



Social Enterprise

OneCommunity™ 2.0 Implementation Guide

© Social Strata, Inc.

1904 Third Avenue, Suite 525 • Seattle, WA 98101

Phone 206.283.5999 • Fax 206.299.4773

Document Last Revised: January 2, 2010

v20090513

Brian Lenz, Jonmark Weber, Benjamin Waldher

Social Strata, Inc., OneCommunity™ Implementation Guide ("API")
Version: 20090513
Status: Implemented
Author: Brian Lenz, Jonmark Weber, Benjamin Waldher
Copyright 2010 Social Strata, Inc.
1904 3rd Ave, Suite 525, Seattle, WA 98101
All rights reserved.

NOTICE:

This API is protected by copyright and the information described herein may be protected by one or more U.S. patents, foreign patents, or pending applications. Except as provided under the following license, no part of this API may be reproduced in any form by any means without the prior written authorization of Social Strata, Inc., and its licensors, if any.

By viewing, downloading or otherwise copying this API, you agree that you have read, understood, and will comply with all the terms and conditions set forth herein.

THIS SPECIFICATION IS PROVIDED "AS IS". SOCIAL STRATA, INC., MAKES NO REPRESENTATIONS OR WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT THAT THE CONTENTS OF THE SPECIFICATION ARE SUITABLE FOR ANY PURPOSE OR THAT ANY PRACTICE OR IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER RIGHTS.

THIS SPECIFICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED INTO NEW VERSIONS OF THE SPECIFICATION, IF ANY. SOCIAL STRATA, INC., MAY MAKE IMPROVEMENTS AND/OR CHANGES TO THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS SPECIFICATION AT ANY TIME

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SOCIAL STRATA, INC., OR ITS LICENSORS BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION, LOST REVENUE, PROFITS OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO ANY FURNISHING, PRACTICING, MODIFYING OR ANY USE OF THE SPECIFICATION, EVEN IF SOCIAL STRATA, INC., AND/OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You will indemnify, hold harmless, and defend Social Strata, Inc., and its licensors from any claims arising or resulting from: (i) your use of the API; and/or (ii) any claims that later versions or releases of any API furnished to you are incompatible with the API provided to you under this license.

Use, duplication, or disclosure by the U.S. Government is subject to the restrictions set forth in this license and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii)(Oct 1988), FAR 12.212(a) (1995), FAR 52.227-19 (June 1987), or FAR 52.227-14(ALT III) (June 1987), as applicable.

You may wish to report any ambiguities, inconsistencies, or inaccuracies you may find in connection with your use of the API ("Feedback"). To the extent that you provide Social Strata, Inc., with any Feedback, you hereby: (i) agree that such Feedback is provided on a non-proprietary and non-confidential basis and (ii) grant Social Strata, Inc., a perpetual, non-exclusive, worldwide, fully paid-up, irrevocable license, with the right to sublicense through multiple levels of sublicensees, to incorporate, disclose, and use without limitation the Feedback for any purpose related to the API and future versions, implementations, and test suites thereof.

Table of Contents

Introduction to OneCommunity™	4
About This Document.....	4
Before You Begin.....	4
Implementing the OneCommunity Server API.....	4
Advanced OneCommunity™ 2.0 Server API.....	9
Common Login Questions.....	10
Final Testing.....	10
Appendix A OneCommunity™ Server API.....	11
Appendix B OneCommunity™ Client API.....	14
Appendix C Advanced OneCommunity™ Actions.....	15

Introduction to OneCommunity™

Social Enterprise has the ability to delegate member registration, authentication (and to an extent, authorization to external member management systems), and synchronize member profile data via OneCommunity 2.0.

OneCommunity 2.0 defines a client API (which is built into Social Enterprise) and a server API (which you must implement). All OneCommunity API requests are made via browser HTTP redirects.

The API is based on a 'handshake' model of authentication. A security token is generated and confirmed in order to ensure the authenticity of any login request. Communications between your server and Social Enterprise are direct HTTP redirects and requests. The actual login takes place within your server, meaning that passwords are never part of the OneCommunity transaction.

Successful OneCommunity integrations have been written in JSP, ASP, Perl, ColdFusion and Vignette. A variety of web programming languages can be used to build the integration. Database back ends can range from simple flat files to Enterprise Oracle data stores. The OneCommunity API is platform and database independent, and there are no special browser requirements for the end user.

Engineering resources required for implementation are straightforward. The OneCommunity integration requires someone who knows how to access information in your database and present it through standard web server protocols. Basic web programming skills are assumed. Generally, whoever designed the login can also implement the integration within a few days.

Optional implementation of the advanced features of OneCommunity 2.0 is similarly straightforward and requires no new skills, but will take more time to implement. This requires access to extra profile data and requires formatting of this data to meet the standards of the advanced OneCommunity 2.0 features described in this document.

General support for the process is always available from Social Strata and more advanced consulting packages can be created on request.

About This Document

The main body of this document describes the steps necessary to build a OneCommunity server. The appendices describe each step in greater detail. The appendices also contain lists of all parameters accepted for each ACTION type. Please read each appendix before implementing each ACTION type while programming your server.

Advanced implementation sections are also featured in this document. These are completely optional implementation details. Advanced implementation provides profile synchronization and should not be implemented if only login validation is desired.

Before You Begin

Before you begin, be aware of the "bypass login" link so that you can log in even if OneCommunity is not working. This will keep you from being locked out by an incorrect API implementation.

The bypass login URL looks like this:

<http://community.yoursite.com/admin/login>

Implementing the OneCommunity Server API

Follow these steps in building the API handler:

Create a Skeleton

Create a script or servlet that will look at its query arguments, get the ACTION request, any modifying parameters and their values. All accepted modifying parameters are located in the Appendix.

Here is the list of possible ACTION requests that your script or servlet will need to handle:

```
ACTION=ERROR
ACTION=LOGIN
ACTION=QUICK_LOGIN
ACTION=LOGOUT
ACTION=SECURITY_CHECK
ACTION=REGISTER
ACTION=PROFILE
ACTION=LOST_PASSWORD
```

Note: QUICK_LOGIN and LOST_PASSWORD are optional, see below.

More actions are defined later in the advanced section.

Tips:

- Log details of each request you receive, such as query arguments and referrer, to a file or console. That way you can better see what values are being sent between your code and the Social Enterprise OneCommunity Client.
- Consider special logging of ACTION=ERROR events. These usually indicate implementation problems.
- Have each ACTION return a default value until it gets implemented in a subsequent step.

Tests:

- Call each of these actions from a browser. The URLs will be something like:
http://yourserver/eveOC?ACTION=LOGIN
Can you reach the URL? Is the ACTION parameter being parsed?
- If you are doing this work in a development environment, check that the URL is available from outside your firewall.

Social Enterprise MUST be able to call the URL without special authentication or having to accept or send cookies. Failing to make the integration script available from the web is one of the most common errors made in implementation. Be sure that your firewall is accepting requests from the IP used by your Social Enterprise site.

Handle Each of the Actions

Handling ACTION=ERROR:

In the event of an error during the LOGIN, this ACTION will be called by our server.

Display a generic error message to your users. Log the ERROR_MSG parameter and determine the cause of the problem. Usually, any errors here are a result of an implementation problem and not something the user would have any control over or be able to fix. This is why we recommend a generic error message be presented to the user.

Handling ACTION=LOGIN:

ACTION=LOGIN occurs when a user clicks the login link, or tries to access a page that requires permissions granted only to registered members.

ACTION=LOGIN has three parts. The first part is validating the user and/or logging them in to your system. The second part is creating a security token that is unique for this user and this session.

The third part is redirecting the user's browser back to the OneCommunity server to log in to Social Enterprise.

Validating the User:

When Social Enterprise requires authentication for a user, it will redirect the user to your integration URL with the request ACTION=LOGIN. Here is an example:

http://yourserver.com/yourIntegrationURL?ACTION=LOGIN&ON_COMPLETION_URL=http%3A%2F%2Fcommunity.yoursite.com%2F

Your script should parse this request and present your regular user login page. In advanced implementations, your script can detect that a user already has logged in at your site and proceed directly to the redirect outlined below. We do not recommend attempting this until you have confirmed the basic API steps are all functioning as expected.

Creating a Unique Security Token:

Once the user has been validated you want to assign him/her a security token that is unique, and verifiable. The token can be a random string, a hash of time/date, username/time, or any other combination that can be created to produce a unique string. The security token may be any text you care to use. For highest security, a user could receive a different security token each time they log in (e.g. a session ID). That way, the security token will expire and be useless in the event it gets intercepted.

Redirecting the User's Browser to Social Enterprise OneCommunity.

This document assumes that any login is successful and that you have saved the ON_COMPLETION_URL somewhere in the process. This will enable your login process to return the user to an originating page.

Have your script redirect the user's browser back to the Social Enterprise OneCommunity Client, again using ACTION=LOGIN.

[http://community.yoursite.com/OCClient?ACTION=LOGIN&OC_USER_ID=135792468&SECURITY_TOKEN=34YTvfd\\$7Fd4&ON_SUCCESS_URL=http%3A%2F%2Fcommunity.yoursite.com%2F](http://community.yoursite.com/OCClient?ACTION=LOGIN&OC_USER_ID=135792468&SECURITY_TOKEN=34YTvfd$7Fd4&ON_SUCCESS_URL=http%3A%2F%2Fcommunity.yoursite.com%2F)

Notes:

- Do not omit the OC_USER_ID, ON_SUCCESS_URL or SECURITY_TOKEN parameters.
- Be sure to URL-encode the ON_SUCCESS_URL value.
- The ON_SUCCESS_URL should be the ON_COMPLETION_URL passed to you with the ACTION=LOGIN request. If no value was provided, you can choose which page it makes the most sense to send the user to by default.
- You don't have to use the ON_COMPLETION_URL as the ON_SUCCESS_URL. However it will provide a better user experience if you do use it. For example, when someone tries to access a Social Enterprise feature without logging in first, the ON_COMPLETION_URL will be the page they were trying to access (following successful login). By redirecting to this URL they can complete whatever task they were in the middle of performing.
- You'll also want to decide if you need the AUTO_LOGIN parameter. Details of this parameter are in Appendix A.

You can 'redirect' the browser in one of two ways: either by outputting a standard HTML meta-refresh page containing the redirect URL; or you can redirect the browser directly with a print "location:redirectURL..." The first is better for testing because it allows you to view the redirect URL easily in the HTML source code. The second is much faster but is more difficult to debug.

A very fine tool for tracking and debugging requests is the Firefox extension called LiveHTTPHeaders. This browser is available from mozilla.org. You may also use any packet sniffer to see what requests are passed to the browser.

If you are not properly redirected, check the logging you created for ACTION=ERROR. Sometimes you can examine the URL location in your browser for some error codes. If you wish to see a log of the entire transaction, append &EMAIL_FOR_LOGGING=youremail@address.com to the end of the request in this step. For example:

http://yourserver.com/yourIntegrationURL?ACTION=LOGIN&ON_COMPLETION_URL=http%3A%2F%2Fcommunity.yoursite.com%2F&EMAIL_FOR_LOGGING=youremail@address.com

You will want to substitute in your own address. Be sure to remove this debugging parameter before staging to production.

Handling ACTION=QUICK_LOGIN:

Quick login behaves exactly the same way as login, except the quick login request is a POST request rather than a GET request, and includes the parameters 'login' and 'password'. These parameters will contain the user's login credentials, thus skipping the step of rendering a login page. This allows the user to log in from a form rendered on a community page.

This action is optional, and may be skipped if quick logins are not used in your Social Enterprise site. However, if quick logins are used in your community, this action is required.

Handling ACTION=LOGOUT:

Redirect the user to your normal logout page and perform any actions necessary to log the user out of your system (e.g. deleting cookies). Eve will automatically log the user out of its own system before sending this request (so there is no need to further interact with the OneCommunity client on this transaction).

Handling ACTION=SECURITY_CHECK:

Here is an example of the call-back request that Social Enterprise will make to your OneCommunity Server (not the end user):

http://yourserver.com/yourIntegrationURL?ACTION=SECURITY_CHECK&OC_USER_ID=auser&SECURITY_TOKEN=atokenlikeasessionid&IS_USER_DATA_REQUIRED=Y

You will need to verify the request, then create a response which will provide data to Social Enterprise in the format required by this API.

Important! If you are not implementing the advanced One Community features, a minimal correct response must contain at least the EMAIL parameter and should look like:

EMAIL=test@test.com

If you are implementing the advanced OneCommunity features, respond with a PROFILE_UPDATE which provides a 'version' number for a profile. This number should be updated to a greater number every time a profile is updated. This allows the system to detect if its profile is out of sync with the OneCommunity server. For this, it is appropriate to simply use a timestamp of the last update time of the user's profile. If you are not implementing the advanced OneCommunity features, PROFILE_UPDATE may be omitted.

A full list of parameters accepted by Social Enterprise can be found in Appendix A. This example assumes that your script is automatically passing the 200 OK and associated headers produced from a standard HTTP response.

Be sure that the request response content-type header is sent as text/plain. The data values have to be URL encoded (or else '=', '&', '?', ' ' and '/' characters in user names and other fields will

cause parsing problems). Most programming languages provide methods to perform this encoding (e.g., in Java use the URI Class or in PHP use rawurlencode ["String"]).

Your code will need to ensure the user id matches the security token. The purpose of the token is to confirm that the login request was generated by your own server and not someone spoofing the ACTION=LOGIN request into their browser. If a SECURITY_TOKEN does not match, return either a 500 error header or the SECURITY_CHECK_ERROR parameter instead of valid data.

Tips:

- For initial testing disable the security token test or make it always return true until you are sure the full OneCommunity transaction is working correctly.
- The security token may be any text you care to use. For highest security, a user could receive a different security token each time they log in (e.g., a session id). That way, the security token will expire and be useless in the event it gets intercepted.
- Be sure there are no linefeeds or white space before, inside or after the SECURITY_CHECK string. Accented characters, if any, should be encoded UTF-8.
- The OC_USER_ID is often the database key (e.g., a username or row id). It is something that will let you uniquely identify a user in your system.

Tests:

- Simulate this Social Enterprise call-back request to your server using your own browser.

```
http://yourserver.com/yourIntegrationURL?  
ACTION=SECURITY_CHECK&OC_USER_ID=auser&SECURITY_TOKEN=atokenlikeasessionid  
&IS_USER_DATA_REQUIRED=Y
```

Is your application successfully looking up the OC_USER_ID and returning data? Does your browser return a simple data line as above?

- Try your script with an invalid OC_USER_ID + SECURITY_TOKEN combination. Does the page return a non-200 status code or return a SECURITY_CHECK_ERROR parameter?

Handling ACTION=REGISTER:

Redirect to your normal new user registration page.

When the user has finished registration they would normally be considered logged in. It's a nice touch to send an ACTION=LOGIN request to Social Enterprise following a successful registration.

Handling ACTION=PROFILE:

Redirect to your normal user profile edit page. You will need to capture your own site's session ID or cookie to identify the user.

Handling ACTION=LOST_PASSWORD:

Redirect to your lost password page.

This action is optional, and may be skipped if quick logins are not used in your Social Enterprise Site. However, if quick logins are used, this action is required.

Advanced OneCommunity™ 2.0 Server API

New ACTION requests

Two new actions must be added to those you have already implemented:

```
ACTION=PULL_PROFILE  
ACTION=PUSH_PROFILE
```

Handling ACTION=PULL_PROFILE:

This action is performed directly by Social Enterprise to request all fields of a user's profile. Because the request is made by Social Enterprise, session and cookie information can not be used to determine a user, so OC_USER_ID is supplied containing the user's ID. This request must contain all information from Appendix D. Extra profile information must also be provided.

Custom profile field display names are used for parameter names. Any unknown or incorrect field names/values will be quietly ignored.

Handling ACTION=PUSH_PROFILE:

This is a POST request made by Social Enterprise whenever user profile information is updated. The request will contain the new profile data for the updated user, intended for the OC Server to accept and update a user's profile. To identify the user, OC_USER_ID will also be provided. Profile parameters will be sent as post parameters.

For security, a private key and a timestamp are applied. The private key is a random string which is shared between the OC Server and Social Enterprise. The control panel has a field for the private key. Before sending PUSH_PROFILE request, Social Enterprise will MD5 sum the post body with a newline ('\n') and the private key appended. This hash will then be added to the post body as "SECURITY_SIGNATURE" and sent to the OC Server. Your OC Server implementation should then remove the "SECURITY_SIGNATURE", append a newline and the private key, perform the hash and verify that the "SECURITY_SIGNATURE" that was sent was correct. The TIMESTAMP field is used as a unix timestamp, and must use UTC time.

Important! It is critical that you implement this security check. Without it, profile data is updatable by any system and could cause you to lose all of your profile data.

Common Login Questions

Q: When users log in to my regular site, do I have to log them into Social Enterprise?

A: No, but logging a user into Social Enterprise at the same time they log into your regular site is the simplest way to ensure the user appears logged in on both systems.

If you did not pass the login information to Social Enterprise, it would appear they were not logged in when they visit (and in fact, they won't be logged in on that domain). If the user then clicks on an Social Enterprise login link, they will be redirected to your OneCommunity Server. Your server should detect the user is already logged in and redirect them back without delay.

If you make your pages viewable by registered users only, then it is possible to delay the ACTION=LOGIN redirect until the user visits the Social Enterprise site without making the user interface inconsistent. Social Enterprise will see they are not logged in and then direct them to your server. Your server will then direct them back to Social Enterprise, which will verify the security token with your server.

From the user's perspective, these transactions should provide only a moment's delay, depending on server and network performance at both ends.

Q: When a user logs out of my regular site, do I have to log them out of Social Enterprise also?

A: No. But the user remains logged into Social Enterprise until their session expires, which occurs when they close their browser or after 30 minutes of inactivity.

Final Testing

Everything should be implemented now. Did you create your bypass URL? If you have not, do it now. If your OneCommunity server has faults, and you haven't created a bypass URL, the next steps could lock you out of your site.

Once you have your bypass URL, enable OneCommunity for your site and take your implementation for a test drive.

Appendix A OneCommunity™ Server API

Request ACTION=LOGIN:

Begins when the Social Enterprise OneCommunity Client makes an ACTION=LOGIN request (to the OneCommunity Server). For this request the OneCommunity Server is responsible for authenticating the user before performing the subsequent redirect to the OneCommunity Client ACTION=LOGIN.

The OneCommunity Server is able to ‘automatically’ authenticate users from a server credentials cookie, but if the Social Enterprise OneCommunity Client has supplied the optional AUTO_LOGIN=N then the OneCommunity Server should force users to re-authenticate (re-enter their credentials).

Notes: When rendering a OneCommunity Server login page;

- The optional AUTO_LOGIN parameter reflects the user’s preference regarding automatic logins. If the site doesn’t allow auto logins, then AUTO_LOGIN should always equal ‘N’. The user-specific AUTO_LOGIN parameter value will reflect the value supplied for AUTO_LOGIN when performing the post-authentication Social Enterprise OneCommunity Client ACTION=LOGIN redirect. The value for this parameter is typically gathered through a ‘Remember Me’ checkbox on the login form.
- A user could register instead of logging in by making a Register process available on the ACTION=LOGIN page. Once a user has successfully registered with your OneCommunity Server, the login process then proceeds as normal by performing the subsequent Social Enterprise OneCommunity Client ACTION=LOGIN redirect.

It’s possible that a single OneCommunity Server may provide authentication services for multiple Social Enterprise instances. For this reason, the OneCommunity Client ALWAYS supplies a value for the optional CLIENT_URL parameter. The OneCommunity Server can use this URL for OneCommunity Client requests and to uniquely identify Social Enterprise instances.

Parameter Name	Required	Description
ON_COMPLETION_URL	Optional	Where should the OneCommunity Server redirect following successful authentication? If not supplied then it is up to the OneCommunity Server to determine a default URL. The ON_SUCCESS_URL parameter for the Social Enterprise OneCommunity Client ACTION=LOGIN request should be set to this value.
AUTO_LOGIN	Optional	Does this particular user want to be automatically authenticated? (cookie)
CLIENT_URL	Optional	The URL for Social Enterprise OneCommunity Client requests. Useful if more than one Social Enterprise instance is authenticating with this OneCommunity Server.
EMAIL_FOR_LOGGING	Optional	Emails a log of the entire transaction to the email set as the value of this parameter.

Request ACTION=REGISTER:

Operates essentially the same as the ACTION=LOGIN. Following successful user registration, the OneCommunity Server is expected to redirect the user to the Social Enterprise OneCommunity Client ACTION=LOGIN as per the standard login process.

Parameter Name	Required	Description
ON_COMPLETION_URL	Optional	Where should the OneCommunity Server redirect to following successful registration->authentication? If not supplied then it is up to the OneCommunity Server to determine a default URL. The ON_SUCCESS_URL parameter for the Social Enterprise OneCommunity Client ACTION=LOGIN request should be set to this value.
CLIENT_URL	Optional	The URL for OneCommunity Client requests. Useful if more than one Social Enterprise instance authenticating with this OneCommunity Server.

Request ACTION=PROFILE:

Called by the Social Enterprise OneCommunity Client to allow users to update their OneCommunity Server managed profile information.

Parameter Name	Required	Description
ON_COMPLETION_URL	Optional	Where should the OneCommunity Server redirect to following a profile update? If not supplied then it is up to the OneCommunity Server to determine a default URL.
CLIENT_URL	Optional	The URL for OneCommunity Client requests. Useful if more than one Social Enterprise instance authenticating with this OneCommunity Server.
OC_USER_ID	Optional	The OneCommunity Server ID for the relevant user. If not supplied then it is expected that the server will use the user's session/cookie information.

Request ACTION=LOGOUT:

The Social Enterprise OneCommunity Client has already 'logged out' the user locally prior to requesting this OneCommunity Server action. The OneCommunity Server is then expected to log the user out (including removing credentials, cookies, etc.)

Parameter Name	Required	Description
ON_SUCCESS_URL	Optional	Where should the OneCommunity Server redirect to following successful logout? If not supplied then it is up to the OneCommunity Server to determine a default URL.
CLIENT_URL	Optional	The URL for OneCommunity Client requests. Useful if more than one Social Enterprise instance authenticating with this OneCommunity Server.

Request ACTION=SECURITY_CHECK :

The Social Enterprise OneCommunity Client, as part of handling the ACTION=LOGIN request, will make a request (ACTION=SECURITY_CHECK) to the OneCommunity Server in order to retrieve updated user profile information. This is the only OneCommunity request action that does NOT involve an HTTP redirect. For this request, the OneCommunity Server will indicate success/failure by setting the HTTP result code to 200 (request AOK), non-200 (an error), or providing the SECURITY_CHECK_ERROR parameter with a value set to an error message (an error). Apart from verifying the supplied security token, the OneCommunity Server will also be expected to return updated user profile information (only if the Social Enterprise OneCommunity Client supplies IS_USER_DATA_REQUIRED=Y).

Notes: When returning user profile information;

- Must be returned as a URL query string (e.g. DISPLAY_NAME=foo&IS_EMAIL_VERIFIED=Y)
- The USERNAME parameter is case insensitive and must be unique. If no USERNAME is passed through, the Social Enterprise OneCommunity Client will default this parameter to the OC_USER_ID value. Please ensure both the USERNAME and OC_USER_ID will be unique for a user when stored as lowercase (otherwise you will get a "Username already in use" error for duplicate usernames, or you will override an existing users data with a duplicate user id).
- If a profile field value is not passed through, the original value will be retained. This is useful if you wish to manage some user profile fields from Social Enterprise itself (and not overwrite them when handling the SECURITY_CHECK action). For example, you may choose to not pass through the USER_TITLE, IS_BANNED, BANNED_REASON parameters. Parameters that are passed through will overwrite existing values, including changes created by a user in their Social Enterprise profile (if enabled).
- Response should be returned text/plain.
- There should be nothing else on the page. There should be no HTML headers or footers, no Javascript, or HTML code of any sort on the page.

Parameter Name	Required	Description
SECURITY_TOKEN	Mandatory	The security token generated during the previous OneCommunity Server ACTION=LOGIN request, the OneCommunity Server ensures these match.
OC_USER_ID	Mandatory	The OneCommunity Server user ID value.
SECURITY_CHECK_ERROR	Optional	Allows a helpful error message to be displayed to the end user.

Request ACTION=ERROR:

Will be called by the Social Enterprise OneCommunity Client if an error is encountered while performing a Social Enterprise OneCommunity Client ACTION=LOGIN. This ensures that any errors during authentication 'appear' as OneCommunity Server errors.

Parameter Name	Required	Description
ERROR_MSG	Optional	Error message text.

Appendix B OneCommunity™ Client API

Request ACTION=LOGIN:

This action is called by the Social Enterprise OneCommunity Client when it needs to authenticate the current user. The Social Enterprise OneCommunity Client will then make a request to the OneCommunity Server ACTION=SECURITY_CHECK to validate the login request and request updated profile information.

Parameter Name	Required	Description
ON_SUCCESS_URL	Mandatory	Where should the Social Enterprise OneCommunity Client redirect to following successful local authentication? This value is equal to the parameter supplied as ON_COMPLETE_URL to the OneCommunity Server ACTION=LOGIN.
ON_FAILURE_URL	Optional	An alternate base URL for use when reporting errors via the OneCommunity Server ACTION=ERROR. Defaults to the OneCommunity Server URL.
OC_USER_ID	Mandatory	The OneCommunity Server user ID value. Required during the subsequent OneCommunity Server ACTION=SECURITY_CHECK request.
SECURITY_TOKEN	Mandatory	The OneCommunity Server security token required during the subsequent OneCommunity Server ACTION=SECURITY_CHECK request.
AUTO_LOGIN	Optional	Did the current user indicate that they would like to automatically authenticate with this OneCommunity Server in future via a "Remember me" checkbox on your log in page or other similar facility? If supplied, this value is recorded by the Social Enterprise OneCommunity Client and supplied in future requests to the OneCommunity Server ACTION=LOGIN.

Appendix C Advanced OneCommunity™ Actions

Request ACTION=PULL_PROFILE:

This action is a request for all user profile information, and will supply this parameter:

Parameter Name	Required	Description
OC_USER_ID	Mandatory	The OneCommunity Server user ID value.

The system should respond with, at minimum:

Parameter Name	Description
DISPLAY_NAME	The user's display name.
EMAIL	The user's e-mail.
BIRTHDATE	The user's birthday in the format: YYYYMMDD
GENDER	The user's gender, can be '1', 'm', 'M', or 'Male' for male, for '2', 'f', 'F', 'Female' for female.
COUNTRY	The user's two digit country according to ISO 3166-1-alpha-2
POSTAL_CODE	The user's postal code. Must be valid for user's country
LOCATION	The user's location as entered by them. This is a string.

In addition, any other profile fields will attempt to be matched by Social Enterprise to a custom profile field. Custom profile field parameter names are simply the display name. For example, a profile field parameter could be "What is your favorite color?" including the question mark.

Any unmatchable or invalid data will be quietly ignored by Social Enterprise.

Request ACTION=PUSH_PROFILE:

This action will submit all user profile information, along with:

Parameter Name	Required	Description
OC_USER_ID	Mandatory	The OneCommunity Server user ID value.
SECURITY_SIGNATURE	Mandatory	An MD5 hash of the post body (without the SECURITY_SIGNATURE) with a newline and the private key appended.
TIMESTAMP	Mandatory	The current unix timestamp

This action requires no response.