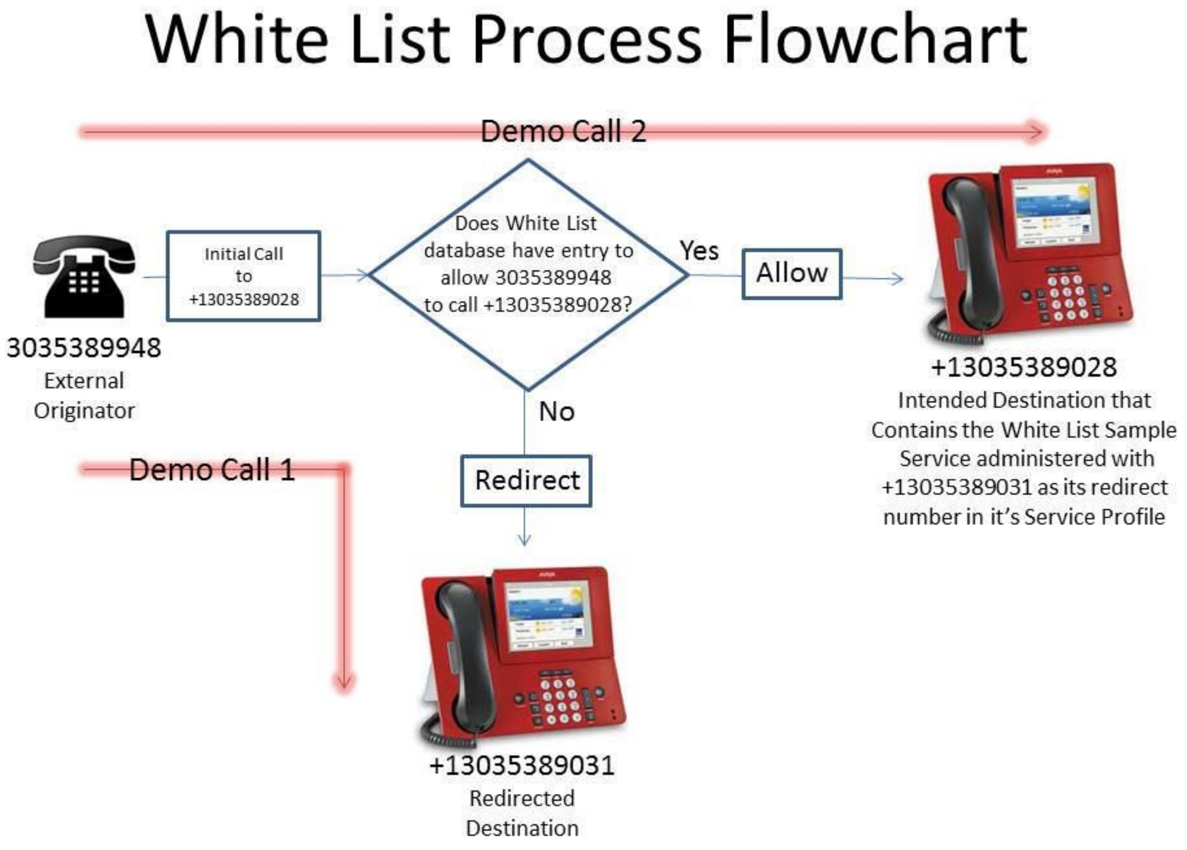
Whitelist Sample Service

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 an external database to determine whether the calling number is allowed to 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
Demo Call 2
Initial Call to +13035389028
Does White List database have entry to allow 3035389948 to call +13035389028?
Yes
Allow
No
Redirect
Demo Call 1
3035389948
External Originator
+13035389028
Intended Destination that Contains the White List Sample Service administered with +13035389031 as its redirect number in it's Service Profile
+13035389031
Redirected Destination

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 System Manager).

As an example, given the following whitelist table, 303-555-1234 and 303-555-5678 are allowed to directly call 3302. The calling_handle needs to match exactly what the network sends when offering the call[1]. 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
```

Note: The Whitelist service reads from a local database on the Avaya Breeze™ instance. This makes for a good demonstration, but is not recommended in production as a cluster of multiple Avaya Breeze instances may be used to service requests and the data would need to remain synchronized across the instances.

Concepts Demonstrated

- Redirecting a call using the Call method divertTo in the Avaya Breeze API
- Reading an attribute from Avaya Breeze the user's Service Profile
- Looking up an EJB from a TheCallListener annotated class
- Accessing an external PostgreSQL database using JPA

Detailed description

The Whitelist service defines the redirect number as a service attribute. Service attributes appear in the 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</portl_ael
            <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 [DEVC: 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 `DestinationFinder` interface, 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 in order 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 Whitelist service is part of the 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

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

- To list entries in the database, run `whitelist_util.sh list`.
- To add a new entry, run `whitelist_util.sh add <called_number> <calling_number>`.
- 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:

An exception similar to the following is logged to the 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 running whitelist_util.sh and then restart the container.