Advanced Features of the SIP Servlets Server

Table of Contents

The advanced features of SIP Servlets include Concurrency and Congestion Control, load balancing and clustering support with the Restcomm Load Balancer.

Media Support

Restcomm SIP Servlets provides support for applications to set up calls through SIP by implementing the SIP Servlets 1.1 Specification.

As most Telco services have the need for managing and controlling media (for example, to play announcements, mix calls and recognize DTMF), Restcomm SIP Servlets allows applications to control media through JSR 309.

JSR 309: Media Server Control API

This specification is a protocol agnostic API for Media Server Control. It provides a portable interface to create media rich applications with IVR, Conferencing, Speech Recognition, and similar features.

Restcomm Media Server provides an implementation of the JSR 309 specification using the MGCP protocol, to allow any Media Server (located in the same Virtual Machine or on a remote server) supporting MGCP to be controlled.

The following examples demonstrate its usage:

- Media Example : a SIP Servlet application showing how to use media capabilities (Media playback, Recording, Text to Speech with FreeTTS and DTMF detection).
- Conference Demo : a Conference Media Server demo application built on GWT with server-push updates.
- Shopping Example : a Converged JEE Application showing SEAM integration, JEE, Media integration with TTS and DTMF support.

Concurrency and Congestion Control

Concurrency and Congestion control refer to settings that define the way in which messages are processed under heavy load. The way Restcomm SIP Servlets Server processes messages in a production environment is crucial to ensure quality of service for customers.

Concurrency control mode tuning affects the way in which the SIP Servlets Server processes messages, whereas Congestion Control tuning affects the point at which the server begins rejecting new requests. Both of these parameters can be set using the following methods:

- Through the SIP Servlets Management Console.
- Editing the server's *server.xml* or *standalone-sip.xml* configuration file.
- From the `dispatcher` MBean.
- From the Embedded Jopr integrated management platform.

Concurrency Control

The JSR 289 expert group does not specify how concurrency control should be implemented.

Restcomm SIP Servlets for JBoss and Restcomm SIP Servlets for Tomcat have concurrency control implemented as a configurable mode, which defines the way in which the SIP Servlets Server processes messages.

It has to be noted that this concurrency control mechanism is not cluster aware and will work per node only, it is not a cluster wide lock.

The following modes are provided, and cater for the particular setup required in an implementation:

None

All SIP messages are processed as soon as possible in a thread from the global thread pool.

Transaction

Bypass the SIP Servlets request/response executors, and utilize the JAIN SIP built-in Transaction serialization to manage race conditions on the same transaction.

SipSession

SIP messages are processed as soon as possible except for messages originating from the same `SipSession`. These messages are excluded from any simultaneous processing.

SipApplicationSession

SIP messages are processed as soon as possible, with the guarantee that no two messages from the same `SipSession` *or* from the same `SipApplicationSession` will ever be processed simultaneously. Of all the available methods, this mode is the best choice for guaranteed thread-safety.

Congestion Control

Restcomm Sip Servlets currently provides the following congestion control mechanisms:

Changing Congestion Control Settings

All the settings and congifurations starting with gov.nist.javax.sip are located in the $JBOSS-HOME/standalone/configuration/mss-sip-stack.properties file. The section below will provide further details.

- Congestion control is largely application-specific and it is implemented in a pipeline. First messages arrive in the JAIN SIP message queue where they will wait on the locks needed before they can be processed. To avoid keeping too many messages in the queue and potentially running out of memory, older messages are discarded without any error indication. This prevents spam and flood DoS attacks to accumulate large backlog and render the server unresponsive. It also guranatees flood recovery time of 20 seconds or less, in the mean time retransmissions are already queuing so that normal SIP calls can continue without dropping them. After the request has passed the first queue it enters the SIP transaction layer where there is a customizable optional congestion control logic. There is one packaged congestion control algorithm which can be enabled by setting the following property

`gov.nist.javax.sip.SIP_MESSAGE_VALVE=gov.nist.javax.sip.stack.CongestionControlMessageValve`. For this algorithm you can set the limit value by the following property `gov.nist.javax.sip.MAX_SERVER_TRANSACTIONS=2000`. You can also implement your own algorithm and change the class name in `gov.nist.javax.sip.SIP_MESSAGE_VALVE` to activate it.

There is also another optional legacy congestion control stage with another queue where messages can be discarded based on dynamic parameters such as available JVM heap memory or number of messages in the queue. This method will be deprecated and is not recommended. All SIP messages which cannot be processed immediately are put into a queue, and wait for either a free thread or for the lock on their session to be released. The size of the SIP message queue is a tunable parameter, which defaults to `1500`.

- If the SIP Message queue becomes full, the container immediately begins rejecting new SIP requests until the queue clears. This is achieved by using one of the following methods:
  - Sending a `503` SIP error code to the originating application.
  - Dropping incoming messages (according to the specified congestion control policy).
- If the container exceeds the configurable memory threshold (90% by default), new SIP requests are rejected until the memory usage falls below the specified memory threshold. This is achieved by using one of the following methods:
  - Sending a `503` SIP error code to the originating application.
  - Dropping incoming messages (according to the specified congestion control policy).

A background task gathers information about the current server congestion. The data collection interval can be adjusted, and congestion control deactivated, by setting the interval to 0 or a negative value.

The congestion control policy defines how an incoming message is handled when the server is overloaded. The following parameters are configurable:

- DropMessage - drop any incoming message
- ErrorResponse - send a 503 - Service Unavailable response to any incoming request (Default).

Configuring the Concurrency and Congestion Control Settings

The concurrency and congestion control settings can be configured through the SIP Servlets Management Console, using the following methods:

- Through the SIP Servlets Management Console.
- Editing the server's *server.xml* or the *standalone-sip.xml* configuration file.
- From the `dispatcher` MBean.
- From the Embedded Jopr integrated management platform.

Tuning Parameters with the SIP Servlets Management Console

The easiest way to configure the SIP Message Queue Size and Concurrency Control Mode tunable parameters is to open the `SIP Servlets Management Console` in your browser (by going to http://localhost:8080/sip-servlets-management), making your changes, and then

clicking button **Apply**.

Figure 1. SIP Servlets Management Console Concurrency and Congestion Control Tuning Parameters

Concurrency and congestion control settings altered through the SIP Servlets Management Console are not saved to the server.xml on Tomcat, only on JBoss AS7 through the standalone-sip.xmlconfiguration file. To make settings persistent, append the settings to the server.xml file directly.

Making your changes permanent in standalone-sip.xml or server.xml by manual editing

Alternatively, you can edit your server's *standalone-sip.xml* or *server.xml* configuration file, which has the benefit of making your chosen settings changes permanent for Tomcat. Instructions follow, grouped by the SIP Servlets Server you are running:

Procedure: Tuning RestComm SIP Servlets for JBoss Server Settings for Concurrency and Congestion Control

1. Open standalone-sip.xml File

   Open the $JBOSS_HOME/standalone/configuration/standalone-sip.xml configuration file in a text editor.

2. Extract from stanalone-sip.xml file with conccurency configuration

```
<subsystem xmlns="urn:org.mobicents:sip-servlets-as7:1.0" application-
router="dars/mobicents-dar.properties" stack-properties="mss-sip-stack.properties"
path-name="gov.nist" app-dispatcher-
class="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl" concurrency-
control-mode="SipApplicationSession" congestion-control-interval="-1">
    <connector name="sip-udp" protocol="SIP/2.0" scheme="sip" socket-binding="sip-
udp"/>
    <connector name="sip-tcp" protocol="SIP/2.0" scheme="sip" socket-binding="sip-
tcp"/>
    <connector name="sip-tls" protocol="SIP/2.0" scheme="sip" socket-binding="sip-
tls"/>
    <connector name="sip-tls" protocol="SIP/2.0" scheme="sip" socket-binding="sip-
ws"/>
    <connector name="sip-tls" protocol="SIP/2.0" scheme="sip" socket-binding="sip-
wss"/>
</subsystem>
```

Procedure: Tuning RestComm SIP Servlets for Tomcat Server Settings for Concurrency and Congestion Control

1. Open server.xml File

   Open the $CATALINA_HOME/conf/server.xml configuration file in your text editor.

2. Add Parameters to <service> Element

   Locate the <service> element, and add the concurrencyControlMode and/or sipMessageQueueSize attributes.

   Possible values for the concurrencyControlMode attribute include: None, SipSession or SipApplicationSession. SipSession is the value of this attribute when it is not present—and overridden—in server.xml.

3. Define the Correct Attribute Values

   The following default values for the concurrency and congestion control parameters are used regardless of whether the attributes are defined in the server.xml file:

   - sipMessageQueueSize="1500"
   - backToNormalSipMessageQueueSize="1300"
   - congestionControlCheckingInterval="30000" (30 seconds, in milliseconds)
   - memoryThreshold="95" (in percentage)
   - backToNormalMemoryThreshold="90" (in percentage)
   - congestionControlPolicy="ErrorResponse"

     Experimentation is required for these tuning parameters depending on the operating system and server.

Tuning Parameters from the dispatcher MBean

Navigate to the `dispatcher` MBean from Restcomm SIP Servlets for JBoss's JMX console.

All changes performed at run time are effective immediately, but do not persist across reboots for Tomcat, only on JBoss AS7. the server.xml must be appended with the settings in order to make the configuration persistent.

When editing the dispatcher MBean from RestComm SIP Servlets for JBoss's JMX console, values allowed for the concurrency control mode are **None**, **SipSession** or **SipApplicationSession**.

STUN Support

The Session Traversal Utilities for NAT (STUN) prococol is used in Network Address Translation (NAT) traversal for real-time voice, video, messaging, and related interactive IP application communications. This light-weight, client-server protocol allows applications passing through a NAT to obtain the public IP address for the UDP connections the application uses to connect to remote hosts.

STUN support is provided at the SIP connector level, using the STUN for Java project. The STUN for Java project provides a Java implementation of the STUN Protocol (RFC 3489), which allows each SIP connector to select whether it should use STUN to discover a public IP address, and then use this address in the SIP messages sent through the connector.

To make a SIP connector STUN-enabled, three attributes must be appended to the child element in the *server.xml* or child element in *standalone-sip.xml* file. The properties are:

- useStun="true"

  Enables STUN support for this connector. Ensure that the `ipAddress` attribute is not set to `127.0.0.1`.

- stunServerAddress="<Public_STUN_Server>"

  STUN server address used to discover the public IP address of this SIP Connector. See Public STUN Servers for a suggested list of public STUN servers.

- stunServerPort="3478"

  STUN server port of the STUN server used in the `stunServerAddress` attribute. Both TCP and UDP protocols communicate with STUN servers using this port only.

A complete list of available SIP connector attributes and their descriptions is located in the Configuring SIP Connectors and Bindings section of this guide.

A number of public STUN servers are available, and can be specified in the `stunServerAddress`. Depending on the router firmware used, the STUN reply packets' MAPPED_ADDRESS may be changed to the router's WAN port. To alleviate this problem, certain public STUN servers provide

XOR_MAPPED_ADDRESS support. Public STUN Servers provides a selection of public STUN servers.

Table 1. Public STUN Servers

Server Address	XOR Support	DNS SRV Record
stun.ekiga.net	Yes	Yes
stun.fwdnet.net	No	Yes
stun.ideasip.com	No	Yes
stun01.sipphone.com	Yes	No
stun.softjoys.com	No	No
stun.voipbuster.com	No	No
stun.voxgratia.org	No	No
stun.xten.com	Yes	Yes
stunserver.org	Yes	Yes

For more information about NAT traversal best practices, refer to NAT Traversal..

Restcomm vendor-specific Extensions to JSR 289

Restcomm provide Extensions for applications or external systems to interact with the Restcomm SIP Servlets container as well as Extensions not defined in the specification in the JSR 289 specification that can prove useful and might be proposed for inclusion in a next release of the SIP Servlets specification

Javadoc for JSR 289 Extensions

CDI Telco Framework

CDI is the Java standard for dependency injection and contextual lifecycle management, led by Gavin King for Red Hat, Inc. and is a Java Community Process(JCP) specification that integrates cleanly with the Java EE platform. Any Java EE 6-compliant application server provides support for JSR-299 (even the web profile). It seemed a natural fit create a new framework based on CDI for the Telco world.

CDI-Telco-Framework (CTF) from Restcomm brings the power and productivity benefits of CDI into the Restcomm Sip Servlets platform providing dependency injection and contextual lifecycle management for converged HTTP/SIP applications. This new framework is intended to become a replacement for our previous Seam Telco Framework.

CTF mission statement is to simplify SipServlets development by introducing a component based programming model, ease of development by making available SIP utilities out of the box, and finally providing dependency injection and contextual lifecycle management to the SipServlets.

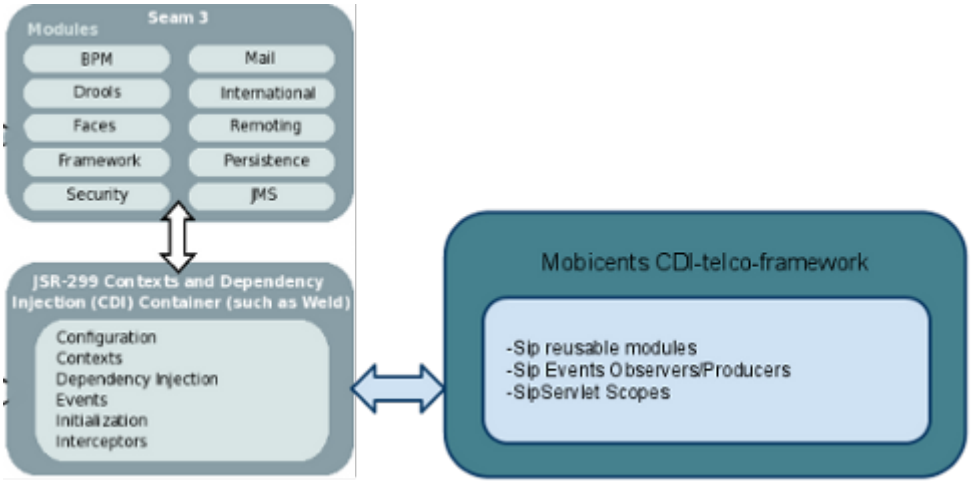

Figure 2. CDI Telco Framework Extension

More information about the CTF can be found on the CDI Telco Framework Documentation.

Diameter Support

The Diameter Protocol (RFC 3588) is a computer networking protocol for Authentication, Authorization, and Accounting (AAA). The Diameter version included in Restcomm SIP Servlets currently support Base, Sh, Ro and Rf.

For more information regarding Diameter support, refer to the Diameter Home Page. For a list of Diameter examples, refer to SIP Servlet Example Applications.

SIP and IMS Extensions

SIP Extensions in the SIP Servlets Server are based on the Internet Engineering Task Force's (IETF) Request for Comments (RFC) protocol recommendations. Supported SIP Extensions lists the supported RFCs for the SIP Servlets Server.

Table 2. Supported SIP Extensions

Extension	RFC Number	Description
DNS	RFC 3263	SIP: Locating SIP Servers
ENUM	RFC 2916	E.164 number and DNS
INFO	RFC 2976	The SIP INFO Method
IPv6	RFC 2460	Internet Protocol, Version 6 (IPv6) Specification
JOIN	RFC 3911	The SIP "Join" Header
MESSAGE	RFC 3428	SIP Extension for Instant Messaging
PATH	RFC 3327	SIP Extension Header Field for Registering Non-Adjacent Contacts

Extension	RFC Number	Description
PRACK	RFC 3262	Reliability of Provisional Responses in the SIP
PUBLISH	RFC 3903	SIP Extension for Event State Publication
REASON	RFC 3326	The Reason Header Field for the Session Initiation Protocol (SIP)
REFER	RFC 3515	The SIP Refer Method
REPLACES	RFC 3891	The SIP "Replaces" Header
STUN	RFC 3489	STUN - Simple Traversal of User Datagram Protocol (UDP) through Network Address Translators (NATs)
SUBSCRIBE/NOTIFY	RFC 3265	SIP-specific Event Notification
Symmetric Response Routing	RFC 3581	An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing
Multipart type	RFC 4662	A Session Initiation Protocol (SIP) Event Notification
To/From Header Modification	RFC 4916	Connected Identity in the Session Initiation Protocol (SIP)

IMS Private Header (P-Header) Extensions are provided according to the recommendations of the 3rd Generation Partnering Project (3GPP) and the IETF. P-Header extensions are primarily used to store information about the networks a call traverses, including security or call charging details.

IMS P-Header Extensions describes the list of supported P-Headers, including links to the relevant ITEF memorandum where available.

Table 3. IMS P-Header Extensions

AuthorizationHeaderIMS	**Defines a new auth-param for the Authorization header used in REGISTER requests.**
PAccessNetworkInfoHeader	Contains information regarding the access network the User Agent (UA) uses to connect to the SIP Proxy. The information contained in this header may be sensitive, such as the cell ID, so it is important to secure all SIP application that interface with this header.
PAssertedIdentityHeader	Contains an identity resulting from an authentication process, derived from a SIP network intermediary. The identity may be based on SIP Digest authentication.

AuthorizationHeaderIMS	**Defines a new auth-param for the Authorization header used in REGISTER requests.**
PAssertedServiceHeader	Contains information used by "trust domains", according to Spec(T) specifications detailed in RFC 3324.
PAssociatedURIHeader	Contains a list of URIs that are allocated to the user. The header is defined in the 200 OK response to a REGISTER request. It allows the User Agent Client (UAC) to determine the URIs the service provider has associated to the user's address-of-record URI.
PathHeader	SIP Extension header, with syntax similar to the RecordRoute header. Used in conjunction with SIP REGISTER requests and 200 class messages in response to REGISTER responses.
PCalledPartyIDHeader	Typically inserted en-route into an INVITE request by the proxy, the header is populated with the Request_URI received by the proxy in the request. The header allows the User Agent Server (UAS) to identify which address-of-record the invitation was sent to, and can be used to render distinctive audio-visual alert notes based on the URI.
PChargingFunctionAddressesHeader	Contains a list of one or more of the Charging Collection Function (CCF) and the Event Charging Function (ECF) addresses. The CCF and ECF addresses may be passed during the establishment of a dialog, or in a standalone transaction.
PChargingVectorHeader	Contains a unique charging identifier and correlation information, which is used by network operators to correctly charge for routing events through their networks.
PMediaAuthorizationHeader	Contains one or more session-specific media authorization tokens, which are used for QoS of the media streams.
PPreferredIdentityHeader	Contains a SIP URI and an optional display-name. For example, "James May" <sip:james@domain.com>. This header is used by trusted proxy servers to identify the user to other trusted proxies, and can be used to select the correct SIP URI in the case of multiple user identities.

AuthorizationHeaderIMS	**Defines a new auth-param for the Authorization header used in REGISTER requests.**
PPreferredServiceHeader	Used by the PAssertedService Header to determine the preferred user service. Multiple PPreferreedService headers may be present in a single request.
PProfileKeyHeader	Contains a key used by a proxy to query the user database for a given profile. The key may contain wildcards that are used as part of the query into the database.
PrivacyHeader	Contains values that determine whether particular header information is deemed as private by the UA for requests and responses.
PServedUserHeader	Contains an identity of the user that represents the served user. The header is added to the initial requests for a dialog or standalone request, which are then routed between nodes in a trusted domain.
PUserDatabaseHeader	Contains the address of the HSS handling the user that generated the request. The header field is added to request routed from an Interrogating Call Session Control Function (I-CSCF) to a Serving Call Session Control Function (S-CSCF).
PVisitedNetworkIDHeader	Contains the identifier of a visited network. The identifier is a text string or token than it known by both the registrar or the home proxy at the home network, and the proxies in the visited network.
SecurityClientHeader, SecurityServerHeader, SecurityVerifyHeader	Contains information used to negotiate the security mechanisms between a UAC, and other SIP entities including UAS, proxy and registrar.
ServiceRouteHeader	Contains a route vector that will direct requests through a specified sequence of proxies. The header may be included by a registrar in response to a REGISTER request.
WWWAuthenticateHeaderIms	Extends the WWWAuthenticateResponse header functionality by defining an additional authorization parameter (auth-param).

SIP Servlets - JAIN SLEE Interoperability

JAIN SLEE is a more complex specification than SIP Servlets, and it has been know as heavyweight and with a steep learning curve. However JAIN SLEE has standardized a high performing event driven application server, an execution environment with a good concurrency model and powerful

protocol agnostic capabilities thus covering a variety of Telco protocols.

SIP Servlets on the other hand is much simpler and easier to get started with. Its focus is on extending the HTTP Servlets and Java EE hosting environments with SIP capabilities. SIP Servlets is more of a SIP programming framework, while JSLEE is a complete, self sufficient application platform. The fact that SIP Servlets is focused on SIP and Java EE makes it a natural fit to build JEE converged applications.

Table 4. SIP Servlets / JAIN SLEE Comparison Table

SIP Servlets	**JAIN SLEE**
Application Architecture	
Based on HTTP Servlets. Unit of logic is the SIP Servlets	Component based, Object Orientated architecture. Unit of logic is the Service Building Block
Composition through Application Router	Composition through parent-child relationship
Application State	
Servlets are stateless	SBBs may be stateful
Shared state stored in a session and visible to all Servlets with access to the session	SBB state is transacted and a property of the SBB itself. Shared state may be stored in a separate ActivityContext via a type safe interface
Concurrency Control	
Application managed: use of Java monitors	System Managed: isolation of concurrent transactions
Facilities (Utilities for Applications)	
Timer, Listeners	Timer, Trace, Alarm, Statistics, Profiles.
Protocol Support	
SIP, HTTP and Media (JSR 309)Protocol agnostic.	Consistent event model, regardless of protocol/resource
Availability Mechanisms	
Container managed state (session object) that can be replicated	Container managed state (SBB CMP, Facility, ActivityContext) that can be replicated
No transaction context for SIP message processing	Transaction context for event delivery
Non transacted state operations	Container managed state operations are transacted
Facilities are non transacted	Facilities, timers, are transacted
No defined failure model	Well defined and understood failure model via transactions
Management	

SIP Servlets	JAIN SLEE
No standard management mechanisms defined	JMX Interface for managing applications, life cycle, upgrades, ...

JSLEE and SIP Servlets target different audiences with different needs, but they can be complementary in a number of real world cases.

SIP Servlets focuses on SIP and its integration with Java EE. It is also more of a SIP framework within Java EE. JSLEE is an event driven application server with protocol agnostic architecture, spanning any legacy or potential future protocols. SIP Servlets applications are generally simpler to implement and accelerate time to market for Web and SIP deployment scenarios. JSLEE has a steeper learning curve and covers a wider set of target deployment environments.

As JBoss is the only vendor to implement both specifications through Restcomm , this makes it a natural fit to build converged and interoperable JSLEE/SIP Servlets applications that are able to comply with standards in a portable manner. We built an application that could leverage standards all the way without resorting to vendor proprietary extensions by making SIP Servlets and JSLEE work together. Our "JSLEE and SIP-Servlets Interoperability with Mobicents Communication Platform" paper describes our approach and the possible different approaches we have identified to achieve the goal of interoperability between SIP Servlets and JSLEE.

You can also use our JSLEE/SIP Servlets interoperability example, showcasing our approach.

Eclipse IDE Tools

The SIP Servlets Eclipse tools assist developers in creating JSR-289 applications with Restcomm. You can use the Dynamic Web Project wizard for converged applications to get started with an empty project, and then test your application with a real SIP Phone right from the IDE.

Figure 3. SIP Servlets Eclipse IDE Tools

Pre-Install requirements

Eclipse 3.4 is required.

Installation

The standard Eclipse update site installation mechanism is leveraged. The Restcomm Update Site is at the following location: http://mobicents.googlecode.com/svn/downloads/sip-servlets-eclipse-update-site. After adding this update site to Eclipse you can proceed with the regular Eclipse Plug-in Installation. If you need help, the process is demonstrated in this video.

SIP Servlets Core Plug-in

This plug-in allows you to create Dynamic Web Projects with the SIP Facet. There are a number of new Dynamic Web Project configurations for Converged applications. It is best to use the ones marked as "recommended". After you complete the wizard, a complete converged project skeleton will be generated. Working with this type of project is similar to working with normal Web projects. You can see a demo here.

SIP Phone Plug-in

The SIP Phone plug-in integrates a SIP phone inside your Eclipse IDE. You can use the phone to test your SIP or Media applications. The phone uses the microphone and speakers on your computer and allows you to simulate real-world scenarios.