version 1.3.10

Installation and Configuration Guide

Version 1.3.10 – November 2011

Contents

Chapter 1

About this Guide

This guide will walk you through the installation and configuration of the SOGo solution. It also covers the installation and configuration of Funambol – the middleware used to synchronize mobile devices with SOGo.

The instructions are based on version 1.3.10 of SOGo, and version 8.7 of Funambol.

The latest version of this guide is available at
http://www.sogo.nu/downloads/documentation.html.

Chapter 2

Introduction

SOGo is a free and modern scalable groupware server. It offers shared calendars, address books, and emails through your favourite Web browser and by using a native client such as Mozilla Thunderbird and Lightning.

SOGo is standard-compliant. It supports CalDAV, CardDAV, GroupDAV, iMIP and iTIP and reuses existing IMAP, SMTP and database servers - making the solution easy to deploy and interoperable with many applications.

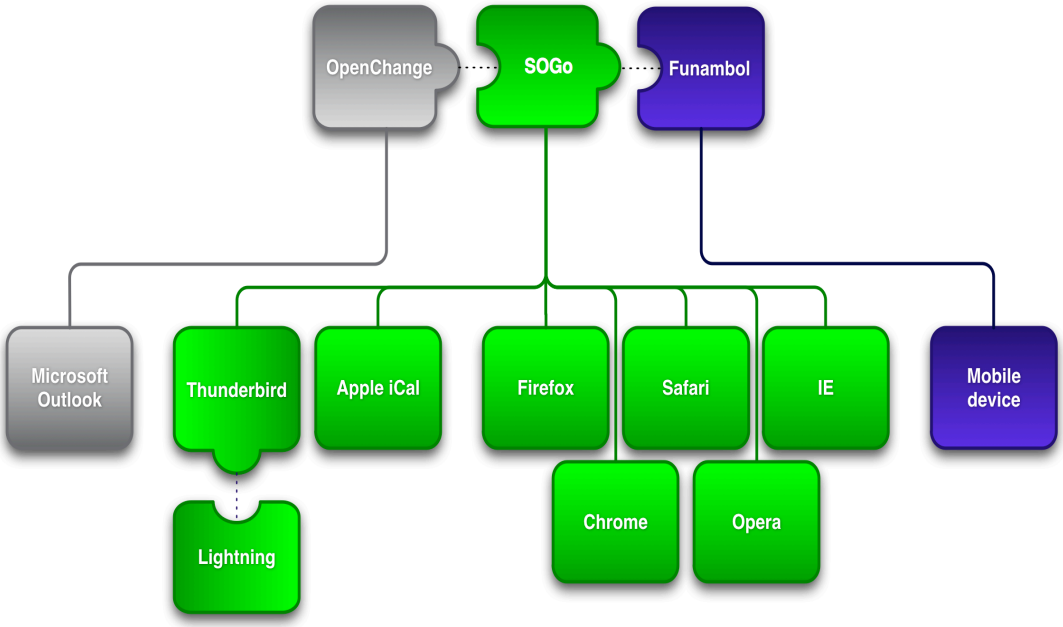
SOGo features :

- Scalable architecture suitable for deployments from dozens to many thousands of users
- Rich Web-based interface that shares the look and feel, the features and the data of Mozilla Thunderbird and Lightning
- Improved integration with Mozilla Thunderbird and Lightning by using the SOGo Connector and the SOGo Integrator
- Two-way synchronization support with any SyncML-capable devices (BlackBerry, Palm, Windows CE, etc.) by using the Funambol SOGo Connector

SOGo is developed by a community of developers located mainly in North America and Europe. More information can be found at http://www.sogo.nu

Architecture

The following diagram demonstrates the SOGo architecture.



Standard protocols such as CalDAV, CardDAV, GroupDAV, HTTP, IMAP and SMTP are used to communicate with the SOGo platform or its sub-components. Mobile devices supporting the SyncML standard use the Funambol middleware to synchronize information.

Chapter 3

System Requirements

Assumptions

SOGo reuses many components in an infrastructure. Thus, it requires the following :

- Database server (MySQL, PostgreSQL or Oracle)
- LDAP server (OpenLDAP, Novell eDirectory, Microsoft Active Directory and others)
- SMTP server (Postfix, Sendmail and others)
- IMAP server (Courier, Cyrus IMAP Server, Dovecot and others)

In this guide, we assume that all those components are running on the same server (i.e., "`localhost`" or "`127.0.0.1`") that SOGo will be installed on.

Good understanding of those underlying components and GNU/Linux is required to install SOGo. If you miss some of those required components, please refer to the appropriate documentation and proceed with the installation and configuration of these requirements before continuing with this guide.

The following table provides recommendations for the required components, together with version numbers :

Database server	PostgreSQL 7.4 or later
LDAP server	OpenLDAP 2.3.x or later
SMTP server	Postfix 2.x
IMAP server	Cyrus IMAP Server 2.3.x or later

More recent versions of the software mentioned above can also be used.

Minimum Hardware Requirements

The following table provides hardware recommendations for the server, desktops and mobile devices :

Server	Evaluation and testing ■ Intel, AMD, or PowerPC CPU 1 GHz ■ 512 MB of RAM (without Funambol, 1 GB RAM otherwise) ■ 1 GB of disk space Production ■ Intel, AMD or PowerPC CPU 3 GHz ■ 2048 MB of RAM ■ 10 GB of disk space (excluding the mail store)
Desktop	General ■ Intel, AMD, or PowerPC CPU 1.5 GHz ■ 1024x768 monitor resolution ■ 512 MB of RAM ■ 128 Kbps or higher network connection Microsoft Windows ■ Microsoft Windows XP SP2 or Vista Apple Mac OS X ■ Apple Mac OS X 10.2 or later Linux ■ Your favourite GNU/Linux distribution
Mobile Device	Any mobile device which supports the SyncML 1.0 or 1.1 standard. Recommended ■ Palm OS based devices with Synthesis SyncML Client ■ Research In Motion (RIM) BlackBerry devices with Funambol client ■ Microsoft Windows Mobile PocketPC or SmartPhone with the Funambol client Apple iPhone / iPod / iPad using Apple iOS 3.0 or later

Operating System Requirements

The following 32-bit and 64-bit operating systems are currently supported by SOGo :

- Red Hat Enterprise Linux (RHEL) Server 5 and 6
- Community ENTerprise Operating System (CentOS) 5 and 6
- Debian GNU/Linux 5.0 (Lenny) and 6.0 (Squeeze)
- Ubuntu 8.10 (Intrepid), Karmic (9.10), Lucid (10.04), Maverick (10.10) and Natty (11.04)

Make sure the required components are started automatically at boot time and that they are running before proceeding with the SOGo configuration. Also make sure that you can install additional packages from your standard distribution. For example, if you are using Red Hat Enterprise Linux 5, you have to be subscribed to the Red Hat Network before continuing with the SOGo software installation.

This document covers the installation of SOGo under RHEL 5.

For installation instructions on Debian and Ubuntu, please refer directly to the SOGo website at http://www.sogo.nu. Under the downloads section, you will find links for installation steps for Debian and Ubuntu.

Note that once the SOGo packages are installed under Debian and Ubuntu, this guide can be followed in order to fully configure SOGo.

Installation

This section will guide you through the installation of SOGo together with its dependencies. The steps described here apply to an RPM-based installation for a Redhat or CentOS distribution.

Software Downloads

SOGo can be installed using the YUM utility. To do so, first create the `/etc/yum.repos.d/inverse.repo` configuration file with the following content :

```
[SOGo]
name=Inverse SOGo Repository
baseurl=http://inverse.ca/downloads/SOGo/RHEL5/$basearch
gpgcheck=0
```

Some of the softwares on which SOGo depends are available from the repository of RPMforge. To add RPMforge to your packages sources, download and install the appropriate RPM package from http://packages.sw.be/rpmforge-release/. Also make sure you enabled the "rpmforge-extras" repository.

For more information on using RPMforge, visit http://rpmrepo.org/RPMforge/Using.

Software Installation

Once the YUM configuration file has been created, you are now ready to install SOGo and its dependencies. To do so, proceed with the following command :

```
yum install sogo
```

This will install SOGo and its dependencies such as GNUstep, the SOPE packages and memcached. Once the base packages are installed, you need to install the proper database connector suitable for your environment. You need to install `sope49-gdl1-postgresql` for the PostgreSQL database system, `sope49-gdl1-mysql` for MySQL or `sope49-gdl1-oracle` for Oracle. The installation command will thus look like this :

```
yum install sope49-gdl1-postgresql
```

Once completed, SOGo will be fully installed on your server. You are now ready to configure it.

Configuration

In this section, you'll learn how to configure SOGo to use your existing LDAP, SMTP and database servers. As previously mentioned, we assume that those components run on the same server on which SOGo is being installed. If this is not the case, please adjust the configuration parameters to reflect those changes.

GNUstep Environment Overview

SOGo makes use of the GNUstep environment. GNUstep is a free software implementation of the OpenStep specification which provides many facilities for building all types of server and desktop applications. Among those facilities, there is a configuration API similar to the "Registry" paradigm in Microsoft Windows. In OpenSTEP, GNUstep and MacOS X, these are called the "user defaults".

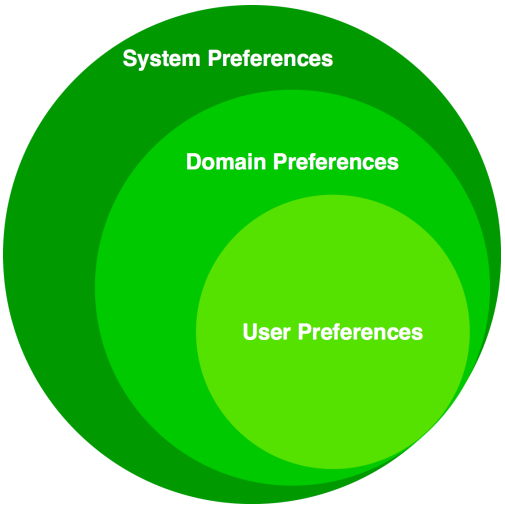
Under GNUstep, a specific file keeps all the user's applications settings and will be edited during our installation. It is located in `/home/sogo/GNUstep/Defaults/.GNUstepDefaults` as the file belongs to the “sogo” user. We will use a command-line tool named `defaults` to edit this configuration database. You can also use your favourite text editor. However, prior to edit this file, it is strongly suggested to make a backup of the file. The GNUstep environment could erase the file if its content is not properly formatted.

The `.GNUstepDefaults` file is a serialized *property list*. This simple format encapsulates four basic data types: arrays, dictionaries (or hashes), strings and numbers. Numbers are represented as-is, except for booleans which can take the unquoted values “YES” and “NO”. Strings are not mandatorily quoted, but doing so will avoid you many problems. A dictionary is a sequence of key and value pairs separated in their middle with a “=” sign. It starts with a “{“ and ends with a corresponding “}”. Each value definition in a dictionary ends with a semicolon. An array is a chain of values starting with “(“ and ending with “)”, where the values are separated with a “,”. Also, the file generally follows a C-style indentation for clarity but this indentation is not required, only recommended.

Each GNUstep application has its own configuration domain. The domain specific to SOGo is named `sogod`. A global domain may also exist, which contains default settings for all applications. This domain is named `NSGlobalDomain` but will not be used here.

Preferences Hierarchy

SOGo supports domain names segregation, meaning that you can separate multiple groups of users within one installation of SOGo. A user associated to a domain is limited to access only the users data from the same domain. Consequently, the configuration parameters of SOGo are defined on three levels:



Each level inherits the preferences of the parent level. Therefore, domain preferences define the defaults values of the user preferences, and the system preferences define the default values of all domains preferences. Both system and domains preferences are defined in the GNUstep user defaults, while the users preferences are configurable by the user and stored in SOGo's database.

To identify the level in which each parameter can be defined, we use the following abbreviations in the tables of this document :

S	Parameter exclusive to the system and not configurable per domain
D	Parameter exclusive to a domain and not configurable per user
U	Parameter configurable by the user

Remember that the hierarchy paradigm allow the default value of a parameter to be defined at a parent level.

General Preferences

Proceed with the following commands to create the basic configuration file for SOGo :

```
su - sogo
defaults write sogod SOGoTimeZone "America/Montreal"
defaults write sogod SOGoMailDomain "acme.com"
defaults write sogod SOGoLanguage English
defaults write sogod SOGoAppointmentSendEMailNotifications YES
defaults write sogod SOGoFoldersSendEMailNotifications YES
defaults write sogod SOGoACLsSendEMailNotifications YES
```

The following table describes the general parameters that can be set :

S	WOWorkersCount	The amount of instances of SOGo that will be spawned to handle multiple requests simultaneously. When started from the init script, that amount is overriden by the "PREFORK" value in `/etc/sysconfig/sogo` or `/etc/default/sogo`. A value of 3 is a reasonable default for low usage. The maximum value depends on the CPU and IO power provided by your machine : a value set too high will actually decrease performances under high load. Defaults to 1 when unset.
S	WOPort	The TCP port used by the SOGo daemon. Defaults to `20000` when unset.
S	WOLogFile	The file path where to log messages. Specify `-` to log to the console. Defaults to `/var/log/sogo/sogo.log`.
S	WOPidFile	The file path where the parent process id will be written. Defaults to `/var/run/sogo/sogo.pid`.
S	SxVMemLimit	Parameter used to set the maximum amount of memory (in megabytes) that a child can use. Reaching that value will force children processes to restart, in order to preserve system memory. Defaults to `384`.
S	SOGoMemcachedHost	Parameter used to set the hostname or IP address of the memcached server. Defaults to `localhost`.
S	SOGoCacheCleanupInterval	Parameter used to set the expiration (in

		seconds) of each object in the cache. Defaults to `300`.
S	SOGoAuthenticationType	Parameter used to define the way by which users will be authenticated. For C.A.S., specify “cas”. For anything else, leave that value empty.
S	SOGoCASServiceURL	When using C.A.S. authentication, this specifies the base url for reaching the C.A.S. service. This will be used by SOGo to deduce the proper login page as well as the other C.A.S. services that SOGo will use.
D	SOGoTimeZone	Parameter used to set a default time zone for users. The default timezone is set to `UTC`. The Olson database is a standard database that takes all the time zones around the world into account and represents them along with their history. On GNU/Linux systems, time zone definition files are available under `/usr/share/zoneinfo`. Listing the available files will give you the name of the available time zones. This could be `America/New_York, Europe/Berlin, Asia/Tokyo` or `Africa/Lubumbashi.` In our example, we set the time zone to `America/Montreal`
D	SOGoMailDomain	Parameter used to set the default domain name used by SOGo. SOGo uses this parameter to build the list of valid email addresses for users. In our example, we set the default domain to `acme.com`
D	SOGoAppointmentSendEMailNotifications	Parameter used to set whether SOGo sends or not email notifications to meeting participants. Possible values are : ■ YES – to send notifications ■ NO – to not send notifications Defaults to NO when unset.
D	SOGoFoldersSendEMailNotifications	Same as above, but the notifications are triggered on the creation of a calendar or an address book.
D	SOGoACLsSendEMailNotifications	Same as above, but the notifications are sent to the involved users of a calendar or address book's ACLs.
D	SOGoCalendarDefaultRoles	Parameter used to define the default roles when giving permissions to a user to access a calendar. Must be an array of up to five strings. Each string defining a role for an

		event category must begin with one of those values: ■ Public ■ Confidential ■ Private And each string must end with one of those values: ■ Viewer ■ DAndTViewer ■ Modifier ■ Responder The array can also contain one or many of the following strings: ■ ObjectCreator ■ ObjectEraser Example: `SOGoCalendarDefaultRoles = ("ObjectCreator", "PublicViewer");` Defaults to no role when unset. Recommended values are `"PublicViewer"` and `“ConfidentialDandTViewer”.`
D	SOGoContactsDefaultRoles	Parameter used to define the default roles when giving permissions to a user to access an address book. Must be an array of one or many of the following strings: ■ ObjectViewer ■ ObjectEditor ■ ObjectCreator ■ ObjectEraser Example: `SOGoContactsDefaultRoles = ("ObjectEditor");` Defaults to no role when unset.
D	SOGoSuperUsernames	Parameter used to set which usernames require administrative privileges over all the users tables. For example, this could be used to post events in the users calendar without requiring the user to configure his/her ACLs. In this case you will need to specify those superuser's usernames like this : `SOGoSuperUsernames = (<username1>[, <username2>, ...]);`
U	SOGoLanguage	Parameter used to set the default language used in the Web interface for SOGo. Possible values are : ■ BrazilianPortuguese ■ Czech ■ Dutch ■ English ■ French ■ German

		■ Hungarian ■ Italian ■ Russian ■ Spanish ■ Swedish ■ Welsh
U	SOGoAppointmentSendEMailReceipts	Parameter used to set whether SOGo sends or not email receipts to the organizer. Possible values are : ■ YES – to send notifications ■ NO – to not send notifications Defaults to NO when unset.
D	SOGoLDAPContactInfoAttribute	Parameter used to specify an LDAP attribute that should be displayed when auto-completing user searches.
D	SOGoiPhoneForceAllDayTransparency	When set to YES, this will force all-day events sent over by iPhone OS based devices to be transparent. This means that the all-day events will not be considered during freebusy lookups. Defaults to NO when unset.
S	SOGoEnablePublicAccess	Parameter used to allow or not your users to share publicly (ie., requiring not authentication) their calendars and address books. Possible values are : ■ YES – to allow them ■ NO – to prevent them from doing so Defaults to NO when unset.
S	SOGoPasswordChangeEnabled	Parameter used to allow or not users to change their passwords from SOGo. Possible values are : ■ YES – to allow them ■ NO – to prevent them from doing so Defaults to NO when unset.
S	SOGoSupportedLanguages	Parameter used to configure which languages are available from SOGo's Web interface. Available languages are specified as an array of string. The default value is : `( "Czech", "Welsh", "English", "Spanish", "French", "German", "Italian", "Hungarian", "Dutch", "BrazilianPortuguese", "Polish", "Russian", Ukrainian", "Swedish" )`

Authentication using LDAP

SOGo can use a LDAP server to authenticate users and, if desired, to provide global address books. SOGo can also use an SQL backend for this purpose (see the section *Authentication using SQL* later in this document). Proceed with the following commands to configure an authentication and global address book using an LDAP directory server :

```
su - sogo
defaults write sogod SOGoUserSources '({CNFieldName = cn;
       IDFieldName = uid; UIDFieldName = uid; IMAPHostFieldName = mailHost;
       baseDN = "ou=users,dc=acme,dc=com";
       bindDN = "uid=sogo,ou=users,dc=acme,dc=com";
       bindPassword = qwerty; canAuthenticate = YES; displayName =
       "Shared Addresses"; hostname = "localhost"; id = public;
       isAddressBook = YES; port=389})'
```

In our example, we use a LDAP server running on the same host where SOGo is being installed.

You can also, using the filter attribute, restrict the results to match various criteria. For example, you could define the following filter to return only *persons* belonging to the organization *Inverse*:

```
    filter = "(objectClass='person' AND o='Inverse' AND active='TRUE')";
```

Since LDAP sources can serve as user repositories for authentication as well as address books, you can specify the following for each source to make them appear in the address book module:

```
   displayName = "<the human identification name of the address book>";
   isAddressBook = YES;
```

For certain LDAP sources, SOGo also supports indirect binds for user authentication. Here is an example :

```
SOGoUserSources =
(
  {
    type = ldap;
    CNFieldName = cn;
    IDFieldName = cn;
    UIDFieldName = sAMAccountName;
    baseDN = "cn=Users,dc=acme,dc=com";
    bindDN = "cn=sogo,cn=Users,dc=acme,dc=com";
    bindFields = (sAMAccountName);
    bindPassword = qwerty;
    canAuthenticate = YES;
    displayName = "Active Directory";
    hostname = 10.0.0.1;
    id = directory;
```

```
    isAddressBook = YES;
    port = 389;
  }
);
```

In this example, SOGo will use an indirect bind by first determining the user DN. That value is found by doing a search on the fields specified in `bindFields`. Most of the time, there will be only one field but it is possible to specify more in the form of an array (for example, `bindFields = (sAMAccountName, cn)`). When using multiple fields, only one of the fields needs to match the login name. In the above example, when a user logs in, the login will be checked against the `sAMAccountName` entry in all the user cards, and once this card is found, the user DN of this card will be used for checking the user's password.

Finally, SOGo supports LDAP-based groups. Groups must be defined like any other authentication sources (ie., `canAuthenticate` must be set to `YES` and a group must have a valid email address). In order for SOGo to determine if a specific LDAP entry is a group, SOGo will look for one of the following objectClass attributes :

- group
- groupOfNames
- groupOfUniqueNames
- posixGroup

You can set ACLs based on group membership and invite a group to a meeting (and the group will be decomposed to its list of members upon save by SOGo). You can also control the visibility of the group from the list of shared address books or during mail autocompletion by setting the `isAddressBook` parameter to `YES` or `NO`. The following LDAP entry shows how a typical group is defined :

```
dn: cn=inverse,ou=groups,dc=inverse,dc=ca
objectClass: groupOfUniqueNames
objectClass: top
objectClass: extensibleObject
uniqueMember: uid=alice,ou=users,dc=inverse,dc=ca
uniqueMember: uid=bernard,ou=users,dc=inverse,dc=ca
uniqueMember: uid=bob,ou=users,dc=inverse,dc=ca
cn: inverse
structuralObjectClass: groupOfUniqueNames
mail: inverse@inverse.ca
```

The corresponding SOGoUserSources entry to handle groups like this one would be :

```
    {
        CNFieldName = cn;
        IDFieldName = cn;
        UIDFieldName = cn;
        baseDN = "ou=groups,dc=inverse,dc=ca”;
        bindDN = "cn=sogo,ou=services,dc=inverse,dc=ca";
        bindPassword = zot;
```

```
    canAuthenticate = YES;
    displayName = “Inverse Groups”;
    hostname = 127.0.0.1;
    id = inverse_groups;
    isAddressBook = YES;
    port = 389;
}
```

The following table describes the possible parameters related to a LDAP source :

D	SOGoUserSources	Parameter used to set the LDAP and/or SQL sources used for authentication and global address books. Multiple sources can be specified as an array of dictionaries. A dictionary that defines an LDAP source can contain the following values:
	type	the type of this user source, set to `ldap` for an LDAP source
	id	the identification name of the LDAP repository. This must be unique – even when using multiple domains.
	CNFieldName	the field that returns the complete name
	IDFieldName	the field that starts a user DN if bindFields is not used. This field must be unique across the entire SOGo domain
	UIDFieldName	the field that returns the login name
	MailFieldNames	an array of fields that returns the user's email addresses (defaults to `mail` when unset)
	SearchFieldNames	an array of fields to to match against the search string when filtering users (defaults to `sn`, `displayName`, and `telephoneNumber` when unset)
	IMAPHostFieldName (optional)	the field that returns the IMAP hostname for the user
	IMAPLoginFieldName (optional)	the field that returns the IMAP login name for the user (defaults to the value of UIDFieldName when unset)
	baseDN	the base DN of your user entries
	KindFieldName (optional)	if set, SOGo will try to determine if the value of the field corresponds to either “group”, “location” or “thing”. If that's the case, SOGo will consider the returned entry to be a resource. For LDAP-based sources, SOGo can also automatically determine if it's a resource if the entry has the “calendarresource” objectClass set.
	MultipleBookingsFieldName (optional)	if set, SOGo will read the value of that field and prevent over-booking resources. If not set, 0 is considered, which means no limit.

filter (optional)	the LDAP filter
scope (optional)	either BASE, ONE or SUB
bindDN	the DN of the login name to use for binding to your server
bindPassword	its password
bindAsCurrentUser	if set to YES, SOGo will always keep binding to the LDAP server using the DN of the currently authenticated user. If bindFields is set, bindDN and bindPassword will still be required to find the proper DN of the user.
bindFields (optional)	an array of fields to use when doing indirect binds
hostname	a space-delimited list of LDAP hostnames or LDAP URLs
port	port number of the LDAP server
encryption (optional)	either SSL or STARTTLS
canAuthenticate	If set to YES, this LDAP source is used for authentication
passwordPolicy	If set to YES, SOGo will use the extended LDAP Password Policies attributes. If you LDAP server does not support those and you activate this feature, every LDAP requests will fail.
isAddressBook	if set to YES, this LDAP source is used as a shared address book (with read-only access)
displayName (optional)	if set as an address book, the human identification name of the LDAP repository

ModulesConstraints (optional)	limits the access of any module through a constraint based on an LDAP attribute; must be a dictionary with keys `Mail`, and/or `Calendar`, for example:
	`ModulesConstraints = {`
	`  Calendar = {`
	`  ou = employees;`
	`  };`
	`};`

The following parameters can be defined along the other keys of each entry of the SOGoUserSources, but can also defined at the domain and/or system levels :

D	SOGoLDAPContactInfoAttribute	Parameter used to specify an attribute that should appear in autocompletion of the web interface.
D	SOGoLDAPQueryLimit	Parameter used to limit the number of returned results from the LDAP server whenever SOGo performs a LDAP query (for example, during addresses completion in a shared address book).

D	SOGoLDAPQueryTimeout	Parameter to define the timeout of LDAP queries. The actual time limit for operations is also bounded by the maximum time that the server is configured to allow. Defaults to 0 (unlimited).

LDAP Attributes Indexing

To ensure proper performance of the SOGo application, the following LDAP attributes must be fully indexed :

- givenName
- cn
- mail
- sn

Please refer to the documentation of the software you use in order to index those attributes.

Authenticating using C.A.S.

SOGo natively supports C.A.S. authentication. For activating C.A.S. authentication you need first to make sure that the `SOGoAuthenticationType` setting is set to “cas” and that the `SOGoCASServiceURL` setting is configured appropriately.

The “tricky” part shows up when using SOGo as a frontend interface to an IMAP server as this imposes constraints needed by the C.A.S. protocol to ensure secure communication between the different services. Failing to take those precautions will prevent users from accessing their mails, while still granting basic authentication to SOGo itself.

The first constraint is that the amount of workers that SOGo uses must be higher than 1 in order to enable the C.A.S. service to perform some validation requests during IMAP authentication. A single worker alone would not, by definition, be able to respond to the C.A.S. requests while treating the user request that required the triggering of those requests. You must therefore configure the `WOWorkersCount` setting appropriately.

The second constraint is that the SOGo service must be accessible and accessed via https. Moreover, the certificate issued by the SOGo server has to be recognized by the C.A.S. service. In the case of a certificate issued by a third-party authority, there should be nothing to worry about. In the case of a self-signed certificate, the certificate must be registered in the trusted keystore of the C.A.S. application. The procedure to achieve this falls out of the scope of this document but can be summarized as importing the certificate in the proper "keystore" using the `keytool` utility and specifying the path for that keystore to the Tomcat instance which provides the C.A.S. service. This is done by tweaking the `javax.net.ssl.trustStore` setting, either in the catalina.properties file or in the command-line parameters.

If any of those constraints is unsatisfied, the webmail interface of SOGo will display an empty email account. Unfortunately, SOGo has no possibility to detect which one is the cause of the problem. The only indicators are log messages that at least pinpoint the symptoms:

"failure to obtain a PGT from the C.A.S. service"

Such an error will show up during authentication of the user to SOGo. It happens when the authentication service has accepted the user authentication ticket but has not returned a "Proxy Granting Ticket".

"a CAS failure occurred during operation...."

This error indicate that an attempt was made to retrieve an authentication ticket for a third-party service. Currently, IMAP is the only such service. Most of the time, this happens as a consequence to the problem described above.

Database Configuration

SOGo requires a relational database system in order to store appointments, tasks and contacts information. It also uses the database system to store personal preferences of SOGo users. In this guide, we assume you use PostgreSQL so commands provided the create the database are related to this application. However, other database servers are supported, such as MySQL and Oracle.

First, make sure that your PostgreSQL server has TCP/IP connections support enabled.

Create the database user and schema using the following commands :

```
su – postgres
createuser --no-superuser --no-createdb –-no-createrole \
      –-encrypted --pwprompt sogo
(specify "sogo" as password)
createdb -O sogo sogo
```

You should then adjust the access rights to the database. To do so, modify the configuration file `/var/lib/pgsql/data/pg_hba.conf` in order to add the following line at the very beginning of the file:

```
host     sogo     sogo      127.0.0.1/32       md5
```

Once added, restart the PostgreSQL database service. Then, modify the SOGo configuration to reflect your database settings :

```
su – sogo
defaults write sogod SOGoProfileURL
      'postgresql://sogo:sogo@localhost:5432/sogo/sogo_user_profile'
defaults write sogod OCSFolderInfoURL
      'postgresql://sogo:sogo@localhost:5432/sogo/sogo_folder_info'
defaults write sogod OCSSessionsFolderURL
      'postgresql://sogo:sogo@localhost:5432/sogo/sogo_sessions_folder'
```

The following table describes the parameters that were set :

D	SOGoProfileURL	Parameter used to set the database URL so that SOGo can retrieve user profiles. For MySQL, set the database URL to something like : `mysql://sogo:sogo@localhost:3306/sogo/sogo_user_profile`
D	OCSFolderInfoURL	Parameter used to set the database URL so that SOGo can retrieve the location of user folders (address books and calendars) For Oracle, set the database URL to something like : `oracle://sogo:sogo@localhost:1526/sogo/sogo_folder_info`
D	OCSSessionsFolderURL	Parameter used to set the database URL so that SOGo can store and retrieve secured user sessions information. For PostgreSQL, the database URL could be set to something like : `postgresql://sogo:sogo@localhost:5432/sogo/sogo_sessions_folder`
D	OCSEMailAlarmsFolderURL	Parameter used to set the database URL for email-based alarms (that can be set on events and tasks). This parameter is relevant only if SOGoEnableEMailAlarms is set to YES. For PostgreSQL, the database URL could be set to something like : `postgresql://sogo:sogo@localhost:5432/sogo/sogo_alarms_folder` See the “EMail reminders” section in this document for more information.

If you're using MySQL, make sure in your `my.cnf` file you have :

```
[mysqld]
...
character_set_server=utf8
character_set_client=utf8

[client]
default-character-set=utf8

[mysql]
default-character-set=utf8
```

and when you create the SOGo database, you correctly specify the charset :

```
create database sogo CHARSET='UTF8';
```

Authentication using SQL

SOGo can use a SQL-based database server for authentication. The configuration is very similar to LDAP-based authentication.

The following table describes all the possible parameters related to a SQL source :

D	SOGoUserSources	Parameter used to set the SQL and/or LDAP sources used for authentication and global address books. Multiple sources can be specified as an array of dictionaries. A dictionary that defines a SQL source can contain the following values :
	type	the type of this user source, set to `sql` for a SQL source
	id	the identification name of the SQL repository. This must be unique – even when using multiple domains.
	viewURL	database URL of the view used by SOGo. The view expects columns to be present. Required columns are : ■ `c_uid` : will be used for authentication – it's a username or username@domain.tld ■ `c_name` : will be used to uniquely identify entries – which can be identical to c_uid ■ `c_password` : password of the user, plain text, md5 or sha encoded ■ `c_cn` : the user's common name ■ `mail` : the user's email address Other columns can exist and will actually be mapped automatically if they have the same name as popular LDAP attributes (such as `givenName`, `sn`, `department`, `title`, `telephoneNumber`, etc.)

userPasswordAlgorithm	the algorithm used for password encryption. Possible values are `none`, `md5` and `sha`.
canAuthenticate	if set to `YES`, this SQL source is used for authentication
isAddressBook	if set to YES, this SQL source is used as a shared address book (with read-only access)
authenticationFilter (optional)	a filter that limits which users can authenticate from this source
displayName (optional)	if set as an address book, the human identification name of the SQL repository
LoginFieldNames (optional)	an array of fields that specifies the column names that contain valid authentication usernames (defaults to `c_uid` when unset)
MailFieldNames (optional)	an array of fields that specifies the column names that hold additional email addresses (beside the `mail` column) for each user
IMAPLoginFieldName (optional)	the field that returns the IMAP login name for the user (defaults to `c_uid` when unset)
KindFieldName (optional)	if set, SOGo will try to determine if the value of the field corresponds to either “group”, “location” or “thing”. If that's the case, SOGo will consider the returned entry to be a resource.
MultipleBookingsFieldName (optional)	if set, SOGo will read the value of that field and prevent over-booking resources. If not set, 0 is considered, which means no limit.

Here is an example of an SQL-based authentication and address book source:

```
SOGoUserSources =
(
  {
    type = sql;
    id = directory;
    viewURL = "postgresql://sogo:sogo@127.0.0.1:5432/sogo/sogo_view";
    canAuthenticate = YES;
    isAddressBook = YES;
    userPasswordAlgorithm = md5;
  }
);
```

Certain database columns must be present in the view/table, such as :

- `c_uid` - will be used for authentication – it's the username or username@domain.tld)
- `c_name` - which can be identical to c_uid – will be used to uniquely identify entries
- `c_password` – password of the user, plain-text, md5 or sha encoded for now

- `c_cn` - the user's common name – such as “John Doe”
- `mail` – the user's mail address

Note that groups are currently not supported for SQL-based authentication sources.

SMTP Server Configuration

SOGo makes use of a SMTP server to send emails from the Web interface, iMIP/iTIP messages and various notifications. Proceed with the following commands to set the parameters for sending mails :

```
su - sogo
defaults write sogod SOGoMailingMechanism smtp
defaults write sogod SOGoSMTPServer localhost
```

The following table describes the parameters that were set :

D	SOGoMailingMechanism	Parameter used to set how SOGo sends mail messages. Possible values are : ▪ `sendmail` – to use the sendmail binary ▪ `smtp` – to use the SMTP protocol
D	SOGoSMTPServer	The DNS name or IP address of the SMTP server used when `SOGoMailingMechanism` is set to `smtp`.
S	WOSendMail	The path of the sendmail binary. Defaults to `/usr/lib/sendmail`.

IMAP Server Configuration

SOGo requires an IMAP server in order to let users consult their email messages, manage their folders and more. Proceed with the following commands to set the parameters for the IMAP server :

```
defaults write sogod SOGoDraftsFolderName Drafts
defaults write sogod SOGoSentFolderName Sent
defaults write sogod SOGoTrashFolderName Trash
defaults write sogod SOGoIMAPServer localhost
```

The following table describes the parameters that were set :

U	SOGoDraftsFolderName	Parameter used to set the IMAP folder name used to store drafts messages. Defaults to “Drafts” when unset.

U	SOGoSentFolderName	Parameter used to set the IMAP folder name used to store sent messages. Defaults to “Sent” when unset.
U	SOGoTrashFolderName	Parameter used to set the IMAP folder name used to store deleted messages. Defaults to “Trash” when unset.
D	SOGoIMAPServer	Parameter used to set the DNS name or IP address of the IMAP server used by SOGo. You can also use SSL or TLS by providing a value using an URL, such as : ■ `imaps://localhost:993` ■ `imaps://localhost:143/?tls=YES`
D	SOGoSieveServer	Parameter used to set the DNS name or IP address of the Sieve (managesieve) server used by SOGo. You must use an URL such as: ■ `sieve://localhost` ■ `sieve://localhost:2000` Note that SSL or TLS are not currently supported.
U	SOGoMailShowSubscribedFoldersOnly	Parameter used to specify if the Web interface should only show subscribed IMAP folders. Possible values are : ■ YES ■ NO Defaults to NO when unset.
D	SOGoIMAPAclStyle	Parameter used to specify which RFC the IMAP server implements with respect to ACLs. Possible values are : ■ `rfc2086` ■ `rfc4314` Defaults to “rfc4314” when unset.
D	SOGoIMAPAclConformsToIMAPExt	Parameter used to specify if the IMAP server implements the Internet Message Access Protocol Extension. Possible values are : ■ YES ■ NO Defaults to NO when unset.
D	SOGoForceIMAPLoginWithEmail	Parameter used to specify if, when logging in to the IMAP server, the primary email address of the user will be used instead of the username. Possible values are : ■ YES ■ NO Defaults to NO when unset.
D	SOGoMailSpoolPath	Parameter used to set the path where temporary email drafts are written. Defaults to `/tmp`.

S	NGImap4ConnectionStringSeparator	Parameter used to set the IMAP mailbox separator. Setting this will also have an impact on the mailbox separator used by Sieve filters. The default separator is "/".
D	NGImap4ConnectionGroupPrefix	Prefix to prepend to names in IMAP ACL transactions, to indicate the name is a group name not a user name. RFC4314 gives examples where group names are prefixed with '$'. Dovecot, for one, follows this scheme, and will, for example, apply permissions for '$admins' to all users in group 'admins' in the absence of specific permissions for the individual user. The default prefix is '$'.

Web Interface Configuration

The following additional parameters only affect the Web interface behaviour of SOGo.

S	SOGoPageTitle	Parameter used to define the Web page title. Defaults to `SOGo` when unset.
U	SOGoLoginModule	Parameter used to specify which module to show after login. Possible values are : ▪ `Calendar` ▪ `Mail` ▪ `Contacts` Defaults to `Calendar` when unset.
S	SOGoFaviconRelativeURL	Parameter used to specify the relative URL of the site favion. When unset, defaults to the file `sogo.ico` under the default web resources directory.
S	SOGoZipPath	Parameter used to specify the path of the zip binary used to archive messages. Defaults to `/usr/bin/zip` when unset.
D	SOGoSoftQuotaRatio	Parameter used to change the quota returned by the IMAP server by multiplying it by the specified ratio. Acts as a soft quota. Example: `0.8`
U	SOGoMailUseOutlookStyleReplies (not currently editable in Web interface)	Parameter used to set if email replies should use Outlook's style. Defaults to NO when unset.
U	SOGoMailListViewColumnsOrder (not currently editable in Web interface)	Parameter used to specify the default order of the columns from the SOGo webmail interface. The parameter is an array, for example :

		`SOGoMailListViewColumnsOrder = (Flagged, Attachment, Priority, From, Subject, Unread, Date, Size);`
D	SOGoVacationEnabled	Parameter used to activate the edition from the preferences window of a vacation message. Requires Sieve script support on the IMAP host. Defaults to NO when unset.
D	SOGoForwardEnabled	Parameter used to activate the edition from the preferences window of a forwarding email address. Requires Sieve script support on the IMAP host. Defaults to NO when unset.
D	SOGoSieveScriptsEnabled	Parameter used to activate the edition from the preferences windows of server-side mail filters. Requires Sieve script support on the IMAP host. Defaults to NO when unset.
D	SOGoMailPollingIntervals	Parameter used to define the mail polling intervals (in minutes) available to the user. The parameter is an array that can contain the following numbers: ▪ `1` ▪ `2` ▪ `5` ▪ `10` ▪ `20` ▪ `30` ▪ `60` Defaults to the list above when unset.
U	SOGoMailMessageCheck	Parameter used to define the mail polling interval at which the IMAP server is queried for new messages. Possible values are : ▪ `manually` ▪ `every_minute` ▪ `every_2_minutes` ▪ `every_5_minutes` ▪ `every_10_minutes` ▪ `every_20_minutes` ▪ `every_30_minutes` ▪ `once_per_hour` Defaults to `manually` when unset.
D	SOGoMailAuxiliaryUserAccountsEnabled	Parameter used to activate the auxiliary IMAP accounts in SOGo. When set to YES, users can add other IMAP accounts that will be visible from the SOGo Webmail interface. Defaults to NO when unset.
U	SOGoDefaultCalendar	Parameter used to specify which calendar is used when creating an event or a task. Possible values

		are : ▪ `selected` ▪ `personal` ▪ `first` Defaults to `selected` when unset.
U	SOGoDayStartTime	The hour at which the day starts (0 through 12). Defaults to 8 when unset.
U	SOGoDayEndTime	The hour at which the day ends (12 through 23). Defaults to 18 when unset.
U	SOGoFirstDayOfWeek	The day at which the week starts in the week and month views (0 through 6). 0 indicates Sunday. Defaults to 0 when unset.
U	SOGoFirstWeekOfYear	Parameter used to defined how is identified the first week of the year. Possible values are : ▪ `January1` ▪ `First4DayWeek` ▪ `FirstFullWeek` Defaults to `January1` when unset.
U	SOGoTimeFormat	The format used to display time in the timeline of the day and week views. Please refer to the documentation for the *date* command or the *strftime* C function for the list of available format sequence. Defaults to `%H:%M`.
U	SOGoCalendarCategories	Parameter used to define the categories that can be associated to events. This parameter is an array of arbitrary strings. Defaults to a list that depends on the language.
U	SOGoCalendarDefaultCategoryColor	Parameter used to define the default colour of categories. Defaults to `#F0F0F0` when unset.
D	SOGoFreeBusyDefaultInterval	The number of days to include in the free busy information. The parameter is an array of two numbers, the first being the number of days prior to the current day and the second being the number of days following the current day. Defaults to `(7, 7)` when unset.
U	SOGoBusyOffHours	Parameter used to specify if off-hours should be automatically added to the free-busy information. Off hours included weekends and periods covered between `SOGoDayEndTime` and `SOGoDayStartTime`. Defaults to NO when unset.
U	SOGoMailMessageForwarding	The method the message is to be forwarded. Possible values are : ▪ `inline`

		▪ `attached` Defaults to `inline` when unset.
U	SOGoMailReplyPlacement	The reply placement with respect to the quoted message. Possible values are : ▪ `above` ▪ `below` Defaults to `below`.
U	SOGoMailSignaturePlacement	The placement of the signature with respect to the quoted message. Possible values are : ▪ `above` ▪ `below` Defaults to `below`.
U	SOGoMailComposeMessageType	The message composition format. Possible values are : ▪ `text` ▪ `html` Defaults to `text`.
S	SOGoEnableEMailAlarms	Parameter used to enable email-based alarms on events and tasks. See the "EMail reminders" section from this document for more information.
U	SOGoContactsCategories	Parameter used to define the categories that can be associated to contacts. This parameter is an array of arbitrary strings. Defaults to a list that depends on the language.
D	SOGoUIAdditionalJSFiles	Parameter used to define the list of additional JavaScript files loaded by SOGo for all displayed web pages. This parameter is an array of strings corresponding of paths to the arbitrary JavaScript files.

SOGo Configuration Summary

The complete SOGo configuration file, which is located in `/home/sogo/GNUstep/Defaults/.GNUstepDefaults` should look like this :

```
{
    "sogod" = {
       SOGoProfileURL =
"postgresql://sogo:sogo@localhost:5432/sogo/sogo_user_profile";
       OCSFolderInfoURL =
"postgresql://sogo:sogo@localhost:5432/sogo/sogo_folder_info";
       OCSSessionsFolderURL =
       “postgresql://sogo:sogo@localhost:5432/sogo/sogo_sessions_folder”;
       SOGoAppointmentSendEMailNotifications = YES;
```

```
        SOGoCalendarDefaultRoles = (
              PublicViewer,
              ConfidentialDAndTViewer
        );
        SOGoLanguage = English;
        SOGoMailDomain = acme.com;
        SOGoDraftsFolderName = Drafts;
        SOGoIMAPServer = localhost;
        SOGoUserSources = (
            {
              type = ldap;
              CNFieldName = cn;
              IDFieldName = uid;
              UIDFieldName = uid;
              baseDN = "ou=users,dc=acme,dc=com";
              bindDN = "uid=sogo,ou=users,dc=acme,dc=com";
              bindPassword = qwerty;
              canAuthenticate = YES;
              displayName = "Shared Addresses";
              hostname = localhost;
              id = public;
              isAddressBook = YES;
              port = 389;
            }
        );
        SOGoMailingMechanism = smtp;
        SOGoSMTPServer = 127.0.0.1;
        SOGoSentFolderName = Sent;
        SOGoTimeZone = America/Montreal;
        SOGoTrashFolderName = Trash;
    };
}
```

Multi-domains Configuration

If you want your installation to isolate two groups of users, you must define a distinct authentication source for each *domain*. Following is the same configuration that now includes two domains (acme and coyote) :

```
{
   "sogod" = {
       SOGoProfileURL =
"postgresql://sogo:sogo@localhost:5432/sogo/sogo_user_profile";
       OCSFolderInfoURL =
"postgresql://sogo:sogo@localhost:5432/sogo/sogo_folder_info";
       OCSSessionsFolderURL =
       “postgresql://sogo:sogo@localhost:5432/sogo/sogo_sessions_folder”;
       SOGoAppointmentSendEMailNotifications = YES;
```

```
SOGoCalendarDefaultRoles = (
       PublicViewer,
       ConfidentialDAndTViewer
);
SOGoLanguage = English;
SOGoMailingMechanism = smtp;
SOGoSMTPServer = 127.0.0.1;
SOGoSentFolderName = Sent;
SOGoTimeZone = America/Montreal;
SOGoTrashFolderName = Trash;
SOGoIMAPServer = localhost;
domains = {
       acme = {
              SOGoMailDomain = acme.com;
              SOGoDraftsFolderName = Drafts;
              SOGoUserSources = (
                  {
                     type = ldap;
                     CNFieldName = cn;
                     IDFieldName = uid;
                     UIDFieldName = uid;
                     baseDN = "ou=users,dc=acme,dc=com";
                     bindDN = "uid=sogo,ou=users,dc=acme,dc=com";
                     bindPassword = qwerty;
                     canAuthenticate = YES;
                     displayName = "Shared Addresses";
                     hostname = localhost;
                     id = public_acme;
                     isAddressBook = YES;
                     port = 389;
                  }
              );
       };
       coyote = {
              SOGoMailDomain = coyote.com;
              SOGoIMAPServer = imap.coyote.com;
              SOGoUserSources = (
                  {
                     type = ldap;
                     CNFieldName = cn;
                     IDFieldName = uid;
                     UIDFieldName = uid;
                     baseDN = "ou=users,dc=coyote,dc=com";
                     bindDN = "uid=sogo,ou=users,dc=coyote,dc=com";
                     bindPassword = qwerty;
                     canAuthenticate = YES;
                     displayName = "Shared Addresses";
                     hostname = localhost;
                     id = public_coyote;
                     isAddressBook = YES;
                     port = 389;
                  }
```

```
                    );
                }
            };
        };
    }
```

The following additional parameters only affect SOGo when using multiple domains.

S	SOGoEnableDomainBasedUID	Parameter used to activate user identification by domain. Users will be able (without being required) to login using the form `username@domain`, meaning that values of `UIDFieldName` no longer have to be unique among all domains but only within the same domain. Internally, users will always be identified by the concatenation of their username and domain. Consequently, activating this parameter on an existing system implies that user identifiers will change and their previous calendars and address books will no longer be accessible unless a conversion is performed. Defaults to `NO` when unset.
S	SOGoLoginDomains	Parameter used to define which domains should be selectable from the login page. This parameter is an array of keys from the `domains` dictionary. Defaults to an empty array, which means that no domains appear on the login page.
S	SOGoDomainsVisibility	Parameter used to set domains visible among themselves. This parameter is an array of arrays. Example: `SOGoDomainsVisibility = ((acme, coyote));` Defaults to an empty array, which means domains are isolated from each other.

Apache Configuration

The SOGo configuration for Apache is located in `/etc/httpd/conf.d/SOGo.conf`.

Upon SOGo installation, a default configuration file is created which is suitable for most configurations.

You must also configure the following parameters in the SOGo configuration file for Apache in order to have a working installation :

```
RequestHeader set "x-webobjects-server-port" "80"
RequestHeader set "x-webobjects-server-name" "yourhostname"
RequestHeader set "x-webobjects-server-url" "http://yourhostname"
```

You may consider enabling SSL on top of this current installation to secure access to your SOGo installation.

See http://httpd.apache.org/docs/2.2/ssl/ for details.

You might also have to adjust the configuration if you have SELinux enabled.

The default configuration will use `mod_proxy` and `mod_headers` to relay requests to the `sogod` parent process. This is suitable for small to medium deployments.

Starting Services

Once SOGo if fully installed and configured, start the services using the following command :

```
service sogod start
```

You may verify using the `chkconfig` command that the SOGo service is automatically started at boot time. Restart the Apache service since modules and configuration files were added :

```
service httpd restart
```

Finally, you should also make sure that the `memcached` service is started and that it is also automatically started at boot time.

Cronjob — EMail reminders

SOGo allows you to set email-based reminders for events and tasks. To enable this, you must enable the SOGoEnableEMailAlarms preference and set the OCSEMailAlarmsFolderURL preference accordingly.

Once you've correctly set those two preferences, you must create a *cronjob* that will run under the "sogo" user. This *cronjob* should be run every minute so an entry in the contrab should be defined like :

```
* * * * *       /path/to/sogo-ealarms-notify
```

Cronjob — Vacation messages expiration

When vacation messages are enabled (see the parameter "SOGoVacationEnabled"), users can set an expiration date to messages auto-reply. For this feature to work, you must create a *cronjob* under the "sogo" user. This *cronjob* should be run daily and defined like so :

```
0 * * * *       /path/to/sogo-tool expire-autoreply
```

Managing User Accounts

Creating the SOGo Administrative Account

First, create the SOGo administrative account in your LDAP server. The following LDIF file (`sogo.ldif`) can be used as an example :

```
dn: uid=sogo,ou=users,dc=acme,dc=com
objectClass: top
objectClass: inetOrgPerson
objectClass: person
objectClass: organizationalPerson
uid: sogo
cn: SOGo Administrator
mail: sogo@acme.com
sn: Administrator
givenName: SOGo
```

Load the LDIF file inside your LDAP server using the following command :

```
ldapadd -f sogo.ldif -x -w qwerty -D cn=Manager,dc=acme,dc=com
```

Finally, set the password (to the value "qwerty") of the SOGo administrative account using the following command :

```
ldappasswd -h localhost -x -w qwerty -D cn=Manager,dc=acme,dc=com
uid=sogo,ou=users,dc=acme,dc=com -s qwerty
```

Creating a User Account

SOGo uses LDAP directories to authenticate users. Use the following LDIF file (`jdoe.ldif`) as an example to create a SOGo user account :

```
dn: uid=jdoe,ou=users,dc=acme,dc=com
objectClass: top
objectClass: inetOrgPerson
objectClass: person
```

```
objectClass: organizationalPerson
uid: jdoe
cn: John Doe
mail: jdoe@acme.com
sn: Doe
givenName: John
```

Load the LDIF file inside your LDAP server using the following command :

```
ldapadd -f jdoe.ldif -x -w qwerty -D cn=Manager,dc=acme,dc=com
```

Finally, set the password (to the value "qwerty") of the SOGo administrative account using the following command :

```
ldappasswd -h localhost -x -w qwerty -D cn=Manager,dc=acme,dc=com
uid=jdoe,ou=users,dc=acme,dc=com -s qwerty
```

As an alternative to using command-line tools, you can also use LDAP editors such as *Luma* or *Apache Directory Studio* to make your work easier. These GUI utilities can make use of templates to create and pre-configure typical user accounts or any standardized LDAP record, along with the correct object classes, fields and default values.

Funambol

The Funambol middleware allows you to synchronize mobile devices with SOGo through the use of the Funambol SOGo Connector. The connector allows any SyncML enabled devices to fully synchronize contacts, events and tasks with SOGo.

First of all, install and configure Funambol v8.7. We suppose Funambol was installed in `/opt/Funambol`.

If running after installation, stop the Funambol server using :

```
/opt/Funambol/bin/funambol.sh stop
```

Download your preferred JDBC driver and move it to :

```
/opt/Funambol/tools/tomcat/lib/
```

The Funambol SOGo Connector currently supports only MySQL, Oracle and PostgreSQL. You can download the jar file for PostgreSQL from http://jdbc.postgresql.org/. For Oracle, please refer to the following site : http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/htdocs/jdbc_10201.html and download the ojdbc14.jar file. For MySQL, please refer to the following site : http://dev.mysql.com/usingmysql/java/

You must also download the JSON.simple package from http://code.google.com/p/json-simple/downloads/list and place it in :

```
/opt/Funambol/tools/tomcat/lib/
```

Then, download the funambol-sogo-1.0.8.s4j file from http://www.sogo.nu/downloads/backend.html and move it to :

```
/opt/Funambol/ds-server/modules
```

Then, open the `/opt/Funambol/ds-server/install.properties` file and add "`funambol-sogo-1.0.8`" at the end of the "`modules-to-install`" line.

Start the Funambol server using :

```
/opt/Funambol/bin/funambol start
```

Next, install the Funambol SOGo Connector within Funambol server by issuing the following commands :

```
cd /opt/Funambol/
./bin/install-modules
```

Answer 'yes' to all questions.

Then, configure the data sources for SOGo. To do so, start the Funambol Administration Tool using the following command :

```
/opt/Funambol/admin/bin/funamboladmin
```

Log in.

Go in Modules > sogo > FunambolSOGoConnector > SOGo SyncSource and add a source for each data type you would like to synchronize. For example, to synchronize an address book, you would specify:

```
Source URI:             sogo-card
Name:                   sogo-card
Supported type:         text/x-vcard
Database URL:           jdbc:postgresql://localhost/sogo
Database username:      sogo
Database password:      sogo
```

You can then do the same (and specify the same database connection information) for events and tasks using `sogo-cal` and `sogo-todo` as sync source names and URI.

If you want to auto-create Funambol user accounts for every users that can authenticate to SOGo, you can use the SOGoOfficer to do so. From the Funambol Administration Tool, in "Server Settings", set the Officer to the following value :

```
ca/inverse/sogo/security/SOGoOfficer.xml
```

and save the preferences. Then, modify the following file :

```
/opt/Funambol/config/ca/inverse/sogo/security/SOGoOfficer.xml
```

change the host property to the host name value of your SOGo server. Change the port property to the port value of your `sogod` daemon. No server restart is required. In our example, the file would look like :

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.0" class="java.beans.XMLDecoder">
    <object class="ca.inverse.sogo.security.SOGoOfficer">
        <void property="host">
            <string>localhost</string>
        </void>
        <void property="port">
            <string>20000</string>
        </void>
    </object>
</java>
```

Using SOGo

SOGo Web Interface

To acces the SOGo Web Interface, point your Web browser, which is running from the same server where SOGo was installed, to the following URL : http://localhost/SOGo

Log in using the “jdoe” user and the “qwerty” password. The underlying database tables will automatically be created by SOGo.

Mozilla Thunderbird and Lightning

Alternatively, you can access SOGo with a GroupDAV and a CalDAV client. A typical well-integrated setup is to use Mozilla Thunderbird and Mozilla Lightning along with Inverse's *SOGo Connector* plug in to synchronize your address books and the Inverse's *SOGo Integrator* plug in to provide a complete integration of the features of SOGo into Thunderbird and Lightning. Refer to the documentation of Thunderbird to configure an initial IMAP account pointing to your SOGo server and using the user name and password mentioned above.

With the SOGo Integrator plug in, your calendars and address books will be automatically discovered when you login in Thunderbird. This plug in can also propagate specific extensions and default user settings among your site. However, be aware that in order to use the SOGo Integrator plug in, you will need to repackage it with specific modifications. Please refer to the documentation published online:

http://www.sogo.nu/downloads/documentation.html.

If you only use the SOGo Connector plug in, you can still easily access your data.

To access your personal address book:

- Choose Go > Address Book.
- Choose File > New > Remote Address Book.
- Enter a significant name for your calendar in the Name field.
- Type the following URL in the URL field:
  http://<hostname>/SOGo/dav/jdoe/Contacts/personal/

- ❑ Click on OK.

To access your personal calendar:

- ❑ Choose Go > Calendar.
- ❑ Choose Calendar > New Calendar.
- ❑ Select On the Network and click on Continue.
- ❑ Select CalDAV.
- ❑ Type the following URL in the URL field:
  http://localhost/SOGo/dav/jdoe/Calendar/personal/
- ❑ Click on Continue.

Apple iCal 3 and 4

Apple iCal 3 or 4 can also be used as a client application for SOGo.

To configure it so it works with SOGo, create a new account and specify, as the Account URL, an URL such as :

http://localhost/SOGo/dav/jdoe/

Note that the trailing slash is important for Apple iCal 3.

Apple AddressBook (Mac OS X 10.6)

Apple AddressBook on Mac OS X 10.6 (Snow Leopard) can be configured to use SOGo.

In order to make this work, you must add a new virtual host in your Apache configuration file to listen on port 8800 and handle requests coming from iOS devices.

The virtual host should be defined like :

```
<VirtualHost *:8800>
  RewriteEngine Off
  ProxyRequests Off
  SetEnv proxy-nokeepalive 1
  ProxyPreserveHost On
  ProxyPassInterpolateEnv On
  ProxyPass /principals http://127.0.0.1:20000/SOGo/dav/ interpolate
  ProxyPass /SOGo http://127.0.0.1:20000/SOGo interpolate
  ProxyPass / http://127.0.0.1:20000/SOGo/dav/ interpolate
```

```
  <Location />
    Order allow,deny
    Allow from all
  </Location>
  <Proxy http://127.0.0.1:20000>
    RequestHeader set "x-webobjects-server-port" "8800"
    RequestHeader set "x-webobjects-server-name" "acme.com:8800"
    RequestHeader set "x-webobjects-server-url" "http://acme.com:8800"
    RequestHeader set "x-webobjects-server-protocol" "HTTP/1.0"
    RequestHeader set "x-webobjects-remote-host" "127.0.0.1"
    AddDefaultCharset UTF-8
  </Proxy>
  ErrorLog /var/log/apache2/ab-error.log
  CustomLog /var/log/apache2/ab-access.log combined
</VirtualHost>
```

This configuration is also required if you want to configure a CardDAV account on a Apple iOS device (version 4.0 and later).

Funambol / Mobile Devices

You can synchronize contacts, events and tasks from SOGo with any mobile devices that support SyncML.

Your Funambol server needs to be accessible from outside of your internal network as synchronization happens over the air ("OTA").

Once you've made your server visible from the Internet, please specify the following URL in your SyncML client:

```
http://<external IP address>:8080/funambol/ds
```

Then, specify the following data sources:

- For contacts: `sogo-card`
- For events: `sogo-cal`
- For tasks: `sogo-todo`

The user name / password is the same as the one you can use to log in SOGo.

For more details on mobile devices, such as Apple iPhone, please refer to the *SOGo Mobile Devices – Installation and Configuration* guide available from http://www.sogo.nu.

Upgrading

This section describes what needs to be done when upgrading to the current version of SOGo from the previous release.

1.3.8 to 1.3.9
For Red Hat-based distributions, version 1.23 of GNUstep will be installed. Since the location of the Web resources changes, the Apache configuration file (SOGo.conf) has been adapted. Verify your Apache configuration if you have customized this file.
1.3.9 to 1.3.10
No specific update procedure required.

Chapter 10

Additional Information

For more information, please consult the online FAQs (Frequently Asked Questions) :

http://www.sogo.nu/english/support/faq.html

You can also read the mailing archives or post your questions to it. For details, see :

https://inverse.ca/sogo/lists

Chapter 11

Commercial Support and Contact Information

For any questions or comments, do not hesitate to contact us by writing an email to :

support@inverse.ca

Inverse (http://inverse.ca) offers professional services around SOGo and Funambol to help organizations deploy the solution and migrate from their legacy systems.