

smoothwall[®]

The Web You Want

Smoothwall Products

Federated Login Developer's Guide

Smoothwall® Federated Login, Developer's Guide, November 2014

Smoothwall publishes this guide in its present form without any guarantees. This guide replaces any other guides delivered with earlier versions of Federated Login.

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Smoothwall.

For more information, contact: docs@smoothwall.net

© 2001 – 2014 Smoothwall Ltd. All rights reserved.

Trademark notice

Smoothwall and the Smoothwall logo are registered trademarks of Smoothwall Ltd.

Linux is a registered trademark of Linus Torvalds. Snort is a registered trademark of Sourcefire INC.

DansGuardian is a registered trademark of Daniel Barron. Microsoft, Internet Explorer, Window 95, Windows 98, Windows NT, Windows 2000 and Windows XP are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Netscape is a registered trademark of Netscape Communications Corporation in the United States and other countries. Apple and Mac are registered trademarks of Apple Computer Inc. Intel is a registered trademark of Intel Corporation. Core is a trademark of Intel Corporation.

All other products, services, companies, events and publications mentioned in this document, associated documents and in Smoothwall software may be trademarks, registered trademarks or service marks of their respective owners in the UK, US and/or other countries.

Acknowledgements

Smoothwall acknowledges the work, effort and talent of the Smoothwall GPL development team:

Lawrence Manning and Gordon Allan, William Anderson, Jan Erik Askildt, Daniel Barron, Emma Bickley, Imran Chaudhry, Alex Collins, Dan Cuthbert, Bob Dunlop, Moira Dunne, Nigel Fenton, Mathew Frank, Dan Goscomb, Pete Guyan, Nick Haddock, Alan Hourihane, Martin Houston, Steve Hughes, Eric S.

Johansson, Stephen L. Jones, Toni Kuokkanen, Luc Larochelle, Osmar Lioi, Richard Morrell, Piere-Yves Paulus, John Payne, Martin Pot, Stanford T. Prescott, Ralf Quint, Guy Reynolds, Kieran Reynolds, Paul Richards, Chris Ross, Scott Sanders, Emil Schweickerdt, Paul Tansom, Darren Taylor, Hilton Travis, Jez Tucker, Bill Ward, Rebecca Ward, Lucien Wells, Adam Wilkinson, Simon Wood, Nick Woodruffe, Marc Wormgoor.

Federated Login contains graphics taken from the Open Icon Library project <http://openiconlibrary.sourceforge.net/>

Address	Smoothwall Limited 1 John Charles Way Leeds. LS12 6QA United Kingdom
Email	info@smoothwall.net
Web	www.smoothwall.net
Telephone	USA and Canada: 1 800 959 3760 United Kingdom: 0870 1 999 500 All other countries: +44 870 1 999 500
Fax	USA and Canada: 1 888 899 9164 United Kingdom: 0870 1 991 399 All other countries: +44 870 1 991 399

Contents

	About This Guide.....	1
	Audience and Scope	1
	Organization and Use	1
	Conventions.....	2
Chapter 1	Introduction to Federated Login.....	3
	About Federated Login and Smoothwall	3
Chapter 2	Working with Federated Login.....	5
	Prerequisites	5
	Further Information.....	6
	Registering Your Google App Engine Application	6
	Configuring Access to Your Google Application	8
	Creating Your Google App Engine Application	8
	Creating the Google App Engine Directory	8
	Creating the Configuration File	9
	Creating Your Google App Script.....	9
Chapter 3	Deploying Federated Login	11
	Uploading Your Google App Engine Application	11
	Enabling Federated Login	12
	Using More Than One Federated Login Provider	13
	Logging on Using Federated Login	13
	Using External Proxy Clients	14
Appendix A	Google App Engine Files	15
	app.yaml.....	15
	main.py.....	16

About This Guide

This guide explains how to create and deploy a Google App Engine™ application which enables Smoothwall users to use Federated Login to authenticate themselves using their Google credentials.

Audience and Scope

This guide is aimed at administrators deploying network authentication using Federated Login.

This guide assumes the following prerequisite knowledge:

- An overall understanding of the functionality of the Smoothwall System
- Familiarity with using web browsers
- Familiarity with programming languages

Organization and Use

This guide is made up of the following chapters and appendices:

- *Chapter 1, Introduction to Federated Login* on page 3
- *Chapter 2, Working with Federated Login* on page 5
- *Chapter 3, Deploying Federated Login* on page 11
- *Appendix A: Google App Engine Files* on page 15

Conventions

The following typographical conventions are used in this guide:

Item	Convention	Example
Key product terms	Initial Capitals	Unified Threat Management
Cross-references and references to other guides	<i>Italics</i>	Refer to the <i>Unified Threat Management Administration Guide</i>
Filenames and paths	Courier	The <code>portal.xml</code> file
Variables that users replace	Courier Italics	<code>http://<my_ip>/portal</code>

To save paper, this guide is designed for double-sided printing.

1 Introduction to Federated Login

About Federated Login and Smoothwall

Federated Login enables users of the Smoothwall authentication service to authenticate themselves using third party authentication systems, such as a Google application, using their Google user credentials.

To use Federated Login, a program or script is needed to bridge the Smoothwall Federated Login API and the third party authentication system. In this document, we go through a worked example using the Google App Engine™, however, the example is equally applicable to other systems.

Note: Federated Login only provides authentication. It does not provide directory (group) functionality, that is, you only get a username and do not get group mapping. To map users to policies, you could, for example, export and use your local user list.

2 Working with Federated Login

This chapter describes how to create a Google App Engine application, including:

- *Prerequisites* on page 5
- *Registering Your Google App Engine Application* on page 6
- *Configuring Access to Your Google Application* on page 8
- *Creating Your Google App Engine Application* on page 8

Prerequisites

To create a Google application, you must have:

- A Google account
You can find more information here: <https://accounts.google.com/SignUp>
- Downloaded and installed the Google App Engine SDK for Python
For more information, see <https://developers.google.com/appengine/downloads>

The steps entailed are as follows:

- Decide on a unique identifier for your Google application and register it
- Create the application
- Upload the application and set any exceptions

Further Information

At the time of writing, Google™ makes available the following guides and FAQs about working with Google applications:

- Getting Started Guide: <https://developers.google.com/appengine/docs/python/gettingstarted/>
- Developer's Guide: <https://developers.google.com/appengine/docs/>
- FAQs: <https://developers.google.com/appengine/kb/>

Registering Your Google App Engine Application

As your Google app must have a unique identifier, the first step is to decide what you want to call it and then register it.

To register your Google application, do the following:

1. Visit <https://appengine.google.com/start>

Welcome to Google App Engine

Before getting started, you want to learn more about developing and deploying applications. Learn more about Google App Engine by reading the [Getting Started Guide](#), the [FAQ](#), or the [Developer's Guide](#).

Create Application

© 2008 Google | [Terms of Service](#) | [Privacy Policy](#) | [Blog](#) | [Discussion Forums](#) | [Project](#) | [Docs](#)

2. Click **Create Application**.

If you have created Google applications before, go to *step 5*. If this is the first time, you will be prompted to verify your account.

Verify Your Account by SMS

To create applications with Google App Engine, you need a verification code. Select the country and carrier for your mobile phone and enter your mobile phone number. The verification code will be sent to it via SMS. Note you will only need to verify your account once.

Country and Carrier:

Other (Not Listed) ▾

If your country and carrier are not on the list, select Other (Not Listed). [What carriers are supported?](#)

Mobile Number:

Include your [country code](#) and full phone number. eg. +1 650 555 1212

Send

© 2008 Google | [Terms of Service](#) | [Privacy Policy](#) | [Blog](#) | [Discussion Forums](#) | [Project](#) | [Docs](#)

3. Enter your mobile number and click **Send**.

An Authentication Code Has Been Sent to ██████████

Within a few minutes, you should receive a text message on your phone that includes a verification code. When you receive it, enter it below. If you don't receive the text message, [try sending it again](#), or see the [App Engine FAQ](#).

Enter Account Code:

Send

© 2008 Google | [Terms of Service](#) | [Privacy Policy](#) | [Blog](#) | [Discussion Forums](#) | [Project](#) | [Docs](#)

- When you receive the authentication code, enter it and click **Send**.

Google displays the following screen:

Create an Application

You have 10 applications remaining.

Application Identifier:
 .appspot.com

All Google account names and certain offensive or trademarked names may not be used as Application Identifiers. You can map this application to your own domain later. [Learn more](#)

Application Title:

Displayed when users access your application.

Authentication Options (Advanced): [Learn more](#)
Google App Engine provides an API for authenticating your users, including **Google Accounts**, **Google Apps**, and **OpenID**. If you choose to use this feature for some parts of your site, you'll need to specify now what type of users can sign in to your application:

- Open to all Google Accounts users (default)**
If your application uses authentication, anyone with a valid Google Account may sign in.
- Restricted to the following Google Apps domain:**

e.g. foo.com
 If your application uses authentication, **only members of this Google Apps domain may sign in**. If your organization uses Google Apps, use this option to create an application (e.g. an HR tracking tool) that is only accessible to accounts on your Google Apps domain. This option cannot be changed once it has been set.
- (Experimental) Open to all users with an OpenID Provider**
If your application uses authentication, anyone who has an account with an OpenID Provider may sign in.

Terms of Service:

Your Agreement with Google

This License Agreement for Google App Engine (the "**Agreement**") is made and entered into by and between Google Inc., a Delaware corporation, with offices at 1600 Amphitheatre Parkway, Mountain View 94043 ("**Google**") and the business entity agreeing to these terms ("**Customer**"). This Agreement is effective as of the date Customer clicks the "I Accept" button below (the "**Effective Date**"). If you are accepting on behalf of Customer, you represent and warrant that: (i) if you have full legal authority to bind Customer to this Agreement, (ii) you have read and understand this Agreement, and (iii) you agree, on behalf of Customer, to this Agreement. If you do not have the legal authority to bind Customer, please do not click

I accept these terms.

- Enter the following information:

Field	Description
Application Identifier	Enter a unique app ID for your Google app.
Application Title	Enter a name for the app which will be displayed to app users.
Authentication Options	Select Restricted to the following Google Apps domain and enter your domain name.

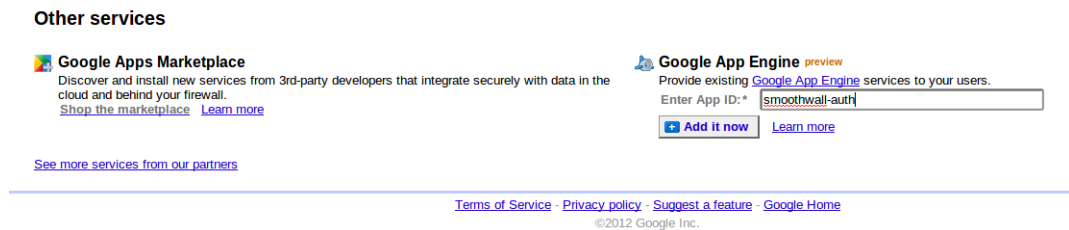
- Click **I accept these terms**, and then click **Create Application**. Google registers your app.

Configuring Access to Your Google Application

In order to restrict access to your Google application to members of a particular Google Apps domain, you must add the application ID to your Google Apps account using the Google Apps control panel.

To configure access to your Google app, do the following:

1. Browse to <https://www.google.com/a/<Your Domain>/SelectServices>



2. Enter your application's unique ID, and click **Add it now**.
3. When prompted, read and accept the Google App Engine terms and conditions, and click **Activate this service**.

Creating Your Google App Engine Application

How you create your Google app and configure its settings file depends on your environment. Microsoft Windows and Apple Mac users can use the graphical user interface provided with their Google App Engine SDK, or use the command line interface. Linux and other platform users can only use the command line interface (CLI).

To cover all users, the following sections explain how to write your application, and configure settings using the CLI.

Note: Programming language Python version 2.7 is required.

Creating the Google App Engine Directory

You must create a directory to store your Google application, and its settings file.

To create a Google App Engine directory, do the following:

- In your `google_appengine` directory, create a `<your_app_id>` directory.

Note: You do not need to use the application's ID as the directory name, or create it in the `google_appengine` directory, but doing so makes it easier to identify and maintain applications.

The next step is to create the your Google application's configuration file.

Creating the Configuration File

Settings for your Google application are stored in the configuration file `app.yaml`, located in the `<your_app_id>` directory. The file specifies runtime configuration, such as, how URL paths are associated with request handles, and static files. It also specifies information about the application code, such as, application ID, and version number.

You can either download an example file from <http://smoothwall.com/en-gb/get-support/product-manuals/federated-login-guide>, or create your own using a text editor. For an example of the `app.yaml` configuration file, see *app.yaml* on page 15.

You must customize the `app.yaml` configuration file as follows:

- Ensure you add the application's unique ID as:
`application: <your_app_id>`
- Ensure you add the application's version number as:
`version: <version>`
- To only allow authorized users to reach the application, add
`login: required`

Tip: Indentations are vital in Python scripts. Confirm that the indentations are exactly as they are in the example above.

Creating Your Google App Script

You must create the authentication script that handles the federated login for the Google App Engine application. The script must be named `main.py`, and be located in the `<your_app_id>` directory. You can either download an example file from <http://smoothwall.com/en-gb/get-support/product-manuals/federated-login-guide>, or create your own using a text editor. For an example of a `main.py` file, see *main.py* on page 16.

The following must be considered:

- A pre-shared key must be set:
`psk=' <pre-shared_key>'`
- Use a GET request to pass through the parameters:
`id, clientid, timestamp, loginurl, targeturl, and hash`
- A check should be made to ensure received data is real, and not modified:
`valdiate_hash`

The md5 hash must consist of `id, clientid, timestamp, loginurl, targeturl, and psk`

- A new md5 hash must be generated to allow the Smoothwall System to check received data validity

The md5 hash must consist of `id`, `timestamp`, `clientid`, `email`, `userdata`, `timesetamp`, and `psk`

- Encode the parameters to send:

`encode_url`

- Send parameters as:

`u` = user's email address

`o` = target URL

`d` = user data

`f` = new Federated Login hash

3 Deploying Federated Login

This chapter describes how to enable Federated Login, including:

- *Uploading Your Google App Engine Application* on page 11
- *Enabling Federated Login* on page 12
- *Logging on Using Federated Login* on page 13
- *Using External Proxy Clients* on page 14

Uploading Your Google App Engine Application

You must upload your Google App Engine application, and add any exceptions needed.

Note: You may need to add a new or change existing exceptions for `accounts.google.???`, where `???` is determined by the country you are in; for example, in the UK, `accounts.google.co.uk`.

Two factor authentication will require you to create an application-specific password for the app temporarily to upload the files to Google Apps. Alternatively, you can use `--oauth2`.

When you deploy the application to Google, use an unfiltered connection. Do not go through Guardian.

To upload your Google application and add exceptions:

1. On the command line, use: `appcfg.py update <your_app_id>_directory`
2. If you are behind a proxy, add exceptions for:
`<your_app_id>.appspot.com`
`google.com`
`accounts.google.com.`

Enabling Federated Login

You must configure your Smoothwall System to use Federated Login.

To enable Federated Login, do the following:

1. Using the command line interface (CLI) of your Smoothwall System, change to:
`/modules/auth/settings/ssllogin/`
2. Using a text editor, create a new file called `federated.json`.
3. Edit `federated.json` as follows:

```
[
  {
    "id" : "google",
    "type" : "basic",
    "url" : "http://<your_app_id>.appspot.com/ourlogin",
    "name" : "Test federated login",
    "psk" : "this is a secret"
  }
]
```

where:

- `"url" : "http://<your_app_id>.appspot.com/ourlogin"` links to where your application is located
- `"name" : "Test federated login"` determines what text is displayed on the SSL login page, for example:



- `"psk" : "this is a secret"` matches what `psk` is in the script. For more information, see *Creating Your Google App Engine Application* on page 8.

Using More Than One Federated Login Provider

It is possible to define more than one Federated Login provider.

To do so, add a list element to your `federated.json` file, with the same data structure for each provider. For example, for two Federated Login providers, add:

```
[
  {
    "id" : "google",
    "type" : "basic",
    "url" : "http://<your_app_id>.appspot.com/ourlogin",
    "name" : "Test federated login",
    "psk" : "this is a secret"
  },
  {
    "id" : "google2",
    "type" : "basic",
    "url" : "http://<another_page>.appspot.com/
ourlogin>",
    "name" : "another federated login",
    "psk" : "this is a secret"
  }
]
```

For three providers, add a third list element, and so on.

Logging on Using Federated Login

When you have created your Google application, and enabled and tested Federated Login, it is ready for you and your users to use.

To log in using Federated Login, do the following:

1. When prompted to log in when trying to access content, click the link as specified in the "name" parameter of the `federated.json` file — see *Enabling Federated Login* on page 12.
2. When prompted, enter your Google credentials to log in.

In log files, usernames are in the form of their email address.

Note: When logging out, users must log out of both the SSL Login page and Google to ensure they are logged out fully. To help users log out from Google, we suggest making this link: <https://www.google.com/accounts/Logout> available to users, for example, as a bookmark.

Using External Proxy Clients

You can allow users to authenticate using Federated Login from outside the network.

To successfully enforce this, you must configure your Smoothwall System to accept the hostname in redirect requests, using a hostname which refers to the external public IP address of the system, and the private IP address if also being used internally.

You do this as follows:

1. Using the command line interface (CLI) of your Smoothwall System, change to:

```
/settings/main/
```

2. Using a text editor, edit the `settings` file.

You may want to create a backup of this file first.

3. Add the following line:

```
USE_HOSTNAME_IN_REDIRECTS=on
```

4. Save and exit your text editor.
5. Reboot your Smoothwall System.

Appendix A: Google App Engine Files

This appendix describes each Google App Engine application file available for download from <http://smoothwall.com/en-gb/get-support/product-manuals/federated-login-guide>, including:

- *app.yaml* on page 15
- *main.py* on page 16

app.yaml

This is the configuration file for your Google App Engine application:

```
#Edit this section#####
application: <your_app_id>
#####
version: 1
runtime: python27
api_version: 1
threadsafe: true

handlers:
- url: /ourlogin
  script: main.app
  login: required
  secure: always
```

main.py

This is the authentication script that handles the federated login for the Google App Engine application:

```
import webapp2
from google.appengine.api import users
import hashlib
import urllib2

#Edit this section####
psk='this is a secret'
#####

class RequestHandler(webapp2.RequestHandler):
    #retrieve params from GET
    def get_params(self):
        params = {'id': self.request.get('id'),
                  'clientid': self.request.get('clientid'),
                  'timestamp': self.request.get('timestamp'),
                  'loginurl': self.request.get('loginurl'),
                  'targeturl': self.request.get('targeturl'),
                  'hash': self.request.get('hash')}
        return params

    def validate_hash(self, params):
        h = hashlib.md5(params['id'] + params['clientid'] +
                        params['timestamp'] + params['loginurl'] +
                        params['targeturl'] + psk).hexdigest()
        if h == params['hash']:
            return True
        else:
            return False

    def encode_url(self, s):
        return urllib2.quote(s.encode("utf8"))

    #generate federated token
    def generate_hash(self, return_params):
        return return_params['id'] + ',' + return_params['timestamp'] + \
            ',' + hashlib.md5(return_params['clientid'] + \
            return_params['email'] + return_params['userdata'] + \
            return_params['timestamp'] + psk).hexdigest()

    #prepare parameters to send back
    def get_return_params(self, params):
        return_params = {'id': params['id'],
                          'email': str(users.get_current_user().email()),
                          'targeturl': params['targeturl'],
                          'userdata': '',
                          'timestamp': params['timestamp'],
                          'clientid': params['clientid']}
        return_params['f'] = self.generate_hash(return_params)
        for i in return_params:
```

```
        return_params[i] = self.encode_url(return_params[i])
    return return_params

#Send confirmation back to Smoothwall
def send_params(self, return_params, goto):
    url = "%s?u=%s&o=%s&d=%s&f=%s" % (goto, return_params['email'],
        return_params['targeturl'], return_params['userdata'],
        return_params['f'])
    self.redirect(str(url))

def get(self):
    params = self.get_params()
    if self.validate_hash(params):
        return_params = self.get_return_params(params)
        self.send_params(return_params, params['loginurl'])
    else:
        #throw error 403 ( Forbidden )
        self.abort(403)

app = webapp2.WSGIApplication([('/ourlogin', RequestHandler), ],
    debug=True)
```


smoothwall[®]

The Web You Want