AT&T Developer Program

AT&T API Platform SDKs for Windows
Wrapper Library

Developer Guide

Revision Date **December 21, 2012**

Legal Disclaimer

This document and the information contained herein (collectively, the "**Information**") is provided to you (both the individual receiving this document and any legal entity on behalf of which such individual is acting) ("**You**" and "**Your**") by AT&T, on behalf of itself and its affiliates ("**AT&T**") for informational purposes only. AT&T is providing the Information to You because AT&T believes the Information may be useful to You. The Information is provided to You solely on the basis that You will be responsible for making Your own assessments of the Information and are advised to verify all representations, statements and information before using or relying upon any of the Information. Although AT&T has exercised reasonable care in providing the Information to You, AT&T does not warrant the accuracy of the Information and is not responsible for any damages arising from Your use of or reliance upon the Information. You further understand and agree that AT&T in no way represents, and You in no way rely on a belief, that AT&T is providing the Information in accordance with any standard or service (routine, customary or otherwise) related to the consulting, services, hardware or software industries.

AT&T DOES NOT WARRANT THAT THE INFORMATION IS ERROR-FREE. AT&T IS PROVIDING THE INFORMATION TO YOU "AS IS" AND "WITH ALL FAULTS." AT&T DOES NOT WARRANT, BY VIRTUE OF THIS DOCUMENT, OR BY ANY COURSE OF PERFORMANCE, COURSE OF DEALING, USAGE OF TRADE OR ANY COLLATERAL DOCUMENT HEREUNDER OR OTHERWISE, AND HEREBY EXPRESSLY DISCLAIMS, ANY REPRESENTATION OR WARRANTY OF ANY KIND WITH RESPECT TO THE INFORMATION, INCLUDING, WITHOUT LIMITATION, ANY REPRESENTATION OR WARRANTY OF DESIGN, PERFORMANCE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, OR ANY REPRESENTATION OR WARRANTY THAT THE INFORMATION IS APPLICABLE TO OR INTEROPERABLE WITH ANY SYSTEM, DATA, HARDWARE OR SOFTWARE OF ANY KIND. AT&T DISCLAIMS AND IN NO EVENT SHALL BE LIABLE FOR ANY LOSSES OR DAMAGES OF ANY KIND, WHETHER DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, SPECIAL OR EXEMPLARY, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF GOODWILL, COVER, TORTIOUS CONDUCT OR OTHER PECUNIARY LOSS, ARISING OUT OF OR IN ANY WAY RELATED TO THE PROVISION, NON-PROVISION, USE OR NON-USE OF THE INFORMATION, EVEN IF AT&T HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES OR DAMAGES.

Table of Contents

1. Introduction

The AT&T API Platform SDKs for Windows ('AT&T SDKs') combine the power of the Microsoft® .NET platform and AT&T API Platform so that developers can quickly bring robust C# and VisualBasic applications to market.

AT&T SDKs include:

1. **Portable Wrapper library** to develop great Apps and distribute on following platforms:
   - Windows Store
   - Windows Phone 8
   - .NET Framework 4.5

   This document describes how to set up and use this Wrapper library.

2. Visual Studio Extensions for:
   - Windows RT
   - Windows Phone 8

   These extensions can be directly downloaded from Microsoft Visual Studio Gallery into Visual Studio IDE. Please check developer.att.com for details.

The Wrapper library significantly reduces the complexity of building applications that use the AT&T Platform REST services. This distribution of Wrapper library includes:

- Full API documentation guides to installation and setup
- Working code samples

2. API Access

This release of Wrapper library provides access to the following AT&T API Platform APIs:

- SMS API
- MMS API
- Speech API

The complete documentation for the AT&T API Platform can be found at http://developer.att.com.

3. Classes in the Wrapper Library

The Wrapper library consists of two types of classes that are used to address the AT&T REST APIs:

- Service classes – SmsService, MmsService , SpeechService, and AuthorizationService that provides wrapper functions around appropriate AT&T Platform services. An application creates an instance of the required service class by passing the API Key, Secret Key, and End Point URL inside the AttServiceSettings class. When the wrapper functions in the service are invoked by the application, an instance of the corresponding response class is returned. Service classes are located in ATT.WP8.SDK namespace.
- Response classes – These classes provide wrappers for de-serializing the responses returned by the AT&T Platform services. These classes are located in ATT.WP8.SDK.Entities namespace.

4. SDK Wrapper and Sample App Packaging

The SDK Wrapper package contains the Wrapper library, licensing terms, and documentation:

- LICENSE.txt - A file containing the licensing terms for using the SDK Wrapper classes and sample applications.
- bin - A directory containing the SDK Wrapper class library ATT.WP8.SDK.dll.
- docs - A directory containing SDK documentation.

The Sample App package contains sample application for the C# language that showcase the usage of the Wrapper library.

5. Installation

To install the Wrapper library, do the following:

1. Download the SDK package from AT&T developer portal.
2. Unzip the SDK package
3. Add reference to ATT.WP8.SDK.dll from your project in Visual Studio

6. Prerequisites

For Deployment:

- Onto .NET Framework 4.5 Computers
  - Microsoft® Windows 8 or
  - Microsoft® Windows XP/Vista/7 with .NET Framework 4.5 installed
- Into Windows Store. Download and use on:
  - Microsoft® Windows 8 Computers/Devices or
  - Microsoft® Windows RT Devices
- Onto Windows Phone 8 Devices
  - Microsoft® Windows Phone 8 (latest OS update installed)

For Development:

- Microsoft® Windows 8, Microsoft® Visual Studio 2012, Windows Phone 8 SDK

7. AT&T Services Configuration

Each AT&T service requires API Key and Secret Key values that were assigned to your application when you registered it with AT&T. Also the service's root end point URL must be configured. The Wrapper library contains a special class AttServiceSettings that includes appropriate properties:

- *string ClientId* – represents API Key
- *string ClientSecret* - represents Secret Key
- *Uri EndPoint* - represents service endpoint root

Each service class has constructor that requires instance of AttServiceSettings class as parameter. For example to create SmsService we need to create instance of AttServiceSettings class, fill properties with API Key, Secret Key and End point URL and then pass this instance to SmsService constructor:

```
var attSettings = new AttServiceSettings(
       "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //API key, replace xxx with
real value
       "xxxxxxxxxxxxxxxx", //Secret Key, replace xxx with real value
       new Uri(@"https:\\api.att.com") //End point
       );

var smsService = new SmsService(attSettings);
```

8. SMS Service

The Wrapper library provides access to the SMS Service methods which implemented in ATT.WP8.SDK .SmsService class with following methods:

- `async Task<SmsResponse> SendSms(IEnumerable<string> phoneNumbers, string message)`
  - phoneNumbers – Specifies a list of phone numbers to send the SMS message to.
  - message – Specifies the text of SMS message.
  - returns – An instance of SmsResponce that represent an SMS REST service response.
- `async Task<DeliveryInfoList> GetSmsStatus(string smsId)`
  - smsId – Specifies the SMS message id that is extracted from the instance of SmsResponce that is returned by the SendSms method.
  - returns – An instance of DeliveryInfoList that contains delivery status for each phone number where the SMS message was sent.
- `async Task<InboundSmsMessageList> GetInboundSmsMessages(string shortCode)`
  - shortCode – Short code of your AT&T Application that is used to get the SMS message.
  - returns – A list of SMS messages via the instance of InboundSmsMessageList.

To call any method the SmsService instance must be created via a constructor with an instance of AttServiceSettings:

```
var attSettings = new AttServiceSettings(
     "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //API key, replace xxx with
real value
     "xxxxxxxxxxxxxxxx", //Secret Key, replace xxx with real value
     new Uri(@"https:\\api.att.com") //End point
     );

var smsService = new SmsService(attSettings);
```

To call the SendSms method provide phone numbers and message text via appropriate parameters:

```
var phoneNumbers = new List<string> { "xxxxxx",
                                      "xxxxxx", "xxxxxx" };
var message = "Hello World";

SmsResponse response = await smsService.SendSms(phoneNumbers, message);
```

To call the GetSmsStatus method the message Id must be extracted from SmsResponse instance returned by previous call of SendSms:

```
string msgId = response.Id;

DeliveryInfoList deliveryInfo = await smsService.GetSmsStatus(msgId);
```

To call the GetInboundSmsMessages method provide the short code phone number to get their SMS messages:

```
var shortCode = "xxxxxx";

InboundSmsMessageList inboundSms = await
smsService.GetInboundSmsMessages(shortCode);
```

9. MMS Service

The Wrapper library provides access to the MMS Service methods that are implemented in the ATT.WP8.SDK.MmsService class with following methods:

- `async Task<MmsResponse> SendMms(IEnumerable<string> phoneNumbers, string message, IEnumerable<ContentInfo> attachments, MmsPriority priority = MmsPriority.Normal)`
  - phoneNumbers – Specifies a list of phone numbers where the MMS message is sent.
  - message – Specifies the text of MMS message.
  - attachments – Specifies a list of ContentInfo items.
  - priority – Specifies the MMS priority from the MmsPriority enumeration.
  - returns – An instance of MmsResponce that represent an MMS REST service response.
- `async Task<DeliveryInfoList> GetMmsStatus(string mmsId)`
  - mmsId – Specifies the MMS message id that is extracted from an instance of MmsResponce that is returned by the SendMms method.
  - returns – An instance of DeliveryInfoList that contains delivery status for each phone number where the SMS message was sent.

To call any MmsService method an instance must be created via a constructor with aninstance of AttServiceSettings:

```
var attSettings = new AttServiceSettings(
  "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //API key, replace xxx with real
value
  "xxxxxxxxxxxxxxxx", //Secret Key, replace xxx with real value
  new Uri(@"https:\\api.att.com") //End point
  );

var mmsService = new MmsService(attSettings);
```

To call the SendMms method provide phone numbers, message text, and priority via appropriate parameters. Attachments must be populated to instances of the ContentInfo class that contains property Content for as byte array and property

Name from where file extension is extracted to map on appropriate HTTP ContentType header value. The following code example sends MMS with a single attachment extracted from the “c:\temp\picture.jpg” file:

```
List<string> phoneNumbers = new List<string> { "xxxxxx",
                                    "xxxxxx", "xxxxxx" };
var message = "Hello World";
var fullFileName = @"c:\temp\picture.jpg";
byte[] fileContent = File.ReadAllBytes(fullFileName);
var contentInfo = new ContentInfo {Content=fileContent,
                                     Name=Path.GetFileName(fullFileName)};
MmsResponse response = await mmsService.SendMms(phoneNumbers, message,
        new List<ContentInfo>{contentInfo}, MmsPriority.Default);
```

To call the GetMmsStatus method the message Id must be extracted from MmsResponse instance returned by previous call of SendMms:

```
string msgId = response.Id;
DeliveryInfoList deliveryInfo = await mmsService.GetMmsStatus(msgId);
```

10.Speech Service

The Wrapper library provides access to the Speech Service methods which are implemented in the ATT.WP8.SDK.SpeechService class with the following method:

- `async Task<SpeechResponse> SpeechToText(ContentInfo content, XSpeechContext speechContext = XSpeechContext.Generic, SpeechXArgs xargs = null)`
    - content – Specifies an instance of ContentInfo that represents the sound file content.
    - speechContext – Specifies value from the XSpeechContext enumeration.
    - SpeechXArgs – Specifies custom attributes.

To call the SpeechToText method the SpeechService instance must be created via a constructor that uses an instance of AttServiceSettings:

```
var attSettings = new AttServiceSettings(
  "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //API key, replace xxx with real
value
  "xxxxxxxxxxxxxxx", //Secret Key, replace xxx with real value
  new Uri(@"https:\\api.att.com") //End point
  );

var speechService = new SpeechService(attSettings);
```

Then provide sound file content as a ContentInfo instance, speech context from the XSpeechContext enumeration, and custom attributes via an instance of SpeechXArgs.

```
var fullFileName = @"c:\temp\sound.wav";
var contentInfo = new ContentInfo { Content =
File.ReadAllBytes(fullFileName),
                        Name = Path.GetFileName(fullFileName) };
SpeechResponse response = await speechService.SpeechToText(contentInfo,
                        XSpeechContext.BusinessSearch);
```

11. Authorization Service

Each AT&T REST service requires authorization with API Key, Secret Key and scope. Scope represented by enumeration ATT.WP8.SDK. ScopeType and identifies appropriate service SMS, MMS, Speech etc. SmsService, MmsService and Speech service make authorization behind the scene with API Key and Secret Key and scope restricted for given service SMS, MMS and SPEECH respectively. But it's possible to make authorization once and then use authorization token in service classes preventing each service class to authorization with restricted scope.

The following code example perform authorization for SMS, MMS and speech and then share authorization token between service classes.

```
var attSettings = new AttServiceSettings(
    "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //API key, replace xxx with real
value
    "xxxxxxxxxxxxxxx", //Secret Key, replace xxx with real value
    new Uri(@"https:\\api.att.com") //End point
var authService = new AuthorizationService(TestConfig.AttSettings);
var scopeList = new List<ScopeType>
                            {
                                ScopeType.SMS,
                                ScopeType.MMS,
                                ScopeType.SPEECH
                            };
string scopeString = ScopeList.ScopesList(scopeList);
OAuthToken clientCredential = await authService.GetClientCredentials(null,
scopeString);

var smsServ = new SmsService(attSettings);
smsServ.ClientCredential = clientCredential;

var mmsServ = new MmsService(attSettings);
mmsServ.ClientCredential = clientCredential;

var speechServ = new SpeechService(attSettings);
speechServ.ClientCredential = clientCredential;
```