

Kolab 2 - Free Software Groupware Project

Architecture Paper Version 1.9.9+CVS Draft

 *erfrakon*

Erlewein, Frank, Konold & Partner

Kolab 2 - Free Software Groupware Project: Architecture Paper Version 1.9.9+CVS Draft

by  and Erlewein, Frank, Konold & Partner

Published October 18th, 2004 + CVS

This documentation was written in SGML using the DocBook DTD. HTML and Postscript output is generated automatically and depends on the tools used.

Windows XP®, Windows NT®, Microsoft Exchange® and Microsoft Outlook® are registered trademarks of Microsoft Corporation Inc. Toltec Connector™ is a trademark of Radley Network Technologies CC. HotSync® is a registered trademark of Palm Inc.. K Desktop Environment™ and KDE™ are trademarks of the KDE e.V.

All other herein mentioned trademarks belong to their respective owners. The use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Finally, the authors of this book are not liable for any errors found as well as anything that may cause a fault. However, if that does occur, please notify the authors that corrections can be made. Furthermore, the reader must also agree to use the information in this book at his/her own risk and relinquish the authors, from any mistakes due to this book. If not, please stop reading now.

BECAUSE THE CONTENT IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE CONTENT, TO THE EXTENT PERMITTED BY APPLICABLE LAW. THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE CONTENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK OF USE OF THE CONTENT IS WITH YOU. SHOULD THE CONTENT PROVE FAULTY, INACCURATE, OR OTHERWISE UNACCEPTABLE YOU ASSUME THE COST OF ALL NECESSARY REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MIRROR AND/OR REDISTRIBUTE THE CONTENT AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE CONTENT, EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Revision History

Revision 1.0	September 10th 2002
Revision 1.0.1	September 19th 2002
Revision 1.0.2	January 13th 2003
Revision 1.0.3	January 28th 2003
Revision 1.1	February 20th 2003
Revision 1.9	March 16th 2004
Revision 1.9.1	April 21th 2004
Revision 1.9.2	October 3rd 2004
Revision 1.9.9	October 18th, 2004

Kolab 2 XML Storage format. Language improvements and updates to Kolab2.
Revision CVS \$Revision: 1.19 \$ \$Date: 2006/09/21 13:10:38 \$
Added changes in ldap structure between Server 2.0 and 2.1.

Table of Contents

1. Kolab 2 Free Software Groupware Overview	1
1.1. Email Functionality	1
1.2. Contacts	2
1.3. Address Book	2
1.4. Calendar Entries	2
1.5. Notes	3
1.6. Task Lists	3
1.7. Shared Resources	3
1.8. Print Services	3
1.9. Palm PDA Synchronization	3
2. Communication between Client and Server	5
2.1. Protocols	5
2.2. Used File Format Standards	5
2.3. Sending Email	5
2.4. Receiving Email	6
2.5. Vacation Functionality	7
2.6. Contacts	7
2.7. Address Book	7
2.8. Groupware Calendar	7
2.9. Automatic rules based handling of invitations	10
2.10. Extended Freebusy lists	12
2.11. Notes	12
2.12. Task Lists	12
2.13. Management of Shared Resources	12
2.14. Access Control and Multiple Identities	13
2.15. Remarks	15
3. The Communication between the Clients	16
3.1. Group Event Notifications	16
3.1.1. A Simple Calendar Event	16
3.1.2. Modify an Existing Calendar Event	17
3.1.3. Free-Busy Lists	18
3.1.4. Using Free-Busy Lists	19
3.2. Message Receive Confirmation	19
3.3. Sending Notes	19
3.4. Attached Business Cards	21
3.5. Multi-User Task Lists	21
3.6. HotSync PDA Synchronization	22
4. Kolab Server Components	23
4.1. OpenLDAP2 Directory Server	23
4.1.1. LDAP - Server requirements:	23
4.1.2. Top Level LDAP Structure	24
4.1.3. LDAP Attributes	25
4.1.4. LDAP Object Classes	31
4.1.5. LDAP OID Numbers	33
4.1.6. LDAP Access Control Lists	33

4.1.7. LDAP Business Card.....	35
4.1.8. LDAP-Changes for Kolab Server 2.1	36
4.2. Postfix Mail Server.....	39
4.2.1. LDAP Connection	40
4.3. Antivirus and Antispam Daemon.....	40
4.3.1. Antivirus - Amavis Daemon.....	41
4.3.2. Antispam - Spamassassin	42
4.3.3. Free Software Antivirus Scan Engine ClamAV.....	43
4.3.4. Mandatory SMTP Email Addresses	46
4.4. Cyrus IMAP Daemon.....	46
4.4.1. LDAP Connectivity	47
4.4.2. Cyrus Sieve.....	47
4.4.3. Access Control Lists.....	47
4.4.4. Cyrus Quota.....	50
4.5. ProFTP Daemon.....	52
4.6. Kolab Server Backup Strategy	52
4.6.1. Backup of IMAP Store.....	52
4.6.2. Backup of LDAP Directory.....	54
4.7. Administrator User Interface.....	54
4.8. Multi Location Setup	55
4.9. Maintenance of Kolab server	56
4.9.1. Installation of Kolab server	56
4.9.2. Upgrade of Kolab server.....	56
4.9.3. Changing Manager/calendar/nobody password	56
5. Windows Kolab Clients	58
5.1. Language dependencies	59
6. The KDE Client.....	60
6.1. Language dependencies	60
A. The KDE client File Formats.....	61
A.1. Calendar Event.....	61
A.2. Note	62
A.3. Contact	62
A.4. Task Lists.....	63
A.5. Free-Busy Lists	64
B. Legacy File Formats.....	66
B.1. Calendar Event	66
B.2. Note	67
B.3. Contact.....	68
B.4. Task.....	69

List of Tables

3-1. Calendar Event	16
3-2. Maintaining the Free-Busy List.....	18
3-3. Using the Free-Busy List.....	19
3-4. Sharing a Note	20
3-5. Sharing a Business Card.....	21
3-6. Assigning Tasks to Other Users	21

Chapter 1. Kolab 2 Free Software Groupware Overview

The Kolab Groupware is a highly scalable and secure solution and strictly based on a client/server architecture. The server part of the groupware solution, herein referred to as "Kolab Server" runs on GNU/Linux, namely on a current distribution out of Debian, Red Hat or SUSE in addition to FreeBSD. Most testing was done on Debian and SUSE.

We minimized the usage of Linux distribution dependencies in order to gain maximum portability. In order to be able to apply readily available security updates in a timely manner employment of a well-maintained GNU/Linux distribution like the above mentioned seems desirable. The goal is that standard security updates as provided by these vendors can be installed without affecting the Kolab server's functionality.

The Kolab server is distributed as OpenPKG packages which allows for a single source for our many target platforms.

It is not the scope of this architecture to integrate with the underlying operating system in such a way that the administration of the operating system is accomplished with it.

The free software groupware client application is developed as a native KDE 3 application. Microsoft Windows XP based clients access the Kolab groupware server using Microsoft Outlook 2002 with the Toltec Connector Plugin installed (reference platform). A high-level description of the complete free software groupware solution, seen from the users point of view, follows. The list was created with existing proprietary groupware products in mind, namely Microsoft Exchange Server with Outlook 2002 clients on Windows NT.

1.1. Email Functionality

The common internet email functionality is provided:

- sending email via SMTP over TLS and receive it via IMAP over TLS (preferably) or plain SMTP and IMAP (alternatively for backward compatibility reasons)
- support for strong cryptography for email bodies and attachments in addition to the security features provided by the transmission protocol (which can be found in the former project "Ägypten")
- optional support of message receive confirmations (the user gets prompted for an acknowledgment)
- adding priorities to emails (importance header)
- vacation message functionality
- email forwarding functionality

- receiving and sending business cards along with email

Every standard-conforming mail user agent (MUA) and web browser can be used to access the Kolab server for this functionality.

1.2. Contacts

The client application maintains a private address book ("Contacts", German: "Kontakte"). Note that this is very different from a global address book. A users private address book is stored on the Kolab server in IMAP folders. A bidirectional vCard interface is provided. In consequence, vCards received as email attachments can be easily added to a users contacts.

In Kolab 2 the newly introduced cross-platform XML-storage format allows a server side personal addressbook. This is interchangeable between Kolab clients including Microsoft Outlook with the Toltec Connector.

In Kolab 2 a user may have multiple contact folders. One is marked as the default contact folder though. Last but not least contact folders can be directly shared between users using IMAP.

1.3. Address Book

A global shared address book is available independently from the above mentioned contacts. Address book entries are maintained inside a LDAP directory on the Kolab server and can be exported to vCard and XML format. The necessary conversion is done by the Kolab client application.

A Kolab user or Kolab maintainer can change the LDAP based personal contact information by using the Kolab web administration interface.

1.4. Calendar Entries

A user can have private calendar events. Although these events are visible for other users using published free-busy lists, the exact event information is hidden from them. They only see that a given private event of another user exists. Private events are saved on the Kolab server in an IMAP folder. Users can schedule group events and invite other users. An existing group event can be modified by the user who originally created it. Examples for group events are:

- Group meetings
- Conferences

Group events are saved like private calendar entries on the Kolab server and differ only by their attributes and access permissions. When creating a new group event, one can check the availability of the desired attendees. We refer to this later as "publishing and checking free-busy lists".

In addition Kolab 2 allows for multiple calendar folders for every Kolab user and knows about group calendars shared directly via IMAP. There is always one default calendar folder for every Kolab account.

1.5. Notes

A user may take private notes which are then stored on the Kolab server. Notes can easily be mailed to other users and shared with them, as copies in addition to sharing them via IMAP.

1.6. Task Lists

Users can maintain task lists with priorities. These are saved on the Kolab server. Items on the task list can be assigned to different users and added to their task lists as well, as they receive and accept them. Task lists are private if no items are meant for other users.

Task list may be shared using IMAP folders

1.7. Shared Resources

Shared resources (e. g. meeting rooms or pool cars) can be managed in the same way as regular group calendar events.

Basically this means that shared resources are invited to events like any Kolab user.

In addition Kolab 2 implements automatic handling of shared resources with resource dependent configurable policies.

1.8. Print Services

Emails, calendar events, task lists, contacts and notes are easily printable by the client application.

1.9. Palm PDA Synchronization

The contacts, calendar events, notes, and task lists can be synchronized with a personal digital assistant (PDA) bidirectionally. The HotSync protocol is used and guarantees compatibility to a wide range of PDA devices. The reference platform is a 3Com Palm V running Palm OS v3.1.

Chapter 2. Communication between Client and Server

A technical description of the communication between KDE clients and the Kolab server follows.

2.1. Protocols

The protocols were selected with the following criteria in mind:

- proper standardization e.g. by the Internet Engineering Task Force (IETF, <http://www.ietf.org>)
- open standard in the sense that a Free Software implementation is available
- existing Free Software implementations must scale very well

In general simplicity is preferred over complexity. The limitation to a small set of well-established and widely used protocols is a major design goal.

This leads to the following protocols used in the project:

- Lightweight Directory Access Protocol Version 3 (LDAP, IETF RFC 2251)
- File Transfer Protocol (FTP, IETF RFC 765)
- Simple Mail Transfer Protocol (SMTP, IETF RFC 2821), SMTP over Secure Socket Layer / Transport Layer Security (SMTP over SSL/TLS, IETF RFC 2246)
- Internet Message Access Protocol (IMAP, IETF RFC 1730), IMAP over SSL/TLS (IETF RFC 2595)
- Post Office Protocol Version 3 (POP3, IETF RFC 1939), POP3 over SSL/TLS (IETF RFC 2595)
- Hyper Text Transfer Protocol (HTTP, IETF RFC 2616), HTTP over SSL/TLS (IETF RFC 2818)
- HotSync Protocol (this does not gather some of the above criteria but seems to be widely accepted and free implementations exist)

2.2. Used File Format Standards

- multi-part MIME email as standardized by the IETF
- iCalendar and vCard as standardized by the Internet Mail Consortium (<http://www.imc.org>)
- XML Kolab format as standardized by the Kolab community (<http://www.kolab.org/>)

2.3. Sending Email

The Kolab server runs the Postfix Mail Transfer Agent (MTA) and acts as SMTP relay for all users. However it is possible to decouple this functionality from the mailboxes and establish it on another server, for load-sharing and high availability purposes. Security is provided using two methods:

1. on the application level: Email can be encrypted using OpenPGP/GPG
2. on the transport level: the SMTP relay accepts SMTP over TLS connections

For compatibility reasons, plain SMTP is also supported.

3. on the transport level: for Kolab users we use sender authentication so that recipients can trust in the from-address of Kolab messages.

2.4. Receiving Email

Users read their email via TLS secured IMAP from the Kolab server. New user accounts are created on the Kolab server as LDAP based Kolab accounts (not as regular GNU/Linux accounts), using the mailbox name `user.<username>` below the `domain` hierarchie. The latter is a requirement resulting from the Cyrus IMAP daemon and prepares for future multi-domain capabilities.

The Postfix MTA on the groupware server delivers email for local users to the Cyrus IMAP Daemon via the Local Mail Transport Protocol (LMTP) to their respective IMAP mailboxes, which are implemented in a format similar to the `maildir` format (not traditional `mbox` style, every email is a regular GNU/Linux text file).

The emails are then further processed by the users client desktop application. Note that Cyrus maps the above mentioned mailbox `user.<username>` to a configurable directory tree within the Kolab servers filesystem. The email messages are transferred via IMAP synchronization to the client application using the disconnected IMAP capabilities of the client. This synchronization functionality also provides the required offline functionality.

The messages are identified via a unique IMAP id. Optionally, users can physically receive their email messages via POP3 over SSL from the groupware server and the further processing is done locally on the client. For compatibility reasons, plain IMAP and unencrypted POP3 message transfers are also supported.

POP3 capabilities are a mandatory requirement for proper functioning of the Toltec Connector

Note that at this point every SMTP and IMAP capable email Client can access a Kolab server's mailboxes and distribute email messages via the Kolab server, provided that the server IP address, the

username and the corresponding password are correct.

2.5. Vacation Functionality

The vacation functionality is configured by the user via a simple webinterface available via HTTPS provided by the Kolab server or built-in SIEVE capabilities in the client. The actual vacation functionality is implemented on the server using Sieve (IETF RFC 3028). A Free Software implementation of this scripting language for the Cyrus IMAP daemon is available.

Note that Sieve's capabilities reach far beyond a simple vacation functionality. Therefore we gain great flexibility for future extensions. The server side scripting is a sensitive area e.g. activities run contrary to server scalability and performance and therefore must be used with caution if we intend scalability to many thousands of users on commodity hardware.

2.6. Contacts

The client application stores contacts in IMAP subfolders marked using IMAP annotations as contact folders on the Kolab server. The actual entries are represented as multi-part MIME emails with included Kolab XML format address data as MIME parts. A single contact folder is marked as the default contact folder.

See the appendix for the exact file format.

2.7. Address Book

The global address book is stored inside a LDAP directory running on the Kolab server driven by OpenLDAP v2 (implementing LDAP protocol version 3). The clients access it by using the LDAP v3 protocol. The GUI on the client provides read only access to all address data stored inside the directory.

The global address book is maintained via the Kolab web administration GUI or any other readily available LDAP tool.

The LDAP scheme is designed in order to allow easy bidirectional conversion to vCard and the Kolab XML format.

The need to change someones own address book entry is considered to be rare and therefor a plain web interface is sufficient. A user can modify his own LDAP entry using a HTTPS protected web interface. The look and feel is analogous to the public phone book dialog.

2.8. Groupware Calendar

Calendar entries are stored in the users IMAP store. Like all normal folders the Calendar folders are subfolders of the users INBOX. For every account including normal users account, resource accounts and group accounts one of the potentially many calendar folders is marked as the default calendar using the IMAP annotation extension.

Calendar entries are represented as multi-part MIME messages with the information stored in the Kolab XML format. In addition to the XML part a client may choose to store data in an additional MIME part. E.g. the Toltec Connector chooses to use an additional TNEF MIME part.

The Kolab clients must preserve any unknown MIME part and any unknown XML tags.

The iCalendar format encapsulated in a MIME part used by the Kolab 1 KDE client is still available as an option for those not requiring full interoperability with MS Outlook / Toltec clients.

Please see the appendix for the exact file formats and examples.

Kolab 2 allows a more sophisticated concept for modelling group calendaring. While Kolab 1 basically build upon the abilities to upload and retrieve freebusy information and invitations via SMTP emails Kolab 2 allows the simultaneous sharing of group accounts across client platforms.

In order to be able to share the same account across users and platforms the cross platform Kolab XML format was developed. In addition the sharing is more sophisticated now with the ability to manage access with ACLs. ACLs are much superior in sharing passwords with regards to flexibility and security.

The concept of plain freebusy lists (fb) is enhanced with the possibility of extended freebusy lists (xfb). Extended free busy lists are not intended for the simple gantt diagrams in the Kolab clients but for extended reporting features using the Kolab webinterface.

The ability of storing group events just like personal events in the same IMAP sub folder on the Kolab server remains in case the group event is created by traditional invitations. These group events are shared via exchanging multi-part MIME emails using iCalendar format and have to be accepted and therewith put into their respective calendar folders by the attendees. The handling of acceptance status is a unique feature of the traditional group scheduling.

The individual free-busy lists consist of a compacted subsets of the iCalendar data covering a user definable time range into the future. This time range is a user specific setting in the LDAP directory.

Free-busy lists are published on the Kolab server and are available via https. The free-busy lists are referenced via URLs and are retrievable by all registered clients from the Kolab server. The free-busy lists are named after the rule `<username>.ifb` and follow the iCalendar file format.

Kolab 2 is able to create the free-busy lists on the server so that the clients currently don't need to publish free-busy lists anymore.

In the future clients may upload free-busy lists to the Kolab server using secure WebDAV. This will be useful for calendars not being stored on the Kolab server but which are still relevant for the free-busy status of a user. Such a usage model might for example be motivated by privacy concerns.

The free-busy list of a user is the consolidated superset of all free-busy relevant calendars.

The relevance of a Kolab folder is controlled by the presence of an IMAP annotation and its value. The IMAP annotation is named `/vendor/kolab/incidences-for`. The possible shared values (visible to everyone) are

- nobody

This Kolab folder is not relevant for the creation of freebusy and alarm data for anyone.

- admins

This Kolab folder is relevant for the creation of freebusy and alarm data for everyone with administration permissions on the folder (this is the default value, if the annotation isn't set).

- readers

This Kolab folder is relevant for the creation of freebusy and alarm data for everyone with read access.

The incidencesfor annotation are used for calendars and tasks.

This allows a Kolab user to organize the calendaring by having many calendars folders in one account.

Group calendars with the free-busy annotation (`/vendor/kolab/incidences-for`) set and having the value of this IMAP annotation (`value.shared`) set to `readers` are taken into account for the free-busy lists for all Kolab users having read permissions on this calendar folder. This "push" semantic within a Kolab server domain is intentional and requires some cooperation between users.

Conceptually this means that the administrators of the group account push the free-busy relevance on the Kolab users by means of granting them read access.

Support for insecure FTP and HTTP protocols to upload and download free-busy lists is discouraged for Kolab 2. Users are advised to use https instead.

For the future upload capability of private local free-busy information to the Kolab server Windows XP systems can use secure WEBDAV to upload freebusy lists like KDE clients are able to do this since Kolab 1. The WEBDAV functionality can be obtained for legacy Windows 2000 clients using the Novell Webdrive or other 3rd party software which allows to map a WEBDAV resource to a drive letter.

The Kolab client obtains the freebusy lists for all potential participants of a meeting and displays them in a gantt chart. This chart allows for the setting of the preview time e.g. next week, next month or any other period of time. In addition to normal users the gantt chart displays also aggregated rows representing either personal or central distribution lists. For later feature the client must be able to aggregate the individual freebusy lists of the individual members. The color representation shall indicate if the period of time is occupied is free for all members, some members or busy for everyone.

An analogous gantt chart interface is also implemented on the Kolab server using Apache, PHP and producing plain xhtml. An emphasis is here the ability to print the chart nicely. The Outlook Plugin must provide a mean to access this server based gantt chart by either integrating the functionality, embedding the html view from the server or as a last resort call the standard webbrowser with the appropriate URL.

The logic behind the calendar events and their handling is entirely done by the client applications. The server mainly acts as a network storage in this regard.

The webclient is extended to be able to act as a full Kolab groupware client and allows to view the freebusy lists of predefined groups in a calendar and in a gantt view. All clients must respect the privacy flag for individual calendar entries. It is possible to publish the resulting views on the Kolab server permanently. These views can be generated automatically and the access is controlled via LDAP based access controls. The apache webserver checks the access permissions before granting read-only access using the WEBDAV protocol.

2.9. Automatic rules based handling of invitations

The Kolab server implements a server side mechanism for handling of invitations. This feature is typically used to handle bookings of resources (e.g. a room, car, lcd projector etc.).

For the automatic handling of invitations access control lists are used which define the action to be taken depending on the senders address.

These ACLs are maintained in the LDAP directory. In case no ACL does match the request is queued in the INBOX of the recipient and the sender is informed about the fact that his request is pending.

There are five different kind of Kolab invitation policy ACLs. Typically these values are prefixed with the primary email address of a sender to whom they shall apply. An omitted prefix represents an anonymous policy and means "apply to everyone".

The sender is always informed via traditional accept and reject invitation messages using iCalendar format.

- ACT_ALWAYS_ACCEPT

In this case no conflict detection is taking place. If the organizer (sender of the invitation) changes the appointment this information is also propagated to the automatic account like traditional meeting requests. As a result the meeting is also moved in the automatic account. Whenever a change happens to the calendar of the automatic account the corresponding freebusy list is regenerated.

- ACT_ALWAYS_REJECT

Invitations from senders which match this ACL are immediately rejected. In case this is an anonymous ACL then this account will not accept any invitations from any sender.

- ACT_REJECT_IF_CONFLICTS

The invitation is accepted in case there is no conflict with other appointments and immediately rejected otherwise. The check happens by referring to the uptodate free-busy list.

- ACT_MANUAL_IF_CONFLICTS

The invitation is accepted in case there is no conflict with other appointments. In case a conflict is detected the invitation is queued in the INBOX. It is expected that a user is manually maintaining the INBOX regularly.

- ACT_MANUAL

This is the default case for every account. Invitations are just queued in the INBOX of the recipient. The recipient then decides upon the request manually.

In addition administrating access to resource accounts is controlled analogous to all other accounts with IMAP access control lists. This means access control lists are based on user accounts and group membership. The later allows for flexible handling of changing membership without the need to permanently maintain all individual access permissions in case a user changes.

The implementation of the webclient and the automatic handling of invitations is handled in such a way that this web component can but is not required to run on the primary Kolab server. From the point of view of the Kolab server the automatic account can be distributed like any other account in a multi-location setup. This allows for clean and secure separation while keeping scalability.

An automatic booking of a resource or the publishing of a meeting to a group is simply done by adding the corresponding account to the list of invitations e.g. <Audi.A4@erfrakon.de> or <group.kolab@erfrakon.de>.

2.10. Extended Freebusy lists

A server process based on the webclient extracts all information from the calendar folder of predefined accounts and aggregates them in extended freebusy list on demand. Extended freebusy lists are syntactically equivalent to traditional freebusy list but on one hand aggregate the calendar data of multiple accounts and on the other hand prepend the subject of the individual entries with the username followed with the original subject while honouring the private flag. The webclient talks to the Cyrus IMAP server with the credentials of the user.

The individual account must allow the extended freebusy list user access to its calendar folder otherwise the calendar folder of this user is simply skipped.

The server process generates a html gantt chart with a row for each individual user and an aggregated row with the summed up freebusy times. We reuse the gantt code from the Taskjuggler project.

2.11. Notes

Notes are stored on the Kolab server inside the users IMAP sub folder "Notes" (German: "Notizen"). Physically, they are represented as multi-part MIME emails with the actual note being a MIME part. See the appendix for the exact file format.

2.12. Task Lists

Task lists are stored on the Kolab server inside the users IMAP sub folder "Tasks" (German: "Aufgaben"). Physically, they are multi-part MIME emails with the actual task list data being a MIME part and following the iCalendar standard (vToDo, IETF RFC 2446). See the appendix for the exact file format.

2.13. Management of Shared Resources

The Kolab server assigns a dedicated IMAP identity to every shared resource. These identities do not differ technically from real users. Reserving a car or a room for example is just arranging a meeting with the shared resource's assigned IMAP user. Two modes of operation are supported:

1. manual mode: a real user monitors, a shared resources mailbox in addition to his own mailbox and accepts or declines events on behalf of the shared resource.
2. automatic mode: via Sieve scripting the resource mailbox is monitored; the scripting takes care of automatically publishing it's free-busy list and accepts or declines events on the basis of availability of the resource.

2.14. Access Control and Multiple Identities

Every user has its own personal account with full administrative privileges. This means that every user is allowed to manage the access permission of all folders belonging to him. The user may use these privileges to grant different levels of access to other named users or groups. In addition the user may use local distribution lists or central distribution lists instead of plain users as the entities to grant access permissions.

Local distribution list

Personal distribution list

A special kind of contact in the contact folder consisting of a concrete list of named users. This feature is already built into Outlook but needs to be implemented in the KDE Kontact client in a compatible manner. Kontact uses references to unique contact entries in the contact folders.

Central distribution list

Kolab maintains central distribution list with GroupOfNames LDAP objects. see also <http://www.alvestrand.no/objectid/2.5.6.9.html>

```
groupOfNames OBJECT-CLASS ::= {
    SUBCLASS OF { top }
    MUST CONTAIN { commonName | member }
    MAY CONTAIN { description |
        organizationName |
        organizationalUnitName |
        owner |
        seeAlso |
        businessCategory
    }
}
```

```
ID id-oc-groupOfNames  
}
```

The KDE Kolab client gets central distribution list from LDAP and is able to create a local copy in the address book which is then stored in the contacts folder.

Access permissions (read, write,) for distribution lists are identical to contacts. The access control lists (ACLs) are manipulated from the client via the IMAP ACL extensions for both the Windows client and the KDE client.

The user interface for manipulating the ACLs shall be available via right mouse button on a folder in the folder view. In addition this user interface must also be accessible via the menu and a dialog.

When accessing folders of other users the identity of the user does *not* change. The client must provide in the GUI a mean to configure the displayed prefix to folders not belonging to the current user. Internally the server uses the prefix "user". There are two kind of such folders.

1. Folders of other users with explicit access privileges granted either to the user or to a group where the user belongs to. Typically this involves folders in secretary/boss situations or small adhoc teams.
2. Shared folders not belonging to any specific user with explicit access privileges granted either to the user or to a group where the user belongs to. These global shared folders get typically administered from the maintainers and administrators of the server and serve a larger group of people for an extended period of time.

When working with any folder the default identity for sending messages (e.g. email, calendar invitations etc.) is the real. In addition the user has the choice of graphically choosing the effective identity before finally sending the message via SMTP to the server.

The KDE Kontact client already implements this functionality for email messages and is extended to provide the same functionality for all other activities like invitations, accepting meetings, tasks etc.

For groupware folders (calendars, contacts, tasks and notes) this has to be implemented correctly in addition to the ability to handle an arbitray number of groupware folders.

The Kolab Client knows about its standard folders for each identity for the groupware functionality. This is required in order to allow the client to know where to store accepted invitation, tasks, notes or contacts which it got via email and also in case a user has multiple personal calendar folders etc.

When granting access to a subfolder it is the responsibility of the user to allow sufficient permissions for the parent folders so that the subfolder is indeed accessible to the other users of the group. This is analogous to traditional unix filesystem permissions.

Outlook 2002 already knows about multiple accounts. Providing the required functionality for emails is handled via the selection of the account before sending the message. Using multiple accounts for invitations is possible though awkward with Outlook. The later is required if the secretary does not want to become the organizer of a the meeting she is organizing on behalf of the boss.

The Outlook Plugin could be extended to provide a better user interface for this szenario. In Kontakt this functionality is provided analogous to choosing the sender identity when sending email. The client must make sure that not only the SMTP sender address is adapted but also the correct content is generated.

An extension to the traditional SMTP message handling is the configurable Kolab feature to be able to verify the SMTP sending address of the sender against a whitelist. This whitelist is looked up in the LDAP directory for every sending request. The email address the user used to authenticate itself with the MTA (Postfix) ist an implicit entry in this whitelist. The MTA falls back to rewrite the sender address to this default address. This address is afterwards like all sender addresses subject to LDAP based canonification (e.g. konold@erfrakon.de is rewritten to martin.konold@erfrakon.de).

For this, we need a modified version of the Postfix mail server.

2.15. Remarks

The support for message receive confirmations and attaching business cards is a matter of the end to end communication between the clients.

The print services and the Palm OS (HotSync) Synchronization depend fully on the client installation and is as such not related to the client server communication.

Chapter 3. The Communication between the Clients

In principle all information is exchanged via multi-part MIME email messages between the KDE clients. The receiving user decides about every incoming event, note, task list, etc. and depending on the decision the KDE client application moves the email to the corresponding IMAP folder for further processing. Therefore the client application has means to detect the type of an incoming email and classify it into one category out of note, task list, contact, calendar event, and ordinary email.

The national language support is handled by the clients. UTF-8 encoding is used for special characters. Common ASCII characters are a subset of the UTF-8 encoding. Outlook and the KDE client are interoperable in this respect. This has been verified using german umlaut characters using Outlook 2000 on WindowsNT and the KDE client on Linux 2.4.

3.1. Group Event Notifications

Invitations to a group event are sent via a multi-part MIME email with iCalendar information included as a MIME part. The user can decide either to accept the event (and thereby import it into his own calendar) or decline it. When an event is declined, an according email is generated and sent back to the originator of the group event. When an event is accepted the event gets added to the users personal calendar and the users own free-busy list gets updated and later published.

There is a special mode that must be used by the KDE client when communicating with Microsoft Outlook. In this case calendar events are sent as plain MIME email with the iCalendar data being the body of the email. This non RFC-conformant mode is not recommended by the IETF but the only way to communicate with broken Windows clients. In fact the KDE client can send and receive calendar events in this format, in addition to it's own format as outlined above. We will refer to this mode of operation as "legacy support".

In order to be most compatible to real world szenarios this legacy mode will be the default.

The most important calendar use cases follow. Other cases can be derived from them and therefore are not included herein.

3.1.1. A Simple Calendar Event

Table 3-1. Calendar Event

Step	Action	Remarks
1	Originator sends Email with attached iCalendar to attendees	Calendar event gets saved in originator's own calendar
2	Attendee's receive the calendar event as Email in their INBOX (German "Posteingang")	Attendee has to open the email and is prompted to accept, conditionally accept or decline the event; no storage operation takes place so far
3a	Attendee A accepts the event and sends back a confirmation; attached is the slightly modified iCalendar (attributes used: DTSTAMP and ATTENDEE)	Calendar event gets saved in attendee A's calendar (subset of original event including participants; only originator has full information e.g. the state of accepts/declines)
3b	Originator gets email with the slightly modified iCalendar attached	Originator learns about the update and refreshes the event in his calendar after the users confirmation (mark attendee as having accepted)
3c	Attendee B declines the event and sends back a refusal; attached is the slightly modified iCalendar (attributes used: DTSTAMP and ATTENDEE)	Calendar event is not saved into attendee B's calendar
3d	Originator gets email with the slightly modified iCalendar attached	Originator learns about the update and refreshes the event in his calendar after the users confirmation (mark attendee as having declined)

3.1.2. Modify an Existing Calendar Event

Note that the whole group calendar system does not automatically keep consistency over different users. This is impossible because of the accept/decline choice and the fact that a calendar event can be privately modified.

3.1.2.1. Originator modifies Event

The originator may change any aspect of a calendar event after it was settled between the attendees. If for example a calendar event's duration gets modified, the above process (3.1.1) starts again. In the originator's calendar the state of the attendees therefore gets reset. By having a unique message ID it is guaranteed that the iCalendar matches again to the same calendar event. This guarantees that a calendar

event can be shifted without ending up to be created multiple times. When new attendees are added or some are deleted, the originator gets prompted on whether the notification (again a iCalendar entry) should be send to all people involved or only to those who are actually affected by the change.

3.1.2.2. Attendee modifies Event

An attendee can modify the time parameters of a calendar event which doesn't originate from himself. But then his calendar falls out of sync. The change will be overwritten if the originator sends an update. A warning concerning this is presented to the user.

3.1.3. Free-Busy Lists

KDE clients can optionally (and should) publish their own calendar summary information on the Kolab server. Using this information, other users can check for availability of desired attendees for a proposed calendar event before issuing the actual invitation. Therewith a meeting can be scheduled more efficiently and according to the attendee's free time slots.

A free-busy file follows the iCalendar format and contains only the calendar event time data for a given user. The user can choose how long into the future his free-busy information should be published on the Kolab server. On the Kolab server all free-busy data is stored and is accessible in different ways for legacy and KDE clients.

The KDE clients store their "Free-Busy" information on the Kolab server using Web-DAV for upload and HTTPS for download. The legacy clients use anonymous FTP to upload their free-busy information into the very same directory as the KDE clients do. Accessing the free-busy information is done via HTTP on the legacy clients.

In configurable intervals, the client application publishes free-busy data:

Table 3-2. Maintaining the Free-Busy List

Step	Action	Remarks
1	Get through all calendar data from "now" to a time T in the future (default: two months)	the KDE client holds most of it's calendar data in the process memory, so this does not necessarily mean an overhead
2	Compute a consolidated iCalendar based on the individual iCalendar data files	

Step	Action	Remarks
3	Publish the iCalendar free-busy list on the Kolab server	This is preferably done using Web-DAV via HTTPS; Windows clients use anonymous FTP and retrieve via HTTP (Kolab legacy mode)

3.1.4. Using Free-Busy Lists

When a user proposes a new calendar event, he chooses the participants and at the same time, transparently, the free-busy information of the according users is retrieved from the Kolab server for display during the creation of the event.

Table 3-3. Using the Free-Busy List

Step	Action	Remarks
1	the KDE client learns about attendees in a dialog	User enters a list of email addresses in the calendar event dialog
2	Display attendee's calendar schedule (only times) using a simple bar diagram and let the user choose a free time frame	Retrieve free-busy list of that user (filename is computable: <user>.vfb) by HTTPS or, in the windows case with HTTP (Kolab legacy mode)

3.2. Message Receive Confirmation

When a message with a request for receive confirmation arrives, the user is prompted whether a confirmation shall be send to the originator of the according email. The system is based on the corresponding IETF RFC standards and supports the following actions:

- disallow generation of the receive confirmation (privacy mode)
- allow generation of the receive confirmation (collaboration mode)
- the user on who's behalf the receive message is generated can assign a different email address to receive the confirmation

3.3. Sending Notes

Usually Notes contain only text and they are saved on the Kolab server in a special IMAP subfolder of the creating user. Notes can be categorized. German standard categories are:

- Favoriten
- Feiertag
- Festtagsgrüße
- Geschäftlich
- Geschenke
- Hauptkunde
- Ideen
- International
- Konkurrenz
- Persönlich
- Schlüsselpersonen
- Status
- Strategien
- Telefonanrufe
- Verschiedenes
- Wartet
- Wichtige Kontakte
- Zeit und Ausgaben
- Ziele
- Zulieferer

The list of categories can be edited by the user. New categories can be added, existing entries can be modified or deleted.

Notes are saved as multi-part MIME messages on the Kolab server. The first part of the email is of type text/plain and has the textual representation of the note. The second part of the message consists of the note with further attributes encoded as SMTP headers. For sending notes to Windows clients a special legacy format must be readable and writeable by the KDE client. See the appendix for the exact file format.

Table 3-4. Sharing a Note

Step	Action	Remarks
1	Forward note to another user	Note is sent as multi-part MIME email message (two parts - readable text)
2	As the recipient opens the email he can import the note into his own note folder	Note is moved into notes folder, so it's shared by creating a copy

3.4. Attached Business Cards

Users can exchange business cards (contacts or his own personal data). The card can be exchanged by email - thereby using the format VCARD inside a part in a multi-part MIME email (content type text/x-vcard). Business cards can contain categories (see above). Windows clients can attach OLE objects like MS-Word, Excel, PowerPoint or similar. The free software client does ignore OLE data, as it is not usable on non-Microsoft platforms. Attaching complete files or sending URL references is the preferred portable and more efficient way. For compatibility reasons, platform independence and severe security concerns the free software client does not make use of OLE objects.

Table 3-5. Sharing a Business Card

Step	Action	Remarks
1	Forward business card to another user	Note is sent as multi-part MIME email
2	As the recipient opens the email he can import the business card into his contacts	multi-part MIME email gets stored inside the users private contacts folder

3.5. Multi-User Task Lists

A task can be send to other users via a multi-part MIME email. The task list is represented by a MIME part containing a iCalendar (vToDo). For sending tasks to Windows clients a special legacy format must be readable and writeable by the KDE client. See the appendix for the exact file format.

Table 3-6. Assigning Tasks to Other Users

Step	Action	Remarks
------	--------	---------

Step	Action	Remarks
1	Create a task for another user in the own task list and send	The task is sent as multi-part MIME email containing a vToDo
2	As the recipient opens the email he can import the task into his own task list	Task (multi-part MIME email) is moved into the users tasks folder

Windows clients can attach OLE objects like MS-Word, Excel, PowerPoint or similar. The free software client does ignore OLE data, as it is not usable on non-Microsoft platforms. Attaching complete files or sending URL references is the preferred portable and more efficient way. For compatibility reasons, platform independence and severe security concerns the free software client does not make use of OLE objects.

3.6. HotSync PDA Synchronization

The KDE client synchronizes using existing HotSync software (KDE Kitchensync and Conduits). Windows clients synchronize using existing proprietary software, which probably also make use of the HotSync Protocol. Further investigation is not done here.

Chapter 4. Kolab Server Components

Whatever configuration data can be held inside the LDAP directory will be stored there. Still, there is the need for individual configuration files of services that are run on the Kolab server:

1. OpenLDAP2
2. Postfix
3. Cyrus Imapd
4. ProFTPD
5. Apache 1
6. Inet Daemon
7. Cyrus SASL 2

4.1. OpenLDAP2 Directory Server

LDAP keeps the central user data including credentials. Cyrus authenticates against LDAP via the Cyrus-sasl library (version 2) and the saslauthd daemon. It is part of the security concept of the Kolab Groupware server to not actually have the groupware users as GNU/Linux shell accounts on the server.

Alternative implementations may also use different ways for authentication but then they must also create their own derivatives of the administration tools.

4.1.1. LDAP - Server requirements:

We require the possible use of SSL/TLS secured LDAP connections to the LDAP Server.

```
TLSCertificateFile    cert.pem
TLSCertificateKeyFile key.pem
```

The Kolab daemon implements the LDAP replication protocol in order to get automatically notified when data in the LDAP directory changes. Therefore the LDAP server must enable replication for this host and port where the Kolab daemon listens. In the common case this is the port 9999 on localhost.

```
replica host=127.0.0.1:9999
        binddn="cn=replicator"
        bindmethod=simple credentials=secret
```

A directory service is optimized for speed with regards to read operations. A typical Kolab LDAP directory server fits even for very large installation in the main memory of the machine running the services. In order to further speed up common search operations we use indices.

```
index  objectClass  eq
index  uid           eq
index  mail          eq
index  alias         eq
```

4.1.2. Top Level LDAP Structure

It is difficult to find a commonly accepted LDAP scheme. It seems, most real life LDAP installations go for the domain oriented approach and lay out the structure after an existing domain/subdomain structure.

The most widely accepted and standardized object for storing personal data clearly is "inetOrgPerson".

The top hierarchy without any user specific data looks like this:

```
# Top Node
dn: dc=max, dc=kde, dc=org
dc: kde.org
objectClass: top
objectClass: domain

# tree node to hold internally visible address book entries
dn: cn=internal,dc=max,dc=kde,dc=org
cn: internal
objectClass: top
objectClass: namedObject

# tree node to hold externally (anonymous LDAP) visible addresses
dn: cn=external,dc=max,dc=kde,dc=org
cn: external
objectClass: top
objectClass: namedObject

# tree node to hold group objects from type inetOrgPerson
dn: cn=groups,dc=max,dc=kde,dc=org
cn: groups
objectClass: top
objectClass: namedObject
```

```

# tree node to hold resource from type kinetOrgPerson
dn: cn=resources,dc=max,dc=kde,dc=org
cn: resources
objectClass: top
objectClass: namedObject

# administrative user group
dn: cn=admin,dc=max,dc=kde,dc=org
cn: admin
objectClass: top
objectClass: groupOfNames
member: cn=manager,dc=max,dc=kde,dc=org
member: ...

# maintainer group - can create, modify, and delete user accounts
dn: cn=maintainer,dc=max,dc=kde,dc=org
cn: maintainer
objectClass: top
objectClass: groupOfNames
member: cn=TEST Admin,dc=max,dc=kde,dc=org
member: ...

# Kolab server configuration data for the various server processes
dn: k=kolab,dc=max,dc=kde,dc=org
fqhostname: max.kde.org
postfix-mydomain:
postfix-relaydomains:
postfix-mydestination:
postfix-mynetworks:
postfix-transport:
cyrus-autocreatequota:
cyrus-admins:
cyrus-imap:
cyrus-pop3:
cyrus-imaps:
cyrus-pop3s:
cyrus-sieve:
apache-http:
proftpd-userPassword:
objectClass:
objectClass:
proftpd-ftp:

```

Everything is build upon that base structure. In the future there may be several domains to be hosted on one LDAP server. With this concept, each domain would require such a base structure of its own. It may be located somewhere within a larger LDAP tree.

4.1.3. LDAP Attributes

```

# helper attribute to make the kolab root easily findable in
# a big ldap directory
attributetype ( 1.3.6.1.4.1.19414.2.1.1
  NAME ( 'k' 'kolab' )
  DESC 'Kolab attribute'
  SUP name )

# kolabDeleteflag used to be a boolean but describes with Kolab 2
# the fqdn of the server which is requested to delete this objects
# in its local store
attributetype ( 1.3.6.1.4.1.19414.2.1.2
  NAME 'kolabDeleteflag'
  DESC 'Per host deletion status'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

# alias used to provide alternative rfc822 email addresses for kolab users
attributetype ( 1.3.6.1.4.1.19414.2.1.3
  NAME 'alias'
  DESC 'RFC1274: RFC822 Mailbox'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

# kolabEncryptedPassword is an asymmetrically (RSA) encrypted copy of the
# cleartext password. This is required in order to pass the password from
# the maintenance/administration application to the kolabHomeServer running the
# resource handler application in a secure maner
attributetype ( 1.3.6.1.4.1.19419.2.1.4
  NAME 'kolabEncryptedPassword'
  DESC 'base64 encoded public key encrypted Password'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

# hostname including the domain name like kolab-master.bsi.de
attributetype ( 1.3.6.1.4.1.19414.2.1.5
  NAME ( 'fqhostname' 'fqdnhostname' )
  DESC 'Fully qualified Hostname including full domain component'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

# fqdn of all hosts in a multi-location setup
attributetype ( 1.3.6.1.4.1.19414.2.1.6
  NAME 'kolabHost'
  DESC 'Multivalued -- list of hostnames in a Kolab setup'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

```

```

# fqdn of the server containing the actual user data
attributetype ( 1.3.6.1.4.1.19419.1.1.1.1
  NAME 'kolabHomeServer'
  DESC 'server which keeps the users mailbox'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

# flag for allowing unrestricted length of mails
attributetype ( 1.3.6.1.4.1.19419.1.1.1.2
  NAME 'unrestrictedMailSize'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )

# allow delegates to act in your name (vacation/secretary boss use case)
# we use the syntax of rfc822 email addresses in order identify
# users allow to act in the name of others
attributetype ( 1.3.6.1.4.1.19419.1.1.1.3
  NAME 'kolabDelegate'
  DESC 'Kolab user allowed to act as delegates - RFC822 Mailbox/Alias'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

# For user, group and resource Kolab accounts
# Describes how to respond to invitations
# We keep the attribute as a string, but actually it can only have one
# of the following values:
#
# ACT_ALWAYS_ACCEPT
# ACT_ALWAYS_REJECT
# ACT_REJECT_IF_CONFLICTS
# ACT_MANUAL_IF_CONFLICTS
# ACT_MANUAL
# In addition one of these values may be prefixed with a primary email
# address followed by a colon like
# user@domain.tld: ACT_ALWAYS_ACCEPT
attributetype ( 1.3.6.1.4.1.19419.1.1.1.4
  NAME ( 'kolabInvitationPolicy' 'kolabResourceAction' )
  DESC 'Used by user, group and resource accounts to determine how to respond to invitatio
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

# time span from now to the future used for the free busy data
# measured in days
attributetype ( 1.3.6.1.4.1.19419.1.1.1.5
  NAME 'kolabFreeBusyFuture'
  DESC 'time in days for fb data towards the future'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )

```

```

# time span from now to the past used for the free busy data
# measured in days
attributetype ( 1.3.6.1.4.1.19419.1.1.1.6
  NAME 'kolabFreeBusyPast'
  DESC 'time in days for fb data towards the past'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )

# fqdn of the server as the default SMTP MTA
# not used in Kolab 2 currently as in Kolab 2 the
# default MTA is equivalent to the kolabHomeServer
attributetype ( 1.3.6.1.4.1.19419.1.1.1.7
  NAME 'kolabHomeMTA'
  DESC 'fqdn of default MTA'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

#####
# postfix attributes #
#####

attributetype ( 1.3.6.1.4.1.19414.2.1.501
  NAME 'postfix-mydomain'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

attributetype ( 1.3.6.1.4.1.19414.2.1.502
  NAME 'postfix-relaydomains'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

attributetype ( 1.3.6.1.4.1.19414.2.1.503
  NAME 'postfix-mydestination'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

attributetype ( 1.3.6.1.4.1.19414.2.1.504
  NAME 'postfix-mynetworks'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

attributetype ( 1.3.6.1.4.1.19414.2.1.505
  NAME 'postfix-relayhost'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

attributetype ( 1.3.6.1.4.1.19414.2.1.506

```

```

NAME 'postfix-transport'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

attributetype ( 1.3.6.1.4.1.19414.2.1.507
  NAME 'postfix-enable-virus-scan'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )

attributetype ( 1.3.6.1.4.1.19414.2.1.508
  NAME 'postfix-allow-unauthenticated'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )

#####
# cyrus imapd attributes #
#####

attributetype ( 1.3.6.1.4.1.19414.2.1.601
  NAME 'cyrus-autocreatequota'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )

attributetype ( 1.3.6.1.4.1.19414.2.1.602
  NAME 'cyrus-admins'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

# enable plain imap without ssl
attributetype ( 1.3.6.1.4.1.19414.2.1.603
  NAME 'cyrus-imap'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )

# enable legacy pop3
attributetype ( 1.3.6.1.4.1.19414.2.1.604
  NAME 'cyrus-pop3'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )

# user specific quota on the cyrus imap server
attributetype ( 1.3.6.1.4.1.19414.2.1.605
  NAME 'cyrus-userquota'
  DESC 'Mailbox hard quota limit in MB'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )

# cyrus imapd access control list
# acls work with users and groups
attributetype ( 1.3.6.1.4.1.19414.2.1.651
  NAME 'acl'

```

```

EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

# enable secure imap
attributetype ( 1.3.6.1.4.1.19414.2.1.606
  NAME 'cyrus-imaps'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )

# enable secure pop3
attributetype ( 1.3.6.1.4.1.19414.2.1.607
  NAME 'cyrus-pop3s'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )

# enable sieve support (required for forward and vacation services)
attributetype ( 1.3.6.1.4.1.19414.2.1.608
  NAME 'cyrus-sieve'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )

# installation wide percentage which determines when to send a
# warning to the user
attributetype ( 1.3.6.1.4.1.19414.2.1.609
  NAME 'cyrus-quotawarn'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )

#####
# apache and php attributes #
#####

# enable plain http (no ssl)
attributetype ( 1.3.6.1.4.1.19414.2.1.701
  NAME 'apache-http'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )

# Allow freebusy download without authenticating first
attributetype ( 1.3.6.1.4.1.19414.2.1.702
  NAME 'apache-allow-unauthenticated-fb'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )

#####
# proftpd attributes #
#####

attributetype ( 1.3.6.1.4.1.19414.2.1.901
  NAME 'proftpd-defaultquota'
  EQUALITY integerMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )

```

```

attributetype ( 1.3.6.1.4.1.19414.2.1.902
  NAME 'proftpd-ftp'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )

attributetype ( 1.3.6.1.4.1.19414.2.1.903
  NAME 'proftpd-userPassword'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

```

4.1.3.1. attribute type delegate

With Kolab a user (kInetOrgPerson) may define who is allowed to act on behalf of herself. The optional multiple delegate attributes contain the UIDs of the persons. This property is checked when using the Kolab smtp daemon (Postfix) to send emails.

4.1.3.2. attribute type unrestrictedMailSize

This attribute is currently unused.

4.1.4. LDAP Object Classes

```

# main kolab server configuration
# storing global values and user specific default values
# like kolabFreeBusyFuture and kolabFreeBusyPast
objectclass ( 1.3.6.1.4.1.19414.2.2.1
  NAME 'kolab'
  DESC 'Kolab server configuration'
  SUP top STRUCTURAL
  MUST k
  MAY ( kolabHost $
    postfix-mydomain $
    postfix-relaydomains $
    postfix-mydestination $
    postfix-mynetworks $
    postfix-relayhost $
    postfix-transport $
    postfix-enable-virus-scan $
    postfix-allow-unauthenticated $
    cyrus-autocreatequota $
    cyrus-quotawarn $
    cyrus-autocreatequota $
    cyrus-admins $

```

```

    cyrus-imap $
    cyrus-pop3 $
    cyrus-imaps $
    cyrus-pop3s $
    cyrus-sieve $
    apache-http $
    apache-allow-unauthenticated-fb $
    proftpd-ftp $
    proftpd-defaultquota $
    kolabFreeBusyFuture $
    kolabFreeBusyPast $
    uid $
    userPassword ) )

# shared folders are typically visible to everyone subscribed to
# the server without the need for an extra login
objectclass ( 1.3.6.1.4.1.19414.2.2.9
    NAME 'kolabSharedFolder'
    DESC 'Kolab public shared folder'
    SUP top STRUCTURAL
    MUST cn
    MAY ( acl $
        cyrus-userquota $
        kolabHomeServer $
        kolabDeleteflag ) )

# used as a plain node for the LDAP tree. In contrast to unix filesystem directories
# LDAP nodes can and often do also have contents/attributes. We use kolabNamedObject
# in order to put more structure in the directory tree.
objectclass ( 1.3.6.1.4.1.5322.13.1.1
    NAME 'kolabNamedObject'
    SUP top STRUCTURAL
    MAY ( cn $ ou ) )

# kolab account
# we use an auxiliary in order to ease integration
# with existing inetOrgPerson objects
# Please note that userPassword is a may
# attribute in the schema but is mandatory for
# Kolab
objectclass ( 1.3.6.1.4.1.19414.3.2.2
    NAME 'kolabInetOrgPerson'
    DESC 'Kolab Internet Organizational Person'
    SUP top AUXILIARY
    MAY ( c $
        alias $
        kolabHomeServer $
        kolabHomeMTA $
        unrestrictedMailSize $
        kolabDelegate $
        kolabEncryptedPassword $
        cyrus-userquota $
        kolabInvitationPolicy $

```

```

        kolabFreeBusyFuture $
        calFBURL $
        kolabDeleteflag ) )

# kolab organization with country support
objectclass ( 1.3.6.1.4.1.19414.3.2.3
  NAME 'kolabOrganization'
  DESC 'RFC2256: a Kolab organization'
  SUP organization STRUCTURAL
  MAY ( c $
    mail $
    kolabDeleteflag $
    alias ) )

# kolab organizational unit with country support
objectclass ( 1.3.6.1.4.1.19414.3.2.4
  NAME 'kolabOrganizationalUnit'
  DESC 'a Kolab organizational unit'
  SUP organizationalUnit STRUCTURAL
  MAY ( c $
    mail $
    kolabDeleteflag $
    alias ) )

# kolab groupOfNames with extra kolabDeleteflag
objectclass ( 1.3.6.1.4.1.19414.3.2.5
  NAME 'kolabGroupOfNames'
  DESC 'Kolab group of names (DNs) derived from RFC2256'
  SUP groupOfNames STRUCTURAL
  MAY kolabDeleteflag )

```

4.1.5. LDAP OID Numbers

An OID number is a hierarchical identifier with periods separating each level. Each attribute and each objectclass must have a unique OID number. In order to allow easy consolidation of an Kolab LDAP server with other LDAP servers we require a globally unique OID number. The Kolab Project obtained the base OID 1.3.6.1.4.1.19414 from IANA to which we append values creating our own unique OID numbers.

If the OID does not start with 1.3.6.1.4.1.19414 the attribute or objectclass is a copy from some other vendor. You can research who owns an OID from the Alvestrand Data <http://www.alvestrand.no/objectid/top.html> web site OID lookup service

The suffixes used for the Kolab project are arbitrary unique without any internal semantics and get administered via the Kolab cvs server.

4.1.6. LDAP Access Control Lists

Access to individual attributes or objects is controlled via static LDAP access control lists. LDAP ACLs are not meant to be changed on a daily basis as editing them requires a server restart.

In order to still remain flexible the OpenLDAP server allows for regular expressions in access control lists.

In the following program listing we used the machine name `kepler`.

In general please beware of parsing issues with the OpenLDAP implementation. Unfortunately we encountered multiple times unexpected results due to parsing issues which did *not* result in error messages or warnings.

```

access to attr=userPassword
    by group="cn=admin,dc=kepler" =wx
    by group="cn=maintainer,dc=kepler" =wx
    by self =wx
    by anonymous =x
    by * none stop

access to attr=encryptedPassword
    by group="cn=admin,dc=kepler" =w
    by group="cn=maintainer,dc=kepler" =w
    by self =wx
    by * none stop

access to attr=mail
    by group="cn=admin,dc=kepler" write
    by group="cn=maintainer,dc=kepler" write
    by * read stop

access to attr=alias
    by group="cn=admin,dc=kepler" write
    by group="cn=maintainer,dc=kepler" write
    by * read stop

access to attr=uid
    by group="cn=admin,dc=kepler" write
    by group="cn=maintainer,dc=kepler" write
    by * read stop

access to dn="cn=nobody,dc=kepler"
    by anonymous auth stop

access to dn="cn=manager,dc=kepler"
    by dn="cn=nobody,dc=kepler" read
    by self write

```

```

    by anonymous auth stop

access to dn="cn=admin,dc=kepler"
    by group="cn=admin,dc=kepler" write
    by dn="cn=nobody,dc=kepler" read
    by self write
    by anonymous auth stop

access to dn="cn=maintainer,dc=kepler"
    by group="cn=admin,dc=kepler" write
    by dn="cn=nobody,dc=kepler" read
    by self write
    by anonymous auth stop

access to dn.regex="(.*,)?cn=internal,dc=kepler"
    by group="cn=admin,dc=kepler" write
    by group="cn=maintainer,dc=kepler" write
    by self write
    by dn="cn=nobody,dc=kepler" read
    by anonymous auth stop

access to dn.regex="(.*,)?cn=external,dc=kepler"
    by group="cn=admin,dc=kepler" write
    by group="cn=maintainer,dc=kepler" write
    by * read stop

access to dn="cn=external,dc=kepler"
    by dn="cn=nobody,dc=kepler" read
    by * search stop

access to dn="cn=internal,dc=kepler"
    by dn="cn=nobody,dc=kepler" read
    by * search stop

access to dn="k=kolab,dc=kepler"
    by group="cn=admin,dc=kepler" write
    by group="cn=maintainer,dc=kepler" read
    by dn="cn=nobody,dc=kepler" read
    by * none stop

access to *
    by self write
    by group="cn=admin,dc=kepler" write
    by group="cn=maintainer,dc=kepler" write
    by * read stop

```

4.1.7. LDAP Business Card

The attributes of an user entry (normal and privileged users) are layed out after the following example:

```
dn: cn=Hans Schmid, dc=max, dc=kde, dc=org
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: Top
givenname: Hans
title: Dipl.-Phys.
uid: hans99
sn: Schmid
mail: schmid@max.kde.org
userpassword:
cn: Hans Schmid
mailalias: hans.schmid@bsi.de
o:
ou:
street:
postalCode:
l:
c:
telephoneNumber:
facsimileTelephoneNumber:
```

Further attributes can be easily added to the object class as the project advances and other requirements are identified.

4.1.8. LDAP-Changes for Kolab Server 2.1

This section describes the changes in the LDAP structure that have been made between version 2.0 and 2.1 of the Kolab server. The main difference between the two versions is the support for multiple email-domains. The implementation of that feature in Kolab is relatively simple and the LDAP changes to accommodate this feature have been kept to a minimum.

4.1.8.1. Multidomainsupport

Multidomainsupport affects the following places in the LDAP structure immediately:

- The domains the kolab server is responsible for.

All domains are listed in the Kolab configuration object (dn: k=kolab,<base-dn>) in the attribute postfix-mydestination postfix-mydestination defines which domains the postfix MTA is responsible for.

- Domain-maintainers

Domain-maintainers are internal Accounts which have basically the same permissions as maintainers, with the exception that they only have access to a subset of the domains of the Kolab server. To which domains a domain-maintainer has write access to is defined by the domain objects.

There is one domain object for each domain and is represented by an LDAP entry under

```
dn: cn=domains,cn=internal,<base-dn>
cn: domains
objectClass: top
objectClass: kolabNamedObject
```

with the following structure:

```
dn: cn=<domain>,cn=domains,cn=internal,<base-dn>
objectClass: top
objectClass: kolabGroupOfNames
cn: <domain>
```

The members of this kolabGroupOfNames are the domain-maintainers who have access to the accounts of this domain.

Since "members" is a MUST attribute of kolabGroupOfNames, at least one member has to be given. In a standard Kolab installation this is the manager account (dn: cn=manager,cn=internal,<base-dn>).

Kolab's web-admin interface creates the domain objects when the first domain maintainers are defined.

Every domain-maintainer is an LDAP-objekt of the form:

```
dn: cn=<CN>,cn=internal,<base-dn>
objectClass: top
objectClass: inetOrgPerson
objectClass: kolabInetOrgPerson
sn: <SN>
cn: <CN>
userPassword:: <pwd>
uid: <uid>
```

Like maintainers and admins, domain-maintainer are members of a special kolabGroupOfNames:

```
dn: cn=domain-maintainer,cn=internal,<base-dn>
cn: domain-maintainer
```

```
objectClass: top  
objectClass: kolabGroupOfNames
```

In a stand kolab installation, the only member of this group is the manager account.

Domain-maintainers are only used by kolab's web-admin interface. Installation that use some other means to administrate the server may ignore domain-maintainers.

Which accounts, distribution lists, etc. belong to which domain is described in the sections below.

4.1.8.2. Associating Accounts with Domains

A kolab account belongs to the domain that is used as the domain of its primary email address, that is, the value of the "mail" attribute.

4.1.8.3. Distributionlists

In Kolab server 2.1, the CN of a distributionlist is its email-address. In version 2.0 it was only the local part of the email-address (the part before the @). This was changed in 2.1 to make it possible to have multiple distribution lists with the same local names in different domains.

A distributionlist belongs to the domain that is used as the domain in the value of the "mail" attribute.

4.1.8.4. Resource Manager

The attribute kolabEncryptedPassword is no longer used in version 2.1. In Kolab Server 2.0 it was used by the resource manager to have write access to the calendar folders of resource and group accounts. In 2.1 the resource manager always uses the calendar user for this.

4.1.8.5. Shared Folders

The object class kolabSharedFolder has two new attributes:

`kolabFolderType`

Defines the value of the IMAP annotation "folder-type" as used for the resource folders of normal kolab accounts. More details about this attribute can be found in the file kolab2.schema shipped with Kolab server 2.1.

alias

This attribute is not used yet by Kolab.

Furthermore, just as for distributionlists, in Kolab version 2.1 the domain is now part of the CN aufgenommen. Instead of simply <foldername> as in 2.0, it's now <foldername>@<domain>.

The domain given in the CN is the domain to which the shared folder belongs. Note that this is different from to accounts and distributionlists where the domain is taken from the "mail" attribute.

4.1.8.6. Postfix Relayport

The kolab configuration object (dn: k=kolab,<base-dn>) has a new attribute, postfix-relayport. This is the port of the relayhost for the postfix MTA. This attribute is used in main.cf.template.

4.1.8.7. Other schema changes

The object class kolabInetOrgPerson has a new attribute kolabComment. This is not used yet in Kolab.

The following attribute have been explicitly marked as "single value":

```
kolabFreeBusyFuture
kolabFreeBusyPast
kolabHomeMTA
kolabVacationBeginDateTime
kolabVacationEndDateTime
kolabVacationResendInterval
kolabVacationReplyToUCE
kolabForwardKeepCopy
kolabForwardUCE
postfix-enable-virus-scan
postfix-allow-unauthenticated
cyrus-autocreatequota
cyrus-imap
```

4.2. Postfix Mail Server

Postfix is a scalable, secure implementation of a SMTP mail transfer agent for UNIX operating systems. It is part of nearly all Linux distributions today and is considerably well documented and easy to

administer. We try to avoid dependencies on postfix-specific features. The client server architecture is modeled in such a way that we can even replace Postfix with another capable MTA like Exim or Zmailer later, if desired. This is however not part of this project.

4.2.1. LDAP Connection

Postfix is capable of using the LDAP storage for the following options:

<code>ldapsource_server_host</code>	LDAP Server IP
<code>ldapsource_server_port</code>	TCP Port
<code>ldapsource_timeout</code>	Query Timeout
<code>ldapsource_result_attribute</code>	
<code>ldapsource_bind</code>	
<code>ldapsource_bind_dn</code>	
<code>ldapsource_bind_pw</code>	
<code>ldapsource_search_base</code>	
<code>ldapsource_query_filter</code>	

At runtime Postfix can retrieve the following parameters from the LDAP directory:

<code>alias_maps</code>	replaces <code>/etc/aliases</code>
<code>canonical_maps</code>	Canonical Address Mapping (inc. envelope)
<code>lmtp_sasl_password_maps</code>	
<code>local_recipient_maps</code>	
<code>recipient_canonical_maps</code>	
<code>relocated_maps</code>	
<code>sender_canonical_maps</code>	
<code>smtp_sasl_password_maps</code>	
<code>transport_maps</code>	Static SMTP Routes
<code>virtual_maps</code>	Virtual Address Mapping

Kolab Version uses LDAP whenever possible. (This is a change from Kolab 1.0)

4.3. Antivirus and Antispam Daemon

4.3.1. Antivirus - Amavis Daemon

Amavisd is a high-performance interface between mailer (Postfix) and content checkers: virus scanners (ClamAV) and antispam scanners (SpamAssassin). It is written in Perl for maintainability, without paying a significant price for speed. It talks to Postfix via ESMTP.

Recent versions of amavisd integrate spamassassin. So the setup of spamassassin is straight forward and we use the available OpenPKG package.

Amavisd uses directories for unpacking, quarantining infected messages, and quarantining messages that it could not process correctly. These directories must exist and be writable by amavis when it starts.

Maintaining these directories and especially the quarantine is the job of a systemadministrator (root). It is possible to give another administration user without root privileges access to the quarantine though it is no good idea to forward malicious messages to some email account.

The later would pose a security thread to whomever who processes the messages from the quarantine with an email client. Even though currently the vast majority of malicious messages is targeted towards vulnerabilities in Windows products (mainly MS Outlook and MS Internet Explorer) there is a real risk of viruses and other malware taking advantage of security flaws in Linux or webbased applications.

```
use strict;

$MYHOME = '@@kolab_prefix@@/var/amavisd';
$mydomain = '@@postfix-mydomain@@';
$daemon_user = 'amavisd';
$daemon_group = 'amavisd';
$daemon_chroot_dir = $MYHOME;

$QUARANTINEDIR = "$MYHOME/quarantine";
$TEMPBASE = "$MYHOME/tmp";
$ENV{TMPDIR} = $TEMPBASE;
$helpers_home = $MYHOME;

$forward_method = 'smtp:127.0.0.1:10025';
$notify_method = $forward_method;
$inet_socket_port = 10024;
$inet_socket_bind = '127.0.0.1';
@inet_acl = qw( 127.0.0.1 );

@bypass_virus_checks_acl = qw( . );
@local_domains_acl = ( ".$mydomain" );

$DO_SYSLOG = 1; # (1 = syslog, 0 = logfile)
```

```

$LOGFILE = "$MYHOME/amavis.log";
$log_level = 5; # (0-5)

$hdrfrom_notify_sender = 'SpamAssassin helpdesk@domain.com';
$notify_spam_sender_tmpl = read_text("$MYHOME/notify_spam_sender.txt");

$final_spam_destiny = D_PASS; # Set to D_BOUNCE to block/notify, D_PASS to pass through

read_hash(\%whitelist_sender, '@@kolab_prefix@@/var/amavisd/whitelist');
read_hash(\%blacklist_sender, '@@kolab_prefix@@/var/amavisd/blacklist');
read_hash(\%spam_lovers, '@@kolab_prefix@@/var/amavisd/spam_lovers');

#defending against mail bombs
$MAXLEVELS = 14; # Maximum recursion level for extraction/decoding
$MAXFILES = 1500; # Maximum number of extracted files
$MIN_EXPANSION_QUOTA = 100*1024; # bytes (default undef, not enforced)
$MAX_EXPANSION_QUOTA = 300*1024*1024; # bytes (default undef, not enforced)
$MIN_EXPANSION_FACTOR = 5; # times original mail size (must be specified)
$MAX_EXPANSION_FACTOR = 500; # times original mail size (must be specified)

$path = '@@kolab_prefix@@/usr/sbin:@@kolab_prefix@@/usr/bin:@@kolab_prefix@@/bin'

#$banned_filename_re = new_RE();

$file = 'file';
$arc = ['nomarch', 'arc'];
$gzip = 'gzip';
$bzip2 = 'bzip2';
$uncompress = ['uncompress', 'gzip -d', 'zcat'];
$lha = 'lha';
$unarj = 'unarj';
$unrar = 'unrar';
$zoo = 'zoo';

# SpamAssassin settings
$sa_local_tests_only = 0;
$sa_auto_whitelist = 1; # comment this line out to turn off auto whitelist
$sa_mail_body_size_limit = 64*1024; # 64KB

$sa_tag_level_deflt = 3.0; # controls adding the X-Spam-Status and X-Spam-Level headers
$sa_tag2_level_deflt = 6.3; # controls adding 'X-Spam-Flag: YES', and editing Subject,
$sa_kill_level_deflt = $sa_tag2_level_deflt; # triggers spam evasive actions:

$sa_spam_subject_tag = '***SPAM*** ';
$sa_debug = 1; # comment this line out to turn off debugging

1; # insure a defined return

```

4.3.2. Antispam - Spamassassin

Recent versions of spamassassin are integrated into amavisd.

4.3.2.1. Spamassassin - User Interface

In the future we intend to integrate possible the horde sam module for maintaining user specific white- and blacklists <http://cvs.horde.org/cvs.php/sam/>.

4.3.3. Free Software Antivirus Scan Engine ClamAV

The costs of scanning for viruses and other malware skyrocks while the quality of many proprietary virus scanner option declines. A special problem is here the need for the supplier of proprietary software for diversification. This leads to technical suboptimal solutions like large and unreliable software for the plain download of pattern files. We therefore experience for example ever changing download URLs for new patterns and the need to update many components of the solution regularly.

Fortunately there have been recently very positiv reports (http://www.pcwelt.de/news/viren_bugs/37827/2.html) about the opensource ClamAV virus scanner. For Kolab we use the current ClamAV OpenPKG package. Of course it is easily possible to install a proprietary scanner either to replace or supplement ClamAV.

The Kolab ClamAV package is based on the OpenPKG package. The configuration of ClamAV `clamav.conf` is

```
##
## config template file for the Clam AV daemon
## Please read the clamav.conf(5) manual before editing this file.
##

LogFile @@@kolab_prefix@@@/tmp/clamd.log

LogFileMaxSize 8M

# Log time with an each message.
LogTime

# Enable verbose logging.
#LogVerbose

# This option allows you to save the process identifier of the listening
# daemon (main thread).
PidFile @@@kolab_prefix@@@/var/run/clamd.pid

TemporaryDirectory @@@kolab_prefix@@@/var/tmp
```

```

# Path to the database directory.
# Default is the hardcoded directory (mostly /usr/local/share/clamav,
# but it depends on installation options).
DatabaseDirectory @@@kolab_prefix@@@/var/lib/clamav

# The daemon works in local or network mode. Currently the local mode is
# recommended for security reasons.

# Path to the local socket. The daemon doesn't change the mode of the
# created file (portability reasons). You may want to create it in a directory
# which is only accessible for a user running daemon.
LocalSocket @@@kolab_prefix@@@/tmp/clamd

# Remove stale socket after unclean shutdown.
FixStaleSocket

# TCP port address.
TCPSocket 3310

# TCP address.
# By default we bind to INADDR_ANY, probably not wise.
# Enable the following to provide some degree of protection
# from the outside world.
TCPAddr 127.0.0.1

# Maximum length the queue of pending connections may grow to.
# Default is 15.
MaxConnectionQueueLength 30

# When activated, input stream (see STREAM command) will be saved to disk before
# scanning - this allows scanning within archives.
StreamSaveToDisk

# Close the connection if this limit is exceeded.
StreamMaxLength 30M

# Maximal number of a threads running at the same time.
# Default is 5, and it should be sufficient for a typical workstation.
# You may need to increase threads number for a server machine.

MaxThreads 10

# Thread (scanner - single task) will be stopped after this time (seconds).
# Default is 180. Value of 0 disables the timeout. SECURITY HINT: Increase the
# timeout instead of disabling it.
ThreadTimeout 500

# Maximal depth the directories are scanned at.
MaxDirectoryRecursion 45

# Follow a directory symlinks.
# SECURITY HINT: You should have enabled directory recursion limit to

```

```

# avoid potential problems.
FollowDirectorySymlinks

# Follow regular file symlinks.
FollowFileSymlinks

# Do internal checks (eg. check the integrity of the database structures)
# By default clamd checks itself every 3600 seconds (1 hour).
SelfCheck 600

# Execute a command when virus is found. In the command string %v and %f will
# be replaced by the virus name and the infected file name respectively.
#
# SECURITY WARNING: Make sure the virus event command cannot be exploited,
#                   eg. by using some special file name when %f is used.
#                   Always use a full path to the command.
#                   Never delete/move files with this directive !
#VirusEvent /usr/local/bin/send_sms 123456789 "VIRUS ALERT: %f: %v"

# This option enables scanning of Microsoft Office document macros.
ScanOLE2

##
## Mail support
##

# Uncomment this option if you are planning to scan mail files.
ScanMail

##
## Archive support
##

# Comment this line to disable scanning of the archives.
ScanArchive

# By default the built-in RAR unpacker is disabled by default because the code
# terribly leaks, however it's probably a good idea to enable it.
ScanRAR

# Options below protect your system against Denial of Service attacks
# with archive bombs.

# Files in archives larger than this limit won't be scanned.
# Value of 0 disables the limit.
# WARNING: Due to the unrarlib implementation, whole files (one by one) in RAR
#           archives are decompressed to the memory. That's why never disable
#           this limit (but you may increase it of course!)
ArchiveMaxFileSize 20M

```

```
# Archives are scanned recursively - e.g. if Zip archive contains RAR file,  
# the RAR file will be decompressed, too (but only if recursion limit is set  
# at least to 1). With this option you may set the recursion level.  
# Value of 0 disables the limit.  
ArchiveMaxRecursion 5  
  
# Number of files to be scanned within archive.  
# Value of 0 disables the limit.  
ArchiveMaxFiles 1000  
  
# Mark potential archive bombs as viruses (0 disables the limit)  
ArchiveMaxCompressionRatio 200  
  
# Use slower decompression algorithm which uses less memory. This option  
# affects bzip2 decompressor only.  
#ArchiveLimitMemoryUsage  
  
# Mark encrypted archives as viruses (Encrypted.Zip, Encrypted.RAR).  
ArchiveDetectEncrypted
```

4.3.4. Mandatory SMTP Email Addresses

Common Internet practice and internet standards (RFCs) require the existence of the following three email addresses.

1. The smtp address `postmaster@example.com` is required by Internet standards to contact the email administrator of the domain `example.com`.
2. The `hostmaster@example.com` address is used to contact the DNS administrator of `example.com`.
3. The internet email users expect `abuse@example.com` to be used to report abuses involving the mail domain `example.com`.

Email to all three addresses is forwarded to a normal Kolab user account setable in the Kolab Web Admin GUI.

Technically Kolab uses a LDAP based virtual table for the mapping of these mandatory email addresses to a normal Kolab user.

4.4. Cyrus IMAP Daemon

Cyrus is a widely-spread IMAP daemon for Unix environments. It stores emails using the maildir format. That basically results in a directory structure with each email represented by single file. The IMAP users are managed completely separate from the GNU/Linux system accounts. User quotas and access control list are provided by the daemon independent of the operating system.

Cyrus' most interesting feature is its excellent scalability. A single Cyrus instance can serve up to thousand simultaneous interactive IMAP users. The performance of the filesystem storage and the network bandwidth/latency largely determine the maximal scalability. Both of them need to be maximized to optimize the IMAP server's performance on a given hardware. During runtime, approximately one megabyte of main memory per concurrent IMAP instance (that is one interactive user) is to be estimated.

4.4.1. LDAP Connectivity

Cyrus can make use of an LDAP directory as backend for its authentication mechanism. This is a widely used feature and can be determined to be very practical and functional. It therefore uses the Cyrus SASL 2 library with its `saslauthd` daemon.

4.4.2. Cyrus Sieve

The historic Unix tool `procmail` can not be used to filter incoming email for Cyrus IMAP users. This is due to a special delivery procedure which must be followed to deliver mail to a local IMAP user. In addition heavy usage of `procmail` limits the scalability of the server significantly. The Cyrus IMAP daemon supports therefore a special scripting language that is tightly coupled to the running daemon. This language is called Sieve and it is fully standardized by the IETF. Sieve is a general purpose message filtering language. In this project, we use its capability for creating vacation messages and to handle managed shared resources automatically.

Server scripting and scalability are a sensitive issue. The scalability of the whole groupware solution depends heavily on the fact that all processor intensive logic is implemented on the client side. If the server has to process additional CPU- or I/O-intensive tasks then a performance degradation is to be expected. Putting the load on the clients basically means that the total processing power is increased with the number of clients.

The design principle for employing server side scripting in scalable solutions is: "avoid resource intensive server operations - smart clients scale better than smart servers"

4.4.3. Access Control Lists

Access Permissions are handled on *folder* level not on a message level. The user shall be able to query the GUI about access permissions of every folder via RMB menu (menu reachable via clicking right mouse button). The user is able to manipulate the permissions of his folders using all Kolab clients including the webclient.

On the server side we have to implement multiple things:

1. Provide an API and usage documentation for the access permissions (See definitions below for NONE, READ, APPEND, WRITE and ALL)

2. Provide a webgui for watching/manipulating access permissions for the user
3. Provide a webgui for watching/manipulating access permissions for the maintainers/administrators

API for access permissions as implemented in the cyrus imapd. The affected IMAP commands are written in brackets.

The basic internal IMAP ACLs

l	Lookup (visible to LIST/LSUB/UNSEEN)
r	Read (SELECT, CHECK, FETCH, PARTIAL, SEARCH, COPY source)
s	Seen (STORE \SEEN)
w	Write flags other than \SEEN and \DELETED
i	Insert (APPEND, COPY destination)
c	Create (subfolders)
d	Delete (STORE \DELETED, EXPUNGE)
a	Administer (SETACL)

These Cyrus access permission are combined in different ways in order to obtain Kolab access permissions. The user shall neither directly see nor manipulate the Cyrus access permissions.

The Kolab Access permissions

NONE
-
READ
lrs

APPEND

lrsi

WRITE

lrsiwcd

ALL

lrsiwcdca

Only these five high level Kolab access control permissions shall be used and available via the Kolab clients to the user. The Kolab client must use the native Cyrus ACLs in the backend.

In the `imapd.conf` we specify the `admins/maintainers` users which have `l` and `a` access permissions implicitly on every folder.

In the configuration file `imapd.conf` we specify the default access control list for public folders. This configuration option is stored and maintained as always in the LDAP directory.

Normal folders inherit the access control list from its parent folder.

For special purpose (e.g. bulletin boards) we use the following access control list

The special purpose Kolab Access permissions

READ ANON

lr

READ HIDDEN

rs

The `READ ANON` Kolab access permission is used if the specified user/group shall see the folder and can read it, but the server does not preserve the "Seen" and "Recent" flags. This set of rights is primarily useful for anonymous IMAP and public shared folders, where the permanent update of the flags by one user would be confusing the other users.

The `READ HIDDEN` Kolab access permission is used if the specified user/group shall be able to read the mailbox and the server preserves the "Seen" and "Recent" flags, but the mailbox is not visible to the users through the various mailbox listing commands. The user/group must know the name of the mailbox to be able to access it. This set of rights is useful for non public shared folders and avoid unnecessary leak of information (names of folders not accessible to the user) and also avoid questions at the helpdesk.

Basically the user is using the Kolab client using his primary identity. Based on this identity he gains access to folders owned by other users or the system (shared folders). There is no change of identity when accessing foreign folders. The server checks for access via the user identity / credentials provided when connecting to the Cyrus imap server.

When listing the available folders from the server in the imap subscription dialog the user sees all folders with the lookup permission in the folder tree.

Whenever sending a message (e.g. an email) via SMTP the user is asked which sender identity is to be used comparable to the current KDE 3.2.1 KMail implementation. There shall be no popups or modal dialogs in the way when doing so. The decision has finally to be made immediately before message is being send.

When editing/creating (IMAP append) groupware messages then the identity within the message must be set accordingly in order to be useful. (e.g. organizer of an event)

The GUI of the Kolab client shall be aware of the ACLs and behave accordingly. E.g. dont allow saving of a message in a dIMAP folder if write permissions are lacking. This is required in order to prevent problems when used in disconnected mode. The Kolab client has to keep the acl information uptodate in its local cache in order to make delayed access permission problems unlikely.

Nevertheless the Kolab client must be able to handle and resolve a potential a potential lock (e.g. permission to write to a folder with pending new messages got lost recently). The Kolab client shall continue to synchronize as much as possible, warn the user via a dialog about the fact that some folder could not be written to and preserve the messages. The conflict resolution e.g. mving or deleting the conflicting messages is up to the manual procedure by the user.

Kolab only ever uses positive access permissions. This means that the Kolab access permissions only control the access via allow parameters for users and groups. There is no concept of negative access permissions as well known from Apache like deny statements.

4.4.4. Cyrus Quota

Quotas allow server administrators to limit resources used by hierarchies of mailboxes on the server.

The Cyrus IMAP server supports quotas on storage, which is defined as the number of bytes of the relevant RFC-822 messages, in kilobytes. Each copy of a message is counted independently, even when the server can conserve disk space use by making hard links to message files. The additional disk space overhead used by mailbox index and cache files is not charged against a quota.

Quotas are applied to quota roots, which can be at any level of the mailbox hierarchy. Quota roots need not also be mailboxes. For the Kolab Server we limit ourself to top level quota roots. This means we only

apply quotas to the complete mailbox of a user (root of an individual account) and to the root of shared folder hierarchies.

Quota roots are created automatically when they are mentioned in the `setquota` command. Quota roots may not be deleted through the protocol. We automatically create the quota roots when applying the initial quota values to the quota roots.

4.4.4.1. Hard Quota

Normally, in order for a message to be inserted into a mailbox, the quota root for the mailbox must have enough unused storage so that inserting the message will not cause the block quota to go over the limit.

Mail delivery is a special case. In order for a message to be delivered to a mailbox, the quota root for the mailbox must not have usage that is over the limit. If the usage is not over the limit, then one message may be delivered regardless of its size. This puts the mailbox's usage over the quota, causing a user to be informed of the problem and permitting them to correct it. If delivery were not permitted in this case, the user would have no practical way of knowing that there was a mail that could not be delivered.

If the usage is over the limit, then the mail delivery will fail with a temporary error. This will cause the delivery system to re-attempt delivery for a couple of days (permitting the user time to notice and correct the problem) and then return the mail to the sender.

The Kolab clients must be able to handle the hard quota limit and provide a localized and meaningful error message. The message which could not get synchronized must remain in the folder. User is ask to manually solve the problem e.g. deletion of messages.

Each maintainer of a domain gets a daily quota report via email. This report has different sections for hard quota, soft quota and filesystem usage for everyone else. The frequency or disabling of this feature is configurable by an administrator (No GUI currently).

In order to be able to deal with quotas correctly the KDE client must first delete messages before trying to send new messages.

4.4.4.2. Soft Quota

When a user selects a mailbox whose quota root has usage that is close to or over the limit and the user has `d` rights on the mailbox, the server will issue an alert notifying the user that usage is close to or over the limit. The threshold of usage at which the server will issue quota warnings is set by the `quotawarn` configuration option.

The server only issues warnings when the user has `d` rights because only users with `d` rights are capable of correcting the problem.

While hard quotas have a system wide default in and are per user quantities the soft quota only makes sense as a system wide policy (e.g. 80%).

4.5. ProFTP Daemon

ProFTPD offers good security features such as a change root environment and a fine granular access configuration. Herein it can be configured to allow a store-only incoming directory for the free-busy lists for all groupware users. Apart from that it is completely replaceable by any other standard compliant FTP daemon, if desired.

Its only functionality on the Kolab server is the legacy mode to enable Windows clients to publish their free-busy lists via anonymous FTP on the server. If only KDE clients connect, the FTP functionality is not needed.

FTP is deactivated by default, for security reasons and is considered deprecated. It is to be expected that the FTP service will be removed with some future release of Kolab.

4.6. Kolab Server Backup Strategy

One of the biggest advantages of using the maildir format on the Kolab server is, that no special procedures need to be designed to deal with the well known open file issues of other solutions. A differential backup strategy with a few incremental data backups can afford to just ignore the fact that some files throughout the IMAP server filesystem structure may actually be opened during backup time. It is also possible to restore single mailboxes or even single emails with small to zero additional effort, compared to restoring any other file on the system. This is one of the biggest benefits over other available solutions which sometimes operate on a single large binary object like a multi-gigabyte database, that actually would require a shutdown or a highly sophisticated online-backup mechanism for save backup and restore procedures.

4.6.1. Backup of IMAP Store

Backup and restore of messages can be done with commonly available tools. There is no need for special backup clients.

4.6.1.1. Backup of IMAP messages

The source for the backup of the IMAP store is `/kolab/var/imapd/spool`. Each individual message is contained in a separate file. The normal message files (emails, events, tasks and notes) use an integer with a trailing dot (.) as filename. This integer is increased for every message in a folder and never reused. So the filename is unique within a folder at all times.

4.6.1.2. Restore of IMAP messages

Restoring of message is basically simply recovering the message files from the backup and then running `reconstruct` in order to cleanly integrate the recovered message in the imap databases.

The largest database is the mailbox directories. Each mailbox directory contains the following files:

1. message files

There is one file per message, containing the message in RFC 822 format. Lines in the message are separated by CRLF, not just LF. The file name of each message is the message's UID followed by a dot (.).

2. cyrus.header

This file contains a magic number and variable-length information about the mailbox itself.

3. cyrus.index

This file contains fixed-length information about the mailbox itself and each message in the mailbox.

4. cyrus.cache

This file contains variable-length information about each message in the mailbox.

5. cyrus.seen

This file contains variable-length state information about each reader of the mailbox who has "s" permissions.

The `reconstruct` program can be used to recover from corruption in mailbox directories. If `reconstruct` can find existing header and index files, it attempts to preserve any data in them that is not

derivable from the message files themselves. The state `reconstruct` attempts to preserve includes the flag names, flag state, and internal date. It derives all other information from the message files.

An administrator may recover from a damaged disk by restoring message files from a backup and then running `reconstruct` to regenerate what it can of the other files.

The `reconstruct` program does not adjust the quota usage recorded in any quota root files. After running `reconstruct`, it is advisable to run `quota -f` in order to fix the quota root files.

4.6.2. Backup of LDAP Directory

The data of the OpenLDAP server is stored in `/kolab/var/openldap/openldap-data/`. Doing a snapshot (e.g. using LVM tools) of this directory and its contents is the main step when backing up the LDAP directory.

The files in the `openldap-data` directory are actually Berkley DB databases. When recovering `bdb` files it is highly recommended to run `db_recover` on them in order to commit any missing database logs.

4.6.2.1. Portable Backup of LDAP Directory

The most portable way of doing a backup of the LDAP directory is to use `slapcat`. This creates a textual representation of the directory using the LDAP Directory Interchange Format (LDIF). For consistency it is highly recommended to stop the directory service while doing a `slapcat`.

4.6.2.2. Restoring a portable Backup of LDAP Directory

The LDIF file obtained from `slapcat` can be easily added to an otherwise empty LDAP server using `slapadd`.

Please note that the LDIF file generated by `slapcat` is suitable for use with `slapadd` but not `ldapadd`. As the entries are in database order, not superior first order, they cannot be loaded with `ldapadd` without first being reordered.

In case uninterrupted operation of the directory server is required you can simply replicate the directory server to a slave server setup. It is then safe to stop the slave during the backup operation without interrupting the main LDAP directory service.

4.7. Administrator User Interface

The Kolab server is administrated via a PHP based webinterface protected by SSL. The administrator interface is divided into the following subsections

1. Server Setup
2. Setup of Server Name, Domain Name, General Email Option, Enable/Disable of Legacy modes
3. Vacation Setup
4. Administrator interface for vacation messages
5. User Accounts: individual accounts with all user specific data including password, electronic business cards and the language setting for Microsoft Outlook.
6. Administration of shared folders

4.8. Multi Location Setup

Some organization are distributed over several distinct locations with slow wide area network (WAN) links between the locations and a fast local area network (LAN). In order to provide optimal performance for all users it is advisable to have a Kolab server at each location.

Each Kolab server primarily serves the local users while all Kolab servers talk to the central Kolab LDAP server. In order to further increase speed and lower the dependency on the availability of the central LDAP server the location may choose to run a local secondary LDAP server as long as all write operations are done on the central server.

The basic idea is that all Kolab server see the very same directory and learn from the homeServer LDAP attribute which are their users. Every Kolab server knows about *all* Kolab users and therefore is able to grant access to folders to users belonging to different Kolab home servers.

In addition this makes moving users from one location to the other very easy as the required account is already known to the target machine. A nifty and reliable to migrate IMAP folders is <http://gopher.quux.org:70/devel/offlineimap/>.

The Kolab clients are supposed to be able to use the homeServer attribute in order to figure out where to lookup the freebusy list of a user. Unfortunately some legacy clients (esp. MS Outlook) lack this ability so you are either forced to keep the contents of the freebusy directory (small amount of data) synchronized using external tools like rsync over ssh run from a cron job or you limit yourself to a single Kolab server within your organization to keep the freebusy lists of all users.

There is no final rule which solution works best for integrating legacy clients and the decision depends on many factors including locality of typical requests and network availability.

With Kolab 3.0 we will use Apache `mod_rewrite` modules in order to have a transparent server side support for legacy Kolab clients. We use this `mod_rewrite` in order to process the download requests with a php script, fetch the required freebusy list from the appropriate Kolab server and then present the result to the requesting client.

4.9. Maintenance of Kolab server

4.9.1. Installation of Kolab server

4.9.2. Upgrade of Kolab server

It is safe to expect that for security reasons, new features or bugfixes a new release of the Kolab server is often desirable.

We use the same `obmtool` for upgrading which is used for installing the Kolab server.

A safe method for updating in case a short downtime is acceptable is first to become the superuser of the system and then get the proper shell environment. Afterwards the new kolab rpm packages get installed and compiled if required. The `obmtool` takes care about downloading further required packages automatically. The intermediate `kolabconf` which happens to depend on OpenLDAP is sometimes mandatory for doing an update of the current configuration of the Kolab services. Last but not least all Kolab server processes get started.

Further refinement typically happens via the Kolab web administration gui.

1. `#su -`
2. `#!/kolab/etc/rc --eval all env`
3. `#!/kolab/etc/rc all stop`
4. `#!/obmtool kolab` (from the correct location of course)
5. `#!/kolab/etc/rc openldap start` (only openldap is needed when running kolabconf)
6. `#!/kolab/sbin/kolabconf`
7. `#!/kolab/etc/rc openldap stop`
8. `#!/kolab/etc/rc all start`

Just keep an eye out during the for `.rpmsave` files. They will cause the different services not to start...

4.9.3. Changing Manager/calendar/nobody password

The manager password on a Kolab server can be changed using the kolabpasswd command like this.

- `#!/kolab/bin/kolabpasswd`

The system passwords for calendar and nobody are internal only, so they are rarely changed. Nevertheless kolabpassword is also able to change these with:

- `#!/kolab/bin/kolabpasswd calendar`
- `#!/kolab/bin/kolabpasswd nobody`

Please note that in a Kolab multi-location setup the passwords must be changed on each individual node separately.

Chapter 5. Windows Kolab Clients

There are multiple Kolab clients on the Windows platform including proprietary plugins for Outlook like the Toltec Connector.

The mainly tested Windows client application to cooperate with the Kolab Server 2.0 and the KDE client Kontact is Outlook 2002 or later on Windows XP with the Toltec Connector plugin installed. No modifications are made to this proprietary software. But some optional settings in the "Extras->Options" dialog must be made to insure interoperability:

The following settings have to be made within Outlook to share the freebusy information on the Kolab server. By default Outlook uses an anonymous ftp server to publish the freebusy data. To retrieve freebusy information of participants Outlook uses the http method.

With Kolab 2 we do not require the legacy ftp upload feature anymore. Instead freebusy data can be requested at anytime from the Kolab server which is then generating the data on the fly. This allows for more sophisticated freebusy data independent of the limited Outlook capabilities. E.g. it is possible with Kolab 2 that multiple calendars including group calendars contribute to the freebusy data.

Due to the fact that the legacy ftp protocol is unsecure we very much discourage the usage of the real user password as stored in the LDAP directory in case the legacy ftp upload feature is required to integrate non Kolab 2 clients with the Kolab server.

For http access of the freebusy information on the Kolab server a valid username and password has to be given. The credentials will be verified via SASL authentication on the LDAP directory of the server. The procedure of setting the required options within Outlook follows (german version). One can also refer to the excellent Toltec Connector documentation

<http://www.toltec.co.za/downloads/tokhowto-1.0.pdf> for doing this.

1. "Einstellungen->Kalenderoptionen" "Besprechungsanfragen standardmäßig als iCAL senden"
"Frei/Gebucht Optionen->Unter dieser URL veröffentlichen" URL: empty "Frei/Gebucht Optionen->Unter dieser URL suchen" URL: `http://<USERNAME>:<PASSOWRD>@<kolab server>/freebusy/%NAME%.ifb`
2. "Email Format" "Senden im Nachrichtenformat" : "Nur Text"
3. "Internet Mail" "MIME : Quoted Printable"

Outlook with the Toltec Connector 1 operates on the IMAP folder slightly differently from the KDE client. The file format for storing notes, task lists, calendar events, and contacts is a multi-part MIME message with an additional TNEF (Transport Neutral Encapsulation Format) encoded MIME part being an additional information carrier. TNEF is a proprietary Microsoft format which serializes MAPI objects. However, with the above adjustments made to the configuration of Outlook it can be advised to send and receive calendar events, notes, contacts, and task lists via the almost open format we describe later in the appendix.

The next generation of the Toltec Connector (version numbers 2.x) is in development and aims at being fully compatible with the Kolab 2 XML storage format.

5.1. Language dependencies

Due to the fact that Microsoft Outlook does not only localize the user visible interface but also internal data including non user visible filenames, directory names, etc. it was not possible to change the language of the Microsoft Outlook without loosing old data. This problem has been solved with Kolab 2 by using IMAP annotations which define the intended purpose of a folder. E.g. it defines that a folder independent of its name shall be the default calendar store.

Chapter 6. The KDE Client

The KDE client is implemented using these existing Free Software KDE projects and their technologies. This development effort is directly conducted in the KDE CVS repository and shall be integrated in future releases of the existing Free Software projects:

- KDE Kontact (used as a container for KParts)
- KMail KPart (with Ägypten/SPHINX encryption extensions)
- KOrganizer KPart
- KAddressbook KPart
- KPilot with Conduits

The following implementation activities have been identified:

- Integrate the components with the graphical Kontact shell. Kontact acts as a kparts container and runs all the KDE components for the PIM solution as embedded kparts.
- Add proper offline support to the IMAP kioslave called disconnected IMAP (dIMAP).
- Extend the GUI to support further functionality.
- implement the new Kolab XML storage format in addition to the existing iCalendar format in order to become fully interoperable with clients that are not based on KOrganizer (e.g. Outlook based clients).
- implement logic for complex groupware functionality
- implement further features as given in the previous chapters
- keep iCalendar format for everything sent over SMTP while use the new XML storage format and the new IMAP features like annotations to make the clients better interoperate with each other.
- development of migration tools to convert mailboxes between usage with the KDE client and Outlook/Toltec

A more detailed description is out of the focus of this document.

6.1. Language dependencies

In order to stay compatible to Microsoft Outlook clients in Kolab 1 it was required that the KDE client determined the language used to access folders on the Kolab server. With Kolab 2 this is much more cleanly and more robustly solved using IMAP annotations. IMAP annotation allows us to add small annotations to an IMAP folder describing its use. Currently annotations are used to define calendar, task and contact folders. One of each category of groupware folders is defined as the default folder for its type. E.g. there is always a default calendar which is used for merging incoming event requests etc. In addition IMAP folder annotations are used for marking calendars as relevant for the creation of freebuys information. Basically this means that the Kolab 2 clients will be interoperable with each other independent of the localizations and languages used.

Appendix A. The KDE client File Formats

A.1. Calendar Event

(conformant to IETF RFC 2445)

```
Received: ...
From: "Tassilo Erlewein" <tassilo@kolabserver.kde.org>
To: <test1@kolabserver.kde.org>
Subject: WG: Wichtige Notiz
Date: Mon, 9 Sep 2002 17:26:22 +0200
Message-ID: ...
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="-----_NextPart_000_0013_01C25826.04F609F0"
Importance: Normal
```

This is a multi-part message in MIME format.

```
-----_NextPart_000_0013_01C25826.04F609F0
Content-Type: text/plain;
        charset="iso-8859-1"
Content-Transfer-Encoding: 7bit
```

This is an alternative representation of a TEXT/CALENDAR MIME Object

```
When: 7/1/1997 10:00AM PDT - 7/1/97 10:30AM PDT
Where:
Organizer: fool@example.com
Summary: Phone Conference
```

```
-----_NextPart_000_0013_01C25826.04F609F0
Content-Type: text/calendar; method=REQUEST;
        charset="utf-8"
Content-Transfer-Encoding: quoted-printable
```

```
BEGIN:VCALENDAR
PRODID:-//ACME/DesktopCalendar//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VEVENT
ORGANIZER:mailto:fool@example.com
ATTENDEE;ROLE=CHAIR;ATTSTAT=ACCEPTED:mailto:fool@example.com
ATTENDEE;RSVP=YES;TYPE=INDIVIDUAL:mailto:foo2@example.com
DTSTAMP:19970611T190000Z
DTSTART:19970701T170000Z
DTEND:19970701T173000Z
SUMMARY:Phone Conference
```

```
UID:calsvr.example.com-8739701987387771
SEQUENCE:0
STATUS:CONFIRMED
END:VEVENT
END:VCALENDAR
-----=_NextPart_000_0013_01C25826.04F609F0--
```

A.2. Note

```
Received: ...
From: "Tassilo Erlewein" <tassilo@kolabserver.kde.org>
To: <test1@kolabserver.kde.org>
Subject: WG: Wichtige Notiz
Date: Mon, 9 Sep 2002 17:26:22 +0200
Message-ID: ...
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="-----=_NextPart_000_0013_01C25826.04F609F0"
Importance: Normal
```

This is a multi-part message in MIME format.

```
-----=_NextPart_000_0013_01C25826.04F609F0
Content-Type: text/plain;
        charset="iso-8859-1"
Content-Transfer-Encoding: 7bit
```

(text representation of the note, yet to be determined)

```
-----=_NextPart_000_0013_01C25826.04F609F0
Content-Type: text/x-note
        charset="utf-8"
Content-Transfer-Encoding: quoted-printable
```

Here comes the note text ...

```
-----=_NextPart_000_0013_01C25826.04F609F0--
```

A.3. Contact

```
Received: ...
From: "Tassilo Erlewein" <tassilo@kolabserver.kde.org>
```

```
To: >test1@kolabserver.kde.org>
Subject: WG: Wichtige Notiz
Date: Mon, 9 Sep 2002 17:26:22 +0200
Message-ID: ...
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="-----_NextPart_000_0013_01C25826.04F609F0"
Importance: Normal
```

```
This is a multi-part message in MIME format.
-----_NextPart_000_0013_01C25826.04F609F0
Content-Type: text/plain;
        charset="iso-8859-1"
Content-Transfer-Encoding: 7bit
```

(text representation of the vcard, to be determined)

```
-----_NextPart_000_0013_01C25826.04F609F0
Content-Type: text/x-vcard;
        name="Kalle Dalheimer.vfb"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment;
        filename="Kalle Dalheimer.vfb"
```

```
BEGIN:VCARD
VERSION:2.1
N:Dalheimer;Kalle
FN:Kalle Dalheimer
ORG:Klarvielens Kunsult
TITLE:CEO
TEL;WORK;VOICE:1413413241
REV:20020909T150816Z
END:VCARD
```

```
-----_NextPart_000_0013_01C25826.04F609F0--
```

A.4. Task Lists

(conformant to IETF RFC 2445)

```
Received: ...
From: "Tassilo Erlewein" <tassilo@kolabserver.kde.org>
To: <test1@kolabserver.kde.org>
Subject: WG: Wichtige Notiz
Date: Mon, 9 Sep 2002 17:26:22 +0200
Message-ID: ...
MIME-Version: 1.0
```

```
Content-Type: multipart/mixed;
    boundary="-----_NextPart_000_0013_01C25826.04F609F0"
Importance: Normal
```

This is a multi-part message in MIME format.

```
-----_NextPart_000_0013_01C25826.04F609F0
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: 7bit
```

(text representation to ToDo, yet to be determined)

```
-----_NextPart_000_0013_01C25826.04F609F0
Content-Type: text/calendar
Content-Transfer-Encoding: 7bit
```

```
BEGIN:VCALENDAR
PRODID:-//ACME/DesktopCalendar//EN
METHOD:REQUEST
VERSION:2.0
BEGIN:VTODO
ORGANIZER:Mailto:A@example.com
ATTENDEE;ROLE=CHAIR:Mailto:A@example.com
ATTENDEE;RSVP=TRUE:Mailto:B@example.com
ATTENDEE;RSVP=TRUE:Mailto:C@example.com
ATTENDEE;RSVP=TRUE:Mailto:D@example.com
DTSTART:19970701T170000Z
DUE:19970722T170000Z
PRIORITY:1
SUMMARY:Create the requirements document
UID:calsrv.example.com-873970198738777-00@example.com
SEQUENCE:0
DTSTAMP:19970717T200000Z
STATUS:Needs Action
END:VTODO
END:VCALENDAR
```

```
-----_NextPart_000_0013_01C25826.04F609F0--
```

A.5. Free-Busy Lists

(conformant to IETF RFC 2445, note that this is not an email format)

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//RDU Software//NONSGML HandCal//EN
```

```
BEGIN:VFREEBUSY

ORGANIZER:MAILTO:jsmith@host.com
DTSTART:19980313T141711Z
DTEND:19980410T141711Z
FREEBUSY:19980314T233000Z/19980315T003000Z
FREEBUSY:19980316T153000Z/19980316T163000Z
FREEBUSY:19980318T030000Z/19980318T040000Z
URL:http://www.host.com/calendar/busytime/jsmith.ifb
END:VFREEBUSY
END:VCALENDAR
```

Appendix B. Legacy File Formats

To visualize the format of the appropriate message type we give examples of messages which were sent by Outlook with Bynari plugin and stored on the Cyrus IMAP server. In addition to the formats presented here it is necessary that the KDE client can handle multi-part MIME emails with TNEF encoded attachments (known as "winmail.dat", containing MAPI object information). To explain the TNEF format is out of the scope of this document.

B.1. Calendar Event

```
Received: ...
From: "Achim Frank" <achim@kolabserver.kde.org>
To: <test1@kolabserver.kde.org>
Subject: test1 - meeting
Date: Sun, 8 Sep 2002 18:49:21 +0200
Message-ID: ...
MIME-Version: 1.0
Content-Type: text/calendar; method=REQUEST;
    charset="utf-8"
Content-Transfer-Encoding: quoted-printable
X-Priority: 3 (Normal)
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook CWS, Build 9.0.2416 (9.0.2911.0)
Importance: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2919.6700

BEGIN:VCALENDAR
PRODID:-//Microsoft Corporation//Outlook 9.0 MIMEDIR//EN
VERSION:2.0
METHOD:REQUEST
BEGIN:VEVENT
ATTENDEE;CN=3Dtest1@kolabserver.kde.org;ROLE=3DREQ-PARTICIPANT;RSVP=3DTRUE:MAILTO=
:test1@kolabserver.kde.org
ORGANIZER:MAILTO:achim@kolabserver.kde.org
DTSTART:20020908T193000Z
DTEND:20020908T200000Z
LOCATION:daheim
TRANSP:OPAQUE
SEQUENCE:0
UID: ...
DTSTAMP:20020908T164921Z
DESCRIPTION:Zeit: Sonntag\, 8. September 2002 21:30-22:00 (GMT+01:00)
    Amsterdam\, Berlin\, Bern\, Rom\, Stockholm\, Wien.\nOrt:
    daheim\n\n*~*~*~*~*~*~*~*~*~*\n\ntest1 - einladung
SUMMARY:test1 - meeting
PRIORITY:5
CLASS:PUBLIC
```

```
BEGIN:VALARM
TRIGGER:PT15M
ACTION:DISPLAY
DESCRIPTION:Reminder
END:VALARM
END:VEVENT
END:VCALENDAR
```

B.2. Note

```
Received: ...
From: "Tassilo Erlewein" <tassilo@kolabserver.kde.org>
To: <test1@kolabserver.kde.org>
Subject: WG: Wichtige Notiz
Date: Mon, 9 Sep 2002 17:26:22 +0200
Message-ID: ...
MIME-Version: 1.0
Content-Type: multipart/mixed;
    boundary="-----_NextPart_000_0013_01C25826.04F609F0"
X-Priority: 3 (Normal)
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook CWS, Build 9.0.2416 (9.0.2911.0)
Importance: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2919.6700
```

This is a multi-part message in MIME format.

```
-----_NextPart_000_0013_01C25826.04F609F0
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: 7bit
```

```
-----_NextPart_000_0013_01C25826.04F609F0
Content-Type: message/rfc822
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment
```

```
Subject: Wichtige Notiz
Date: Tue, 3 Sep 2002 18:13:16 +0200
MIME-Version: 1.0
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: 8bit
X-Priority: 3 (Normal)
```

X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook CWS, Build 9.0.2416 (9.0.2911.0)
Importance: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2919.6700

hier eine sehr wichtige Notiz:
bla bla bla

-----=_NextPart_000_0013_01C25826.04F609F0--

B.3. Contact

Received: ...
From: "Tassilo Erlewein" <tassilo@kolabserver.kde.org>
To: <test1@kolabserver.kde.org>
Subject: WG: Kalle Dalheimer
Date: Mon, 9 Sep 2002 17:08:26 +0200
Message-ID: ...
MIME-Version: 1.0
Content-Type: multipart/mixed;
 boundary="-----_NextPart_000_0002_01C25823.836243B0"
X-Priority: 3 (Normal)
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook CWS, Build 9.0.2416 (9.0.2911.0)
Importance: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2919.6700

This is a multi-part message in MIME format.

-----=_NextPart_000_0002_01C25823.836243B0
Content-Type: text/plain;
 charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

-----=_NextPart_000_0002_01C25823.836243B0
Content-Type: text/x-vcard;
 name="Kalle Dalheimer.vfb"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment;
 filename="Kalle Dalheimer.vfb"

BEGIN:VCARD
VERSION:2.1
N:Dalheimer;Kalle

FN:Kalle Dalheimer
ORG:Klarvielens Kunsult
TITLE:CEO
TEL;WORK;VOICE:1413413241
REV:20020909T150816Z
END:VCARD

-----=_NextPart_000_0002_01C25823.836243B0--

B.4. Task

Received: ...
From: "Tassilo Erlewein" <tassilo@kolabserver.kde.org>
To: <test1@kolabserver.kde.org>
Subject: WG: sdljlsjsg
Date: Mon, 9 Sep 2002 17:29:17 +0200
Message-ID: ...
MIME-Version: 1.0
Content-Type: multipart/mixed;
 boundary="-----=_NextPart_000_0018_01C25826.6D250760"
X-Priority: 3 (Normal)
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook CWS, Build 9.0.2416 (9.0.2911.0)
Importance: Normal
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2919.6700

This is a multi-part message in MIME format.

-----=_NextPart_000_0018_01C25826.6D250760
Content-Type: text/plain;
 charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

-----=_NextPart_000_0018_01C25826.6D250760
Content-Type: message/rfc822
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment

From: "Tassilo Erlewein" <tassilo@kolabserver.kde.org>
To: <tassilo@kolabserver.kde.org>
Subject: sdljlsjsg
Date: Tue, 3 Sep 2002 18:29:56 +0200
MIME-Version: 1.0
Content-Type: text/plain;

```
charset="iso-8859-1"  
Content-Transfer-Encoding: 7bit  
X-Priority: 3 (Normal)  
X-MSMail-Priority: Normal  
X-Mailer: Microsoft Outlook CWS, Build 9.0.2416 (9.0.2911.0)  
Importance: Normal  
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2919.6700
```

```
-----=_NextPart_000_0018_01C25826.6D250760--
```