RestComm User Guide

The Guide to the RestComm

Jean Deruelle <jean.deruelle@gmail.com>
Thomas Quintana <quintana.thomas@gmail.com>

RestComm User Guide: The Guide to the RestComm

by Jean Deruelle and Thomas Quintana

Abstract

RestComm is a carrier-grade open source platform that provides developers the tools to integrate fax, voice, and SMS functionality in to their own applications with ease. RestComm is designed to have 100% compatibility with Twilio's APIs allowing easy porting between platforms. Furthermore, the RestComm platform is built on top of the industry leading Mobicents Sip Servlet Container and Mobicents Media Server providing the robustness and performance these platforms are already known to deliver.

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the Liberation Fonts [https://fedorahosted.org/liberation-fonts/] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight key caps and key-combinations. For example:

> To see the contents of the file `my_next_bestselling_novel` in your current working directory, enter the **cat my_next_bestselling_novel** command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key cap, all presented in Mono-spaced Bold and all distinguishable thanks to context.

Key-combinations can be distinguished from key caps by the hyphen connecting each part of a key-combination. For example:

> Press **Enter** to execute the command.
>
> Press **Ctrl**+**Alt**+**F1** to switch to the first virtual terminal. Press **Ctrl**+**Alt**+**F7** to return to your X-Windows session.

The first sentence highlights the particular key cap to press. The second highlights two sets of three key caps, each set pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in `Mono-spaced Bold`. For example:

> File-related classes include `filesystem` for file systems, `file` for files, and `dir` for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialogue box text; labelled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose System > Preferences > Mouse from the main menu bar to launch Mouse Preferences . In the Buttons tab, click the Left-handed mouse check box and click Close to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a gedit file, choose Applications > Accessories > Character Map from the main menu bar. Next, choose Search > Find… from the Character Map menu bar, type the name of the character in the Search field and click Next . The character you sought will be highlighted in the Character Table . Double-click this highlighted character to place it in the Text to copy field and then click the Copy button. Now switch back to your document and choose Edit > Paste from the gedit menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in Proportional Bold and all distinguishable by context.

Note the > shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select Mouse from the Preferences sub-menu in the System menu of the main menu bar' approach.

`Mono-spaced Bold Italic` or *`Proportional Bold Italic`*

Whether Mono-spaced Bold or Proportional Bold, the addition of Italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **ssh *username* @ *domain.name*** at a shell prompt. If the remote machine is `example.com` and your username on that machine is john, type **ssh john@example.com** .
>
> The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the `/home` file system, the command is **mount -o remount /home** .
>
> To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release*** .

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as a *server-pool* . Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules* (*MPMs*). Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

1.2. Pull-quote Conventions

Two, commonly multi-line, data types are set off visually from the surrounding text.

Output sent to a terminal is set in `Mono-spaced Roman` and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in `Mono-spaced Roman` but are presented and highlighted as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }

}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

A note is a tip or shortcut or alternative approach to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring Important boxes won't cause data loss but may cause irritation and frustration.

Warning

A Warning should not be ignored. Ignoring warnings will most likely cause data loss.

Chapter 1. Introduction to TelScale RestComm

TelScale RestComm is a Communications Platform to rapidly build fax, voice, and text messaging applications, using your existing web development skills.

It's a complete turnkey Telco 2.0 SaaS solution ready to deploy in your cloud based on the Twilio [www.twilio.com] simple and effective APIs and 100% Open Source built on the next generation Cloud Communications platform TelScale.

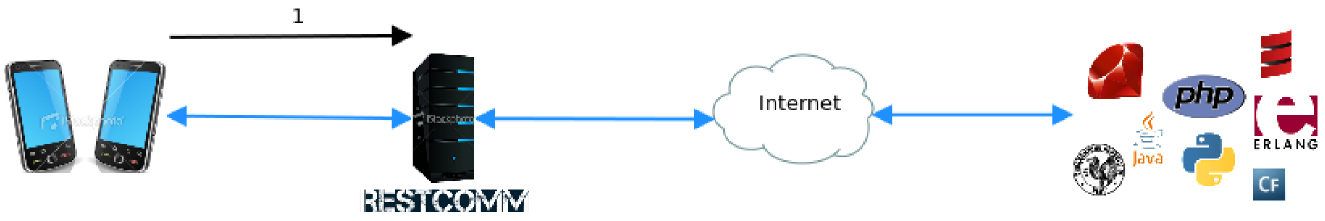
1.1. How it Works

In order to demonstrate how TelScale RestComm works we will go over an application that instructs TelScale RestComm to answer a phone call, say "Hello World" and finally hang up the call.

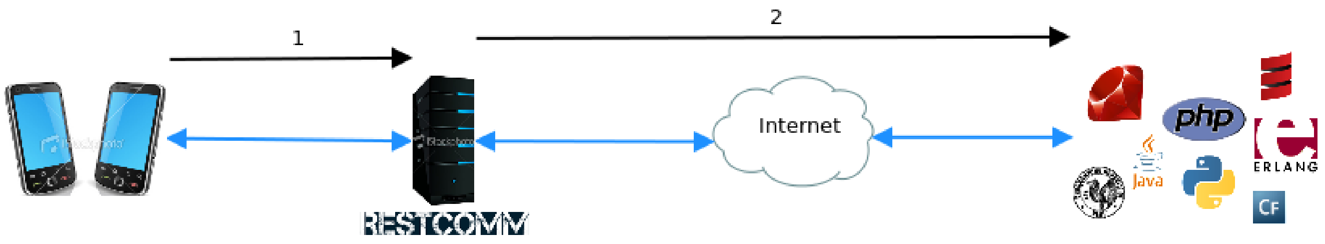
Note

A more thorough explanation of the TelScale RestComm Markup Language is available in Chapter 3. TelScale RestComm Markup Language

In this particular example, the first thing that must happen is that RestComm receives a call.

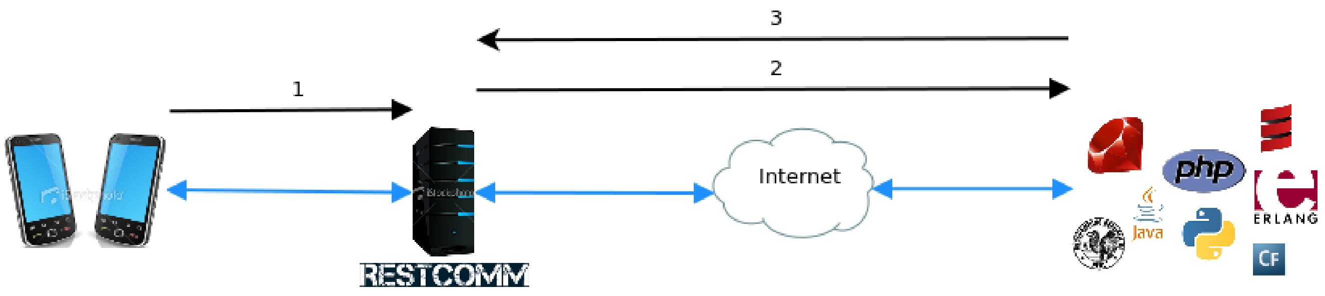


After RestComm can confirm that the call is destined for an application that it handles. RestComm calls out to your application for instructions on how to handle the new incoming call. Keep in mind that when RestComm calls out to your application it provides useful information such as from what number the call was dialed, to what number the call is destined, caller ID information, etc.

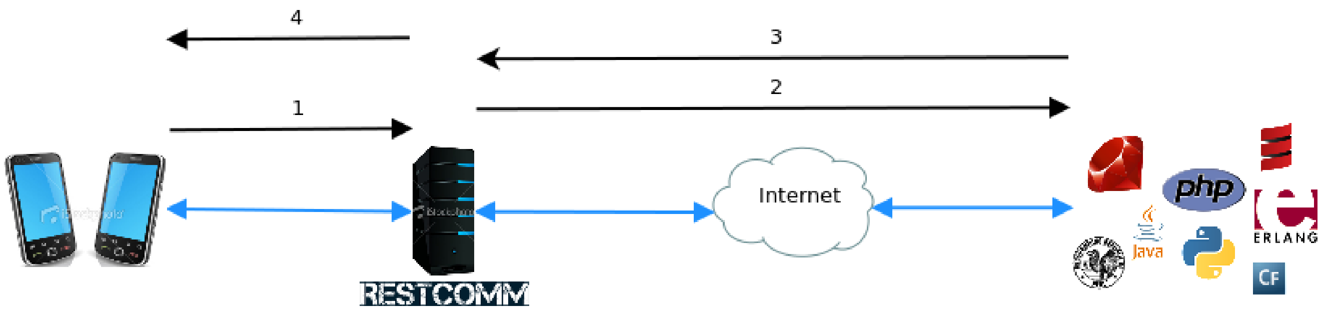


Once your application responds with instructions on how to handle the call, RestComm gets busy executing the provided instruction set.

```
<Response>
  <Say>Hello World!</Say>
</Response>
```



In this case RestComm synthesizes the text to speech, says "Hello World!" to the caller and hangs up.



1.2. Getting Started

Okay... Lets get started! In this chapter we will go over the steps necessary to deploy and run RestComm on the local host for development. First we will discuss how to get started using the binary distribution and for those more adventurous of you we will then cover building from the source code.

1.2.1. Requirements

RestComm is built as a SIP Servlet Application and has a couple of dependencies which are listed below.

Container (Required). TelScale RestComm supports TelScale Sip Servlets (JBoss AS5 - Tomcat 6.x - Tomcat 7.x)

Media Server (Required). TelScale RestComm supports Mobicents Media Server 3.0.0

1.3. Using a binary distribution

The easiest way to get started with RestComm is to download and unzip a binary distribution that includes Mobicents Sip Servlet Container + Mobicents Media Server + RestComm to your local machine.

JBoss AS Binary Distribution. In the next few steps we will deploy the RestComm binary that comes with the JBoss container. This process is very simple, it only requires a few steps: First, we download the zip file. Second, we unzip it. Third, we launch the Mobicents Media Server. Fourth and finally, we launch the JBoss container that already has RestComm pre-deployed.

We will start by downloading the latest binary snapshot and unzipping it to the curring working directory.

```
  ~]$ wget https://mobicents.ci.cloudbees.com/job/RestComm/
lastSuccessfulBuild/artifact/restcomm-saas-jboss-1.0.0.BETA2-SNAPSHOT.zip
  ~]$ unzip restcomm-saas-jboss-1.0.0.BETA2-SNAPSHOT.zip
```

Okay now lets start the Mobicents Media Server and the JBoss Container.

```
  ~]$ cd restcomm-saas-jboss-1.0.0.BETA2-SNAPSHOT/mobicents-media-server/
bin/
  ~]$ chmod +x run.sh
  ~]$ ./run.sh &
  ~]$ cd ../../bin
  ~]$ ./run.sh &
```

Tomcat Binary Distribution. For those of you who need a more light-weight container we will also cover using a Tomcat based distribution in the following steps.

```
  ~]$ wget https://mobicents.ci.cloudbees.com/job/RestComm/
lastSuccessfulBuild/artifact/restcomm-saas-tomcat-1.0.0.BETA2-SNAPSHOT.zip
  ~]$ unzip restcomm-saas-tomcat-1.0.0.BETA2-SNAPSHOT.zip
```

Okay now lets start the Mobicents Media Server and the Tomcat Container.

```
  ~]$ cd restcomm-saas-tomcat-1.0.0.BETA2-SNAPSHOT/mobicents-media-server/
bin/
  ~]$ chmod +x run.sh
  ~]$ ./run.sh &
  ~]$ cd ../../bin
  ~]$ ./startup.sh
```

1.4. Building from source code

In this section we will download the source code from the Git repository build RestComm and deploy it. This process has a few steps which we will explain as we execute them. Finally, we highly recommend using the containers supplied with the binary downloads as they have several extra libs in the lib folder to resolve database driver dependencies.

We will start by downloading the source code from the Git repo on Google Code.

```
~]$ git clone https://code.google.com/p/restcomm/
```

Once we have the code we will build it using maven and run the regression test suite.

```
~]$ cd restcomm
~]$ mvn clean install
~]$ cd restcomm.core
~]$ mvn clean install
```

The compiled distribution will be located in the `restcomm` directory and the archived WAR file wile be located in the `restcomm.war` file. Both of these will be located in the `target` directory inside the `restcomm.core` directory which should be the current working directory. To deploy the war file you would first remove restcomm if it's already deployed in the container and then redeploy it by copying the WAR file to the appropriate directory.

Deploying RestComm to the JBoss Container. In order to deploy RestComm to the JBoss Container we will first remove the version that comes pre-deployed and then we will copy our `restcomm` directory with the compiled distribution to the `deploy` directory.

```
~]$ rm -r $JBOSS_HOME/server/default/deploy/restcomm.war
~]$ cp -r target/restcomm $JBOSS_HOME/server/default/deploy/restcomm.war
```

Deploying RestComm to the Tomcat Container. In order to deploy RestComm to the Tomcat Container we will first remove the version that comes pre-deployed and then we will copy our `restcomm` directory with the compiled distribution to the `webapps` directory.

```
   ~]$ rm -r $TOMCAT_HOME/webapps/restcomm
   ~]$ cp -r target/restcomm $TOMCAT_HOME/webapps/restcomm
```

1.5. Binding the Demo Application to a Phone Number.

Now we have RestComm deployed and we're ready to bind a phone number to the demo application and finally call the phone number to see what happens. In order to interact with RestComm we have provided a set of Restful APIs which are very similar to Twilio's Restful APIs. You will learn more about these APIs in a later chapter. We will be using Curl to interact with these APIs through out the rest of this guide.

Note

There are wrapper libraries available to interact with RestComm and facilitate interaction for many of the popular development platforms.

With the following Curl command we will bind the demo application located @ http://127.0.0.1:8080/restcomm/demo/hello-world.xml to phone number 1234.

```
   ~]$ curl --data "PhoneNumber=1234&VoiceUrl=http://127.0.0.1:8080/restcomm/
 demo/hello-world.xml&VoiceMethod=POST"
 http://
 ACae6e420f425248d6a26948c17a9e2acf:77f8c12cc7b8f8423e5c38b035249166@127.0.0.1:8080/
 restcomm/2012-04-24/Accounts/ACae6e420f425248d6a26948c17a9e2acf/
 IncomingPhoneNumbers.json
```

1.6. Making the First Call!

Finally, spin up your favorite softphone and dial sip:1234@127.0.0.1:5080

Keep in mind that before you continue you should configure your ASR, Fax, SMS, and TTS plugins in order to have everything function as you would expect it to.

Chapter 2. Configuration

In order for RestComm to function properly it must first be configured. In this chapter we will cover the settings for RestComm and it's respective plug-ins. In order to make RestComm easy to manage all of the configuration is done in only one file. The file is located at `RESTCOMM_HOME/WEB-INF/conf/restcomm.xml` and is composed of the sections that follow.

Note

RestComm provides a default set of plug-ins for the storage engine, fax service, SMS aggregator, automatic speech recognition, and speech synthesis. All these services are pluggable and RestComm is not limited to any service provider or software platform whom's services it consumes.

2.1. Runtime Settings

The runtime-settings are used by RestComm at runtime to customize it's behavior. A list of the runtime settings and a description is provided below.

Table 2.1. Runtime Settings

Element	Description
api-version	The version of the RestComm API that we will be executing by default.
beep-audio-file	The audio file to play for the 'beep' and 'playBeep' attributes.
reject-audio-file	The audio file to play for the <Reject> verb using the 'rejected' attribute.
alert-on-enter-file	The audio file to play when a participant enters a conference room.
alert-on-exit-file	The audio file to play when a participant exits a conference room.
ringback-audio-file	The audio file to play when a caller is waiting for a call to be bridged using the <Dial> verb.
conference-music-file	The default audio file to play in a conference room. This is equivalent to setting the waitUrl attribute for the <Conference> noun.
error-dictionary-uri	The root URI to the error dictionary. This dictionary contains descriptions and possible solutions to the error codes that RestComm will return as part of it's Notifications resource.

Element	Description
recordings-path	The path to the directory used to save recordings made by RestComm.
outbound-proxy-user	The user name used to authenticate with the outbound proxy. (Optional)
outbound-proxy-password	The password used to authenticated with the outbound proxy. (Optional)
outbound-proxy-uri	The SIP URI to the outbound proxy. Note: Do not prepend 'sip:' to the proxy uri. If necessary the port can be included, for example thomas@localhost:5080

Restcomm Resource Security. The security model is based on role based access control. RestComm defines a set of permissions that can be defined for each role. There are two predefined roles Administrator and Developer. The Developer role can be configured or removed all together depending on your needs but the Administrator role can not be changed. The Administrator role can not be modified or removed and it has access to every resource.

First we will define the list of permissions and what they mean.

Table 2.2. RestComm Permissions

Permission	Description
Create	The role has access to create a type resource.
Read	The role has access to read a type of resource.
Modify	The role has access to modify a type of resource.
Delete	The role has access to delete a type of resource.

These permissions apply to every resource exposed by the Restful APIs. Once a role is defined it can be specified for new Account resources that are created.

Note

If no role is specified when creating a new Account resource then the new Account will inherit the security role of the account that created it. If this role is the Administrator role the system may become compromised.

Wildcard Permission. The asterisk '*' is a wildcard that means grant all permissions and can be used in place of typing out all the permissions.

To see how this all comes together please check out the `restcomm.xml` configuration file.

2.2. Dao Manager

The data access object manager is used to gain access to the data store that RestComm will use.

The default data access object manager is based on MyBatis and provides access to RDBM systems. Alternatively, a MongoDB implementation is included.

Table 2.3. MyBatis Settings

Element	Description
configuration-file	The path to the mybatis.xml configuration file.
data-files	The path to the data files used to store the database tables.
sql-files	The path to the XML files containing the SQL statements that will be executed by the database.

Table 2.4. MongoDB Settings

Element	Description
host	The IP address or host name of MongoDB server.
port	The port on the MongoDB server to use.
database	The name of the database to use.
user	The user name used to authenticate with MongoDB (Optional).
password	The password used to authenticate with MongoDB (Optional).

2.3. Media Server Manager

The media server manager is responsible for all the media servers managed by RestComm. The media server manager is also responsible for load-balancing connections to the media servers that it manages. When configuring the media server manager multiple media servers can be defined. Below is a description of settings for each media server.

Table 2.5. Media Server Manager Settings

Property	Description
local-address	The local IP address for the MGCP stack.
local-port	The local port for the MGCP stack.

Property	Description
remote-address	The IP address for the media server.
remote-port	The port for the media server.

Note

Currently, load-balancing uses a naive round-robin implementation but in future implementations the algorithm will be based on media server load.

2.4. SMS Aggregator

The SMS aggregator is responsible for the handling of SMS messages on behalf of RestComm.

Requirements. The default SMS aggregator is Essendex. Therefore, an account must be created @ http://www.esendex.com [http://www.esendex.com/] before any text messages can be sent.

Below is a list of settings that must be configured to send text messages.

Table 2.6. SMS Aggregator Settings

Element	Description
user	The user name used to authenticate with Essendex.
password	The password used to authenticate with Essendex.
account	The account used to authenticate with Essendex.

2.5. Fax Service

The fax service sends faxes out on behalf of RestComm.

Requirements. The default fax service is provided by Interfax. Therefore, an account must be created @ http://www.interfax.net [http://www.interfax.net/] before any faxes can be sent.

Table 2.7. Fax Service Settings

Element	Description
user	The user name used to authenticate with Interfax.
password	The password used to authenticate with Interfax.

Element	Description
maximum-file-size	The maximum files size allowed to be delivered via fax.
timeout	The timeout period before an undeliverable fax is considered as failed.

2.6. Speech Recognizer

This speech recognizer turns speech in to text on behalf of Restcomm.

Requirements. The default speech recognizer uses the ASR service provided by iSpeech. Therefore, an account must be created @ http://www.ispeech.org [http://www.ispeech.org/] before any speech can be converted to text.

Table 2.8. Speech Recognizer Settings

Element	Description
api-key	The Web API key provide by iSpeech. This is used as an authentication token for the service.

2.7. Speech Synthesizer

The speech synthesizer turns text to speech on behalf of RestComm.

Requirements. The default speech synthesizer uses the TTS service provided by Acapela. Therefore, an account must be created @ http://www.acapela-vaas.com [http://www.acapela-vaas.com/] before any text can be converted to speech.

Table 2.9. Acapela Group Speech Synthesizer Settings

Element	Description
cache-path	The location on the file system to use for caching TTS results.
service-root	The acapela group service root URI.
application	The application used to authenticate with the acapela group.
login	The login used to authenticate with the acapela group.
password	The password used to authenticate with the acapela group.
speakers	A list of male and female speakers for different languages. These can be replaced

Element	Description
	with other alternatives provided by the acapela group.

Chapter 3. RestComm Markup Language

The RestComm Markup Language (RCML) is composed of a set of XML tags that can be used to instruct RestComm on how to handle an on-going phone call. RestComm applications are built by combining these XML verbs and nouns in a way that is sensible for a given set of application requirements. In this chapter we will discuss what a request from RestComm to your application looks like as well as what the response from your application should look like. Then, we will dive in to how each verb and noun in the XML instruction set is used.

3.1. RestComm Request

How RestComm Passes Data to Your Application. The way RestComm passes data to you application depends on the request method for the given URI. If the request method is GET then the data is passed in the query string or the part after the question mark. If the request method is POST then the data is sent a multi-part form data just like when a browser submits a form.

RestComm Voice Request. Any time RestComm makes a request to you applications it will include the following data as request parameters.

Table 3.1. Request Parameters

Parameter	Description
CallSid	The unique identifier for this call.
AccountSid	Your account id.
From	The phone number of the originator of the call.
To	The phone number of the call recipient.
CallStatus	The status of the call. The possible values are queued, ringing, in-progress, completed, busy, failed, and no-answer.
ApiVersion	The version of the RestComm API used to handle this call.
Direction	The direction of the call. The possible values are inbound and outbound-dial.
CallerName	The caller ID for the caller in the case of inbound calls.

3.2. Your Response

In your response to the request from RestComm you want to provide RCML that will instruct RestComm on how to handle the current call.

MIME Types. RestComm supports the MIME types described in the table below.

Table 3.2. Supported MIME Types

Parameter	Description
text/xml, application/xml	RestComm interprets the returned document as an XML instruction set.

Note

When your application returns the RCML document the root element of the document must always be <Response> or the parser will complain.

3.3. Say

The <Say> verb is used to synthesize text to speech and play it to the remote party. The voices supported depends on the TTS Service provider plug-in. Below are the voices for our default TTS service provider plug-in which uses Acapela Voice as a Service.

Table 3.3. Say Attributes

Name	Allowed Values	Default Value
voice	man, woman	man
language	bf, bp, en, en-gb,cf, cs, dan, fi es, fr, de, el, it, nl, no, pl, pt, ru, ar, ca, sv, tr	en
loop	integer > 1	1

voice. The 'voice' attribute allows you to select the gender of the voice used to synthesize the text to speech for playback.

language. The 'language' attribute allows you pick a specific language for speech synthesis. RestComm currently supports languages 'bf' (Belgium-French), 'bp' (Brazilian-Portugues), 'en' (English), 'en-gb' (British-English), 'cf' (Canadian-French), 'cs' (Czech), 'dan' (Dannish), 'fi' (Finnish), 'es' (Spanish), 'fr' (French), 'de' (German), 'el' (Greek), 'it' (Italian), 'nl' (Netherlands-Dutch), 'no' (Norwegian), 'pl' (Polish), 'pt' (Portuguese), 'ru' (Russian), 'ar' (Saudi-Arabia Arabic), 'ca' (Spain Catalan), 'sv' (Swedish), and 'tr' (Turkish).

loop. The 'loop' attribute specifies how many times you'd like the text repeated. Specifying '0' will cause the the <Say> verb to loop until the call is hung up.

Nesting. The <Say> verb can not have any other verbs or nouns nested. Only text.

Examples. For an example of how to use the <Say> verb see below.

```
<Response>
 <Say>Hello World</Say>
</Response>
```

3.4. Play

The <Play> verb is used to play an audio file to the remote party.

Table 3.4. Play Attributes

Name	Allowed Values	Default Value
loop	integer > 1	1

loop. The 'loop' attribute specifies how many times you'd like the audio file to be repeated. Specifying '0' will cause the the <Play> verb to loop until the call is hung up.

Table 3.5. Supported Audio Formats

MIME type	Description
audio/wav	wav format audio
audio/wave	wav format audio
audio/x-wav	wav format audio

Nesting. The <Play> verb can not have any other verbs or nouns nested. Only a URL.

Examples. For an example of how to use the <Play> verb see below.

```
<Response>
 <Play>http://foobar.com/demo.wav</Play>
</Response>
```

3.5. Gather

The <Gather> verb "gathers" digits that a caller enters into his or her telephone keypad. When the caller is done entering digits, RestComm submits that digits to the provided 'action' URL in an HTTP GET or POST request. If no input is received before timeout, <Gather> falls through to the next verb in the RestComm document. You may optionally nest <Say>, <Play>, and <Pause> verbs within a <Gather> verb while waiting for input. This allows you to read menu options to the caller while letting her enter a menu selection at any time. After the first digit is received the audio will stop playing.

Table 3.6. Gather Attributes

Name	Allowed Values	Default Value
action	relative or absolute URL	current document URL
method	GET, POST	POST
timeout	positive integer	5 seconds
finishOnKey	any digit, #, *	#
numDigits	integer >= 1	unlimited

action. The 'action' attribute takes an absolute or relative URL as a value. When the caller has finished entering digits RestComm will make a GET or POST request to this URL including the parameters below. If no 'action' is provided, RestComm will by default make a POST request to the current document's URL.

Table 3.7. Request Parameters

Parameter	Description
Digits	The digits the caller pressed, excluding the finishOnKey digit.

method. The 'method' attribute takes the value 'GET' or 'POST'. This tells RestComm whether to request the 'action' URL via HTTP GET or POST.

timeout. The 'timeout' attribute sets the limit in seconds that RestComm will wait for the caller to press another digit before moving on and making a request to the 'action' URL. For example, if 'timeout' is '10', RestComm will wait ten seconds for the caller to press another key before submitting the previously entered digits to the 'action' URL. RestComm waits until completing the execution of all nested verbs before beginning the timeout period.

finishOnKey. The 'finishOnKey' attribute lets you choose one value that submits the received data when entered. For example, if you set 'finishOnKey' to '#' and the user enters '1234#', RestComm will immediately stop waiting for more input when the '#' is received and will submit "Digits=1234" to the 'action' URL. Note that the 'finishOnKey' value is not sent. The allowed values are the digits 0-9, '#' , '*' and the empty string (set 'finishOnKey' to ''). If the empty string is used, <Gather> captures all input and no key will end the <Gather> when pressed. In this case RestComm will submit the entered digits to the 'action' URL only after the timeout has been reached. The value can only be a single character.

numDigits. The 'numDigits' attribute lets you set the number of digits you are expecting, and submits the data to the 'action' URL once the caller enters that number of digits.

Nesting. You can nest the following verbs within <Gather>: <Say>, <Play>, <Pause>

Examples. For an example of how to use the <Gather> verb see below.

```
<Response>
 <Gather action="handle-user-input.php" numDigits="1">
  <Say>Welcome to TPS.</Say>
  <Say>For store hours, press 1.</Say>
  <Say>To speak to an agent, press 2.</Say>
  <Say>To check your package status, press 3.</Say>
 </Gather>
 <!-- If customer doesn't input anything, prompt and try again. -->
 <Say>Sorry, I didn't get your response.</Say>
 <Redirect>handle-incoming-call.xml</Redirect>
</Response>
```

3.6. Record

The <Record> verb records the caller's voice and returns to you the URL of a file containing the audio recording.

Table 3.8. Record Attributes

Name	Allowed Values	Default Value
action	relative or absolute URL	current document URL
method	GET, POST	POST
timeout	positive integer	5
finishOnKey	any digit, #, *	1234567890*#
maxLength	integer greater than 1 with the number of seconds to wait	3600 (1 hour)
transcribe	true, false	false
transcribeCallback	relative or absolute URL	none
playBeep	true, false	true

action. The 'action' attribute takes an absolute or relative URL as a value. When recording is finished RestComm will make a GET or POST request to this URL including the parameters below. If no 'action' is provided, <Record> will default to requesting the current document's URL. After making this request, RestComm will continue the current call using the RCML received in your response. Any RCML verbs occuring after a <Record> are unreachable. There is one exception: if RestComm receives an empty recording, it will not make a request to the 'action' URL. The current call flow will continue with the next verb in the current RCML document.

Table 3.9. Request Parameters

Parameter	Description
RecordingUrl	The URL of the recorded audio.

Parameter	Description
RecordingDuration	The time duration of the recorded audio.
Digits	The digits the caller pressed, excluding the finishOnKey digit.

method. The 'method' attribute takes the value 'GET' or 'POST'. This tells RestComm whether to request the URL via HTTP GET or POST.

timeout. The 'timeout' attribute tells RestComm to end the recording after a number of seconds of silence has passed.

finishOnKey. The 'finishOnKey' attribute lets you choose a set of digits that end the recording when entered. For example, if you set 'finishOnKey' to '#' and the caller presses '#', RestComm will immediately stop recording and submit 'RecordingUrl', 'RecordingDuration', and the '#' as parameters in a request to the 'action' URL. The allowed values are the digits 0-9, '#' and '*'. Unlike <Gather>, you may specify more than one character as a 'finishOnKey' value.

maxLength. The 'maxLength' attribute lets you set the maximum length for the recording in seconds.

transcribe. The 'transcribe' attribute tells RestComm that you would like a text representation of the audio of the recording.

transcribeCallback. The 'transcribeCallback' attribute is used in conjunction with the 'transcribe' attribute. It allows you to specify a URL to which RestComm will make an asynchronous POST request when the transcription is complete. This is not a request for RCML and the response will not change call flow, but the request will contain the standard RCML request parameters as well as 'TranscriptionStatus', 'TranscriptionText', 'TranscriptionUrl' and 'RecordingUrl'. If 'transcribeCallback' is specified, then there is no need to specify 'transcribe=true'. It is implied. If you specify 'transcribe=true' without a 'transcribeCallback', the completed transcription will be stored for you to retrieve later (see the REST API Transcriptions section), but RestComm will not asynchronously notify your application.

Table 3.10. Request Parameters

Parameter	Description
TranscriptionText	Contains the text of the transcription.
TranscriptionStatus	The status of the transcription attempt: either 'completed' or 'failed'.
TranscriptionUrl	The URL for the transcription's REST API resource.
RecordingUrl	The URL for the transcription's source recording resource.

playBeep. The 'playBeep' attribute allows you to toggle between playing a sound before the start of a recording.

Nesting. The <Record> verb can not have any other verbs or nouns nested.

Examples. For an example of how to use the <Record> verb see below.

```
<Response>
 <Record maxLength="30"/>
</Response>
```

3.7. Fax

The <Fax> verb sends a fax to some a fax machine.

Table 3.11. Fax Attributes

Name	Allowed Values	Default Value
to	phone number	see below
from	phone number	see below
action	relative or absolute URL	none
method	GET, POST	POST
statusCallback	relative or absolute URL	none

to. The 'to' attribute takes a valid E.164 phone number as a value. RestComm will send a fax to this number. When sending a fax during an incoming call, 'to' defaults to the caller. When sending an fax during an outgoing call, 'to' defaults to the called party. The value of 'to' must be a valid phone number.

from. The 'from' attribute takes a valid E.164 phone number as an argument. When sending a fax during an incoming call, 'from' defaults to the calling party. When sending a fax during an outgoing call, 'from' defaults to the called party. The value of 'from' must be a valid phone number.

action. The 'action' attribute takes a URL as an argument. After processing the <Fax> verb, RestComm will make a GET or POST request to this URL with the form parameters 'FaxStatus' and 'FaxSid'. Using an 'action' URL, your application can receive synchronous notification that the message was successfully enqueued. If you provide an 'action' URL, RestComm will use the RCML received in your response to the 'action' URL request to continue the current call. Any RCML verbs occuring after a <Fax> which specifies an 'action' attribute are unreachable. If no 'action' is provided, <Fax> will finish and RestComm will move on to the next RCML verb in the document. If there is no next verb, RestComm will end the phone call.

Table 3.12. Request Parameters

Parameter	Description
FaxSid	The Sid for the Sms message.

Parameter	Description
FaxStatus	The current status of the Sms message. Either 'sent' or 'failed'.

method. The 'method' attribute takes the value 'GET' or 'POST'. This tells RestComm whether to request the 'action' URL via HTTP GET or POST. This attribute is modeled after the HTML form 'method' attribute.

statusCallback. The 'statusCallback' attribute takes a URL as an argument. When the fax is actually sent, or if sending fails, RestComm will make an asynchronous POST request to this URL with the parameters 'FaxStatus' and 'FaxSid'. Note, 'statusCallback' always uses HTTP POST to request the given url.

Table 3.13. Request Parameters

Parameter	Description
FaxSid	The Sid for the fax message.
FaxStatus	The current status of the fax. Either 'sent' or 'failed'.

Nesting. The <Fax> verb can not have any other verbs or nouns nested.

Examples. For an example of how to use the <Fax> verb see below.

```
<Response>
 <Fax>This is a test fax.</Fax>
</Response>
```

3.8. Sms

The <Sms> verb sends an SMS message to a phone number during a phone call.

Table 3.14. Sms Attributes

Name	Allowed Values	Default Value
to	phone number	see below
from	phone number	see below
action	relative or absolute URL	none
method	GET, POST	POST
statusCallback	relative or absolute URL	none

to. The 'to' attribute takes a valid E.164 phone number as a value. RestComm will send an SMS message to this number. When sending an SMS during an incoming call, 'to' defaults to the caller.

When sending an SMS during an outgoing call, 'to' defaults to the called party. The value of 'to' must be a valid phone number.

from. The 'from' attribute takes a valid E.164 phone number as an argument. When sending an SMS during an incoming call, 'from' defaults to the calling party. When sending an SMS during an outgoing call, 'from' defaults to the called party.

action. The 'action' attribute takes a URL as an argument. After processing the <Sms> verb, RestComm will make a GET or POST request to this URL with the form parameters 'SmsStatus' and 'SmsSid'. Using an 'action' URL, your application can receive synchronous notification that the message was successfully enqueued. If you provide an 'action' URL, RestComm will use the RCML received in your response to the 'action' URL request to continue the current call. Any RCML verbs occuring after an <Sms> which specifies an 'action' attribute are unreachable. If no 'action' is provided, <Sms> will finish and RestComm will move on to the next RCML verb in the document. If there is no next verb, RestComm will end the phone call.

method. The 'method' attribute takes the value 'GET' or 'POST'. This tells RestComm whether to request the 'action' URL via HTTP GET or POST. This attribute is modeled after the HTML form 'method' attribute.

statusCallback. The 'statusCallback' attribute takes a URL as an argument. When the SMS message is actually sent, or if sending fails, RestComm will make an asynchronous POST request to this URL with the parameters 'SmsStatus' and 'SmsSid'. Note, 'statusCallback' always uses HTTP POST to request the given url.

Table 3.15. Request Parameters

Parameter	Description
SmsSid	The Sid for the Sms message.
SmsStatus	The current status of the Sms message. Either 'sent' or 'failed'.

Nesting. The <Sms> verb can not have any other verbs or nouns nested.

Examples. For an example of how to use the <Sms> verb see below.

```
<Response>
 <Sms>Hello World!</Sms>
</Response>
```

3.9. Dial

The <Dial> verb connects the current caller to another phone. If the called party picks up, the two parties are connected and can communicate until one hangs up. If the called party does not pick up, if a busy signal is received, or if the number doesn't exist, the dial verb will finish.

Table 3.16. Dial Attributes

Name	Allowed Values	Default Value
action	relative or absolute URL	no default for <Dial>
method	GET, POST	POST
timeout	positive integer in seconds	30 seconds
timeLimit	positive integer (seconds)	14400 seconds (4 hours)
callerId	a valid phone number, or client identifier if you are dialing a <Client>.	Caller's callerId

action. The 'action' attribute takes a URL as an argument. When the dialed call ends, RestComm will make a GET or POST request to this URL including the parameters below. If you provide an 'action' URL, RestComm will continue the current call after the dialed party has hung up, using the RCML received in your response to the 'action' URL request. Any RCML verbs occuring after a <Dial> which specifies an 'action' attribute are unreachable. If no 'action' is provided, <Dial> will finish and RestComm will move on to the next RCML verb in the document. If there is no next verb, RestComm will end the phone call.

Table 3.17. Request Parameters

Parameter	Description
DialCallStatus	The outcome of the <Dial> attempt. See the DialCallStatus section below for details.
DialCallSid	The call sid of the new call leg. This parameter is not sent after dialing a conference.
DialCallDuration	The duration in seconds of the dialed call. This parameter is not sent after dialing a conference.

Table 3.18. DialCallStatus Values

Parameter	Description
completed	The called party answered the call and was connected to the caller.
busy	RestComm received a busy signal when trying to connect to the called party.
no-answer	The called party did not pick up before the timeout period passed.
failed	RestComm was unable to route to the given phone number. This is frequently caused by

Parameter	Description
	dialing a properly formated but non-existent phone number.
canceled	The call was canceled via the REST API before it was answered.

method. The 'method' attribute takes the value 'GET' or 'POST'. This tells RestComm whether to request the 'action' URL via HTTP GET or POST. This attribute is modeled after the HTML form 'method' attribute.

timeout. The 'timeout' attribute sets the limit in seconds that <Dial> waits for the called party to answer the call.

timelimit. The 'timeLimit' attribute sets the maximum duration of the <Dial> in seconds.

callerId. The 'callerId' attribute lets you specify the caller ID that will appear to the called party when RestComm calls. By default, when you put a <Dial> in your RCML response to RestComm's inbound call request, the caller ID that the dialed party sees is the inbound caller's caller ID. If you are dialing to a <Client>, you can set a client identifier as the callerId attribute. For instance, if you've set up a client for incoming calls and you are dialing to it, you could set the callerId attribute to client:thomas.

Nesting. You can nest the following nouns within <Dial>: <Number>, <Client>, <Conference>

Examples. For an example of how to use the <Dial> verb see below.

```
<Response>
 <Dial>1-444-555-666</Dial>
</Response>
```

3.9.1. Number

The <Number> noun specifies a phone number to dial. You can use multiple <Number> nouns within a <Dial> verb to simultaneously call all of them at once. The first call to pick up is connected to the current call and the rest are hung up.

Table 3.19. Number Attributes

Name	Allowed Values	Default Value
sendDigits	any digits	none
url	any url	none

sendDigits. The 'sendDigits' attribute tells RestComm to play DTMF tones when the call is answered. This is useful when dialing a phone number and an extension. RestComm will dial the number, and when the automated system picks up, send the DTMF tones to connect to the extension.

url. The 'url' attribute allows you to specify a url for a RCML document that will run on the called party's end, after he/she answers, but before the parties are connected. You can use this RCML to privately play or say information to the called party, or provide a chance to decline the phone call using <Gather> and <Hangup>. The current caller will continue to hear ringing while the RCML document executes on the other end. RCML documents executed in this manner are not allowed to contain the <Dial> verb.

Examples. For an example of how to use the <Number> noun see below.

```
<Response>
 <Dial>
  <Number sendDigits="wwww1234">1-444-555-6666</Number>
 </Dial>
</Response>
```

3.9.2. Client

The <Client> noun specifies a client identifier to dial. You can use multiple <Client> nouns within a <Dial> verb to simultaneously attempt a connection with many clients at once. The first client to accept the incoming connection is connected to the call and the other connection attempts are canceled.

Examples. For an example of how to use the <Client> none see below.

```
 <Response>
  <Dial>
   <Client>thomas</Client>
  </Dial>
 </Response>
```

3.9.3. Conference

The <Conference> noun allows you to connect to a named conference room and talk with the other callers who have also connected to that room. The name of the room is up to you and is namespaced to your account.

Table 3.20. Conference Attributes

Name	Allowed Values	Default Value
muted	true, false	false

Name	Allowed Values	Default Value
beep	true, false	true
startConferenceOnEnter	true, false	true
endConferenceOnExit	true, false	false
waitUrl	RCML url, empty string	default RestComm hold music
waitMethod	GET or POST	POST
maxParticipants	positive integer <= 40	40

muted. The 'muted' attribute lets you specify whether a participant can speak in the conference. If this attribute is set to 'true', the participant will only be able to listen to people in the conference.

beep. The 'beep' attribute lets you specify whether a notification beep is played to the conference when a participant joins or leaves the conference.

startConferenceOnEnter. This attribute tells a conference to start when this participant joins the conference, if it is not already started. If this is false and the participant joins a conference that has not started, they are muted and hear background music until a participant joins where startConferenceOnEnter is true. This is useful for implementing moderated conferences.

endConferenceOnExit. If a participant has this attribute set to 'true', then when that participant leaves, the conference ends and all other participants drop out. This is useful for implementing moderated conferences that bridge two calls and allow either call leg to continue executing RCML if the other hangs up.

waitUrl. The 'waitUrl' attribute lets you specify a URL for music that plays before the conference has started. The URL may be a WAV or a RCML document that uses <Play> or <Say> for content. This defaults to a selection of Creative Commons licensed background music, but you can replace it with your own music and messages. If the 'waitUrl' responds with RCML, RestComm will only process <Play>, <Say>, and <Redirect> verbs. If you do not wish anything to play while waiting for the conference to start, specify the empty string (set 'waitUrl' to '').

waitMethod. This attribute indicates which HTTP method to use when requesting 'waitUrl'. It defaults to 'POST'. Be sure to use 'GET' if you are directly requesting static audio files such as WAV files so that RestComm properly caches the files.

maxParticipants. This attribute indicates the maximum number of participants you want to allow within a named conference room. The default maximum number of participants is 40. The value must be a positive integer less than or equal to 100.

Examples. For an example of how to use the <Conference> noun see below.

```
<Response>
 <Dial>
  <Conference>1234</Conference>
 </Dial>
```

```
</Response>
```

3.9.4. Uri

The <Uri> noun specifies a SIP URI to dial. You can use multiple <Uri> nouns within a <Dial> verb to simultaneously attempt a connection with many user agents at once. The first user agent to accept the incoming connection is connected to the call and the other connection attempts are canceled.

Examples. For an example of how to use the <Uri> none see below.

```
<Response>
 <Dial>
  <Uri>sip:thomas@127.0.0.1:5080</Uri>
 </Dial>
</Response>
```

3.10. Hangup

The <Hangup> verb ends a call.

Nesting. The <Hangup> verb can not have any other verbs or nouns nested.

Examples. For an example of how to use the <Hangup> verb see below.

```
<Response>
 <Hangup/>
</Response>
```

3.11. Redirect

The <Redirect> verb transfers control of a call to the RCML at a different URL. All verbs after <Redirect> are unreachable and ignored.

Table 3.21. Redirect Attributes

Name	Allowed Values	Default Value
method	GET, POST	POST

method. The 'method' attribute takes the value 'GET' or 'POST'. This tells RestComm whether to request the URL via HTTP GET or POST.

Nesting. The <Redirect> verb can not have any other verbs or nouns nested.

Examples. For an example of how to use the <Redirect> verb see below.

```
<Response>
 <Redirect>http://foobar.com/instructions</Redirect>
</Response>
```

3.12. Reject

The <Reject> verb rejects an incoming call to your RestComm endpoint. This is useful for blocking unwanted calls. If the first verb in a RCML response is <Reject>, RestComm will not pick up the call. The call ends with a status of 'busy' or 'no-answer', depending on the verb's 'reason' attribute. Any verbs after <Reject> are unreachable and ignored.

Table 3.22. Reject Attributes

Name	Allowed Values	Default Value
reason	rejected, busy	rejected

reason. The reason attribute takes the values "rejected" and "busy." This tells RestComm what message to play when rejecting a call. Selecting "busy" will play a busy signal to the caller, while selecting "rejected" will play a standard not-in-service response.

Nesting. The <Reject> verb can not have any other verbs or nouns nested.

Examples. For an example of how to use the <Reject> verb see below.

```
<Response>
 <Reject reason="busy"/>
</Response>
```

3.13. Pause

The <Pause> verb waits silently for a specific number of seconds. If <Pause> is the first verb in a RCML response, RestComm will wait the specified number of seconds before picking up the call.

Table 3.23. Pause Attributes

Name	Allowed Values	Default Value
length	integer > 0	1 second

length. The 'length' attribute specifies how many seconds RestComm will wait silently before continuing on.

Nesting. The <Pause> verb can not have any other verbs or nouns nested.

Examples. For an example of how to use the <Pause> verb see below.

```
<Response>
 <Pause length="5"/>
</Response>
```

Chapter 4. Restful APIs

The RestComm Restufl APIs are very similar to Twilio's and allow you to query meta-data about your account, phone numbers, calls, text messages, and recordings. Through the Restful API you can also start outbound calls and send text messages.

Resource Encoding. When an HTTP client makes a request to RestComm for a resource the HTTP client has the option to pick which encoding is used for the response. Currently, RestComm supports XML and JSON encoding. The default encoding is XML but to receive the resource in JSON just append .json to the end of the resource name.

4.1. Accounts

Accounts and subaccounts are useful for things like segmenting phone numbers and usage data for your users and controlling access to data.

Account Resource URI. /2012-04-24/Accounts/{AccountSid}

Table 4.1. Resource Properties

Property	Description
Sid	A string that uniquely identifies this account.
DateCreated	The date that this account was created.
DateUpdated	The date that this account was last updated.
FriendlyName	A description of this account, up to 64 characters long. By default the FriendlyName is your email address.
Status	The status of this account. Possible values are active, suspended, and closed.
AuthToken	The authorization token for this account. This should not be shared.
Uri	The URI for this account, relative to http://localhost:port/restcomm.

4.1.1. Supported Operations

HTTP GET. Returns the representation of an Account resource, including the properties above.

HTTP POST/PUT. Modifies an Account resource and returns the representation, including the properties above. Below you will find a list of optional parameters.

Table 4.2. Request Parameters

Parameter	Description
FriendlyName	A description of this account, up to 64 characters long.
Status	The status of this account. Possible values are active, suspended, and closed.
Password	A password that will be used to generate the AuthToken for the new Account resource.

4.1.2. Account List Resource

Account List Resource URI. /2012-04-24/Accounts

4.1.2.1. Supported Operations

HTTP GET. Returns the list representation of all the Sub-Account resources for this Account, including the properties above.

HTTP POST. Creates a new Sub-Account and returns the representation of the Sub-Account resource, including the properties above. Below you will find a list of required and optional parameters.

Table 4.3. Request Parameters

Parameter	Description
EmailAddress(Required)	The email address to use for this account.
FriendlyName	A description of this account, up to 64 characters long. Default, is your email address.
Status	The status of this account. Default is active, possible values are active, suspended, and closed.
Password(Required)	A password that will be used to generate the AuthToken for the new Account resource.
Role(Required)	The security role that this Account resource will use. If no role is provided then the role of the account resource creating this will be inherited to the new Account resource and may compromise the system.

4.2. IncomingPhoneNumbers

An IncomingPhoneNumber instance resource represents a RestComm phone number.

IncomingPhoneNumber Resource URI. /2012-04-24/Accounts/{AccountSid}/IncomingPhoneNumbers/{IncomingPhoneNumberSid}

Binding a Phone Number to an Application. When using RestComm the way to bind a phone number to an application is be creating a new IncomingPhoneNumber resource and providing the VoiceUrl to your application.

Table 4.4. Resource Properties

Property	Description
Sid	A string that uniquely identifies this incoming phone number.
DateCreated	The date that this incoming phone number was created.
DateUpdated	The date that this incoming phone number was last updated.
FriendlyName	A formatted version of this phone number.
AccountSid	The unique id of the Account that owns this phone number.
PhoneNumber	The incoming phone number in E.164 format ex. +2223334444
ApiVersion	Calls to this phone number will create a new RCML session with this API version.
VoiceCallerIdLookup	Look up the caller's caller-ID name. Either true or false.
VoiceUrl	The URL RestComm will request when this phone number receives a call.
VoiceMethod	The HTTP method RestComm will use when requesting the above Url. Either GET or POST.
VoiceFallbackUrl	The URL that RestComm will request if execution of VoiceUrl fails for any reason.
VoiceFallbackMethod	The HTTP method RestComm will use when requesting the VoiceFallbackUrl. Either GET or POST.
StatusCallback	The URL that RestComm will request to pass status parameters (such as the call state) to your application.
StatusCallbackMethod	The HTTP method RestComm will use to make requests to the StatusCallback URL. Either GET or POST.

Property	Description
SmsUrl	The URL that RestComm will request when receiving an incoming SMS message to this number. This may not be supported. Please consult with your DID provider.
SmsMethod	The HTTP method RestComm will use when making requests to the SmsUrl. Either GET or POST.
SmsFallbackUrl	The URL that RestComm will request if SmsUrl fail for any reason. Please see SmsUrl as this feature may not be supported.
SmsFallbackMethod	The HTTP method RestComm will use when making requests to SmsFallbackUrl. Either GET or POST.
Uri	The URI for this incoming phone number, relative to http://localhost:port/restcomm.

4.2.1. Supported Operations

HTTP GET. Returns the representation of an IncomingPhoneNumber resource, including the properties above.

HTTP POST/PUT. Modifies an IncomingPhoneNumber resource and returns the representation, including the properties above. Below you will find a list of optional parameters.

Table 4.5. Request Parameters

Parameter	Description
FriendlyName	A formatted version of this phone number.
ApiVersion	Calls to this phone number will create a new RCML session with this API version.
VoiceCallerIdLookup	Look up the caller's caller-ID name. Either true or false.
VoiceUrl	The URL RestComm will request when this phone number receives a call.
VoiceMethod	The HTTP method RestComm will use when requesting the above Url. Either GET or POST.
VoiceFallbackUrl	The URL that RestComm will request if execution of VoiceUrl fails for any reason.

Parameter	Description
VoiceFallbackMethod	The HTTP method RestComm will use when requesting the VoiceFallbackUrl. Either GET or POST.
StatusCallback	The URL that RestComm will request to pass status parameters (such as the call state) to your application.
StatusCallbackMethod	The HTTP method RestComm will use to make requests to the StatusCallback URL. Either GET or POST.
SmsUrl	The URL that RestComm will request when receiving an incoming SMS message to this number. This may not be supported. Please consult with your DID provider.
SmsMethod	The HTTP method RestComm will use when making requests to the SmsUrl. Either GET or POST.
SmsFallbackUrl	The URL that RestComm will request if SmsUrl fail for any reason. Please see SmsUrl as this feature may not be supported.
SmsFallbackMethod	The HTTP method RestComm will use when making requests to SmsFallbackUrl. Either GET or POST.

HTTP DELETE. Releases an IncomingPhoneNumber from the user's Account.

4.2.2. IncomingPhoneNumber List Resource

IncomingPhoneNumber List Resource URI. /2012-04-24/Accounts/{AccountSid}/IncomingPhoneNumbers

4.2.2.1. Supported Operations

HTTP GET. Returns the list representation of all the IncomingPhoneNumber resources for this Account, including the properties above.

HTTP POST. Creates a new IncomingPhoneNumber and returns the representation of the resource, including the properties above. Below you will find a list of required and optional parameters.

Table 4.6. Request Parameters

Parameter	Description
PhoneNumber(Required)	The phone number you want to provision.

Parameter	Description
FriendlyName	A formatted version of this phone number.
ApiVersion	Calls to this phone number will create a new RCML session with this API version.
VoiceCallerIdLookup	Look up the caller's caller-ID name. Either true or false.
VoiceUrl	The URL RestComm will request when this phone number receives a call.
VoiceMethod	The HTTP method RestComm will use when requesting the above Url. Either GET or POST.
VoiceFallbackUrl	The URL that RestComm will request if execution of VoiceUrl fails for any reason.
VoiceFallbackMethod	The HTTP method RestComm will use when requesting the VoiceFallbackUrl. Either GET or POST.
StatusCallback	The URL that RestComm will request to pass status parameters (such as the call state) to your application.
StatusCallbackMethod	The HTTP method RestComm will use to make requests to the StatusCallback URL. Either GET or POST.
SmsUrl	The URL that RestComm will request when receiving an incoming SMS message to this number. This may not be supported. Please consult with your DID provider.
SmsMethod	The HTTP method RestComm will use when making requests to the SmsUrl. Either GET or POST.
SmsFallbackUrl	The URL that RestComm will request if SmsUrl fail for any reason. Please see SmsUrl as this feature may not be supported.
SmsFallbackMethod	The HTTP method RestComm will use when making requests to SmsFallbackUrl. Either GET or POST.

4.3. Calls

A Call represents a connection between a phone or user agent and RestComm. This may be inbound or outbound. The Calls list resource represents the set of phone calls originated and terminated from an account.

Call Resource URI. /2012-04-24/Accounts/{AccountSid}/Calls/{CallSid}

Table 4.7. Resource Properties

Property	Description
Sid	A string that uniquely identifies this call.
ParentCallSid	A string that uniquely identifies the call that created this leg.
DateCreated	The date that this call was created.
DateUpdated	The date that this call was last updated.
AccountSid	The unique id of the Account that created this call.
To	The phone number or identifier that will be the recipient of this call.
From	The phone number or identifier that originated this call.
PhoneNumberSid	If the call was inbound, this is the Sid of the IncomingPhoneNumber that received the call.
Status	A string representing the status of the call. Possible values are queued, ringing, in-progress, completed, failed, busy and no-answer.
StartTime	The start time of the call. Empty if the call has not yet been started.
EndTime	The end time of the call. Empty if the call has not ended..
Duration	The length of the call in seconds.
Direction	A string describing the direction of the call. Possible values are inbound, outbound-api, and outbound-dial
CallerName	If this call was an incoming call, the caller's name. Empty otherwise.
Uri	The URI for this account, relative to http://localhost:port/restcomm.

4.3.1. Supported Operations

HTTP GET. Returns the representation of a Call resource, including the properties above.

4.3.2. Call List Resource

Call List Resource URI. /2012-04-24/Accounts/{AccountSid}/Calls

4.3.2.1. Supported Operations

HTTP GET. Returns the list representation of all the Call resources for this Account, including the properties above.

HTTP POST. Makes a new Call and returns the representation of the Call resource, including the properties above. Below you will find a list of required and optional parameters.

Table 4.8. Request Parameters

Parameter	Description
From(Required)	The phone number to use as the caller id.
To(Required)	The phone number to call.
Url(Required)	The fully qualified URL that should be executed when the call connects.
Method	The HTTP method RestComm should use when making its request to the above Url. Defaults to POST.
FallbackUrl	The URL that RestComm will request if execution of Url fails for any reason.
FallbackMethod	The HTTP method that RestComm should use to request the FallbackUrl. Must be either GET or POST. Defaults to POST.
StatusCallback	A URL that RestComm will request when the call ends to notify your app.
StatusCallbackMethod	The HTTP method RestComm should use when requesting the above StatusCallback. Defaults to POST.
Timeout	The number of seconds that RestComm should allow the phone to ring before assuming there is no answer. The default is 60 seconds.

4.4. SMS Messages

An SMS Message resource represents an inbound or outbound SMS message.

SMS Message Resource URI. /2012-04-24/Accounts/{AccountSid}/SMS/Messages/{SMSMessageSid}

Table 4.9. Resource Properties

Property	Description
Sid	A string that uniquely identifies this SMS Message.
DateCreated	The date that this SMS Message was created.
DateUpdated	The date that this SMS Message was last updated.
DateSent	The date that the SMS was sent or received by RestComm.
AccountSid	The unique id of the Account that sent or received this SMS message.
From	The phone number or short code that initiated the message.
To	The phone number or short code that received the message.
Body	The text body of the SMS message. Up to 160 characters long.
Status	The status of this SMS message. Possible values are queued, sending, sent, failed, and received.
Direction	The direction of this SMS message. Possible values are incoming, outbound-api, outbound-call.
ApiVersion	The API version RestComm used to handle the SMS message.
Uri	The URI for this account, relative to http://localhost:port/restcomm.

4.4.1. Supported Operations

HTTP GET. Returns the representation of an SMS Message resource, including the properties above.

4.4.2. SMS Message List Resource

SMS Message List Resource URI. /2012-04-24/Accounts/{AccountSid}/SMS/Messages

4.4.2.1. Supported Operations

HTTP GET. Returns the list representation of all the Call resources for this Account, including the properties above.

HTTP POST. Sends a new SMS Message and returns the representation of the SMS Message resource, including the properties above. Below you will find a list of required and optional parameters.

Table 4.10. Request Parameters

Parameter	Description
From(Required)	A phone number that is enabled for SMS.
To(Required)	The destination phone number in E.164 format.
Body(Required)	The text of the message you want to send, limited to 160 characters.

4.5. Recordings

Recordings are generated when you use the <Record> verb. Those recordings are hosted with RestComm for you to retrieve. The Recordings list resource represents the set of an account's recordings.

Recording Resource URI. /2012-04-24/Accounts/{AccountSid}/Recordings/{RecordingSid}

To download the audio file just append .wav after the RecordingSid.

Table 4.11. Resource Properties

Property	Description
Sid	A string that uniquely identifies this recording.
DateCreated	The date that this recording was created.
DateUpdated	The date that this recording was last updated.
AccountSid	The unique id of the Account that created this recording.
CallSid	The unique id of the call during which the recording was made.
Duration	The length of the recording, in seconds.
ApiVersion	The API version in use during the recording.
Uri	The URI for this account, relative to http://localhost:port/restcomm.

4.5.1. Supported Operations

HTTP GET. Returns the representation of a Recording resource, including the properties above.

HTTP DELETE. Removes the recording from the account.

4.5.2. Recording List Resource

Recording List Resource URI. /2012-04-24/Accounts/{AccountSid}/Recordings

4.5.2.1. Supported Operations

HTTP GET. Returns the list representation of all the Recording resources for this Account, including the properties above.

4.6. Transcriptions

A Transcription resource represents a transcription of a recording. A transcription is a text version of a recording produced using automatic speech recognition.

Transcription Resource URI. /2012-04-24/Accounts/{AccountSid}/Transcriptions/{TranscriptionSid}

Table 4.12. Resource Properties

Property	Description
Sid	A string that uniquely identifies this transcription.
DateCreated	The date that this transcription was created.
DateUpdated	The date that this transcription was last updated.
AccountSid	The unique id of the Account that created this transcription.
Status	A string representing the status of the transcription. Possible values are in-progress, completed, and failed.
RecordingSid	The unique id of the Recording this Transcription was made of.
Duration	The duration of the transcribed audio, in seconds.
TranscriptionText	The text content of the transcription.
Uri	The URI for this account, relative to http://localhost:port/restcomm.

4.6.1. Supported Operations

HTTP GET. Returns the representation of a Transcription resource, including the properties above.

HTTP DELETE. Removes the Transcription from the account.

4.6.2. Transcription List Resource

Transcription List Resource URI. /2012-04-24/Accounts/{AccountSid}/Transcriptions

4.6.2.1. Supported Operations

HTTP GET. Returns the list representation of all the Transcription resources for this Account, including the properties above.

4.7. Notifications

A Notification resource represents a single log entry made by RestComm while handling your calls or your use of the Restful APIs. It is very useful for debugging purposes. The Notifications list resource represents the set of notifications generated for an account.

Notification Resource URI. /2012-04-24/Accounts/{AccountSid}/Notifications/{NotificationSid}

Table 4.13. Resource Properties

Property	Description
Sid	A string that uniquely identifies this transcription.
DateCreated	The date that this transcription was created.
DateUpdated	The date that this transcription was last updated.
AccountSid	The unique id of the Account that created this transcription.
CallSid	CallSid is the unique id of the call during which the notification was generated. Empty if the notification was generated by the Restful APIs without regard to a specific phone call.
ApiVersion	The RestComm API version in use when this notification was generated. May be empty for events that don't have a specific API version.
Log	An integer log level corresponding to the type of notification: 0 is ERROR, 1 is WARNING.
ErrorCode	A unique error code for the error condition. You can lookup errors, in our Error Dictionary.
MoreInfo	A URL for more information about the error condition. The URL is a page in our Error Dictionary.
MessageText	The text for the notification.

Property	Description
MessageDate	The date the notification was actually generated
RequestUrl	The URL of the resource that caused the notification to be generated.
RequestMethod	The HTTP method in use for the request that caused the notification to be generated.
RequestVariables	The HTTP GET or POST variables that RestComm generated and sent to your server. Also, if the notification was generated by the Restful APIs, this field will include any HTTP POST or PUT variables you sent.
ResponseHeaders	The HTTP headers returned by your server.
ResponseBody	The HTTP body returned by your server.
Uri	The URI for this account, relative to http://localhost:port/restcomm.

4.7.1. Supported Operations

HTTP GET. Returns the representation of a Notification resource, including the properties above.

4.7.2. Notification List Resource

Notification List Resource URI. /2012-04-24/Accounts/{AccountSid}/Notifications

4.7.2.1. Supported Operations

HTTP GET. Returns the list representation of all the Notification resources for this Account, including the properties above.