ARL-TN-0807 ● DEC 2016

US Army Research Laboratory

Implementation of Sensor Twitter Feed Web Service Server and Client

by Bhagyashree V Kulkarni and Michael H Lee

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TN-0807 ● DEC 2016

US Army Research Laboratory

Implementation of Sensor Twitter Feed Web Service Server and Client

by Bhagyashree V Kulkarni
University of Maryland

Michael H Lee
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
December 2016	Technical Note	

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Implementation of Sensor Twitter Feed Web Service Server and Client	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)	5d. PROJECT NUMBER
Bhagyashree V Kulkarni and Michael H Lee	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
US Army Research Laboratory ATTN: RDRL-CII-B 2800 Powder Mill Road Adelphi, MD 20783-1138	ARL-TN-0807

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Many social media platforms have been introduced in recent years. The Sensor Twitter Feed Web Service was developed using the RESTful web service architecture to research how a social media platform can be leveraged to disseminate Army-relevant information across complex networks. The core of the Sensor Twitter Feed Web Service leverages Twitter's Web Service application programming interface (API) to access Twitter. This report describes the system overview, Twitter API authentication, usage information, and limitations.

15. SUBJECT TERMS

Twitter, Web Service, RESTful Web Service, social media, complex networks

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Michael H Lee
Unclassified	Unclassified	Unclassified	UU	18	19b. TELEPHONE NUMBER (Include area code) 301-394-5608

Contents

List of Figures

1. Introduction

Many social media platforms have been introduced in recent years. As such, the Army is interested in researching how a social media platform can be leveraged to disseminate Army-relevant information across complex networks.

This project was initiated by the US Army Research Laboratory's (ARL) Battlefield Information Processing Branch within the Computational and Information Sciences Directorate, which serves as the principal Army organization for basic and applied research in information sciences, network sciences, battlefield environment, and advanced computing and computational sciences to provide the Warfighter with knowledge superiority and ensure US military superiority. Within this context, a Twitter-enabled web service, the Sensor Twitter Feed Web Service, was developed to test and conduct basic and applied research to advance the understanding and development of fundamental theories and methodologies for effective contextual interactions between information and Warfighters.

The goal of the Sensor Twitter Feed Web Service is to acquire and transmit data for further use in different systems at ARL. The Sensor Twitter Feed Web Service is able to read data from and post data to Twitter, and can be swapped with a similar short message social network site, if required. The system was developed using the RESTful web service architecture, which enables publishing or consuming sensor data (e.g., images, text), such as weather, traffic volumes, public Internet-available cameras, and specialized devices over social networking sites.

The Sensor Twitter Feed Web Service is an intermediary transport for the web service payload content (i.e., read from or write to a Twitter feed). Based on requests for information from various types of other similar sensors or human-stimulated requests from application user interfaces, the web service responds with content for use in the requesting context. Received/posted data may be archived in a database for storage, presented visually, processed analytically, or a combination of all 3 functions. The functionality is reusable across other applications at ARL and further enhancements will be made to include any other necessary features in the future.

2. Rest Architecture

REST stands for representational state transfer, which is an architectural style for networked hypermedia applications. It is primarily used to build web services that are lightweight, maintainable, and scalable. A service based on REST is called a RESTful service. REST is not dependent on any protocol, but almost every

RESTful service uses the Hypertext Transfer Protocol (HTTP) as its underlying protocol for data communication. This service revolves around a resource where every component is a resource and a resource is accessed by a common interface using HTTP standard methods.

In the REST architecture, a REST server simply provides access to resources and the REST client accesses and presents the resources. Each resource is identified by a Uniform Resource Identifier. REST supports the use of various representations to represent a resource like text, JavaScript Object Notation (JSON), and Extensible Markup Language (XML).

The following are some of the HTTP methods that are commonly used in a REST-based architecture:

1) GET: Provides read-only access to a resource.
2) PUT: Used to create a new resource.
3) DELETE: Used to remove a resource.
4) POST: Used to update an existing resource or create a new resource.

Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to interprocess communication on a single computer. This interoperability (e.g., between Java and Python or Windows and Linux applications) is due to the use of open standards.

3. Twitter API

The scope of this project is limited to exchanging data over Twitter. Twitter's REST application programming interface (API) has been incorporated in the project for this purpose. The Twitter API provides programmatic access to read and write Twitter data. The REST API identifies Twitter applications and users using OAuth, which is an open protocol that allows secure authorization in a simple and standard method from web, mobile, and desktop applications. OAuth is required to authorize a user to access Twitter's API. In this project, an application-only form of OAuth authentication was employed. Application-only authentication is a form of authentication where the application makes API requests on its own behalf, without a user context. For this purpose, it is necessary to have an application associated with a Twitter account. Applications can be created on the Twitter developer site at https://dev.twitter.com/apps (see the Appendix for instructions). A user must login into a Twitter account to do so. Access tokens are generated once the application is created. To use OAuth, an application must perform the following:

1) Obtain access tokens to act on behalf of a user account.

2) Authorize all HTTP requests it sends to Twitter's APIs.

A test account was created on Twitter for this project and an application named "Sensor Twitter Feeds" was created. The access tokens or access keys generated for this application are used to authorize requests to the Twitter API. The access tokens are stored in the *AuthenticationKeys.properties* file in the server project. These keys are used to initialize every request sent to the Twitter API. The Twitter API returns responses in JSON format. In this project, the JSON responses are converted to XML using a wrapper class named *Response.java* in the server project.

The *AuthenticationKeys.properties* is formatted as follows:

```
consumerKey = aBc123…
consumerSecretKey = dEF456…
accessTokenKey = GhI789…
accessTokenSecretKey = jKl012…
```

The consumer key and consumer secret key are used for authorizing the application. The access token key and access token secret key are tokens used by the end users in the application's context.

4. Sensor Twitter Feeds Web Service

Based on the REST architecture, the web service interacts with the Twitter platform using Twitter's API. There are 2 components in the project. First is the server side project that defines the web service based on the REST architecture and uses Twitter API to access Twitter. Second is the client project that consumes the web service. Both projects are developed as Maven projects. Maven supports dependency management and will retrieve them transitively. Hence, it provides the tools needed to manage the complexity inherent in dependency management. Figure 1 shows an overview diagram of the system.

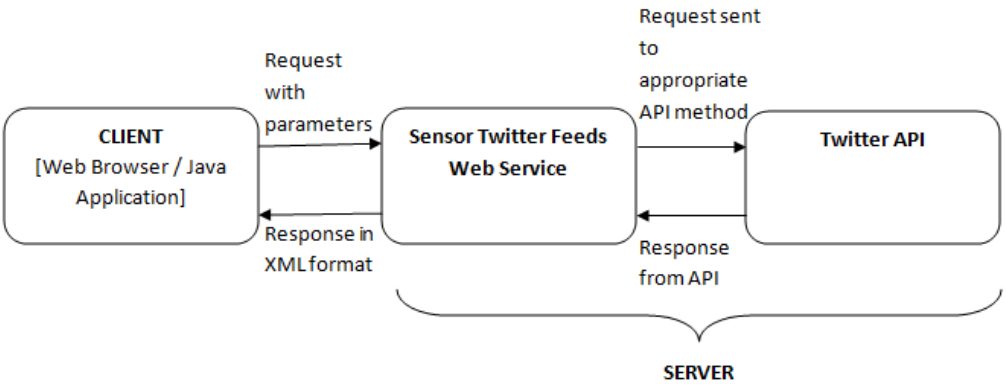

Fig. 1 Sensor Twitter Feed Web Service system overview diagram

The server component of the Sensor Twitter Feeds Web Service unwraps the request from the client side and sends it to Twitter API with the appropriate parameters. After receiving the response from Twitter API, the web service wraps the response in XML format and returns the response to the client.

5. Server Project

The server project defines an interface for operations to make requests to the Twitter API. At the service side, the handling of request and response objects is defined. The project should be deployed on Apache Tomcat Server 8.0. The web service is developed using Jersey framework, which is an open-source framework for developing RESTful web services in Java that provides support for JAX-RS APIs. Currently, the web service supports the following operations:

1) GET TWEETS
   This operation obtains most recent tweets posted by the authenticating user from the user's home timeline. There is one parameter for this operation:
   a) **count (optional)**: Specifies the number of records to retrieve. It must be less than or equal to 200; defaults to 20. The value of count is best thought of as a limit to the number of tweets to return, because suspended or deleted content is removed after the count has been applied.

2) SEARCH TWEETS
   This operation searches for tweets matching a specified query. There are 2 parameters for this operation:
   a) **query (required)**: A UTF-8, URL-encoded text search query of 500 characters maximum, including operators.
   b) **count (optional)**: The number of tweets to return per page, up to a maximum of 100; defaults to 15.

3) POST TWEETS
   This operation posts a tweet to the user's Twitter home timeline. There are 2 parameters for this operation:
   a) **text (required)**: The text of the status update, typically up to 140 characters. The URL link wrapping may affect the character counts. There are some special commands in this field to be aware of. For instance, preceding a message with "D" or "M" and following it with a screenname can create a direct message to that user if the relationship allows for it.

b) **mediaLink(optional)**: The URL of the image to be posted along with the text.
OR
mediaFile(optional): The name of the image file to be loaded from within the project classpath resources folder (i.e., src/main/resources).

The second parameter can either be the mediaLink or the mediaFile. Both parameters cannot be used at the same time. The supported formats for image are PNG, JPEG, BMP, WEBP, and GIF.

Special characters, such as @, $, and so on, and whitespaces can be included in the search text or text to post and do not need to be handled separately, as the web service will internally encode the string and convert the whitespaces into +, @ into %40, $ into %24, and so on.

If any of the operations are unsuccessful, the web service returns a response object with the error message sent by the Twitter API response.

6. Adding New Functionality

We advise implementing new functionality to the Sensor Twitter Feed Web Service for the server project. A method signature should be created in the interface *TwitterActionsI.java* and the method should be implemented in the *TweetUsingJava.java class*. The appropriate Twitter API method should be known while implementing this new method.

Lastly, in the *TwitterWebService.java* class, appropriate URL mapping needs to be added for the new functionality. For example, if an "update tweet" function is planned, *TwitterWebService.java* needs to define a new method with **@path("updateTweet")**. Once implemented, a user can access this method with the URL http://localhost:8080/SensorTwitterFeeds/twitterfeeds/updateTweet. This method should call the appropriate implementation defined in *TweetUsingJava. java* to obtain response from Twitter API. Finally, the method should wrap the JSON response in XML format.

7. Client Project

This part of the project is the client side that consumes the REST web service from Sensor Twitter Feeds. The client side was developed using a Jersey-client framework.

The Sensor Twitter Feeds Web Service can be consumed in 2 ways:

1) *Via browser requests through a URL:*

In the web browser, provide the URL for accessing the operation and pass the required parameters through the URL. The base URL is http://localhost:8080/SensorTwitterFeeds/twitterfeeds/. Add in the parameters defined in the previous section to invoke the operations.

A sample web service request and response would look like:

a) GET TWEETS

Request:
http://localhost:8080/SensorTwitterFeeds/twitterfeeds/getTweets?count=4

Response:

```
<response>
<reasonPhrase>OK</reasonPhrase>
<responseText>
<tweet>Hello Twitter 3 from server project</tweet>
<tweet>Hello Twitter 2 from server project</tweet>
<tweet>Hello Twitter 1 from server project</tweet>
<tweet>Hello Twitter from server project</tweet>
</responseText>
<reasonCode>200</reasonCode> </response>
```

b) SEARCH TWEETS

Request:
http://localhost:8080/SensorTwitterFeeds/twitterfeeds/searchTweets?query=Maryland&count=5

Response:

```
<response>
<reasonPhrase>OK</reasonPhrase>
<responseText>
<tweet>
Love free stuff #galaxys7edge #samsunggearvr #samsung @
Glen Burnie, Maryland https://t.co/rwtg2kxhNu
</tweet>
<tweet>
Candidates – Maryland’s 8th Congressional District – April
Technology Outreach Scorecard https://t.co/OyKg7F5oXo
#MD08 #MDPolitics
</tweet>
<tweet>
RT @FoxNews: Poll average shows @realDonaldTrump with a
commanding lead in Maryland. @TeamCavuto
https://t.co/Z1FRBRce82
</tweet>
<tweet>
```

```
#MTVStars Lady Gaga Poll average shows realDonaldTrump
with a commanding lead in Maryland. TeamCavuto
https://t.co/S5CynhcllL
</tweet>
<tweet>
RT @People4Bernie: Dear Maryland we're hearing buzz Bernie
will be in Baltimore on Saturday #FeelTheBern Volunteer at
a campaign office htt…
</tweet>
</responseText>
<reasonCode>200</reasonCode>
</response>
```

c) POST TWEETS

Request:

http://localhost:8080/SensorTwitterFeeds/twitterfeeds/postTweet?text=Test Tweet

Request with image URL:

http://localhost:8080/SensorTwitterFeeds/twitterfeeds/postTweet?text=Test Tweet&mediaLink=http://www.construplan.co.uk/wp-content/uploads/2015/07/Test-Post.jpg

Request with image filename:

http://localhost:8080/SensorTwitterFeeds/twitterfeeds/postTweet?text=Test Tweet&mediaFile=index.png

Response:

```
<response>
<reasonPhrase>OK</reasonPhrase>
<responseText/>
<reasonCode>200</reasonCode>
</response>
```

2) *By running the client project as a Java application:*

The operations that are to be invoked by the web service can be defined in the *WebServiceClient.java* class. The request-response of the web service is handled using the web resource object and the client response object. This client class is executed as a Java application to obtain the response from the web service.

8. Conclusion

The Sensor Twitter Feed Server and Client is a simple and scalable web service-based system for disseminating information by leveraging the Twitter platform. This report described the REST architecture, Twitter API, and the implementation and functionality of the Sensor Twitter Feed server and client. Instructions for expanding on the features of the system are also described.

9. Limitations and Future Extensions

The following are the limitations of and future extensions for the Sensor Twitter Feeds Web Service:

1) *Interface implementation changes:* The interface *TwitterActionsI* in the server project can be implemented to connect to other social networking platforms. The methods are generic enough to allow a common signature for implementation with other social networking sites.

2) *Multiple images uploading:* The web service allows uploading only one image at a time. The functionality should be extended to allow uploading multiple images.

3) *Videos uploading:* The functionality should be improved to enable uploading of videos. At present, it only allows uploading a single image.

4) *Handling multiple page results:* It is possible only to obtain results from the first page of the Twitter timeline. The functionality can be extended to allow results from multiple pages from a timeline.

5) *Authentication handling:* Currently, the requests to Twitter API are initialized with access tokens from the test account. This can be improved by finding a better way to send access tokens for different users making requests to the web service.

Appendix. Steps to Create an Application at a Twitter Developer's Website

In case a new application needs be associated with the project, it can be created at the Twitter developer's website. The Twitter account needs to have a valid phone number associated with it to be able to create an app.

To create a new application, perform the following steps:

1) Log into Twitter.
2) Go to https://dev.twitter.com/apps.
3) Click "Create New App".
4) A new page with the *Create an application* form requires basic information about the application:
    a) In the Name field, name the application in 32 characters or fewer. This name is presented to users when they are prompted to authorize the application to access their Twitter information.
    b) In the Description field, describe the application in 10 to 200 characters. Again, this is presented to users on the authorization screens.
    c) In the Website field, give a URL that points the user back to the application, where they can download it or find out more information. As with Name and Description, this field is presented on the user-facing authorization screens.
    d) The Callback URL field can specify the URL were Twitter should redirect after a successful authorization. It is best to leave this field blank and explicitly specify the callback URL at authorization.
    e) The developer's *Rules of the Road* section outlines rules one must agree to follow when building an application that uses Twitter's API. The rules include style guidelines on how to present tweets and cautions against simply re-creating the functionality of Twitter's own clients. It is recommended to read these rules closely to avoid violating them. To agree to the rules, check "Yes, I agree".
    f) A captcha challenge ensures that the application has not been set up through an automated process.
    g) Click "Create your Twitter application" to complete the form and go to the application settings page.
5) After the Twitter application is created, the access tokens can be found under Keys and Access Tokens tab.

6) Access levels for the application, such as read/write, can be set under the Permissions tab.

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA

2 (PDF) DIRECTOR US ARMY RESEARCH LAB RDRL CIO L IMAL HRA MAIL & RECORDS MGMT

1 (PDF) GOVT PRINTG OFC A MALHOTRA

5 (PDF) DIRECTOR US ARMY RESEARCH LAB RDRL CII B
STEPHEN M RUSSELL
ROBERT P WINKLER
ADRIENNE J RAGLIN
MICHAEL H LEE
RDRL CII
DEBORAH A WELSH

1 (PDF) UNIVERSITY OF MARYLAND B KULKARNI