Zebra Voice Client

Version 9.0.24403

Workcloud Communication

Programmer Guide

ZEBRA

MN-001718-10EN Rev A

Copyright

2025/03/17

Terms of Use

Proprietary Statement

This manual contains proprietary information of Zebra Technologies Corporation and its subsidiaries (“Zebra Technologies”). It is intended solely for the information and use of parties operating and maintaining the equipment described herein. Such proprietary information may not be used, reproduced, or disclosed to any other parties for any other purpose without the express, written permission of Zebra Technologies.

Product Improvements

Continuous improvement of products is a policy of Zebra Technologies. All specifications and designs are subject to change without notice.

Liability Disclaimer

Zebra Technologies takes steps to ensure that its published Engineering specifications and manuals are correct; however, errors do occur. Zebra Technologies reserves the right to correct any such errors and disclaims liability resulting therefrom.

Limitation of Liability

In no event shall Zebra Technologies or anyone else involved in the creation, production, or delivery of the accompanying product (including hardware and software) be liable for any damages whatsoever (including, without limitation, consequential damages including loss of business profits, business interruption, or loss of business information) arising out of the use of, the results of use of, or inability to use such product, even if Zebra Technologies has been advised of the possibility of such damages. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

Contents

Contents

About this Guide

This guide provides application programming interface (API) information for writing applications that use Zebra Voice. The guide assumes that the reader is familiar with Zebra Voice intents.

The API consists of a series of intents that can be invoked via the following methods:

1. **StageNow**: A Zebra Voice intent can be generated using **StageNow** as described at techdocs.zebra.com/mx/intent/. The intent can be sent immediately, scheduled, or triggered off by pressing a button or sensor.
2. **MDM**: A Zebra Voice intent can be generated from many MDMs. A couple of examples are provided below:
   - **Soti Mobicontrol**: soti.net/mc/help/v14.4/en/scriptcmds/reference/sendintent
   - **Airwatch/Workspace ONE** docs.vmware.com/en/VMware-Workspace-ONE-UEM/index
3. **Zebra Voice application**: A Zebra Voice application can generate an intent to another application on the device. For more information, refer to developer.android.com/guide/components/intents-filters .

When Zebra Voice receives the intent, it takes the action indicated by the intent. The available intents are detailed in the following sections and include initiating Voice calls, sending messages, signing in/out of the application, and configuring the client.

Chapter Descriptions

The following chapters cover in this guide:

- About this Guide chapter provides explanation about the notational conventions, service information, related documentation, and revision history.
- Actions chapter provides information about enabling background voice configuration.
- Headless Mode provides information about configuring the PTT Pro application.
- Voice Connecter provides information about WCCvoiceConnector library and initiating and controlling voice call remotely .

Notational Conventions

The following notational conventions make the content of this document easy to navigate.

- **Bold** text is used to highlight the following:
  - Dialog box, window, and screen names
  - Dropdown list and list box names
  - Checkbox and radio button names
  - Icons on a screen
  - Key names on a keypad
  - Button names on a screen
- Bullets (•) indicate:
  - Action items
  - List of alternatives
  - Lists of required steps that are not necessarily sequential
- Sequential lists (for example, those that describe step-by-step procedures) appear as numbered lists.

Service Information

If you have a problem with your equipment, contact Zebra Global Customer Support for your region. Contact information is available at: zebra.com/support.

When contacting support, please have the following information available:

- Serial number of the unit
- Model number or product name
- Software/firmware type and version number

Zebra responds to calls by email, telephone, or fax within the time limits set forth in support agreements.

If your problem cannot be solved by Zebra Customer Support, you may need to return your equipment for servicing and will be given specific directions. Zebra is not responsible for any damages incurred during shipment if the approved shipping container is not used. Shipping the units improperly can possibly void the warranty.

If you purchased your Zebra business product from a Zebra business partner, contact that business partner for support.

Revision History

Revision	Date	Description
MN001718A01, Rev A	02/2016	Initial release.
MN001718A02, Rev A	05/2019	Added a new section, Signing Out Registered Users.
MN001718A03, Rev A	06/2021	Updated for WFC Voice version 9.0.21109.

Revision	Date	Description
MN001718A04, Rev A	11/2021	Updated the Make Call intents, versions supported, and UI changes.
MN001718A05, Rev A	03/2022	Updated intents for Subscribe, Android 11, and above.
MN001718A06, Rev A	09/2022	Fixed the bug contacts_url in the Import Contacts topic.
MN-001718-07EN Rev A	01/2024	Added Data Consent and Pinboard Integration - Intents/APIs.
MN-001718-08EN Rev A	04/2024	Rebranded to Workcloud Communication.
MN-001718-09EN Rev A(v9.0.24304)	12/2024	Updated Phone State Monitoring.
MN-001718-10EN Rev A(v9.0.24403)	03/2025	Added Implementing Shared URI Reporting via Intent.

Actions

This section provides descriptions and methods to initiate actions available in the Zebra Voice API.

Load New Configuration File

The Zebra Voice app reads the XML configuration file and restarts services with the new configuration. This is used during initial deployment or after an upgrade.

Prerequisites

- Zebra Voice Client is loaded on the device.
- An XML file with configuration parameters has been prepared and is available.
- All versions of Zebra Voice Client 9.0 support these intents.

NOTE: The URI may be a network location or a file location on the device.

Intent Definition

Name	Description
Action	`wfc.voice.ACTION_NEW_CONFIG`
Intent Type	`startActivity`
Extra 0	This extra parameter defines the URI of the configuration file.
Type	`String`
Name	`profile_uri`
Value	URI pointing to file location

ADB Examples

File located on the network:

```
$ adb shell am start -a wfc.voice.ACTION_NEW_CONFIG --es profile_uri https://
example.com/voice/profiles/1234
```

File located on the device:

```
$ adb shell am start -a wfc.voice.ACTION_NEW_CONFIG --es profile_uri /
 sdcard/new_profile.xml
```

Pushing the Config XML to the device (For Android 11):

```
adb push d:\adb\WFCConnect.xml /enterprise/device/settings/WFConnect
```

```
adb shell am start -a wfc.voice.ACTION_NEW_CONFIG --es profile_uri /
enterprise/device/settings/WFConnect/WFConnect.xml
```

Update Configuration Parameters

The Zebra Voice app updates parameters and reloads services. It may be simpler to use this intent when updating one or two parameters, as an option to loading a new configuration file.

Prerequisites

- Zebra Voice Client is loaded on the device.
- All versions of Zebra Voice Client 9.0 support these intents.

Intent Definition

Name	Description
Action	`wfc.voice.ACTION_UPDATE_CONFIG`
Intent Type	`startActivity`
Extra 0	This extra parameter specifies the configuration parameter and value. This extra parameter may be present once or multiple times to update multiple configuration parameters.
Type	`String`
Name	Parameter(s) as defined in the appropriate Zebra Voice PBX Administrator Guide.
Value	Corresponding value

NOTE: The URI may be a network location or a file location on the device.

ADB Example

Updating the log level in the client:

```
$ adb shell am start -a wfc.voice.ACTION_UPDATE_CONFIG --es log_level Error
```

Update with Token

This intent is used to pass a token to the Provisioning Manager for the following purposes:

- Receive configuration information associated with the token in return. After application is successfully configured, the application restarts. For more information on specific configuration parameters related to PBX, refer to the Voice PBX Administrator Guide.
- Associate Voice Client with a customer account for licensing the client. The token may or may not have configuration information associated with it.

Prerequisites

- The Provisioning Manager must be set up with the token used in the intent. Refer to the Workcloud Communication Provisioning Manager Customer Administrator Guide for more information on tokens and associated configuration information with them.
- All versions of Voice Client 9.0 support these intents.

Intent Definition

Name	Description
Action	`android.intent.action.VIEW` with URI and package
Intent Type	`startActivity`
URI	`wfcvp://<token>`
Package	`com.symbol.wfc.voice`

NOTE: Upon a fresh installation of the application, Zebra Voice contacts the Provisioning Manager, using its device ID as the token, and receives any configuration information associated with its device ID in return.

ADB Example

```
$ adb shell am start -a android.intent.action.VIEW -d "wfcvp://ACME-3016"
 com.symbol.wfc.voice
```

Configure the Subscribe Feature

The following intent can be used to subscribe or unsubscribe the departments when used along with Extension Manager and Asterisk PBX.

Prerequisites

- Zebra Voice Client is loaded on the device.
- Zebra Voice Client version 9.0.21410 and above.
  - Asterisk PBX
  - Extension Manager

Intent Definition

Name	Description
Action	wfc.voice.SUBSCRIBE
Intent Type	startActivity
Extra 0	This parameter defines the names of the subscriptions which are subscribed or unsubscribed.
Type	Array
Name	subscriptions
Value	Subscription strings
Extra 1	This optional parameter defines whether the subscriptions are subscribed (true) or unsubscribed (false). If this parameter is not present, it sets to `true` by default..
Type	Boolean
Name	clear
Value	true or false

ADB Examples

Subscribe to departments. If any departments were previously selected, the new departments are added to the existing list. Multiple departments can be separated by a comma without space.

```
$ adb shell am start -a wfc.voice.SUBSCRIBE --esa subscriptions "Food,Toys"
```

Unsubscribe from all departments and subscribe to new departments. Only the new departments are selected:

```
$ adb shell am start -a wfc.voice.SUBSCRIBE --esa subscriptions "Food" --ez
 clear true
```

Unsubscribe from all departments:

```
$ adb shell am start -a wfc.voice.SUBSCRIBE --ez clear true
```

Zebra Voice Sign-out Action

The `wfc.voice.SIGN_OUT` action achieves different goals depending on the presence and value of the change parameter. It can be used to invoke the following behavior in the Zebra Voice Client:

- **Reload** (formerly called Sign out/Reload): Use with Extension Manager to sign out and reload the configuration based on the configuration parameters or use it to simply force Zebra Voice Client to re-register when configured without Extension Manager.
- **Change Department**: Use with Extension Manager to sign out and reload the configuration with a list of all available departments. Users can select new departments to register with multiple extensions. In legacy configurations, this is the same as reload.

- **Add Department**: Use with Extension Manager to sign out and reload the configuration with a list of all available departments, where the originally configured department is auto-selected. Users can add or change departments to register with multiple extensions. In legacy configurations, this is the same as reload.
- **Sign Out**: Use to sign out and de-register from the PBX. The app remains signed-out until the user manually restarts the app or the app receives an external intent to reload. Sign Out functions the same with or without Extension Manager.

Prerequisites

- Zebra Voice Client is loaded on the device and the user is signed in.
- All versions of Zebra Voice Client 9.0 support these intents.

Intent Definition

Name	Description
Action	`wfc.voice.SIGN_OUT`
Intent Type	`broadcast`
Extra 0	This extra parameter specifies the requested behavior when using the voice client alone or in combination with the Extension Manager.
Type	`String`
Name	change
Value	Reload: `change` parameter is not present or `change` =2
	Change Department: `change`=1
	Add Department:`change`=3
	Sign Out: `change`=0

ADB Examples

Reload:

```
$ adb shell am broadcast -a wfc.voice.SIGN_OUT
```

Change Department:

```
$ adb shell am broadcast -a wfc.voice.SIGN_OUT --es change 1
```

Add Department:

```
$ adb shell am broadcast -a wfc.voice.SIGN_OUT --es change 3
```

Sign Out:

```
$ adb shell am broadcast -a wfc.voice.SIGN_OUT --es change 0
```

Quit

Use Quit to sign out, de-register from the PBX, and quit all application services.

Prerequisites

- Zebra Voice Client is loaded on the device.
- All versions of Zebra Voice Client 9.0 support these intents.

Indent Definition

Name	Description
Action	`wfc.voice.STOP_APP`
Indent Type	`broadcast`

ADB Example

```
$ adb shell am broadcast -a wfc.voice.STOP_APP
```

Make a Call

There are several intents which can be used to make a voice call. The different intents are supported due to the differing requirements third-party applications have, but they all result in initiating a call in Zebra Voice.

Emergency call is supported by all the intents in this section by ensuring that a call can be made at all times, even if the device is locked.

The intents can also be used to bring up the dialpad by omitting the number or address of the intended recipient. The regular dialer is brought up when the device is unlocked; the emergency dialer is brought up when the device is locked.

Supported Schema

The following table summarizes the intents and the supported schemas for each.

Intent	Supported Schemas
wfc.voice.ACTION_BUTTON	"tel", "sip", "csip", "sips"
android.intent.action.VIEW	"tel", "sip", "csip", "sips
android.intent.action.CALL	"tel", "sip", "sips"
android.intent.action.DIAL	"tel", "sip", "sips"

Prerequisites

- Zebra Voice Client is loaded on the device, configured for PBX connectivity, and registered to the PBX.
- All versions of Zebra Voice Client 9.0 support these intents.

Make a Call with wfc.voice.ACTION_BUTTON

Name	Description
Action	`wfc.voice.ACTION_BUTTON`
Intent Type	`startActivity`
Data Scheme	tel, sip, csip, or sips
Extra 0	
Type	`String`
Name	action
Value	CALL
Extra 1	
Type	`String`
Name	value
Value	Number or address of the intended recipient

Make a Call with android.intent.action.VIEW

Name	Value
Action	`android.intent.action.VIEW`
Intent Type	`startActivity`
Data Scheme	tel, sip, csip, or sips followed by ":" and the number or address of the intended recipient

Make a Call with android.intent.action.DIAL

Name	Description
Action	`android.intent.action.CALL`
Intent Type	`startActivity`
Data Scheme	tel, sip, or sips followed by ":" and the number or address of the intended recipient.

NOTE: Zebra Voice can be registered to multiple PBX lines simultaneously. This intent is sent to the current line, which can be selected via the Zebra Voice dial pad. The user can switch between lines by clicking "Local line" info field.

ADB Examples

```
$ adb shell am start -a wfc.voice.ACTION_BUTTON -d 'tel:' --es action CALL--
es value 8471234567$
```

```
$ adb shell am start -a android.intent.action.CALL -d sip:2001
```

```
$ adb shell am start -a android.intent.action.VIEW -d csip:2001
```

```
$ adb shell am start -a android.intent.action.DIAL -d tel:555
```

The following intent brings the regular dialer up when the device is unlocked and the emergency dialer when the device is locked.

```
adb shell am start -a android.intent.action.DIAL -d sip:
```

End a Call

Use the End Call action to end an active call session.

Prerequisites

- Zebra Voice Client is loaded on the device and configured for PBX connectivity.
- Zebra Voice Client is currently involved in a call.
- The minimum required Zebra Voice Client version is 9.0.20302.

Intent Definition

Name	Description
Action	`com.zebra.wfc.ACTION_END_ACTIVE_SESSION`
Intent Type	`broadcast`

ADB Example

```
$ adb shell am broadcast -a com.zebra.wfc.ACTION_END_ACTIVE_SESSION
```

Import Contacts

Import contacts from a CSV file. The file is stored on the device or on a server.

Prerequisites

- The Zebra Voice Client is loaded to the device.
- The CSV file is stored on the device or on a server.
- All versions of Zebra Voice Client 9.0 support these intents.

Intent Definition

Name	Description
Action	`wfc.voice.SYNC_CONTACTS`
Intent Type	`startActivity`
Extra 0	Location of the contacts CSV file

Name	Description
Type	`String`
Name	`contacts_url`
Value	URL of a downloadable CSV file or a file name on the SD card
Extra 1	This extra parameter specifies if the existing contacts should be deleted or not before applying the CSV file
Type	`boolean`
Name	`reset`
Value	`true`: the app deletes all existing contacts prior to loading the CSV file `false`: existing contacts are not deleted

ADB Examples

To load a file stored on the device, and clear existing contacts before applying the file:

```
$ adb shell am start -a wfc.voice.SYNC_CONTACTS --es contacts_url /sdcard/
 contacs.csv --ez reset true
```

To load a file stored on a network, and leave existing contacts:

```
$ adb shell am start -a wfc.voice.SYNC_CONTACTS --es contacts_url https://
example.com/voice/contacts/1234 --ez reset false
```

Import Contacts Template

This section provides a description and example of the CSV file template used to import contacts.

- The following fields are required:
  - `contactId`–Use a unique integer for each contact. If a contact already exists with the same contactId, the existing record is updated. To delete a contact, set this to a value less than zero.
  - `firstName`–First name
  - `lastName`–Last name
- Provide at least one of the following:
  - `cellNumber`–Cell phone number
  - `officeNumber`–Office phone number
  - `homeNumber`–Home phone number

Optional Fields

- `group`–List group names separated by semicolons.

- `ringtone`–Provide a ringtone for the contact using one of the following value types:
  - `Name`–Name of an existing Android ringtone
  - `URL`–Link to the downloadable music file
  - `Filename`–Path to the music file on the SD card
  - Empty string–Leave blank to use the default ringtone.
- `photo`–URL to a downloadable image file or the path to a file on the SD card. Supported image formats are PNG and JPG.

Table 1 Sample Import

group	contact Id	first Name	last Name	cell Number	office Number	home Number	ringtone	photo
Work; Home; Paint	1	Mike	Smith	2725	847-123- 4560	6540		/sdcard/ WFConnect / pic1.jpg
Work; Home	2	John	Page	2726		6541	sample.com / ringtone25.ogg	/sdcard/ WFConnect / pic2.jpg
Paint	3	Frank	Flyer	2727			/sdcard/ WFConnect / ring4.ogg	sample.com / photo.jpg
Electronics	4	Emilia	Kratzer	2728		6542		/sdcard/ WFConnect / pic4.jpg
	-5	Kate	Perry	2729				

NOTE: The column headings in the import CSV file must be one word. For example, `contact Id` should be `contactId`. The spaces in the sample are to format the table.

Phone State Monitoring

The Zebra Voice Client broadcasts its status using the following intent. This is an intent that the Zebra Voice Client sends. A third-party application can register to receive the intents.

Prerequisites

- Zebra Voice Client is loaded to the device.
- The third-party application must register to receive the intents as described in the Phone State Example.
- All versions of Zebra Voice Client 9.0 support these intents.

Intent Definition

Name	Description
Action	`wfc.voice.PHONE_STATE`
extras	
registration_state	PBX registration state `ACTIVE`|`ACTIVE_DND`|`CONNECTING`|`INACTIVE`|`EXTENSION_UNAVAILABLE`

Name	Description
state	Voice call state `IDLE|CALLING|RINGING|ACTIVE`
number	The phone number for the current session that is reported when the voice call state changes (optional).
line_id	The line number is reported when one of the line registers (optional).
line_extension	Line extension that is reported when one of the lines registers (optional).
line_registered	A boolean expression that is reported when one of the lines registers (optional).
suspended	A boolean expression that is reported when the state is `IDLE` because one of the sessions is suspended or on hold (optional).

NOTE: The suspended extra is only available in Zebra Voice version 9.0.21104 or later. The EXTENSION_UNAVAILABLE registration state is broadcasted solely when no extension is available for the specific department or PFM role chosen by the user.

Phone State Example

The following code example registers the phone state intent from a third-party app.

```
// Create a broadcast receiver
BroadcastReceiver mMessageReceiver = new BroadcastReceiver()
{
      @Override
      public void onReceive(Context context, Intent intent)
        {
            Log.i(TAG,
                  "Received PHONE_STATE from WFCVoice “
                + " registration_state=" +
 intent.getStringExtra("registration_state")
                + “ call state=" + intent.getStringExtra("state")
                + " number=" + intent.getStringExtra("number")
                + " line_id=" + intent.getStringExtra("line_id")
                + " line_extension=" +
 intent.getStringExtra("line_extension")
                + " line_registered=" +
 intent.getBooleanExtra("line_registered" false)
                );
       }
};

//Register broadcast receiver in the Activity
IntentFilter mMessageReceiver = new IntentFilter();
requestFilter.addAction(“wfc.voice.PHONE_STATE”);
registerReceiver(mMessageReceiver, requestFilter);
```

Display Data Consent

All the users of the Zebra Voice Client must accept/decline the data consent agreement before using the Voice Client application.

Prerequisites

- Zebra Voice Client must be installed on the devices.
- The Zebra Voice Client version 9.0.23305+ is required.

Table 2 Intent Definition

Name	Description
Action	com.symbol.wfc.voice/com.zebra.wfc.dialer.DialerActivity
Intent Type	StartActivity
Extra 0	This parameter can be used to disable or enable the display of the Data Consent Screen after installation. This parameter is optional.
Type	Boolean
Name	showDisclosure
Value	True: Displays the Data Consent screen. False: Does not display the Data Consent screen.

ADB Command to Display the Data Consent ShowDisclosure

```
adb shell am start -n com.symbol.wfc.voice/
com.zebra.wfc.dialer.DialerActivity  --ez showDisclosure "true"
```

ADB Command to Bypass the Data Consent ShowDisclosure

```
adb shell am start -n com.symbol.wfc.voice/
com.zebra.wfc.dialer.DialerActivity  --ez showDisclosure "false"
```

General Rules Related to Data Consent

The Data Consent screen appears when the Zebra Voice Client for Android is activated first time. The app does not work until the consent is accepted/declined.

- When the user opens Voice Client application for the first time after installation, the Data Consent screen displays. The App allows the user to proceed to the Voice Client for Android screen only after the Data Consent is accepted. If the user declines it, the application is closed there.
- If the user reopens the application, the Data Consent screen is displayed again. The user must accept the Data Consent to proceed further to configure the Voice Client for Android.
- Similarly, the showDisclosure must be set to true in ADB Commands to proceed to get the Voice Client Configuration screen.

Bypass Data Consent Using StageNow

An administrator can use StageNow to configure to bypass the Data Consent windows of the Zebra Voice Client.

Example:

```
<wap-provisioningdoc>
 <characteristic version="10.5" type="Intent">
 <parm name="Action" value="StartActivity" />
 <parm name="ActionName" value="com.symbol.wfc.voice/
com.zebra.wfc.dialer.DialerActivity" />     <parm name="Package"
 value="com.symbol.wfc.voice" />
 <parm name="Class" value="com.zebra.wfc.dialer.DialerActivity" />
 <characteristic type="Extra">
 <parm name="ExtraType" value="boolean" />
 <parm name="ExtraName" value="showDisclosure" />
 <parm name="ExtraValue" value="false" />
 </characteristic> </characteristic>
</wap-provisioningdoc>
```

Pinboard Integration - Intents/APIs to Share Call Data

Generic Intent to Get Call Data on Demand

The Voice Client exposed the following intent com.zebra.wfc.voice.GET_INFO to get the specific category of call-data count from the Voice Client on-demand basis. Any third-party applications need to follow the following-mentioned steps to get the count from the Voice Client.

Code Snippet

```
Intent i= new Intent("com.zebra.wfc.voice.GET_INFO"); i.putExtra("call_type",
 "MISSED"); //Same can be done for other call_type as well sendBroadcast(i);
```

1. Register and Listen to the Following Broadcast

```
IntentFilter call_info= new IntentFilter();
call_info.addAction("com.zebra.wfc.voice.CALL_INFO");
registerReceiver(mReceiver,call_info);

public void onReceive(Context context, Intent intent) {
    String action= intent.getAction();
    if(action.equals("com.zebra.wfc.voice.CALL_INFO")) {
       if (intent.hasExtra("MISSED")) { //Same can be done for other
 call_type as well
          count = intent.getIntExtra("MISSED", 0);
          Log.d("Call data:- "+count)
       } // ERROR handling
```

```
    if(intent.hasExtra("ERROR")){
       String error= "ERROR: " + intent.getStringExtra("ERROR");
       Log.d(TAG,error);
          }
      }
}
```

The above example shows that after following the above steps, any 3rd party application can trigger an intent to receive a specific call count from Voice Client.

Generic Intent Broadcast Call information

The Voice Client exposed the intent com.zebra.wfc.voice.CALL_INFO_BROADCAST to get the specific category call data count from the Voice Client whenever there is a change in call history information.

Any third-party applications need to follow the following steps to get the count from the Voice Client.

Code Snippet to Register CALL_INFO_BROADCAST

```
IntentFilter call_info = new IntentFilter();
 call_info.addAction("com.zebra.wfc.voice.CALL_INFO_BROADCAST");
registerReceiver(mReceiver,call_info);
```

Code Snippet to Listen to the CALL_INFO_BROADCAST

```
public void onReceive(Context context, Intent intent) {
   String action= intent.getAction();
   if(action.equals("com.zebra.wfc.voice.CALL_INFO_BROADCAST")) {
      if (intent.hasExtra("MISSED") { //Same can be done for other Extras as
 well
      count = intent.getIntExtra("MISSED", 0);
      Log.d("Call data:- "+count)
       }
    }
 }
```

Generic Intent to Open Voice History Fragment

The Voice Client exposed the following intent calllog.open_history to open the Voice Client History Fragment with the specific category of filter selected option on-demand basis. Any third-party applications need to follow the following steps to request for opening Voice Client History Fragment with a specific filter set.

Code Snippet for calllog.open_history Intent i= new

```
Intent("com.zebra.wfc.voice.calllog.open_history");
i.putExtra("call_type", "MISSED"); //Same can be done for other call_type as
 well
sendBroadcast(i);
```

Implementing Shared URI Reporting Via Intent

A new intent has been introduced that provides additional information on registration use cases. Third-party applications can subscribe to this intent to access detailed insights about these cases.

The new intent is activated when:

- A profile is applied to the Voice Client.
- A reload is executed on the Voice Client.
- The Voice Client is signed out/logged out.
- The Profile Client executes a role switch, applying a new profile to the Voice Client.

In all the aforementioned use cases, the intent records which extension is deregistered and which extension the Voice Client is registered with. Additionally, it records if there is a failure during the registration of an extension.

Intent Definition

Extra Parameter	Name	Description
Action		`wfc.voice.REGISTRATION_INFO`
Extras		
Extra 0	action_name	The `action_name` extra contains one of the following strings: • `switch_role` - This is returned if the `switch_role` operation is executed. • `reload` - This is returned if the `reload` operation is executed inside Voice Client. • `sign_out` - This is returned if the `sign_out` operation is executed inside Voice Client. • `initialized` - This is returned if the extension is initialized inside Voice Client. **NOTE:** • if `action_name` is `switch_role` or `reload` in the particular scenario only `current_extension_info` and `previous_extension_info` extra are broadcasted. • if `action_name` is `sign_out` or `initialized` the particular scenario only `extension_info` extra is broadcasted.
Extra 1	var_location	This `var_location` extra contains the shared profile URL, used to load the profile.
Extra 2	previous_extension_info	The `previous_extension_info` contains details about the previously registered extension, presented in a key-value format. The keys include `site_id`,

Extra Parameter	Name	Description
		`department`, `extension`, `sip_sipid`, `pbxType`, `extension_applied`, and `failure_reason`, with their corresponding values returned. The expected corresponding value for the following-mentioned keys: The keys `site_id`, `department`, `extension`, `sip_sipid`, `pbxType`, and `failure_reason`, values are any string. The `extension_applied` value either 0 or 1(integer) if 0 in `failure_reason` is broadcasted.
Extra 3	current_extension_info	The current_extension_info offers details about the currently registered extension in a key-value format. The keys include `site_id`, `department`, `extension`, `sip_sipid`, `pbxType`, and `failure_reason`, values are any string. The expected corresponding value for the following-mentioned keys: The keys `site_id`, `department`, `extension`, `sip_sipid`, `pbxType`, and `failure_reason`, values are any string. The `extension_applied` value is either 0 or 1 (integer) if 0 in `failure_reason` is broadcasted.
Extra 4	extension_info	The `extension_info` contains details about the registered extension in a key-value format. The keys include `site_id`, `department`, `extension`, `sip_sipid`, `pbxType`, `extension_applied`, and `failure_reason`, with their corresponding values are returned. The expected corresponding value for the following-mentioned keys: The keys `site_id`, `department`, `extension`, `sip_sipid`, `pbxType`, and `failure_reason`, values can be any string. The `extension_applied` value either 0 or 1(integer) if 0 in `failure_reason` is broadcasted.

Code Snippet Example

```
IntentFilter intent = new IntentFilter();
intent.addAction("wfc.voice.REGISTRATION_INFO");
registerReceiver(mReceiver,intent);
private final BroadcastReceiver broadcastReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {
if (intent.getAction() != null && intent.getAction().
```

```
   equals ("wfc.voice.REGISTRATION_INFO")) {
   String intentAction = intent.getAction();
   Bundle intentExtras = intent.getExtras();
if (intentExtras != null) {
String varLocation = intentExtras.getString("var_location");
String actionName = intentExtras.getString("action_name");
if (actionName != null) {
if (actionName.equals("reload")) {
String previousExtensionInfo = intentExtras.getString
("previous_extension_info");
String currentExtensionInfo = intentExtras.getString
("current_extension_info");
Log.d("RegistrationInfo: ", "IntentAction: " + intentAction +
                   "\nactionName: " + actionName + "\nvarLocation: "
+ varLocation + "\nPrevious Extension Info: "
+ previousExtensionInfo + "\nCurrent Extension Info: " +
 currentExtensionInfo);
           } else {
String extensionInfo = intentExtras.getString("extension_info");
Log.d("RegistrationInfo: ", "IntentAction: " + intentAction +
                 "\nactionName: " + actionName + "\nvarLocation: "
              + varLocation + "\nExtension Info: " + extensionInfo);
        try {
JSONArray extensionInfoJSONArray = new JSONArray(extensionInfo);
 for (int i = 0; i < extensionInfoJSONArray.length(); i++) {
   JSONObject line = (JSONObject) extensionInfoJSONArray.get(i);
         if (line != null) {
 String siteId = line.getString("site_id");
 String department = line.getString("department");
 String sipSipId = line.getString("sip_sipid");
 String pbxType = line.getString("pbxType");
 int extensionApplied = line.getInt("extension_applied");
 Log.d("LineInfo", "siteId: " + siteId + ", department: "
                   + department + ", sipSipId: " + sipSipId + ",
  pbxType: " + pbxType + ", extensionApplied: "
     + extensionApplied);
     if (extensionApplied == 0) {
       String failureReason =  line.getString("failure_reason")
       Log.d("LineInfo", "failureReason: " + failureReason);
                           }
                       }
                   }
               } catch (JSONException e) {
                   throw new RuntimeException(e);
               }
           }
       }
    }
}
}
};
```

Headless Mode

Configure the Zebra Voice as a background service (headless mode) to allow third-party apps to control voice calls remotely and display the call status on their own user interface (UI) screens.

When headless mode is enabled:

- The Zebra Voice home dashboard screen is hidden. Launching the Zebra Voice app displays the dial pad or contacts screen.
- Access Zebra Voice settings by swiping down from the top of the screen and touching the Zebra Voice notification.

Enabling Headless Mode

You can enable headless mode in the XML configuration file or in the app.

- In the WFConnect XML configuration file, set the `headless_mode` parameter to `true`.
- From the Zebra Voice app, touch the three-line menu at the top of the screen and go to **Settings** > **Advanced Settings** > **UI Settings** > **Headless Mode**.

Voice Connector

WFCVoiceConnector is an Android library that allows third-party apps to establish a service connection with a voice app running in headless mode.

The `WFCVoiceConnector.aar` library is part of WFCDemo project.

WFCDemo

WFCDemo is a sample project for an Android app that uses the WFCVoiceConnector library. The app has a UI screen that allows a user to initiate and control voice calls remotely through the service connector project.

The WFCDemo project is available at github.com/ZebraDevs/wfc-voice-demo/.