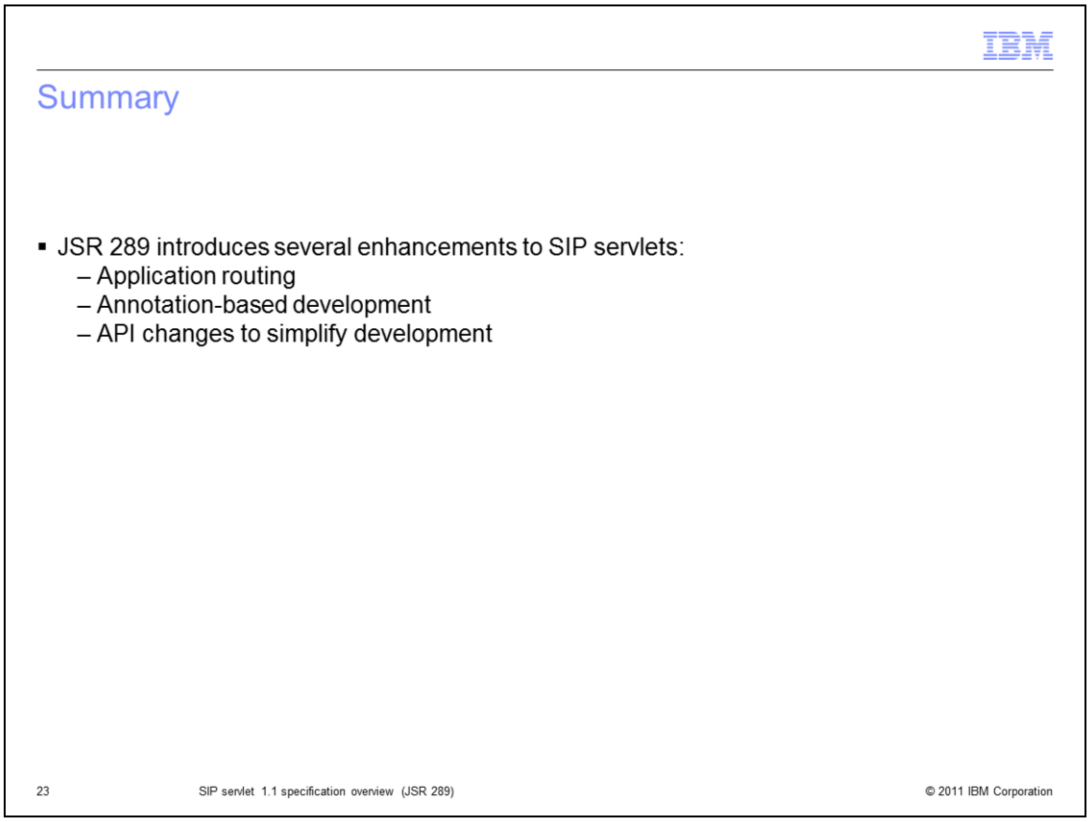
Using multihomed hosting requires both application changes and configuration changes. The SIP servlet specification 1.1 includes new APIs for multihomed support, and any application wanting to take advantage of multihomed hosting needs to use these new APIs. The APIs make available a list of outbound interfaces that is maintained by the SIP container and available to applications through a context attribute. The application must set the interface on the Proxy, the ProxyBranch, or the SipSession object before sending any outbound requests. The container sees the interface attribute and notifies the proxy which outbound interface needs to be used to send the outbound request. In order to take advantage of multihomed hosting, the SIP proxy must be configured with the appropriate outbound interfaces. Multihomed hosting is configured at the proxy level, not the SIP container level, so a multihomed topology is only supported in a network deployment environment. The next two sections of the presentation describe the multihomed hosting APIs and SIP proxy configuration in more detail.



Section

Summary and reference

This section contains a summary and references.



Summary

- JSR 289 introduces several enhancements to SIP servlets:
  - Application routing
  - Annotation-based development
  - API changes to simplify development

23 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

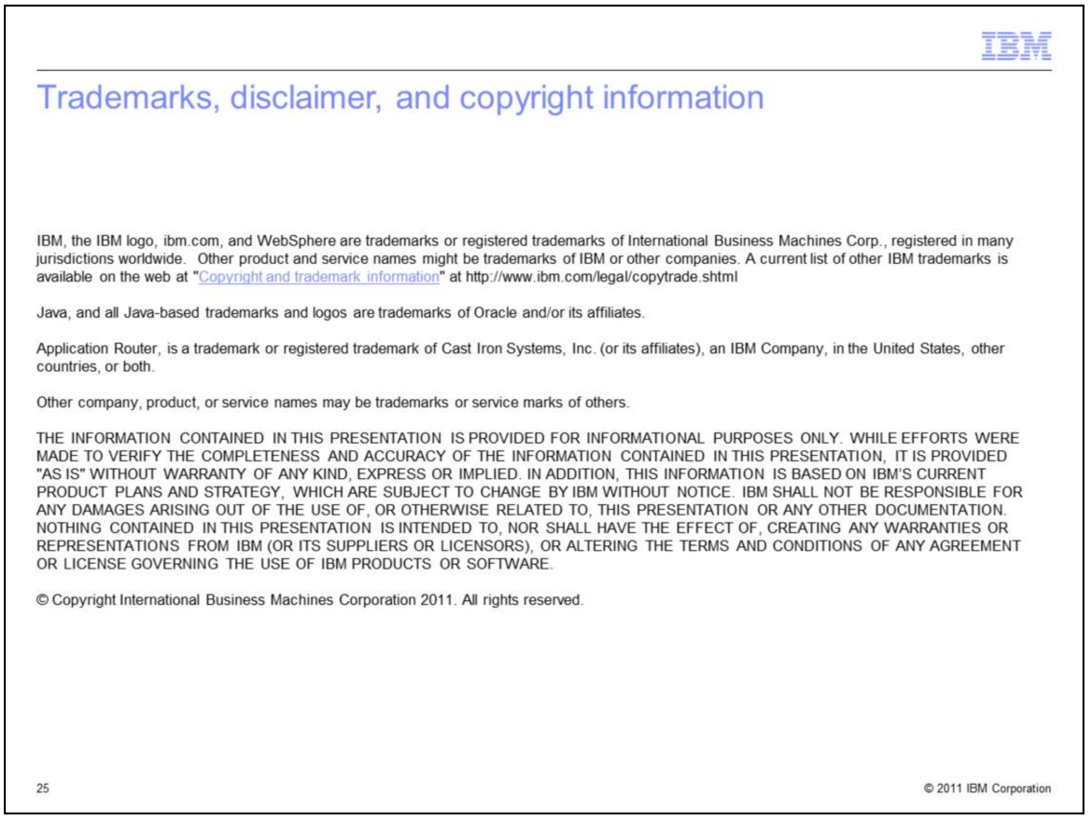
The JSR 289 specification introduces several new features for SIP applications. Application routing provides a mechanism for removing application composition logic from applications and simplifying integration of application components. Annotations speed up SIP servlet application development and reduce the need for deployment descriptors. Other API changes, like improved support for converged applications and the B2buaHelper class, also simplify application development.

Reference

- JSR 289 specification
  http://jcp.org/aboutJava/communityprocess/final/jsr289/index.html
- SIP overview in IBM Education Assistant
  http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.was_v6/was/6.1/Architecture/WASv61_SIP_overview/player.html

24 SIP servlet 1.1 specification overview (JSR 289) © 2011 IBM Corporation

This page contains a link to the official JSR 289 specification document, and a general SIP education module for WebSphere Application Server V6.1 that is available on IBM Education Assistant.



Trademarks, disclaimer, and copyright information