

Administering FirstClass Web Services

[About FirstClass Web Services](#)

[FirstClass Web Services components](#)

[Configuring the web server](#)

[Configuring security settings](#)

[Configuring limits on activity](#)

[Overriding FCWS configuration with a web browser](#)

[Startup parameters](#)

[Standard groups as they apply to FirstClass Web Services](#)

[Setting the expiry period for peer-registered users](#)

[Administering communities](#)

[Assigning permissions to groups of communities](#)

[Making groups of users members of communities](#)

[Changing the expiry period of community content](#)

[Administering the Pulse](#)

[Publishing community creation in the Pulse](#)

[Allowing liking](#)

[Creating tailored Pulse views](#)

[Applicable FirstClass scripting commands](#)

[Customizing the appearance of FirstClass Web Services](#)

[Custom desktop screens](#)

[Custom FirstClass GO login screens](#)

[How updates affect your customization](#)

[Getting session usage statistics](#)

[Checking FCWS status with a web browser](#)

[If you use another email system](#)

[Hiding personal information for whole groups](#)

[Hiding personal information for individuals](#)

[Giving users access to their external calendars](#)

[Using OAuth2 authentication for external calendars](#)

[Setting FCWS as an OAuth2 ticket granting authority](#)

[Registering external applications](#)

[Authorizing FirstClass access from external applications](#)

[Submitting the access token](#)

[Obtaining a new access token with the refresh token](#)

[Making external applications available to users](#)

[Writing custom applications](#)

[Exposing groups in applications](#)

[Password security in applications](#)

About FirstClass Web Services

FirstClass Web Services provides your users with access to their FirstClass content using either desktop browsers or mobile devices.



Note

If you disable the instant messaging privilege for a group, that group won't see the FirstClass Web Services "Chats" pane.

If you need to know what version of FirstClass Web Services you are currently running, hover over your server name at the top of your FirstClass Web Services desktop.

[Top](#)

FirstClass Web Services components

FirstClass Web Services uses these FirstClass components:

- the server

This is the same core server application that you also administer for classic FirstClass features. It stores users, their profiles, community feeds, documents, wikis, and the state of all objects.

- the FirstClass web server

The FirstClass web server sits between the server and users' browsers. It translates user requests to the FCP protocol and sends them on to the server for processing. Responses from the server are received by the web server in FCP protocol, and the web server translates them into HTML pages and other formats required by the browsers. The web server also updates the browser clients in real time with changes that occur on the server.

[Top](#)

Configuring the web server

To configure the web server (for example, to tell the web server what server(s) to connect to), update the fcws.cfg file in the fcws folder. The fcws folder is located in FirstClass Web Services (Windows: \Program Files, Mac: /Library, Linux: /opt).

This configuration file is a simple text file formatted as case-sensitive keyword=value pairs (like a Windows INI file). It is fully commented with explanations of the keywords.



Note

You will also see a file called fcws_release.cfg. This file contains the default configuration file for the current release. If you have changed fcws.cfg, you can use this file as a reference for shipping defaults, and check for any keywords that you might want to add to your fcws.cfg file.

Configuring security settings

The file default_clean.cfg in ..\fcws\directory contains the default security settings for FirstClass Web Services. Each setting is commented out and includes a commented description. Simply uncommenting a setting here without changing it won't affect anything, because these are all the default settings anyway.

If you want to change any of these settings, rename a copy of this file to a meaningful name (for example, myclean.cfg), change its settings as required, and uncomment the changed settings.

In the fcws.cfg file, add the following line (this line shows the example file name myclean.cfg):

```
SanitizerConfigFile=myclean.cfg
```

Configuring limits on activity

You can set limits on individual thread activity and session activity in the fcws.cfg file. When these limits are reached, warnings are logged.

The parameters that control these limit are:

MaximumThreadActivity=*seconds*

Example:

```
MaximumThreadActivity=120
```

The maximum number of seconds a thread can be active before triggering a warning.

MaximumSessionActivity=*requests*

Example:

```
MaximumSessionActivity=4
```

The maximum number of requests per second that can be issued by a session before triggering a warning.

The number of requests are calculated in the interval between two consecutive watchdog-timer scans (a configurable interval that must be 60 seconds minimum).

This maximum is intended to detect any denial-of-service behavior in a session.

[Top](#)

Overriding FCWS configuration with a web browser

The FirstClass Web Server Configuration Tool allows you to temporarily reconfigure FCWS using a web browser. Configuration is done on the fly, without having to restart the server. This is useful for setting tracing and debugging options.

For details on using this tool, see the document describing [synchronization](#).

[Top](#)

Startup parameters

The following cherrypy command line startup parameters are only necessary if you need to customize your configuration:

```
parser.add_option('-H', '--host', help='Specify the host name for the CherryPy Server.
0.0.0.0 will serve on all interfaces. (default: %default)', default='0.0.0.0', dest='host'
)

parser.add_option('-f', '--fcserverport', help='Specify the FCP port number used to
connect to the FirstClass Server. (default: %default)', type=int, default=510,
dest='fcs_port')

parser.add_option('-s', '--sslport', help='Specify the SSL / HTTPS port number. (default:
%default)', type=int, default=443, dest='ssl_port')

parser.add_option('-k', '--key', help='Specify the SSL key file name or path. (default:
%default)', default='key.pem', dest='ssl_private_key')

parser.add_option('-c', '--cert', help='Specify the SSL cert file name or path.
(default: %default)', default='cert.pem', dest='ssl_certificate')

parser.add_option('-i', '--intermediate', help='Specify the SSL intermediate certificate
file name or path. (default: %default)', default=None, dest='ssl_certificate_chain')

parser.add_option('-t', '--sslthreads', help='Specify the number of threads for the SSL
CherryPy Server. (default: %default)', type=int, default=500, dest='ssl_thread_pool')

parser.add_option('-e', '--enablehttp', action='store_true', help='Enable the HTTP
server. (default: %default)', default=False, dest='http_enabled')

parser.add_option('-p', '--httpport', help='Specify the HTTP port number. (default:
%default)', type=int, default=80, dest='port')

parser.add_option('--httpthreads', help='Specify the number of threads for the HTTP
CherryPy Server. (default: %default)', type=int, default=100, dest='http_thread_pool')

parser.add_option('--sslprotocol', help='Specify cryptographic protocol for the HTTP
CherryPy Server. (default: %default)', default='TLSv1', dest='ssl_protocol')

parser.add_option('--socketquesize', help='Specify CherryPy Server socket queue size.
(default: %default)', type=int, default=5, dest='socket_que_size')

parser.add_option('--sockettimeout', help='Specify CherryPy Server socket timeout.
(default: %default)', type=int, default=10, dest='socket_timeout')

parser.add_option('--configfile', help='Specify custom configuration file name. (default:
%default)', default=None, dest='config_file')
```

- means short parameter name

-- means long parameter name

default means the default value if the parameter is omitted

[Top](#)

Standard groups as they apply to FirstClass Web Services

Group	Description
User groups	
Regular Users	Regular users can create communities and invite people to join those communities (peer invitations). Regular users also have an unrestricted Directory view of all other users on the system and can search for, and locate,

all but Secret communities.

Community Regular Users	Community regular users are like regular users, except that they don't see personal information like a mailbox (they can only post to communities and conferences), Contacts folder, or personal calendar. They also can't: <ul style="list-style-type: none"> • be subadministrators • create mail rules • see private directory objects • generate receipts for messages.
Remote Users	Remote users can't do peer invitations or search for any content other than that which is on their home page.
Peer Registered Users	Peer registered users are external users created through peer invitations to communities. This group is used purely for identification, so that you can locate all users created through the peer registration process.
Pending Invite	This is a temporary group that is added to all users who have been created through the peer invitation process. Once a user logs in, they are automatically removed from this group.
Owner	When a user creates a community, he/she is designated as the owner of that community. You will never see a User Information form with this group in it. It is used to define community permissions.

Container templates

All Communities	All communities created in FirstClass Web Services.
All Profiles	All the profiles created in FirstClass Web Services. This group controls permissions for the Profile container (My Shared Documents for organizations who used FirstClass prior to version 12.0). To be able to comment on others' blogs, users must belong to a group with Contributor permission.
Private Community	All private communities created in FirstClass Web Services.
Public Community	All public communities created in FirstClass Web Services.
Public Reader Community	All public read-only communities created in FirstClass Web Services.
Secret Community	All secret communities created in FirstClass Web Services.

Setting the expiry period for peer-registered users

When users are invited to join a community through the peer invitation process, they are given a limited number of days to connect to the server. If they don't connect during this time, their accounts expire and are deleted automatically.

This time period is controlled by a setting on the Pending Invite user group. The default is seven days. If you want to change this:

- 1 Open the Groups folder on the administrator's Desktop.
- 2 Open the Pending Invite group.
- 3 Type the new expiry period at "Inactivity limit before deletion" on the Limits tab.

[Top](#)

Administering communities

Each community has an owner. This is the person who either created the community or has control of it.

Communities can also have moderators. Moderators are users who have been given permission to approve community content in communities that require approval of member contributions, and to delete community content.

Assigning permissions to groups of communities

There are several types of communities, each associated with permissions that control what users can do in a community of that type, and whether they can access it.

You can assign permissions to a group of communities by making them all the same community type. This is done by means of container templates, which function like user groups for users.



Caution

Don't attempt to adjust the permissions on any of the container templates. Doing so may result in unexpected behavior. Exception: You will need to turn on "Create conferences" if you want users to be able to create subcontainers in communities, and you are updating from a FirstClass version prior to 12.0 (organizations who started using FirstClass with 12.0 have this permission turned on by default).

Container templates, like user groups, are in the Groups folder on the administrator's Desktop.

Container templates can also be used to restrict or limit the Directory view for a group of users.

These are the container templates you can use with FirstClass Web Services:

Template	Description
All Profiles	Controls permissions for all user profiles. The owner of the profile is the controller. By default, regular users are permitted to view all user profiles and to contribute to all users' blogs. Remote users can't do either of those things.
All Communities	Sets the view properties for all communities and applies to all communities.
Community types	
Public Community*	Makes all communities using this template Public communities. All users in this community can view and contribute to the community.
Public Reader Community*	Makes all communities using this template Public Reader communities. All users in this community can view the community but can't contribute to it.
Private Community	Makes all communities using this template Private communities. Only community members can open the community. Nonmembers must request a membership if they wish to view content.
Secret Community	Makes all communities using this template Secret communities. Users who are not part of this community can't find it through searches. They must be invited to join.

* Container templates for public communities must have "Open by name" selected in their permissions in order for users to be able to open them through searches.

Making groups of users members of communities

Administrators can "push out" communities to a group of users so that those users are automatically members and see the communities on their homepages.

Perhaps you have a group of users who are in your Human Resources department. They all have been added to a custom group and you want to make that group a member of a Human Resources-specific community.

This can be done by locating the original community and placing a link to the community on the model Desktop for that group:

- 1 Click List Directory from the administrator's Desktop.
- 2 List and select the community owner.
- 3 Click Desktop to open this user's Desktop.

Leave this Desktop open while you perform the next steps.

- 4 Open the Groups folder on the administrator's Desktop.
- 5 Open the group that you want to make a member of this community.
- 6 Click Model Desktop on the group form.
- 7 Shift-click and drag the community icon from the owner's Desktop to the group's model Desktop.
This creates a link to the community on the group's model Desktop.
- 8 Close both Desktops to initiate the modelling.

Changing the expiry period of community content

Uploaded documents and wikis in communities never expire and must be manually deleted by the owner or moderator of the community, or by the person who uploaded the document or created the wiki.

Community topic posts, however, have an expiry period that is controlled by the All Communities container template. By default, the expiry is set to Never.

If you want the posts in a particular community to expire (and therefore automatically be removed from the community), you can adjust the expiry period at the community level. This will override the setting that is set at the template level.

To adjust the expiry period for an individual community:

- 1 Click List Directory from the administrator's Desktop.
- 2 List and select the community owner.
- 3 Click Desktop to open this user's Desktop.
- 4 Select the community.
- 5 Choose Collaborate > Permissions.
- 6 Click the disclosure triangle on the left to open the Permissions form fully.
- 7 Enter your preferred expiry period at Message expiry (days).

Your change is saved when you close this form.

[Top](#)

Administering the Pulse

Every regular FirstClass Web Services user's homepage contains a Pulse pane. The Pulse lists users' status posts, and comments on those posts, in real time.

By default, a Pulse container is modelled on the Regular Users and Community Regular Users Desktops.

Publishing community creation in the Pulse

In addition to user statuses, the Pulse can include notifications when users create public communities. If you want these notifications in the Pulse, ensure that "Publish creation to Pulse" on the Pulse containers' container template form is selected.

Allowing liking

You can enable liking of Pulse entries the same way it is enabled for conferences and communities.

In the Pulse container's permissions, choose Like to allow users to like something, but be able to change their minds, or Like (Locked) to allow users to like something, but not be able to change their minds.

These are the only two options applicable to the Pulse.

Creating tailored Pulse views

Although normally all FirstClass Web Services users see the same Pulse, you may decide to present focused versions of the Pulse to different user groups.

To give individual groups their own versions of the Pulse:

- 1 Remove the Pulse container from the Regular Users and Community Regular Users Desktops.

Note

You can repurpose this Pulse container for one of your groups.

- 2 Create a conference for each version of the Pulse.

The names you give the Pulse conferences are unimportant; FirstClass Web Services homepages will always display "Pulse".

- 3 Alias the appropriate conference to each group's Model Desktop.



Note

With multiple Pulses, you may wish to put the conferences in the Private Community group.

- 4 Set the subtype in each conference's [properties](#) to Pulse.

[Top](#)

Applicable FirstClass scripting commands

Many of the FirstClass scripting commands aren't valid for FirstClass Web Services. You are advised to restrict the use of scripting to those commands listed below when administering FirstClass Web Services. Using other, unsupported commands can potentially break functionality.

The scripting commands that are available to you for FirstClass Web Services are:

Command	Purpose
ACTIVATE	Activates a superuser account when you want FCWS to act as an OAuth2 ticket granting authority.
ADD	Registers users on your system and places them in user groups.
PGADD	Adds group memberships.
PGDEL	Removes group memberships.
DEL	Deletes a user from your system.
GET	Can be used to retrieve user information.
PUT	Can be used to change user information.
NEW	Creates FirstClass Web Services objects most commonly used to create groups.
LIST	Lists the entire user base, including addresses and user groups.
REPLY	Sends a message back to the administrator when the command is completed.

[Top](#)

Customizing the appearance of FirstClass Web Services

Custom desktop screens

To customize any FirstClass Web Services desktop screen, including the login screen, create a custom HTML template with the same name as the default template. You will find the default templates in `fcws/localized/html` and `fcws/html`.

Place your custom template in `fcws/localized/language/user/html`.

For more information, see the FirstClass Web Services API Guide.

Custom FirstClass GO login screens

You can add the following images to customize the FirstClass GO mobile client login screen:

- a tiled background image

- a top banner image
- a bottom banner image.

Create a folder called "custom" in fcws/images, and place your custom images in that folder.



Note

Ensure that the string `fc-Login-form` is somewhere in your custom login screen's HTML. It is both the ID and the class of the div containing the login form in the default login template. Android devices look for this string to verify that the login form has been received.

Adding a tiled background

An image intended for tiling over the whole login screen can be any size (down to one pixel for a solid color).

Name this file `mobile_bg.png`.

Adding a top banner

An image intended as a banner at the top of the login screen must be 1024x72 pixels. The image will be anchored to the top left of the screen and won't be scaled regardless of device size or orientation.

Name this file `mobile_top.png`.

Adding a bottom banner

An image intended as a banner under the Login button must be 320x72 pixels. The image will be centred on the screen.

Name this file `mobile_bottom.png`.

How updates affect your customization

When you update FirstClass Web Services, your customized resources are treated differently depending on whether this is a nontransitional update (for example, 12.0.0 to 12.0.1) or a transitional update (for example, 12.0.2 to 12.1.0) or. In both cases, your customization is still available to you.

For a nontransitional update, any of the following resources that are customized are copied to the new version:

- images
- css and themes
- html.

Because the FirstClass Web Services installer can't detect if a sprite has been customized, the new version's sprites are placed in a "new_sprites" folder. The sprites from your current version are restored to the new version, and you can decide which set to use.

For a transitional update, those customized resources are placed in a "localized_*current version*" folder (for example, localized_16.0.0). The new version is installed with no customization. You can copy customized resources to the new version as needed.

As a backup measure, your current version is archived so that you can manually restore customized resources. You'll find this archive in FirstClass Web Services/Pkg Archives/*version*. *Version* is a folder named with the date archived, the archived version, and the platform (for example, 2016-05-30_112632_fcws-16.0.0.020N-win).

[Top](#)

Getting session usage statistics

To get current FirstClass Web Services session usage statistics, log into FirstClass Web Services as admin, then choose Web Statistics from the application menu.

Set the number of times to poll for statistics and the interval between polls, then choose Start. To stop polling before the specified number of polling times, choose Stop.

[Top](#)

Checking FCWS status with a web browser

You can use the FirstClass Web Server Configuration Tool to check FCWS status on a browser, just as you can use it to configure FCWS. Choose the information you want to see at Select an Option.

In addition, you can use the following commands:

Syntax

Sample reply

`http://<fcws server>/ping`

FirstClass Synchronization Services example:

OpenText FC Sync Services 12.1.035 (Windows)

`http://<fcws server>/info`

FirstClass Synchronization Services example:

OpenText FC Sync Services 12.1.035 (Windows)

Supported Commands: Sync,FolderSync,GetItemEstimate,Search,Settings,Ping,Provision

Supported ActiveSync Protocols: 2.5,12.0,12.1,14.0,14.1

WSGI Server Configuration:

CherryPy WSGI Server 192.168.15.75 on SSL port 8443 with certificate cert.pem and key key.pem, running 5 threads.

OpenSSL 1.0.1h release using SSL protocol TLSv1.

CherryPy WSGI Server 192.168.15.75 on HTTP port 8000, running 10 threads

`http://<fcws server>/?config`

FirstClass Server port number: 510

WebSocket connections: ENABLED

Network is secure: HTTPS is not mandatory for security critical operations

Validation key: ENFORCED

Outbound HTML is sanitized: NO

Watch-dog task interval: 60 seconds

Session timeout: 120 minutes

Delayed disconnect timeout: 2 seconds

Logging level: DEBUG

Valid FirstClass Servers:

default : 192.168.15.75

Morpheus : 192.168.124.88

FCOL : fc.firstclass.com

Alternate FirstClass Authentication Servers:

fc.firstclass.com

192.168.15.75

192.168.124.88

External Integration Services Host: 192.168.120.61 on port 7000 (connect timeout 5 seconds)

Thread counting: ON

Warn if thread runs longer than 120 seconds

Warn if session continually issues more than 4 requests per seconds

Supported Languages:

French
English
German
Italian
Danish
Finnish
Japanese
Spanish
Dutch
Norwegian
Swedish

`http://<fcws server>/?config&ReplyAsJSON`

```
{
  "Configuration": {
    "AuthenticationFCServers": "fc.firstclass.com;192.168.15.75;192.168.124.88;",
    "MustUseVKey": "1",
    "MaximumSessionActivity": "4",
    "TraceComponents": "HTTP;HTTP-IN|votovic;EIS-RQ|votovic,barb,ron;",
    "ValidFCServers": "default,192.168.15.75;Morpheus,192.168.124.88;",
    "SanitizeOutbound": "0",
    "cssTheme": "firstclass",
    "MaximumThreadActivity": "120",
    "NetworkIsSecure": "1",
    "SessionTimeout": "120 ",
    "WebSocketEnabled": "1",
    "FCSPort": "510",
    "LoggingLevel": "DEBUG",
    "EISHost": "192.168.120.61",
    "DelayedDisconnectTimeout": "2",
    "SanitizerConfigFile": "None",
    "EISPort": "7000",
    "ThreadCounting": "1",
    "WatchDogInterval": "1",
    "EISConnectTimeout": "5"}
}
```

`http://<fcws server>/?ThreadCount=GET`

System Thread Count

sessions :0
threads :1
ssl_threads :0
thread_pool_size :10
ssl_thread_pool_size:10

**Note**

To turn on thread counting (the number of active HTTP and HTTPS threads in the system), specify `ThreadCounting=1` in the `fcws.cfg` file. When thread counting is on, if the thread count for a thread pool reaches the maximum, a warning message appears in the log. There is either insufficient thread pool capacity or a possible denial-of-service attack.

```
http://<fcws server>/?ThreadCount=GET&ReturnAsJSON
```

```
{"SESSION":{"threadcount":{"users":0,"threads":1,"ssl_threads":0,"thread_pool_size":10,"ssl_thread_pool_size":10}}}
```

```
http://<fcws server>/?EISStatus
```

OpenText FC WebServer 12.1.0.030 (FC WebAPI 1) (Windows)

External Integration Services Host: 192.168.120.153 on port 7000 (connect timeout 10 seconds)

Status: CONNECTED - Data port: 7001

or, depending on EIS state:

OpenText FC WebServer 12.1.0.030 (FC WebAPI 1) (Windows)

External Integration Services is not configured on this web server.

or:

OpenText FC WebServer 12.1.0.030 (FC WebAPI 1) (Windows)

External Integration Services Host: 192.168.120.153 on port 7005 (connect timeout 10 seconds)

Status: NOT CONNECTED

[Top](#)

If you use another email system

If your organization doesn't use FirstClass email (or some of your users use a different email system), you can hide the personal information components of the Web Services client (mailbox, contacts, personal calendars) from users. This information can be hidden for whole groups or on an individual basis.

These are the implications for users who can't see this information:

- they can only connect using a web browser or mobile device
- they don't have a MAILBOX pane
- they can only address messages to conferences and post to communities (no private email)
- they will see a CALENDARS pane in place of the personal CALENDAR pane

This pane is a place for them to combine views of other calendars and to create tasks. They can create events only in public calendars, and can only invite people external to FirstClass to their events. Because these invitations go out as messages, it's possible for recipients to reply. Reply messages are delivered to the first published parent conference of the calendar or, if none is found, to the owner of the calendar.

- they don't have a Contacts folder
- they don't see anyone external to FirstClass when they open permissions forms or membership lists.

These users must set an external forwarding address in their preferences and specify Redirect as the forwarding method. Audit will report all of these users who haven't set a forwarding address. You can use the following FirstClass scripting command to supply a user's forwarding address:

```
PUT PREFERENCES userID 4 0 address 1104 7 1 1111 7 1
```

For example, for a user with userID sbram@otsw.com and an email address that is also sbram@otsw.com, the syntax would be:

PUT PREFERENCES sbram@otsw.com 4 0 sbram@otsw.com 1104 7 1 1111 7 1

Hiding personal information for whole groups

To hide personal information for a group, open their Group Privileges form, then clear the "Personal Information Management (PIM)" checkbox on the Features tab. This checkbox is selected by default.

When you clear this checkbox, you must also select "External notification" on the Features tab.

Hiding personal information for individuals

To hide personal information for an individual, open their user information form, then choose Community Regular User at "Class" on the User Information tab. This user license type hides personal information regardless of the groups to which the user belongs.

[Top](#)

Giving users access to their external calendars

External Integration Services (EIS) gives your users access to their data on external calendars, such as Google calendars. EIS works with any external server that supports CalDAV, integrating third-party content in the FirstClass Web Services client without importing that content to the FirstClass Network Store.

External calendar information isn't synced with FirstClass. It is shown in FirstClass Web Services format, as an option for users to choose when combining calendars. Users can't update external calendar information.

To make an external calendar available to FirstClass Web Services users, you must add the calendar to the External Applications folder on the administrator's Desktop. This adds the calendar to the list of container objects that users can create.

To add a calendar to this folder, choose File > New Document Special > EIS Setup, then fill in the [EIS Setup](#) form. You will be asked to supply the calendar name as your users will see it, a description of it, the URL to it, and the protocol used to connect to it (CalDAV), as well as the groups that can access it.

Users can then open a calendar in FirstClass Web Services and add that calendar using the calendar's New menu. This will add the external calendar to the list of calendars that they can combine. They can also add the calendar to their homepages by creating a container there and choosing the calendar from the New Container popup.

The first time users choose to combine the external calendar, they will be asked for their external calendar login credentials. The FirstClass server will then store those credentials, so that they won't have to log in the next time. If the credentials on the external calendar change, users will once more be asked to log in.

Any permissions limitations for users on the external calendar are honored by FirstClass.

Using OAuth2 authentication for external calendars

You can make the external calendar authenticate logins using OAuth2. To do this:

- 1 Register with the owner of the external calendar app (Google example shown below).
- 2 Complete the EIS Setup form for this external calendar.

Registering with Google

These instructions are provided as an example of how to register with Google. Note that Google may change this procedure at any time.

To register with Google:

- 1 Create a Google account as the FirstClass administrator (so that it can be used by any FirstClass administrator).
- 2 Go to <https://developers.google.com> using this account.
- 3 Go to Google Developers Console.
- 4 Choose Create a project at "Select a project".
- 5 Type a project name (for example, FCWS Google Calendar).

- 6** Click Create.
- 7** Click Use Google APIs.
- 8** Click CalDAV API.
- 9** Click Enable API.
- 10** Click Go to Credentials.
- 11** Ensure CalDAV API is the selected API at "Which API are you using?".
- 12** Choose Web server (e.g. node.js, Tomcat) at "Where will you be calling the API from?".
- 13** Select "User data" at "What data will you be accessing?".
- 14** Click What credentials do I need?.
- 15** Type a name for your OAuth 2.0 client ID.
- 16** Type the URL of your FirstClass web server, followed by /oauth2, at "Authorized redirect URIs" (for example, https://fcws.example.com/oauth2).
- 17** Click Create client ID.
- 18** Supply information you want your users to see when they add a Google calendar to their FirstClass calendar at "Set up the OAuth2 consent screen".
 Leave the email address as is. It won't be displayed to your users. Provide information that will make your users comfortable giving consent when Google presents them with the consent screen. For example, include your organization's name in the product name, your logo (click "More customization options"), and your home page URL.
- 19** Click Download to download your credentials in a JSON file that contains information needed when you complete the EIS Setup form for the Google calendar.
- 20** Click the project name on the API Manager screen to go to the credentials page for this project.
 This page displays the client ID and client secret information that you need when completing the EIS Setup form.

[Top](#)

Setting FCWS as an OAuth2 ticket granting authority

You can set up FCWS to use OAuth2 to authenticate users accessing their FirstClass accounts from external applications. To do this:

- 1** Create a trusted superuser account on your FirstClass server.
 A trusted superuser account has special privileges that allow it to act on behalf of other users requesting access tokens. It obtains an access token and sends it to the requesting user.
- 2** Run the following FirstClass scripting command to activate the user:

```
ACTIVATE USER userID TRUSTED
```
- 3** Log into the FirstClass Web Server Configuration Tool.
- 4** Choose Generate SHA512 Digest.
- 5** Supply the user ID and password of the superuser you created.
- 6** Click Digest.
- 7** Copy the SHA512 digest for this user that is displayed after you click Digest.
- 8** Update the fcws.cfg file with the following information:

```
oAuth2User=user ID of superuser
oAuth2Hash=copied SHA512 digest for this user
MyIP=FCWS server IP address
```
- 9** Add the following line to the [Setup] section of the server's fcs.ini file:

```
TrustedIPs=semicolon-delimited list of trusted IP addresses
```

Registering external applications

Before FCWS can authenticate requests from an external web application, that application must send you a request to be registered. Upon receipt of the request:

- 1 Type the following URL in a web browser to start the FirstClass OAuth2 Server Registration Tool:
`http://FCWS_server_URL/oauth2setup`
- 2 Choose Register Web Application at Select an Option.
- 3 Supply the name of the application and the redirect_uri provided in the registration request, then click Register.

The generated credentials for this application are displayed.

authorization_uri	The location of the initial authorization request.
resource_uri	Where to submit the access token to gain access to the resource.
ticket_granting_uri	Where to send a request for and obtain an access token.
client_id	This must be submitted when requesting an access token.
client_secret	This must be submitted when requesting an access token.



Note

You can unregister an external application by choosing Unregister Web Application at Choose an Option.

Authorizing FirstClass access from external applications

An authorization request is an HTTPS GET request for an authorization code sent to the authorization_uri along with the resource_uri, client_id and response_type=code as URL query parameters. Here is an example of an authorization request:

```
https://fcws.firstclass.com/oauth2/authorize?response_type=code&resource_uri=https%3A%2F%2Ffcws%2Ffirstclass%2Fcom&client_id=kjdshfh57ty8ghodig485godfi
```

Optionally, a query parameter `stat=arbitrary data` may be sent, and it will be echoed in the reply.

After the resource_uri and client_id are verified, the client is prompted with the FirstClass Login screen.

Upon successful login, the client is again prompted for access permissions. The access permissions prompt has three options:

- Cancel - access to the FirstClass account isn't granted; abort authentication
- Once - the access token is granted for one time only, and no refresh token is issued for subsequent logins
- Allow - full access is granted, with an access token and refresh token.

Both Once and Allow cause the FirstClass server to redirect the client to its redirect_uri for code-token exchange. Example:

```
HTTP/1.1 302 Found
```

Location:

```
https://example.app.com/oauth?code=JHBYCTRCYUYB876GYTY&state=xyz
```

The external web server can now issue a request for an access token:

```
POST /oauth2/token HTTP/1.1
```

```
Content-Type: application/x-www-form-urlencoded
```

```
code=JHBYCTRCYUYB876GYTY&
```

```
client_id=kjdshfh57ty8ghodig485godfi&
```

```
client_secret=fdlkgjwo85up49684hgpwiuth2&
```

```
redirect_uri=https%3A%2F%2Fexample%2Fapp%2Fcom/oauth&
```

```
grant_type=authorization_code
```

A successful response is JSON:

```
{
  "access_token":"fFAGRNJru1FTz70BzhT3Zg",
  "expires_in":120,
  "token_type":"Bearer"
  "refresh_token":"KHKUYguyfgdwouy56854gg"
}
```

Submitting the access token

There are two ways to access a FirstClass account with the access token:

- submit the token in an HTTP GET as a query parameter:

```
GET https://fcws.firstclass.com?access_token=fFAGRNJru1FTz70BzhT3Zg
```

- submit the token in an HTTP GET as header parameter authorization:

```
GET https://fcws.firstclass.com/oauth2/token
```

```
Authorization: Bearer 1/fFAGRNJru1FTz70BzhT3Zg
```

Obtaining a new access token with the refresh token

To obtain a new access token, the external application must issue HTTP POST:

```
POST /oauth2/token HTTP/1.1
```

```
Content-Type: application/x-www-form-urlencoded
```

```
grant_type=refresh_token&
refresh_token=KHKUYguyfgdwouy56854gg
client_id=kjdshfh57ty8ghodig485godfi&
client_secret=fdlkgjwo85up49684hgpwiuth2&
redirect_uri=https%3A%2F%2Fexample%2Fapp%2Fcom/oauth
```

The response is JSON:

```
HTTP/1.1 200 OK
```

```
{
  "access_token":"JHJHJHJNJru1FTz70BzhT3Zg",
  "expires_in":120,
  "token_type":"Bearer",
}
```

Note that the request for a new access token with the refresh token requires all client credentials to be submitted along with the refresh token.

[Top](#)

Making external applications available to users

The Apps pane in the Web Services client gives users access to HTML applications external to FirstClass.

To populate this pane, you must add the applications to the External Applications folder on the administrator's Desktop.

To add an application to this folder and specify which groups can access it, choose File > New Document Special > External Application, then fill in the [External Application](#) form.

To make an external application auto open, choose Properties from its menu in FirstClass Web Services, then tick the "Auto open" checkbox.

[Top](#)

Writing custom applications

Exposing groups in applications

If your organization will be writing applications that use FirstClass data, you can tag groups that you want exposed to Web Services clients. This provides a way to identify a group independently of the group name. Group names can then change, or the application writer can be ignorant of the actual group names, without affecting the application code.

To tag a group for this purpose, supply the tag at "Group external tag" on the Admin tab of the group's privileges form.

Password security in applications

FirstClass clients provide secure connections, with encrypted passwords. If your organization works with our API to customize Web Services, this security isn't guaranteed. (connections that aren't over SSL don't provide password encryption in a meaningful way).

To guard against security issues in this case, the FirstClass server does the following if it detects an attempt to change a password on a nonsecure connection:

- sets a random password for the account
- disables the account
- notifies the administrator both on the console and by sending an email.

A single audit will also disclose disabled accounts.

To reactivate a disabled account, you must:

- change the account password on the user's information form
- use the [ACTIVATE](#) command to activate the account.

[Top](#)