Callingpolicies Sample Snap-in

Introduction

The Calling Policies snap-in demonstrates a number of features available to the snap-in writer. These include the abilities to allow a call to its original target, drop the call, redirect the call, or to demonstrate Parallel Forking, Sequential Forking, and Serial Calling.

The Callingpolicies sample snap-in prompts the calling party to press a digit that will determine which one of the following operations is performed:

1. Allow operation: The call is allowed to proceed to its original destination (called party).
2. Drop operation: The call is dropped.
3. Redirect operation: The call is immediately redirected to a configured destination.
4. Parallel Forking operation: The call is forked to the configured destinations.
5. Sequential Forking operation: The alerting call will be redirected to a configured destination if the original destination doesn't answer within 10 seconds
6. Make Another Call operation: The call will be redirected to a configured destination when the original destination (called party) drops from the call.
7. Call Back operation: The caller will be called back if he/she drops from the call.

Overview

The Callingpolicies snap-in is a terminating side, called party snap-in. This means that it is invoked for calls to users who have the snap-in enabled in their Service Profile. It also means that the snap-in is not invoked for users making calls. Termination Sequence must be properly configured for called party (see figure 2).

When invoked, the sample snap-in instructs the Avaya Aura® Media Server to play the Callingpolicies.wav file. The Callingpolicies.wav file is located at Callingpolicies/Callingpolicies-war/src/main/webapp/Callingpolicies.wav in the SDK.
The wav file is part of the sample snap-in war. This file does not need to be installed on Avaya Aura® Media Server. As long as Avaya Aura® Media Server is installed and configured properly according to the [DEVC: Deployment and Administration guides], Avaya Aura® Media Server will be able to retrieve the wav file via HTTPs from the sample snap-in and play it to the caller.

Additionally, to establish trust between Avaya Aura® Media Server and Avaya Breeze® platform for HTTPs to work, please refer to two sections of the "Administering Avaya Aura® Media Server 7.6" document (https://downloads.avaya.com/css/P8/documents/100178657),
chapter4:

- "network settings configuration" -> "configuring connection security options"
- "importing a trust certificate to the trust store" -> "security configuration"

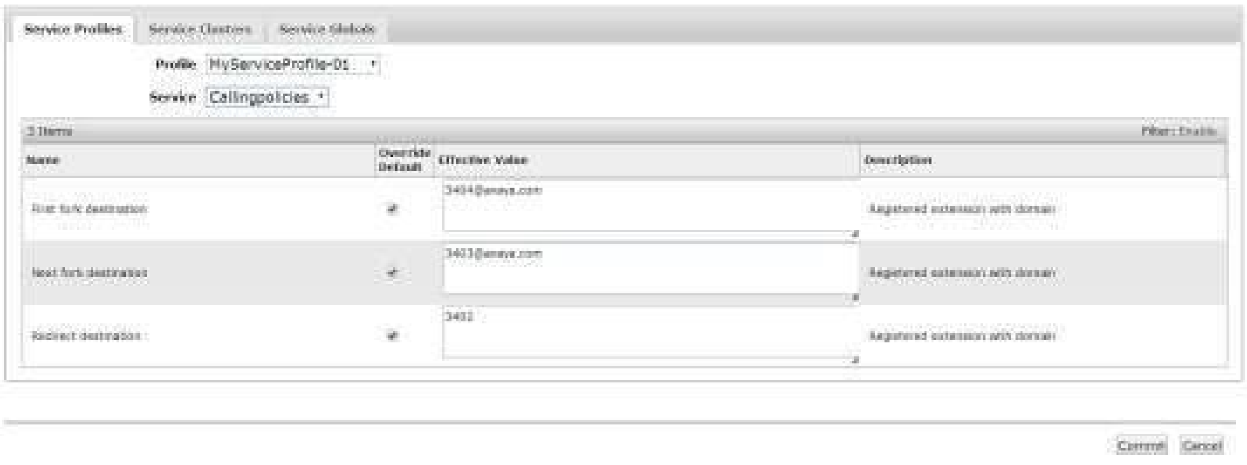
For assistance on exporting trusted certificate, please refer to "Administering Avaya Aura® Session Manager-release 6.3" in the "certificate management" section.

The snap-in prompts the user to dial a digit to select from a list of options. If no input is received from the caller within 60 seconds, the call is dropped.

If an invalid digit, not corresponding to one of the seven operations 1,2,3,4,5,6,7 is pressed, then the call is dropped as well.

Attributes for this service are configured in the Service Profile for Avaya Breeze in System Manager, as shown in figure 1. The first two attributes represent the two fork destinations. The last attribute represents the redirect destination. The destinations can be any phone number. The forking operation proceeds to call both the configured destinations as well as the called party.

Figure 1



Profile: MyServiceProfile-01

Service: Callingpolicies

3 Items

Name	Override Default	Effective Value	Description
First fork destination	✓	3404@avaya.com	Registered extension with domain
Next fork destination	✓	3403@avaya.com	Registered extension with domain
Redirect destination	✓	3402	Registered extension with domain

Attributes are defined in the `properties.xml` descriptor.

Note that this example shows configuration of attributes at a Service Profile level. For more information on snap-in attributes, including information about configuring attributes at the cluster or global level, refer to the Service Development guide [DEVC: Service Development Guide].

Concepts Demonstrated

- Reading attributes from the user's Service Profile.
- Allowing the call to proceed to the called party using Avaya Breeze API's `Call.allow` method.
- Allowing the call to be dropped by the called party using Avaya Breeze API's `Call.drop` method.
- Redirecting a call using the Avaya BreezeAPI's `Call.divertTo` method. This method diverts a call to a different destination.
- Parallel Forking a call using the Avaya Breeze API's `Call.addParticipant` method. This method adds an additional participant to the call.

- Sequential Forking on a call using a Avaya Breeze API's `Call.dropParticipant` and `Call.addParticipant` methods. The `Call.dropParticipant` method drops a given participant (alerting party in this case) from the call leaving single leg call active. The `Call.addParticipant` method adds an additional participant to the call.

- Calling a party back after an unexpected drop using a Avaya Breeze API's `Call.addParticipant` method after the calling party drops form the call. The `Call.addParticipant` method adds new participant (original caller party in this case) to the call. This could be considered as a "Call Back" operation.

- Calling an alternate party after an original participant drops. This operation uses the Avaya Breeze API's `Call.addParticipant` method after the original destination (called party) drops form the call. The `Call.addParticipant` method adds new participant to the call. This could be considered as a "Make another Call" operation.

- Generating and setting a `playItem` object, playing a prompt and collecting a digit. This shows the use of methods such as `setSource`, used to set the wav file path, `setInterruptible`, that defines whether or not audio announcements may be interrupted when a digit is detected, `and setIterateCount`, that sets the number of times the prompt is played .

Detailed description

The service framework invokes the class `CallingPoliciesCallListener` recognized by the fact that it extends the `CallListenerAbstract` class, and is annotated with `TheCallListener`. The `Call Object` is then passed on to the method `executePromptAndCollectOperation()` of the class `PromptAndCollectOperationImpl` that plays the announcement and collects a digit.

The above mentioned class makes use of `MediaService.createplayItem()` that sets the configuration of the prompt such as `setSource()` method, that specifies the wav file location, `setInterruptible()`, that defines whether or not audio announcements may be interrupted when a digit is detected, and `setIterateCount()`, that specifies the number of times prompt can be played.

In addition to creating `PlayItem`, `MediaService.digitOptions()` is used to specify digit options such as `setNumberOfDigits()`, that specifies the number of digits to be collected, `setTerminationKey()`, to set the key that would terminate the digit collection, `setTimeout()`, to specify the milliseconds to wait to receive the first digit. `MediaService.promptAndCollect()` is invoked to play the announcement to the participant. Next, the `digitCollected()` method of the `CallingPoliciesMediaListener` class is

invoked, after the digit has been collected. In addition to validating the digit collected, it invokes the

`forkTo()` method of the `ForkingOperationImpl` class and the

`getCallPolicies().setCallTerminationPolicy(), addParticipant(),`

`dropParticipant(), divertTo(), allow()` and `drop()` methods belonging to the

`Call` Avaya Breeze API. The `ServiceAttributeReaderImpl` class and `getForkedDestinations()` methods are used to obtain the service attributes from the attribute page.

For Sequential Forking, Make Another Call and Call Back options the Call Termination Policy is set to CallTerminationPolicy.NO_PARTICIPANT_REMAINS. It means to clear the call when no Participant remains in the call (i.e. the last remaining Participant dropped).

Media Server Inclusion policy determines how long Avaya Aura® Media Server will be used during the call. MediaServerInclusion.AS_NEEDED is a default value and it means that Media Server will be removed from the call when it is not needed. MediaServerInclusion.INCLUDED indicates that the Media Server will be used for the whole duration of the call. Default Media Server Inclusion policy is sufficient for Callingpolicies snap-in.

Snap-in Invocation

There are two places that properties files have been configured to ensure that the Callingpolicies snap-in gets invoked appropriately.

In the properties.xml document under Callingpolicies-svar/src/main/resources, the following two properties have been populated, indicating that this is a called party snap-in:

```
<term_order>1</term_order>
<term_group>1</term_group>
```

Additionally, the following rule was populated in the CARRule.xml file located under Callingpolicies-war/src/main/resources. This rule is required for any snap-in that uses Avaya Aura® Media Server, as Callingpolicies does to play announcements.

```
<TerminatingServiceRule desc="Interested in Callingpolicies featureURI as
named app">
<FeatureURI>Callingpolicies</FeatureURI>
</TerminatingServiceRule>
```

Installation and Configuration

The Callingpolicies snap-in can be found in the SDK zip at: `/samples/Callingpolicies`. Change directory to where the snap-in resides and compile it (`mvn clean install`). Then load and install the svar on System Manager, enable it in Service Profiles, and administer the forking and redirect destinations for a Service Profile.

For information on installing the snap-in, assigning it to users, and configuring the display string attribute for Service Profiles see the [DEVC: Installing, Configuring and Testing a CE Service guide].

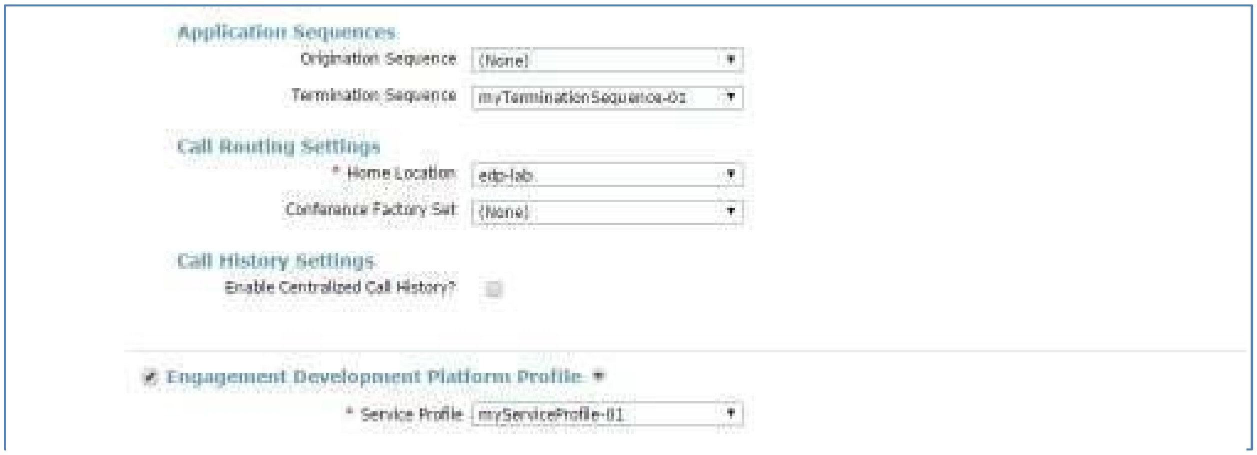
Testing the Snap-in using an example

Assuming five users: called party: 3401@avaya.com, calling party:3402@avaya.com, first fork destination: 3403@avaya.com, next fork destination: 3404@avaya.com, redirect destination: 3300@avaya.com

1. Please note: the communication profile for the station form for the called user must be configured as shown in Figure 2 below with the service profile included.
2. Call 3401 from an external number. 3401 is the called party.
3. The announcement is played.
4. Based on the digit pressed, call is forked, redirected, allowed to continue or dropped.

   - For 1= Allow, the called party (in our case, 3401@avaya.com) rings.
   - For 2= Drop, the call is dropped and the calling party may hear a tone indicating that the call has been dropped.
   - For 3= Redirect, it redirects to the destination provided in the service attribute.
   - For 4= Parallel Forking, both fork destinations and the original destination are alerted.
   - For 5= Sequential Forking, original destination is ringing for 10 seconds. Then original destination stops ringing and the new destination (redirecting number attribute) is alerted.
   - For 6= Serial Calling (Make Another Call), original destination is alerted. Original destination then answers the call. If the original destination drops the call, then the snap-in will add new party to the call (redirecting number attribute). Original calling party will hear the ring-back tone.
   - For 7= Serial Calling (Call Back), original destination (3401@avaya.com in our example) is alerted. Original destination then answers the call. If the calling party drops from the call, then the snap-in will call that party back.. The original destination (3401@avaya.com) will hear the ring-back tone.
   - For any other digit than 1,2,3,4,5,6,7 the call is dropped.

*Please Note: The caller cannot be a Communication Manager station.



Application Sequences

Origination Sequence: (None)

Termination Sequence: myTerminationSequence-01

Call Routing Settings

* Home Location: edp-lab

Conference Factory Set: (None)

Call History Settings

Enable Centralized Call History? ☐

☑ Engagement Development Platform Profile *

* Service Profile: myServiceProfile-01

Figure 2

Troubleshooting

Problem:
When a call is made, no announcement is played.

Action:

1. Please make sure Avaya Aura® Media Server has been configured and is up and running.
2. Please make sure the snap-in has been assigned to the called party as shown in Figure 2.