Installation and Configuration Guide

for version 3.0.1

Installation and Configuration Guide

Version 3.0.1 - February 2016

Table of 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 SOGo ActiveSync support – the solution used to synchronize mobile devices with SOGo.

The instructions are based on version 3.0.1 of SOGo.

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

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.

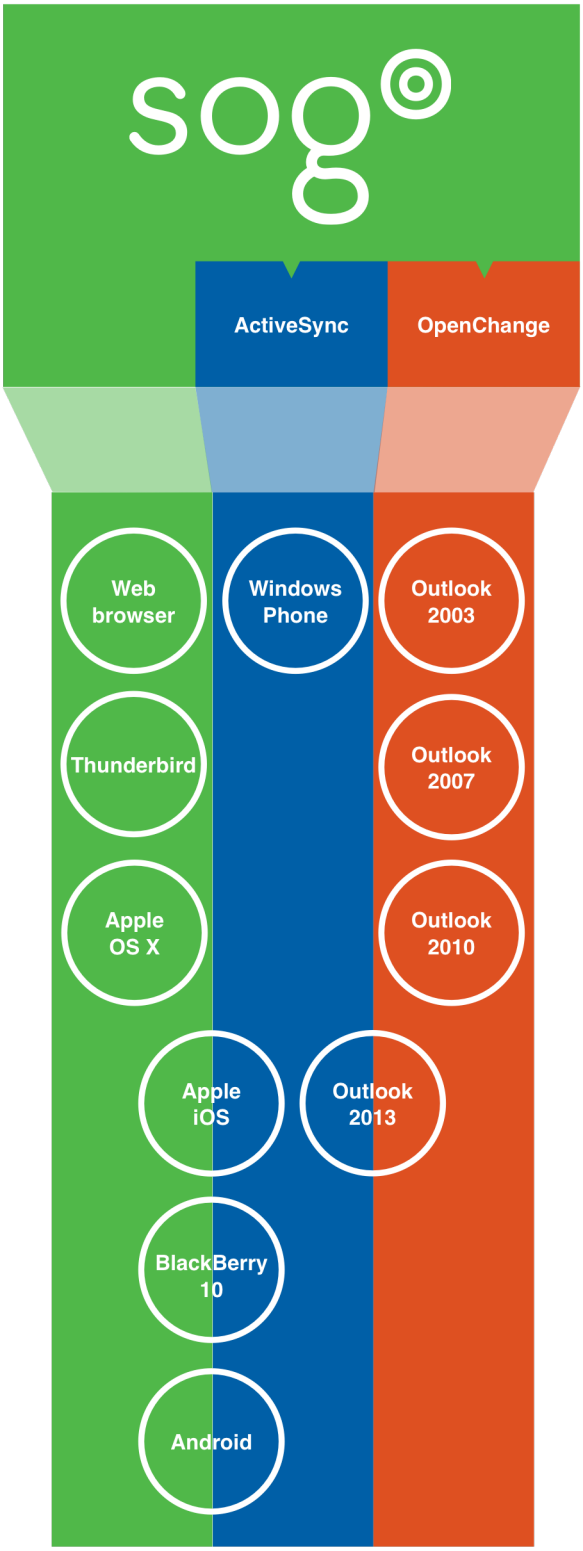
SOGo is standard-compliant. It supports CalDAV, CardDAV, GroupDAV, iMIP and iTIP and reuses existing IMAP, SMTP and database servers - making the solution easy to deploy and interoperable with many applications.

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
- Native compatibility for Microsoft Outlook 2003, 2007, 2010, and 2013
- Two-way synchronization support with any Microsoft ActiveSync-capable device, or Outlook 2013

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 and Compatibility



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 Microsoft ActiveSync protocol are also supported.

To install and configure the native Microsoft Outlook compatibility layer, please refer to the *SOGo Native Microsoft Outlook Configuration Guide*.

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)

If you plan to use ActiveSync, an IMAP server supporting the ACL, UIDPLUS, QRESYNC, ANNOTATE (or X-GUID) IMAP extensions is required, such as Cyrus IMAP version 2.4 or later, or Dovecot version 2.1 or later. If your current IMAP server does not support these extensions, you can use Dovecot's proxying capabilities.

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 ▪ 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 CalDAV, CardDAV or Microsoft ActiveSync.

Operating System Requirements

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

- Red Hat Enterprise Linux (RHEL) Server 5, 6 and 7
- Community ENTerprise Operating System (CentOS) 5, 6 and 7
- Debian GNU/Linux 6.0 (Squeeze) to 8.0 (Jessie)

- Ubuntu 12.04 (Precise) to 14.04 (Trusty)

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.

Note

This document covers the installation of SOGo under RHEL 6.

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 Red Hat or CentOS 6 distribution. Most of these steps should apply to all supported operating systems.

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/rhel-v3/6/$basearch
gpgcheck=0
```

Some of the softwares on which SOGo depends are available from the repository of RepoForge (previously known as RPMforge). To add RepoForge 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 RepoForge, visit http://repoforge.org/use/.

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".

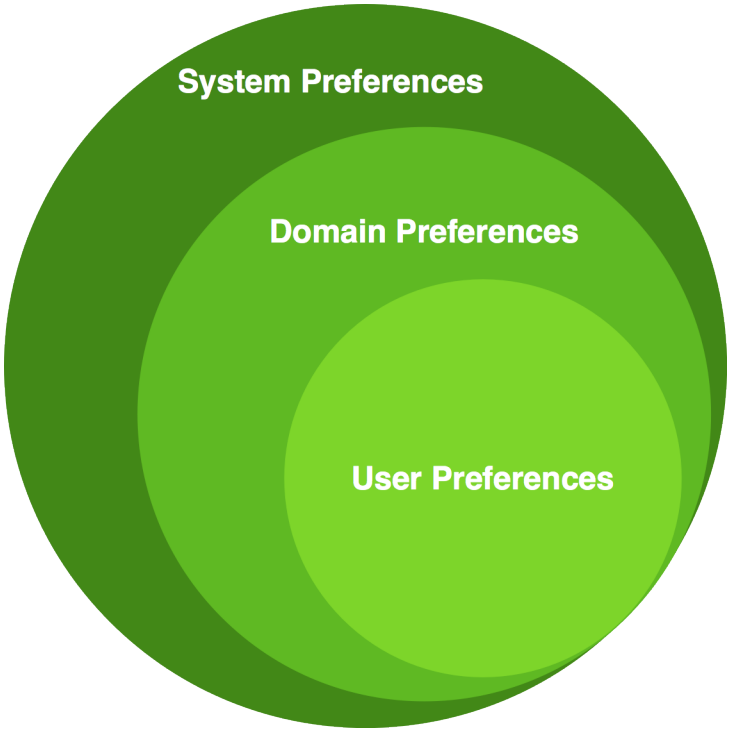
In SOGo, the user's applications settings are stored in `/etc/sogo/sogo.conf`. You can use your favourite text editor to modify the file.

The `sogo.conf` 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. Block comments are delimited by `/*` and `*/` and can span multiple lines while line comments must start with `//`.

The configuration must be contained in a root dictionary, thus be completely wrapped within curly brackets `{ [configuration] }`. If SOGo refuses to start due to syntax errors in its configuration file, `plparse` is helpful for finding these, as it indicates the line containing the problem.

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 `/etc/sogo/sogo.conf`, 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

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	WOListenQueueSize	This parameter controls the backlog size of the socket listen queue. For large-scale deployments, this value must be adjusted in case all workers are busy and the parent processes receives lots of incoming connections. Defaults to 5 when unset.
S	WOPort	The TCP listening address and port used by the SOGo daemon. The format is `ipaddress:port`. Defaults to `127.0.0.1:20000` 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	WOWatchDogRequestTimeout	This parameter specifies the number of minutes after which a busy child process will be killed by the parent process. Defaults to 10 (minutes). Do not set this too low as child processes replying to clients on a slow internet connection could be killed prematurely.
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 and optionally the port of the memcached server. A path can also be used if the server must be reached via a Unix socket. Defaults to `localhost`. See `memcached_servers_parse(3)` for details on the syntax.
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 SAML2, specify `saml2`. For anything else, leave that value empty.
S	SOGoTrustProxyAuthentication	Parameter used to set whether HTTP username should be trusted. Defaults to `NO` when unset.
S	SOGoEncryptionKey	Parameter used to define a key to encrypt the passwords of remote Web calendars when *SOGoTrustProxyAuthentication* is enabled.
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.
S	SOGoCASLogoutEnabled	Boolean value indicating whether the "Logout" link is enabled when using C.A.S. as authentication mechanism. The "Logout" link will end up calling *SOGoCASServiceURL*/logout to terminate the client's single sign-on C.A.S. session.
S	SOGoAddressBookDAVAccessEnabled	Parameter controlling WebDAV access to the Contacts collections. This can be used to deny access to these resources from Lightning for example. Defaults to `YES` when unset.
S	SOGoCalendarDAVAccessEnabled	Parameter controlling WebDAV access to the Calendar collections. This can be used to deny access to these resources from Lightning for example. Defaults to `YES` when unset.
S	SOGoSAML2PrivateKeyLocation	The location of the SSL private key file on the filesystem that is used by SOGo to sign and encrypt communications with the SAML2 identity provider. This file must be generated for each running SOGo service (rather than host). Make sure this file is readable by the SOGo user.
S	SOGoSAML2CertiticateLocation	The location of the SSL certificate file. This file must be generated for each running SOGo service. Make sure this file is readable by the SOGo user.
S	SOGoSAML2IdpMetadataLocation	The location of the metadata file that describes the services available on the SAML2 identify provider. The content of this file is usually generated directly by your SAML

		2.0 IdP solution. For example, using SimpleSAMLphp, you can get the metadata directly from https://MYSERVER/simplesaml/saml2/idp/metadata.php Make sure this file is readable by the SOGo user.
S	SOGoSAML2IdpPublicKeyLocation	The location of the SSL public key file on the filesystem that is used by SOGo to sign and encrypt communications with the SAML2 identity provider. This file should be part of the setup of your identity provider. Make sure this file is readable by the SOGo user.
S	SOGoSAML2IdpCertificateLocation	The location of the SSL certificate file. This file should be part of the setup of your identity provider. Make sure this file is readable by the SOGo user.
S	SOGoSAML2LoginAttribute	The attribute provided by the IdP to identify the user in SOGo.
S	SOGoSAML2LogoutEnabled	Boolean value indicated whether the "Logout" link is enabled when using SAML2 as authentication mechanism. When using this feature, SOGo will invoke the IdP to proceed with the logout procedure. When the user clicks on the logout button, a redirection will be made to the IdP to trigger the logout.
S	SOGoSAML2LogoutURL	The URL to which redirect the user after the "Logout" link is clicked. SOGoSAML2LogoutEnabled must be set to YES. If unset, the user will be redirected to a blank page.
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. Defaults roles are ignored for public accesses. 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 `Confidential-DAndTViewer`.
D	SOGoContactsDefaultRoles	Parameter used to define the default roles when giving permissions to a user to access an address book. Defaults roles are ignored for public accesses. 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: ▪ `Arabic` ▪ `Basque` ▪ `BrazilianPortuguese` ▪ `Catalan` ▪ `Czech` ▪ `Danish` ▪ `Dutch` ▪ `English` ▪ `Finnish` ▪ `French` ▪ `German` ▪ `Hungarian` ▪ `Icelandic` ▪ `Italian` ▪ `NorwegianBokmal` ▪ `NorwegianNynorsk` ▪ `Polish` ▪ `Russian` ▪ `Slovak` ▪ `SpanishSpain` ▪ `SpanishArgentina` ▪ `Swedish` ▪ `Ukrainian` ▪ `Welsh`
D	SOGoNotifyOnPersonalModifications	Parameter used to set whether SOGo sends or not email receipts when someone changes his/her own calendar. Possible values are: ▪ `YES` – to send notifications ▪ `NO` – to not send notifications Defaults to `NO` when unset. User can overwrite this from the calendar properties window.

D	SOGoNotifyOnExternalModifications	Parameter used to set whether SOGo sends or not email receipts when a modification is being done to his/her own calendar by someone else. Possible values are: ▪ `YES` – to send notifications ▪ `NO` – to not send notifications Defaults to `NO` when unset. User can overwrite this from the calendar properties window.
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. For this feature to work properly when authenticating against AD or Samba4, the LDAP connection must use SSL/TLS. Server side restrictions can also cause the password change to fail, in which case SOGo will only log a *Constraint violation (0x13)* error. These restrictions include password too young, complexity constraints not satisfied, user cannot change password, etc... Also note that Samba has a minimum password age of 1 day by default.
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: `("Arabic", "Basque", "Catalan", "Czech", "Dutch", "Danish", "Welsh", "English", "SpanishSpain", "SpanishArgentina", "Finnish", "French", "German", "Icelandic", "Italian", "Hungarian", "BrazilianPortuguese", "NorwegianBokmal", "NorwegianNynorsk", "Polish", "Russian", "Slovak", "Ukrainian", "Swedish")`
D	SOGoHideSystemEMail	Parameter used to control if SOGo should hide or not the system email address (UIDFieldName@SOGoMailDomain). This is currently limited to CalDAV (calendar-user-address-set). Defaults to `NO` when unset.
D	SOGoSearchMinimumWordLength	Parameter used to control the minimum length to be used for the search string (attendee completion, address book search, etc.) prior triggering the server-side search operation. Defaults to `2` when unset – which means a search operation will be triggered on the 3rd typed character.
S	SOGoMaximumFailedLoginCount	Parameter used to control the number of failed login attempts required during *SOGoMaximumFailedLoginInterval* seconds or more. If conditions are met, the account will be blocked for *SOGoFailedLoginBlockInterval* seconds since the first failed login attempt. Default value is `0`, or disabled.
S	SOGoMaximumFailedLoginInterval	Number of seconds, defaults to `10`.
S	SOGoFailedLoginBlockInterval	Number of seconds, defaults to `300` (or 5 minutes). Note that *SOGoCacheCleanupInterval* must be set to a value equal or higher than *SOGoFailedLoginBlockInterval*.
S	SOGoMaximumMessageSubmissionCount	Parameter used to control the number of email messages a user can send from SOGo's webmail interface, to *SOGoMaximumRecipientCount*, in *SOGoMaximumSubmissionInterval* seconds or more. If conditions are met or exceeded, the user won't be able to send mails for *SOGoMessageSubmissionBlockInterval* seconds. Default value is `0`, or disabled.
S	SOGoMaximumRecipientCount	Maximum number of recipients. Default value is `0`, or disabled.
S	SOGoMaximumSubmissionInterval	Number of seconds, defaults to `30`.
S	SOGoMessageSubmissionBlockInterval	Number of seconds, default to `300` (or 5 minutes). Note that *SOGoCacheCleanupInterval*

		must be set to a value equal or higher than *SOGoFailedLoginBlockInterval*.

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). Insert the following text into your configuration file to configure an authentication and global address book using an LDAP directory server:

```
SOGoUserSources = (
    {
        type = ldap;
        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 = "ldap://127.0.0.1:389";
        id = public;
        isAddressBook = YES;
    }
);
```

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, in your `.GNUstepDefaults` file, the following filter to return only entries belonging to the organization *Inverse* with a *mail* address and not *inactive*:

```
filter = "(o='Inverse' AND mail='*' AND status <> 'inactive')";
```

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 = "<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 = ldap://10.0.0.1:389;
        id = directory;
        isAddressBook = YES;
    }
);
```

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:

```
{
    type = ldap;
    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 = ldap://127.0.0.1:389;
    id = inverse_groups;
    isAddressBook = YES;
}
```

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 of a user. The returned value **must be unique across the whole SOGo installation** since it is used to identify the user in the `folder_info` database table.

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 either an URI to the IMAP server as described for SOGoIMAPServer, or a simple server hostname that would be used as a replacement for the hostname part in the URI provided by the *SOGoIMAPServer* parameter.
IMAPLoginFieldName (optional)	The field that returns the IMAP login name for the user (defaults to the value of *UIDFieldName* when unset).
SieveHostFieldName (optional)	The field that returns either an URI to the SIEVE server as described for *SOGoSieveServer*, or a simple server hostname that would be used as a replacement for the hostname part in the URI provided by the *SOGoSieveServer* parameter.
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)	The value of this attribute is the maximum number of concurrent events to which a resource can be part of at any point in time. If this is set to `0`, or if the attribute is missing, it means no limit. If set to `-1`, no limit is imposed but the resource will be marked as busy the first time it is booked.
filter (optional)	The filter to use for LDAP queries, it should be defined as an EOQualifier. The following operators are supported: ▪ `<>` – inequality operator ▪ `=` – equality operator Multiple qualifiers can be joined by using `OR` and `AND`, they can also be grouped together by using parenthesis. Attribute values should be quoted to avoid unexpected behaviour.

	For example: `filter = "(objectClass='mailUser' OR objectClass='mailGroup') AND accountStatus='active' AND uid <> 'alice'";`
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 URLs or LDAP hostnames. LDAP URLs are specified in RFC 4516 and have the following general format: `scheme://host:port/DN?attributes?scope?filter?extensions` Note that SOGo doesn't currently support DN, attributes, scope and filter in such URLs. Using them may have undefined side effects. URLs examples: ▪ `ldap://127.0.0.1:3389` ▪ `ldaps://127.0.0.1` ▪ `ldap://127.0.0.1/????!StartTLS`
port(deprecated)	Port number of the LDAP server. A non-default port should be part of the ldap URL in the hostname parameter.
encryption (deprecated)	Either `SSL` or `STARTTLS` SSL should be specified as `ldaps://` in the LDAP URL. STARTTLS should be specified as a LDAP Extension in the LDAP URL (e.g. `ldap://127.0.0.1/????!StartTLS`)
userPasswordAlgorithm	The algorithm used for password encryption when changing passwords without Password Policies enabled. Possible values are: `none`, `plain`, `crypt`, `md5`, `md5-crypt`, `smd5`, `cram-md5` and `sha`, `sha256`, `sha512` and its ssha (e.g. `ssha` or `ssha256`) vari-

	ants (plus setting of the encoding with `.b64` or `.hex`). For a more detailed description see http://wiki.dovecot.org/Authentication/PasswordSchemes. Note that `cram-md5` is not actually using cram-md5 (due to the lack of challenge-response mechanism), its just saving the intermediate MD5 context as Dovecot stores in its database.
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. Note that some LDAP servers require LDAP/SSL for password policies to work. This is the case for example with 389 Directory Server.
updateSambaNTLMPasswords	If set to `YES`, SOGo will automatically update the sambaNTPassword and sambaLMPassword attributes when changing passwords. The attributes must be called sambaNTPassword and sambaLMPassword. You must also make sure the correct ACL is set in your LDAP server to allow users to change their own sambaNTPassword and sambaLMPassword password attributes. Defaults to `NO` when unset.
isAddressBook	If set to `YES`, this LDAP source is used as a shared address book (with read-only access). Note that if set to `NO`, autocompletion will not work for entries in this source and thus, free-busy lookups.
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`, and/or `ActiveSync` for example:
	
	`ModulesConstraints = {`
	`  Calendar = {`
	`  ou = employees;`
	`  };`
	`};`

mapping	A dictionary that maps contact attributes used by SOGo to the LDAP attributes used by the schema of the LDAP source. Each entry must have an attribute name as key and an array of

		strings as value. This enables actual fields to be mapped one after another when fetching contact informations. See the LDAP Attribute Mapping section below for an example and a list of supported attributes.
	objectClasses	When the *modifiers* list (see below) is set, or when using LDAP-based user addressbooks (see *abOU* below), this list of object classes will be applied to new records as they are created.
	GroupObjectClasses	A list (array) of names identifying groups within the LDAP source. If not set, SOGo will use `group`, `groupofnames`, `groupofuniquenames` and `posixgroup`.
	modifiers	A list (array) of usernames that are authorized to perform modifications to the address book defined by this LDAP source.

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.

LDAP Attributes Mapping

Some LDAP attributes are mapped to contacts attributes in the SOGo UI. The table below list most of them. It is possible to override these by using the *mapping* configuration parameter.

For example, if the LDAP schema uses the *fax* attribute to store the fax number, one could map it to the *facsimiletelephonenumber* attribute like this:

```
mapping = {
  facsimiletelephonenumber = ("fax", "facsimiletelephonenumber");
};
```

Name	
First	givenName
Last	sn
DisplayName	displayName *or* cn *or* givenName + sn
Nickname	mozillanickname
Internet	
Email	mail
Secondary email	mozillasecondemail
ScreenName	nsaimid
Phones	
Work	telephoneNumber
Home	homephone
Mobile	mobile
Fax	facsimiletelephonenumber
Pager	pager
Home	
Address	mozillahomestreet + mozillahomestreet2
City	mozillahomelocalityname
State/Province	mozillahomestate
Zip/Postal Code	mozillahomepostalcode
Country	mozillahomecountryname
Web page	mozillahomeurl
Work	
Title	title

Department	ou
Organization	o
Address	street + mozillaworkstreet2
City	l
State/Province	st
Zip/Postal code	postalCode
Country	c
Web page	mozillaworkurl
Other	
Birthday	birthyear-birthmonth-birthday
Note	description

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 used by the SOGo server has to be recognized and trusted 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 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. On debian, the SOGo certificate can also be added to the truststore as follows:

```
openssl x509 -in /etc/ssl/certs/sogo-cert.pem -outform DER \
  -out /tmp/sogo-cert.der
keytool -import -keystore /etc/ssl/certs/java/cacerts  \
  -file /tmp/sogo-cert.der -alias sogo-cert
# The keystore password is 'changeit'
# tomcat must be restarted after this operation
```

The certificate used by the CAS server must also be trusted by SOGo. In case of a self-signed certificate, this means exporting tomcat's certificate using the `keytool` utility, converting it to PEM format and appending it to the `ca-certificates.crt` file (the name and location of that file differs between distributions). Basically:

```
# export tomcat's cert to openssl format
keytool -keystore /etc/tomcat7/keystore  -exportcert -alias tomcat | \
 openssl x509 -inform der >tomcat.pem

Enter keystore password:  tomcat

# add the pem to the trusted certs
cp tomcat.pem /etc/ssl/certs
cat tomcat.pem  >>/etc/ssl/certs/ca-certificates
```

If any of those constraints is not satisfied, 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 such as IMAP or sieve. Most of the time, this happens as a consequence to the problem described above. To troubleshoot these issues, one should be tailing `cas.log`, pam logs and sogo logs.

Currently, SOGo will ask for a CAS ticket using the same CAS service name for both IMAP and Sieve. **When CASifying sieve, this means that the `-s` parameter of `pam_cas`should be the same for both IMAP and Sieve**, otherwise the CAS server will complain:

```
ERROR [org.jasig.cas.CentralAuthenticationServiceImpl] - ServiceTicket
[ST-31740-hoV1brhhwMNfnBkSMVUw-ocas] with service [imap://myimapserver
does not match supplied service [sieve://mysieveserver:2000]
```

Finally, when using imapproxy to speed up the imap accesses, the SOGoIMAPCASServiceName should be set to the actual imap service name expected by pam_cas, otherwise it will fail to authenticate incoming connection properly.

Authenticating using SAML2

SOGo natively supports SAML2 authentication. Please refer to the documentation of your identity provider and the SAML2 configuration keys that are listed above for proper setup. Once a SOGo instance is configured properly, the metadata for that instance can be retrieved from `http://`

`<hostname>/SOGo/saml2-metadata` for registration with the identity provider. SOGo will dynamically generate the metadata based on the SOGoSAML2CertificateLocation's content and the SOGo server name.

When using SimpleSAMLphp, make sure the convert OID to names by modifying your `metadata/saml20-idp-hosted.php` to contain something like this:

```
        'attributes.NameFormat' => 'urn:oasis:names:tc:SAML:2.0:attrname-
format:uri',
        'authproc' => array(
                100 => array('class' => 'core:AttributeMap', 'oid2name'),
        ),
```

If you want to test the IdP-initiated logout using SimpleSAMLphp, you can do so by opening the following URL:

```
https://idp.example.org/simplesaml/saml2/idp/SingleLogoutService.php?
ReturnTo=www.sogo.nu
```

In order to relay authentication information to your IMAP server and if you make use of the CrudeSAML SASL plugin, you need to make sure that *NGImap4AuthMechanism* is configured to use the `SAML` mechanism. If you make use of the CrudeSAML PAM plugin, this value may be left empty.

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 file (`/etc/sogo/sogo.conf`) to reflect your database settings:

```
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";
```

The following table describes the parameters that were set:

S	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`.
S	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`.
S	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`.
S	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
```

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, crypt, 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 default algorithm used for password encryption when changing passwords. Possible values are: `none`, `plain`, `crypt`, `md5`, `md5-crypt`, `smd5`, `cram-md5`, `ldap-md5`, and `sha`, `sha256`, `sha512` and its ssha (e.g. `ssha` or `ssha256`) variants. Passwords can have the scheme prepended in the form `{scheme}encryptedPass`. If no scheme is given, *userPasswordAlgorithm* is used instead. The schemes listed above follow the algorithms described in

	http://wiki.dovecot.org/Authentication/PasswordSchemes. Note that `cram-md5` is not actually using cram-md5 (due to the lack of challenge-response mechanism), its just saving the intermediate MD5 context as Dovecot stores in its database.
prependPasswordScheme	The default behaviour is to store newly set passwords without the scheme (default: `NO`). This can be overridden by setting to `YES` and will result in passwords stored as `{scheme}encryptedPass`.
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). Note that if set to `NO`, autocompletion will not work for entries in this source and thus, free-busy lookups.
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)	Aan array of fields that specifies the column names that hold additional email addresses (beside the `mail` column) for each user.
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 `c_uid` when unset).
SieveHostFieldName (optional)	The field that returns the Sieve hostname for the user.
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)	The value of this field is the maximum number of concurrent events to which a resource can be part of at any point in time. If this is set to `0`, or if the attribute is missing, it means no limit and the resource will always be marked as free. If set to `-1`, no limit is imposed but the resource will be marked as busy the first time it is booked. If greater than 0, the

		resource will get marked as busy once it reaches the value.
	DomainFieldName (optional)	If set, SOGo will use the value of that field as the domain associated to the user. See the *Multi-domains Configuration* section in this document for more information.

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.

The following table describes the related parameters.

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`.

D	SOGoSMTPAuthenticationType	Activate SMTP authentication and specifies which type is in use. Current, only `PLAIN` is supported and other values will be ignored.
S	WOSendMail	The path of the sendmail binary. Defaults to `/usr/lib/sendmail`.
D	SOGoForceExternalLoginWithEmail	Parameter used to specify if, when logging in to the SMTP 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.

IMAP Server Configuration

SOGo requires an IMAP server in order to let users consult their email messages, manage their folders and more.

The following table describes the related parameters.

U	SOGoDraftsFolderName	Parameter used to set the IMAP folder name used to store drafts messages. Defaults to `Drafts` when unset. Use a `/` as a hierarchy separator if referring to an IMAP subfolder. For example: `INBOX/Drafts`.
U	SOGoSentFolderName	Parameter used to set the IMAP folder name used to store sent messages. Defaults to `Sent` when unset. Use a `/` as a hierarchy separator if referring to an IMAP subfolder. For example: `INBOX/Sent`.
U	SOGoTrashFolderName	Parameter used to set the IMAP folder name used to store deleted messages. Defaults to `Trash` when unset. Use a `/` as a hierarchy separator if referring to an IMAP subfolder. For example: `INBOX/Trash`.
U	SOGoJunkFolderName	Parameter used to set the IMAP folder name used to store junk messages. Defaults to `Junk` when unset.

		Use a `/` as a hierarchy separator if referring to an IMAP subfolder. For example: `INBOX/Junk`. Also see the SOGoMailJunkSettings for more options regarding junk/not-junk actions.
D	SOGoIMAPCASServiceName	Parameter used to set the CAS service name (URL) of the imap service. This is useful if SOGo is connecting to the IMAP service through a proxy. When using `pam_cas`, this parameter should be set to the same value as the `-s` argument of the imap pam service.
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` ▪ `sieve://localhost:2000/?tls=YES` Note that TLS is supported but SSL is not.
D	SOGoSieveFolderEncoding	Parameter used to specify which encoding is used for IMAP folder names in Sieve filters. Defaults to `UTF-7`. The other possible value is `UTF-8`.
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	SOGoForceExternalLoginWithEmail	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. If you change this value, you must also modify the daily cronjob `sogo-tmpwatch`. Defaults to `/var/spool/sogo`.
S	NGMimeBuildMimeTempDirectory	Parameter used to set the path where temporary files will be stored by SOPE when dealing with MIME messages. Defaults to `/tmp`.
S	NGImap4DisableIMAP4Pooling	Disables IMAP pooling when set to `YES`. Enable pooling by setting to `NO` or using a caching proxy like imapproxy. The default value is `YES`.
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 `/`.
S	NGImap4AuthMechanism	Trigger the use of the IMAP `AUTHENTICATE` command with the specified SASL mechanism. Please note that feature might be limited at this time.
D	NGImap4ConnectionGroupIdPrefix	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.

		When enabling this parameter, one must also enable the associated cronjob in `/etc/cron.d/sogo` in order to activate automatic vacation message expiration. See the *Cronjob — Vacation messages expiration* section below for details.
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	SOGoForwardConstraints	Parameter used to set constraints on possible addresses used when automatically forwarding mails. When set to `0` (default), no constraint is enforced. When set to `1`, only internal domains can be used. When set to `2`, only external domains can be used.
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	SOGoCalendarCategoriesColors	Parameter used to define the colour of categories. This parameter is a dictionary of category name/color.

		Defaults to `#F0F0F0` for all categories when unset.
U	SOGoCalendarEventsDefaultClassification	Parameter used to defined the default classification for new events. Possible values are: ▪ `PUBLIC` ▪ `CONFIDENTIAL` ▪ `PRIVATE` Defaults to `PUBLIC` when unset.
U	SOGoCalendarTasksDefaultClassification	Parameter used to defined the default classification for new tasks. Possible values are: ▪ `PUBLIC` ▪ `CONFIDENTIAL` ▪ `PRIVATE` Defaults to `PUBLIC` when unset.
U	SOGoCalendarDefaultReminder	Parameter used to defined a default reminder for new events. Possible values are: ▪ `-PT5M` ▪ `-PT10M` ▪ `-PT15M` ▪ `-PT30M` ▪ `-PT45M` ▪ `-PT1H` ▪ `-PT2H` ▪ `-PT5H` ▪ `-PT15H` ▪ `-P1D` ▪ `-P2D` ▪ `-P1W`
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	SOGoMailCustomFullName	The string to use as full name when composing an email, if *SOGoMailCustomFromEnabled* is set in the user's domain defaults. When unset, the full name specified in the user sources for the user is used instead.
U	SOGoMailCustomEmail	The string to use as email address when composing an email, if *SOGoMailCustomFromEnabled* is set in the user's domain defaults. When unset, the email specified in the user sources for the user is used instead.
U	SOGoMailReplyPlacement	The reply placement with respect to the quoted message. Possible values are: ▪ `above` ▪ `below` Defaults to `below`.
U	SOGoMailReplyTo	The email address to use in the `reply-to` header field when the user sends a message. Ignored when empty.
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. Defaults to `NO` when unset. For this feature to work correctly, one must also set the *OCSEMailAlarmsFolderURL* parameter and enable the associated cronjob. See the *Cronjob — 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 a 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. The paths are relative to the `WebServerResources` directory, which is usually found under `/usr/lib/GNUstep/SOGo/.`
D	SOGoMailCustomFromEnabled	Parameter used to allow or not users to specify custom "From" addresses from SOGo's preferences panel. Defaults to `NO` when unset.
D	SOGoSubscriptionFolderFormat	Parameter used to set the default formatting of a subscription folder name. Available variables are: ▪ `%{FolderName}` ▪ `%{UserName}` ▪ `%{Email}` Defaults to `%{FolderName} (%{UserName} < %{Email}>)` when unset.
D	SOGoUIxAdditionalPreferences	Parameter used to enable an extra preferences tab using the content of the template named `UIxAdditionalPreferences.wox`. This template should be put under `~sogo/GNUstep/Library/SOGo/Templates/PreferencesUI/`.
D	SOGoMailJunkSettings	Parameter used to enable email junk settings. The value is a dictionary and the follow keys are supported: `vendor` (which must be set to "generic" for now), `junkEmailAddress` which sets the email address to whom SOGo will send junk mails to, `notJunkEmailAddress` which sets the email address to whome SOGo will send non-junk mails to and `limit`, which is an integer value and sets the maximum number of mails that will be attached to a junk/not junk report sent by SOGo. Example: `SOGoMailJunkSettings = { vendor = "generic"; junkEmailAddress = "spam@foo.com"; notJunkEmailAddress = "ham@foo.com"; limit = 10; };`
D	SOGoMailKeepDraftsAfterSend	Parameter used to keep mails in the drafts folder once they have been sent by SOGo. Defaults to `NO` when unset.

SOGo Configuration Summary

The complete SOGo configuration file `/etc/sogo/sogo.conf` should look like this:

```
{
    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;
    SOGoTimeZone = America/Montreal;
    SOGoMailDomain = acme.com;
    SOGoIMAPServer = localhost;
    SOGoDraftsFolderName = Drafts;
    SOGoSentFolderName = Sent;
    SOGoTrashFolderName = Trash;
    SOGoJunkFolderName = Junk;
    SOGoMailingMechanism = smtp;
    SOGoSMTPServer = 127.0.0.1;
    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;
        }
    );
}
```

Multi-domains Configuration

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

```
{
...
    domains = {
        acme.com = {
            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.com = {
            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 enable 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. If you prefer having the domain names listed, just use these as keys for the the `domains` dictionary.
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 thechkconfigcommand 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.

A commented out example should have been installed in `/etc/cron.d/sogo`, to enable it, simply uncomment it.

As a reference, the *cronjob* should de defined like this:

```
* * * * *     /usr/sbin/sogo-ealarms-notify
```

If your mail server requires use of SMTP AUTH, specify a credential file using `-p /path/to/credFile`. This file should contain the username and password, separated by a colon (`username:password`)

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 run a *cronjob* under the "sogo" user.

A commented out example should have been installed in `/etc/cron.d/sogo`. To work correctly this tool must login as an administrative user on the sieve server. The required credentials must be specified in a file by using `-p /path/to/credFile`. This file should contain the username and password, separated by a colon (`username:password`).

The *cronjob* should look like this:

```
0 0 * * * sogo /usr/sbin/sogo-tool expire-autoreply -p /etc/sogo/sieve.creds
```

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.

Microsoft Enterprise ActiveSync

SOGo supports the Microsoft ActiveSync protocol.

ActiveSync clients can fully synchronize contacts, emails, events and tasks with SOGo. Freebusy and GAL lookups are also supported, as well as "Smart reply" and "Smart forward" operations.

To enable Microsoft ActiveSync support in SOGo, you must install the required packages.

```
yum install sogo-activesync libwbxml
```

Once installed, simply uncomment the following lines from your SOGo Apache configuration:

```
ProxyPass /Microsoft-Server-ActiveSync \
        http://127.0.0.1:20000/SOGo/Microsoft-Server-ActiveSync \
        retry=60 connectiontimeout=5 timeout=360
```

Restart Apache afterwards.

The following additional parameters only affect SOGo when using ActiveSync:

S	SOGoMaximumPingInterval	Parameter used to set the maximum amount of time, in seconds, SOGo will wait before replying to a Ping command. If not set, it defaults to `10` seconds.
S	SOGoMaximumSyncInterval	Parameter used to set the maximum amount of time, in seconds, SOGo will wait before replying to a Sync command. If not set, it defaults to `30` seconds.
S	SOGoInternalSyncInterval	Parameter used to set the maximum amount of time, in seconds, SOGo will wait before doing an internal check for data changes (add, delete, and update). This parameter must be lower than *SOGoMaximumSyncInterval* and *SOGoMaximumPingInterval*. If not set, it defaults to `10` seconds.
S	SOGoMaximumSyncResponseSize	Parameter used to overwrite the maximum response size during a Sync operation. The value is in kilobytes. Setting this to 512 means the response size will be of 524288 bytes or less. Note that if you set the value too low and a

		mail message (or any other object) surpasses it, it will still be synced but only this item will be. Defaults to `0`, which means no overwrite is performed.
S	SOGoMaximumSyncWindowSize	Parameter used to overwrite the maximum number of items returned during a Sync operation. Defaults to `0`, which means no overwrite is performed. Setting this parameter to a value greater than `512` will have unexpected behaviour with various ActiveSync clients.
S	SOGoEASDebugEnabled	Parameter used to log the complete request and response of every single EAS command. Defaults to `NO`, which means no logging is performed.

Please be aware of the following limitations:

- Outlook 2013 does not search the GAL. One possible alternative solution is to configure Outlook to use a LDAP server (over SSL) with authentication. Outlook 2013 also does not seem to support multiple address books over ActiveSync.
- To successfully synchronize Outlook email categories, a corresponding mail label (Preferences→Mail Options) has to be created manually in SOGo for each label defined in Outlook. The name in SOGo and in Outlook must be identical.
- Make sure you do not use a self-signed certificate. While this will work, Outlook will work intermittently as it will raise popups for certificate validation, sometimes in background, preventing the user to see the warning and thus, preventing any synchronization to happen.
- ActiveSync clients keep connections open for a while. Each connection will grab a hold on a sogod process so you will need a lot of processes to handle many clients. Make sure you tune your SOGo server when having lots of ActiveSync clients.
- Repetitive events with occurrences exceptions are currently not supported.
- Outlook 2013 Autodiscovery is currently not supported.
- Outlook 2013 freebusy lookups are supported using the Internet Free/Busy feature of Outlook 2013. Please see http://support.microsoft.com/kb/291621 for configuration instructions. On the SOGo side, *SOGoEnablePublicAccess* must be set to `YES` and the URL to use must be of the following format: `http://<hostname>/SOGo/dav/public/%NAME%/freebusy.ifb`
- If you have very large mail folders (thousands of messages), you will need to adjust the word size of your IMAP server. In Dovecot, the parameter to increase is "imap_max_line_length" while under Cyrus IMAP Server, the parameter is "maxword". We suggest a buffer of 2MB.

In order to use the SOGo ActiveSync support code in production environments, you need to get a proper usage license from Microsoft. Please contact them directly to negotiate the fees associated to your user base.

To contact Microsoft, please visit:

http://www.microsoft.com/en-us/legal/intellectualproperty/ and send an email to iplicreq@microsoft.com

Inverse inc. provides this software for free, but is not responsible for anything related to its usage.

Microsoft Enterprise ActiveSync Tuning

First of all, it is important to know that most EAS devices will keep HTTP connections open to SOGo (and thus, Apache) for a long time. This is required for "push" to work properly. Connections can stay open for up to one hour, or 3600 seconds.

The first parameter to check is related to Apache's proxying to SOGo:

```
ProxyPass /Microsoft-Server-ActiveSync \
 http://127.0.0.1:20000/SOGo/Microsoft-Server-ActiveSync \
 retry=60 connectiontimeout=5 timeout=360
```

The above line sets a timeout for up to 360 seconds, or 6 minutes. If you want to let EAS clients keep their HTTP connections open for up to an hour, you must change the timeout parameter and set it to 3600.

If you change this value, the WOWatchDogRequestTimeout parameter must be changed accordingly in SOGo's configuration file (/etc/sogo/sogo.conf). By default, a SOGo child process is allowed to handle a request that can take up to 10 minutes before it gets killed by its parent process. When using EAS "push", the client expects to keep its connection open for up to one hour - so the WOWatchDogRequestTimeout, which is set in minutes, must be adjusted accordingly.

EAS clients will keep HTTP connections open for a long time during these two EAS commands: Ping and Sync. By default, SOGo will prevent EAS clients from keeping connections for a long time. This is to avoid the situation where all SOGo child processes would be monopolized by EAS clients - rendering the SOGo web interface or DAV interface unavailable. The default SOGo behavior is thus similar to disable EAS push entirely.

Two SOGo configuration parameters are available to modify this behavior: SOGoMaximumPingInterval (set by default to 10 seconds) and SOGoMaximumSyncInterval (set by default to 30 seconds). If you want connection to stay open for up to one hour, you should set these slightly under 3600 seconds (say 3540 - or 59 minutes). During a long-lived HTTP connection, the SOGo child process will perform internal polling to detect changes and return them to the EAS client if any changes are found. The parameter used to control this is SOGoInternalSyncInterval. By default, polling is done every 10 seconds. This might generate too much load on large-scale system.

The last configuration parameter to adjust is WOWorkersCount - which sets the number of SOGo child process that will be used to handle requests. You should have at least one child per EAS device configured to use "push". You must also have more children than you have EAS devices configured to use "push" - in order to handle normal SOGo requests to its Web or DAV interfaces.

Here are some usage examples for EAS devices using "push". In all cases, the Apache timeout is set to 3600 and the WOWatchDogRequestTimeout parameter is set to 60.

Example 1 - 100 users, 10 EAS devices:

```
WOWorkersCount = 15;
SOGoMaximumPingInterval = 3540;
SOGoMaximumSyncInterval = 3540;
SOGoInternalSyncInterval = 30;
```

Example 2 - 1000 users, 100 EAS devices:

```
WOWorkersCount = 120;
SOGoMaximumPingInterval = 3540;
SOGoMaximumSyncInterval = 3540;
SOGoInternalSyncInterval = 60;
```

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://localhost/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

Apple iCal 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

Since Mac OS X 10.6 (Snow Leopard), Apple AddressBook 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 an Apple iOS device (version 4.0 and later).

Microsoft ActiveSync / Mobile Devices

You can synchronize contacts, emails, events and tasks from SOGo with any mobile devices that support Microsoft ActiveSync. Microsoft Outlook 2013 is also supported.

The Microsoft ActiveSync server URL is generally something like: `http://localhost/Microsoft-Server-ActiveSync`.

Upgrading

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

2.3.1
The SOGoCalendarDefaultCategoryColor default has been removed. If you want to customize the color of calendar categories, use the SOGoCalendarCategories and SOGoCalendarCategoriesColors defaults.
2.3.0
Run the shell script `sql-update-2.2.17_to_2.3.0.sh` or `sql-update-2.2.17_to_2.3.0-mysql.sh` (if you use MySQL). This will grow the "participant states" field of calendar quick tables to a larger size and add the the "c_description" column to calendar quick tables. Moreover, if you are using a multi-domain configuration, make sure the keys for your domains match the email domains you have defined.
2.2.8
The configuration configuration parameters were renamed: ▪ *SOGoMailMessageCheck* was replaced with *SOGoRefreshViewCheck* ▪ *SOGoMailPollingIntervals* was replaced with *SOGoRefreshViewIntervals* Backward compatibility is in place for the old preferences values.
2.0.5

The configuration is now stored in /etc/sogo/sogo.conf. Perform the following commands as root to migrate your previous user defaults:
 	
`install -d -m 750 -o sogo -g sogo /etc/sogo`
`sudo -u sogo sogo-tool dump-defaults > /etc/sogo/sogo.conf`
`chown root:sogo /etc/sogo/sogo.conf`
`chmod 640 /etc/sogo/sogo.conf`
`sudo -u sogo mv ~/GNUstep/Defaults/.GNUstepDefaults \`
`~/GNUstep/Defaults/GNUstepDefaults.old`

2.0.4
The parameter *SOGoForceIMAPLoginWithEmail* is now deprecated and is replaced by *SOGoForceExternalLoginWithEmail* (which extends the functionality to SMTP authentication). Update your configuration if you use this parameter.

The sogo user is now a system user. For new installs, this means that `su - sogo` won't work anymore. Please use `sudo -u sogo <cmd>` instead. If used in scripts from cronjobs, `requiretty` must be disabled in sudoers.
1.3.17
Run the shell script `sql-update-1.3.16_to_1.3.17.sh` or `sql-update-1.3.16_to_1.3.17-mysql.sh` (if you use MySQL). This will grow the "cycle info" field of calendar tables to a larger size.
1.3.12
Once you have updated and restarted SOGo, run the shell script `sql-update-1.3.11_to_1.3.12.sh` or `sql-update-1.3.11_to_1.3.12-mysql.sh` (if you use MySQL). This will grow the "content" field of calendar and addressbook tables to a larger size and fix the primary key of the session table.
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.

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://lists.inverse.ca/sogo

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 to help organizations deploy the solution and migrate from their legacy systems.