

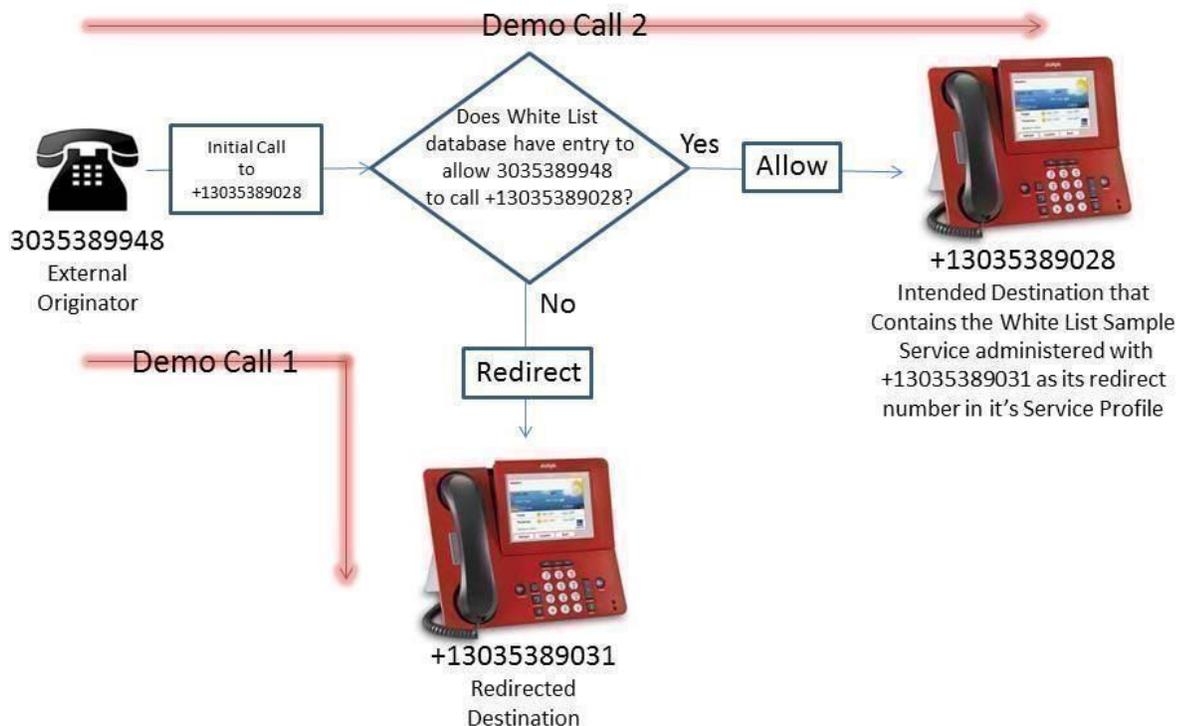
# Whitelist Sample Service

## Introduction

The Whitelist sample service is a broker service which either allows an incoming call to proceed to its original destination (called party) or redirects the call to another destination (e.g., call center). The service consults with database to determine whether the calling number can directly call the called party.

Example of a use case: a stock broker may wish to accept direct calls from VIP clients while sending other calls to a call center for processing.

## White List Process Flowchart



## Overview

The Whitelist service is a called party service. This means that it is invoked for calls to users who have the service enabled in their Service Profile. It also means that the service is not invoked for users making calls.

When invoked for a call, the service queries the whitelist database to determine whether there is an associated entry for calling party's number (calling\_handle) with the called party's number (called\_handle) in the whitelist table. If the calling party is present in the table for the called party, the service allows the call to proceed as dialed. However, if that is not the case, the call is redirected to the number administered for the service in the user's Service Profile (The Redirect Number administered on Service Attributes screen on Avaya Aura® System Manager).

As an example, given the following whitelist table, 303-555-1234 and 303-555-5678 can directly call 3302. The calling\_handle needs to match exactly what the network sends when offering the call. However, 303-555-0000 is not allowed to call 3302 directly; that call would be redirected to the administered number.

White List		called_handle
calling_handle		
3302		3035551234
3302		13035551234
3302		+13035551234
3302		3035555678
3302		13035555678
3302		+13035555678

## Concepts Demonstrated

- Redirecting a call using the Call method divertTo in the Avaya Breeze® platform API<sup>[2]</sup>
- Reading an attribute from Avaya Breeze the user's Service Profile<sup>[2]</sup>
- Looking up an EJB from a TheCallListener annotated class<sup>[2]</sup>
- Accessing an external PostgreSQL database using JPA<sup>[2]</sup>

## Detailed description

The Whitelist service defines the redirect number as a service attribute. Service attributes appear in the Avaya Aura® System Manager UI where they can be administered on a per-profile basis. Attributes are defined in the properties.xml descriptor. Below is the fragment of the properties.xml file that defines the redirectNumber attribute.

```

<attribute name="redirectNumber">
  <displayName>Redirect Number</displayName>
  <helpInfo>Alternate destination for the call. The correct format is:
    handle@domain For instance, +17205551212@example.com
</helpInfo>
  <global>>false</global>
  <validation name="AnyString">
    <type>STRING</type>
  </validation>
  <portlet_changeable>>false</portlet_changeable>
  <admin_visible>>true</admin_visible>
  <admin_changeable>>true</admin_changeable>
  <factory>
    <value>3301@avaya.com</value>
    <user_changeable>>true</user_changeable>
  </factory>
</attribute>

```

For detailed information about defining service attributes, see the Service Development guide [DEVCC: Service Development Guide]

The heart of the service is the inbound call listener class defined in `WhiteList.java`. This class is recognized by the framework as an inbound call listener because it extends the `CallListenerAbstract` abstract class and is annotated with `TheCallListener`. When a call is delivered to the service, the `callIntercepted` method will be invoked by the framework.

The service makes use of EJBs. The central EJB from the client's (`WhiteList`) perspective is `DestinationFinderImpl`. It implements the `getDestination` method in the `DestinationFinderinterface`, which returns the actual destination for the call. The `callIntercepted` method uses JNDI to look up a `DestinationFinderImpl` EJB.

The service first uses the `DestinationFinder` to look up a destination for the call given the calling party and called party. Next it instructs the framework to redirect the call to the new destination. Note that for simplicity's sake the `divertTo` method is used for both redirecting calls to the alternate destination and allowing calls to proceed as dialed (by redirecting to their original destination). Two potential alternatives to this approach:

1. Use `divertTo` only in the case where the caller is not on the called party's whitelist and allow otherwise.

2. Use divertTo only in the case where the caller is not on the called party's whitelist and do not explicitly use allow. The default behavior on a call is allow.

The DestinationFinderImpl uses PermissionAgentImpl along with WhiteListDaoImpl to query the external database to determine if a call from user A to user B is allowed to proceed as dialed. The AlternateDestinationFinderImpl is used to look up the redirect number in the called party's Service Profile. If the database cannot be accessed, all calls will be redirected.

## Installation and Configuration

The cluster must have the "Enable Cluster Database" option checked during the installation of the Snap-in, as the database will be stored in Cluster Database.

The Whitelist service is part of the Avaya Breeze SDK (/samples/Whitelist). Change directory to where the service resides and compile it (mvn clean install). Then load and install the svar on System Manager, enable it in Service Profiles, and administer the redirect number for a Service Profile. See the process defined here [DEVC: Installing, Configuring and testing a Avaya Breeze Service ]

## Administering user whitelists

There are multiple ways to manage data in the whitelist database now.

**REST APIs are included with the Snap-in. Using Postman we can insert, query and delete data from database.**

Example,

1. List all Whitelist entries

GET http://<cluster IP or Asset IP>/services/WhiteList/ws/entries

Sample Input, GET <http://10.133.132.62/services/WhiteList/ws/entries>

Sample Output,

```
[{"calledHandle":"12340","callingHandle":"12342"}, {"calledHandle":"1234t55635","callingHandle":"12342345356"}, {"calledHandle":"12370011","callingHandle":"5078888"}, {"calledHandle":"123837","callingHandle":"12345634"}, {"calledHandle":"1238070","callingHandle":"123456"}]
```

2. Add an entry

POST http://<cluster IP or Asset IP>

/services/WhiteList/ws/entry/callingParty/<CalingPartyHandle>/calledParty/<CalledPartyHandle>

Sample Input, POST <http://10.133.132.62/services/WhiteList/ws/entry/callingParty/123/calledParty/123>

Sample Output,

```
[{"calledHandle":"12340","callingHandle":"12342"}, {"calledHandle":"1234t55635","callingHandle":"12342345356"}, {"calledHandle":"123","callingHandle":"123"}] (Entry Added)
```

3. Query an entry

GET http://<cluster IP or Asset IP>

/services/WhiteList/ws/entry/callingParty/<CallingPartyHandle>/calledParty/<CalledPartyHandle>

Sample Input, GET <http://10.133.132.62/services/WhiteList/ws/entry/callingParty/123/calledParty/123>

Sample Output,

```
[{"calledHandle":"123","callingHandle":"123"}]
```

4. Query an entry by called handle

GET http://<cluster IP or Asset IP>

/services/WhiteList/ws/entry/calledParty/<CalledPartyHandle>

Sample Input, GET <http://10.133.132.62/services/WhiteList/ws/entry/calledParty/123>

Sample Output,

```
[{"calledHandle":"123","callingHandle":"123"}]
```

5. Remove an entry

DELETE http://<cluster IP or Asset IP>

/services/WhiteList/ws/entry/callingParty/<CallingPartyHandle>/calledParty/<CalledPartyHandle>

Sample Input, DELETE <http://10.133.132.62/services/WhiteList/ws/entry/callingParty/123/calledParty/123>

Sample Output,

```
[{"calledHandle":"12340","callingHandle":"12342"}, {"calledHandle":"1234t55635","callingHandle":"12342345356"}] (Entry Removed)
```

**There's a User Interface for the Whitelist Sample Snap-in, which can be used to insert, search and delete data from database.**

It can be accessed via <http://<cluster IP or Asset IP>/services/WhiteList/index.html>

- a. Entries can be added through the called party handle and calling party fields provided.
- b. Search can be done through either called party handle or calling party handle. The search box is provided for the same.
- c. One can refresh the table entries through the 'Refresh' button provided

- d. One or more entries can be deleted at a time. Just select the check box corresponding to the row of a particular entry or list of entries and click 'Remove'

**A shell script is included on the Avaya Breeze server to administer the whitelist database for this sample service.**

1. To list entries in the database, run 'whitelist\_util.sh list' or just run 'whitelist\_util' from the Avaya Breeze Server console
2. To add a new entry, run 'whitelist\_util.sh add <called\_number> <calling\_number>'.
3. To remove an entry, run 'whitelist\_util.sh remove <called\_number> <calling\_number>'.

### Testing the service

Assuming three users: A, B, and C

1. Enable the whitelist service for user B.
2. Configure user C as the redirect destination in user B's Service Profile.
3. Make a call from user A to user B and verify that the call is redirected to user C. Note that the call is redirected because user A is not in user B's whitelist
4. Use whitelist\_util.sh to add a whitelist entry for A calling B.
5. Make another call from A to B and verify that the call goes through this time.

### Troubleshooting

Problem:

#### WhiteList Entries

<input type="checkbox"/>	Called Handle	Calling Handle
<input type="checkbox"/>	12340	12342
<input type="checkbox"/>	1234t55635	12342345356
<input type="checkbox"/>	12370011	5078888
<input type="checkbox"/>	123837	12345634
<input type="checkbox"/>	1238070	123456
<input type="checkbox"/>	1234	1234
<input type="checkbox"/>	3453653	3r2223rt234t34

Showing 1 to 10 of 12 rows  rows per page

1 2

#### Add a whitelist entry

Called Party	<input type="text" value="Enter Called Party handle"/>
Calling Party	<input type="text" value="Enter Calling Party handle"/>
<input type="button" value="Add Entry"/>	

An exception similar to the following is logged to the `/var/log/Avaya/mgmt/zephyr/debug.log`:

```
0000001e JPAPUnitInfo E CWWJP0015E: An error occurred in the
org.apache.openjpa.persistence.PersistenceProviderImpl persistence provider when it attempted to create the
container entity manager factory for the whitelist persistence unit. The following error occurred:
<openjpa-2.1.2-SNAPSHOT-r422266:1333100 fatal user error>
org.apache.openjpa.persistence.ArgumentException: Could not invoke the static newInstance method on
the named factory class "com.ibm.ws.persistence.jdbc.kernel.WsJpaJDBCBrokerFactory".
```

**ACTION:**

Make sure that the whitelist database has been created by using any one of the method mentioned above in “Administering user whitelists” section.

## **Remove the service from Call**

This feature allows WhiteList to request that Avaya Breeze be removed from the call signaling path without the call being dropped. This helps conserve Breeze resources and removes Breeze as a potential point of failure in the call. It can be invoked irrespective of admin as per above section to allow call to proceed to dialed number or in case of diversion of call due to absence of whitelist rule.

### **Concepts Demonstrated**

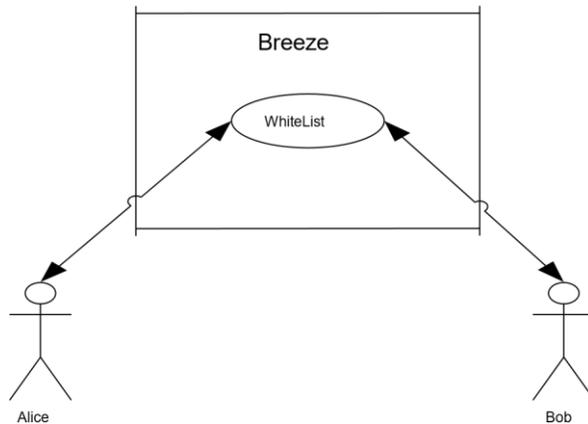
- This feature uses the Collaboration API’s ServiceManager method `removeServiceFromCall` to remove WhiteList / Breeze from the call. This is illustrated in `WhitelistResource.java` in the method `removeServiceFromCall`.
- The WhiteList application is notified of the success / failure of the operation through its `WhiteListServiceListener.java` class. This listener has a `TheServiceListener` annotation and it implements the `ServiceListener` interface.

**Notes:**

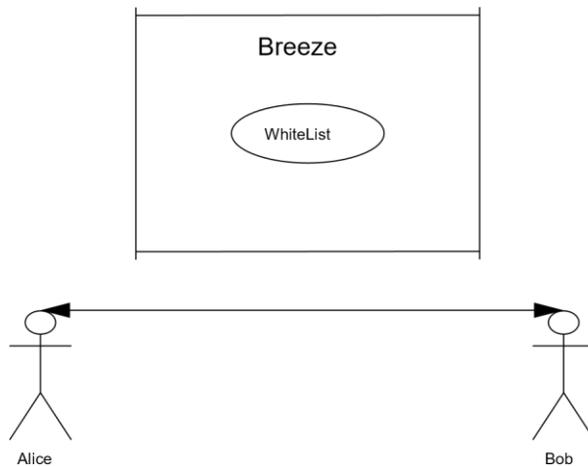
- Care must be taken while provisioning the Implicit User application sequence for the called party. If the default domain setting of “Any” is used, Avaya Breeze and the WhiteList snap-in will be sequenced anew after invoking `removeServiceFromCall`. To avoid this situation, it is required to specify the domain when configuring the Implicit User application sequence.
- If there are any Calling Party applications for the caller, the Avaya Breeze will be re-sequenced rather than being removed completely from the call.
- This operation may only be invoked if a single snap-in is involved in the call. If there may be more than one snap-in, this operation should be removed

**Call Flows**

Before invoking `ServiceManager.removeServiceFromCall`, the signalling call flow is as shown below



After successful invocation of `ServiceManager.removeServiceFromCall`, the signaling call flow is as shown below



## REST APIS

**REST APIs are included with the Snap-in. Using request in postman collection, we can remove the Whitelist service from call**

- Install and administer Whitelist service as detailed in section above for Bob.
- Place call from caller Alice to Bob.
- Answer the call on Bob's end.
- Locate the Universal Call ID – UCID in below service log file :  
`/var/log/Avaya/services/WhiteList/WhiteList.log`
- Use this UCID in the REST API like below.
- Verify that the service was removed from Call when success logs below are creating in the service log.

Example,

GET `http://<cluster IP or Asset IP>/services/WhiteList/ws/entry/removeServiceFromCall/<ucid>`

Sample input, (GET)

<http://10.133.132.62/services/WhiteList/ws/entry/removeServiceFromCall/00010000031537557983>

Sample output, (GET) called removeServiceFromCall on call= local.1537557829458\_66!  
10.133.132.61

Sample log (WhiteList.log):

2018-09-21 13:26:23,522 [SipContainerPool : 3] WhiteList INFO - WhiteList-3.6.0.0.0 - callIntercepted  
ucid=00010000031537557983

2018-09-21 13:26:23,525 [SipContainerPool : 3] WhiteList INFO - WhiteList-3.6.0.0.0 -  
isWhiteListedNumber from named query whiteListEntry=null

2018-09-21 13:26:37,275 [SipContainerPool : 2] WhiteList INFO - WhiteList-3.6.0.0.0 -  
removeServiceFromCallSucceeded for callId=local.1537557829458\_66!10.133.132.61