PULSE - API

The purpose of this document is to provide example requests and responses for all relevant Pulse services requests

Contents

Introduction

This guide is intended to be a reference for developers interested in creating applications that interact with Pulse activities. Pulse's RESTful API provides access to resources such as comments, likes, mentions, and subscriptions. This guide provides the following information:

Getting Started: Provides general information about accessing Pulse data via the RESTful API

Authentication: Provides an overview of authenticating using Windows Authentication

REST API Conventions: Provides an overview of the key resources within the Pulse API

Resource Details: Provides details on accessing and posting using the RESTful API

Common API Tasks: Provides examples of common tasks

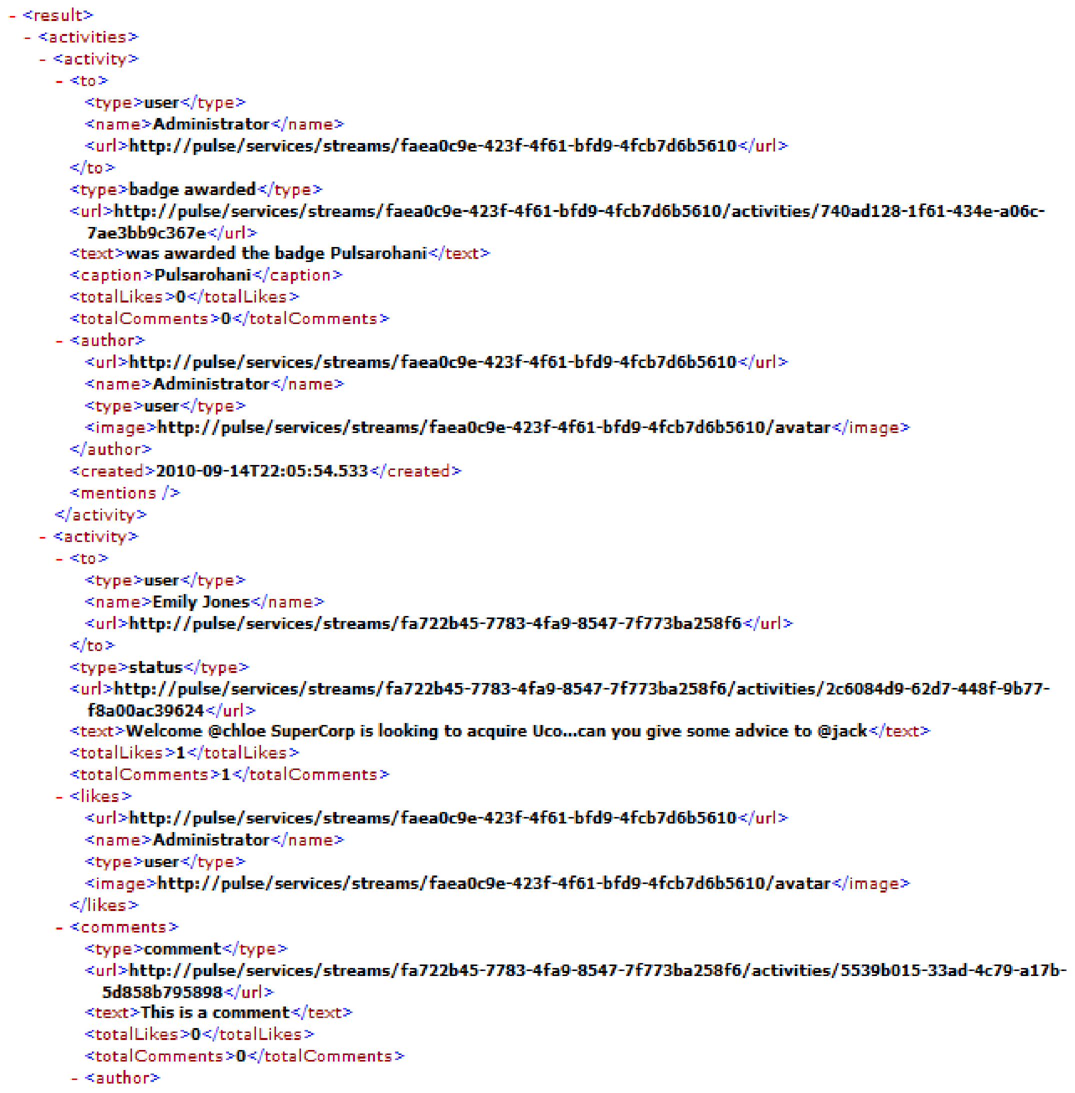
Getting Started

The easiest way to get an idea of the format used by the Pulse services is to navigate to your feed in a browser. Open your browser and go to your feed in the Pulse Web UI. The address bar should display a URL like:

http://pulse/services/streams/954c9f99-f10f-45d7-8639-bbced8a3aca0/activities

To view this feed in XML format, simply add /services/ to the front of the URL:

```
http://pulse/services/streams/954c9f99-f10f-45d7-8639-bbced8a3aca0/
activities
```

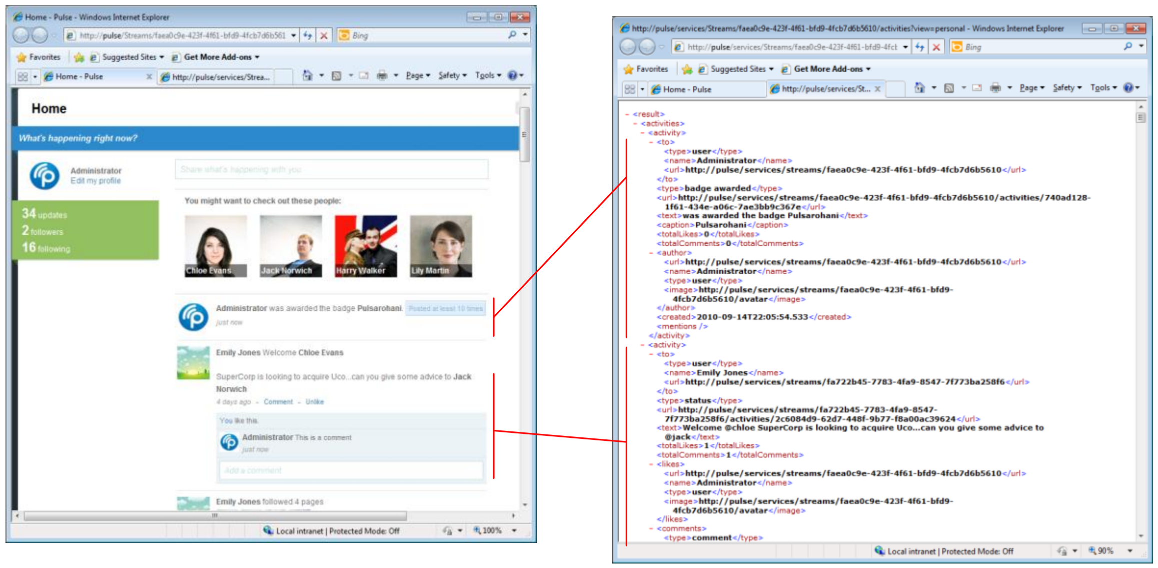


```
- <result>
  - <activities>
    - <activity>
      - <to>
          <type>user</type>
          <name>Administrator</name>
          <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-4fcb7d6b5610</url>
        </to>
        <type>badge awarded</type>
        <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-4fcb7d6b5610/activities/740ad128-1f61-434e-a06c-
          7ae3bb9c367e</url>
        <text>was awarded the badge Pulsarohani</text>
        <caption>Pulsarohani</caption>
        <totalLikes>0</totalLikes>
        <totalComments>0</totalComments>
      - <author>
          <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-4fcb7d6b5610</url>
          <name>Administrator</name>
          <type>user</type>
          <image>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-4fcb7d6b5610/avatar</image>
        </author>
        <created>2010-09-14T22:05:54.533</created>
        <mentions />
      </activity>
    - <activity>
      - <to>
          <type>user</type>
          <name>Emily Jones</name>
          <url>http://pulse/services/streams/fa722b45-7783-4fa9-8547-7f773ba258f6</url>
        </to>
        <type>status</type>
        <url>http://pulse/services/streams/fa722b45-7783-4fa9-8547-7f773ba258f6/activities/2c6084d9-62d7-448f-9b77-
          f8a00ac39624</url>
        <text>Welcome @chloe SuperCorp is looking to acquire Uco...can you give some advice to @jack</text>
        <totalLikes>1</totalLikes>
        <totalComments>1</totalComments>
      - <likes>
          <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-4fcb7d6b5610</url>
          <name>Administrator</name>
          <type>user</type>
          <image>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-4fcb7d6b5610/avatar</image>
        </likes>
      - <comments>
          <type>comment</type>
          <url>http://pulse/services/streams/fa722b45-7783-4fa9-8547-7f773ba258f6/activities/5539b015-33ad-4c79-a17b-
            5d858b795898</url>
          <text>This is a comment</text>
          <totalLikes>0</totalLikes>
          <totalComments>0</totalComments>
        - <author>
```

Figure 1: Common Response

Anatomy of a <result>

The result returned from the Stream service is a collection of **activity** elements that represents a feed. Mapping UI elements to the results is very straight forward. The following represents a typical response



Home

What's happening right now?

Administrator
Edit my profile

34 updates
2 followers
16 following

Share what's happening with you

You might want to check out these people:

Chloe Evans
Jack Norwich
Harry Walker
Lily Martin

Administrator was awarded the badge **Pulsarohani**. Posted at least 10 times
just now

Emily Jones Welcome **Chloe Evans**

SuperCorp is looking to acquire Uco...can you give some advice to **Jack Norwich**
4 days ago - Comment - Unlike

You like this.
Administrator This is a comment
just now

Add a comment

Emily Jones followed 4 pages

```
- <result>
 - <activities>
  - <activity>
   - <to>
      <type>user</type>
      <name>Administrator</name>
      <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-4fcb7d6b5610</url>
    </to>
    <type>badge awarded</type>
    <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-4fcb7d6b5610/activities/740ad128-
      1f61-434e-a06c-7ae3bb9c367e</url>
    <text>was awarded the badge Pulsarohani</text>
    <caption>Pulsarohani</caption>
    <totalLikes>0</totalLikes>
    <totalComments>0</totalComments>
   - <author>
      <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-4fcb7d6b5610</url>
      <name>Administrator</name>
      <type>user</type>
      <image>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
        4fcb7d6b5610/avatar</image>
    </author>
    <created>2010-09-14T22:05:54.533</created>
    <mentions />
  </activity>
  - <activity>
   - <to>
      <type>user</type>
      <name>Emily Jones</name>
      <url>http://pulse/services/streams/fa722b45-7783-4fa9-8547-7f773ba258f6</url>
    </to>
    <type>status</type>
    <url>http://pulse/services/streams/fa722b45-7783-4fa9-8547-
      7f773ba258f6/activities/2c6084d9-62d7-448f-9b77-f8a00ac39624</url>
    <text>Welcome @chloe SuperCorp is looking to acquire Uco...can you give some advice to
      @jack</text>
    <totalLikes>1</totalLikes>
    <totalComments>1</totalComments>
   - <likes>
      <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-4fcb7d6b5610</url>
      <name>Administrator</name>
      <type>user</type>
      <image>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
        4fcb7d6b5610/avatar</image>
    </likes>
   - <comments>
      <type>comment</type>
```

Example: Using .NET to execute Pulse operations

The following is a basic example of using .NET to interact with the Pulse API

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.IO;
using System.Xml;
namespace Neudesic.Pulse.Prototypes.Common
{
    public class PulseUtility
    {
        public static string PostStatusUpdate(string streamUrl, string
text, string link)
        {
            string postData = CreateStatusUpdateBody(text, link);
            byte[] byteArray = Encoding.UTF8.GetBytes(postData);
            StringBuilder result = new StringBuilder();

            WebRequest post = WebRequest.Create(streamUrl);
            {
                post.Method = "POST";
                post.ContentType = "text/xml";
                post.ContentLength = byteArray.Length;
                post.Credentials =
System.Net.CredentialCache.DefaultCredentials;
            }


            Stream dataStream = post.GetRequestStream();
            {
                dataStream.Write(byteArray, 0, byteArray.Length);
                dataStream.Close();
            }
            WebResponse response = post.GetResponse();
            {

result.AppendLine(((HttpWebResponse)response).StatusDescription);
                dataStream = response.GetResponseStream();
            }
            StreamReader reader = new StreamReader(dataStream);
            string responseFromServer = reader.ReadToEnd();
            result.AppendLine(responseFromServer);
            reader.Close();
            dataStream.Close();
            response.Close();
            return result.ToString();
        }
        public static string PostStatusUpdateUsingWebClient(string
streamUrl, string text, string link)
        {
            string postData = CreateStatusUpdateBody(text, link);
            byte[] byteArray = Encoding.UTF8.GetBytes(postData);
            WebClient wc = InitWebClient();

            string response = wc.UploadString(streamUrl, postData);
            return response;
        }


        public static string GetUserStreamUrl(string pulseBaseUrl, 
string userName)
        {

            using (WebClient client = InitWebClient())
            {
                client.Headers[HttpRequestHeader.ContentType] = 
"text/xml";
                byte[] result = client.DownloadData(pulseBaseUrl + 
"/services/streams/" + userName);
                string xml = 
System.Text.ASCIIEncoding.ASCII.GetString(result);
                XmlDocument doc = new XmlDocument();
                {
                    doc.LoadXml(xml);
                    return 
doc.SelectSingleNode("//stream/url").InnerText;
                }
            }
        }
        public static string GetStream(string pulseBaseUrl, string 
streamId)
        {
            WebClient wc = InitWebClient(true);
            byte[] result = wc.DownloadData(pulseBaseUrl + 
"/services/streams/" + streamId);
            return System.Text.ASCIIEncoding.ASCII.GetString(result);

        }
        public static string GetStream(string streamUrl)
        {
            WebClient wc = InitWebClient(false);
            byte[] result = wc.DownloadData(streamUrl);
            return System.Text.ASCIIEncoding.ASCII.GetString(result);

        }
        private static WebClient InitWebClient(bool json=false)
        {
            WebClient wc = new WebClient();
            {
                if(json)
                    wc.Headers[HttpRequestHeader.ContentType] = 
"application/json";
                else
                    wc.Headers[HttpRequestHeader.ContentType] = 
"text/xml";
                wc.Encoding = Encoding.UTF8;
                wc.Credentials = 
System.Net.CredentialCache.DefaultCredentials;
            }
            return wc;
        }
        private static string CreateStatusUpdateBody(string text,
string link)
        {
            StringBuilder body = new StringBuilder();
            body.Append("<status>");
            if (!String.IsNullOrEmpty(text))
            {
                body.Append(String.Format("<text>{0}</text>", text));
            }
            if (!String.IsNullOrEmpty(link))
            {
                body.Append(String.Format("<link>{0}</link>", link));
            }
            body.Append("</status>");
            return body.ToString();
        }

    }
}
```

REST API Conventions

The Pulse Web API is a RESTful, resource-oriented architecture that is made available to developers of custom client applications and for use in system integration scenarios. The only requirements for use of the API are the ability to make HTTP requests and the ability to parse XML responses. The section covers the following topics:

REST API Overview

The Pulse REST API adheres to REST web standards. Designed to be resource centric, the API keeps data references in the URL path and uses links to tie resources together. Method information goes into the standard HTTP verbs. For example, if you were interested in retrieving information about a specific Pulse Stream, you might construct the following HTTP request

```
GET /services/streams/<activity-id>
```

Since you want to retrieve information you use the HTTP GET verb. The path /services/streams/<activity-id> references the resource for information about a specific Stream. The response that comes back not only includes information about the Stream but also includes links or references that you can use to retrieve more information such as links to the *User* that the *Message* was *To*. For example, the following illustrates the data a response might include:

```
<result>
  <activities>
    <activity>
```

```
      <to>
        <type>user</type>
        <name>Administrator </name>
        <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610</url>
      </to>
…
    </activity>
…
</result>
```

You have access to Pulse web functionality using a programmatic REST API. Each API function or method follows the REST conventions. That is, each API method:

- Is modeled as a resource
- Has links to its children and any related resources
- Is invoked using one of the HTTP verbs: POST, GET, PUT, and DELETE.

This section describes the general API conventions and protocols. The reference section contains detailed descriptions for each API resource, including how the resource works with the four API verbs…

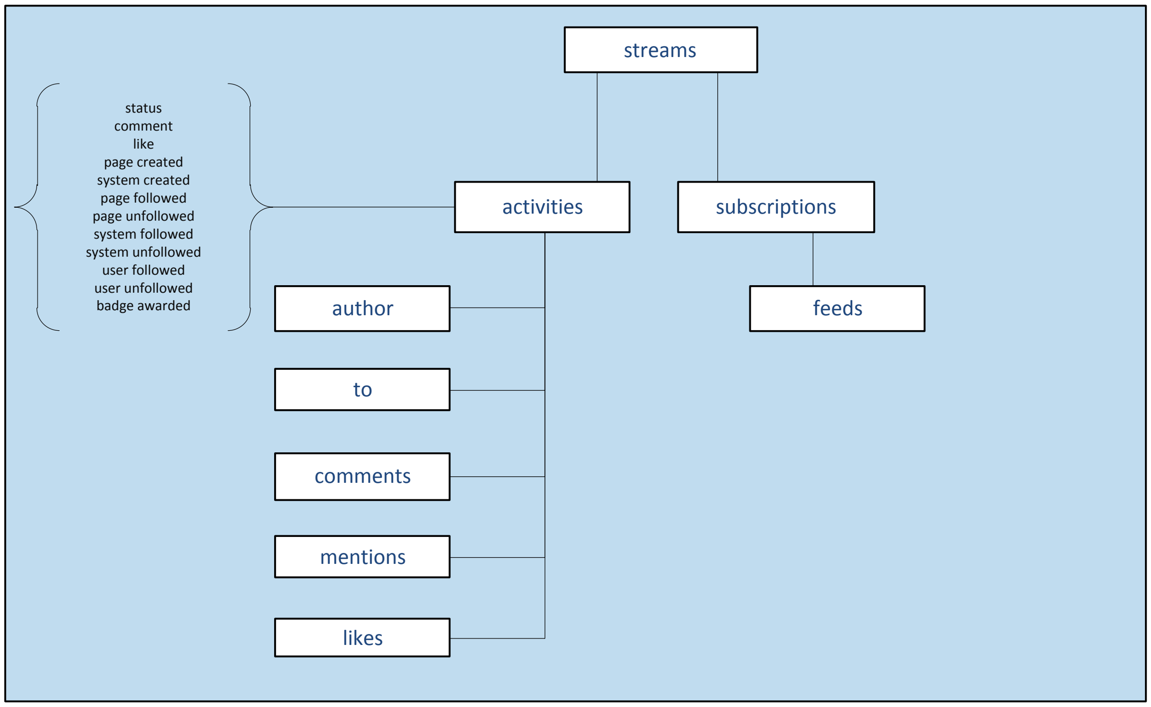
Basic Terminology

This section defines some of the basic terminology.

- **User**: Represents a user within Pulse
- **System**: Represents a system within Pulse which happens to be a special type of User
- **Page**: Represents a Stream that can in fact have unique permissions
- **Stream**: All users, systems and pages have Streams. Streams provides links to the activities and subscriptions for that stream
- **Activity**: Represents an event within Pulse. For example, a comment, a like, a page created, etc…
- **Subscription**: Represents a subscription to a Stream. This basically states that, for example, *"User A **follows** Stream A"*

Organization of the Pulse Resources

The API resources can be represented by the following diagram



Resource Details

This section provides details on each of the values returned within all resources of the Pulse REST API.

Activity

An Activity is a resource that links to several other resources. The following list each attribute of an Activity.

Supported HTTP Methods

GET, POST, DELETE

Attributes

Element	Description	Type
to	The feed that the message was posted to	To
type	The type of message (status, comment, like, etc.)	String
url	A unique URL representing the activity.	String
text	The text of the activity	String
picture	<todo>	String

link	<todo>	String
caption	<todo>	String
description	<todo>	String
totalLikes	The total number of likes that the message has received	Int
totalComments	The total number of comments that the message has received	Int
likes	<todo>	Collection of **Author**
comments	<todo>	Collection of **Activity**
author	The author of the message	Author
client	<todo>	Client
location	<todo>	Location
created	The date and time that the message was posted	DateTime
on	<todo>	String
mentions	<todo>	Collection of **Mention**

Example: GET

```
    <activity>
      <to>
        <type>user</type>
        <name>Emily Jones</name>
        <url>http://pulse/services/streams/<stream-id></url>
      </to>
      <type>status</type>
      <url>http://pulse/services/streams/<stream-
id>/activities/<activity-id></url>
      <text>
        Welcome @chloe

        SuperCorp is looking to acquire Uco...can you give some advice
to @jack
      </text>
      <totalLikes>1</totalLikes>
      <totalComments>1</totalComments>
      <likes/>
      <comments/>
      <author>
        <url>http://pulse/services/streams/<stream-id></url>
        <name>Emily Jones</name>
        <type>user</type>
        <image>http://pulse/services/streams/<stream-id>/avatar</image>
      </author>
      <created>2010-09-10T17:06:24.95</created>
      <mentions/>
    </activity>
```

Example: POST

The following are examples of well formatted POSTs to a Stream's Activities

Posting a text Message

To make a simple text based post, execute an HTTP POST request to the Activities of the intended Stream containing a body in the following format. For example:

URL

```
http://pulse/services/streams/<stream-id>/activities
```

Body

```
<status>
  <text>Example Status Update</text>
</status>
```

Response

Inspect the Response Headers "Location" for the URL to the newly created Activity

```
Location: http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/activities/aca2aeaa-e7a1-4e3f-b6fa-d9549eb4ca58
```

Posting an Image

Posting an image is a 2 step process. First you must use the UploadImage service to upload the image. UploadImage will return you a link to the image location. Use this link in your POST. For example:

URL

```
http://pulse/services/streams/<stream-id>/activities
```

Body:

```
<status>
  <text>This is an image post</text>
  <link>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/images/3739954b-4c76-4d59-9cf4-
b32c3dd63753?size=150</link>
</status>
```

Response:

Inspect the Response Headers "Location" for the URL to the newly created Activity

```
Location: http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/activities/aca2aeaa-e7a1-4e3f-b6fa-d9549eb4ca58
```

Posting a File

Posting a file is a 2 step process. First you must use the UploadFile service to upload the image. UploadFile will return you a link to the image location. Use this link in your POST. For example:

URL

```
http://pulse/services/streams/<stream-id>/activities
```

Body:

```
<status>
  <text>This is an file post</text>
  <link>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/files/484601ee-1552-485c-b3b5-a6946b5b1928"</link>
</status>
```

Response

Inspect the Response Headers "Location" for the URL to the newly created Activity

```
Location: http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/activities/aca2aeaa-e7a1-4e3f-b6fa-d9549eb4ca58
```

Posting a Like

To like or unlike a specific activity, append "/likes" to the activity URL and make a POST request containing a body in the following format:

URL

```
http://pulse/services/streams/<stream-id>/activities/<activity-
id>/likes
```

Body:

```
<like>
  <isLiked>true</isLiked>
</like>
```

Response

```
HTTP 200
```

Posting a Unlike

To like or unlike a specific activity, append "/likes" to the activity URL and make a POST request containing a body in the following format:

URL

```
http://pulse/services/streams/<stream-id>/activities/<activity-
id>/likes
```

Body:

```
<like>
  <isLiked>false</isLiked>
</like>
```

Response

```
HTTP 200
```

Posting a Comment

To post a comment about a specific message, append "/comments" to the activity URL and make a POST request containing a body in the following format:

URL

```
http://pulse/services/streams/<stream-id>/activities/<activity-
id>/comments
```

Body:

```
<comment>
  <text>text<text>
</comment>
```

Response

```
HTTP 200
```

Example: DELETE

To delete an Activity execute an HTTP DELETE request to the Activity Url. For example,

```
DELETE /services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/activities/30685f0d-e8fd-4458-9ef3-247a740fa442
```

Stream

Author

An author is the creator of an Activity

Supported HTTP Methods

NONE

Attributes

Element	Description	Type
type	The type of author (user, system)	String
url	A unique URL to the authors stream	String
name	The name of the Author	String

image	The image/avatar of the Author	String

Example

```
      <author>
        <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610</url>
        <name>Administrator</name>
        <type>user</type>
        <image>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/avatar</image>
      </author>
```

To

Represent the user, system, or page the activity was directed to.

Supported HTTP Methods

NONE

Attributes

Element	Description	Type
type	The type of To (user, system, page)	String
url	A unique Url to the stream of the "To"	String
name	Name of the User, System, or Page the activity was directed to.	String

Example

```
      <to>
        <type>user</type>
        <name>Administrator </name>
        <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610</url>
      </to>
```

Mention

Supported HTTP Methods

POST

Attributes

Element	Description	Type
text	The feed that the message was posted to	string
url	A unique URL representing the activity.	String
name	The text of the activity	String

Client

Element	Description	Type
name	TODO	String

Location

Element	Description	Type
name	TODO	String
geo	TODO	String

Embedding Pulse Conversations

Pulse supports the ability to add conversations to any application that supports HTML. For example, if you have a custom JSP or ASP.NET application, you can immediately embed Pulse. This uses a common web pattern of embedding javascript that is responsible for the rendering of the user experience. The following is a walkthrough to accomplish this…

Basics

Any Pulse Activity stream can be embedded. You can get to the necessary embed link simply by appending **/embed** to any Activity Url. For example, assume the following URL is a Pulse Activity stream

```
http://pulse /streams/3538058b-6cd8-4a4a-89f8-d8c514225df5/
```

By appending **/embed** you will be given the appropriate javascript embed code. For example, the following URL

```
http://pulse /streams/3538058b-6cd8-4a4a-89f8-d8c514225df5/embed
```

Returns

```
var dFrame = document.createElement("iframe"); dFrame.name = "pulse-
embed"; dFrame.className = "pulse-embed"; dFrame.frameBorder = 0;
dFrame.scrolling = 'NO'; dFrame.style.width = "550px";
dFrame.style.height = "550px"; dFrame.src =
'http://pulse/streams/a05b835f-7f2f-47bf-9536-
aa14bd449f85/chromelessactivities?view=&xdPath=&url='+escape('/Streams/
a05b835f-7f2f-47bf-9536-aa14bd449f85/embed');
document.body.appendChild(dFrame); function setContentHeight(h) { if(h
!= null && h != "") { dFrame.style.height = h +"px"; } };
```

You can use this is combination with a **DIV** tag to embed an existing Pulse conversation in any application that supports HTML.

Embedding an existing Pulse Conversation

Here is a complete example using the URL above

```
<html>
<body>
<!-- DIV TAG TO INSERT PULSE INTO -->
<!-- NOTE: The DIV id is passed as a querystring parameter to the
/embed -->
<div id="myPulseDiv" />
<SCRIPT type=text/javascript src="
http://pulse.neudesic.com/streams/3538058b-6cd8-4a4a-89f8-
d8c514225df5/embed?targetElement=myPulseDiv"></SCRIPT>
</body>
```

Embedding the current Users Feed

Here is a complete example for embedding the current user's stream:

```
<html>
<body>
<!-- DIV TAG TO INSERT PULSE INTO -->
<!-- NOTE: The DIV id is passed as a querystring parameter to the
/embed -->
<div id="myPulseDiv" />
<SCRIPT type=text/javascript src="
http://pulse.neudesic.com/streams/my/embed?targetElement=myPulseDiv"></
SCRIPT>
</body>
```

Embedding Specific Users Feed

To get to a specific user's feed use the format <pulseUrl>/streams/r/<username>. For example, https://pulse.neudesic.com/streams/r/ty.carlson. The complete example would be

```
<html>
<body>
<!-- DIV TAG TO INSERT PULSE INTO -->
<!-- NOTE: The DIV id is passed as a querystring parameter to the
/embed -->
```

```
<div id="myPulseDiv" />
<SCRIPT type=text/javascript src="
http://pulse.neudesic.com/streams/r/ty.carlson?targetElement=myPulseDiv
"></SCRIPT>
</body>
```

Cross-browser and Sizing

For the best user experience we recommend extending the host application to include a file that allows for cross-domain/cross-window communication. This is done by creating a basic HTML file that references a javascript within the Pulse server. A basic, hard-coded example follows:

Hard-Coded Example(xd_pulse.html)

```
<html><body>
<script src="http://pulse.neudesic.com/scripts/sdk/xd_comm.js"
type="text/javascript"></script>
</body></html>
```

Adding xd_pulse.html to your application

This page needs to live in the same domain space as the hosting application. For example, if your custom web application is at the url: **http://myapp.mycompany.com** you will want your *xd_pulse.html* page to exist within that namespace. For example, **http://myapp.mycompany.com/xd_pulse.html**

Referencing xd_pulse.html

Once you have created the xd_pulse.html page, you will want to update your EMBED to include a pointer the page. The **/embed** accepts a parameter **xdPath**. xdPath should be a relative pointer to your xd_pulse.html. For example:

```
<SCRIPT type=text/javascript src="
http://pulse.neudesic.com/streams/my/embed?targetElement=myPulseDiv&xdP
ath=xd_pulse.html"></SCRIPT>
```

Example String Formats for Embedding

```
        public static readonly string ExistingFeedFormat = "<div
id=\"{0}\"></div><script type=\"text/javascript\"
src=\"{1}/embed?url={2}&targetElement={0}&xdPath={3}&customCss={4}\"></script>";
        public static readonly string CurrentUserEmbedFormat = "<div
id=\"{0}\"></div><script type=\"text/javascript\"
src=\"{1}/streams/my/embed?url={2}&targetElement={3}&view=personal&xdPath={4}&cust
omCss={5}\"></script>";
        public static readonly string ThisItemEmbedFormat = "<div
id=\"{0}\"></div><script type=\"text/javascript\"
src=\"{1}/streams/pages/embed?title={2}&url={3}&targetElement={0}&xdPath={4}&custo
mCss={5}\"></script>";
```

```
        public static readonly string PulseFeatureEmbedFormat = "<div 
id=\"{0}\"></div><script type=\"text/javascript\" 
src=\"{1}/embed/{2}url={3}&targetElement={0}&xdPath={4}&customCss={5}\"></script>"
;
        public static readonly string SpecificUserStreamEmbedFormat = "<div 
id=\"{0}\"></div><script type=\"text/javascript\" 
src=\"{1}/embed/streams/r/{2}?url={3}&targetElement={0}&view=personal&xdPath={4}&c
ustomCss={5}\"></script>";
```

Example: Using Javascript to build a Pulse conversation

The following example uses Javascript to build a custom Pulse conversation for the current web page.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" 
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Client-side Item Level Conversations</title>

<script type="text/javascript">
    var pulseUrl = 'https://pulse.neudesic.com';
    var pagesUrl = '/streams/pages/embed?';

    function getEmbed(title, url, xdPath) {
        return pulseUrl + pagesUrl + 'title=' + escape(title) + '&url=' + 
escape(url) + '&xdPath=' + escape(xdPath);
    }
    function insertScript(src) {
        var headID = document.getElementsByTagName("head")[0];
        var newScript = document.createElement('script');
        newScript.type = 'text/javascript';
        newScript.src = src;
        headID.appendChild(newScript);
    }
    function getUrl() {
        return window.location;
    }
    function getTitle() {
        return document.title;
    }
    function getXdPath()
    {
        return 'xd_pulse.html';
    }
</script>
</head>
<body>
    <div id='myPulseDiv'></div>
    <script type="text/javascript">
        insertScript(getEmbed(getTitle(), getUrl(), getXdPath()));
    </script>
</body>
</html>
```

Common API Tasks

This section provides examples for common API tasks. Each provides an Example URL, and Example Body when applicable.

Getting a User's Profile

To view a user's profile information, navigate to the author URL. This will give you the URLs of any feeds exposed by the user, as well as the basic profile information for the user.

Example URL

```
http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-4fcb7d6b5610
```

Retrieved From

```
    <activity>
      <to>
        <type>user</type>
        <name>Administrator </name>
        <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610</url>
      </to>
      …
    </activity>
```

Example Result

```
<stream>
  <type>user</type>
  <feeds>
    <name>Administrator </name>
    <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/activities</url>
    <isPublic>true</isPublic>
  </feeds>
  <name>Administrator</name>
  <emailAddress>administrator@office.neudesic.dev</emailAddress>
  <firstName>Administrator</firstName>
  <title>Pulse Administrator</title>
  <location>Irvine</location>
  <imageUrl>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/avatar</imageUrl>
  <phone>555.555.5555</phone>
  <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610</url>
</stream>
```

Getting a User's Subscriptions

To view a user's subscriptions append "/subscriptions" to the user URL. This will give you their subscriptions in the following format:

Example URL

```
http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-4fcb7d6b5610/subscriptions
```

Retrieved From

```
    <activity>
      <to>
        <type>user</type>
        <name>Administrator </name>
        <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610</url>
      </to>
      …
    </activity>
```

Example Result

```
<result>
  <feeds>
    <url>http://pulse/services/streams/65e78527-f88c-4726-8164-
5f0945d09889/activities</url>
    <name>Zenflex</name>
  </feeds>
  <feeds>
    <url>http://pulse/services/streams/2608a488-6b87-4b5b-adb0-
3f884fdb5d18/feeds/f036ddbb-8718-4b47-8ca1-
2c327c76fb45/activities</url>
    <name>Wins and Losses</name>
  </feeds>
</result>
```

Getting a User's Avatar

User avatars may be requested by navigating to the **imageUrl** listed in the avatar node. To request the avatar at a different resolution, add a size parameter to the query string (example ?size=50).

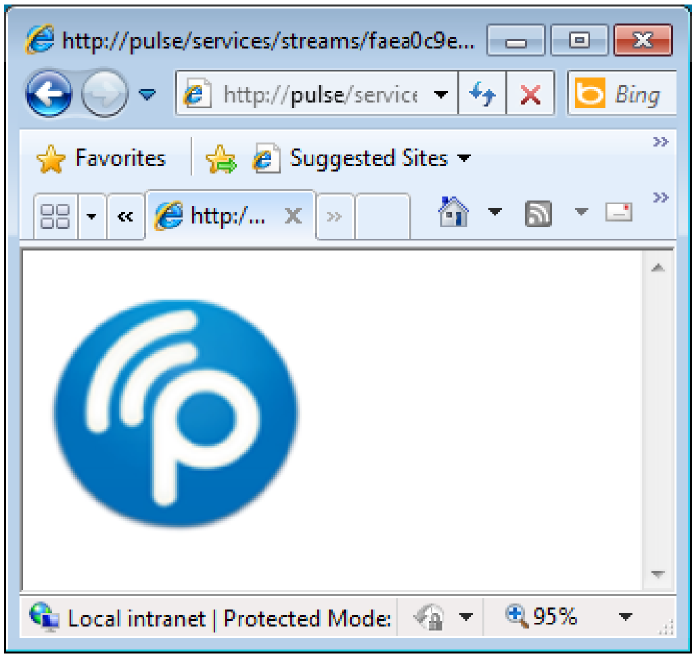
Example URL

```
http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/avatar
```

```
<stream>
  <type>user</type>
  <feeds>
```

```
    <name>Administrator </name>
    <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/activities</url>
    <isPublic>true</isPublic>
  </feeds>
  <name>Administrator</name>
  <emailAddress>administrator@office.neudesic.dev</emailAddress>
  <firstName>Administrator</firstName>
  <title>Pulse Administrator</title>
  <location>Irvine</location>
  <imageUrl>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/avatar</imageUrl>
  <phone>555.555.5555</phone>
  <url>http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610</url>
</stream>
```

Example Result


http://pulse/services/streams/faea0c9e...
http://pulse/service
Bing
Favorites
Suggested Sites
http:/...
Local intranet | Protected Mode:
95%

Modifying Subscriptions

To modify the state of a subscription to a feed, POST a message to the user's subscription URL in the following format:

Example URL

```
http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/subscriptions
```

Example Input

```
<subscribe>
  <url> http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610</url>
  <isSubscribed>true</isSubscribed>
</subscribe>
```

Getting a list of Systems

TODO

Create a System

TODO

Create a Page

To create a Page use the "`/services/public/streams/pages`" and make a POST request containing a body in the following format:

URL

```
http://pulse/services/public/streams/pages
```

Body:

```
<page>
  <name>My Page</name>
  <description>My Page</description>
  <isBrowseable>true</isBrowseable>
</page>
```

Response

Inspect the Response Headers "Location" for the URL to the newly created Page

```
Location: http://pulse/services/streams/faea0c9e-423f-4f61-bfd9-
4fcb7d6b5610/activities/aca2aeaa-e7a1-4e3f-b6fa-d9549eb4ca58
```

Getting a list of Pages

TODO

Getting a list of People

TODO

Getting the URL to a User's Profile using a Login Name

TODO

Follow a Person

TODO

Unfollow a Person

TODO