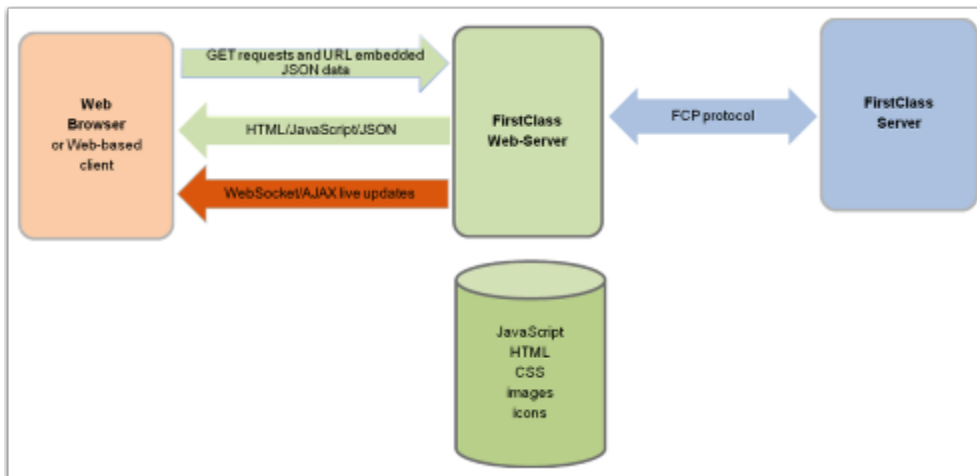


FirstClass 12 API

Introduction

An application programming interface (API) is a protocol intended to be used as an interface by software components to communicate with each other. The FirstClass Web Services server (FCWS) is a Python implementation of a web server that facilitates access to FirstClass server data and functionality. The FirstClass web client (or any other web-based client) uses HTML 5.0 to communicate with the FirstClass web server. Requests issued to the FirstClass web server are translated into the FirstClass-proprietary protocol FCP and sent to the FirstClass server for processing. The FirstClass server responses are also encapsulated in the FCP protocol, which is then processed by the FirstClass web server and subsequently sent back to the web client. The API is documented and web developers can use this API to create applications that can present secure, authenticated data within their own applications.



The FirstClass 12 API Developer Tool

A developer tool has been created to help web developers understand the API documentation and to provide some examples on how to retrieve and parse information returned from an API call to the FirstClass server.



Logging In

As mentioned, the current implementation of the API requires an authenticated login. The tool allows you to choose between a base64 sha 512 hash encrypted or a clear text password. After clicking the Login button, you will see the following:

1. The URL that is sent to the server along with any POST data that accompanies the request.
2. The data that is returned from the server in its raw JSON form.
3. A sample of how a web developer may wish to parse and display the returned data.

FCWS Server: www.firstclassdemo.com Port: 8000

UserID: jschmo
Password: **** ☒ sha 512 hash ☐ Clear Text

Select an Option:

Post/Get

http://www.firstclassdemo.com:8000/?ReplyAsJSON
Data Sent: JSON={"login":
["userid":"jschmo","sha512digest":"NTQ4OWUwZjAzZjVhWmMzMUwY2VlMjFhMjY0Y2RkMTVmNmNmWjI2"]}

Data Received

```
{
  "LOGINREPLY": {
    "DeskObjID": 0,
    "LastLogin": 3441353051,
    "TotalConnect": 639166,
    "DailyRemaining": 853613492,
    "AutoRegAvail": 0,
    "LinkVersion": 13,
    "ServerVersion": 11229,
    "PrefBits": 52256,
    "ServerTZ": 6554336,
    "PrefByte1": 0,
    "Features": 2145107639,
    "SysCharSet": 65001,
    "ServerTime": 3441353803,
    "Features2": 4293391328,
    "RemoteSiteID": 18840287,
    "WebSocketEnabled": 1,
    "SiteName": "FirstClass Demo",
    "DeskTopWebID": "DeskTop00000000000000000000000000000000"
  }
}
```

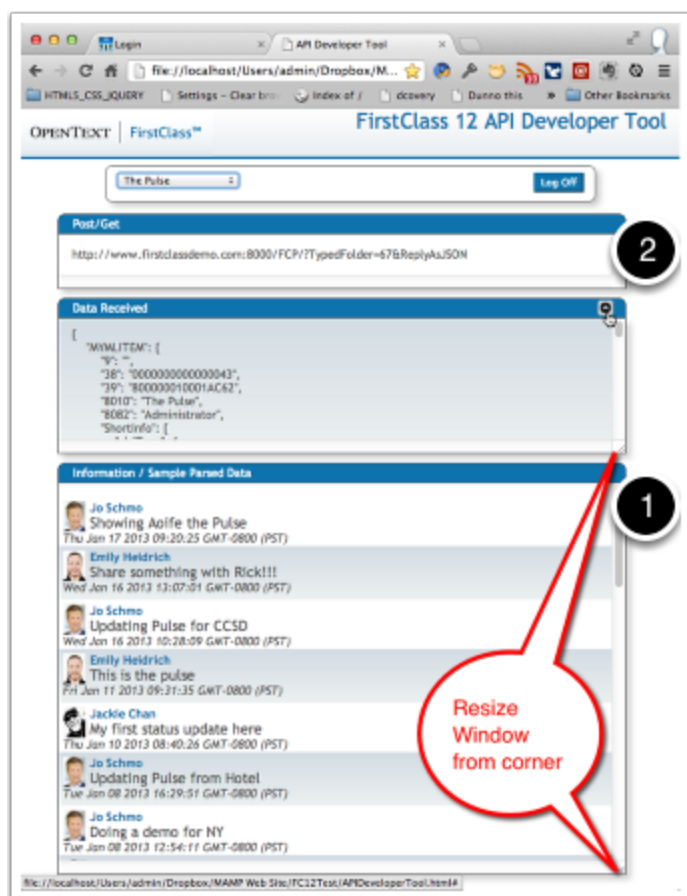
Information / Sample Parsed Data

Logged in to SiteID:	18840287
SiteName:	FirstClass Demo
Total time logged:	7 days, 9 hrs, 32 min, 46 secs
Last Login:	Fri Jan 18 2013 11:24:11 GMT-0800 (PST)
Server Time:	Fri Jan 18 2013 11:36:43 GMT-0800 (PST)
Daily time remaining:	Unlimited

The Interface

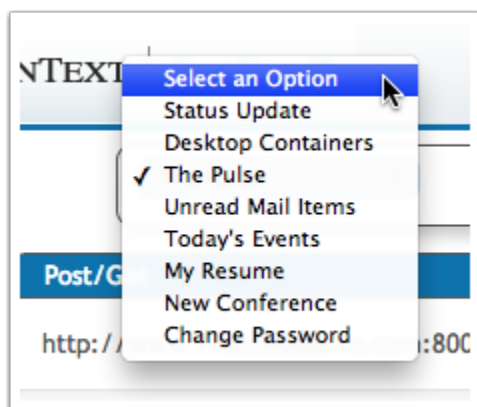
The tool has a simple interface that allows you to expand and shrink the Data Received and Information / Sample Parsed Data sections so you can better view the content.

1. Each window has a resizing corner that you can use to increase or reduce the height.
2. You can dismiss the Data Received window entirely by clicking on the collapse button. This is a toggle so clicking again will expand it back to its previous size.



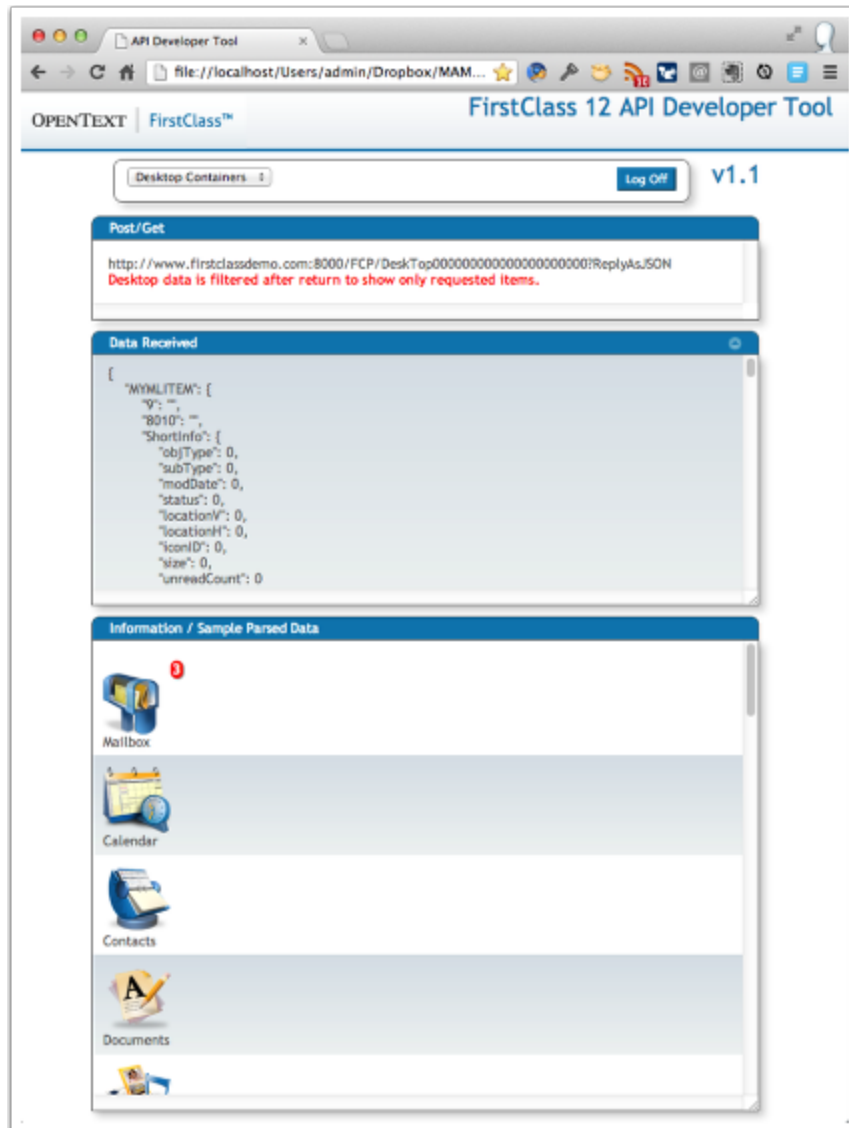
Options

A number of different options are available once logged in. An effort has been made to provide developers with a range of examples that might prove useful if they are trying to incorporate FirstClass content within their own web applications.



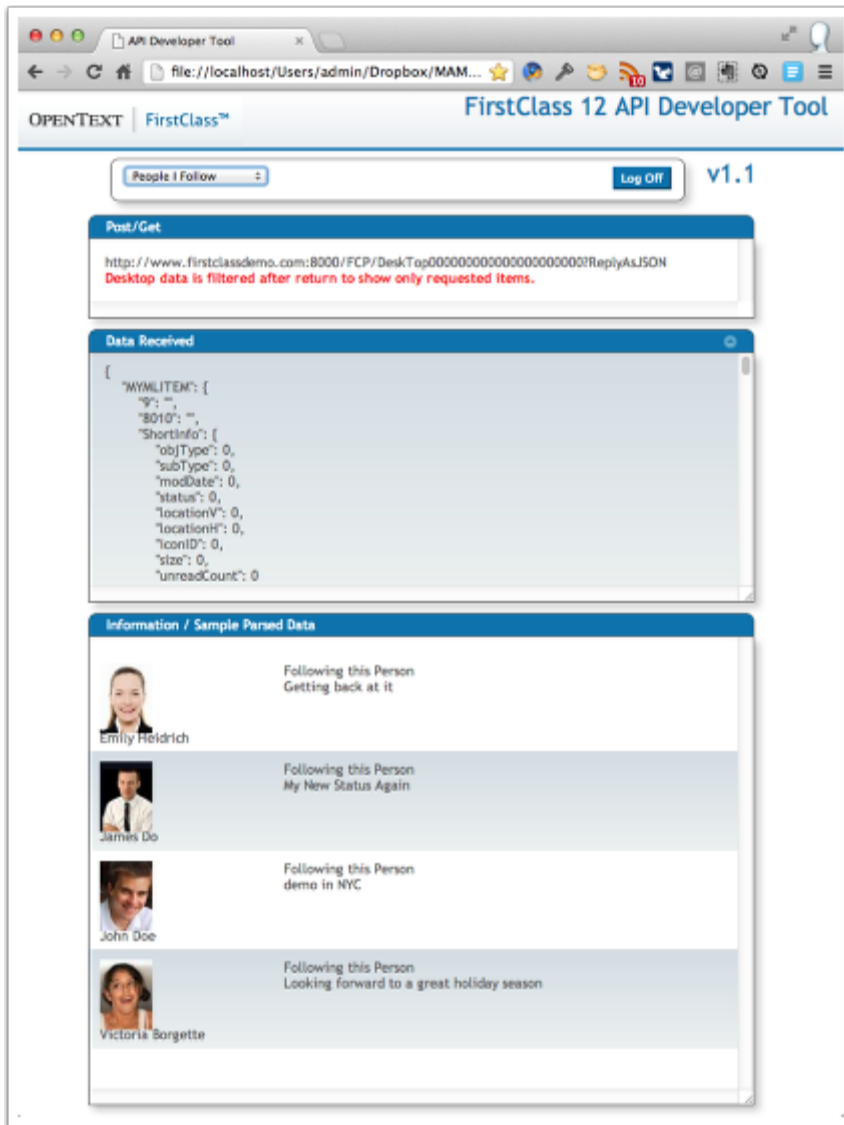
Desktop Containers

Selecting Desktop Containers will retrieve and display all containers on the user's desktop. This starts with the general call to open the user's desktop, fetching everything that is stored on the user's home screen. This includes items that are invisible to the user. In order to display containers only, the developer will need to filter the JSON results to only display the containers.



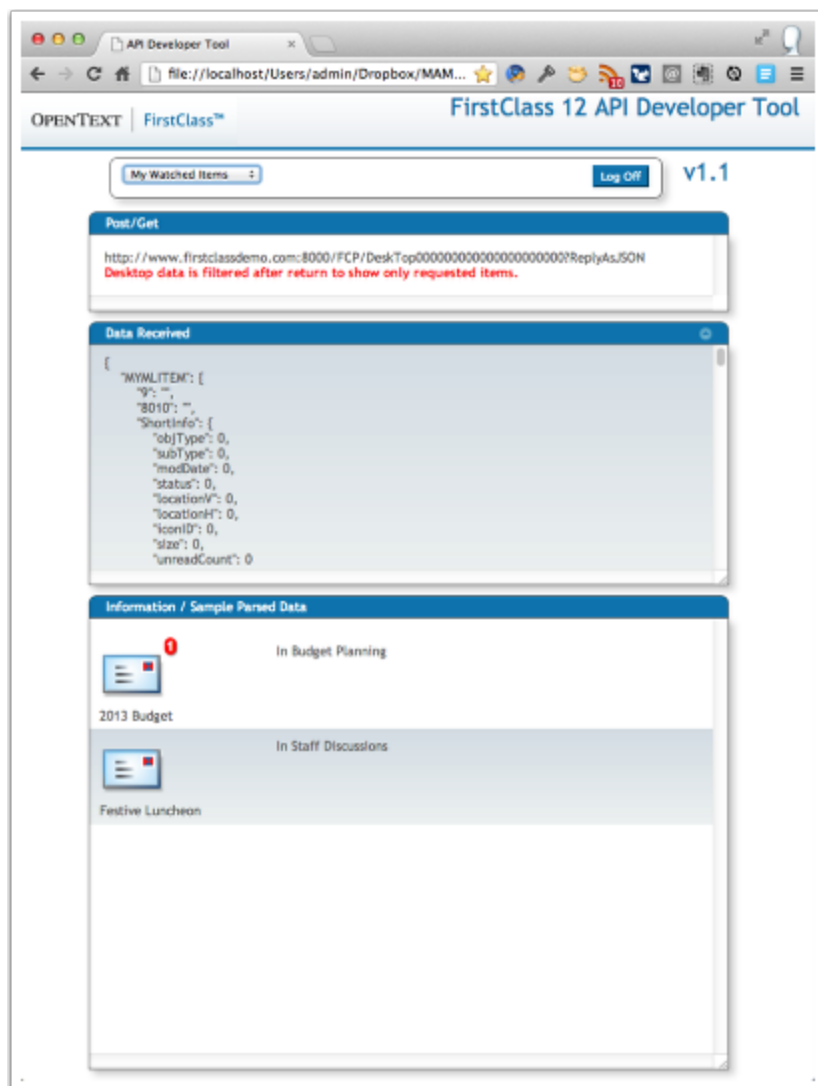
People I follow

As mentioned above, retrieving the desktop retrieves everything that is stored on the user's home screen including items that are invisible to the user. To retrieve the People I follow, the desktop is retrieved and the developer must filter the results to display only people.



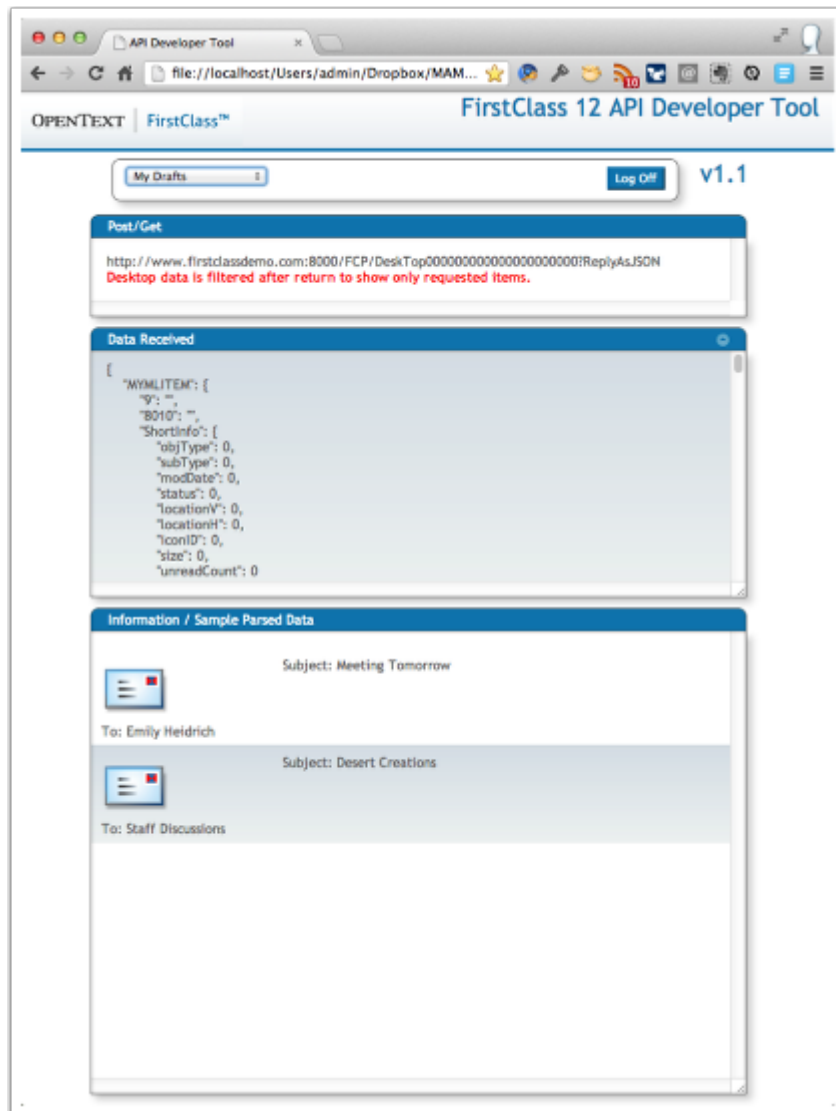
My Watched Items

Similar to other desktop items, to retrieve *My Watched Items*, the desktop is retrieved and the developer must filter the results to display only the watched items.



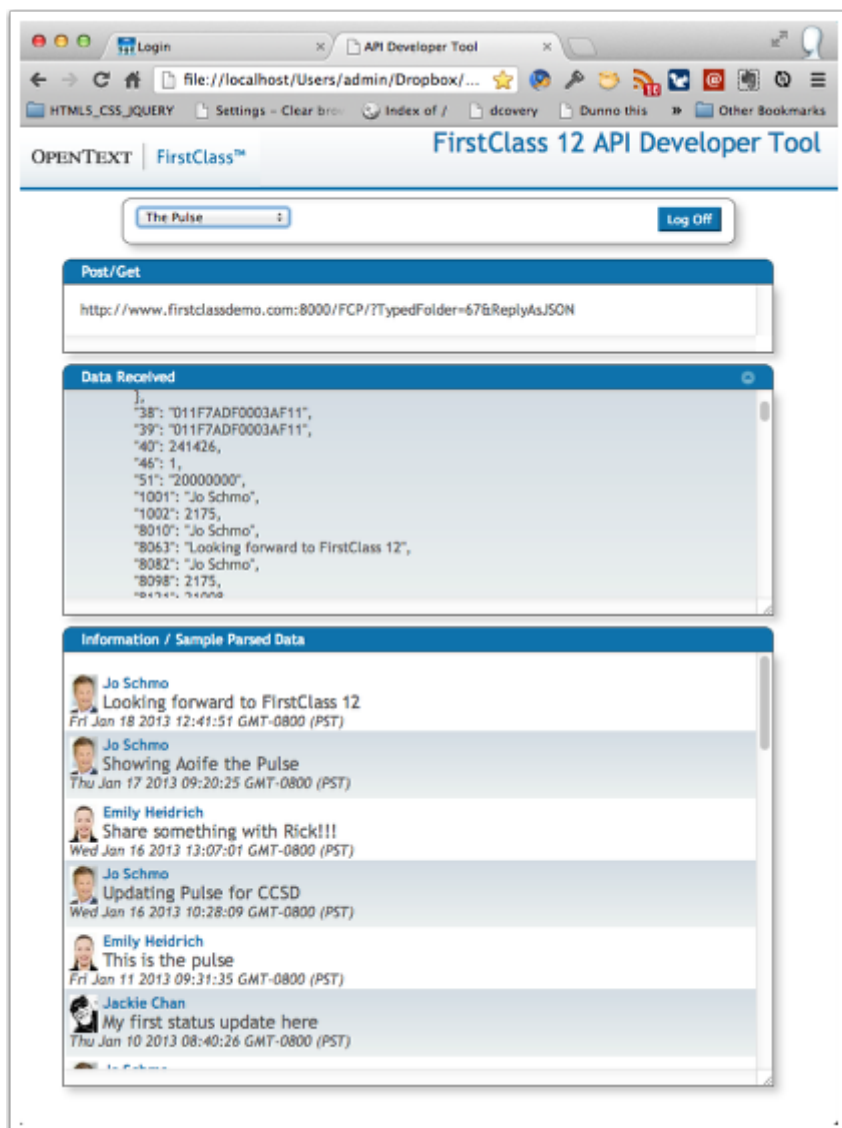
My Drafts

Similar to other desktop items, to retrieve My Drafts, the desktop is retrieved and the developer must filter the results to display only the draft items.



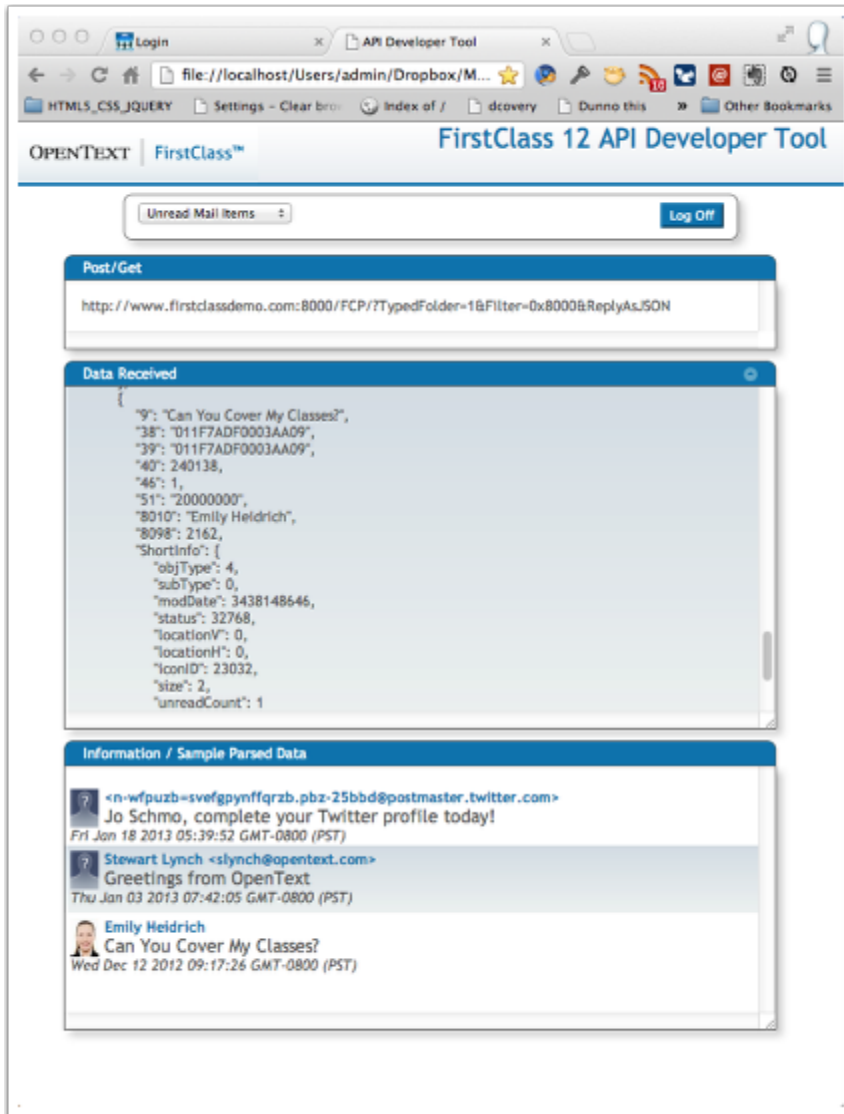
The Pulse

The Pulse returns and displays the current pulse entries for the logged in user.



Unread Mail Items

This Unread Mail Items option will list all of the unread mail items for the currently logged in user.



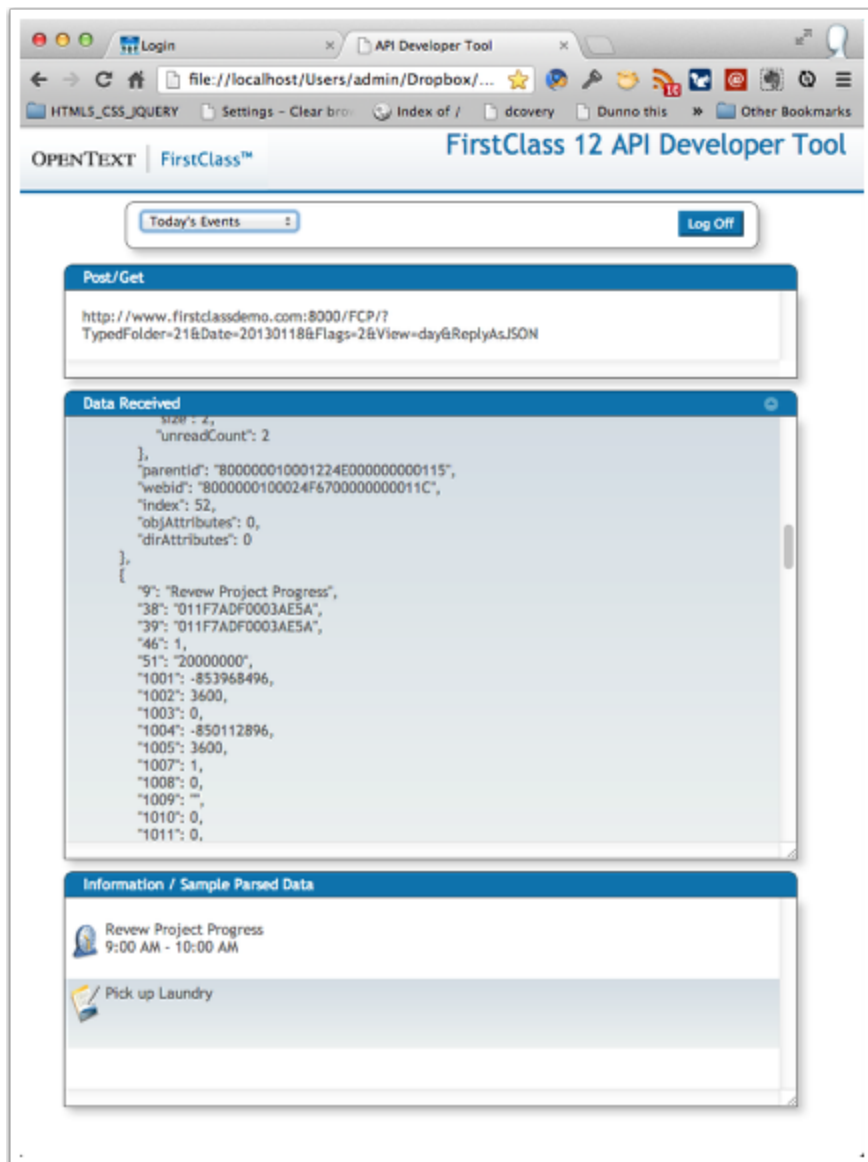
The screenshot shows a web browser window with the 'FirstClass 12 API Developer Tool' interface. The browser's address bar shows a file path: `file:///localhost/Users/admin/Dropbox/M...`. The page has a header with 'OPENTEXT | FirstClass™' and 'FirstClass 12 API Developer Tool'. Below the header, there is a 'Log Off' button and a dropdown menu set to 'Unread Mail Items'. The main content area is divided into three sections:

- Post/Get:** Displays the URL `http://www.firstclassdemo.com:8000/FCP/7TypedFolder=1&Filter=0x8000&ReplyAs=SON`.
- Data Received:** Displays a JSON object representing an email item:


```
{
  "9": "Can You Cover My Classes?",
  "38": "011F7ADF0003AA09",
  "39": "011F7ADF0003AA09",
  "40": 240138,
  "46": 1,
  "51": "20000000",
  "8010": "Emily Heldrich",
  "8098": 2162,
  "ShortInfo": {
    "objType": 4,
    "subType": 0,
    "modDate": 3438148646,
    "status": 32768,
    "locationV": 0,
    "locationH": 0,
    "iconID": 23032,
    "size": 2,
    "unreadCount": 1
  }
}
```
- Information / Sample Parsed Data:** Displays a list of email items with their headers and bodies:
 - From: <n-wfpuzb=svefgpynffqrzb.pbz-25bbd@postmaster.twitter.com>
Subject: Jo Schmo, complete your Twitter profile today!
Date: Fri Jan 18 2013 05:39:52 GMT-0800 (PST)
 - From: Stewart Lynch <slynch@opentext.com>
Subject: Greetings from OpenText
Date: Thu Jan 03 2013 07:42:05 GMT-0800 (PST)
 - From: Emily Heldrich
Subject: Can You Cover My Classes?
Date: Wed Dec 12 2012 09:17:26 GMT-0800 (PST)

Today's Events

Today's Events returns a list of both events and tasks that are scheduled for the current day.



My Résumé

My Résumé returns the user's profile résumé and presents a sample of the retrieved information.

The screenshot shows the 'FirstClass 12 API Developer Tool' interface. At the top, there's a 'Login' button and a 'Log Off' button. Below them is a 'Post/Get' section with a text input field containing the URL: `http://www.firstclassdemo.com:8000/?Resume&ReplyAs.JSON`. Below that is a 'Data Received' section displaying a JSON response. At the bottom is an 'Information / Sample Parsed Data' section showing a profile card for a user named 'jschmo@somewhere.com'.

Post/Get

`http://www.firstclassdemo.com:8000/?Resume&ReplyAs.JSON`

Data Received

```
{
  "clientId": 2175,
  "name": "jschmo@somewhere.com",
  "position": "Director",
  "location": "Vancouver",
  "phone": "123.456.789",
  "mobile": "555.888.9955",
  "email": "jschmo@somewhere.com",
  "bio": "I have been in the business for over 20 years.",
  "languages": "English",
  "education": "BSc",
  "certifications": "BCAMT",
  "skills": "Track and Field",
  "interests": "Skype",
  "social": "jschmo@somewhere.com",
  "status": "cashier",
  "avatar": "http://www.firstclassdemo.com:8000/avatar/jschmo@somewhere.com"
}
```

Information / Sample Parsed Data

	Position: Director
	Location: Vancouver
	Phone: 123.456.789
	Mobile: 555.888.9955
	email: jschmo@somewhere.com

Status Update

The Status Update is an example of a POST to FirstClass that will update the user's status and post to the Pulse. This requires the completion of a field first before submission. A subsequent retrieval of the pulse will show that the entry has been submitted.

The screenshot displays the FirstClass 12 API Developer Tool interface within a web browser. The browser's address bar shows the file path: `file:///localhost/Users/admin/Dropbox/...`. The tool's header includes the OpenText logo, the FirstClass™ logo, and the title "FirstClass 12 API Developer Tool".

The main interface features a "Status Update" section with a dropdown menu set to "Status Update", a text input field containing "Looking forward to FirstClass 12", and a "Log Off" button. Below the input field is a "Post to Pulse" button.

The "Post/Get" section shows the following details:

- URL: `http://www.firstclassdemo.com:8000/?ReplyAsJSON"`
- Data Sent: `JSON=[{"pulsepost":"Looking forward to FirstClass 12"}]`

The "Data Received" section displays the following JSON response:

```
{
  "ERROR": {
    "code": 0
  }
}
```

The "Information / Sample Parsed Data" section shows the message: "No error. Action succeeded."

New Conference

The New Conference option is another example of a POST that requires a name and iconID. This creates the conference on the user's desktop and displays the resulting container and icon in the Information window.

The screenshot shows the 'FirstClass 12 API Developer Tool' interface. At the top, there's a 'New Conference' section with a 'Log Off' button. Below it, there are input fields for 'Conference Name' (containing 'Check it out') and 'Icon ID' (containing '23071'). A 'Create Conference' button is at the bottom of this section.

Below the form, there are three panels:

- Post/Get:** Shows the URL 'http://www.firstclassdemo.com:8000/?ReplyAsJSON' and the data sent: 'Data Sent: JSON={"create":{"webid":"Desktop00000000000000000000","type":1,"name":"Check it out","iconid":23071}}'.
- Data Received:** Displays a JSON response:

```
{
  "MYMLITEM": {
    "g": "",
    "39": "8000000100025299",
    "8010": "Check it out",
    "8098": 2175,
    "ShortInfo": {
      "objType": 1,
      "subType": 0,
      "modDate": 3441357864,
      "status": 0,
      "locationV": 0
    }
  }
}
```
- Information / Sample Parsed Data:** Shows a message 'Conference created on Desktop' with a green checkmark icon and the text 'Check it out'.

Change Password

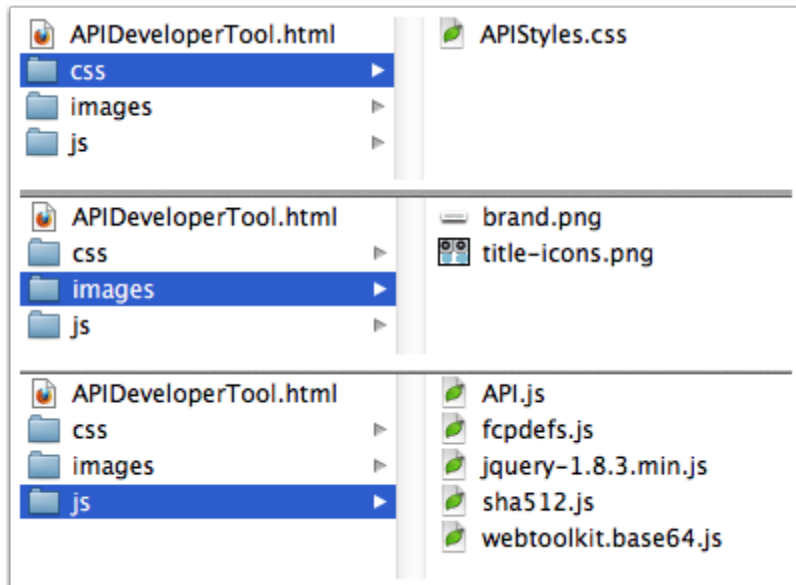
The final example, Change Password is another POST example that requires the old and new password to be submitted (POSTED) to the server. An error code of 0 indicates that the action was successful.

The screenshot displays the FirstClass 12 API Developer Tool interface. At the top, there's a navigation bar with 'OPENTEXT | FirstClass™' and 'FirstClass 12 API Developer Tool'. Below this, a 'Change Password' form is visible, featuring input fields for 'Old Password' (containing 'oldpassword') and 'New Password' (containing 'newpassword'), along with a 'Log Off' button and a 'Change Password' button. The tool shows the following details:

- Post/Get:** The URL is `http://www.firstclassdemo.com:8000/?ReplyAsJSON`. The data sent is a JSON object: `{ "changepassword": { "oldpass": "oldpassword", "newpass": "newpassword" } }`.
- Data Received:** The response is a JSON object: `{ "ERROR": { "code": 0 } }`.
- Information / Sample Parsed Data:** The message states: **No error. Action succeeded.**

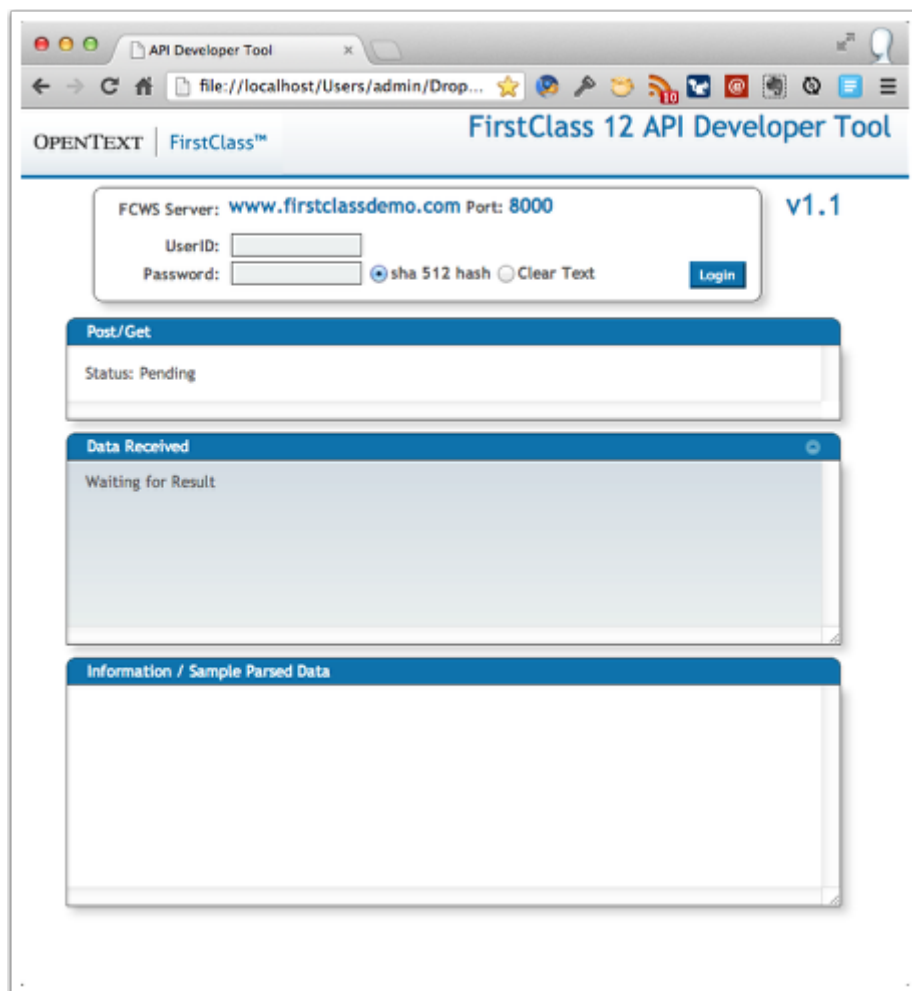
Development Tool Files

The development tool consists of a `APIDeveloperTool.html` file along with the `API.js` javascript file that is the core to understanding how the application functions. There is also an `images` folder and a supporting CSS file for the presentation of the parsed data along with some javascript libraries used by `API.js` to help interpret and present the returned data.



APIDeveloperTool.html

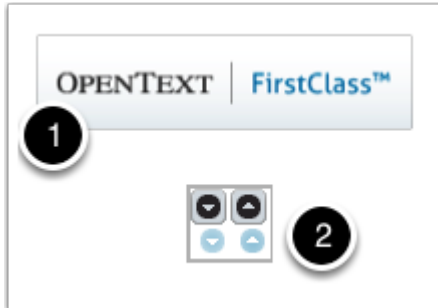
This is the presentation layer that presents the tool to the user in the web browser.



images folder

Consists of:

1. The top logo for the html page
2. A 4 image sprite that is used for the Data Received window collapse/expand button



CSS File

This is used for both styling and layout. Selectors are arranged by category to assist the developer in following through the code.

```
/* Login and option fields
=====*/

#LI_fieldset {
  font-weight: bold;
  margin-left: auto;
  margin-right: auto;
  width: 525px;
  border: 1px solid #555555;
  -webkit-border-radius: 8px;
  -moz-border-radius: 8px;
  border-radius: 8px;
  -moz-box-shadow: 5px 5px 10px #cccccc;
  -webkit-box-shadow: 5px 5px 10px #cccccc;
  box-shadow: 5px 5px 10px #cccccc; }

.LoginEntry {
  float: left; }

.LI_input, .option_input {
  font-family: "Trebuchet MS", Helvetica, sans-serif;
  font-size: 10pt;
  color: #000;
  background: #ECF0F0;
  border: 1px solid #555555;
  padding-left: 5px; }

#postFields1, #postFields2 {
  margin-bottom: 2px;
  clear: both; }

#postLabel1, #postLabel2 {
  width: 10em; }

#LI_login {
  color: #FFF;
  background: #0073ad;
  border: 2px outset #0073ad;
  float: right; }

#actionButton {
  color: #fff;
  background: #666;
  border: 2px outset #222222;
  margin-left: 13em; }

/* Data fields for content display
=====*/

.dataTitle, #receivedHead {
  border: solid 1px #555555;
  -webkit-border-top-left-radius: 8px;
  -moz-border-top-left-radius: 8px;
  -webkit-border-top-right-radius: 8px;
  -moz-border-top-right-radius: 8px;
  border-top-left-radius: 8px;
  border-top-right-radius: 8px;
  color: white;
  margin-left: -1px;
  padding: 3px 16px;
  width: 600px;
  background-color: #0073ad; }
```

JavaScript files

There are 5 javascript files included with the tool. The main one is the API.js along with 4 javascript libraries as described below.



API.js

This is the core file to understanding how the development tool works. Using this file along with the API documentation, experienced developers should be able to interpret and build their own rich web applications to retrieve from and modify data on a FirstClass server. Every attempt has been made to modularize the code into functions so that the developer can reuse it within their own solutions.

```
function get_or_Post(){
    var dataToGet = $('#getItem').val();
    var GetURL = '';
    var dataSent = '';
    switch(dataToGet) {
        case "Select":
            break;
        case "UpdatePulse":
            GetURL= ServerAddr + "/?ReplyAsJSON";
            dataSent = 'JSON={"pulsepost":"' + $("#postInput1").val() + '"}';
            break;
        case "DesktopContainers":
            GetURL= ServerAddr + "/FCP/Desktop00000000000000000007ReplyAsJSON";
            break;
        case "Pulse":
            GetURL = ServerAddr + "/FCP/?TypedFolder=676ReplyAsJSON";
            break;
        case "UnreadMail":
            GetURL= ServerAddr + "/FCP/?TypedFolder=16Filter=@x00005ReplyAsJSON";
            break;
        case "TodayEvents":
            var d = new Date();
            var y = d.getFullYear().toString();
            var m = (d.getMonth() + 1).toString();
            var dy = d.getDate().toString();
            var today = y + ("0" + m).slice(-2) + ("0" + dy).slice(-2);
            GetURL= ServerAddr + "/FCP/?TypedFolder=216Date=" + today + "&Flags=256View=day&ReplyAsJSON";
            break;
        case "MyResume":
            GetURL= ServerAddr + "?Resume&ReplyAsJSON";
            break;
        case "NewConference":
            GetURL= ServerAddr + "/?ReplyAsJSON";
            dataSent = 'JSON={"create":{"webid":"DeskTop000000000000000000","type":1,"name":"' + S
            break;
        case "ChangePassword":
            GetURL= ServerAddr + "/?ReplyAsJSON";
            dataSent = 'JSON={"changepassword":{"oldpass":"' + ($("#postInput1").val() + '"',"newpass'
            break;
    }
    // if the 'getItem' is in this list, it is a 'Post' action, otherwise it is a 'Get' action
    var postItems = ["NewConference", "ChangePassword", "UpdatePulse"];
    if (postItems.indexOf(dataToGet) >= 0) {
        // This is a 'Post' so it requires data to send
        $('#SendData').html(ServerAddr + '/?ReplyAsJSON<br />Data Sent: ' + dataSent);
        createObject(GetURL,dataSent,dataToGet);
    }
    else {
        // This is a 'Get'
        $('#SendData').html(GetURL);
        getObject(GetURL,dataToGet);
    }
}
```

Library javascript files

The 4 library files are included and are required as follows:

- `jquery-1.8.3.min.js` - Though the tool does call to the Google jquery CDN to retrieve this library, it is included as a fallback should that not be available.
- `sha512.js` and `webtoolkit.base64.js` are used to encrypt the password into a base64 sha512 digest for logging in securely to the server.
- `fcpdefs.js` - a FirstClass library containing a number of functions and global variables that can be useful to the developer. In particular, the date parsing functions are used by the tool.