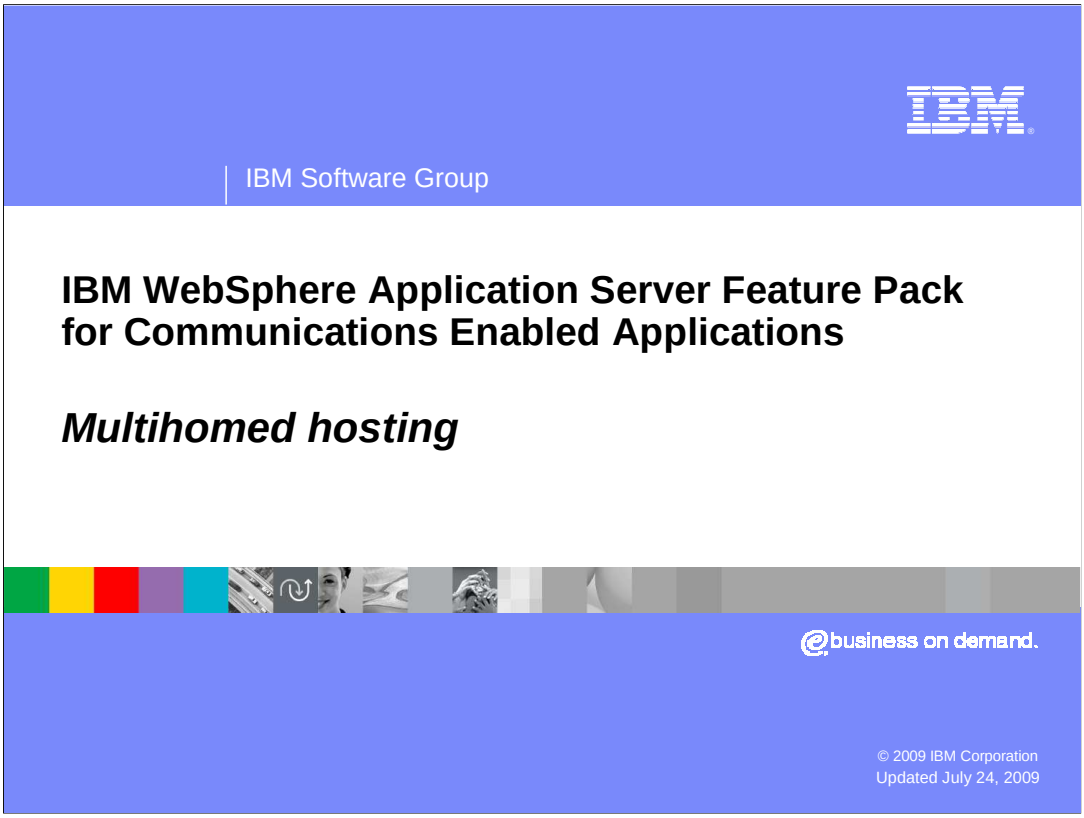


Multihoming allows you to have a single application communicate with different user agent clients and user agent servers on different networks. This functionality is included in the WebSphere® Application Server Feature Pack for Communications Enabled Applications as a part of the JSR 289 implementation.

Agenda

- Multihomed hosting overview
- Multihomed hosting APIs
- Proxy configuration

This presentation begins by providing an overview of multihomed hosting, then describes how to use the multihomed hosting APIs and set up a SIP proxy to use with your multihomed topology.

Section

Multihomed hosting overview

This section provides an overview of the multihomed hosting support included in the WebSphere Application Server Feature Pack for Communications Enabled Applications.

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

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

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

Multihomed hosting APIs

This section provides some examples of the new multihomed hosting APIs that are a part of JSR 289.

Sample code – setting the outbound interface

```
protected void doInvite(SipServletRequest req1) throws ServletException,
                                                       IOException
{
    ...
  //This block of code handles setting of the outbound interface.
  SipSession sipSession = req1.getSession();
  javax.servlet.ServletContext  context = getServletContext();
  java.util.List list =
     (java.util.List)context.getAttribute(javax.servlet.sip.SipServlet.
                                          OUTBOUND_INTERFACES);
  SipURI uri = getProtocolInterface ("udp", list);

  if (uri != null)
  {
    InetSocketAddress inetSocketAddr =
        new InetSocketAddress(uri.getHost(), uri.getPort());
    sipSession.setOutboundInterface(inetSocketAddr);
  }
    ...
}
```


The sample code here shows how to look up and set the required outbound interface on a SipSession. The list of available SIP URIs that the container can use to send outbound requests is exposed through a ServletContext attribute, javax.servlet.sip.outboundInterfaces. That attribute is defined using the static string javax.servlet.sip.SipServlet.OUTBOUND_INTERFACES, as shown here. Once the application has retrieved the list of available outbound interfaces, the application needs to include some logic to determine which interface to associate with this SipSession. In this case, an internal class method getProtocolInterface(), which is defined on the next page, is responsible for identifying the right interface to use. Once the correct interface has been selected, the interface is set on the SipSession using the new setOutboundInterface() method.

Sample code – setting the outbound interface

```
// This method pulls out the first interface in the list for the
// specified protocol

private SipURI getProtocolInterface(String transport,
                                    List outboundInterfaceList)
{
  SipURI uri = null;
  Iterator iterator = outboundInterfaceList.iterator();
  while (iterator.hasNext())
  {
    SipURI tempUri = (SipURI)iterator.next();

    if (tempUri.getTransportParam().equals(transport) == true)
    {
      uri = tempUri;
      break;
    }
  }
  return (uri);
}
```


This is a sample method that selects which outbound interface to use for a particular transport, like UDP or TCP. In this case, the logic is trivial – the method just chooses the first interface for the requested protocol that is defined in the outbound interfaces list. In a typical multihomed topology, the logic needed to identify the right interface to use might be more complicated. Once the outbound interface has been selected, this method returns the SipURI to the caller, so that the outbound interface can be set on the SipSession.

Section

Proxy configuration

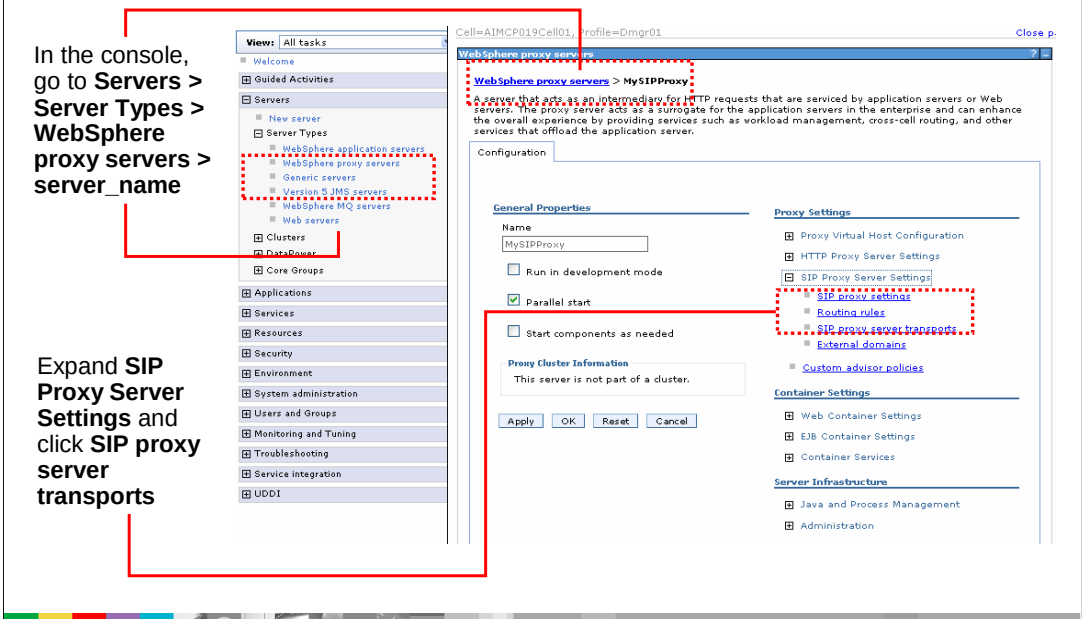
This section describes how to configure a SIP proxy to use in a multihomed topology.

Before configuring the SIP proxy

- Set up your multihome topology, which might include configuring:
  - Multiple networks (for example, routers or switches)
  - Multiple load balancers (if more than one proxy server needs to be configured for each virtual IP)
  - Multiple network cards on each of the available proxy servers
- Define a SIP proxy if you do not already have one
  - Create a SIP proxy server in the console by going to **Servers > Server types > WebSphere proxy servers**, then clicking the **New** button
  - Step through the **Create a new Proxy server** wizard to create your proxy

Before configuring the SIP proxy, you need to set up your multihome topology. This might include setting up multiple network (for example, routers or switches), multiple load balancers (if more than one proxy server needs to be configured for each virtual IP), and multiple network cards on each of the available proxy servers. After you install the network cards, you need to define a SIP proxy if you do not already have one, and configure the loopback addresses. You can create a SIP proxy in the application server's administrative console, using the configuration wizard. You can only configure the SIP proxy server to support multiple interfaces; the SIP container does not support this capability. If your environment contains more than one proxy, it is important that each is configured identically.

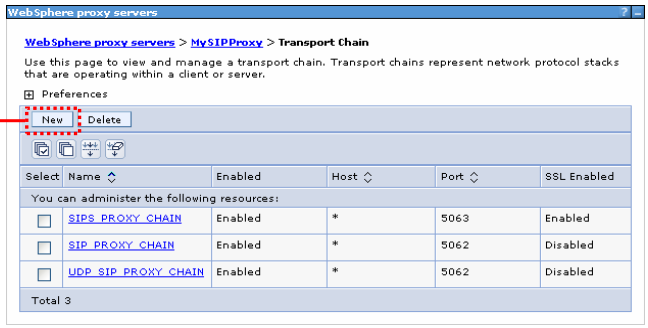
Defining SIP proxy transport chains

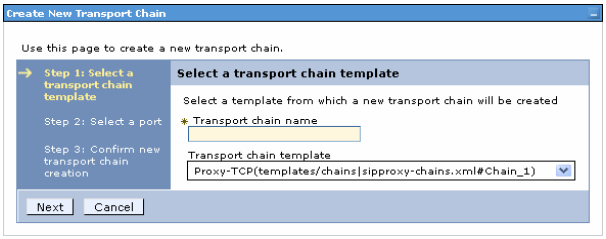


Before you can configure how to control the routing of outbound messages, you need to define all of the outbound interfaces that you are using on each SIP proxy by defining the appropriate transport chains. You can use the administrative console to define a transport chain on the SIP proxy. Start by accessing the **SIP proxy server transports** page, as shown here.

Defining SIP proxy transport chains

Click the **New** button to open the **Create New Transport Chain** wizard

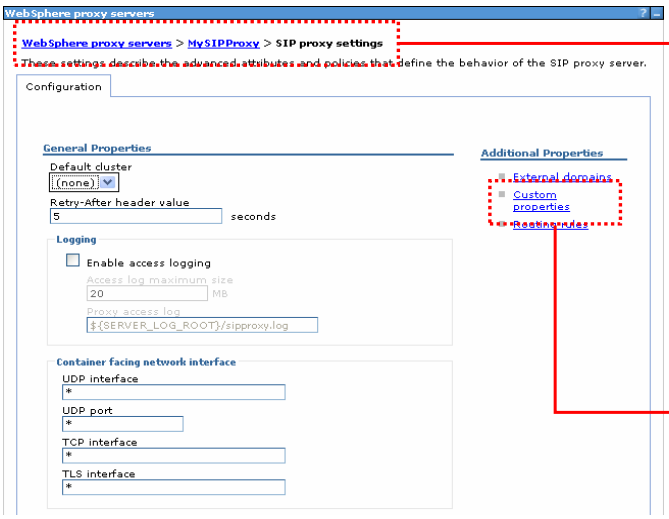




Step through the wizard to create the transport chain; do not use an asterisk (*) for the host name in these chains

Use the administrative console wizard to define a transport chain on the SIP proxy. On the **SIP proxy server transports** page, click the New button to start the **Create New Transport Chain** wizard. Step through the wizard to create your transport chain, being careful not to use an asterisk (*) for the host name in any of these chains. If the transport chains have previously been defined, you can access them through the **SIP proxy settings** panel to verify that a host name or IP address has been configured correctly for each chain.

Assigning default transport chains

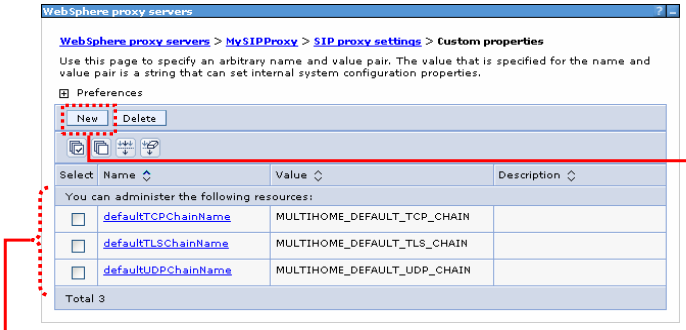


In the console, go to **Servers > Server Types > WebSphere proxy servers > server_name**, then expand **SIP Proxy Server Settings** and click **SIP proxy settings**

Select **Custom properties** to open the properties configuration panel

After all of the required chains have been defined, you need to configure which chain should be used by default for each communication protocol. These interfaces can be defined on the **SIP proxy settings** panel under **Custom properties**.

Assigning default transport chains



Click **New** to define a new custom property for the proxy server

Add the properties **defaultTCPChainName**, **defaultTLSChainName** and **defaultUDPChainName** to define for each protocol the default chain name to use when setOutboundInterface is not called

There are three custom properties that you need to add to your configuration, one for each transport type – defaultTCPChainName, defaultTLSChainName, and defaultUDPChainName. Define these three custom properties by clicking the **New** button on the **Custom properties** page, then providing the property name and value for each transport. The chains that you define here are the default chains that are used to send outbound messages when the setOuboundInterface() method is not called.

Section

Summary and reference

This section contains a summary and reference.

Summary

- Multihomed hosting support allows fine-grained control of outbound requests
- Using multihomed hosting requires both application changes and configuration changes

SIP can support the ability to route outbound SIP requests through more than a single interface with the multihomed host feature of JSR 289. In a multihomed host environment, the SIP container has the ability to select a particular outbound interface for routing messages. The SIP container can accept from the SIP proxy a list of outbound interfaces and expose it to any SIP application. This functionality is for applications that require tighter control over the outgoing request flow. Taking advantage of multihomed hosting requires both application level changes and configuration changes at the SIP proxy level.

Reference

- JSR 289 specification
  - http://jcp.org/aboutJava/communityprocess/final/jsr289/index.html

This page contains a link to the official JSR 289 specification document.

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_CEAFP_MultiHomedHosting.ppt

This module is also available in PDF format at: ../CEAFP_MultiHomedHosting.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:

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

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.