Tonido HTTP API Guide

Codelathe LLC
http://www.tonido.com
7/5/2011

CONTENTS

1 INTRODUCTION

Tonido HTTP API allows clients such as web browsers, mobile devices to programmatically connect and access Tonido Instances running on your Personal Computer or TonidoPlug or any other device.

2 HTTP BASICS

HTTP is the fundamental protocol of the World Wide Web. HTTP is a connectionless request response protocol, meaning there is no concept of a persistent connection between a series of requests.

2.1 REQUEST AND RESPONSE

An HTTP request is a message sent from the client to the server. The server sends back a response. The request and response might contain content called the body. In addition, the response always contains a numeric response code, which lets us know if the request was successful. It gives more detailed information about what exactly happened (e.g. the cause of failure).

2.2 METHODS

HTTP supports several request methods, which help the server know how to handle the request. The HTTP methods for our purposes are:

GET: Used to retrieve a resource (such as a web page or image) from a URL

POST: Used to send data to a server (such as the content of a form) based on a URL

2.3 HEADERS

Finally, in addition to the main content of the request and response, HTTP allows additional data to be sent in the form of headers. They can be sent with the request to the server and the response from the server, and they can contain arbitrary text data. There are many standard headers, and the connection API contains methods for easily accessing some of the most common ones.

3 HTTP INTERFACE

Tonido includes an embedded HTTP server in the runtime. This server handles incoming HTTP calls, transforms it into internal Tonido System Messages and sends it to the appropriate internal services.

4 WARM UP EXERCISE

If you are reading this document, you must be already running a Tonido instance on your PC or your TonidoPlug. If not, you can download pre-built Tonido binaries from the Tonido website (http://www.tonido.com) for your operating system. Just install the binaries and open the webpage (http://127.0.0.1:10001). You will be able to create a Tonido profile and login into Tonido.

After you have successfully logged in, call the *getauthenticationinfo* operation by typing the following URL.

http://james.tonidoid.com/core/getauthenticationinfo (where james is the profile name)

If you are running local Tonido instance, you can connect to the following URL

http://127.0.0.1:10001/core/getauthenticationinfo

Please Note: HTTP (GET/POST) response from Tonido HTTP Server will always be in a XML format.

Tonido web server will return the following XML response.

```
<authenticationinfo>
  <info>
        <profile>james</profile>
        <displayname>james</displayname>
        <peerid>james@tonidoid.com</peerid>
        <authenticated>0</authenticated>
        <isremote>1</isremote>
        <reasoncode>0</reasoncode>
  </info>
</authenticationinfo>
```

Most of the XML elements above are self explanatory, however do not worry if some properties are not clear. We will dive deep as we go through this document.

Let us try to analyze the URL some more. The Tonido HTTP server expects URL to be of a certain form.

HTTP://HOST:PORT/CORE/OPERATION {? OPTIONAL PARAMS VIA GET OR POST}

The "***CORE***" in the above URL refers to Tonido Core, which is the primary software component that provides programmable APIs, manages and provides HTTP and P2P connectivity. For the purpose of this document we will limit ourselves to HTTP connectivity only.

In our first example, we were using *getauthenticationinfo* as the operation that gets the basic authentication detail such as profile name, TonidoID etc of the profile logged into Tonido server.

The Optional Params can be used to pass parameters from client to server. They can either be passed via HTTP Get URL parameters or as POST parameters.

5 TONIDO LOGS

Tonido logging offers a simple but powerful means to follow and debug the communication between your client and Tonido HTTP server. From the Tonido settings screen, go to activity logs tab and set the Log Level to TRACE for maximum information.

To view the log file, you can use note pad or a Log file monitoring tool such as Bare Tail by Bare Metal Soft in Windows. http://www.baremetalsoft.com/baretail/

As an additional activity, repeat the Warm-up exercise and follow the Tonido logs.

6 AUTHENTICATION

In Tonido the first fundamental requirement is the ability to identify a Tonido instance (or a device or user) uniquely. So the first time you install Tonido and open it up, you will have to create a new Tonido account.

After account creation, you login into that account using the password you used. When you login correctly, an authentication cookie is sent back to the browser which is used for all subsequent HTTP calls.

The client has to pass this cookie back to the server with all the calls going forward to successfully communicate with the Tonido HTTP server. Usually, passing of cookies between client and server is handled automatically in the browser. However, if your client does not handle this communication automatically you have to capture and pass this cookie through HTTP header from the client to the server.

7 "HELLO WORLD" FOR TONIDO

"Hello World" for Tonido is not exactly the program that prints Hello World. We will see how to login and authenticate into Tonido. These calls are fundamental to understand the XML messages that are used to communicate between the Tonido server and client.

After starting Tonido, from your browser, login and go to Account tab under settings and make sure you have the optional Remote Login Question and Answer specified. When logged into Tonido from a remote location, in addition to account name and password the user must answer the Remote Login Question.

Remote Login Question is an additional security measure to prevent "phishing" attacks and unauthorized access into Tonido account.

Authentication is a two step process. First, a *loginprofile* command is issued with account name and password. If successful, the server will return the Remote Login Question. Second, an *authenticateprofile* command is issued with the remote login question, remote login answer and password.

Finally, when authentication is complete and successful, the Tonido HTTP server creates a cookie and returns the cookie value to the client through the HTTP header.

7.1 COMMAND: LOGINPROFILE

LoginProfile command accepts the account name and password and attempts to login into the profile. The URL follows the standard format as we discussed above.

HTTP://HOST:PORT/CORE/OPERATION {? OPTIONAL PARAMS }

When applied for *loginprofile* call, the above URL takes the form.

http://host:port/core/loginprofile

The input parameters (see table below) profile, password, safemode and autologin must be passed as name/value pairs from client to Tonido HTTP server as a part of HTML body through the HTTP **POST** method.

In the above *loginprofile* call, the clear text password must be converted to SHA-1 Hash by the client and sent to the server. Converting a clear text to SHA-1 Hash is programming language specific which we will not cover here. The safemode and autologin can pass a default value of Zero.

From the Tonido Client, issue an HTTP POST command with the above URL when your Tonido Development Instance is running. Tonido HTTP server will return an XML result in the following format.

```
<commands>
  <command>
     <type>loginprofile</type>
    <result>2</result>
    <message>FavoritCar </message>
  </command>
</commands>
```

In the above XML response, RESULT 2 indicates that the profile was logged in but not authenticated. An *authenticateprofile* command must be issued along with remote question and answer to authenticate the profile.

As an additional help, use the Tonido Logs to follow the communication between your client and Tonido HTTP server.

Input **HTTP Method: POST**

PARAMETER	Required	DATA TYPE	DESCRIPTION
Profile	Yes	String	account name
Password	Yes	String	SHA-1 hash value of the password
Safemode	Optional. Default 0	Number	Login with safe mode accepted values 0 or 1
Autologin	Optional Default 0.	Number	Allow autologin into this account. Accepted values 0 or 1.

Output

PARAMETER	DESCRIPTION
Type	Name of the call. Possible value is "loginprofile"
Result	Returns 1 if success, 0 is failed and 2 means expects remote login answer.
Message	Returns any error message generated in the call. If Result is 2 then message indicates the "Remote Login Question"

Please note that if the Remote Login Question is not specified for the account then XML response received from the Tonido HTTP server will be

```
<commands>
  <command>
     <type>loginprofile</type>
    <result>1</result>
    <message> </message>
  </command>
</commands>
```

In the above XML response, RESULT 1 indicates profile was logged in and authenticated.

However, a RESULT 0 indicates the *loginprofile* failed. Most likely cause will be password is not correct or the SHA-1 Hash generated for the password does not match the SHA-1 Hash stored in the Tonido Instance.

7.2 COMMAND: AUTHENTICATE PROFILE

AuthenicateProfile command is issued to authenticate a profile with the remote login question and answer. The HTTP POST request follows the following format.

http://host:port/core/authenticateprofile

The input parameters (see table below) secretqn, secretans, and password must be passed as name/value pairs from client to Tonido HTTP server as a part of HTML body through the HTTP Post method. Please note that password is SHA-1 Hash of the clear text password.

Input **HTTP Method: POST**

PARAMETER	Required	DATA TYPE	DESCRIPTION
secretqn	Yes	String	Secret question
secretans	Yes	String	Secret Answer in clear text
password	Yes	Number	SHA-1 hash value of the password

Output

PARAMETER	DESCRIPTION
Type	Name of the call. Possible value is "authenticateprofile"
Result	Returns 1 if success, 0 is failed.
Message	Returns any error message generated in the call.

8 TONIDO XML RESPONSE TYPES

XML response received from Tonido HTTP server follows a standard pattern with items, meta and item element.

Following is the standard XML pattern received from Tonido HTTP server.

```
<items>
  <meta>
    <metaelement1>[Meta Element 1 value] </metaelement1>
    <metaelement2>[Meta Element 2 value] </metaelement2>
  </meta>
  <item>
    <itemdetail1>[Item detail 1 value]<itemdetail1>
    <itemdetail2>[Item detail 1 value]<itemdetail2>
    <itemdetail3>[Item detail 1 value]<itemdetail3>
  </item>
</items>
```

The items and item element will be renamed appropriately for each XML response received as shown below in the samples. On the other hand, meta element is optional.

Following are some of the sample XML response received.

In the *loginprofile* response below, the items and item element are replaced with commands and command element respectively. The command element has type, result and message elements as itemdetail.

```
<commands>
   <command>
      <type>loginprofile</type>
      <result>1</result>
      <message> </message>
   </command>
</commands>
```

In the *getfilelist* response below, the items and item element are replaced with entries and entry element respectively. The entry element has path, dirpath, name, ext, isroot, type, fullfilename, size, modified, favoritelistid, favoriteid, order, fullsize, modifiedepoch elements as itemdetail. Note that, the *getfilelist* also returns the optional meta element with parentpath and total.

```
<entries>
	<meta>
		<parentpath>C:\</parentpath>
		<total>2</total>
	</meta>
	<entry>
		<path>C:\Users\Public</path>
		<dirpath>C:\Users\</dirpath>
		<name>Public</name>
		<ext>n/a</ext>
		<isroot>0</isroot>
		<type>dir</type>
		<fullfilename />
		<size />
		<modified>Feb 6, 2011 11:18 AM</modified>
		<favoritelistid>0</favoritelistid>
		<favoriteid>0</favoriteid>
		<order>0</order>
		<fullsize>0</fullsize>
		<modifiedepoch>1297016311</modifiedepoch>
	</entry>
	<entry>
		<path>C:\Users\sean</path>
		<dirpath>C:\Users\</dirpath>
		<name> sean </name>
		<ext>n/a</ext>
		<isroot>0</isroot>
		<type>dir</type>
		<fullfilename />
		<size />
		<modified>May 29, 2011 11:08 AM</modified>
		<favoritelistid>0</favoritelistid>
		<favoriteid>0</favoriteid>
		<order>0</order>
		<fullsize>0</fullsize>
		<modifiedepoch>1306692531</modifiedepoch>
	</entry>
</entries>
```

9 OTHER API

Coming soon.