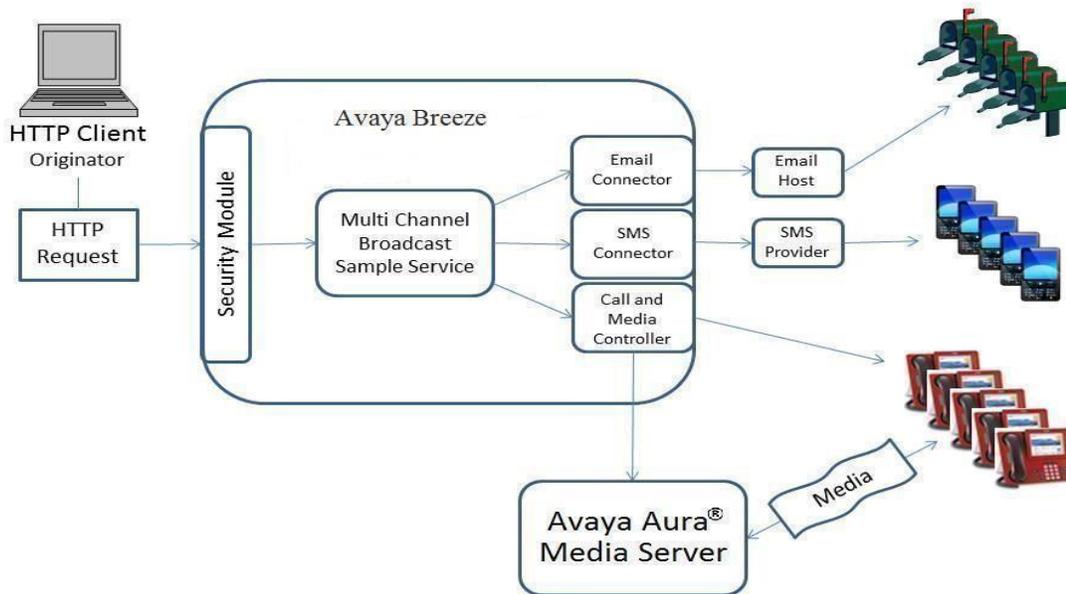


Multi-Channel Broadcast Sample Service

Introduction

The Multi-Channel Broadcast Sample Service (MCBSS) is a web service invoked by an HTTP client request that broadcasts voice and/or text notification messages to recipients via SIP calls, email, and SMS. The MCBSS is not intended to be added to a service profile like the Hello World and White List call intercepting sample services. This is true for all HTTP-only invoked services.

Multi-Channel Broadcast Sample Service Diagram



Overview

The Multi-Channel Broadcast snap-in can be configured for Zang-enabled Breeze Call or Avaya Aura™ SIP Call based.

The MCBSS is invoked when an HTTP request directed to the MCBSS is received by the Avaya Breeze® platform on the Security Module interface. After verifying the data in the HTTP request, the MCBSS retrieves its previously configured Service Global attributes from the Avaya Aura® System Manager and sends the broadcast notifications to the intended recipients utilizing the Call, Email, and/or SMS Avaya Breeze APIs. Call, Media, Email, and SMS listeners are then used to process responses as the MCBSS receives them.

Postman collection and environment files to be used with a web browser are included with the MCBSS to simplify and simulate generation of appropriate HTTP requests directed to the MCBSS.

Concepts Demonstrated

- Reception of inbound HTTP requests destined for the service
- Reading data from the service's Service Global attributes
- Using the CallFactory to generate calls
- Using the Email and SMS Avaya Breeze APIs to generate message requests for lists of recipients

- Using Avaya Aura® Media Server to play previously stored announcement files and collect digits

Detailed Description

ATTRIBUTES

The MCBSS defines multiple attributes that are configurable at the Service Global level in the Avaya Aura® System Manager UI.

Attributes Configuration

When a service is first installed, the factory default value picked by the service writer is used for each attribute for all service profiles. You may override the factory default value by using the Service Globals tab below. If you need to set specific values for attributes in a service profile, then use the Service Profiles tab below.

Service Profiles | Service Clusters | Service Globals

Service: MultiChanBroadcastService

▼ DEFAULT_GROUP

9 Items

Name	Override Default	Effective Value	Description
Announcement Calling Domain	<input type="checkbox"/>	domain.com	String used for displaying the calling domain when originating announcement calls.
Announcement Calling Number	<input type="checkbox"/>	5555555	String used for displaying the calling number when originating announcement calls. Valid values: digits
Announcement Display Name	<input type="checkbox"/>	MultiChanBroadcast Display Name	String used for displaying the calling name when originating announcement calls.
Announcement Prompt And Collect Timeout	<input type="checkbox"/>	90	Number of seconds for the 'Prompt And Collect' of an announcement and digit. This includes the length of time needed to play the announcement. The time remaining is used to collect an acknowledgement digit. Max 300. Valid values: digits
Email 'From' Address	<input type="checkbox"/>	MultiChanBroadcastFrom@domain.com	String used in the 'From' header of the broadcast email.
Email 'Reply-To' Address	<input type="checkbox"/>	MultiChanBroadcastReplyTo@domain.com	String used for replies to the broadcast email.
HTTP Pattern	<input type="checkbox"/>	https://myAvayaBreeze.example.com/services/MultiChanBroadcastService/MultiChanBroadcastServlet	HTTP pattern used to invoke this service.
SMS 'From' Number	<input type="checkbox"/>	5555555	String used in the 'From' field of the broadcast SMS. Valid values: digits
Supplier Id	<input type="checkbox"/>	10000000	Avaya provided supplier id

Voice announcement attributes:

- Announcement Calling Domain – Called party will see this attribute as the domain of the caller.
- Announcement Calling Number – Called party will see this attribute as the number of the caller.
- Announcement Display Name – Called party will see this attribute as the display name of the caller.
- Announcement Prompt And Collect Timeout – The media listener will wait, in seconds, after the announcement has begun playing to stop listening for digits.

Email attributes:

- Email 'From' Address – Email recipients will see this attribute as the 'From' address.
- Email 'ReplyTo' Address – Email recipients will see this attribute as the 'Reply-To' address.

HTTP attribute:

- HTTP Pattern – Informational-only attribute that shows the URL used to invoke the MCBSS service.

SMS attribute:

- SMS 'From' Number – SMS recipients will see this attribute as the 'From' number.

Other attributes:

- **Supplier Id** – Read-only value of Avaya’s Supplier ID.

Note: “Default SIP Domain” should be set in cluster attributes
MCBSS attributes are declared in the `properties.xml` file located at `multichanbroadcast-svar/src/main/resources/properties.xml`. The following snippet of the `properties.xml` file defines the `emailFrom` attribute:

```
<attribute name="emailFrom">
  <displayName>Email 'From' Address</displayName>
  <helpInfo>String used in the 'From' header of the broadcast email.</helpInfo>
  <global>false</global>
  <validation name="anyString">
    <type>STRING</type>
  </validation>
  <admin_visible>true</admin_visible>
  <factory>
    <value>MultiChanBroadcastFrom@domain.com</value>
    <user_changeable>true</user_changeable> </factory>
</attribute>
```

The `<orig_order>1</orig_order>` and `<orig_group>1</orig_group>` elements are also included in the `properties.xml` file. These elements are normally used to designate if the service is a calling or called service. Even though the MCBSS is not a sequenced service and is not intended to be added to a service profile, these elements are still needed by the MCBSS so that it may originate calls.

The file `multichanbroadcast-war/src/main/resources/CARRule.xml` is also required by the MCBSS so that the MCBSS can register with the Custom Application Router (CAR) and originate calls. Otherwise the MCBSS will not deploy correctly.

JAVA SOURCE

com.avaya.zephyr.services.multichanbroadcast

The Java source of the entry point for the MCBSS is in the main HTTP servlet, `MultiChanBroadcastServlet`, contained in `multichanbroadcast-war` in the main package `com.avaya.zephyr.services.multichanbroadcast`. The `MultiChanBroadcastServlet` is responsible for obtaining the service data for MCBSS attributes; invoking the announcement, email, and SMS senders; and communicating the success or failure of the queuing of these broadcast invocations in the HTTP response. The rest of the source code is contained in the `announcementcall`, `email`, and `sms` sub-packages.

Each sub-package contains a sender class and one or more listener classes.

com.avaya.zephyr.services.multichanbroadcast.announcementcall

The `com.avaya.zephyr.services.multichanbroadcast.announcementcall` package contains the voice announcement sender `MultiChanBroadcastAnnouncementSender`, and two listeners, `MultiChanBroadcastCallListener` and `MultiChanBroadcastMediaListener`.

The `MultiChanBroadcastAnnouncementSender` class verifies that the required key-value pairs, `announcementTo` and `announcementFileUri`, sent in the HTTP request to MCBSS contain data needed to send a voice announcement call to a list of recipients. Both fields must be present and contain data in the HTTP request, or an

error string will be returned for announcement call requests in the HTTP response. Each `announcementTo` field may contain one, or a comma delimited list of SIP username's (5555555) or complete SIP URI's (5555555@domain.com). If one of the `announcementTo` fields contain a SIP username, but does not include a domain, the MCBSS will append the MCBSS service global attribute 'Called Default Domain' to the username to generate a complete SIP URI. The MCBSS creates the originator of the call from the MCBSS service global attributes 'Calling Domain', 'Calling Number' and 'Display Name'. The MCBSS then creates a new call for each recipient using this newly created originator. The location of the announcement file and the acknowledgement digit timeout are set on the Call object to be used later after the call has been answered. The method `enableMediaBeforeAnswer()` is invoked on the Call object so that the Avaya Aura® Media Server can be utilized to play the announcement before initiating the call. HTTP responses are returned to the HTTP request indicating if the calls were successfully queued or not.

The `MultiChanBroadcastCallListener` class receives call progress events. This class is recognized by the framework as a call listener because it extends the `CallListenerAbstract` abstract class and is annotated with `TheCallListener`. This means that `MultiChanBroadcastCallListener` is instantiated upon service installation and not explicitly by the MCBSS. It is the call listener for every Call object generated by the MCBSS. When a call is answered, the framework invokes the `callAnswered` method which notifies `MultiChanBroadcastMediaListener` to begin playing the announcement and start digit collection. When a call is terminated, `callTerminated` is invoked notifying `MultiChanBroadcastMediaListener` and logs to `/var/log/Avaya/sm/asm.log` or `/var/log/Avaya/services/MultiChanBroadcastService/MultiChanBroadcastService.log` depending on your logging settings.

The `MultiChanBroadcastMediaListener` class receives play/collect commands from the `MultiChanBroadcastCallListener`, forwards them to the Avaya Aura® Media Server, and processes the media responses received back from the Avaya Aura® Media Server. Only one instance of this class is instantiated by `MultiChanBroadcastCallListener` upon service installation. In order to allow logs for media responses from Avaya Aura® Media Server to contain the called party, two maps (UUID/Call and Call/UUID) have been added. When a UUID is passed in from an Avaya Aura® Media Server response, the UUID/Call map is used to find out which called party the Avaya Aura® Media Server response refers to.

When the `MultiChanBroadcastCallListener` notifies the `MultiChanBroadcastMediaListener` that a call has been terminated, the Call object parameter is used to clear out the entries in both maps.

The `promptAndCollect()` API call invokes the Avaya Aura® Media Server to play an announcement and prepares it for digit collection. The `digitsCollected()` callback is invoked when Avaya Aura® Media Server has detected a DTMF event. Alternatively, a service may utilize the `play()`, and `collect()` API calls and the `playCompleted()`, and `digitsCollected()` callback methods individually.

Note that the MCBSS only plays one announcement file per request. A service may make use of the API's ability to play multiple announcements in a single play requests by passing multiple parameters.

`com.avaya.zephyr.services.multichanbroadcast.email`

The `com.avaya.zephyr.services.multichanbroadcast.email` package contains the email sender `MultiChanBroadcastEmailSender`, and email listener `MultiChanBroadcastEmailListener`.

The `MultiChanBroadcastEmailSender` class verifies that the required field/value pairs, `emailTo`, `emailSubject`, and `emailBody` sent in the HTTP request contain the data needed to send an email to a list of recipients. All fields must be present and contain data in the HTTP request, or an error string will be returned for

email requests in the HTTP response. Each `emailTo` field may contain one, or a comma delimited list of email addresses.

The MCBSS adds the 'From' and 'ReplyTo' email addresses to the email request if they exist in the MCBSS service global attributes. If the 'From' address does not exist, the email request is passed to the email connector in hopes that a 'From' email address is administered in its service global attribute. The MCBSS creates one email request for each HTTP request it receives containing all the email contacts that it sends to the email connector. Unlike the `MultiChanBroadcastAnnouncementSender`, the `MultiChanBroadcastEmailSender` needs to instantiate and attach an instance of `MultiChanBroadcastEmailListener` for every email it creates.

The `MultiChanBroadcastEmailListener` class logs statements and events when receiving responses about the email from the email connector.

`com.avaya.zephyr.services.multichanbroadcast.sms`

The `com.avaya.zephyr.services.multichanbroadcast.sms` package contains the SMS sender `MultiChanBroadcastSmsSender`, and SMS listener `MultiChanBroadcastSmsListener`.

The `MultiChanBroadcastSmsSender` class verifies that the required field/value pairs, `smsTo`, and `smsBody`, sent in the HTTP request contain data needed to send an SMS to a list of recipients. Both fields must be present and contain data in the HTTP request, or an error string will be returned for SMS requests in the HTTP response. Each `smsTo` field may contain one, or a comma delimited list of SMS numbers. The MCBSS creates one SMS request for each HTTP request it receives containing all the SMS contacts that it sends to the SMS connector.

Unlike the `MultiChanBroadcastAnnouncementSender`, and like the `MultiChanBroadcastEmailSender`, the `MultiChanBroadcastSmsSender` needs to instantiate and attach an instance of `MultiChanBroadcastSmsListener` for every SMS it creates.

The `MultiChanBroadcastSmsListener` class logs statements and events when receiving responses about the SMS from the SMS connector.

Installation and Configuration

MCBSS

The MCBSS is included in the SDK in the SDK zip file at `/samples/MultiChanBroadcast`.

- Change your directory to where the MCBSS resides and compile it (`mvn clean install`).
- Load and install the svar on System Manager and administer the MCBSS's Service Global attributes.

ZANG ENABLED BREEZE CALLS

The `ZangCallConnector` snap-in must be installed and configured. See "Deploying Zang-Enabled Avaya Breeze" for additional information about configuring the Zang Call Connector.

POSTMAN (HTTP REQUEST GENERATOR)

In order to demonstrate an incoming HTTP request to Avaya Breeze, we have chosen to include Postman Collection and Postman Environment setting files in the MCBSS (`samples/MultiChanBroadcast/MultiChanBroadcastCollection`). We suggest installing Google Chrome and then installing the 'Postman – REST Client' from the Chrome Web Store. It is free and simple to use once you get the hang of it. Follow the Postman video tutorial in the Web Store and install and configure the Postman Collection and Postman Environment.

The purpose of the Postman Environment is to create default HTTP request URL encoded parameters (default domains, host addresses, etc.) that can be substituted in the Postman Collection. You do not have to use the Postman Environment, but it makes it easier for generating and testing large numbers of voice announcement,

email, and SMS recipients. In the Postman Environment, set the `host` field to the IP address of the Security Module on Avaya Breeze you will be sending your HTTP request to. Do not enter the FQDN or IP address of the Network Management interface of Avaya Breeze! The `servicename` and `servicepath` should be left to their defaults. Change the `toemaildomain` and `announcementcalledomain` fields to their desired values. The `fileURI` is the beginning of the URI of the location of the pre-recorded announcement file available to the Avaya Aura® Media Server used for voice announcement calls.

In order to use the Postman Environment defaults (i.e. `host`, `servicename`, etc.) in the Postman Collection use double braces surrounding the field name. (i.e.

`{{host}}/services/{{servicename}}/{{servicepath}}`, `email@{{toemaildomain}}`, etc.). When editing or viewing the Postman Collection, please use the “`x-www-form-urlencoded`” button to view the data sent in the HTTP request. Leave the line containing the URL to `{{host}}/services/{{servicename}}/{{servicepath}}` unmodified. These are the values used from the Postman Environment. The Postman Collection is defaulted to send an email to one recipient, play an announcement to two recipients (one with a domain and one without), and send an SMS to one recipient. In order to send voice announcements, email, or SMS to multiple contacts you can either add a new row of data with `announcementTo`, `emailTo`, or `smsTo` in the `Key` field, or append multiple recipients to the existing `announcementTo`, `emailTo`, or `smsTo` key-value pairs separating them with a comma.

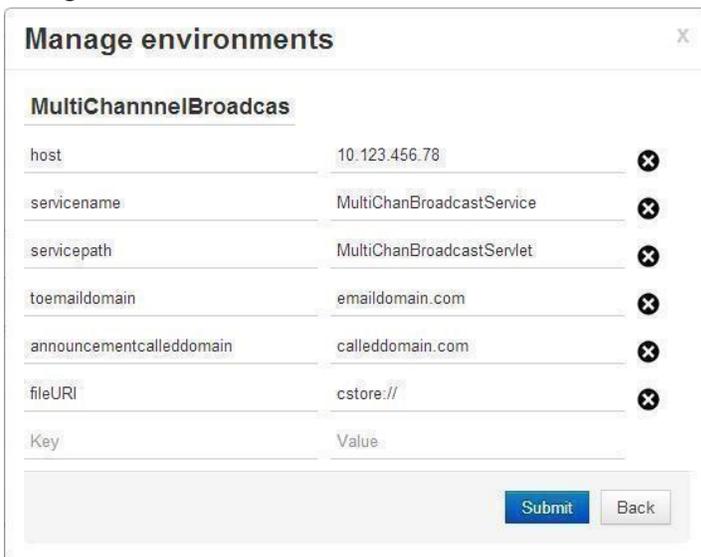
Example: “`emailTo/email1@domain.com`”
“`emailTo/email2@domain.com`”
OR
“`emailTo/email1@domain.com, email2@domain.com`”

If you do not wish to send an email, delete all the key-value pairs that have the key `email`. The same applies for announcements and SMS.

Testing the Service

Test MCBSS to call two SIP stations, email three email boxes, and one SMS enabled phone(789):

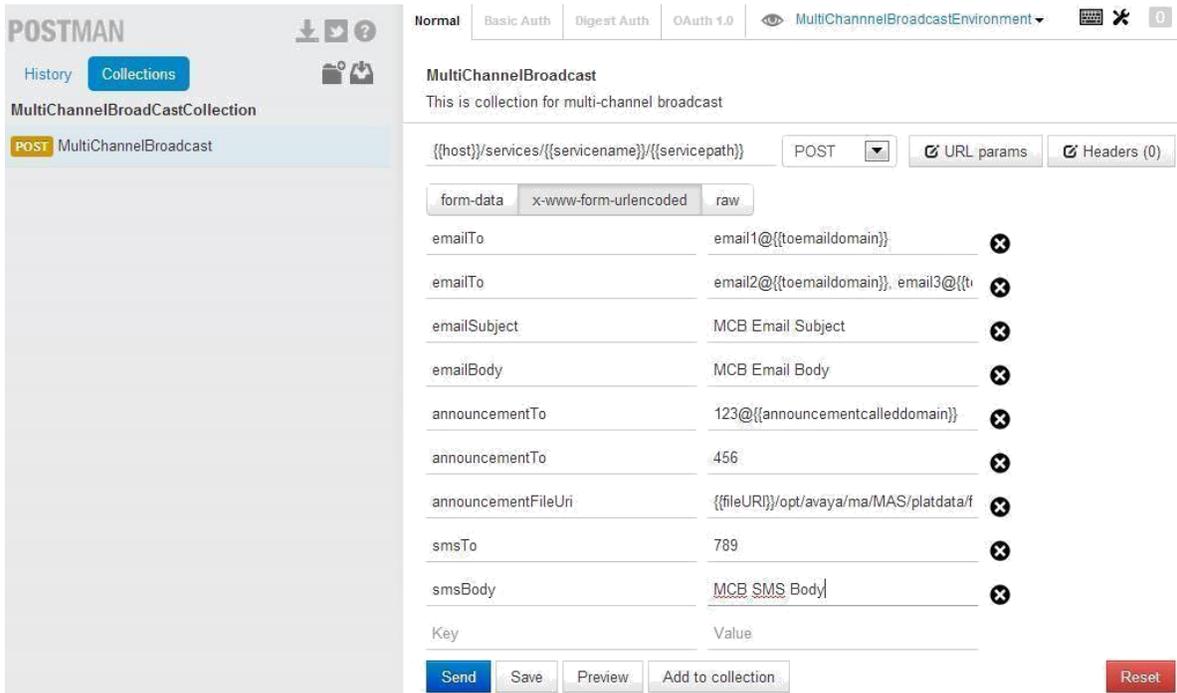
- Configure Postman.



The screenshot shows the 'Manage environments' window in Postman. The environment name is 'MultiChannelBroadcas'. It contains the following key-value pairs:

Key	Value
host	10.123.456.78
servicename	MultiChanBroadcastService
servicepath	MultiChanBroadcastServlet
toemaildomain	emaildomain.com
announcementcalledomain	calledomain.com
fileURI	cstore://
Key	Value

At the bottom of the dialog, there are 'Submit' and 'Back' buttons.



- Send the HTTP request from Postman.
- In Postman, verify the HTTP response from Avaya Breeze resembles the following:
 - Info: Email queued to be sent to email1@emaildomain.com
 - Info: Email queued to be sent to email2@emaildomain.com
 - Info: Email queued to be sent to email3@emaildomain.com
 - Info: Announcement queued to be sent to 123@calleddomain.com
 - Info: Announcement queued to be sent to 456@calleddomain.com
 - Info: SMS queued to be sent to 789
- When the SIP stations ring, listen for the announcement to be played. When you hear the announcement, press a digit to acknowledge you heard the announcement. Avaya Breeze will log this acknowledgement to the operationalEvent.log.
- Verify the three emails were received.
- Verify the SMS message was received.

Troubleshooting

PROBLEM: Receive “503 Service Temporarily Unavailable” HTTP response in Postman when sending an MCBSS request.

ACTION: Verify the Avaya Breeze is in the “Accepting” System State on Avaya Aura® System Manager. This is found on the Elements->Avaya Breeze->Server Administration page.

PROBLEM: Receive “404 Not Found” HTTP response in Postman when sending an MCBSS request.

ACTION: Verify that the service is installed on the Avaya Breeze. If the MCBSS has recently been installed, you may need to wait a few minutes before the service can be invoked.

PROBLEM: Voice announcement recipient does not ring.

ACTION: Verify the `announcementTo` value in the Postman Collection is correct and utilizes the correct domain of the recipient.

PROBLEM: Voice announcement recipient does not hear the recorded announcement.

ACTION: Verify the location of the announcement file on Avaya Aura® Media Server is correct. Instead of using the Postman environment variables, hardcode the location of the file in the `announcementFileUri` value field of the Postman Collection.

PROBLEM: Avaya Breeze log does not record DTMF acknowledgment when Avaya Breeze is under load.

ACTION: Lower the total number of voice announcement recipients in the Postman Collection. We suggest that Postman Collections be limited to 200 voice announcement message recipients per request for Avaya Breeze nodes that are not already under heavy load.