AT&T Developer Program

Getting to Know the AT&T Messaging APIs

Table of Contents

Introduction 2
Prerequisites 4
Conventions Used 5
Setting Up Your Computer for the Messaging Lab Exercises 6
Join the Developer Program 6
Register Your First App 6
Lab #1: Obtaining an Access Token without User Consent 10
API Details 10
Exercise Variant: REST Client (hurl.it) 12
Exercise Variant: Command Line Tool (curl) 14
Summary 15
Lab #2: Sending Text Messages with SMS 16
API Details 16
Exercise Variant: REST Client (hurl.it) 18
Exercise Variant: Command Line Tool (curl) 21
Summary 23
Lab #3: Obtaining an Access Token with User Consent 24
API Details 24
Exercise Variant: REST Client (hurl.it) 26
Exercise Variant: Command Line Tool (curl) 30
Summary 33
Lab #4: In App Messaging from Mobile Number 34
API Details 34
Exercise Variant: REST Client (hurl.it) 35
Exercise Variant: Command Line Tool (curl) 37
Summary 38
Lab #5: Getting Text Messages with SMS 39
API Details 39
Exercise Variant: REST Client (hurl.it) 40
Exercise Variant: Command Line Tool (curl) 42
Summary 44

Lab #6: Sending an Image with MMS 45
API Details 45
Exercise Variant: REST Client (hurl.it) 47
Exercise Variant: Command Line Tool (curl) 50
Additional Exercise: Choose Your Own Text and Image 52
Summary 53
Appendix A: Installing curl for Windows 54
Appendix B: SDKs and Tools 56
Appendix C: End-to-end Messaging Sample Application 57
Running the Sample Messaging App Using Heroku 57
How the App Works 61

© 2012 AT&T Intellectual Property. All rights reserved. AT&T, AT&T logo and all other AT&T marks contained herein are trademarks of AT&T Intellectual Property and/or AT&T affiliated companies. All other trademarks are the property of their owners. Actual results and your experience may vary from those described in this case study. Information and offers subject to change.

Introduction

AT&T's Messaging API suite allows you to create apps that send and receive text and multi-media messages. The following APIs are available:

SMS

Provides the ability to send and receive SMS text messages from and to a "shortcode". A shortcode is an identifying number, similar to a phone number, that is used for an app.

MMS

Similar to SMS, but allows you to send media files along with a message to and from a shortcode.

In App Messaging from Mobile Number

Provides the ability to send text messages to mobile phone numbers and email addresses. The text messages appear to come from the user's phone number, and therefore require the user's consent. This API can also be used to read the user's text messages.

The intent of this booklet is to guide you through a series of simple lab exercises to familiarize you with using the AT&T APIs and to greatly simplify the integration of one or more of them into your own applications.

These lab exercises are targeted toward developers and will be shown in curl and, where possible, a web-based REST client (www.hurl.it). You can use any programming language or REST client that you prefer, however.

The following pages will provide additional information.

- AT&T Developer Program
- APIs Overview
- SMS Overview
- SMS API Documentation

- MMS Overview
- MMS API documentation
- In App Messaging from Mobile Number Overview
- In App Messaging from Mobile Number API Documentation
- Case Studies

If you have any questions, please contact us at developer.program@att.com.

Prerequisites

In order to complete the exercises in this workbook, you will need:

- A browser and internet connection. You can make the API calls by using the website http://hurl.it.
- Optionally: A command line tool called curl (officially known as cURL). See Appendix A on how to download and install curl.

Conventions Used

The following formatting conventions are used in this workbook:

Type	Style	Example
URL	Monospace	https://api.att.com/speech/v3/speechToText
Parameter	*Italic*	*client_id*
Header	**Bold**	**application/json**
Command line	Monospace	curl -X POST --data-binary...
JSON or XML	Monospace	{ "Recognition": { "Status": "OK",...
User interface elements	**Bold**	**Join Now**
File names	**Bold**	**homeBy6.wav**

Setting Up Your Computer for the Messaging Lab Exercises

Pre-requisites: browser
Length: 10 minutes

In this exercise:

Join the Developer Program
Register Your First App

In order to call the AT&T APIs, you will first need to join the AT&T Developer Program. Once you have done so, you can register applications, which will each have their own app key, secret, and sometimes a shortcode. The app key acts as the OAuth client_id, the secret acts as the OAuth client_secret, and the shortcode is the number used as an address for SMS/MMS messages. (Shortcodes are only assigned if the app is going to use SMS/MMS.)

For these exercises, you can register for a trial account, which will let you register up to three applications.

See the Prerequisites section on what you need before starting the exercises.

You will need to register an app in the AT&T Developer Program website to use in the labs.

Join the Developer Program

Follow these steps to join the developer program.

Note: If you already have either a full or trial developer account, simply log in.

1. Launch your browser and go to http://developer.att.com.
2. Click on the **Join Now** link in the upper right hand corner.
3. Close the pop up.
4. Enter your name, username, email, and password.

Congratulations! You're part of the AT&T developer program.

Register Your First App

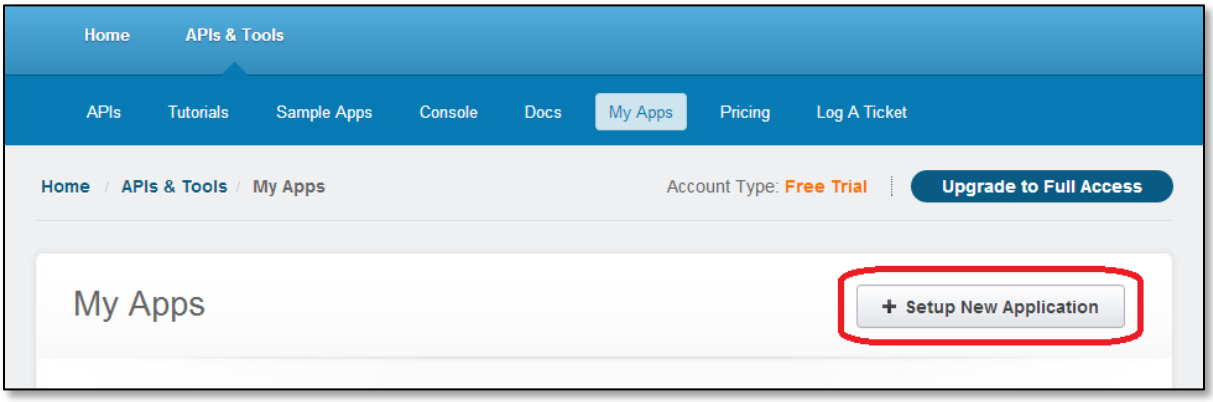
When you register an app, you will have the app key and secret code generated for you automatically. These codes can be viewed any time by logging into your Developer Program account.

Notes:

- If you already have a trial or full developer account, then sign in and click on the **My Apps** button.
- If you already have an app that you want to use for this exercise, make note of its client ID and secret, and edit the settings so that the Messaging checkboxes are checked, as shown in Step 3.

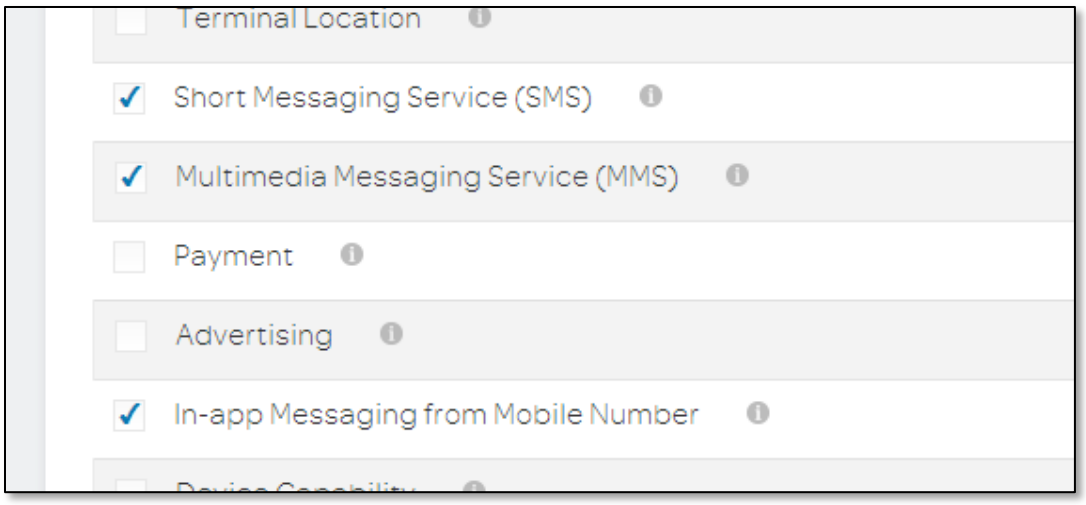
Follow these steps:

1. Click **Setup a New Application**.

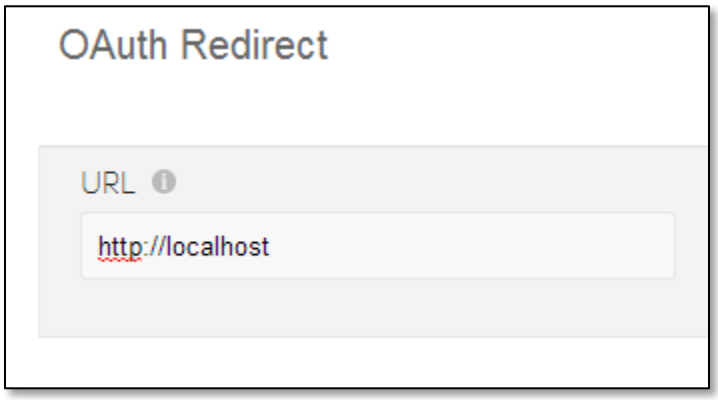


2. Type in an Application name and short description.

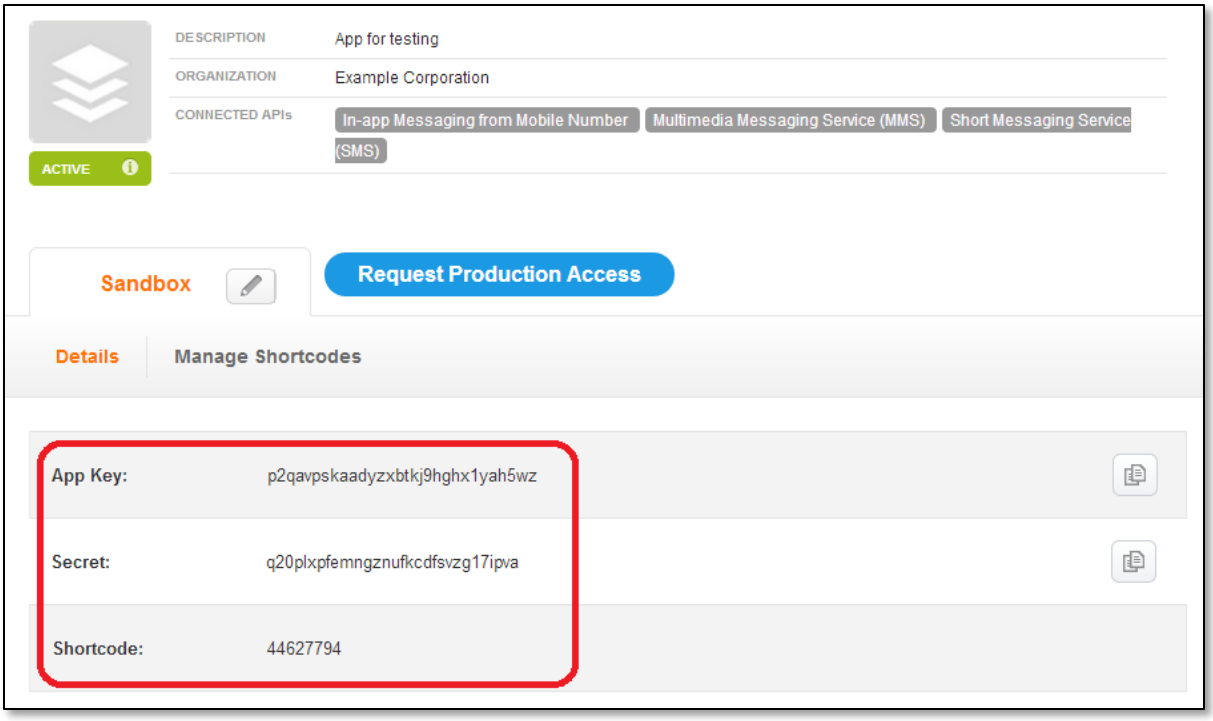
3. Check the following boxes: **In App Messaging from Mobile Number, Multimedia Messaging Service (MMS)**, and **Short Messaging Service (SMS).** (See screenshot that follows.) This will allow you to use these codes for creating apps that convert speech to text and text to speech.



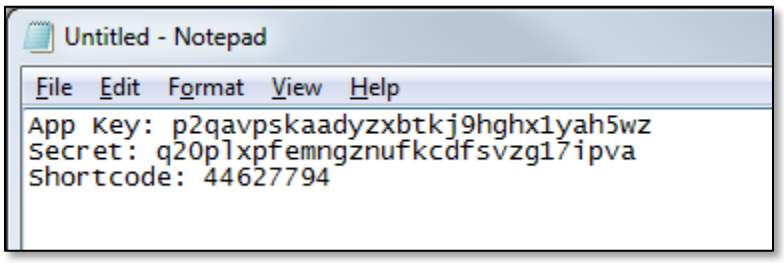
4. Specify **http://localhost** for the OAuth redirect value. This will be explained in more detail in the Obtaining an Access Token with User Consent lab. Actual applications require an externally accessible URL. But, for our lab this will do.



5. Click **Setup Application**.
6. Now you should see information about your application, including App Key, Secret, and Shortcode.



7. Open up a text editor and copy and paste the App Key, Secret, and Shortcode into it so you can easily access them later.



Lab #1: Obtaining an Access Token without User Consent

Pre-requisites: app key, secret, curl or browser
Length: 5 minutes

In this exercise:

- API Details
- Exercise Variant: REST Client (hurl.it)
- Exercise Variant: Command Line Tool (curl)
- Summary

There are two ways to use OAuth to obtain an access token: with or without user consent. API requests only require user consent if they involve user-specific data. For example, sending a message from the user's phone number requires user consent. However, sending messages to or from a shortcode does not require user-specific data, and therefore does not require user consent.

In this exercise, you will learn how to use OAuth to get an access token without consent. Once you have the access token, you can use it to make other API requests. The API requests that you can use are determined by the *scope* parameter, as explained below.

To make HTTP requests, you can use either a graphical client such as the hurl.it website or you can use the curl command line tool.

API Details

To obtain a token without consent, make an HTTP POST request to this URL:

```
https://api.att.com/oauth/token
```

The POST body contains the following parameters in the x-www-form-urlencoded format:

Parameter Name	Description
client_id	Your app's app key
client_secret	Your app's secret
grant_type	Set to **client_credentials** for OAuth requests without consent
scope	Which API requests will be used with this token. We want to use this with SMS, so we'll set it to **SMS**.

Note: The scope indicates which groups of API requests your app has permission to use. You can specify more than one scope by using a comma-separated list. The scope values must be a subset of the scope values that you selected using check boxes when editing the app settings in the developer program website. If you want to choose a scope that was not checked in the app settings, then you can edit the app settings and add it, but only if the app is in the Sandbox realm. If the app has been promoted to production, these settings cannot be changed.

The return value can be in either JSON or XML format. To set it, add a header called **Accept** with a value of **application/json** or **application/xml**. You need to add a header called **Content-Type** with a value of **application/x-www-form-urlencoded** so that it expects that the POST body is a series of key/value pairs, as would be found in a form POST.

More information can be found in the documentation at: OAuth 2.0 Authentication Management.

Exercise Variant: REST Client (hurl.it)

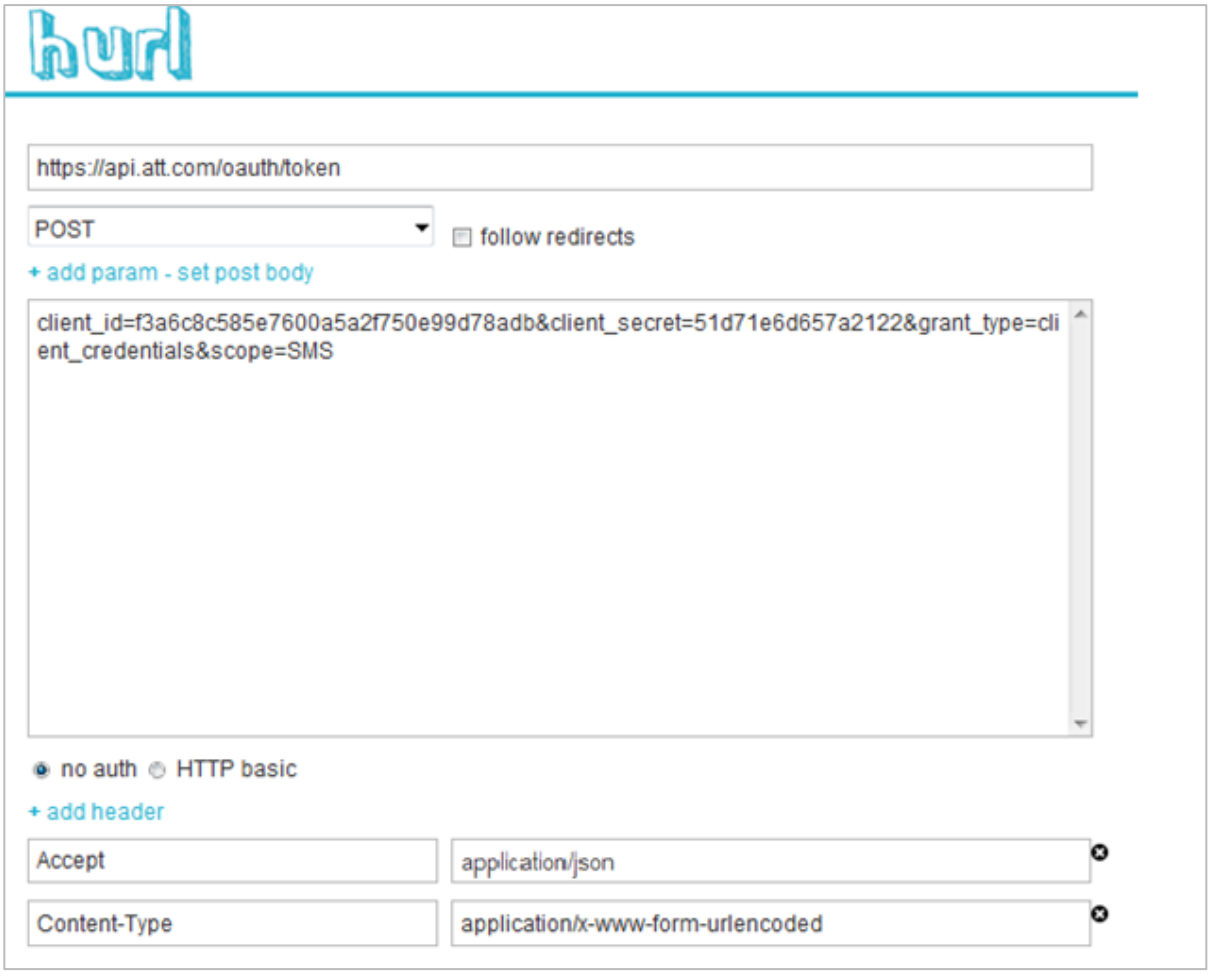
Follow these steps to obtain an access token without consent using the website **hurl.it**:

1. Open http://www.hurl.it in a browser.
2. In the first text box, type https://api.att.com/oauth/token
3. Change the dropdown to **POST**.
4. Click on **set post body**.
5. In the large text box for the POST body, copy and paste the following:

```
client_id={appkey}&client_secret={secret}&grant_type=client_credentials&scope=SMS
```

6. Replace {appkey} with your app key and {secret} with your secret.
7. Click the **+ add header** link to add new headers, if necessary. Add a header with name of **Accept** and value of **application/json**.
8. Add a header with name of **Content-Type** and value of **application/x-www-form-urlencoded**.
9. Click the **Send** button.

The screenshot below shows how to add the information into the hurl.it website.



You should see something like the following returned:

```
{
  "access_token":"4a0e9c2b14a0548a5ddb29eb77e06a8b",
  "token_type":"bearer",
  "expires_in":157679999,
  "refresh_token":"f40caf9ef812618275218602111e874a6f60"
}
```

Copy the value of the **access_token** key and paste it into a text editor for later use. In this case, the token will expire in 157679999 seconds (5 years).

Exercise Variant: Command Line Tool (curl)

Note: The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

Note: All curl commands in this workbook have the **--insecure** flag. This allows curl to perform "insecure" SSL connections so that you don't need the correct certification bundles installed. You may choose instead to install the correct certification bundles and to not use that flag.

Follow these steps to obtain an access token without consent using curl:

1. Paste the following into a text editor:

```
curl --request POST --insecure \
--header "Accept: application/json" \
--header "Content-Type: application/x-www-form-urlencoded" \
--data "client_id={appkey}&client_secret={secret}&\
grant_type=client_credentials&scope=SMS" \
https://api.att.com/oauth/token
```

2. Replace {appkey} with your app key and {secret} with your secret.
3. Open up a terminal (command prompt in Windows) that has curl.
4. Paste the edited curl command into the terminal and hit Enter.

You should see something like the following returned:

```
{
  "access_token":"4a0e9c2b14a0548a5ddb29eb77e06a8b",
  "token_type":"bearer",
  "expires_in":157679999,
  "refresh_token":"f40caf9ef812618275218602111e874a6f60"
}
```

Copy the value of the **access_token** key and paste it into a text editor for later use. In this case, the token will expire in 157679999 seconds (5 years).

Note: If you see the following returned:

```
{
  "error" : "invalid_scope",
  "error_description" : "The requested scope is invalid, unknown,
malformed, or exceeds the previously granted scope."
}
```

This means that you set the *scope* to a value that was not specified in the app settings. Go back to the app settings in the Developer Program website (see Register Your First App) and make sure that the SMS checkbox is checked.

Summary

You've learned how to obtain an access token that can be used for SMS apps that use a shortcode. In the next exercise, you will use this token to make the actual API request.

Lab #2: Sending Text Messages with SMS

Pre-requisites: access token without consent, curl or browser
Length: 15 minutes

In this exercise:

API Details
Exercise Variant: REST Client (hurl.it)
Exercise Variant: Command Line Tool (curl)
Summary

Let's say you are creating a game, and when your user gets a high score, you want to provide the option of automatically sending a message to a group of friends. There are two ways to do this:

1. Send a message that's from the app's shortcode. This is the SMS request, and it does not require user consent, so you can use the access token from the previous exercise.
2. Send a message that's from the user's phone number. This is called In App Messaging from Mobile Number, and it requires user consent, so you will need to obtain the access token in a different manner. This will be shown in the next exercise.

In this exercise, we will lead you through sending an SMS message using the SMS API, as well as checking the message's status.

Note: The current version of the SMS API only supports sending messages to AT&T numbers.

API Details

To send an SMS message, make an HTTP POST request to this URL:

```
https://api.att.com/sms/v3/messaging/outbox
```

The request contains the following headers:

Header Name	Description
Accept	The format of the data that should be returned.
Content-Type	The format of the data that is in the POST body.
Authorization	Has the value **Bearer {token}**, where {token} is your access token.

The POST body contains JSON or XML with these elements:

Header Name	Description
outboundSMSRequest	Contains the SMS information.
address	The telephone number or numbers to send the message. If one number, just use the number as the value. If multiple numbers, then use an array.
message	The message to send.

More information can be found in the documentation at: SMS.

Exercise Variant: REST Client (hurl.it)

Step 1. Send a Message

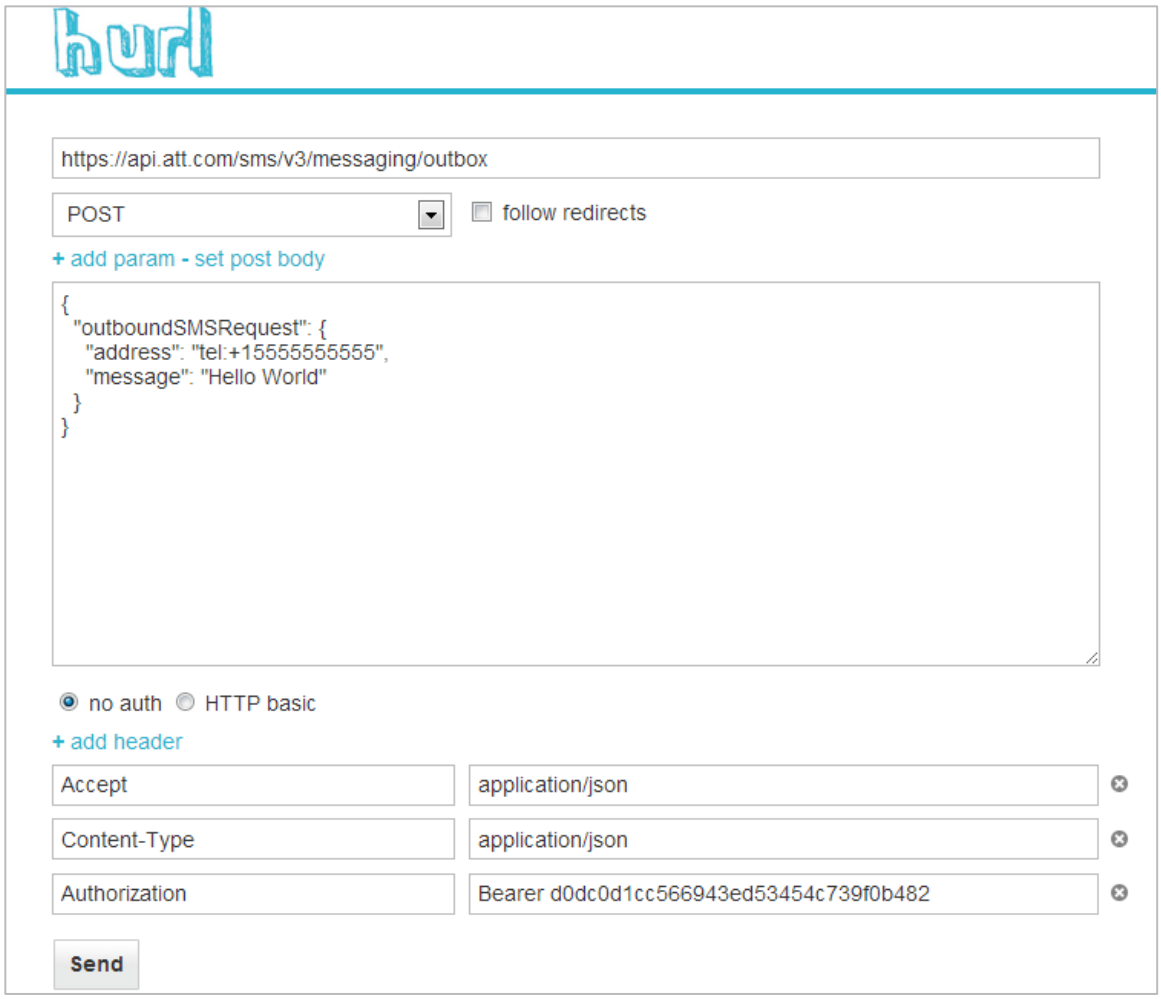
Use the following steps to send an SMS message using the website hurl.it:

1. Open http://www.hurl.it in a browser.
2. For the URL, enter **https://api.att.com/sms/v3/messaging/outbox**.
3. Make sure the dropdown is set to **POST**.
4. If the POST body is not visible, click on **set post body**. Add the following into the POST body box:

```
{
  "outboundSMSRequest": {
    "address": "tel:+15555555555",
    "message": "Hello World"
  }
}
```

5. Click the **+ add header** link to add new headers, if necessary. Add a header with name of **Accept** and value of **application/json**.
6. Add a header with name of **Content-Type** and value of **application/json**.
7. Add a header with name of **Authorization** and value of **Bearer {token}**, where **{token}** is the access token from Lab Exercise #1.

Your request should look like this:



8. Replace the 5's with an AT&T phone number.
9. Click **Send.**

Response

You should see something like the following returned:

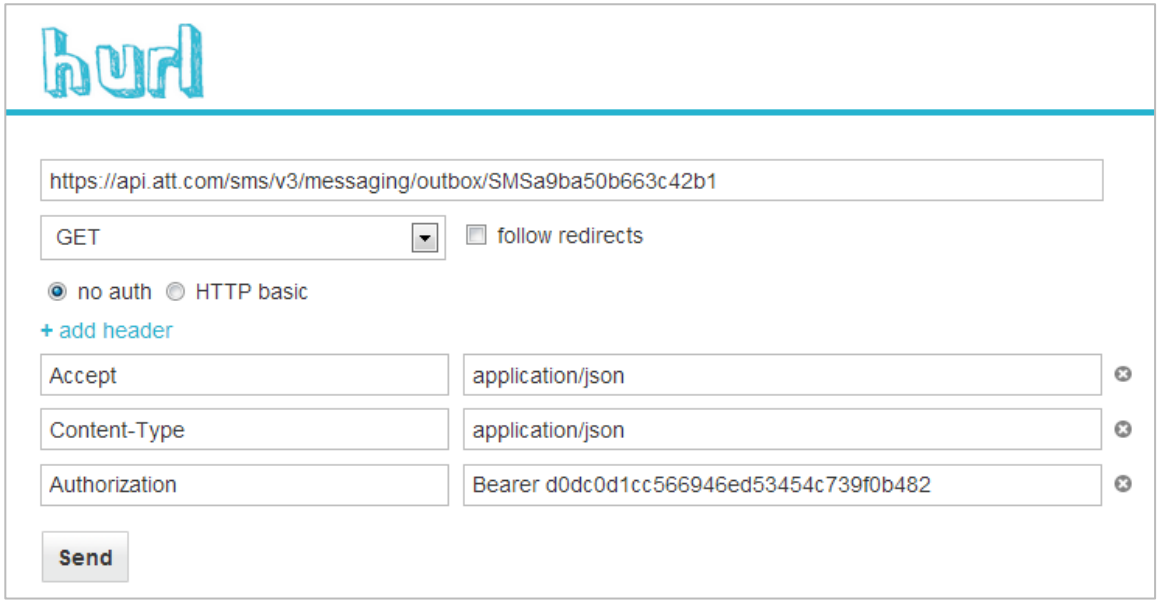
```
{
  "outboundSMSResponse": {
    "messageId": "SMSa98944bbc8535f4d",
    "resourceReference": {
        "resourceURL":
"https://api.att.com/sms/v3/messaging/outbox/SMSa98944bbc8535f4d"
    }
  }
}
```

The AT&T device should have received the "Hello World" text message.

Step 2. Check Delivery Status

Follow these steps. (Screenshot follows.)

1. In the browser, copy the value of the **resourceURL** to the clipboard. Paste it into the URL text box (the first text box in **hurl.it**).
2. Change the dropdown from **POST** to **GET**.
3. Click **Send**.



Response

The response contains information about the message:

```
{
  "DeliveryInfoList": {
    "DeliveryInfo": [
      {
        "Id": "msg0",
        "Address": "tel:+15555555",
        "DeliveryStatus": "DeliveredToTerminal"
      }
    ],
    "ResourceUrl":
"https://api.att.com/sms/v3/messaging/outbox/SMSa98944bbc8535f4d"
  }
}
```

The **DeliveryStatus** value of "DeliveredToTerminal" means that it has arrived on the device where the message was sent to. Other valid values are "DeliveredToNetwork", which means that the message has been delivered to the AT&T network, but has not yet arrived at the device, and "DeliveryImpossible", which means that message was not delivered successfully, such as when the message is not delivered before the message expires.

Exercise Variant: Command Line Tool (curl)

Step 1. Send a Message

Note: The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

Follow these steps to send an SMS message using curl:

1. Paste the following into a text editor:

```
curl --request POST --insecure \
--header "Authorization: Bearer {access token}" \
--header "Accept: application/json" \
--header "Content-Type: application/json" \
--data "{'outboundSMSRequest': { 'address': 'tel:+15555555555',
'message':'Hello World' }}" \
https://api.att.com/sms/v3/messaging/outbox
```

2. Replace {access token} with your with your access token.
3. Replace the 5's with an AT&T phone number.
4. Open up a terminal (command prompt in Windows) that has curl.
5. Paste the edited curl command into the terminal and hit Enter.

Response

You should see something like the following returned:

```
{
  "outboundSMSResponse": {
    "messageId": "SMSa98944bbc8535f4d",
    "resourceReference": {
        "resourceURL":
"https://api.att.com/sms/v3/messaging/outbox/SMSa98944bbc8535f4d"
    }
  }
}
```

The AT&T device should have received the "Hello World" text message.

Step 2. Check Delivery Status

Follow these steps.

1. Paste the following into a text editor:

```
curl --request GET --insecure \
--header "Accept: application/json" \
--header "Authorization: Bearer {access token}" \
https://api.att.com/sms/v3/messaging/outbox/{SMS code}
```

2. Replace {SMS code} with the code at the end of the URL from the previous response and replace {access token} with your access token.
3. Open up a terminal (command prompt in Windows) that has curl.
4. Paste the edited curl command into the terminal and hit Enter.

Response

The response contains information about the message:

```
{
  "DeliveryInfoList": {
    "DeliveryInfo": [
      {
        "Id": "msg0",
        "Address": "tel:+15555555",
        "DeliveryStatus": "DeliveredToTerminal"
      }
    ],
    "ResourceUrl":
"https://api.att.com/sms/v3/messaging/outbox/SMSa98944bbc8535f4d"
  }
}
```

The **DeliveryStatus** value of "DeliveredToTerminal" means that it has arrived on the device where the message was sent to. Other valid values are "DeliveredToNetwork", which means that the message has been delivered to the AT&T network, but has not yet arrived at the device, and "DeliveryImpossible", which means that message was not delivered successfully, such as when the message is not delivered before the message expires.

Summary

You've learned how to use the SMS API to send a message and check its delivery status. Refer to the SMS documentation on how to send to multiple telephone numbers, and how to receive SMS messages.

Lab #3: Obtaining an Access Token with User Consent

Pre-requisites: browser, app key, secret, curl (optional)
Length: 15 minutes

In this exercise:

API Details
Exercise Variant: REST Client (hurl.it)
Exercise Variant: Command Line Tool (curl)
Summary

API requests that make use of user data require user consent. An example would be sending messages from within an app where the messages appear to come from the user's phone. This exercise will lead you through obtaining an access token with user consent so that you can make these API requests.

Obtaining and using a token with user consent requires three steps:

1. Obtaining an authorization code by having a user explicitly grant consent.
2. Using the authorization code to obtain an access token.
3. Using the access token to make API requests.

The first two steps are covered in this lab. Lab #4 shows step 3.

API Details

To obtain an authorization code, have your app navigate to this URL (or display a browser that navigates to this URL):

```
https://api.att.com/oauth/authorize
```

Use the following query parameters:

Parameter Name	Description
client_id	Your app's app key
scope	Which API requests will be used with this code. In this example, we'll set it to **IMMN**, for In App Messaging from a Mobile Number.

Once the user gives consent through the web page, the browser will be redirected to a new URL that you specified when creating the app. There will be a **code** query parameter that contains the authorization code.

To obtain a token with consent, make an HTTP POST request to this URL:

```
https://api.att.com/oauth/token
```

The POST body contains the following parameters in the x-www-form-urlencoded format:

Parameter Name	Description
client_id	Your app's app key
client_secret	Your app's secret
grant_type	Set to **authorization_code** for OAuth requests without consent
code	The authorization code that was returned with the redirected URL.
scope	Which API requests will be used with this token. We want to use this with SMS, so we'll set it to **SMS**.

Note: The scope indicates which groups of API requests your app has permission to use. You can specify more than one scope by using a comma-separated list. The scope values must be a subset of the scope values that you selected using check boxes when editing the app settings in the developer program website. If you want to choose a scope that was not checked in the app settings, then you can edit the app settings and add it, but only if the app is in the sandbox. If the app is in production, you cannot change those settings.

The return value can be in either JSON or XML format. To set it, add a header called **Accept** with a value of **application/json** or **application/xml**.

More information can be found in the documentation at: OAuth 2.0 Authentication Management.

Exercise Variant: REST Client (hurl.it)

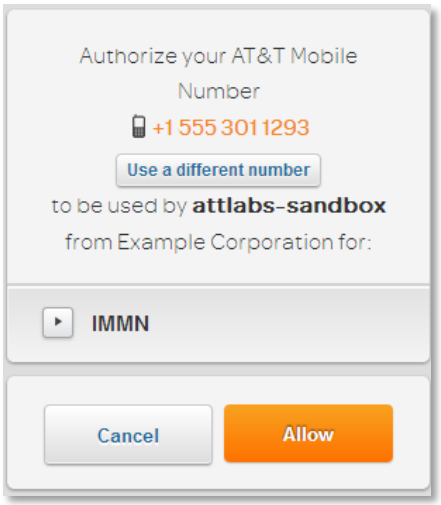
Step 1. Obtain an Access Code

When using your app, users need to explicitly provide consent. To do this, the app navigates to a specific URL within a browser on your device. Let's do that now. Note that we specify a scope of IMMN so that we can call the In App Messaging from Mobile Number API. (Multiple scopes may requested, separated by commas, but for this lab we are only using IMMN.)

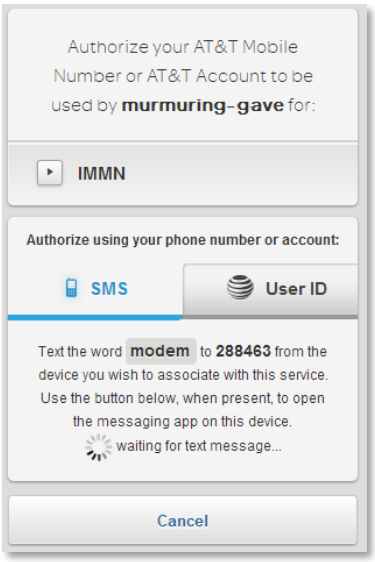
1. Copy the following line into a text editor:

```
https://api.att.com/oauth/authorize?client_id={appkey}&scope=IMMN
```

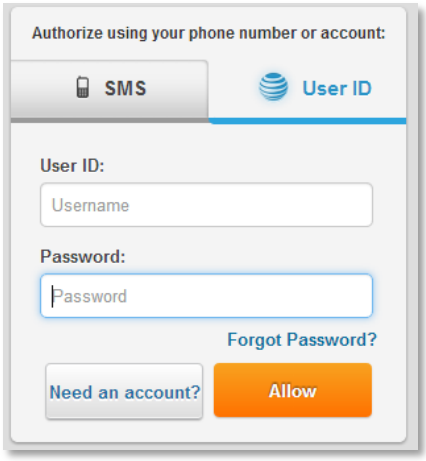
2. Replace {appkey} with the app key you got in Lab Exercise #1.
3. Copy and paste the line into a browser.
    a. If you are using an AT&T network, the following page will appear. (You are on an AT&T network if your internet connection is through an AT&T data plan.) Click **Allow** to give consent.



    b. If you are not using an AT&T network, then you will see the screen below instead. From your device with an AT&T phone number, text the message to the shortcode.



c. If you want to use your an AT&T ID to provide consent, click the **User ID** tab in order to sign in with an AT&T username and password.



4. Click the button that says **Close window**.
5. The browser will be redirected to **http://localhost/index.html?code={code}**. Take the code value and store it for later use.

Note. You might not have a valid **localhost/index.html** on your site, in which case the browser will show an error. This is okay. You just need the value of the **code** parameter. If this were a real app, it would intercept the redirection and take the value of the **code** parameter to be used later.

Now, to step 2, where you will use the authorization code to obtain the access token.

Step 2. Obtain an Access Token

Now that you have the code, you can get the access token through the **/oauth/token** request. In order to prove that you have obtained consent from the user, you need to include the code that you obtained in the previous step. Note that the **grant_type** is set to **authorization_code**. (When user consent is not required, the **grant_type** is often set to **client_credentials**.)

Follow these steps (screen shot follows):

1. Open http://www.hurl.it in a browser.
2. In the first text box, type https://api.att.com/oauth/token
3. Change the dropdown to **POST**.
4. Click on **set post body**.
5. In the large text box for the POST body, copy and paste the following:

```
client_id={appkey}&client_secret={secret}&grant_type=authorization_code&code={code}
```

6. Replace {appkey} with your app key, {secret} with your secret, and {code} with the code from the query parameter in the previous step.
7. Set it so that it has two headers. One which is **Accept** with a value of **application/json**. The other is Content-Type with a value of **application/x-www-form-urlencoded.** Add headers by clicking the **add header** link, and delete headers by clicking the **x** to the right of the header.
8. Click the **Send** button.

Response

You should see the following response:

```
{
  "access_token":"4a0e9c2b14a0548a5ddb29eb77e06a8b",
  "token_type":"bearer",
  "expires_in":157679999,
  "refresh_token":"f40caf9ef81261827521860211le874a6f60"
```

```
}
```

Copy the value of the **access_token** key and paste it into a text editor for later use. In this case, the token will expire in 157679999 seconds (5 years). Use this token when making calls to the In App Messaging from Mobile Number API.

Note: If you see the following returned:

```
{
  "error" : "invalid_scope",
  "error_description" : "The requested scope is invalid, unknown,
malformed, or exceeds the previously granted scope."
}
```

This means that you set the *scope* to a value that was not specified in the app settings. Go back to the app settings in the Developer Program (see Setting Up Your Computer for the Messaging Lab Exercises chapter) and make sure that the In App Messaging from Mobile Number checkbox is checked.

Exercise Variant: Command Line Tool (curl)

Note: The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

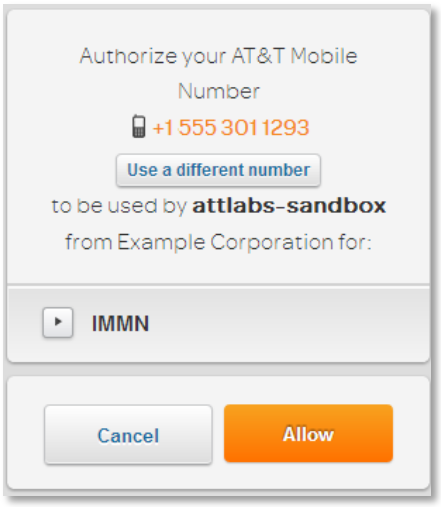
Step 1. Obtain an Access Code

When using your app, users need to explicitly provide consent. To do this, the app navigates to a specific URL within a browser on your device. Let's do that now. Note that we specify a scope of IMMN so that we can call the In App Messaging from Mobile Number API. (Multiple scopes may requested, separated by commas, but for this lab we are only using IMMN.)

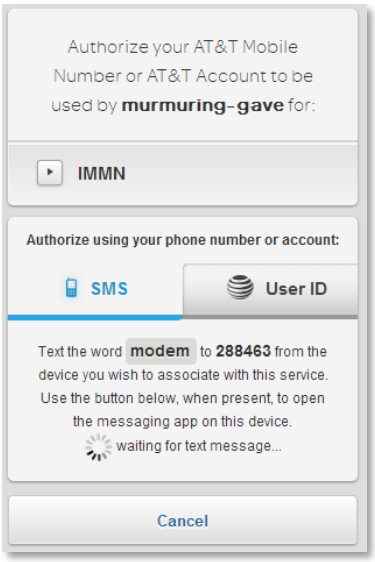
1. Copy the following line into a text editor:

```
https://api.att.com/oauth/authorize?client_id={appkey}&scope=IMMN
```

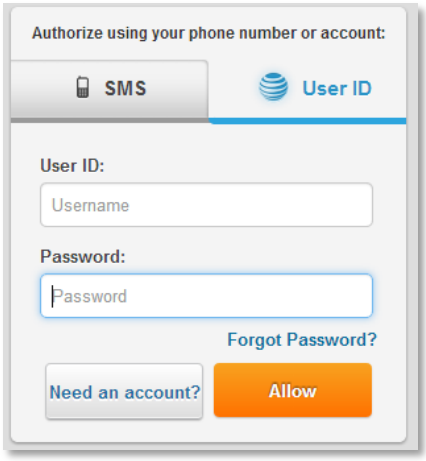
2. Replace {appkey} with the app key you got in Lab Exercise #1.
3. Copy and paste the line into a browser.
    a. If you are using an AT&T network, the following page will appear. (You are on an AT&T network if your internet connection is through an AT&T data plan.) Click **Allow** to give consent.



    b. If you are not using an AT&T network, then you will see the screen below instead. From your device with an AT&T phone number, text the message to the shortcode.



c. If you do not want to use an AT&T device, click the **User ID** tab in order to sign in with an AT&T username and password.



4. Click the button that says **Close window**.
5. The browser will be redirected to **http://localhost/index.html?code={code}**. Take the code value and store it for later use.

Note. You might not have a valid **localhost/index.html** on your site, in which case the browser will show an error. This is okay. You just need the value of the **code** parameter. If this were a real app, it would intercept the redirection and take the value of the **code** parameter to be used later.

Now, to step 2, where you will use the authorization code to obtain the access token.

Step 2. Obtain an Access Token

Now that you have the code, you can get the access token through the **/oauth/token** request. In order to prove that you have obtained consent from the user, you need to include the code that you obtained in the previous step. Note that the **grant_type** is set to **authorization_code**. (When user consent is not required, the **grant_type** is often set to **client_credentials**.)

Follow these steps:

1. Copy the following command to a text editor:

```
curl --request POST --insecure \
--header "Accept: application/json" \
--header "Content-Type: application/x-www-form-urlencoded" \
--data "client_id={appkey}&client_secret={secret}\
&grant_type=authorization_code&code={code}" \
https://api.att.com/oauth/token
```

2. Replace {appkey} with your appkey, {secret} with your secret, and {code} with the code from the query parameter in the previous step.
3. Paste into a console and hit Enter. You should see a result like this:

Response

You should see the following response:

```
{
  "access_token":"4a0e9c2b14a0548a5ddb29eb77e06a8b",
  "token_type":"bearer",
  "expires_in":157679999,
  "refresh_token":"f40caf9ef812618275218602111e874a6f60"
}
```

Copy the value of the **access_token** key and paste it into a text editor for later use. In this case, the token will expire in 157679999 seconds (5 years). Use this token when making calls to the In App Messaging from Mobile Number API.

Note: If you see the following returned:

```
{
  "error" : "invalid_scope",
  "error_description" : "The requested scope is invalid, unknown,
malformed, or exceeds the previously granted scope."
}
```

This means that you set the *scope* to a value that was not specified in the app settings. Go back to the app settings in the developer program and make sure that the In App Messaging from Mobile Number checkbox is checked.

Summary

You've learned how to obtain an access code for APIs that require user consent. This involves having your app navigate to a URL in a browser where the user can either enter an AT&T phone number or username and password. Once consent is obtained, an authorization code is generated. This code is then used to obtain an access token.

Lab #4: In App Messaging from Mobile Number

Pre-requisites: access token with consent, curl or browser
Length: 15 minutes

In this exercise:

API Details
Exercise Variant: REST Client (hurl.it)
Exercise Variant: Command Line Tool (curl)
Summary

Let's get back to our example of creating a game where when your user gets a high score, you want to provide the option of automatically sending a message to a group of friends. With the SMS API, the message sender shows up as the app's shortcode. What if, instead, we wanted the message to come from the user's phone number? In this case, we need the user's consent, so we use the token obtained in the previous exercise.

In this exercise, we'll send a message using the In App Messaging from Mobile Number API. In addition to sending messages to mobile numbers, it can also send them to email addresses and shortcodes.

API Details

To send a message using the In App Messaging from Mobile Number API, make an HTTP POST request to this URL:

```
https://api.att.com/rest/1/MyMessages
```

The request contains the following headers:

Header Name	Description
Accept	The format of the data that should be returned.
Content-Type	The format of the data that is in the POST body.
Authorization	Has the value **Bearer {token}**, where {token} is your access token.

The POST body contains JSON or XML with these elements:

Header Name	Description
Addresses	An array of mobile phone numbers and email addresses.
Text	The message to send.

More information can be found in the documentation at: In App Messaging from Mobile Number.

Exercise Variant: REST Client (hurl.it)

The In App Messaging from Mobile Number API call is a POST, where the body contains data on what the message is and where to send it. In this case we are passing JSON in, so we set the **Content-Type** values to **application/json**. Just to demonstrate that it can return a different format, we will set the **Accept** header to **application/xml** so that XML will be returned. We use the token that we obtained in Lab Exercise #3 in the **Authorization** header.

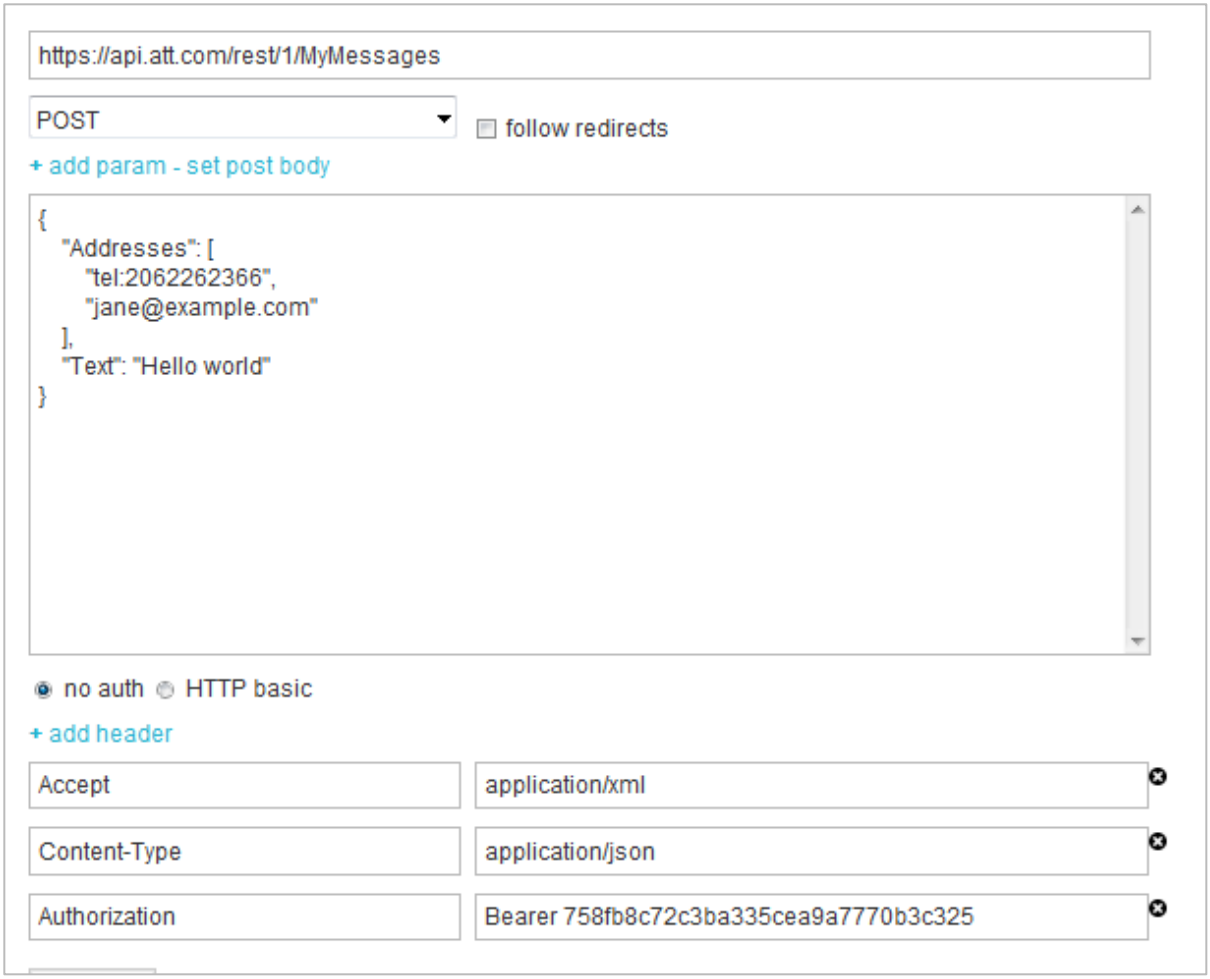
Use the following steps to send an SMS and email message using the website **hurl.it**:

1. Open http://www.hurl.it in a browser.
2. For the URL, enter `https://api.att.com/rest/1/MyMessages`
3. Make sure the dropdown is set to **POST**.
4. If the POST body is not visible, click on **set post body**. Add the following into the POST body box:

```
{
    "Addresses": [
        "tel:5555555555",
        "jane@example.com"
    ],
    "Text": "Hello world"
}
```

5. Click the **+ add header** link to add new headers, if necessary. Let's make the API return XML for a change. Add a header with name of **Accept** and value of **application/xml**.
6. Add a header with name of **Content-Type** and value of **application/json**.
7. Add a header with name of **Authorization** and value of **Bearer {token}**, where **{token}** is the access token with user consent that we obtained in the previous exercise.

Your request should look like this.



8. Replace the 5's with a mobile phone number. It does not have to be an AT&T number.
9. Replace **jane@example.com** with your own email address.
10. Click **Send**.

Response

If successful, you should get a response like this:

```
<Id>
  aUmNnCV7MCTyrkkvQi8
</Id>
```

Check your phone and email to see if you received the messages.

Exercise Variant: Command Line Tool (curl)

Note: The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

The In App Messaging from Mobile Number API call is a POST, where the body contains data on what the message is and where to send it. In this case we are passing JSON in, and we want the return data to be JSON as well, so we set both the **Accept** and **Content-Type** values to **application/json**. We use the token that we obtained in the last exercise in the **Authorization** header.

Follow these steps:

1. Paste the following into a text editor:

```
curl --request POST --insecure \
--header "Authorization: Bearer {access token}" \
--header "Accept: application/json" \
--header "Content-Type: application/json" \
--data "{'Addresses': ['tel:5555555555', 'jane@example.com'], 'Text':
'Hello world'}" \
https://api.att.com/rest/1/MyMessages
```

2. Replace {access token} with your with your access token from the previous exercise.
3. Replace the 5's with a mobile phone number. It does not have to be an AT&T number.
4. Replace **jane@example.com** with your own email address.
5. Open up a terminal (command prompt in Windows) that has curl.
6. Paste the edited curl command into the terminal and hit Enter.

Response

If successful, you should get a response like this:

```
{
 "Id": "CbeOo3YpujSUpR66FGT"
}
```

Check your phone and email to see if you received the messages.

Summary

You've learned how to send messages to both mobile numbers and email addresses using the In App Messaging from Mobile Number API and JSON. Refer to the In App Messaging from Mobile Number documentation to learn how to use XML and URL-encoded messages, as well as how to get message content and headers.

Lab #5: Getting Text Messages with SMS

Pre-requisites: access token without consent, curl or browser
Length: 15 minutes

In this exercise:

API Details
Exercise Variant: REST Client (hurl.it)
Exercise Variant: Command Line Tool (curl)
Summary

For this exercise, we'll send an SMS to a shortcode and then use the SMS API to read that message. Sending an SMS to a shortcode is often used for having people vote on something or send suggestions.

There are two different ways to receive messages sent to a shortcode:

- Polling – Call **GetSMS** at least once every 30 minutes to see if any messages have arrived.
- Listening – Call **ReceiveSMS** to set up a callback that is called when any messages have arrived.

In this exercise, we will use polling.

Unlike previous exercises, we will not provide you with the curl command. You will need to modify the previous command from the Sending Text Messages with SMS exercise.

Note: The current version of the SMS API only supports sending messages to shortcodes from AT&T numbers.

API Details

To get SMS messages, make an HTTP POST request to this URL:

```
https://api.att.com/sms/v3/messaging/inbox/{shortcode}
```

where {shortcode} is the shortcode that you sent the text message to. The request contains the following headers:

Header Name	Description
Accept	The format of the data that should be returned.
Authorization	Has the value **Bearer {token}**, where {token} is your access token.

Note: This method only works with SMS messages, and not MMS messages. Adding a Subject line or any multimedia converts the message to MMS, and it will not appear as an SMS message.

More information can be found in the documentation at: SMS.

Exercise Variant: REST Client (hurl.it)

Step 1. Create a shortcode

For this exercise you need an "offline" shortcode, which does not have URIs for receiving SMS messages. When you registered your App, it should already have a shortcode created as long as you chose the SMS API as one of your scopes. Check to see if your short code is offline by seeing if the URI fields are empty. If not, add a new shortcode and then delete the old one. Follow these steps.

1. Navigate to the developer platform. (http://developer.att.com/)
2. Click on the **APIs and Tools** menu and choose **My APIs.**
3. Click **Manage Sandbox** next to the name of the app that you used for the Sending Text Messages with SMS exercise.
4. You should see information about your app. Click on the **Manage Shortcodes** tab.
5. If you do not have a shortcode with empty URI fields, click the **Add Shortcode** button.
6. Leave the **Type** as **provided**, leave the **Rating** as **standard**, and leave all of the other fields blank. Click **Save.**
7. You should see your new shortcode listed.
8. Delete the old shortcode.

Step 2. Send a Text Message to that Shortcode

From an AT&T device, send a text message to the new shortcode.

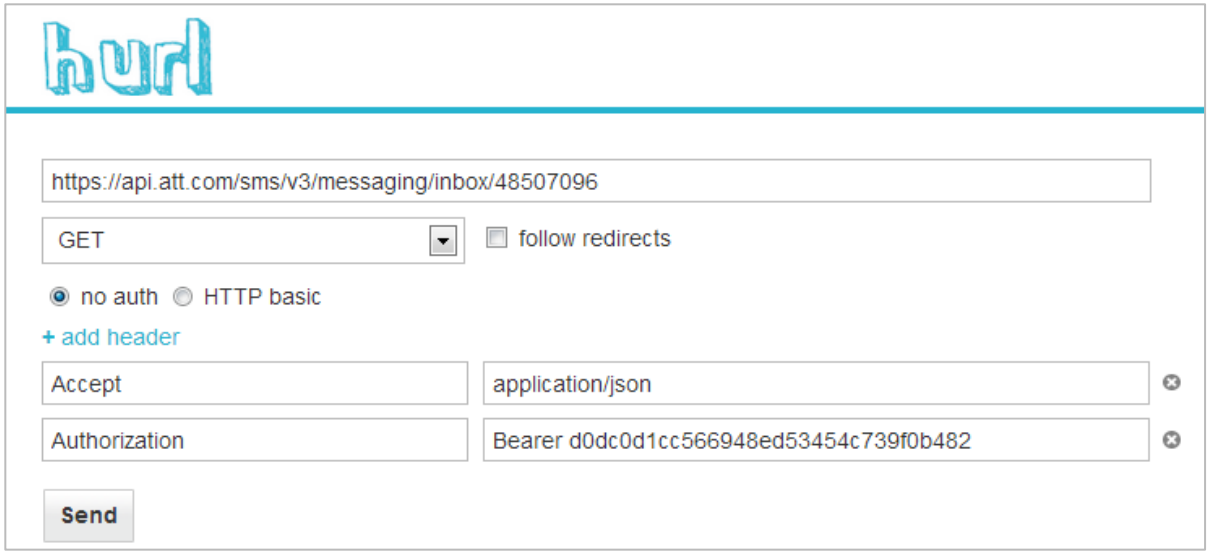
Step 3. Retrieve Text Messages

Follow these steps (screen shot follows):

1. Open http://www.hurl.it in a browser.
2. In the first text box, enter the following URL:

   ```
   https://api.att.com/sms/v3/messaging/inbox/{shortcode}
   ```

   where {shortcode} is the shortcode that you sent the text message to.
3. Set the dropdown to **GET**.
4. Click the **+ add header** link to add new headers, if necessary. Create a header with name of **Accept** and value of **application/json**.
5. Create a header with name of **Authorization** and value of **Bearer {token}**, where **{token}** is the access token from the previous exercise.
6. Click **Send**.



Response

You should see a response that shows the message that you sent, as well as information on the total number of messages sent, taking this form:

```
{
   "InboundSmsMessageList": {
      "InboundSmsMessage": [
         {
            "DestinationAddress": "48507096",
            "MessageId": "msg0",
            "Message": "Test message",
            "SenderAddress": "tel:4258983584"
         }
      ],
      "NumberOfMessagesInThisBatch": "1",
      "ResourceUrl":
"https://api.att.com/sms/v3/messaging/inbox/48507096",
      "TotalNumberOfPendingMessages": "0"
   }
}
```

Note: Once you make this call, it will clear out the inbox. If you make the call again, it will show zero messages in the inbox.

Note: Messages will only stay in the inbox for 30 minutes. After 30 minutes, they will be automatically removed.

Exercise Variant: Command Line Tool (curl)

Note: The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

Step 1. Create a shortcode

For this exercise you need an "offline" shortcode, which does not have URIs for receiving SMS messages. When you registered your App, it should already have a shortcode created as long as you chose the SMS API as one of your scopes. Check to see if your short code is offline by seeing if the URI fields are empty. If not, add a new shortcode and then delete the old one. Follow these steps.

1. Navigate to the developer platform. (http://developer.att.com/)
2. Click on the **APIs and Tools** menu and choose **My APIs.**
3. Click **Manage Sandbox** next to the name of the app that you used for the Sending Text Messages with SMS exercise.
4. You should see information about your app. Click on the **Manage Shortcodes** tab.
5. If you do not have a shortcode with empty URI fields, click the **Add Shortcode** button.
6. Leave the **Type** as **provided**, leave the **Rating** as **standard**, and leave all of the other fields blank. Click **Save.**
7. You should see your new shortcode listed.
8. Delete the old shortcode.

Step 2. Send a Text Message to that Shortcode

From an AT&T device, send a text message to the new shortcode.

Step 3. Retrieve Text Messages

Follow these steps:

1. Paste the following into a text editor:

```
curl --request GET --insecure \
--header "Authorization: Bearer {access token}" \
--header "Accept: application/json" \
https://api.att.com/sms/v3/messaging/inbox/{shortcode}
```

2. Replace {access token} with your with your access token from the previous SMS exercises.
3. Replace the {shortcode} with the shortcode from Step 1.
4. Open up a terminal (command prompt in Windows) that has curl.
5. Paste the edited curl command into the terminal and hit Enter.

Response

You should see a response that shows the message that you sent, as well as information on the total number of messages sent, taking this form:

```
{
   "InboundSmsMessageList": {
      "InboundSmsMessage": [
         {
            "DestinationAddress": "48507096",
            "MessageId": "msg0",
            "Message": "Test message",
            "SenderAddress": "tel:4258983584"
         }
      ],
      "NumberOfMessagesInThisBatch": "1",
      "ResourceUrl":
"https://api.att.com/sms/v3/messaging/inbox/48507096",
      "TotalNumberOfPendingMessages": "0"
   }
}
```


Note: Once you make this call, it will clear out the inbox. If you make the call again, it will show zero messages in the inbox.

Note: Messages will only stay in the inbox for 30 minutes. After 30 minutes, they will be automatically removed.

Summary

You've learned how to use the polling method for obtaining text messages that were sent to a shortcode. Once the request has been made, the inbox is cleared. Refer to the SMS documentation to learn how to use listening to receive SMS messages and their statuses.

Lab #6: Sending an Image with MMS

Pre-requisites: access token without consent, curl or browser, mms_multipart.txt fle.
Length: 10 minutes

For this exercise, we'll send an MMS message with an image to an AT&T device. The message will come from a shortcode. This request involves a multipart POST body that includes both JSON and an image.

Note: The current version of the MMS API only supports sending messages to AT&T numbers.

In this exercise:

API Details
Exercise Variant: REST Client (hurl.it)
Exercise Variant: Command Line Tool (curl)
Summary

API Details

To send an MMS message, make an HTTP POST request to this URL:

```
https://api.att.com/mms/v3/messaging/outbox
```

The request contains the following headers:

Header Name	Description
Accept	The format of the data that should be returned.
Content-Type	The format of the POST data. See note below.
Authorization	Has the value **Bearer {token}**, where {token} is your access token.

Note: The Content-Type will be **multipart/form-data**, with additional attributes of **type**, **start**, and **boundary**, where **type** is the format of the text data in the POST body, **start** is the Content-ID of the text data part of the POST body, and **boundary** is the text that is used as a divider between the multipart sections. See the exercise for an example of how this is constructed.

The POST body part with Content-ID as defined by the **start** attribute contains JSON or XML with these elements:

Header Name	Description
outboundSMSRequest	Contains the SMS information.
Address	The telephone number or numbers to send the message. If one number, just use the number as the value. If multiple numbers, then use an array.
Priority	The message priority.
Subject	The message subject text.

More information can be found in the documentation at: MMS.

Exercise Variant: REST Client (hurl.it)

Step 1. Obtain an Access Token for MMS.

Edit your app in the Developer Portal and make sure that you have the MMS checkbox selected as a scope.

By now you should be an expert in obtaining an access token without consent. Obtain a client_credentials token for a scope of MMS.

Important:

Step 2. Download and Inspect the File mms_multipart.txt.

Download the file **mms_multipart.txt**. Open this file in a text editor.

Note the following about the file:

1. It is a multipart file that contains both the JSON data and an image file.
2. Each part is divided with the string "MMSLabDivider", which acts as a boundary. The first two dividers start with "--". The last one starts with "--" and also ends with "--".
3. The content ID of the first part is <MessageInfo> and the content type is application/json, indicating that the data will be in JSON format.
4. The second part contains headers with the file name and type, as well as the image file itself. In this case, we have used Base64 encoding to make it easier to see the image in a text editor.
5. The file name is specified four times, and the protocol requires these all to be the same value. The four places are:
    a. The **name** attribute of Content-Type.
    b. The Content-ID value.
    c. The **filename** attribute of Content-Disposition
    d. The Content-Location value.

Step 3. Send an MMS Message

In this exercise, we will send an MMS message with a small image file that shows the AT&T logo.

Follow these steps:

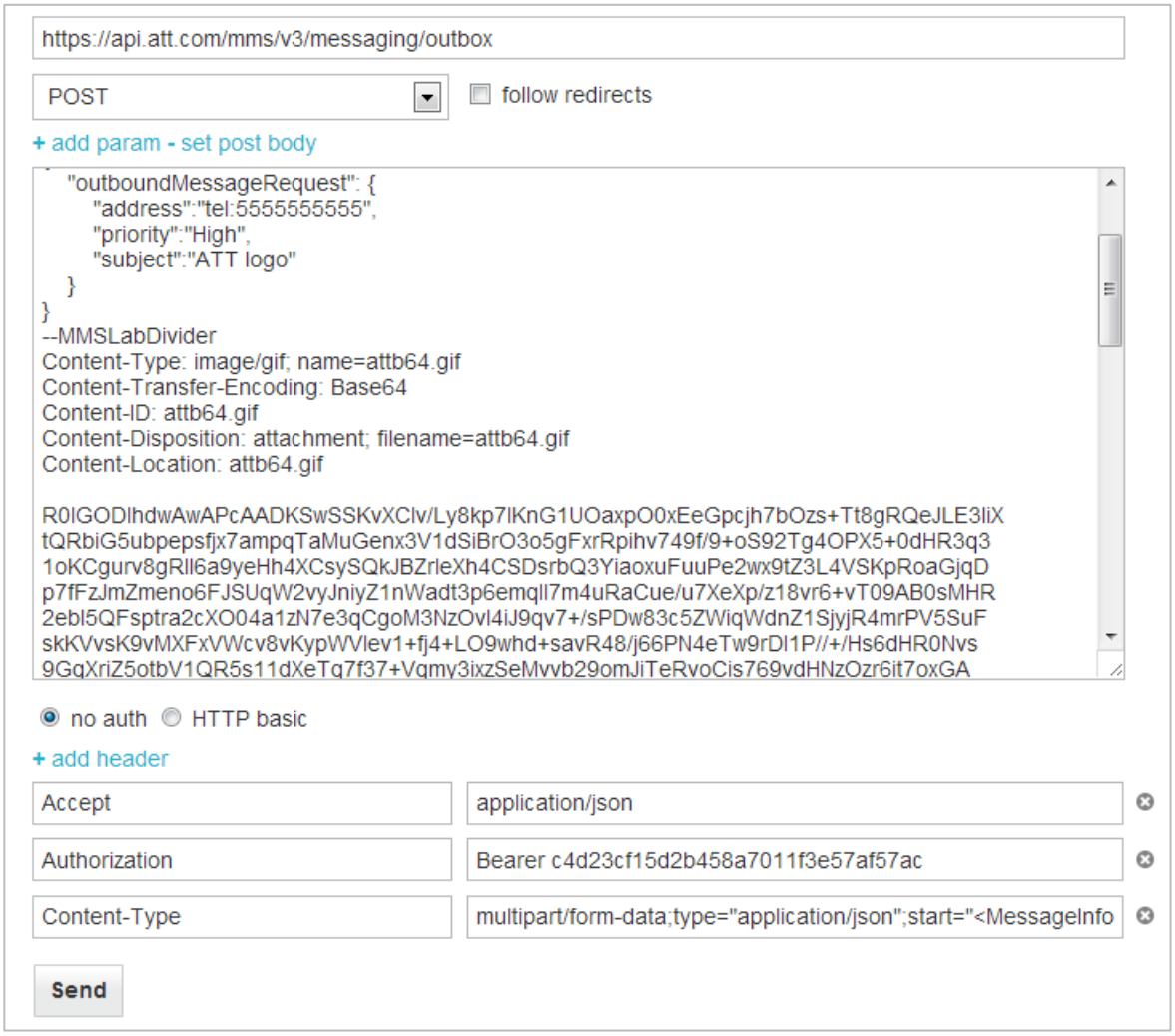
1. Open http://www.hurl.it in a browser.
2. For the URL, enter `https://api.att.com/mms/v3/messaging/outbox`.
3. Set the dropdown to **POST**.
4. Click the **+ add header** link to add new headers, if necessary. Create a header with name of **Accept** and value of **application/json**, so that the return data will be in JSON format.
5. Create a header with name of **Authorization** and value of **Bearer {token}**, where **{token}** is the access token from Step 1.
6. Create a header with the name of Content-Type and a value of

```
multipart/form-data;type="application/json";start="<MessageInfo>";
boundary="MMSLabDivider"
```

This indicates that:

a. The POST body is multi-part.
b. The POST body contains JSON data in its first part
c. The first part has a Content ID of <MessageInfo>
d. The divider between parts is called MMSLabDivider.

7. Select the entire text of mms_multipart.txt and Copy.
8. If the POST body is not visible, click **set post body**. If it is visible, delete everything in it.
9. Paste the text from mms_multipart.txt into the POST body box.



10. Change the 5s in the telephone number to be an AT&T number.
11. Click **Send**.

You should see JSON like the following returned:

```
{
  "outboundMessageResponse": {
    "messageId": "MMSa9880fb57c095369",
    "resourceReference": {
      "resourceURL":
"https://api.att.com/mms/v3/messaging/outbox/MMSa9880fb57c095369"
    }
```

```
    }
}
```

Check the AT&T phone for the MMS message with the AT&T logo image.

Exercise Variant: Command Line Tool (curl)

Step 1. Obtain an Access Token for MMS.

By now you should be an expert in obtaining an access token. Obtain a token for a scope of MMS.

Step 2. Download and Inspect the File mms_multipart.txt.

Download the file **mms_multipart.txt**. Open this file in a text editor.

Note the following about the file:

1. It is a multipart file that contains both the JSON data and an image file.
2. Each part is divided with the string "MMSLabDivider", which acts as a boundary. The first two dividers start with "--". The last one starts with "--" and also ends with "--".
3. The content ID of the first part is <MessageInfo> and the content type is application/json, indicating that the data will be in JSON format.
4. The second part contains headers with the file name and type, as well as the image file itself. In this case, we have used Base64 encoding to make it easier to see the image in a text editor.
5. The file name is specified four times. This is because different devices expect the file name in different places. To be sure that all devices will handle it properly, but the file name into all of them. The four places are:
    a. The **name** attribute of Content-Type.
    b. The Content-ID value.
    c. The **filename** attribute of Content-Disposition
    d. The Content-Location value.

Step 3. Send an MMS Message

Note: The curl exercises are designed for Unix-based terminals, such as Unix, MacOS, and Cygwin. They use the backslash (\) in order to continue onto more than one line. To run them on a Windows command prompt, substitute each backslash for a caret (^), or remove the backslashes and put it all on one line.

Follow these steps:

1. Open **mms_multipart.txt** in a text editor. Change the 5s in the telephone number to be an AT&T number. Save.
2. Paste the following into a text editor:

```
curl --request POST --insecure \
--header "Authorization: Bearer {access token}" \
--header "Accept: application/json" \
--header "Content-Type: multipart/form-data;type\
=\"application/json\"; start=\"<MessageInfo>\";\
boundary=\"MMSLabDivider\"" \
--header "MIME-version: 1.0" \
--data-binary @mms_multipart.txt \
https://api.att.com/mms/v3/messaging/outbox
```

3. Replace {access token} with your with your access token from Step 1.

4. Open up a terminal (command prompt in Windows) that has curl.
5. Paste the edited curl command into the terminal and hit Enter.

You should see JSON like the following returned:

```
{
  "outboundMessageResponse": {
    "messageId": "MMSa9880fb57c095369",
    "resourceReference": {
      "resourceURL":
"https://api.att.com/mms/v3/messaging/outbox/MMSa9880fb57c095369"
    }
  }
}
```

Check the AT&T phone for the MMS message with the AT&T logo image.

Additional Exercise: Choose Your Own Text and Image

For this last step, choose your own message and image to send. Follow these steps:

1. Open **mms_multipart.txt** in a text editor.
2. If you haven't already, change the 5s in the telephone number to be an AT&T number
3. Change the text message to be your own message.
4. Use a web-based Base64 encoder to encode your own image. Then copy and paste it over the existing Base64-encoded image. Be sure that the POST body still ends with:

```
--MMSLabDivider--
```

5. Change the file names to be the new file name in all four places.
6. Save the file.
7. If using hurl.it, copy and paste the new version of **mms_multipart.txt** into the POST body. If using curl, simply run the curl command again.

Summary

You've learned how to use the MMS API to send images from a shortcode to a device. Refer to the MMS documentation to learn how to receive MMS messages sent to a shortcode.

Appendix A: Installing curl for Windows

Pre-requisites: Windows 7 or later

curl (or cURL) is a powerful command-line tool for making HTTP calls. It is already installed on Mac computers.

Downloading curl

Use the following steps to install curl:

1. Open http://curl.haxx.se/dlwiz?type=bin in a browser.
2. Select your operating system in the dropdown box: either Windows /Win32 or Win 64. Click **Select**!
3. For Win 32, choose whether you will use curl in a Windows Command Prompt (**Generic**) or in a Cygwin terminal (**cygwin**). For Win 64, choose whether you will use curl in a Windows Command Prompt (**Generic**) or MinGW (**MinGW64)**. Click **Select!**
4. If required, choose your Windows operating system. **Finish.**
5. Click **Download** for the version which has SSL enabled.
6. Choose a version with support for SSL.
7. Open the downloaded zip file. Extract the files to an easy-to-find place, such as C:\Program Files.

Testing curl

1. Open up the Windows Command Prompt terminal. (From the Start menu, click Run, then type **cmd**.)
2. Set the path to include the directory where you put **curl.exe**. For example, if you put it in **C:\Program Files\curl**, then you would type the following command:

```
set path=%path%;"c:\Program Files\curl"
```

3. Type `curl`.

You should see the following message:

```
curl: try 'curl –help' or 'curl –message' fo r more information
```

This means that curl is installed and the path is correct.

4. Type:

```
curl --insecure https://api.att.com
```

You should see JSON returned:

```
{
"error": "invalid API request"
}
```

Troubleshooting

SSL certification error

If you see an SSL certification error, add a -k flag into your curl command.

https not supported error

If you get an error that says, “Protocol http not supported or disabled in libcurl”, then you need a different version of curl. Make sure you have downloaded one that says SSL enabled. If you have downloaded the Win64 version, try the Win32 version instead, even if you are on a 64 bit machine.

Appendix B: SDKs and Tools

Several Software Development Kits (SDKs) and other tools are available to make it easier to integrate the AT&T APIs into your apps. Rather than requiring you to make low-level HTTP calls, these SDKs and tools provide classes and methods that reduce the amount of code to call the underlying API requests. The following tools and SDKs are available from the Developer Platform website at SDKs and Tools:

- Microsoft® SDK (certified for Windows® 2008 Server)
- Android SDK
- iOS SDK
- AT&T API SDK for Adobe® PhoneGap®
- AT&T API Module for Appcelerator® Titanium®
- AT&T API SDKs for Windows® and Visual Studio® Extensions
- AT&T Toolkit for Salesforce® Platform
- AT&T API Adapters for IBM® Worklight®
- DeviceAnywhere Virtual Developer Lab Emulator
- Appery.io Plugin
- Viafo Plugin
- StackMob Enterprise Marketplace Custom Integration

Also, CodeKits (source code for classes providing wrappers to the APIs) in the following languages will be available soon:

- Java
- C#
- PHP
- Ruby

Finally, you can view sample code for several languages and platforms at the Sample Apps page.

Appendix C: End-to-end Messaging Sample Application

Pre-requisites: git, Heroku (instructions provided)
Length: 30 minutes

In this section:

Running the Sample MessagingApp Using Heroku
How the App Works

This section shows you how to set up an end-to-end application that uses AT&T Messaging APIs. The sample web app allows users to join a group and then send messages to all members of the group as if the message had come from their own phone number.

The user interacts with the web app in two ways:

- Users can register their AT&T phone numbers with the appliccation, granting the application permission to send messages on their behalf
- Users can send a message to a shortcode, and the message is forwarded to all registered users as if sent from the user who sent the message to the shortcode.

The sample application demonstrates the following APIs:

- Authorization with user consent.
- The ReceiveSMS method, which listens for SMS messages sent to a shortcode
- The In App Messaging from Mobile Number API, which sends SMS messages on behalf of a user

Running the Sample Messaging App Using Heroku

The sample app requires a web server, and in this exercise, we will use Heroku. Heroku is a Cloud Service Platform that provides free accounts for limited resources.

You will need git, which you can obtain at: http://git-scm.com/.

Follow these steps to install the sample code:

1. Create an application at the AT&T Developer Portal with both the SMS and IMMN scopes enabled, as in the Setting Up Your Computer for the Messaging Lab Exercises section.
2. Open a terminal (command prompt for Windows).
3. Use git to clone the git respository, using the command:

```
git https://github.com/attdevsupport/ruby_immn_sample.git
```

4. Follow the instructions at Heroku to create an account and log in.

5. Follow the steps at https://devcenter.heroku.com/articles/quickstart to install the Heroku Toolbelt on your local machine.
6. Create the Heroku instance with this command:

```
heroku create
```

7. Push the code from your machine to Heroku with this command:

```
git push heroku master
```

8. Add a Redis instance. Redis is an open source key-value store. For example, you can add it with this command:

```
heroku addons:add redistogo
```

Your Heroku account may need to be verified with a credit card for this to work, even though you won't be charged. Do this on the Heroku website.

9. Note that the code will not start. The app uses environment variables for data that is specific to your app, such as the app key and secret. Set environment variables now with this command:

```
heroku config:set API_KEY=xxxxxx API_SECRET=xxxxxx SHORT_CODE=xxxxxx
```

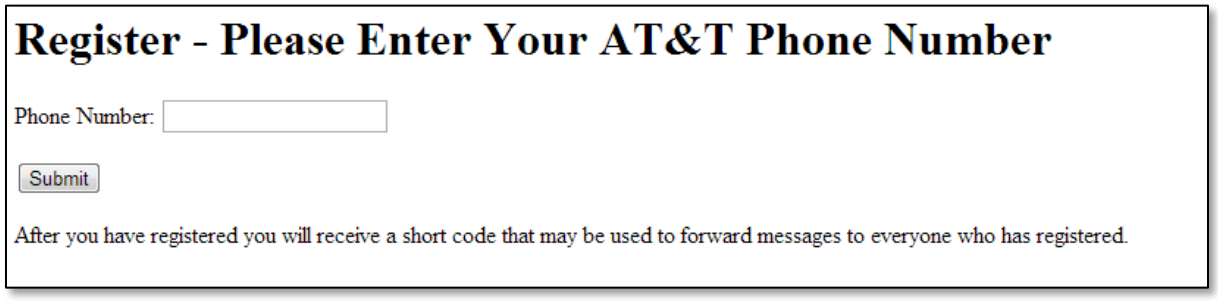
where you use the app key for API_KEY, secret for API_SECRET, and short code for SHORT_CODE. You should see a response such as:

```
Setting config vars and restarting murmuring-wave-1066... done, v4
API_KEY:     xxxxxxxxxxxxxxxxxxxx
API_SECRET: xxxxxxxxxxxxx
SHORT_CODE: xxxxxxx
```

10. Open up the website with this command. Your app will appear in your default browser.

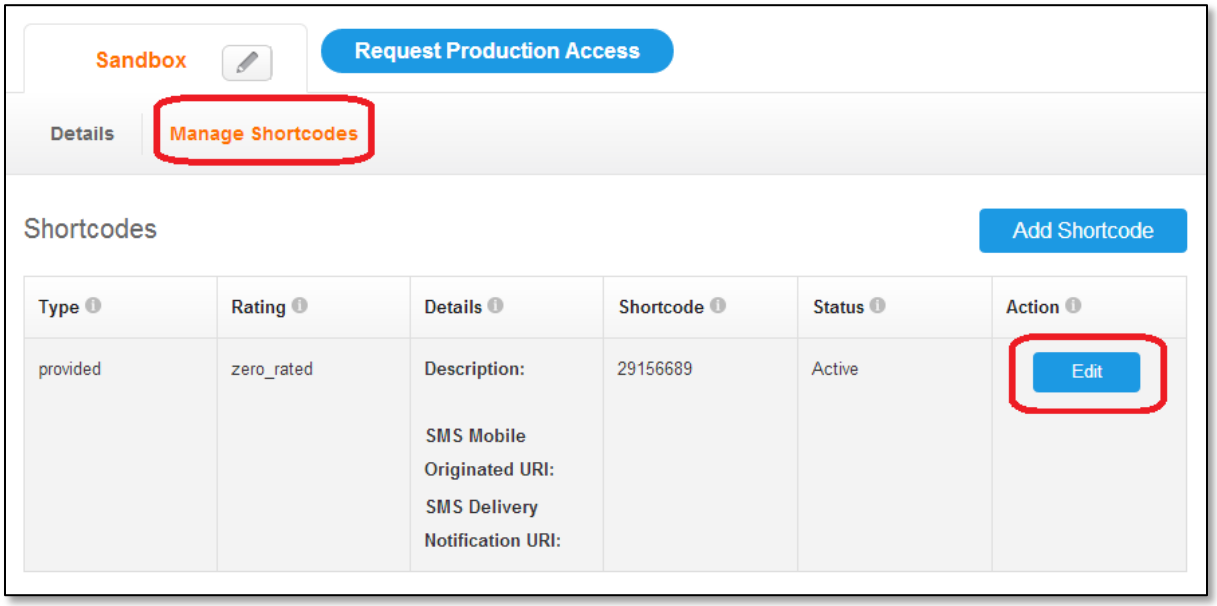
```
heroku open
```

11. You should see a place to enter your phone number. Don't enter anything just yet.

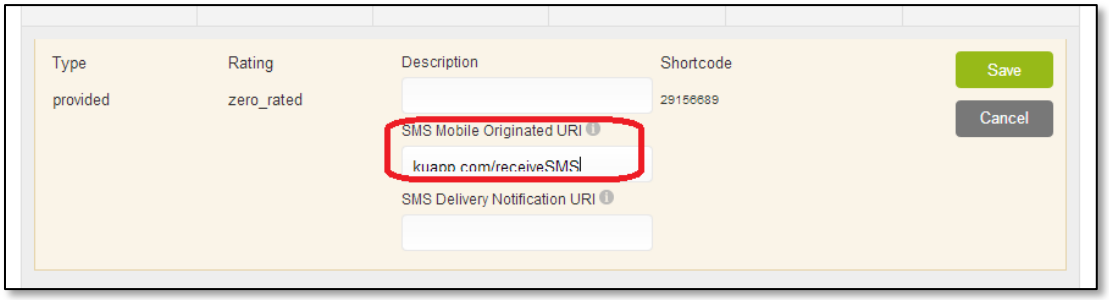


12. Copy the URL of the web page. In another browser tab or window, go to the Developer Portal, and edit your app by clicking on the pen button.
13. Under **OAuth Redirect**, in the **URL** text box, copy the URL of your app, adding `/authcode` at the end. This is the page that will be redirected to once users have granted their consent. Click on **Edit Application** to save.

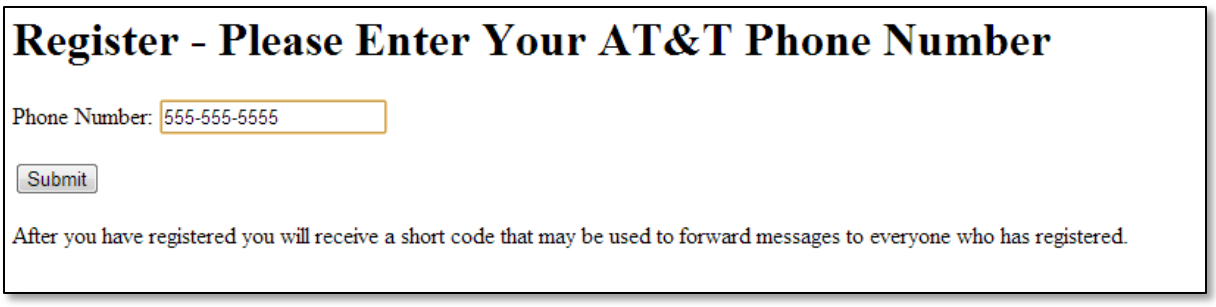
14. Next click on the **Manage Shortcodes** tab. Click **Edit** to edit your app's shortcode.



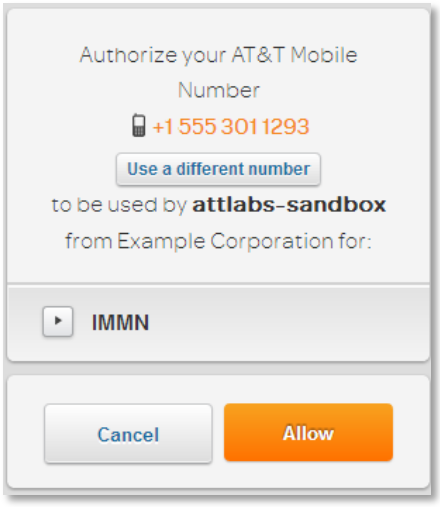
15. At the bottom of the page, copy in the URL of your app in the **SMS Mobile Originated URI**, adding `/receiveSMS` at the end. This is the URL that is used when the shortcode receives an SMS message. Click **Save** to save.



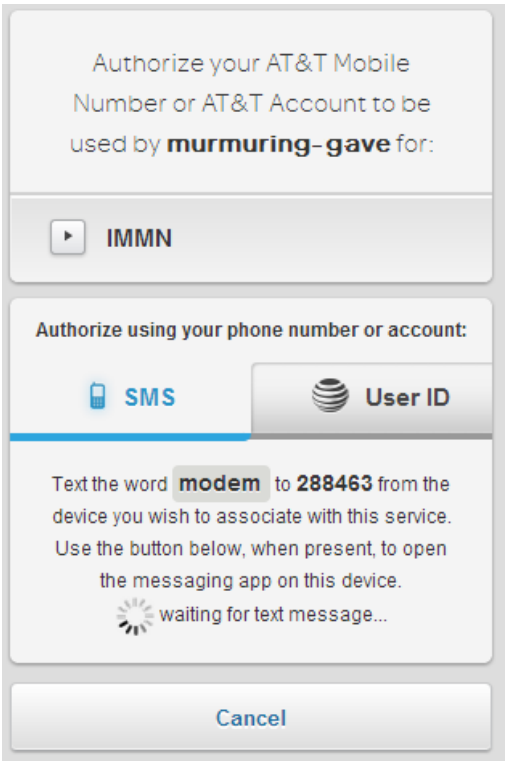
16. Now return to your app and enter in an AT&T phone number and click **Submit**.



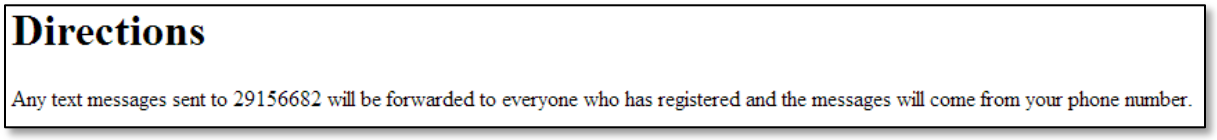
17. You will be asked for consent.
    a. If you are accessing the internet through an AT&T data connection, then you will see the screen below. Click the **Allow** button.



    b. If you are accessing the internet a different way, you have two ways to give consent. One is to text the word to the given shortcode from the device you are using. The other is to click on the **User ID** tab and enter your AT&T username and password.



18. Once consent is granted, after a few seconds, you will be redirected to the app's page that you specified on the Developer Portal. This page takes the authorization code and then displays the directions to use.



19. If you like, have other people go to the app's main page and add other AT&T phone numbers. (Note that you cannot add other phone numbers from the same browser because cookies will have been set indicating that consent has been given for the first phone number entered. To work around this, you would need to delete the cookies between adding each number.)
20. Now, send a text message from the registered phone to the shortcode listed on the Directions page (which is your app's shortcode). The message will be sent to all numbers that have registered with the app.

How the App Works

The Messaging app is created using Ruby and the Sinatra web application library. In the file **immn_sample.rb**, you can find comments that lead you through the process flow of the application.

Step 1. The app home page displays the form that asks for the phone number.
Step 2. When the phone number is posted by the form, the app redirects to the page to obtain consent for In App Messaging from a Mobile Number. Note that the phone number (POST parameter **msisdn**) is passed as a query parameter to the redirect URL in order to handle the various redirects that take place.
Step 3. After the consent process, the browser navigates to the redirect URL, and the code and phone number are retrieved from the query parameters. The phone number and access token

are stored persistently. (See **persistence.rb** for the persistence code.) Then the browser is redirected to the **directions** page.

Step 4. The directions page displays the shortcode to send messages to.

The `post '/receiveSMS'` section is called when an SMS is sent to the shortcode. (This URL is set in the app information in the Developer Portal.) It parses the JSON to read the message, and then uses the In App Messaging from Mobile Number API to send out the message to each phone number.

Note that the ATTWrapper class is included as part of a gem. It contains methods such as:

- **getAuthCodeUrl**: Navigates to the consent page in order to get the authorization code.
- **getAuthCodeToken:** Uses the authorization code to get an authorization token.
- **postInAppMessage:** Sends an SMS message from the user.

To see the source code for the ATTWrapper class, go to https://github.com/attdevsupport/ruby_wrapper/blob/master/lib/ruby_wrapper.rb.