LysKOM Protocol A

Protocol version 10
Edition 10.7
(lyskomd 2.0.7)

by the LysKOM Developers

This is the LysKOM Protocol A specification, edition 10.7. It specifies version 10 of the protocol. It corresponds to version 2.0.7 of lyskomd.

Table of Contents

LysKOM Protocol A

This document specifies version 10 of LysKOM Protocol A. This is edition 10.7 of the specification. It corresponds to version 2.0.7 of lyskomd.

The most up-to-date version if this document can always be found at http://www.lysator.liu.se/lyskom/protocol/.

LysKOM is a conferencing system[1]. Similar systems were QZ-KOM and PortaCOM[2]. The LysKOM system is copyrighted by Lysator Academic Computing Society and distributed under conditions of the GNU Public License. LysKOM and its documentation is provided "as is" without warranty of any kind.

Anything described here as "unspecified" is liable to change in future protocol versions.

This specification is the work of several people. The main contributors have been Per Cederqvist ceder@lysator.liu.se, David Byers byers@lysator.liu.se, Pär Emanuelsson pell@lysator.liu.se, Thomas Bellman bellman@lysator.liu.se, Lars Aronsson aronsson@lysator.liu.se, Linus Tolke linus@lysator.liu.se and Kent Engström kent@lysator.liu.se.

The LysKOM developers can be reached by email to lyskom@lysator.liu.se. If you find any errors in this document, please report them at http://bugzilla.lysator.liu.se/. Use '`lyskomd`' as the product, and '`documentation`' as component.

[1] Or in modern terms, enabling technology for Computer-Supported Cooperative Work (CSCW).

[2] Also known as "PottaKOM" and "BortaKOM."

1 Concepts used in LysKOM

This chapter introduces the concepts used in LysKOM, such as articles, conferences and sessions.

1.1 Articles

An article is represented as a value of the type `Text-Stat` and a string containing the article contents. An article will usually have one or more recipients and may be a comment or footnote to other articles. Each article is kept in the database until it is older than the `nice` value of each of its recipients and it is not marked by any user.

There is an array of `Misc-Info` included in the `Text-Stat`. This array contains information about recipients, senders, comments and footnotes.

Each article is identified by a number, the global[1] article number (the `Text-No`). Global numbers are assigned in ascending order to new articles, and are never reused. If an article has recipients it will also have a local number for each recipient (the `Local-Text-No`). Local numbers are used in some data structures to provide more compact storage and to provide an ordering of articles for a particular recipient. Local numbers are assigned in ascending order and are never reused for a particular recipient, though different recipients will have articles with the same local numbers.

Occasionally it is necessary to map between local and global numbers. The server call `local-to-global` does this (see Section 4.104 [local-to-global], page 109).

1.2 Conferences

Conferences hold articles. They are represented in the protocol as a data type called `Conference`. Each conference has a *creator*, the person who created the conference, and a *supervisor*, a conference whose members can modify the conference. If the supervisor is a person, the members of that person's mailbox are supervisors, which in most cases is only that person. Persons are a special case: in addition to those who are supervisors for a certain person because of the `supervisor` field, the person is always a supervisor of himself. There is one exception to this special case: the person cannot use `set-supervisor` (see Section 4.19 [set-supervisor], page 53) to change who is his supervisor.

We have also introduced a type called `UConference` (pronounced micro-conf-stat) which holds a subset of the information contained in the full `Conference` type. Use the `UConference` type whenever possible since it places a much smaller load on the LysKOM server.

Each conference has a type, which is essentially a collection of boolean flags. Currently the flags `rd-prot`, `letterbox`, `secret`, `original`, `allow-anonymous` and `forbid-secret` are defined.

`rd-prot` The conference is protected from reading by non-members. Persons become members by having one of the existing members or supervisors add him or her to the conference. This restriction is enforced by the server.

[1] The number is not truly global; it is local to a specific LysKOM server.

`original` Conferences of this type are intended for original articles only. Comments are to be redirected to the `super-conf` instead. This restriction is not enforced by the server; clients must implement this functionality. (See Section A.2.4 [Recipients of comments], page 143, for a summary of how a client should select the recipients of a comment.)

`letterbox`
Conferences of this type are connected to persons. Letters to a person are sent to the mailbox and the name of the mailbox is synchronized with the person name. It is currently not possible to explicitly set or clear this flag on a conference.

`secret` Conferences of this type are secret. The server will not divulge any information about the existence of the conference to persons who are not members or supervisors of the conference. If a mailbox is made secret, that person cannot log in using the person name, but must specify a person number instead.

`allow-anonymous`
Conferences of this type accept anonymous articles. Other conferences will reject anonymous articles.

`forbid-secret`
Conferences of this type do not allow secret members. If a conference is changed to this type, preexisting secret members remain secret.

1.3 Persons and Sessions

Persons are represented in the protocol by values of the type `Person`. Associated with persons are statistics, a set of personal flags and a set of privileges (see Section 1.6 [Security], page 8). Persons are also associated with a conference that has the same number as the person and the `letterbox` bit set.

Connections to the server are represented as values of the type `Static-Session-Info`, `Session-Info-Ident` or `Session-Info`. Sessions have session number that are unique for each session in the lifetime of the server execution. A single user can have several sessions running at once. The session is not released until the network connection is closed; a user can log in and out repeatedly in a single session.

1.4 The Misc-Info List

The `Misc-Info` list contains tagged data. The fields are sent in groups pertaining to a particular type of information: information about recipient; carbon copy recipient; blank carbon copy recipient; comment to; footnote to; comment in and footnote in. The information groups may be sent in any order and there may be any number of groups. Within each group the elements are always sent in the order listed below.

1.4.1 Recipient

`recpt` Starts a recipient group. It contains the conference number of a recipient of the article.

`loc-no` Always present within a recipient group. It contains the local text number of the article in the conference specified by the preceding `recpt` field.

`rec-time` If the recipient is a person, this element is added by the server when the recipient marks the article as read. It contains the time when the text was read.

`sent-by` Present when the recipient was added by a person other than the author (after the article was created.) It contains the person number of the person who added the recipient.

`sent-at` Present when the recipient was added after the article was created. It contains the time when the recipient was added.

1.4.2 Carbon Copy (CC) Recipient

The carbon-copy recipient group is identical to the recipient group above. The difference is how new comments to an article with a recipient or carbon-copy recipient are treated. A comment to an article is sent to all recipients, but not to carbon-copy recipients of the original article. This difference is enforced by the clients.

`cc-recpt` Starts a carbon-copy recipient group. It contains the conference number of a carbon-copy recipient of the article.

`loc-no` Always present in a CC recipient group. It contains the local text number of the article in the conference specified by the most recent `cc-recpt` field.

`rec-time` Present after the CC recipient has read the article. It contains the time when the article was read. Since only persons can read articles this will only be seen if the CC recipient is a person.

`sent-by` Present when a CC recipient was added by a person other than the author after the article had been created. It contains the person number of the person who added the CC recipient.

`sent-at` Present when a CC recipient was added after the article had been created. It is the time when the CC recipient was added.

1.4.3 Blank Carbon Copy (BCC) Recipient

The blank carbon-copy recipient group is identical to the carbon-copy recipient group above. The difference is the visibility of the information. A carbon-copy recipient group is visible to anyone that is allowed to fetch both the text status of the involved text and the conference status of the involved conference. (That is, as long as the conference isn't secret everybody is allowed to see the carbon-copy recipient group.)

A BCC recipient group is basically only visible to members and supervisors of the recipient. Persons that have the right to become a member of the recipient can also see it, as can the author of the text (unless the recipient is secret to him). This is enforced by the server.

This type of group was introduced in protocol version 10. When old-style calls such as `get-text-stat-old` (see Section 4.27 [get-text-stat-old], page 59) are used this will be converted to a CC recipient group by the server for the benefit of clients that don't understand this group. (This conversion will of course only be performed when the user is allowed to se the blank carbon copy.)

`bcc-recpt`
: Starts a blank carbon-copy recipient group. It contains the conference number of a blank carbon-copy recipient of the article.

`loc-no`
: Always present in a BCC recipient group. It contains the local text number of the article in the conference specified by the most recent `bcc-recpt` field.

`rec-time`
: Present after the BCC recipient has read the article. It contains the time when the article was read. Since only persons can read articles this will only be seen if the BCC recipient is a person.

`sent-by`
: Present when a BCC recipient was added by a person other than the author after the article had been created. It contains the person number of the person who added the BCC recipient.

`sent-at`
: Present when a BCC recipient was added after the article had been created. It is the time when the BCC recipient was added.

1.4.4 Comment To

`comm-to`
: Always present when the article is a comment to another article.

`sent-by`
: Present when the article was added as a comment by a person other than the author, after the article had been created. It contains the person number of the person who added the article as a comment.

`sent-at`
: Present when the article was added as a comment after the article had been created. It contains the time when it was added as a comment.

1.4.5 Footnote To

`footn-to`
: Always present when the article is a footnote to another article.

`sent-at`
: Present when the article was added as a footnote after the article had been created. It contains the time when it was added as a footnote.

1.4.6 Comment in

`comm-in`
: Present when there are comments to this article. It contains the article number which is a comment to this article.

1.4.7 Footnote in

`footn-in`
: Present when there are footnotes to this article. It contains the article number which is a footnote to this article.

1.5 The Aux-Item List

The aux-item list is used as a generic extension mechanism in the LysKOM server and in protocol A.

1.5.1 About Aux-Items

Aux-items were introduced in protocol version 10 as a mechanism for extending the conference, text and server information structures without changing the protocol. Persons were excluded since nobody could figure out a case where setting an aux-item on the mailbox wasn't as good as setting it on the person (another reason was that I was fed up writing aux-item code by the time they were working on texts and conferences.)

The exact structure of an aux item is specified elsewhere (see Section 3.2 [Auxiliary Information], page 20). The important fields here are the `aux-no`, `tag` and `data` fields.

The `aux-no` field is used to identify an item. The `aux-no` together with a text or conference number uniquely identifies a particular aux item. Items are numbered from one and up within each item list. Once assigned, the `aux-no` for an item is never changed. New items are guaranteed to be assigned numbers that have never been used before within a particular list.

The `tag` field identifies the type of aux item. It is used by the server and by clients to figure out how to interpret the data field, and by the server to decide if the item needs special treatment.

The `data` field is a simple string. The meaning of the string is determined by the `tag` field, but since it is a string, clients that have no understanding of the contents can successfully parse the item anyway (in contrast to items in the misc-info list.)

1.5.2 Predefined Aux-Item Types

Predefined Aux-Item types are part of Protocol A, and clients should support all of them. As with other parts of the protocol, changes to these definitions will be made backwards-compatible, if possible.

Creation and deletion of items with a predefined type can cause arbitrarily complex and wondrous behavior in the server. Furthermore, the server may place constraints on the items with regard to content, flags, who can create them, to what objects they can be attached and so forth. The server may also silently enforce specific values for any field of an item, regardless of what the client requests.

All items with tags in the range 1-9999 and 30000 and up are considered predefined. If a client attempts to create an item with a tag in this range, but the server has no idea what that tag means, the server will return an error (`illegal-aux-item`).

Some items with tags in the range 10000-19999 are also predefined. They are items that initially were reserved for private use for a specific client, but where it was later discovered that the tag was generally useful. They should ideally have been assigned a number in the 1-9999 range from the beginning, but if they already have a widespread use they can be redefined to be predefined. Such redefinition will always be made in cooperation with the client writer.

See Chapter 7 [Aux-Item Types], page 123, for the complete list of predefined Aux-Item types.

1.5.3 Client-Specific Aux-Item Types

Client-specific items do not cause the server to perform any magic. All the flags (except the delete flag) are left untouched, the data is not validated in any way, and anyone can create

any item. If you need more server support than this, your item should be on the predefined list.

All tags in the range 10000-19999 are reserved for clients. Blocks of 100 numbers at a time can be assigned to specific clients. A client should never create items with tags in a range assigned to another client or in an unassigned range. Assigned ranges will never change, except that specific aux-items may be redefined as predefined, as explained in Section 1.5.2 [Predefined Aux-Item Types], page 7.

Currently, the following ranges are assigned to clients:

- 10000-10099: The Elisp Client
- 10100-10199: komimportmail

If you want a range of numbers, send e-mail to the LysKOM development group.

1.5.4 Experimental Aux-Item Types

Experimental numbers are free for all. Use 'em any way you want. All numbers in the range 20000-29999 are for experimental use.

1.5.5 Defining New Aux-Item Types

If you want a new predefined item type, just document what it does, what the data format looks like and what the server is to do with the item and send this to the LysKOM development group. We'll assign a number to your item and put the documentation in this document.

If you're not sure what you want the data to look like yet, make a note in your documentation that the data format might change. Once you have a data format you're happy with, update the documentation so others may use your item.

If you need serious magic in the server (more than can be specified with the lyskomd configuration file), you'll probably have to write the code yourself, or hope that the development group thinks your idea is so cool we do the job for you.

The idea is not to reject any type of item, unless there's already an item type that does the job just as well. Adding item types should be a much less painful process than adding new calls.

1.6 Security

Security in LysKOM is based on two components. Each person has a set of privileges and each session has a security level. Rights in the system require both the sufficient privileges and a sufficient security level. The privileges currently available are `wheel`, `admin`, `statistic`, `create-conf`, `create-pers` and `change-name`. Security levels range from 0 to 255.

`wheel` *Normally not assigned*

Level 0	Person may always log in, even when LysKOM is crowded.
Level 6	Person may set Priv-Bits for all persons.
Level 7	Person may set password for all persons.

Level 8 Person acts as supervisor for everything.

Level 10 Person can read all articles.

`admin` *Normally not assigned*

Level 1 Shut down the server
Set motd-of-kom
Read last-login

Level 2 Read status of secret conferences and persons
Read the protected parts of person and conference statuses
Read the entire text status, even when there are secret recipients

Level 3 Change everybody's names

Level 4 Add/remove members
Add/remove recipients to articles

Level 5 Set super-conference
Remove articles

Level 6 Set administrator

`statistic`
Normally not assigned

Level 2 Read the statistics portions of persons, even if protected

`create-conf`
Normally not assigned[2]

Level 0 Create conferences

`create-pers`
Normally not assigned[3]

Level 0 Create persons

`change-name`
Normally assigned

Level 0 Change names of conferences he is supervisor for

1.7 Membership and Reading

Persons' memberships in conferences are represented in the protocol as arrays of `Membership`-typed values. This structure contains information about how and when the membership was created and which texts have been read in the conference.

There are two kinds of memberships. An active membership indicates that the person is actively participating in the conference, wants to know if there are unread texts and wants to

[2] The LysKOM server is normally configured so that anybody can create new conferences, even if they do not have this privilege.

[3] The LysKOM server is normally configured so that anybody can create new persons, even if they do not have this privilege.

receive messages send to the conference. A passive membership is similar to no membership at all. The person is still a member but will not receive messages sent to the conference and will not be notified when there are new texts. From the user's perspective, passive membership should be like no membership at all, but the server still remembers what the user has read in the conference while he or she was an active member. Since protocol version 10 a bit in the membership type field of the membership structure indicates the type of membership. Previously the server did not support passive memberships, but there was a convention that clients should treat the priority level zero as a passive membership.

The membership record indicates which texts have been read through the `last-text-read` and `read-texts` fields. All texts with local numbers up to `last-text-read` have been read. In addition, all texts with local numbers contained in the `read-texts` array have been read.

Finding out which articles a person has read in a particular conference requires a few calls. Normally, a client will retrieve a batch of perhaps 50 articles at a time. The outline of the process is as follows:

1. Fetch the membership to get the `last-text-read`
2. Use `local-to-global` (see Section 4.104 [local-to-global], page 109) to translate a number of local numbers to global numbers.
3. Remove the global numbers corresponding to local numbers contained in `read-texts` from the result.
4. Get and translate more texts as needed.

The process is complicated because of the translation between local and global text numbers. If the server does not implement the `local-to-global` call (see Section 4.104 [local-to-global], page 109), it is possible to use the less efficient but perfectly serviceable `get-map` call instead (see Section 4.35 [get-map], page 66).

2 Fundamentals of Protocol A

The data types in protocol A come in two flavors. The first (vanilla) are the simple data types from which the LysKOM (chocolate) data types are built. Simple data types include things like integers and strings while complex data types include things such as conferences and people.

2.1 Notation

This specification uses a BNF-like grammar to describe the protocol and its data elements.

Data fields have been given names that start with a lower-case letter.

Fundamental data types have names in all-caps (such as `INT32` and `ARRAY`).

Derived data types have names that start with an upper-case letter. (If the type contains more than one word, all words start with an upper-case letter, like this: `Text-Stat`.) The operator `::=` defines the name to its left.

Comments start with `!` (exclamation mark) and alternatives are separated by a `|` (vertical bar.) A `;` (semicolon) terminates statements in the grammar. In some specifications there are literal strings. There is to be no whitespace before or after literal strings unless there is whitespace in the literal itself.

2.2 Simple Data Types

2.2.1 Integers

`INT32`, `INT16`, `INT8` and `BOOL` are non-negative integers which must fit in 32, 16, 8 and 1 bits, respectively. They are transmitted to the server in ASCII-encoded decimal notation.

2.2.2 Strings

`HOLLERITH` denotes character strings of arbitrary length. They are transmitted as *n*H*text* where *text* is the string and *n* is the number of characters in *text* in decimal notation. All byte values are allowed in the string itself, including nulls.

Long live FORTRAN!

The character set used in the strings is not yet specified by Protocol A. In the future, some Unicode encoding will probably be used, but it is not yet decided which one or how the transition will be handled. Bug 99 is about the need for a Unicode roadmap; check that bug for the current state of the plans.

For now, which character set to use is a local policy of each server installation. There is not yet any way in the protocol to specify the character set that a certain server uses. Most clients currently assume that ISO 8859-1 (Latin-1) is used, and the default collate table of lyskomd also assumes ISO 8859-1. Conference names must currently use an 8-bit character set encoding where whitespace is defined as in ASCII, or conference matching won't work. `get-collate-table` (see Section 4.86 [get-collate-table], page 97) contains some more information about character set issues.

2.2.3 Bit Strings

`BITSTRING` is a string of bits, commonly used for a set of boolean-valued flags. Bit strings are denoted as

```
BITSTRING ( name-1; name-2; name-3; ... )
```

in this specification. They are transmitted as a sequence of ASCII ones and zeroes in the order the fields are listed.

For instance, given the specification

```
shape-of-world ::= BITSTRING (
        is-flat;
        is-round;
        is-2d;
        is-3d;
)
```

most peoples idea of `shape-of-world` would be sent as `0101` (round and three-dimensional.)

2.2.4 Enumerations

`ENUMERATION` is an integer constant. It is transmitted as an INT32, but only fixed values are permitted. Clients should be prepared to receive numbers outside the enumeration and either handle this gracefully as an error or use a reasonable default value in place of an invalid enumeration value.

An enumeration is specified as

```
ENUMERATION (
        name-1=value-1;
        name-2=value-2;
        name-3=value-3;
        ...
)
```

This specification states that name-1 is represented by the integer value-1, name-2 is represented by value-2 and name-3 is represented by value-3.

An enumeration can also be inherited from a SELECTION datatype:

```
Info-type ::= ENUMERATION-OF(Misc-Info)
```

This means that Info-type is an enumeration, that contains the same keys and values as the SELECTION Misc-Info.

For example, in the following specification, the constant guwal will be transmitted as the integer 2, ciokwe as the integer 3, and hopi as the integer 5.

```
language ::= ENUMERATION ( hakka      = 1;
                           guwal      = 2;
                           ciokwe     = 3;
                           yoruba     = 4;
                           hopi       = 5;
        )
```

2.2.5 Arrays

`ARRAY` is a list of a fixed number of elements of a single type. The specification for an array is `ARRAY` *type* where *type* is the type of the array elements.

Arrays are transmitted as an *n* `{` *element element* `...` `}` where *n* is the number of elements and each *element* is an element of the array. A special case is when the array is empty, in which case the server transmits it as `0 *`. Note that the client must always transmit empty arrays as `0 { }`.

In some calls the client can ask the server not to send the array contents, only its length. In these cases the array is transmitted as *n* `*` where *n* is the number of elements in the array.

2.2.6 Selection

`SELECTION` is tagged data. It consists of an INT32 selector followed by a tail of an arbitrary type and is specified as

```
SELECTION (
        n=name          tail : type;
        n=name          tail : type;
        ...
)
```

where each *n* is the selector value, *name* the selector name and *tail* the name of the selector tail and *type* its type. *name* and *tail* are often very similar, such as '`sent-by`' and '`sender`'.

When transmitted, the selector is transmitted as an INT32 followed by the tail belonging to that selector. For instance, given the specification

```
description ::= SELECTION (
       1=name           the_name : HOLLERITH;
       2=age            years    : INT32;
)
```

two legal messages of the type `description` are '`1 4HJohn`' and '`2 18`'.

2.2.7 RPC

`RPC` is a notation used to specify calls to the server. An RPC specification has the following form:

```
RPC (
        call [n] ( request ) -> ( reply ) ;
        call [n] ( request ) -> ( reply ) ;
)
```

where each *call* is the name of a call, *n* is the call number, *request* is a single data element sent as a request and *reply* is a single data element sent in reply from the server.

RPC calls are transmitted as *n request* where *n* and *request* have the same meaning as above. Note that in the client-server dialog a reference number must also be supplied with each request. This reference number is not part of the RPC itself, but is required for communications; see Section 2.4 [Client-Server Dialog], page 15.

2.2.8 Structure

Structures are collections of several data items. In the specification they are written as

```
( name : type ;
  name : type ;
  ...
)
```

where each *name* is the name of a data field and the corresponding *type* is its type.

Structures are transmitted as a sequence of their fields.

2.3 Connecting to the Server

The client-server dialog consists of two phases, establishing the connection and the LysKOM session itself.

A connection to the server is initiated by connecting to the appropriate network port[1] and sending a single letter which is used to select a protocol version. This letter should always be '`A`'[2].

The protocol version letter should be followed by connection information required by that protocol. In protocol A the connection information is a `HOLLERITH` string saying who the user connecting is followed by a newline character.

The format of the string should be '*username* `%` *hostname*'.

When the server has accepted the connection it replies with the string `LysKOM` on a single line.

```
% telnet kom.lysator.liu.se 4894
Trying 130.236.254.151 ...
Connected to varg.lysator.liu.se.
Escape character is '^]'.
A26Hbyers%kajsa.lysator.liu.se
LysKOM
```

Besides the string '`LysKOM`', the server may respond with '`%%LysKOM unsupported protocol.`' or '`%% No connections left.`'.

The '`%%LysKOM unsupported protocol.`' reply is sent if the server does not understand the requested protocol, that is, if the first character is anything but an '`A`'. The connection will be closed by the server as soon as this string has been sent.

The '`%% No connections left.`' reply is sent if the server is not accepting additional connections. This error is transient. The next connection attempt may succeed. Clients should wait a few seconds before attempting to make another connection after receiving this error. The connection will be closed by the server as soon as this string has been sent.

---

[1] The default port for a LysKOM server is 4894.

[2] In the early 1990s there was much discussion about the next generation LysKOM protocol, called "Protocol B", and at one point in time it was believed that "Protocol B" would arrive at the same time as "Emacs 19". "Emacs 19" arrived and was obsoleted by new versions. Meanwhile, it was discovered that "Protocol A" was more extensible that first believed. "Protocol A" will continue to evolve, but it is unlikely that it will ever be superseded by "Protocol B".

2.4 Client-Server Dialog

After connecting (see Section 2.3 [Connecting to the Server], page 14), calls to the server can be made. Most calls require the user to log in, but some calls can be made without a log-in. Calls to the server are made by sending a reference number followed by the call as specified. The call *must* be terminated with a linefeed character (ASCII 0x0A).

```
server-call ::=
        ( ref-no        :       INT32;
          request       :       Protocol-Request;
        )
```

The client may send several requests without waiting for the replies. However, the server will only buffer a limited amount of replies, so the client must check for pending replies from the server at least each time it sends requests to the server.

At some future point the server will reply with the result of the request or an error code preceded by an indicator and the reference number. The replies will be sent in the same order as the requests[3].

```
server-reply ::= ok-reply | error-reply | protocol-error;

ok-reply ::=
        ( "="
          ref-no        :       INT32;
          reply-data;
        )

error-reply ::=
        ( "%"
          ref-no        :       INT32;
          error-code    :       INT32;
          error-status  :       INT32;
        )

protocol-error ::= "%% LysKOM protocol error."
                 | "%%Insane token length."
                 | "%%Insane array size."
```

Our notation is not flexible enough to specify the two-way nature of the communication. `ref-no` in the reply is always the same as `ref-no` in the corresponding request. `reply-data` depends on which request was made and is specified together with each request.

Note that there is no whitespace after the initial indicator in the reply.

Data elements sent from the client to the server are separated by one or more space characters (ASCII 32). An entire call is terminated by a linefeed character (ASCII 10).

Earlier versions of the protocol specified that data elements could be separated by any number of linefeed (ASCII 10), return (ASCII 13), tab (ASCII 9) or space (ASCII 32) characters. Servers should be forgiving and understand requests using the older conventions,

[3] Client writers are encouraged to write the clients so that they are prepared for replies that are sent out-of-order. See Appendix D [Future changes], page 151, for speculations about how that may benefit the client in the future.

but clients must conform to the current convention of separating data elements with spaces and terminating all requests with a linefeed character. The `not-implemented` error code will not be returned properly unless the client follows this requirement.

The server may return two different types of errors. Normal `error-reply` errors are replies to syntactically correct calls to the server, while `protocol-error` are replies to syntactically incorrect calls. See Section 2.5 [Protocol Error Messages], page 16, for more information about `protocol-error`.

If a call cannot complete successfully, LysKOM will respond with an `error-reply`. The meaning of `error-code` and `error-status` varies depending on the request; see Chapter 6 [Error Codes], page 117, and the documentation for each call, for a list of available `error-code` values and what they mean.

A client should be prepared for any error code in response to any call, no matter if the response makes any sense or not. The value returned in `error-status` was more or less undefined before protocol version 10. For protocol version 10, the meaning of `error-status` is defined in this document.

The meaning of `error-status` can be modified by any call. In particular the calls that deal with Misc-Info lists set `error-status` to the index of the misc item that caused the error (if the error was caused by a misc item.)

Client should handle the error messages listed with each call in a graceful manner. In addition, the following error types should always be handled gracefully: `temporary-failure`, `internal-error`, `feature-disabled`, `not-implemented`, `obsolete-call`, `ldb-error`, `out-of-memory`. Client should also be able to handle any error in any situation without choking completely since bugs might cause the wrong error message to be sent or new errors might be added later on.

The server may send asynchronous messages (see Chapter 5 [Asynchronous Messages], page 113) to the client at any time (but not in the middle of a reply). Two important asynchronous messages are `async-new-text` (see Section 5.14 [async-new-text], page 116) and `async-send-message` (see Section 5.11 [async-send-message], page 115).

2.5 Protocol Error Messages

Protocol error messages are sent in reply to syntactically incorrect calls to the server. These should never appear in the client-server dialog. If they do, there is probably a bug in the client.

"%% LysKOM protocol error."
: The client has sent a request that does not comply with protocol A. The server will attempt to recover from this error, but additional calls may be silently lost, and the server may send replies that the client does not expect, and in some cases recovery may not be possible. This error is also returned when the client sends an array that is longer than the maximum allowed by the server.

"%%Insane token length."
: The client has sent a single token that is insanely long. Typically this means that the client has sent several kilobytes worth of digits, or something similar. This is never returned for long strings. The server will automatically disconnect a client who sends a token with an insane length.

"%%Insane array size."
: The client has send an array with a negative length. The server is not easily fooled. The server will automatically disconnect a client who sends an array with an insane length.

3 LysKOM Data Types

In this section the data types specific to LysKOM are defined. Most of these will probably make very little sense until you know what calls there are. This section does not include the server calls or asynchronous messages, even though these are also data types.

Since the types defined here are all based on the simple types, the definitions are more concise in this section.

3.1 Common Types

The types defined in this section are fairly simple and used in many of the more complex data types.

3.1.1 Time

```
Time ::=
      ( seconds          :       INT32;
        minutes          :       INT32;
        hours            :       INT32;
        day              :       INT32;
        month            :       INT32;
        year             :       INT32;
        day-of-week      :       INT32;
        day-of-year      :       INT32;
        is-dst           :       BOOL;
      )
```

`Time` is used to specify times in several data structures. The fields `seconds`, `minutes` and `hours` give wall clock time. `day` is the day of month and `month` is the current month, starting with zero for January. `year` is the number of years since 1900. `day-of-week` is the current weekday, with zero used for Sunday. `day-of-year` is how many days of the year have passed starting with zero and `is-dst` is true when the time indicated is daylight savings time.

When the server receives a `Time` structure from a client it ignores the `day-of-week` and `day-of-year` fields.

All times are expressed in the time zone of the server.

3.1.2 Conference Numbers

```
Conf-No          ::=     INT16;
```

This type denotes a conference number.

3.1.3 Text Numbers

```
Text-No          ::=     INT32;
Local-Text-No    ::=     INT32;
Text-List        ::=
      ( first-local-no   :       Local-Text-No;
        texts            :       ARRAY Text-No;
```

```
        )
```

These three types are used to indicate articles in the LysKOM database. `Text-No` is a global text number and `Local-Text-No` a local text number. `Text-List` is used when a mapping from local to global numbers are required.

3.1.4 Person and Session Numbers

```
Pers-No         ::=     Conf-No;
Session-No      ::=     INT32;
```

`Pers-No` is used to indicate a person. `Session-No` is used in a few data structures relating to information about active LysKOM sessions.

3.2 Auxiliary Information

```
Aux-No   ::=    INT32;

Aux-Item ::=
      ( aux-no      :       Aux-No;
        tag         :       INT32;
        creator     :       Pers-No;
        created-at  :       Time;
        flags       :       Aux-Item-Flags;
        inherit-limit :     INT32;
        data        :       HOLLERITH;
      )

Aux-Item-Input ::=
      ( tag         :       INT32;
        flags       :       Aux-Item-Flags;
        inherit-limit :     INT32;
        data        :       HOLLERITH;
      )


Aux-Item-Flags ::= BITSTRING
      ( deleted;
        inherit;
        secret;
        hide-creator;
        dont-garb;
        reserved2;
        reserved3;
        reserved4;
      )
```

Aux-Item-Input contains a subset of the fields of an Aux-Item. It is used when the client wants to send an Aux-Item to the server, and it only contains the elements that the client can affect. The fields in Aux-Item and Aux-Item-Input have the following meaning:

`aux-no` The number of the item within the list where it is found. This number together with a conference or text number uniquely identifies a particular aux-item. (There is also a global list of Aux-Items for the server. That list is manipulated via the `modify-system-info` request (see Section 4.96 [modify-system-info], page 103), and when using that request the `aux-no` is enough to uniquely identify the aux-item.)

This field is not present in `Aux-Item-Input`. It is assigned by the server.

`tag` The item tag. The tag determines what the data means. See Chapter 7 [Aux-Item Types], page 123.

`creator` The person who created the item, or zero if the item was created anonymously or if the owner is being withheld.

This field is not present in `Aux-Item-Input`. It is assigned by the server.

`created-at`
The time when the item was created.

This field is not present in `Aux-Item-Input`. It is assigned by the server.

`flags` The item flags (see below).

`inherit-limit`
Determines how many times (how deep) an item may be inherited. If zero, the item is inherited an unlimited number of times. If nonzero it is *one more* than the number of additional times the item may be inherited. Thus, 1 means that inheritance will be blocked and 2 means that the item will be inherited only to the next level.

`data` The item data.

The flags that can be set on an aux-item have the following meaning:

`deleted` The item has been deleted, and should not be used for anything.

`inherit` The item will be inherited (if the inherit-limit field allows it.)

`secret` The item will not be revealed to anyone but the item's creator and supervisors of the creator.

`hide-creator`
The item creator will be withheld from everyone but the item's creator and supervisors of the creator.

`dont-garb`
The object the item is set on will not be garbage-collected.

3.3 Conference Types

```
Conf-Type ::= BITSTRING
      ( rd-prot;
        original;
        secret;
        letterbox;
```

```
      )

Extended-Conf-Type ::=  BITSTRING
      ( rd-prot;
        original;
        secret;
        letterbox;
        allow-anonymous;
        forbid-secret;
        reserved2;
        reserved3;
      )

Any-Conf-Type   ::=     Conf-Type | Extended-Conf-Type;
```

These types are used to specify the type of a conference. `Conf-Type` is used in data types and calls that were created before version 8.0 of the protocol and has been augmented in `Extended-Conf-Type`. The type `Any-Conf-Type` is used when either is admissible.

The bits have the following meaning (see Section 1.2 [Conferences], page 3, for more info.)

`rd-prot` If unset anyone can add themselves as members to the conference.

`original` If set, comments are not allowed in the conference.

`secret` If set the conference is secret. It's existence will only be revealed to members and supervisors.

`letterbox`
: Set if the conference is a person's mailbox.

`allow-anonymous`
: Set if anonymous articles are allowed in the conference.

`forbid-secret`
: If set, secret members cannot be added to the conference.

`reserved2`
`reserved3`
: Reserved for future use. The values of these bits should be never be modified or used by clients who do not know their meaning. When a new conference is created these should always be set to zero.

3.4 Conference Search Results

```
Conf-Z-Info ::=
      ( name    :       HOLLERITH;
        type    :       Conf-Type;
        conf-no :       Conf-No;
      )
```

These types are used for the result of some calls that search for conferences based on their names.

3.5 Conference Status Types

```
Garb-Nice ::= INT32;

Conference-Old ::=
        ( name                  :       HOLLERITH;
          type                  :       Conf-Type;
          creation-time         :       Time;
          last-written          :       Time;
          creator               :       Pers-No;
          presentation          :       Text-No;
          supervisor            :       Conf-No;
          permitted-submitters  :       Conf-No;
          super-conf            :       Conf-No;
          msg-of-day            :       Text-No;
          nice                  :       Garb-Nice;
          no-of-members         :       INT16;
          first-local-no        :       Local-Text-No;
          no-of-texts           :       INT32;
        )


Conference ::=
        ( name                  :       HOLLERITH;
          type                  :       Extended-Conf-Type;
          creation-time         :       Time;
          last-written          :       Time;
          creator               :       Pers-No;
          presentation          :       Text-No;
          supervisor            :       Conf-No;
          permitted-submitters  :       Conf-No;
          super-conf            :       Conf-No;
          msg-of-day            :       Text-No;
          nice                  :       Garb-Nice;
          keep-commented        :       Garb-Nice;
          no-of-members         :       INT16;
          first-local-no        :       Local-Text-No;
          no-of-texts           :       INT32;
          expire                :       Garb-Nice;
          aux-items             :       ARRAY Aux-Item;
        )


UConference ::=
        ( name                  :       HOLLERITH;
          type                  :       Extended-Conf-Type;
          highest-local-no      :       Local-Text-No;
          nice                  :       Garb-Nice;
        )
```

These three types are used to specify information about a conference. `Garb-Nice` is a quantity used to specify how long articled are kept in a conference before being removed. `Conference` is the full information about a conference and `UConference` is brief information about a conference.

The fields of `Conference` are

`name` The name of this conference.

`type` The type of the conference.

`creation-time`
: The date and time when the conference was created.

`last-written`
: The date when something was last written in the conference.

`creator` The person who created the conference.

`presentation`
: The article containing the conference presentation or zero if the conference has no presentation.

`supervisor`
: The conference[1] who supervises this conference.

`permitted-submitters`
: The conference whose members[2] may submit articles to the conference, or zero if anyone may do so.

`super-conf`
: This field has two unrelated uses. If the `original` bit is set, clients should send comments to this conference (see Section A.2.4 [Recipients of comments], page 143). If the `permitted-submitters` field rejects a text[3], the server will silently send the text to the `super-conf` conference instead. This redirection mechanism is applied repeatedly until conference that accepts the text is found, or an implementation-defined limit is reached.

`msg-of-day`
: The conference notice, if any.

`nice` The number of days an article should be kept before being removed from the conference.

`keep-commented`
: A text that has comments no older than this number of days will not be removed from the conference.

---

[1] The `supervisor` may be a person, in which case the members of that person's mailbox become supervisors.

[2] `permitted-submitters` can be a person, in which case all persons who are members of the associated mailbox are allowed to submit articles to the conference.

[3] This is true if the text is created using `create-text` (see Section 4.87 [create-text], page 97), `create-anonymous-text` (see Section 4.88 [create-anonymous-text], page 99), `create-anonymous-text-old` (see Section 4.60 [create-anonymous-text-old], page 83) or `create-text-old` (see Section 4.29 [create-text-old], page 61). Future requests that create texts may have other semantics.

`no-of-members`
: The number of members of this conference.

`first-local-no`
: The local number of the oldest existing article in the conference.

`no-of-texts`
: The number of articles in the conference.

`expire`
: This field will be used to control when a conference expires. It is not used at the moment, and should be set to zero for future compatibility.

`aux-items`
: The conference's aux item list.

The fields of `UConference` are

`name`
: The name of this conference.

`type`
: The conference type. Note that this is an extended conference type, unlike the type field of `Conference`.

`highest-local-no`
: The local number of the newest article in the conference.

`nice`
: The number of days an article should be kept before being removed from the conference.

3.6 Archaic way to list conferences

The result of request number 12, lookup-name, cannot be expressed in the grammar used in this document. This is as close as it gets:

```
Conf-List-Archaic ::=
     ( conf-nos    : ARRAY Conf-No;
       conf-types  : ARRAY Conf-Type; ! Sans n; see below
     )
```

The two arrays `conf-nos` and `conf-types` are always the same size. For some obscure reason the size of the second array is not actually transmitted. See also the example in Section 4.13 [lookup-name], page 48.

3.7 Mapping Local to Global Text Numbers

```
Text-Mapping ::=
      ( range-begin           :    Local-Text-No;
        range-end             :    Local-Text-No;
        later-texts-exists    :    BOOL;
        block                 :    Local-To-Global-Block;
      )


Local-To-Global-Block ::= SELECTION
      ( 0=sparse sparse-block :   ARRAY Text-Number-Pair;
```

```
      1=dense  dense-block  :   Text-List;
    )
```

```
Text-Number-Pair ::=
    ( local-number          :   Local-Text-No;
      global-number         :   Text-No;
    )
```

A `Text-Mapping` is used when the client needs to look up which global `Text-No` that corresponds to a `Local-Text-No`. The client uses `local-to-global` (see Section 4.104 [local-to-global], page 109) to ask for information about a few texts starting a a certain local text number, and the server returns the information in a `Text-Mapping`.

`range-begin`
: The first local text number that the client asked about.

`range-end`
: The first local text number that the reply doesn't say anything about. This `Text-Mapping` tells the client about all existing texts from `range-begin` to (but not including) `range-end`.

`later-texts-exists`
: This is true if there are more texts in the conference after this block.

`block`
: The block can be sent i two formats. The server is free to choose which format to use as it pleases; clients must be prepared for any of them.

  - The *sparse* format is useful when many local text number no longer exists. It starts with a `0` that indicates that the sparse format is used, and is followed by an array of `Text-Number-Pair`. The array will always be sorted so that `local-number` always increases.
  - The *dense* format is good when most of the local text numbers exist. It starts with a `1` that indicates that the dense format is used, and is followed by a `Text-List`. The `Text-List` contains `first-local-no` and an array of `Text-No`. The local text number `first-local-no` corresponds to the first `Text-No` in the array, `first-local-no` + 1 corresponds to the second entry in the array, and so on. The array contains a zero to indicate that a certain local text number doesn't exist.

3.8 Server Information

```
Info ::=
    ( version             :   INT32;
      conf-pres-conf      :   Conf-No;
      pers-pres-conf      :   Conf-No;
      motd-conf           :   Conf-No;
      kom-news-conf       :   Conf-No;
      motd-of-lyskom      :   Text-No;
      aux-item-list       :   ARRAY Aux-Item;
    )
```

```
Info-Old ::=
      ( version             :   INT32;
        conf-pres-conf      :   Conf-No;
        pers-pres-conf      :   Conf-No;
        motd-conf           :   Conf-No;
        kom-news-conf       :   Conf-No;
        motd-of-lyskom      :   Text-No;
      )


Version-Info ::=
      ( protocol-version    :   INT32;
        server-software     :   HOLLERITH;
        software-version    :   HOLLERITH;
      )
```

These data types contain information about the LysKOM server. The fields of `Info` and `Info-Old` are

`version` The server version encoded as a number *aabbcc* where *aa* is the major version number, *bb* the minor number and *cc* the secondary minor version. For instance, `10607` is version 1.6.7 of the server. If greater than 10699 the `get-version-info` (see Section 4.76 [get-version-info], page 91) should be used instead.

`conf-pres-conf`
The conference that contains conference presentations.

`pers-pres-conf`
The conference that contains person presentations.

`motd-conf`
The conference that contains conference and person notices.

`kom-news-conf`
The conference that contains news about LysKOM.

`motd-of-lyskom`
The number of an article to display when LysKOM is entered or zero if there is none.

`aux-item-list`
(Not present in `Info-Old`.) A list of aux-items that belong to the server.

The fields of `Version-Info` are:

`protocol-version`
The version of protocol A the server is using. This may be used to ascertain which calls are available.

`server-software`
Human-readable name of the server software.

`software-version`
Human-readable name of the server software version.

3.9 Person Status Types

```
Person ::=
      ( username                 :       HOLLERITH;
        privileges               :       Priv-Bits;
        flags                    :       Personal-Flags;
        last-login               :       Time;
        user-area                :       Text-No;
        total-time-present       :       INT32;
        sessions                 :       INT32;
        created-lines            :       INT32;
        created-bytes            :       INT32;
        read-texts               :       INT32;
        no-of-text-fetches       :       INT32;
        created-persons          :       INT16;
        created-confs            :       INT16;
        first-created-local-no   :       INT32;
        no-of-created-texts      :       INT32;
        no-of-marks              :       INT16;
        no-of-confs              :       INT16;
      )


Personal-Flags ::= BITSTRING
      ( unread-is-secret;
        flg2;
        flg3;
        flg4;
        flg5;
        flg6;
        flg7;
        flg8;
      )



Priv-Bits ::= BITSTRING
      ( wheel;
        admin;
        statistic;
        create-pers;
        create-conf;
        change-name;
        flg7;
        flg8;
        flg9;
        flg10;
        flg11;
        flg12;
```

```
  flg13;
  flg14;
  flg15;
  flg16;
)
```

These types are used to specify information about persons. `Person` contains the information about a person, `Personal-Flags` contains flags set by the user and `Priv-Bits` contains the person's privileges (see Section 1.6 [Security], page 8).

The fields of `Person` are

`username` The name of the user.

`privileges`
: The privileges of the person.

`flags` Flags set by the user.

`last-login`
: The time when the person last logged on or off.

`user-area`
: The text number of the person's user area or zero if the person has no user area.

`total-time-present`
: The number of seconds the person has been using LysKOM.

`sessions` The number of sessions the person has initiated.

`created-lines`
: The number of lines of articles the person has written.

`created-bytes`
: The number of characters the person has written.

`read-texts`
: The number of articles the person has read.

`no-of-text-fetches`
: The number of texts the person has retrieved from the server.

`created-persons`
: The number of other persons this person has created.

`created-confs`
: This holds the number of conferences created by the person.

`first-created-local-no`
: The local number of the earliest existing article written by the person. The local number applies to a local-to-global mapping containing all articles written by the person.

`no-of-created-texts`
: This holds the number of articles written by the person, not counting the articles created before the earliest existing article written by the person. To get the

true number of created articles, compute `first-created-local-no` + `no-of-created-texts` - 1.

`no-of-marks`
: The number of marked texts this person has.

`no-of-confs`
: The number of conferences the person is a member of.

The fields of `Personal-Flags` are

`unread-is-secret`
: If this is set, only the person and his supervisors can see how many texts this persons have read in a certain conference. The server will clear the `read-texts`, `last-time-read` and `last-text-read` fields of `Membership` and `Membership-Old` when other persons asks about the information.

`flg2`
`flg3`
`flg4`
`flg5`
`flg6`
`flg7`
`flg8`
: Reserved for the future. Set them to 0, and ignore the values they have.

3.10 Membership Information

```
Membership-Type ::=     BITSTRING
      ( invitation;
        passive;
        secret;
        reserved1;
        reserved2;
        reserved3;
        reserved4;
        reserved5;
      )
```

The `Membership-Type` type contains flags that modify a membership. It is passed as part of both `Member` and `Membership`. The flags are:

`invitation`
: The member has been invited, but has not yet accepted membership. Clients should set this flag when adding other users as members. The server may force this flag to be set when adding another person as a member of a conference.

`passive`
: The member is not actively participating in the conference. Passive members do not receive group messages and should not be displayed as active members by clients.

`secret`
: The member does not wish to disclose the membership. Secret memberships and members are cleared before being returned to a person who is not a supervisor of the member or has sufficient privileges enabled.

The remaining flags in the `Membership-Type` structure are reserved and should be set to false on all new memberships and retained on all existing memberships.

```
Member          ::=
      ( member          :       Pers-No;
        added-by        :       Pers-No;
        added-at        :       Time;
        type            :       Membership-Type;
      )
```

The `Member` structure encodes information about a member in a conference. It is returned by the `get-members` call (see Section 4.102 [get-members], page 107). The fields of a `Member` structure are

`member` The person who is a member of the conference.

`added-by` The person who created the membership. This field is zero only if the membership was created before protocol version 10 was introduced.

`added-at` The time when the membership was created. This field is meaningless if added-by is zero.

`type` Flags modifying the membership.

The contents of a `Member` structure can be cleared (all fields set to zero) if the person requesting the record has insufficient privileges to see the contents of the structure.

```
Membership-Old ::=
      ( last-time-read  :       Time;
        conference      :       Conf-No;
        priority        :       INT8;
        last-text-read  :       Local-Text-No;
        read-texts      :       ARRAY Local-Text-No;
      )


Membership ::=
      ( position        :       INT32;
        last-time-read  :       Time;
        conference      :       Conf-No;
        priority        :       INT8;
        last-text-read  :       Local-Text-No;
        read-texts      :       ARRAY Local-Text-No;
        added-by        :       Pers-No;
        added-at        :       Time;
        type            :       Membership-Type;
      )
```

The `Membership` structure encodes information about a person's membership in a conference. It is returned by the `query-read-texts` (see Section 4.99 [query-read-texts], page 104) and `get-membership` calls (see Section 4.100 [get-membership], page 105).

`position` The position of this membership in the membership list.

`last-time-read`
: The time when the person last read anything from the conference. This will contain the start of the epoch if the `unread-is-secret` bit of the `Personal-Flags` is set and you don't have enough privileges to get the info anyhow.

`conference`
: The conference this membership data pertains to.

`priority`
: The priority the person has assigned to the conference. The higher the number, the higher the priority. In the past, priority zero indicated a passive membership. This usage is now obsolete.

`last-text-read`
: The local number of last text read in the conference. This will be set to 0 if `unread-is-secret` is set, unless you have access to the data anyhow.

`read-texts`
: Additional texts beyond `last-text-read` that have also been read. This will be empty if `unread-is-secret` is set, unless you have access to the data anyhow.

`added-by`
: The person who created the membership. This field is zero if the membership was created before protocol version 10 was introduced.

`added-at`
: The time when the membership was created. This field is meaningless if added-by is zero.

`type`
: Flags modifying the membership.

A membership record may be cleared by the server (all fields set to zero) if the person requesting the membership has insufficient privileges to see the contents membership, but has sufficient privileges to know about the person.

3.11 Article Marks

```
Mark ::=
        ( text-no       :       Text-No;
          type          :       INT8;
        )
```

This data type hold information about a person's marks on articles.

The fields of `Mark` are

`text-no`
: The text number marked.

`type`
: The mark value.

Before version eight of protocol A, the meaning of the mark value was unspecified. Work is underway to specify the meaning of certain mark values.

3.12 Article Information

```
Misc-Info ::= SELECTION
        ( 0=recpt       recipient       :       Conf-No;
          1=cc-recpt    cc-recipient    :       Conf-No;
          2=comm-to     comment-to      :       Text-No;
          3=comm-in     commented-in    :       Text-No;
          4=footn-to    footnote-to     :       Text-No;
          5=footn-in    footnoted-in    :       Text-No;
          6=loc-no      local-no        :       Local-Text-No;
          7=rec-time    received-at     :       Time;
          8=sent-by     sender          :       Pers-No;
          9=sent-at     sent-at         :       Time;
         15=bcc-recpt   bcc-recipient   :       Conf-No;
       )


Info-Type ::= ENUMERATION-OF(Misc-Info)


Text-Stat-Old ::=
      ( creation-time   :       Time;
        author          :       Pers-No;
        no-of-lines     :       INT32;
        no-of-chars     :       INT32;
        no-of-marks     :       INT16;
        misc-info       :       ARRAY Misc-Info;
      )


Text-Stat ::=
      ( creation-time   :       Time;
        author          :       Pers-No;
        no-of-lines     :       INT32;
        no-of-chars     :       INT32;
        no-of-marks     :       INT16;
        misc-info       :       ARRAY Misc-Info;
        aux-items       :       ARRAY Aux-Item;
      )
```

These two structures contain information about a single article. `Text-Stat` contains core information about the article and `Misc-Info` contains miscellaneous information related to the article.

A `Text-Stat` consists of the following:

`creation-time`
: The time when the article was created.

`author`
: The author of the article.

`no-of-lines`
: The number of lines in the article.

`no-of-chars`
: The number of characters in the article.

`no-of-marks`
: The number of marks added to this article by persons.

`misc-info`
: The `Misc-Info` list for this article.

`aux-items`
: The list of aux items for this article.

`Misc-Info`, when sent to the client, is sent in a particular order (see Section 1.4 [The Misc-Info List], page 4). The variants `Misc-Info` are (briefly):

`recpt` Used to specify recipients of the article.

`cc-recpt` Specifies recipients who have received a copy of the article but who will not receive comments.

`comm-to` Specifies an article this article is a comment to.

`comm-in` Specifies an article in which there are comments to this article.

`footn-to` Specifies an article this article is a footnote to.

`footn-in` Specifies an article to which this article is a footnote.

`loc-no` Specifies the local text number of this article in the conference specified by `recpt`, `cc-recpt` or `bcc-recpt`.

`rec-time` Specifies the time when this article was received by the conference specified by `recpt`, `cc-recpt` or `bcc-recpt`.

`sent-by` Specifies who sent this article to the conference specified by `recpt`, `cc-recpt` or `bcc-recpt`.

`sent-at` Specifies when the article was sent to the conference specified by `recpt`, `cc-recpt` or `bcc-recpt`.

`bcc-recpt`
: Specifies a blind carbon copy recipient. This item is only accepted by protocol version 10 servers and is only sent in responses and messages introduced in protocol version 10 or later.

3.13 Who Information

```
Who-Info-Old ::=
      ( person                  :       Pers-No;
        working-conference      :       Conf-No;
        what-am-i-doing         :       HOLLERITH;
      )
```

```
Who-Info ::=
      ( person                  :       Pers-No;
        working-conference      :       Conf-No;
        session                 :       Session-No;
        what-am-i-doing         :       HOLLERITH;
        username                :       HOLLERITH;
      )

Who-Info-Ident ::=
      ( person                  :       Pers-No;
        working-conference      :       Conf-No;
        session                 :       Session-No;
        what-am-i-doing         :       HOLLERITH;
        username                :       HOLLERITH;
        hostname                :       HOLLERITH;
        ident-user              :       HOLLERITH;
      )
```

These structures are used to retrieve information on who is currently using LysKOM. All the requests using these types are obsolete, but `Who-Info` is used by `async-i-am-on` (see Section 5.5 [async-i-am-on], page 114).

The fields of `Who-Info-Old` are

`person` The person the information is about.

`working-conference`
The conference the person is currently in.

`what-am-i-doing`
A client-supplied string saying what the person is doing.

The fields of `Who-Info` are

`person` The person the information is about.

`working-conference`
The conference the person is currently in.

`session` The person's session number.

`what-am-i-doing`
A client-supplied string saying what the person is doing.

`username` The name of the "real" user constructed from `hostname` and `ident-user` (see below) and information from the client.

The fields of `Who-Info-Ident` are

`person` The person the information is about.

`working-conference`
The conference the person is currently in.

`session` The person's session number.

`what-am-i-doing`
A client-supplied string saying what the person is doing.

`username` The name of the "real" user constructed from `hostname` and `ident-user`.

`hostname` The host the connection originated at.

`ident-user`
The user name according to the Ident dæmon at the user's machine or "unknown" if Ident was not used.

3.14 Session Information

```
Session-Info ::=
      ( person                  :       Pers-No;
        working-conference      :       Conf-No;
        session                 :       Session-No;
        what-am-i-doing         :       HOLLERITH;
        username                :       HOLLERITH;
        idle-time               :       INT32;
        connection-time         :       Time;
      )

Session-Info-Ident ::=
      ( person                  :       Pers-No;
        working-conference      :       Conf-No;
        session                 :       Session-No;
        what-am-i-doing         :       HOLLERITH;
        username                :       HOLLERITH;
        hostname                :       HOLLERITH;
        ident-user              :       HOLLERITH;
        idle-time               :       INT32;
        connection-time         :       Time;
      )


Static-Session-Info ::=
      (
        username                : HOLLERITH;
        hostname                : HOLLERITH;
        ident-user              : HOLLERITH;
        connection-time         : Time;
      )


Session-Flags ::= BITSTRING
      ( invisible;
        user-active-used;
        user-absent;
        reserved3;
        reserved4;
        reserved5;
        reserved6;
```

```
        reserved7;
    )


Dynamic-Session-Info ::=
    ( session                 :       Session-No;
      person                  :       Pers-No;
      working-conference      :       Conf-No;
      idle-time               :       INT32;
      flags                   :       Session-Flags;
      what-am-i-doing         :       HOLLERITH;
    )
```

These data types give information about a particular LysKOM session. The types `Session-Info` and `Session-Info-Ident` have been superseded by `Static-Session-Info` and `Dynamic-Session-Info`. The static session info represents information about a session that does not change during the lifetime of the session. Therefore static session infos can be aggressively cached by the client.

The fields of `Session-Info` are

`person` The person using this session.

`working-conference`
: The conference the session is currently in.

`session` The number of this session.

`what-am-i-doing`
: A client-supplied string saying what the person is currently doing.

`username` The name of the "real" user (see `Who-Info` above.)

`idle-time`
: The number of seconds since `user-active` was used by this session (see Section 4.83 [user-active], page 96), or since the session was created if `user-active` has not been used yet. See also `user-active-used` below.

`connection-time`
: The time when the connection was initiated. This is not the same as the amount of time the person has been on.

The fields of `Session-Info-Ident` are

`person` The person using this session.

`working-conference`
: The conference the session is currently in.

`session` The number of this session.

`what-am-i-doing`
: A client-supplied string saying what the person is currently doing.

`username` The name of the "real" user (see `Who-Info-Ident` above.)

`hostname` The host the connection originated at.

`ident-user`
: The user name according to the Ident dæmon at the user's machine or "unknown" if Ident was not used.

`idle-time`
: The number of seconds since `user-active` was used by this session (see Section 4.83 [user-active], page 96), or since the session was created if `user-active` has not been used yet. See also `user-active-used` below.

`connection-time`
: The time when the connection was initiated. This is not the same as the amount of time the person has been on.

The fields of `Static-Session-Info` are

`username` The name of the "real" user (see `Who-Info-Ident` above.)

`hostname` The host the connection originated at.

`ident-user`
: The user name according to the Ident dæmon at the user's machine or "unknown" if Ident was not used.

`connection-time`
: The time when the connection was initiated. This is not the same as the amount of time the person has been on.

The bits in `Session-Flags` are:

`invisible`
: When true, the user requested an invisible session (see Section 4.63 [login], page 85). Sessions where no-one is logged in also have the `invisible` flag set.

`user-active-used`
: When true, the `user-active` call (see Section 4.83 [user-active], page 96) has been issued by the session, which in turn means that `idle-time` fields of `Dynamic-Session-Info`, `Session-Info` and `Session-Info-Ident` are valid. When false, the `idle-time` fields contains the number of seconds since the session was created.

`user-absent`
: This flag is currently not used.

The fields of a `Dynamic-Session-Info` are

`session` The session number of the session.

`person` The person currently logged on, or zero if the session has not yet logged on.

`working-conference`
: The conference the session is currently in.

`idle-time`
: The number of seconds since `user-active` was used by this session (see Section 4.83 [user-active], page 96), or since the session was created if `user-active` has not been used yet. See also `user-active-used` above.

`flags` The dynamic session flags (see above.)

`what-am-i-doing`
What the client is doing. This string is set by the client.

4 Protocol Requests

This chapter documents all calls that can be made to the server. All calls are annotated with the protocol version in which they appeared and their current status.

The heading for each call looks something like this:

create-person-old [5] (1) Obsolete (10)

The heading consists of several parts:

The name: '`create-person-old`'
: This is the name of the call. The name is not considered part of the protocol. It may change in future versions of this document. Since the name is never sent over the network it doesn't matter that much.

The call number: '`[5]`'
: The call number is what really matters, since it is sent over the network. It will never change.

Introduced: '`(1)`'
: The protocol version when the call was first implemented. Some calls added more functionality in a later protocol version. The description for those calls describes such changes.

The status: '`Obsolete`'
: The status of the call (see below).

Made obsolete: '`(10)`'
: This figure is only present for some obsolete calls, and it states in which protocol version the call was obsoleted.

The status of a call can be any of:

'`Experimental`'
: The call is experimental. No client should rely on the existence of this call. Experimental calls that are useful will usually become recommended in future versions.

'`Recommended`'
: The call is a standard call. Clients are recommended to use these calls rather than experimental or obsolete ones. Servers are required to implement all recommended calls.

'`Obsolete`'
: The call should no longer be used by clients. Servers should implement these, or they will be incompatible with old client versions. *Please note:* the documentation for the obsolete calls may be incomplete. Many of them perform compatibility magic to ensure that they never return anything that old clients don't expect. This compatibility magic is often documented, but we may have forgotten to document it in some places.

A note about the examples: The examples consist of a number of calls and replies. Calls are set in a normal typewriter font. Replies are set in a slanted typewriter font. Extra newlines are sometimes inserted in the examples to avoid overly long lines.

4.1 login-old [0] (1) Obsolete

```
login-old [0] (( person : Pers-No;
                 passwd : HOLLERITH )) -> ( );
```

Log in as a person. This call has been replaced by `login` (see Section 4.63 [login], page 85).

Error codes

`undefined-person`
: `person` is not an existing person.

`login-disallowed`
: Logins are not allowed and `person` does not have the `wheel` bit set.

`invalid-password`
: `passwd` is not the correct password for `person`. The `error-status` indicates the person number.

4.2 logout [1] (1) Recommended

```
logout [1] ( )
        -> ( );
```

Log out from LysKOM.

This call does not disconnect the session; use `disconnect` for that (see Section 4.56 [disconnect], page 80).

After issuing a `logout` call the client can reconnect as the same or a different person using the `login` call (see Section 4.63 [login], page 85).

For a client that needs to log in as several different users, issuing multiple logout and login requests during one session is faster and places less load on the server than does creating new sessions.

Error codes

This call never fails.

4.3 change-conference [2] (1) Recommended

```
change-conference [2] ( conference : Conf-No )
        -> ( ) ;
```

Change current conference of a session. This call used to be called pepsi (the name was a very obscure and not very funny joke.)

Error codes

`login-first`
: The session is not logged in yet.

`undefined-conference`
: Conference `conference` does not exist or is secret.

`conference-zero`
: `conference` is zero.

`not-member`
: The user currently logged in is not a member of `conference`.

4.4 change-name [3] (1) Recommended

```
change-name [3] (( conference : Conf-No;
                   new-name   : HOLLERITH ))
        -> ( ) ;
```

This call changes the name of a conference or a person. To change the name of a conference the session issuing the call must be logged in as a person who either has special privileges or is the supervisor of the conference.

Error codes

`login-first`
: The session is not logged in yet.

`undefined-conference`
: Conference `conference` does not exist or is secret.

`conference-zero`
: `conference` is zero.

`permission-denied`
: Permission denied. The `change-name` bit is not set or the user does not have enough access to `conference`.

`conference-exists`
: `new-name` is already occupied by another conference.

`string-too-long`
: `new-name` is too long for a valid conference name.

`bad-name` There are invalid characters in `new-name`.

4.5 change-what-i-am-doing [4] (1) Recommended

```
change-what-i-am-doing [4] ( what-am-i-doing : HOLLERITH )
        -> ( );
```

This call tells the server what the logged-in user is doing. The string is usually displayed when a user requests that a client list who is using LysKOM. Clients are encouraged to use this call creatively.

Error codes

`string-too-long`
: `what-am-i-doing` is too long.

4.6 create-person-old [5] (1) Obsolete (10)

```
create-person-old [5] (( name   : HOLLERITH;
                         passwd : HOLLERITH ))
        -> ( Pers-No );
```

This call requests that the server create a new person with the name and password given as arguments. To create a person the session may have to be logged in as a person with sufficient privileges, if the server is configured that way.

If the session was not logged in an automatic visible login to the new person will be performed.

The new person will be a member of exactly one conference: the associated mailbox. That membership will have priority 255 and (of course) position 0. All flags of the membership will be 0.

Example:

```
1 5 24HLysKOM Statistics Daemon 6Hsecret
=1 6
```

This example creates a new person named "LysKOM statistics Daemon" with the password "secret". The server has returned the person number six for the person.

Error codes

`login-first`
: The session is not logged in and the server does not allow person creation before logging in.

`permission-denied`
: The server does not allow everyone to create person and the person currently logged on does not have the `create-pers` bit set.

`conference-exists`
: There is already a conference named `name`.

`invalid-password`
: The string `passwd` is not a valid password.

4.7 get-person-stat-old [6] (1) Obsolete

```
get-person-stat-old [6] (( person : Pers-No;
                           mask   : INT32 ))
        -> ( Person );
```

This call returns information about a person. If the low bit of `mask` is not set, the empty string is returned instead of the `username` field. This call is obsolete and has been replaced by `get-person-stat` (see Section 4.50 [get-person-stat], page 77).

Error codes

`login-first`
: This call can't be executed without logging in first.

`undefined-person`
: Person `person` does not exist.

`undefined-conference`
: Conference `person` does not exist or is secret.

`conference-zero`
: `person` is zero.

4.8 set-priv-bits [7] (1) Recommended

```
set-priv-bits [7] (( person     : Pers-No;
                     privileges : Priv-Bits )) -> ( );
```

This call sets the privileges of `person` to `privileges`. See Section 1.6 [Security], page 8. To successfully issue this call the session must be logged in as a person with sufficient privileges.

Example:

```
1 7 6 0010000000000000
=1
```

This example sets the privileges of person 6 to nothing but `statistic`. This particular set of privileges might be useful for a person used by a statistics-collecting dæmon.

Error codes

`login-first`
: This call can't be executed without logging in first.

`undefined-person`
: `person` is not a valid person.

`permission-denied`
: The person currently logged in does not have the `wheel` bit set and privilege level set to 6 or higher.

4.9 set-passwd [8] (1) Recommended

```
set-passwd [8] (( person  : Pers-No;
                  old-pwd : HOLLERITH;
                  new-pwd : HOLLERITH )) -> ( );
```

This call is used to set the password of `person` to `new-pwd`. Any person may set it's own password. In addition persons with sufficient privileges may set other persons' passwords. The password of the person currently logged in must match `old-pwd`.

Example:

```
1 8 5 6Hgazonk 7Ha9go8Hz
=1
```

This example sets the password of the LysKOM administrator to “a9go8Hz” provided that the old password was “gazonk”.

Error codes

`login-first`
: This call cannot be executed without logging in.

`undefined-person`
: `person` is not a valid person.

`permission-denied`
: Attempt to change password of another person without being the supervisor of that person and without the `wheel` bit set and privilege level 7 or higher enabled.

`invalid-password`
: `old-pwd` is invalid or `new-pwd` is invalid as a password.

4.10 query-read-texts-old [9] (1) Obsolete (10)

```
query-read-texts-old [9] (( person     : Pers-No;
                            conference : Conf-No ))
        -> ( Membership-Old );
```

This call is used to find the number of unread texts in a conference. The data it returns is actually a membership structure which specifies which texts have been read. It is up to the client to transform the data to a more usable form. `person` is the person being queried and `conference` is the conference in question.

Calling `query-read-texts-old` does not require the session to be logged in.

Example:

```
1 9 6 1
=1 32 5 11 12 7 93 1 193 1 1 20 133 3 { 135 136 137 }
```

This example finds the read texts for user 6 in conference 1. The returned data indicates that the user last read conference 1 (the tenth number) on Monday July 12th, 1993 at 11:05:32 (the first nine numbers), that the person has assigned priority 20 to the conference (the eleventh number) and that all articles up to and including local number 133 plus articles 135, 136 and 137 have been read.

Error codes

`undefined-person`
: `person` does not exist, or no access to person.

`undefined-conference`
: Conference `conference` does not exist, or is secret.

`conference-zero`
: `conference` is zero.

`not-member`
: `person` is not a member of `conference`.

4.11 create-conf-old [10] (1) Obsolete (10)

```
create-conf-old [10] (( name : HOLLERITH;
                        type : Any-Conf-Type ))
        -> ( Conf-No );
```

This call is used to create new conferences. `name` is the name of the new conference and `type` is its type. If successful, the call returns the conference number of the newly created conference.

To use this call the session must have logged in as a user with privileges to create conferences (see Section 1.6 [Security], page 8).

Example:

```
1 50 8
%1 9 0
1 10 13HInlägg }t mig 00001000
=1 8
1 50 8
=1 13HInlägg }t mig 0000
  43 9 17 14 5 96 5 165 1
  43 9 17 14 5 96 5 165 1
  5 0 5 0 5 0 77 0 1 0
```

This example creates a new conference named "Inlägg }t mig"[1] which accepts all users as members and accepts anonymous articles. The server returns 7 as the new conference number.

Error codes

`login-first`
: Login required before issuing this call.

`permission-denied`
: The server does not allow everyone to create a conference and user does not have the `create-conf` bit set.

`conference-exists`
: A conference named `name` already exists.

`bad-name` `name` contains invalid characters.

`string-to-long`
: `name` is too long to be used as a conference name.

`secret-public`
: The conference type `type` has the `secret` bit set, but the `rd-prot` bit is cleared.

[1] This conference is a standard Lysator conference. It's all Padrone's fault.

4.12 delete-conf [11] (1) Recommended

```
delete-conf [11] ( conf : Conf-No ) -> ( );
```

This call deletes the conference `conf` from the LysKOM database. Only privileged users and the supervisors of a conference may delete it. If the conference is a mailbox the corresponding person will also be deleted.

Example:

```
1 50 7
=1 4HTest 1001
  16 4 19 10 5 96 1 161 1
  16 4 19 10 5 96 1 161 1
  7 0 7 0 0 0 77 1 1 0
1 11 7
=1
1 50 7
%1 9 0
```

This example shows the successful deletion of conference number seven.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: `conf` does not exist or is secret.

`conference-zero`
: `conf` is zero.

`permission-denied`
: Not supervisor of `conf` and not enough privileges enabled.

`undefined-person`
: `conf` is a mailbox but does not exist as a person (the database is corrupt.)

4.13 lookup-name [12] (1) Obsolete

```
lookup-name [12] ( name : HOLLERITH )
        -> ( Conf-List-Archaic );
```

This call returns a list of conferences matching the string `name`. lookup-name has been superseded by `lookup-z-name` (see Section 4.77 [lookup-z-name], page 92).

Example:

```
1 12 3Ha d
=1 3 { 217 674 1582 } { 0000 1001 0000 }
2 12 3H:::
=2 0 * *
3 76 3Ha d 1 1
=3 3 { 31HAlkohol- (och annan) drogdebatt 0000 217
       13HAnna Degerman 1001 674
       27HAnarchy discussion (import) 0000 1582 }
4 76 3H::: 1 1
=4 0 *
```

This example shows two attempts to look up a name. The first example looks up '`a d`' and finds three matches (the middle, number 674, is a person). The second example looks up '`:::`' which doesn't match any conference (or person). Call number 3 and 4 issues the same lookup using the `lookup-z-name` call (see Section 4.77 [lookup-z-name], page 92). (The return value for call number 3 has been broken into three lines to fit on the page.)

Error codes

This call always succeeds.

4.14 get-conf-stat-older [13] (1) Obsolete

```
get-conf-stat-older [13] (( conf-no : Conf-No;
                            mask    : INT32 ))
        -> ( Conference-Old );
```

This call retrieves the information associated with conference `conf-no` in the LysKOM server. This call should no longer be used; use `get-conf-stat` instead (see Section 4.92 [get-conf-stat], page 102).

The `mask` argument determines if the name is returned or not. If the lowest bit is 1 the name is returned, otherwise the empty string is returned instead of the name.

Error codes

`undefined-conference`
: `conf-no` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

4.15 add-member-old [14] (1) Obsolete (10)

```
add-member-old [14] (( conf-no  : Conf-No;
                       pers-no  : Pers-No;
                       priority : INT8;
                       where    : INT16 ))
        -> ( );
```

Make the person `pers-no` a member of conference `conf-no`. The membership priority is set to `priority` and its position in the membership list is set to `where`.

This call can be used to change the priority and position of a conference in the person's membership list if the person is already a member of the conference.

In protocol version 10, setting the priority to zero sets the passive bit in the membership. The actual priority is not changed.

Example:

```
1 46 119 0 10 0
=1 1 { 49 14 17 13 8 91 5 255 1 119 255 0 0 * }
1 14 1 119 250 0
=1
1 46 119 0 10 0
=1 2 { 52 30 14 11 5 96 2 162 1 1 250 0 0 *
       49 14 17 13 8 91 5 255 1 119 255 0 0 * }
```

This example makes person 119 (me) a member of conference number 1. The priority is set to 250 and the conference is placed first in the membership list. The first and last calls of the example show the membership list for person 119 before and after the call.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: Conference `conf-no` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

`undefined-person`
: Person `pers-no` does not exist

`access-denied`
: Not enough permissions or privileges to add members to conference `conf-no`.

`permission-denied`
: Person `pers-no` is already a member of conference `conf-no`, but the person logged on is not a supervisor and does not have enough privileges to change the priorities of person `pers-no`.

4.16 sub-member [15] (1) Recommended

```
sub-member [15] (( conf-no : Conf-No;
                   pers-no : Pers-No ))
        -> ( );
```

Removes the person `pers-no` from the membership list of conference `conf-no` and remove the conference from the person's list of memberships.

Example:

```
1 46 5 0 100 0
=1 2 { 44 14 19 10 5 96 1 161 1 1   0 0 0 *
       49 14 17 13 8 91 5 255 1 5 255 0 0 * }
1 15 1 5
=1
1 46 5 0 100 0
=1 1 { 49 14 17 13 8 91 5 255 1 5 255 0 0 * }
```

This example shows how person 5 is removed from conference one. The calls to get-membership demonstrate the effects on the LysKOM database.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: Conference `conf-no` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

`undefined-person`
: Person `pers-no` does not exist.

`not-member`
: Person `pers-no` is not a member of conference `conf-no`.

`permission-denied`
: Not supervisor of conference `conf-no` or not supervisor of person `pers-no` and not enough privileges to issue the call anyway. `error-status` contains the conference number.

4.17 set-presentation [16] (1) Recommended

```
set-presentation [16] (( conf-no : Conf-No;
                         text-no : Text-No ))
        -> ( );
```

This call sets the presentation text of the conference or person `conf-no` to the text `text-no`. To remove a presentation, use a `text-no` of zero. This call protects the new presentation from being deleted automatically and removes such protection from the old presentation. In lyskomd this is implemented by increasing the mark count on presentation texts.

Example:

```
1 50 6
=1 11HDavid Byers 1001
  26 15 11 9 5 96 0 160 1
  26 15 11 9 5 96 0 160 1
  5 0 5 0 5 0 77 1 1 0
1 16 6 1
=1
1 50 6
=1 11HDavid Byers 1001
  26 15 11 9 5 96 0 160 1
  26 15 11 9 5 96 0 160 1
  5 1 5 0 5 0 77 1 1 0
```

This example shows how the presentation of person 6 is being changed. To start with, the person had no presentation, as is shown by the `get-conf-stat-old` call (see Section 4.51 [get-conf-stat-old], page 78). Later, after `set-presentation` has been called, the presentation field has changed.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: Conference `conf-no` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

`permission-denied`
: Not enough permissions to change presentation of conference `conf-no`.

`no-such-text`
: Text `text-no` does not exist or no read permission.

`mark-limit`
: Adding a mark to text `text-no` would cause it to have too many marks.

4.18 set-etc-motd [17] (1) Recommended

```
set-etc-motd [17] (( conf-no : Conf-No;
                     text-no : Text-No ))
        -> ( );
```

This call sets the message of the day on the conference or person `conf-no` to the article `text-no` and removes the old message. To remove an old message without setting a new one, use a `text-no` of zero. This call protects the new message from automatic deletion and removes such protection from the old message just as `set-presentation` (see Section 4.17 [set-presentation], page 51).

Example:

```
1 50 6
=1 11HDavid Byers 1001
  26 15 11 9 5 96 0 160 1
  26 15 11 9 5 96 0 160 1
  5 0 5 0 5 0 77 1 1 0
1 17 6 1
=1
1 50 6
=1 11HDavid Byers 1001
  26 15 11 9 5 96 0 160 1
  26 15 11 9 5 96 0 160 1
  5 0 5 0 5 1 77 1 1 0
```

This example shows how text number one is used as the message of the day for conference six (which happens to be a mailbox.) The `get-conf-stat-old` calls before and after demonstrate the change in the conference structure.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: Conference `conf-no` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

`permission-denied`
: Not enough permissions to set the MOTD of conference `conf-no`.

`mark-limit`
: Adding a mark to text `text-no` would cause it to have too many marks.

4.19 set-supervisor [18] (1) Recommended

```
set-supervisor [18] (( conf-no : Conf-No;
                       admin   : Conf-No ))
        -> ( );
```

The `set-supervisor` call changes the supervisor of an existing conference. The result is that all members of the conference `admin` become supervisors of the conference `conf-no`. Typically, but not always, `admin` will be a mailbox.

Example:

```
1 50 4
=1 17HNyheter om LysKOM 0000
  48 11 17 13 8 91 5 255 1
  15 12 11  9 5 96 0 160 1
  0 0 0 0 0 0 77 1 1 1
1 18 4 6
=1
1 50 4
=1 17HNyheter om LysKOM 0000
  48 11 17 13 8 91 5 255 1
  15 12 11  9 5 96 0 160 1
  0 0 6 0 0 0 77 1 1 1
```

This example makes the members of conference six supervisors of conference four (which is usually the "News about LysKOM" conference). The change in the conference structure is evident from the `get-conf-stat-old` calls (see Section 4.51 [get-conf-stat-old], page 78) before and after the `set-supervisor` call. Note that the original supervisor was not set. In order to change the supervisor of such a conference, the session issuing the call must have administration privileges.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: Conference `conf-no` or conference `admin` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

`permission-denied`
: Not enough permissions or privileges to change the supervisor of conference `conf-no`.

4.20 set-permitted-submitters [19] (1) Recommended

```
set-permitted-submitters [19] (( conf-no  : Conf-No;
                                 perm-sub : Conf-No ))
        -> ( );
```

This call grants the right to send articles to the conference `conf-no` to all members of the conference `perm-sub`. If `perm-sub` is 0, everybody can send articles to the conference. (This is the default setting of new conferences and persons.)

When a person tries to submit an article but does not have the right to do so, the server will send the article to the super-conference instead.

Example:

```
1 50 4
=1 17HNyheter om LysKOM 0000
  48 11 17 13 8 91 5 255 1
  15 12 11  9 5 96 0 160 1
  0 0 6 0 0 0 77 1 1 1
1 19 4 1
=1
1 50 4
=1 17HNyheter om LysKOM 0000
  48 11 17 13 8 91 5 255 1
  15 12 11  9 5 96 0 160 1
  0 0 6 1 0 0 77 1 1 1
```

This example shows how all members of conference one are given permission to send articles to conference four. From the beginning, only members of conference four were permitted to submit articles. The change is evident from the `get-conf-stat-old` calls (see Section 4.51 [get-conf-stat-old], page 78) before and after the `set-permitted-submitters` call.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: Conference `conf-no` or conference `perm-sub` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

`permission-denied`
: Not enough permissions to change the permitted submitters of conference `conf-no`.

4.21 set-super-conf [20] (1) Recommended

```
set-super-conf [20] (( conf-no    : Conf-No;
                       super-conf : Conf-No ))
        -> ( );
```

Makes the conference `super-conf` the super-conference of the conference `conf-no`. See Section 3.5 [Conference Status Types], page 23, for more info on what this field is used for.

Example:

```
1 50 4
=1 17HNyheter om LysKOM 0000
  48 11 17 13 8 91 5 255 1
  15 12 11  9 5 96 0 160 1
  0 0 6 0 0 0 77 1 1 1
2 20 4 8
=2
3 50 4
=3 17HNyheter om LysKOM 0000
  48 11 17 13 8 91 5 255 1
  15 12 11  9 5 96 0 160 1
  0 0 6 0 8 0 77 1 1 1
```

This example demonstrates how the super-conference of conference 1 is set to conference 8. The calls to `get-conf-stat-old` (see Section 4.51 [get-conf-stat-old], page 78) demonstrate the change in the conference structure.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: Conference `conf-no` or conference `super-conf` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

`permission-denied`
: Not enough permissions to change super-conference of conference `conf-no`.

4.22 set-conf-type [21] (1) Recommended

```
set-conf-type [21] (( conf-no : Conf-No;
                      type : Any-Conf-Type ))
        -> ( );
```

Sets the conference type of conference `conf-no` to `type`. Before protocol version 8, `type` could only be four bits. Starting with protocol version 8, either a four-bit `Conf-Type` or an eight-bit `Extended-Conf-Type` is allowed.

Example:

```
1 78 4
=1 17HNyheter om LysKOM 00001000 1 77
1 21 4 00000000
=1
1 78 4
=1 17HNyheter om LysKOM 00000000 1 77
```

This example shows a user removing the `allow-anonymous` bit from conference four. The `get-uconf-stat` call (see Section 4.79 [get-uconf-stat], page 93) shows all eight bits of the conference type before and after the `set-conf-type` call.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: Conference `conf-no` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

`secret-public`
: `type` has `secret` bit but not `rd-prot`.

`permission-denied`
: Not enough permissions or privileges to change the conference type of conference `conf-no`.

`letterbox`
: Attempt to change the `letterbox` flag.

`invalid-membership-type`
: Attempt to change to a conference type that is not compatible with one of more members of the conference (for example, attempting to set `forbid-secret` on a conference with a secret member.) `error-status` is the id of the first person with an incompatible membership type.

4.23 set-garb-nice [22] (1) Recommended

```
set-garb-nice [22] (( conf-no : Conf-No;
                      nice : Garb-Nice ))
        -> ( );
```

Sets the expiration time for articles in conference `conf-no` to `nice` days. An article that is older than the maximum expiration time of each conference it is sent to may be deleted by the LysKOM server unless it has marks.

Example:

```
1 78 4
=1 17HNyheter om LysKOM 00000000 1 77
1 22 4 7
=1
1 78 4
=1 17HNyheter om LysKOM 00000000 1 7
```

This example shows the expiration time of conference four being lowered from 77 to just seven days.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: Conference `conf-no` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

`permission-denied`
: Not enough permissions to change the expiration time for conference `conf-no`.

4.24 get-marks [23] (1) Recommended

```
get-marks [23] ( )
        -> ( ARRAY Mark );
```

This call returns the list of marks the current user has set.

Example:

```
1 23
=1 3 { 13020 100 13043 95 12213 95 }
```

In this example, the current user has three marks, one on text 13020 with mark type 100, one on text 13042 with mark type 95 and one on text 12213 with mark type 95. The maximum number of marks may be arbitrarily limited in the LysKOM server.

Error codes

`login-first`
: Login required before issuing this call.

4.25 mark-text-old [24] (1) Obsolete

```
mark-text-old [24] (( text      : Text-No;
                      mark-type : INT8 ))
        -> ( );
```

This call has been replaced by `mark-text` (see Section 4.73 [mark-text], page 89) and `unmark-text` (see Section 4.74 [unmark-text], page 90) and should no longer be used. This call can only set `mark-type` to a value in the range 1 to 255 (inclusive). If `mark-type` is set to 0 the mark will be removed.

Error codes

`login-first`
: Login required before issuing this call.

`no-such-text`
: The text `text` does not exist.

`permission-denied`
: No read permission on text `text`.

`undefined-person`
: The person currently logged in does not exist (can't happen error.)

`too-many-marks`
: Marking the text would cause the person doing the marking to have too many marks, or cause the text to have too many marks.

4.26 get-text [25] (1) Recommended

```
get-text [25] (( text       : Text-No;
                 start-char : INT32;
                 end-char   : INT32 ))
        -> ( HOLLERITH );
```

Retrieve text number `text` from the LysKOM database, starting at position `start-char` and ending at position `end-char`. The first character in the text is numbered 0 and the last can be retrieved using `get-text-stat` (see Section 4.91 [get-text-stat], page 101). It is also permitted to request a character position beyond the actual end of the text, in which case as much text as is available will be returned.

Example:

```
1 25 100 0 32766
=1 25HYawn ^JNothing is happening

2 25 100 5 32766
=2 20HNothing is happening
3 25 100 0 3
=3 4HYawn
```

In the example, `^J` represents a newline.

In this example, text 100 is requested three times, first from position 0 to position 32766, then from position 5 to position 32766 and finally from position 0 to position 4. The first reply contains the entire text, the following two contain only the requested portion.

Error codes

`no-such-text`
: The text `text` does not exits or no read permission. This error code will also be used when attempting to fetch texts without logging in first. (There are some texts that are readable without logging in: the motd-of-lyskom (see Section 4.42 [set-motd-of-lyskom], page 72), and texts with the `world-readable` aux item set on them.)

`text-zero`
: Attempt to retrieve text number 0.

`index-out-of-range`
: `start-char` is larger than the length of the text.

4.27 get-text-stat-old [26] (1) Obsolete (10)

```
get-text-stat-old [26] ( text-no : Text-No )
        -> ( Text-Stat-Old );
```

Get information about text number `text-no`. The text-stat contains information about the size of the text, its recipients, comments, author and more.

For compatibility reasons this call will only return the misc-infos 0=recpt, 1=cc-recpt, 2=comm-to, 3=comm-in, 4=footn-to, 5=footn-in, 6=loc-no, 7=rec-time, 8=sent-by and

9=sent-at. Newer misc-infos will either be removed or converted to a similar one. Specifically, 15=bcc-recpt may (at the servers discretion) be converted to 1=cc-recpt or omitted entirely.

Example:

```
1 26 100
=1 7 35 16 15 6 96 1 196 1 14 1 22 1
  7 { 0 7 6 85 0 15 6 1
  8 13 9 12 37 16 15 6 96 1 196 1
  3 311 }
```

In this example, text number 100 was created by person 7 at approximately 4:35PM on July 15 1996. Its recipients are conferences 7 and 15, and it was sent to conference 15 by person 13 at 16:37 on the day it was created. The text has a single comment: text 311.

Error codes

`no-such-text`
: The text `text-no` does not exist, or no read access. This error code will also be used when attempting to fetch texts without logging in first. (There are some texts that are readable without logging in: the motd-of-lyskom (see Section 4.42 [set-motd-of-lyskom], page 72), and texts with the `world-readable` aux item set on them.)

`text-zero`
: Attempt to retrieve text number 0.

4.28 mark-as-read [27] (1) Recommended

```
mark-as-read [27] (( conference : Conf-No;
                     text       : ARRAY Local-Text-No ))
        -> ( );
```

Marks text `text` in conference number `conference` as read for the current user. This call updates the membership record for the user.

Example:

```
1 9 6 7
=1 20 32 11 17 6 96 3 198 1 7 1 240 0 *
1 78 7
=1 13HInlägg }t mig 00001000 241 1
1 27 7 1 { 241 }
=1
1 9 6 7
=1 20 32 11 17 6 96 3 198 1 7 1 241 0 *
```

This example shows person 6 marking local text number 241 in conference 7 as read. In the first query-read-texts call the person has read local text 240, but nothing higher. The mark-as-read call is reflected in the second query-read-texts call, where the user is seen to have read text 241 in conference 7.

To mark a global text number as read it is necessary to translate it into local text numbers by looking at the Misc-Info list in the Text-Stat and calling mark-as-read once for each recipient.

There is no need to call mark-as-read on deleted texts. The server will automatically mark them as read, sooner or later.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: The conference `conference` does not exist or is secret.

`conference-zero`
: `conference` is zero.

`not-member`
: The person logged on is not a member of conference `conference`.

`no-such-local-text`
: One of the numbers in `text` is not a local text number in `conference`. The error argument indicates the index of the invalid number.

`local-text-zero`
: One of the numbers in `text` is zero.

4.29 create-text-old [28] (1) Obsolete (10)

```
create-text-old [28] (( text        : HOLLERITH;
                        misc-info   : ARRAY Misc-Info ))
        -> ( Text-No );
```

Creates a new text with contents from `text` and recipients etc. defined by `misc-info` (see Section 1.4 [The Misc-Info List], page 4). In addition to the result, the server will send an asynchronous message to all members of any of the recipients of the new text. It is not defined whether this messages comes before or after the reply to the create-text message. Clients should be prepared for either situation.

The text up to the first line feed is considered to be the subject line. The remaining text is the message body. Although messages with only a subject are valid, clients should avoid letting users create such messages.

The only Misc-Info items valid for this call are `recpt`, `cc-recpt`, `bcc-recpt` (protocol version 10), `comm-to` and `footn-to`.

Example:

```
1 28 20HExample\\nMessage body 3 { 0 5 1 112 2 33467 }
:16 0 33502 13 16 15 16 6 97 3 196 1 119 1 20 0
    5 { 0 5 6 148 1 112 6 3438 2 33467 }
=1 33502
```

In this example, person 119 creates a text containing a subject and a one-line body. The recipient of the text is conference five, conference 112 is a CC recipient and the text is a comment to text 33467. The server reply indicates that the new text has been given number 33502. Finally there is an asynchronous message sent to all members of recipient conferences. Note how the message was sent before the reply to the client. The misc-info

list in this message has two additional fields, the local numbers of the text in each of its recipient conferences.

Error codes

`login-first`
: Login required before issuing this call.

`string-too-long`
: The string `text` is longer than the maximum length of a message.

`temporary-failure`
: The text could not be created at the moment.

`no-such-text`
: Attempt to comment or footnote a non-existent or secret text.

`not-author`
: Attempt to footnote a text authored by someone else.

`footnote-limit`
: Attempt to footnote a text with the maximum number of footnotes already set.

`comment-limit`
: Attempt to comment a text with the maximum number of comments already set.

`access-denied`
: Attempt to send a text to a conference failed because no access to the conference or any super conference that will accept a text.

`anonymous-rejected`
: Attempt to send an anonymous text to a conference that does not accept anonymous texts.

`illegal-misc`
: Invalid misc-info list. A recipient is listed more than once or there is an unknown misc item in the misc-info list.

4.30 delete-text [29] (1) Recommended

```
delete-text [29] ( text : Text-No )
        -> ( );
```

Deletes the text `text` from the LysKOM database, if the person issuing the command may do so.

Example:

```
1 29 33467
=1
```

This simple example shows the deletion of text number 33467.

Error codes

`login-first`
: Login required before issuing this call.

`no-such-text`
: The text `text` does not exist or no read access.

`not-author`
: The person logged in is not the text author or supervisor of the text author.

4.31 add-recipient [30] (1) Recommended

```
add-recipient [30] (( text-no       : Text-No;
                      conf-no       : Conf-No;
                      recpt-type    : Info-Type ))
        -> ( );
```

Adds `conf-no` as recipient to text `text-no`. If `recpt-type` is 1, then a `cc-recpt` (see Section 1.4 [The Misc-Info List], page 4) is created; otherwise a `recpt` is created.

Since protocol version 8 this call can also be used to change a `cc-recpt` into a `recpt` and vice versa by simply adding a recipient that already exists.

Since protocol version 10 the `recpt-type` parameter is a `Misc-Info`. Only infos `recpt`, `cc-recpt` and `bcc-recpt` are accepted. In protocol version 9 and earlier this argument was a `BOOL`, that indicated if the recipient should be a `cc-recpt` (when true) or `recpt` (when false).

Example:

```
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1 0 *
1 30 1 5 0
=1
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1
   3 { 0 5 6 1 9 34 34 17 17 6 97 4 197 1 }
1 30 1 5 1
=1
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1
   3 { 1 5 6 1 9 34 34 17 17 6 97 4 197 1 }
```

This example show how conference 5 is added first as a recipient of text 1, then changed to a carbon-copy recipient. The misc-info list reflects these changes.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: The conference `conf-no` does not exist.

`conference-zero`
: `conf-no` is zero.

`no-such-text`
: The text `text-no` does not exist.

`already-recipient`
: The conference `conf-no` is already a recipient of the same type as `recpt-type`.

`illegal-info-type`
: `recpt-type` is not `recpt`, `cc-recpt` or `bcc-recpt`.

`recipient-limit`
: The text already has the maximum number of recipients.

`permission-denied`
: Attempt to change recipient types of a recipient, but not the author of the text or supervisor of the recipient.

`access-denied`
: Attempt to add a conference as recipient that the person logged on does not have permission to add texts to. The client may have to chase the super conf chain.

4.32 sub-recipient [31] (1) Recommended

```
sub-recipient [31] (( text-no   :   Text-No;
                      conf-no   :   Conf-No ))
        -> ( );
```

Removes `conf-no` from the list of recipients of text `text-no`. Recipients may be removed by the author of the text or by the supervisor of the recipients of the text or by the supervisor of the author.

Example:

```
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1
   3 { 1 5 6 1 9 34 34 17 17 6 97 4 197 1 }
1 31 1 5
=1
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1 0 *
1 31 1 5
%1 30 0
```

In this example, conference 5 is removed from the recipient list of text number 5. When the call is repeated, the server simply returns an error since conference 5 is not a recipient of the text.

Error codes

`login-first`
: Login required before issuing this call.

`no-such-text`
: The text `text-no` does not exist or is secret.

`not-recipient`
: The conference `conf-no` is not a recipient of text `text-no`.

`undefined-conference`
: The conference `conf-no` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

`permission-denied`
: Not supervisor of text author or conference, and not sender of text to `conf-no` and not enough privileges set and enabled.

4.33 add-comment [32] (1) Recommended

```
add-comment [32] (( text-no     :       Text-No;
                    comment-to  :       Text-No ))
        -> ( );
```

Add a comment link between the text `comment-to` and the text `text-no` (`text-no` becomes a comment to the text `comment-to`). This call is used to add comment links after a text has been created. The normal procedure for creating comments is to add a `comm-to` element to the text's misc-info list when the text is created (see Section 1.4 [The Misc-Info List], page 4).

Example:

```
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1 0 *
1 26 2
=1 49 49 18 17 6 97 4 197 1 5 1 52 1 2 { 0 2 6 1 }
1 32 2 1
=1
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1 1 { 3 2 }
1 26 2
=1 49 49 18 17 6 97 4 197 1 5 1 52 1
   4 { 0 2 6 1 2 1 9 19 52 18 17 6 97 4 197 1 }
```

In this example, text number two is added as a comment to text number one. The change is reflected in the Misc-Info List of the texts.

Error codes

`login-first`
: Login required before issuing this call.

`index-out-of-range`
: The texts `text-no` and `comment-to` are identical. The `error-status` is `text-no`.

`no-such-text`
: The text `text-no` of `comment-to` are undefined.

`comment-limit`
: The text `comment-to` already has the maximum number of comments.

4.34 sub-comment [33] (1) Recommended

```
sub-comment [33] (( text-no      :       Text-No;
                    comment-to   :       Text-No ))
        -> ( );
```

This call removes the text `text-no` from `comment-to`'s list of comments.

Example:

```
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1 1 { 3 2 }
1 26 2
=1 49 49 18 17 6 97 4 197 1 5 1 52 1
   4 { 0 2 6 1 2 1 9 19 52 18 17 6 97 4 197 1 }
1 33 2 1
=1
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1 0 *
1 26 2
=1 49 49 18 17 6 97 4 197 1 5 1 52 1 2 { 0 2 6 1 }
```

In this example text 2 is a comment to text 1, as shown by the misc-info lists of the two texts. The `sub-comment` is called. The misc-info lists are changed to reflect the change.

Error codes

`login-first`
: Login required before issuing this call.

`no-such-text`
: One of the texts `text-no` or `comment-to` does not exist.

`not-comment`
: The text `text-no` is not a comment to `comment-to`.

`permission-denied`
: Not supervisor of author of `text-no` or not sender of the comment and not enough privileges set and enable to complete the call anyway.

4.35 get-map [34] (1) Obsolete (10)

```
get-map [34] (( conf-no         :       Conf-No;
                first-local-no  :       Local-Text-No;
```

```
                    no-of-texts     :       INT32 ))
        -> ( Text-List );
```

This call has been superseded by `local-to-global` (see Section 4.104 [local-to-global], page 109).

This call retrieves an array mapping local text numbers to global numbers. It is most often used to get a list of unread texts in a conference. Clients will usually use the `query-read-texts` (see Section 4.99 [query-read-texts], page 104) or `get-membership` (see Section 4.100 [get-membership], page 105) calls to find the last local number a user has read in a particular conference, then use the `get-map` call to retrieve the global numbers of all unread texts in the conference.

The `conf-no` parameter specifies which conference to get the map of. `first-local-no` is the local number of the first text returned by the call. `no-of-texts` is the maximum number of text the client wants.

The result is a list of global text numbers. The first element of the list is the global number of local number `first-local-no`, specified by the call; the second element is the global number of local number `first-local-no` plus one; and so forth. The list returned by the server is at most `no-of-texts` long, but may be shorter if the call specifies more texts that there are in the conference.

If `first-local-no` is higher than the highest local text number, the server will return an error.

If `first-local-no` is lower than the lowest number that still exists, the server will set `first-local-no` in the returned `Text-List` to the first text that still exists. The size of the returned array will be decreased by the same amount as `first-local-no` is increased. This may result in an empty array being returned. (This paragraph applies even when `first-local-no` is 0.)

If no texts at all exists in `conf-no` the resulting array will be empty, and `first-local-no` will be set to the number the next text to be created will receive.

Example:

```
1 34 119 10 5
=1 10 5 { 0 0 466 478 391 }
2 34 119 16 5
=2 16 3 { 481 0 491 }
3 34 119 19 5
%3 16 0
4 34 120 1 5
=4 4 2 { 480 485 }
5 34 120 1 2
=5 4 0 *
```

This example shows five `get-map` calls. The first retrieves the mappings of local numbers 10 to 15; the second call returns local numbers 16 to 18. As this example shows the maps are not necessarily sorted in ascending order, since texts may be added after their creation, and the maps may contain zeroes anywhere. These represent texts that have been removed for some reason.

Since the first example returned two leading zeroes we can be certain that at least one text with a local text number lower than 10 still exists. Otherwise the result would have been truncated in the front as it is in examples 4 and 5.

The third exchange in the example shows what happens when `first-local-no` is too large.

The forth and fifth examples shows what happens when an attempt to retrieve a mapping from a conference where the first local text numbers have been deleted. In the example local text numbers 1, 2 and 3 no longer exist, and 4 corresponds to 480, and 5 to 485.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: Conference `conf-no` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

`access-denied`
: Conference `conf-no` is read protected.

`no-such-local-text`
: `first-local-no` is higher than the highest local text number that ever has existed in this conference.

4.36 get-time [35] (1) Recommended

```
get-time [35] ( )
        -> ( Time );
```

This call simply returns the local time according to the server.

Example:

```
1 35
=1 23 47 19 17 6 97 4 197 1
```

This example demonstrates the call. According to the server the time is 19:47:23, Thursday July 17, 1997. The result also shows that it is the 197th day of the year, and that daylight savings time is in effect.

Error codes

This call always succeeds

4.37 get-info-old [36] (1) Obsolete (10)

```
get-info-old [36] ( )
        -> ( Info-Old );
```

This call returns the `Info-Old` structure for the server (see [Info-Old], page 27). Clients should call this in order to find out which conferences are used for presentations and such.

This call has been superseded by `get-info` (see Section 4.95 [get-info], page 103).

This call can be issued without logging in.

Example:

```
1 36
=1 10900 1 2 3 4 1
```

In this example, the server version is 1.9, the conference for presentation of new conferences is conference 1, the conference for presentation of new persons is conference 2, the conference for door messages is conference 3, the LysKOM news conference is conference 4 and the login message is text number 1.

Error codes

This call always succeeds.

4.38 add-footnote [37] (1) Recommended

```
add-footnote [37] (( text-no    :       Text-No;
                     footnote-to:       Text-No ))
        -> ( );
```

Add a footnote link between the text `footnote-to` and the text `text-no` (`text-no` becomes a footnote to the text `footnote-to`). This call is used to add footnote links after a text has been created. Only the author of both texts is allowed to add the footnote link.

Example:

```
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1 0 *
1 26 2
=1 49 49 18 17 6 97 4 197 1 5 1 52 1 2 { 0 2 6 1 }
1 37 2 1
=1
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1 1 { 5 2 }
1 26 2
=1 49 49 18 17 6 97 4 197 1 5 1 52 1
   4 { 0 2 6 1 4 1 9 19 52 18 17 6 97 4 197 1 }
```

In this example, text number two is added as a footnote to text number one. The change is reflected in the Misc-Info List of the texts.

Error codes

`login-first`
: Login required before issuing this call.

`no-such-text`
: The text `text-no` or `footnote-to` does not exist or is secret.

`index-out-of-range`
: Maximum number of texts in database already.

`not-author`
: Not author of `footnote-to`.

`footnote-limit`
: Text `footnote-to` already has the maximum number of footnotes.

`already-footnote`
: Text `text-no` is already a footnote to `footnote-to`.

4.39 sub-footnote [38] (1) Recommended

```
sub-footnote [38] (( text-no            :       Text-No;
                     footnote-to        :       Text-No ))
        -> ( );
```

This call removes the text `text-no` from `footnote-to`'s list of footnotes. Only the author of a footnote may remove it.

Example:

```
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1 1 { 5 2 }
1 26 2
=1 49 49 18 17 6 97 4 197 1 5 1 52 1
   4 { 0 2 6 1 4 1 9 19 52 18 17 6 97 4 197 1 }
1 38 2 1
=1
1 26 1
=1 2 22 12 17 6 97 4 197 1 5 4 256 1 0 *
1 26 2
=1 49 49 18 17 6 97 4 197 1 5 1 52 1 2 { 0 2 6 1 }
```

In this example text 2 is a footnote to text 1, as shown by the misc-info lists of the two texts. The `sub-footnote` is called. The misc-info lists are changed to reflect the change.

Error codes

`login-first`
: Login required before issuing this call.

`no-such-text`
: The text `text-no` or `footnote-to` does not exist or is secret.

`not-footnote`
: The text `text-no` is not a footnote to `footnote-to`.

`permission-denied`
: Not supervisor of the author of `text-no` and not enough privileges set and enabled to complete call anyway.

4.40 who-is-on-old [39] (1) Obsolete

```
who-is-on-old [39] ( )
        -> ( ARRAY Who-Info-Old );
```

This call is obsolete. Use `get-static-session-info` (see Section 4.85 [get-static-session-info], page 97) and `who-is-on-dynamic` (see Section 4.84 [who-is-on-dynamic], page 96) instead. If the server does not support these calls, use `who-is-on` (see Section 4.52 [who-is-on], page 78) instead.

The returned list contains all sessions where a person is logged in and the invisible flag of the session is unset.

Error codes

This call always succeeds.

4.41 set-unread [40] (1) Recommended

```
set-unread [40] (( conf-no      :       Conf-No;
                   no-of-unread :       INT32 ))
        -> ( );
```

Only read the last `no-of-unread` in the conference `conf-no`. This call modifies the `last-text-read` of current person's membership for the conference. This call is sometimes used to implement the "only read last N texts" command found in many clients. Due to possible race conditions[2], this call is usually better implemented using the `set-last-read` call (see Section 4.78 [set-last-read], page 92) which explicitly sets the `last-text-read` field of the membership.

Example:

```
1 9 5 6
=1 1 34 21 17 6 97 4 197 1 6 100 0 0 *
1 40 6 0
=1
1 9 5 6
=1 1 34 21 17 6 97 4 197 1 6 100 4 0 *
```

This example shows that person 5 last read text 0 in conference 6 (and since 0 is an illegal local text number, that implies that the person has not read anything in the conference.) After calling set-unread and asking to have zero unread texts in conference 6, this is reflected by the call to query-read-texts.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: The conference `conf-no` does not exist or is secret.

`not-member`
: Not a member of conference `conf-no`.

[2] Another client might create a new text immediately before the server processes this `set-unread` call, so you might end up setting `last-text-read` to something unexpected.

4.42 set-motd-of-lyskom [41] (1) Recommended

```
set-motd-of-lyskom [41] ( text-no : Text-No )
        -> ( );
```

This call sets the login message of LysKOM. It can only be executed by a privileged person, with the proper privileges enabled. A somewhat less convenient way of doing this is to use the `set-info` call (see Section 4.80 [set-info], page 94).

Example:

```
1 36
=1 10900 1 2 3 4 0
1 41 435
=1
1 36
=1 10900 1 2 3 4 435
```

This example shows how the login message of LysKOM is set using the set-motd-of-lyskom call. The results of the `get-info` calls demonstrate the effect.

Error codes

`login-first`
: Login required before issuing this call.

`permission-denied`
: Administrator bit not set or privilege level not enabled.

`no-such-text`
: The text `text-no` does not exist.

`mark-limit`
: The text `text-no` already has the maximum number of marks.

4.43 enable [42] (1) Recommended

```
enable [42] ( level : INT8 )
        -> ( );
```

Sets the security level for the current session to `level`. See Section 1.6 [Security], page 8, for details about security levels. The only levels that make any sense right now are 0 and 255. This call may be issued by any person, but without the right privilege bits set, it has no effect.

Example:

```
1 41 1
%1 12 0
1 42 255
=1
1 41 1
=1
```

This example shows how `enable` makes a privileged call possible, in this case a call to `set-motd-of-lyskom` (see Section 4.42 [set-motd-of-lyskom], page 72).

Error codes

`login-first`
: Login required before issuing this call.

4.44 sync-kom [43] (1) Recommended

```
sync-kom [43] ( )
        -> ( );
```

This call instructs the LysKOM server to make sure the permanent copy of its database is current. Processing of requests is normally blocked until this call has completed, but the exact details depend on the server implementation. This call is privileged in most implementations.

Example:

```
1 42 255
=1
1 43
:0 7
:0 7
=1
```

This example shows how the `enable` call is used to enable all privileges (see Section 4.43 [enable], page 72), then the `sync-kom` call is used to save the database. The server responds with two asynchronous messages signaling that the database is being saved.

Error codes

`login-first`
: Login required before issuing this call.

`permission-denied`
: Administrator bit not set or privileges not enabled.

4.45 shutdown-kom [44] (1) Recommended

```
shutdown-kom [44] ( exit-val : INT8 )
        -> ( );
```

This call instructs the server to save all data and shut down. `exit-val` is currently not used. This call is privileged.

Example:

```
1 42 255
=1
1 44 0
=1
:2 13 5 3
```

This example shows the shutdown of a server. The asynchronous message sent after the call returns is the result of a session being forced to log out.

Error codes

`login-first`
: Login required before issuing this call.

`permission-denied`
: Administrator bit not set or privileges not enabled.

4.46 broadcast [45] (1) Obsolete

```
broadcast [45] ( message : HOLLERITH )
        -> ( );
```

This call can been replaced by `send-message` (see Section 4.54 [send-message], page 79). It is a privileged call.

Error codes

`login-first`
: Login required before issuing this call.

`string-too-long`
: The string `message` is too long.

`feature-disabled`
: Messages have been disabled.

4.47 get-membership-old [46] (1) Obsolete (10)

```
get-membership-old [46] (( person          : Pers-No;
                           first           : INT16;
                           no-of-confs     : INT16;
                           want-read-texts : BOOL ))
                -> ( ARRAY Membership-Old );
```

This call retrieves the membership record for a list of conferences for a single person. `person` is the person whose memberships are to be retrieved. `first` is the first position in the membership list to retrieve, numbered from 0 and up. `no-of-confs` is the number of membership records to retrieve. If `want-read-texts` is '`0`' the server will not send the contents of the `read-texts` array of the memberships. (The size will be transmitted, but a single asterisk ('`*`') will be sent instead of the array itself.)

The server will return a membership list that is shorter than `no-of-confs` if `no-of-confs + first` is larger than the number of conferences the person is a member of.

Servers that support protocol version 10 will return a priority of zero if the passive bit in the membership record has been set (either by a set-membership-type or by setting the priority of the conference to zero.)

Example:

```
1 46 5 0 3 1
=1 2 { 49 14 17 13 8 91 5 255 1 5 255 0 0 *
   20 14 22 17 6 97 4 197 1 6 100 2 0 * }
1 46 5 0 1 1
=1 1 { 49 14 17 13 8 91 5 255 1 5 255 0 0 * }
1 46 5 1 4 1
=1 1 { 20 14 22 17 6 97 4 197 1 6 100 2 0 * }
```

In this example we retrieve the memberships of person 5. The first call asks for three memberships, starting with number 0. Since this person is only a member of two conferences, the list returned only contains two memberships. (An extra newline has been inserted in the result of the first call to make the result more readable.) The next two calls retrieve a single membership each, the first by asking for only one, and the second by asking for four memberships, starting with number 1.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-person`
: The person `person` does not exist.

`undefined-conference`
: The conference `person` does not exist or is secret.

`index-out-of-range`
: `first` is higher than the index of the last conference in the person's membership list.

4.48 get-created-texts [47] (1) Obsolete (10)

```
get-created-texts [47] (( person        :   Pers-No;
                          first         :   Local-Text-No;
                          no-of-texts   :   INT32 ))
                -> ( Text-List );
```

This call is obsolete; instead you should use `map-created-texts` (see Section 4.105 [map-created-texts], page 110).

This call returns a list of the texts written by a person. `person` is the person whose created texts are to be retrieved. `first` is the first text to retrieve. `no-of-texts` is the number of texts to retrieve.

This call is essentially the same as `get-map` (see Section 4.35 [get-map], page 66), but instead of returning the texts sent to a single conference, it returns the texts written by a single person to any conference. The number of texts written by any one person is contained in the Person record for that person.

If `first` is lower than the first text written by `person` that still exists, it will be automatically increased to the first still existing text written by `person`. The `get-created-texts` will still attempt to return information about `no-of-texts` texts. (In this regard `get-map`

and `get-created-texts` differ, since `get-map` will never ever return information about a later text than specified in the arguments to the call.[3])

Example:

```
1 47 5 0 100
=1 1 8 { 1 2 3 4 5 6 7 8 }
2 47 5 4 2
=2 4 2 { 4 5 }
3 47 10 8 8
=3 12 8 { 309 312 324 327 329 339 0 387 }
```

In this example we have retrieved all texts written by person five. The first call asked for 100 texts, but only 8 were returned, which implies that person number 5 only created a total of 8 texts. We can also see that person 5 wrote all the first 8 texts in the conference system. The second call shows how a part of the map can be retrieved.

The third call asks for eight texts written by person 10, starting with the eighth number. Since the first eleven texts written by that person no longer exists the server instead returns information about eight texts staring from the twelfth text person 10 created. One of the eight texts has been deleted.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-person`
: The person `person` does not exist or is secret.

`undefined-conference`
: The conference `person` does not exist or is secret.

`no-such-local-text`
: `first` is higher than the local text number of the last created text.

4.49 get-members-old [48] (1) Obsolete (10)

```
get-members-old [48] (( conf            : Conf-No;
                        first           : INT16;
                        no-of-members   : INT16 ))
        -> ( ARRAY Pers-No );
```

This call returns a list of members of the conference `conf`. `first` is the first index in the membership to return, numbered from zero and up. `no-of-members` is the maximum number of members to return.

[3] This difference was not intentional, but it is now too late to change the semantics of either `get-map` or `get-created-texts`. Besides, they are both obsolete calls.

Example:

```
1 48 1 0 100
=1 4 { 7 8 9 10 }
1 48 6 0 100
=1 4 { 5 7 9 10 }
1 48 6 2 2
=1 2 { 9 10 }
```

In this example the client first requests the first 100 members in conference 1. The second request is for the first 100 members of conference 6. The last request is for members 2 and 3 in conference 6. As can be seen from the examples, the returned list is truncated if there are fewer members than requested.

Error codes

`undefined-conference`
: The conference `conf` does not exist or is secret.

`index-out-of-range`
: `first` is higher than the number of members in `conf`.

4.50 get-person-stat [49] (1) Recommended

```
get-person-stat [49] ( pers-no : Pers-No )
        -> ( Person );
```

This call returns the person `pers-no`. This call does not return all the information a client usually needs since the name is not included in the Person data structure. Use `get-conf-stat` (see Section 4.92 [get-conf-stat], page 102) on the same number to get additional information about the person.

Example:

```
1 49 8
=1 44Hbyers@lage.lysator.liu.se 0000010000000000 00000000
   44 21 19 18 6 97 5 198 1 0 2 3 0 0 0 0 0 0 1 0 0 2
1 50 8
=1 11HPaul Dekker 1001 8 6 19 18 6 97 5 198 1
   8 6 19 18 6 97 5 198 1 8 0 8 0 0 0 77 1 1 0
```

This simple example shows how person number 8 is retrieved from the server. The second call shows the `get-conf-stat-old` call on the same ID number.

Error codes

`undefined-person`
: The person `pers-no` does not exist.

`undefined-conference`
: The conference `pers-no` does not exist or is secret.

4.51 get-conf-stat-old [50] (1) Obsolete (10)

```
get-conf-stat-old [50] ( conf-no : Conf-No )
        -> ( Conference-Old );
```

This call retrieves the conference data structure for conference number `conf-no`.

Important note: This call does not return the extra flag bits that were introduced in protocol version 8. To get this information, use the `get-uconf-stat` call instead. However, clients should be able to handle `Conference-Old` structures with an arbitrary number of flag bits since we may decide to change the behavior of this call in the future.

Example:

```
1 50 1
=1 27HPresentation (av nya) möten 0000 48 11 17 13 8 91 5 255
   1 18 34 21 17 6 97 4 197 1 0 0 0 0 0 0 77 0 1 1
1 50 8
=1 11HPaul Dekker 1001 8 6 19 18 6 97 5 198 1
   8 6 19 18 6 97 5 198 1 8 0 8 0 0 0 77 1 1 0
```

This simple example retrieves conferences 1 and 8 from the server. Conference 1 is a regular conference, and conference 8 is a mailbox.

Error codes

`undefined-conference`
: The conference `conf-no` does not exist or is secret.

4.52 who-is-on [51] (1) Obsolete (9)

```
who-is-on [51] ( )
        -> ( ARRAY Who-Info );
```

This call is obsolete. Please use `who-is-on-dynamic` (see Section 4.84 [who-is-on-dynamic], page 96) and `get-static-session-info` calls instead (see Section 4.85 [get-static-session-info], page 97). Nonetheless, servers should support this call since many clients still use it.

This call should simply return a list of visible sessions (sessions where a person is logged in and the invisible flag is unset). The data structure is described elsewhere (see [Who-Info], page 34).

Error codes

This call always succeeds.

4.53 get-unread-confs [52] (1) Recommended

```
get-unread-confs [52] ( pers-no : Pers-No )
        -> ( ARRAY Conf-No );
```

This call returns a list of conferences in which the person `pers-no` may have unread texts. This call will return a result for any valid `pers-no`. To retrieve information about secret

persons, or to get information about unread texts in secret conference, the session must log on as a person with access to that information.

The result is guaranteed to include all conferences where `pers-no` has unread texts. It may also return some extra conferences. Passive memberships are never returned.

The returned conference numbers will be returned in the same order as they appear on the persons list of memberships.

Example:

```
1 52 7
=1 2 { 1 6 }
1 52 1
%1 10 0
1 52 1000
%1 10 0
```

This example shows how a session first retrieves the list of conferences in which person 7 has unread texts. The next request is for the unread conferences of person 1, but that happens to be a conference. The last request is for the unread conferences of person 1000, but that person didn't exist in the test database.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-person`
: Person `pers-no` does not exist or is secret.

4.54 send-message [53] (1) Recommended

```
send-message [53] (( recipient  :   Conf-No;
                     message    :   HOLLERITH ))
        -> ( );
```

This call sends the message `message` to all members of `recipient` that are currently logged in. If `recipient` is 0, the message is sent to all sessions that are logged in.

Example:

```
1 53 4 14HThis is a test
=1
1 53 1 14HThis is a test
:3 12 1 8 14HThis is a test
=1
1 53 0 14HThis is a test
:3 12 0 8 14HThis is a test
=1
1 53 5 14HThis is a test
%1 16 0
1 53 3 14HThis is a test
%1 42 0
```

Error codes

`login-first`
: Login required before issuing this call.

`string-too-long`
: The string `message` is too long.

`undefined-conference`
: Conference `recipient` does not exist or is secret.

`feature-disabled`
: The message feature has been disabled in the server.

`message-not-sent`
: The message was not sent for some other reason. Perhaps the recipient is not accepting messages or there are no viable recipients for a group message.

4.55 get-session-info [54] (1) Obsolete

```
get-session-info [54] ( session-no : Session-No )
        -> ( Session-Info );
```

This call is obsolete. It has been replaced by `get-session-info-ident` (see Section 4.65 [get-session-info-ident], page 86), which in turn is also obsolete. See `get-session-info-ident` for more information.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-session`
: The session `session-no` does not exist.

4.56 disconnect [55] (1) Recommended

```
disconnect [55] ( session-no : Session-No )
        -> ( );
```

This call disconnects the session `session-no` from the LysKOM server. A session can always disconnect itself, even without logging in. If the session is logged in as user *foo* it can also disconnect any session logged in as a person for which *foo* is the supervisor.

Session number zero is always interpreted as the session making the call, so the easiest way to disconnect the current session is to disconnect session zero.

Example:

```
1 56
=1 7
1 55 7
=1
:2 13 8 7
Connection closed by foreign host.
```

In this example the client asks for its own session number, then disconnects itself (disconnection session 0 would have had the same effect.) The asynchronous message sent just before the session is disconnected is the logout message for the user that was logged on in the session. The "Connection closed by foreign host." is not part of the server output. This message was generated by telnet.

Error codes

`login-first`
: Login required before issuing this call if `session-no` is not the session issuing the call.

`permission-denied`
: Attempt to disconnect another session and not supervisor of person logged in and not enough privileges set and enabled to complete the call anyway.

`undefined-session`
: The session `session-no` does not exist.

4.57 who-am-i [56] (1) Recommended

```
who-am-i [56] ( )
        -> ( Session-No );
```

This call simply returns the session number of the session issuing the call.

Example:

```
1 56
=1 7
```

In this example the session number of the session issuing the call is seven.

Error codes

This call always succeeds.

4.58 set-user-area [57] (2) Recommended

```
set-user-area [57] (( pers-no   :   Pers-No;
                      user-area :   Text-No ))
        -> ( );
```

This call sets the user-area field for the person `pers-no` in the database to the text `user-area`. The user area is used to store client data for a particular person. See Chapter 10 [The User Area], page 135, for more details.

Example:

```
1 49 7
=1 25Hdavby@lage.lysator.liu.se 0000010000000000 00000000
   6 58 21 19 6 97 6 199 1 0 458 7 3 12 7 12 0 0 3 0 0 4
1 57 7 11
=1
1 49 7
=1 25Hdavby@lage.lysator.liu.se 0000010000000000 00000000
   6 58 21 19 6 97 6 199 1 11 458 7 71 2592 7 13 0 0 3 1 0 4
```

In this example the user area of person 7 is set to text number 11. The original user area was text numbers zero, which means that the person had no user area.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-person`
: The person `pers-no` does not exist or is secret.

`permission-denied`
: Not enough access to person `pers-no` to complete the call.

4.59 get-last-text [58] (3) Recommended

```
get-last-text [58] ( before : Time )
        -> ( Text-No );
```

This call returns the number of the last text created before `before`. There is no guarantee that the text is readable by the person making the request, or that the text even exists.

This call assumes that all texts are written in chronological order, when the time is expressed in the local time zone of the server. That may not always be the case in real life. When daylight savings time reverts to standard time the same time span will occur twice. The clock of the server may also have been adjusted manually from time to time. This protocol specification does not mandate what the server should do in such cases.

Example:

```
1 58 49 6 22 19 6 97 6 199 1
=1 11
1 58 49 6 22 18 6 97 6 199 1
=1 8
1 58 49 6 22 1 6 97 6 199 1
=1 0
```

In this example the text created most recently before 22:06 on July 19, 1997 was text number 11; the text created most recently before 22:06 on July 18 was text number 8; and the text created most recently before 22:06 on July 1st was text number 0, which means that there is no text that old in the database.

Error codes

This call never fails.

4.60 create-anonymous-text-old [59] (3) Obsolete (10)

```
create-anonymous-text-old [59] (( text      : HOLLERITH;
                                  misc-info : ARRAY Misc-Info ))
        -> ( Text-No );
```

Similar to `create-text-old` (see Section 4.29 [create-text-old], page 61), but the text is created with the `author` field set to zero. Not even the server has a record of who created the text.

The original intended use for this call was for importing texts from other sources, such as WWW, FTP or Gopher, but some clients include explicit support for sending anonymous texts to a server.

It is only possible to send anonymous texts to a conference with the right flag bit set.

The only Misc-Info items valid for this call in the `misc-info` array are `recpt`, `cc-recpt`, `bcc-recpt` (introduced in protocol version 10), `comm-to` and `footn-to`.

Example:

```
1 28 20HExample\\nMessage body 3 { 0 5 1 112 2 33467 }
:16 0 33502 13 16 15 16 6 97 3 196 1 0 1 20 0
    5 { 0 5 6 148 1 112 6 3438 2 33467 }
=1 33502
```

In this example, person 119 creates a text containing a subject and a one-line body. The recipient of the text is conference five, conference 112 is a CC recipient and the text is a comment to text 33467. The server reply indicates that the new text has been given number 33502. Finally there is an asynchronous message sent to all members of recipient conferences. Note how the message was sent before the reply to the client. The misc-info list in this message has two additional fields, the local numbers of the text in each of its recipient conferences.

Error codes

`login-first`
: Login required before issuing this call.

`string-too-long`
: The string `text` is longer than the maximum length of a message.

`temporary-failure`
: The text could not be created at the moment.

`no-such-text`
: Attempt to comment or footnote a non-existent or secret text.

`not-author`
: Attempt to footnote a text authored by someone else.

`footnote-limit`
: Attempt to footnote a text with the maximum number of footnotes already set.

`comment-limit`
: Attempt to comment a text with the maximum number of comments already set.

`access-denied`
: Attempt to send a text to a conference failed because no access to the conference or any super conference that will accept a text.

`anonymous-rejected`
: Attempt to send an anonymous text to a conference that does not accept anonymous texts.

`illegal-misc`
: Invalid misc-info list. A recipient is listed more than once or there is an unknown misc item in the misc-info list.

4.61 find-next-text-no [60] (3) Recommended

```
find-next-text-no [60] ( start : Text-No )
        -> ( Text-No );
```

This call returns the next readable text in the database created after text `start`. `start` does not have to be a valid or readable text number, as shown in the examples.

Example:

```
1 60 0
=1 2
1 60 2
=1 4
```

This example shows how to retrieve the first readable text in the LysKOM database by calling `find-next-text-no` with `start` set to zero. In the example, the first text is number 2. The second example gets the text following number 2, which happens to be text number 4.

Error codes

`no-such-text`
: There is no text following text `start`.

4.62 find-previous-text-no [61] (3) Recommended

```
find-previous-text-no [61] ( start : Text-No )
        -> ( Text-No );
```

This call returns the first readable text in the database created most recently before `start`. `start` does not have to be a valid or readable text number, as shown in the examples.

Example:

```
1 61 134217727
=1 11
1 61 4
=1 2
```

This example shows that the last readable text in the database is number 11 (unless by some odd coincidence all text from 11 to text number 134217727 have been deleted.) It also shows that the most recent text before number 4 is text number 2.

Error codes

`no-such-text`
: There is no text preceding text `start`.

4.63 login [62] (4) Recommended

```
login [62] (( person        :   Pers-No;
              passwd        :   HOLLERITH;
              invisible     :   BOOL ))
        -> ( );
```

This call is used to log in. The session is logged in as person number `person` if `passwd` is the correct password for that person. If `invisible` is true, the session is invisible: it will not be returned by `who-is-on` (see Section 4.52 [who-is-on], page 78) and `who-is-on-ident` (see Section 4.64 [who-is-on-ident], page 86), and the dynamic session info (see [Dynamic-Session-Info], page 37) will have the invisible flag set.

Invisible sessions are primarily used by software agents that do not act on the behalf of real users.

Example:

```
1 62 7 6Hgazonk 1
=1
1 62 7 6Hgazonk 0
:2 9 7 1
1 62 7 6Hgazonk 0
:2 13 7 1
:2 9 7 1
=1
```

This example first shows a session log in as person seven with the invisible flag set. Because of this the asynchronous login message is not sent. The second call logs in as person seven again. This time a login message is sent, but not a logout message since the login was invisible. The third example shows a third login as person 7, but this time both the logout and login messages are sent.

Error codes

`undefined-person`
: The person `person` does not exist.

`login-disallowed`
: Logins have been disabled and person `person` does not have enough privileges to override.

`invalid-password`
: The password `passwd` is not the password of `person` and the currently logged in person is not the supervisor of `person` and does not have enough privileges set and enabled to log in anyway.

`conference-zero`
: Attempt to log in as person number 0.

4.64 who-is-on-ident [63] (4) Obsolete (9)

```
who-is-on-ident [63] ( )
        -> ( ARRAY Who-Info-Ident );
```

This call is obsolete. It has been replaced by `who-is-on-dynamic` (see Section 4.84 [who-is-on-dynamic], page 96) and `get-static-session-info` (see Section 4.85 [get-static-session-info], page 97). It returns a list of all visible sessions.

Error codes

This call always succeeds.

4.65 get-session-info-ident [64] (4) Obsolete (9)

```
get-session-info-ident [64] ( session-no : Session-No )
        -> ( Session-Info-Ident );
```

This call is obsolete. Use `who-is-on-dynamic` (see Section 4.84 [who-is-on-dynamic], page 96) combined with `get-static-session-info` (see Section 4.85 [get-static-session-info], page 97) instead.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-session`
: The session `session-no` does not exist.

4.66 re-lookup-person [65] (5) Obsolete

```
re-lookup-person [65] ( regexp : HOLLERITH )
        -> ( ARRAY Pers-No );
```

This call is obsolete. It has been replaced by `re-z-lookup` (see Section 4.75 [re-z-lookup], page 91). It returns a list of persons matching the regular expression `regexp`. The regexp syntax used is that of the `ed`(1) Unix utility.

Error codes

`regexp-error`
: Error compiling the regexp `regexp`. Perhaps the pattern is not a correct regexp.

4.67 re-lookup-conf [66] (5) Obsolete

```
re-lookup-conf [66] ( regexp : HOLLERITH )
        -> ( ARRAY Conf-No );
```

This call is obsolete. It has been replaced by `re-z-lookup` (see Section 4.75 [re-z-lookup], page 91). It returns a list of conferences matching the regular expression `regexp`. The regexp syntax used is that of the `ed`(1) Unix utility.

Error codes

`regexp-error`
: Error compiling the regexp `regexp`. Perhaps the pattern is not a correct regexp.

4.68 lookup-person [67] (6) Obsolete

```
lookup-person [67] ( name : HOLLERITH )
        -> ( ARRAY Pers-No );
```

This call is obsolete. It has been replaced by `lookup-z-name` (see Section 4.77 [lookup-z-name], page 92).

This call returns a list of persons with names matching the contracted name in `name`. See Chapter 8 [Name Expansion], page 131, for a description of the matching process.

Error codes

This call always succeeds.

4.69 lookup-conf [68] (6) Obsolete

```
lookup-conf [68] ( name : HOLLERITH )
        -> ( ARRAY Conf-No );
```

This call is obsolete. It has been replaced by `lookup-z-name` (see Section 4.77 [lookup-z-name], page 92).

This call returns a list of conferences with names matching the contracted name in `name`. See Chapter 8 [Name Expansion], page 131, for a description of the matching process.

Error codes

This call always succeeds.

4.70 set-client-version [69] (6) Recommended

```
set-client-version [69] (( client-name      :   HOLLERITH;
                           client-version   :   HOLLERITH ))
        -> ( );
```

This call is used to tell the server which client and which version of that client is being used. The name of the client is passed in `client-name` and the version in `client-version`. The information sent in this call is made available to other sessions through the `get-client-name` (see Section 4.71 [get-client-name], page 88) and `get-client-version` calls (see Section 4.72 [get-client-version], page 89). This call should be used exactly once per session.

The following names are currently registered:

elisp-client	The famous CPU hog.
nilkom	C++ experiment
tkom	C++ experiment
getmail	Postmaster
ttykom	The one and second tty client by Linus
WinKOM	Windows client
JySKom	JSK Web Client

Example:

```
1 56
=1 7
2 69 11Helisp-client 4H0.45
=2
3 70 7
=3 11Helisp-client
4 71 7
=4 4H0.45
```

In this example the `who-am-i` call is used to find the ID of the current session (see Section 4.57 [who-am-i], page 81). Next, `set-client-version` is used to set the name of the client to "elisp-client" and the version to "0.45". The third call is to `get-client-name`, which returns the string just sent to the server (see Section 4.71 [get-client-name], page 88). Finally `get-client-version` (see Section 4.72 [get-client-version], page 89) is used to retrieve the client version of session number 7, which is, as expected, the string "0.45".

Error codes

`string-too-long`
: The string `client-name` or `client-version` is too long.

`client-is-crazy`
: The client attempted to use this call more than once. The `error-status` is undefined.

4.71 get-client-name [70] (6) Recommended

```
get-client-name [70] ( session : Session-No )
        -> ( HOLLERITH );
```

This call returns the name of the client that owns session number `session`. This client name string returned is the one set by the client using `set-client-version` (see Section 4.70 [set-client-version], page 88). If `set-client-version` has not been issued in session number `session`, the empty string is returned.

See Section 4.70 [set-client-version], page 88, for an example of this call.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-session`
: The session `session` does not exist.

4.72 get-client-version [71] (6) Recommended

```
get-client-version [71] ( session : Session-No )
        -> ( HOLLERITH );
```

This call returns the version of the client that owns session number `session`. This client version string returned is the one set by the client using `set-client-version` (see Section 4.70 [set-client-version], page 88). If `set-client-version` has not been issued in session number `session`, the empty string is returned.

See Section 4.70 [set-client-version], page 88, for an example of this call.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-session`
: The session `session` does not exist.

4.73 mark-text [72] (4) Recommended

```
mark-text [72] (( text      :   Text-No;
                  mark-type :   INT8 ))
        -> ( );
```

This call associates the mark `mark-type` with the text `text`. The list of marks set by a person can be retrieved using the `get-marks` call (see Section 4.24 [get-marks], page 58).

Currently, servers do not associate any particular meaning to the different types of marks, but that may change in the future. Currently, servers should not delete texts that have marks, except by user request.

Example:

```
1 23
=1 0 *
2 72 110 230
=2
3 23
=3 1 { 110 230 }
```

This example shows how a person with no marks set sets mark 230 on text number 110. The calls to `get-marks` (see Section 4.24 [get-marks], page 58) show the effect of the call.

Error codes

`login-first`
: Login required before issuing this call.

`no-such-text`
: The text `text` does not exists or is secret.

`permission-denied`
: No read access to text `text`.

`undefined-person`
: The person currently logged in does not exist (can't happen error.)

`mark-limit`
: Already the maximum number of marks on text `text`.

4.74 unmark-text [73] (6) Recommended

```
unmark-text [73] ( text-no : Text-No )
        -> ( );
```

This call removes any marks the logged-in person has set on the text `text-no`.

Example:

```
1 23
=1 1 { 110 230 }
2 73 110
=2
3 23
=3 0 *
```

This example shows how a user with a mark set on text number 110 removes it using the `unmark-text` call.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-person`
: The person currently logged in does not exist (can't happen error.)

`not-marked`
: The text `text-no` was not marked.

4.75 re-z-lookup [74] (7) Recommended

```
re-z-lookup [74] (( regexp          :   HOLLERITH;
                    want-persons    :   BOOL;
                    want-confs      :   BOOL ))
            -> ( ARRAY Conf-Z-Info );
```

This call returns a list of those conferences and/or persons matching the regular expression `regexp`. If `want-confs` is true, then the result will include non-mailbox conferences. If `want-persons` is true, then the result will include mailbox conferences.

See also Section 4.77 [lookup-z-name], page 92, for an alternative way to look up names.

Refer to Chapter 8 [Name Expansion], page 131, for more details on how name lookup works.

Example:

```
1 74 2H.* 1 1
=1 4 { 15HTest Conference 0000 10 11HDavid Byers 1001 6
       21HTrains (-) Discussion 0000 11 4HJohn 1001 9 }
2 74 2H.* 0 1
=2 2 { 15HTest Conference 0000 10
       21HTrains (-) Discussion 0000 11 }
3 74 7H T.*[cC]
1 1
=3 2 { 15HTest Conference 0000 10
       21HTrains (-) Discussion 0000 11 }
```

This example shows three calls to `re-z-lookup`. The first call returns all conferences and persons in the entire database, in this case two conferences and two persons. The second example uses the same regular expression, but in this case, the call specifies that the result is only to contain conferences, so the two persons are not returned. The third example simply returns all names matching the pattern "T.*[cC]".

Error codes

`regexp-error`
: Error compiling the regexp `regexp`. Perhaps the pattern is not a correct regexp.

4.76 get-version-info [75] (7) Recommended

```
get-version-info [75] ( )
        -> ( Version-Info );
```

This call returns information about the server version. The data returned by this call are primarily useful for presenting to the user. A client should not use this call to determine what the server's capabilities are.

Example:

```
1 75
=1 9 7Hlyskomd 5H1.9.0
```

This example lets us know that the server is lyskomd, version 1.9.0, which at the time of writing this is the only really usable server.

Error codes

This call always succeeds.

4.77 lookup-z-name [76] (7) Recommended

```
lookup-z-name [76] (( name          :   HOLLERITH;
                      want-pers     :   BOOL;
                      want-confs    :   BOOL ))
        -> ( ARRAY Conf-Z-Info );
```

This call looks up the name `name` in the server, and returns a list of all matching conferences and/or persons. If `want-confs` is true, then the result will include conferences that are not mailboxes. If `want-pers` is true, then the result will include conferences that are mailboxes.

See also Section 4.75 [re-z-lookup], page 91, for an alternative way to look up names.

Refer to Chapter 8 [Name Expansion], page 131, for details on the matching process.

Example:

```
1 76 0H 1 1
=1 4 { 15HTest Conference 0000 10 11HDavid Byers 1001 6
       21HTrains (-) Discussion 0000 11 4HJohn 1001 9 }
2 76 0H 0 1
=1 2 { 15HTest Conference 0000 10
      21HTrains (-) Discussion 0000 11 }
3 76 3HT C 1 1
=3 1 { 15HTest Conference 0000 10 }
```

This example shows three calls to `lookup-z-name`. The first call retrieves all conferences and persons in the server. The second request looks up the same name as the first, but this time the result is restricted to conferences. The final request requests all conferences and persons matching the pattern “T C”.

Error codes

This call always succeeds.

4.78 set-last-read [77] (8) Recommended

```
set-last-read [77] (( conference :   Conf-No;
                      last-read  :   Local-Text-No ))
        -> ( );
```

This call tells the server that the last local text number the person issuing the call has read in conference `conference` is `last-read`. This call is typically used when a user wants to have a specific number of unread texts in a particular conference.

Example:

```
1 9 7 6
=1 2 4 22 18 6 97 5 198 1 6 100 6 0 *
2 77 6 3
=2
3 9 7 6
=3 2 4 22 18 6 97 5 198 1 6 100 3 0 *
```

This example shows how person 7 originally had read everything up to and including local text number 6 in conference 6. After the call to `set-last-read`, the `query-read-texts` (see Section 4.99 [query-read-texts], page 104) call reports that person 7 has read everything up to and including local text number 3.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: The conference `conference` does not exist or is secret.

`not-member`
: Not a member of conference `conference`.

4.79 get-uconf-stat [78] (8) Recommended

```
get-uconf-stat [78] ( conference : Conf-No )
        -> ( UConference );
```

This call returns some information about conference `conference`. The information it returns is sufficient for most uses of conference information, and this call should be used instead of `get-conf-stat` wherever possible (see Section 4.92 [get-conf-stat], page 102). It uses less bandwidth and the lyskomd server always keeps all `UConference` objects in memory, so this call is significantly faster than `get-conf-stat`.

This is also currently the only way to get all the flag bits of the conference.

Example:

```
1 50 6
=1 8HTestconf 0000 1 34 21 17 6 97 4 197 1
   37 3 22 18 6 97 5 198 1 5 4 5 0 5 0 77 4 1 6
2 78 6
=2 8HTestconf 00001000 6 77
3 50 7
=3 11HDavid Byers 1111 13 4 19 18 6 97 5 198 1
   13 4 19 18 6 97 5 198 1 7 0 7 0 0 0 77 1 1 0
4 78 7
=4 11HDavid Byers 11111000 0 77
```

This example shows the difference between `get-conf-stat-old` (see Section 4.51 [get-conf-stat-old], page 78) and `get-uconf-stat`. In the first two examples conference 6 is retrieved, and in the second two, conference 7, which happens to be a person, is retrieved. Note the difference in length of the flag field.

Error codes

`undefined-conference`
: The conference `conference` does not exist or is secret.

`conference-zero`
: `conference` is zero.

4.80 set-info [79] (9) Recommended

```
set-info [79] ( info : Info-Old )
        -> ( );
```

This call sets the server information retrieved by `get-info-old` (see Section 4.37 [get-info-old], page 68). The version number in the info structure is ignored (but must be present); all other fields are stored permanently in the LysKOM database. This is a privileged call.

Example:

```
1 79 10901 1 2 3 4 1080
=1
```

This example sets the conference presentation conference to one, the user presentation conference to two, the motd conference to three and the news conference to four. It also sets the login message to text 1080. It also attempts to set the version number to 1.9.1, but that number is silently ignored by the server.

Error codes

`login-first`
: Login required before issuing this call.

`permission-denied`
: Administrator bit not set or privileges not enabled.

`undefined-conference`
: One of the conferences in `info` does not exist.

`no-such-text`
: The MOTD text in `info` does not exist.

`mark-limit`
: The MOTD text in `info` already has the maximum number of marks.

`text-zero`
: This error message should never be returned, but lyskomd 1.9.0 erroneously returned this error message if the MOTD text in `info` was set to 0. That should mean that there should be no message of the day, and the current implementation of the server does accept MOTD to be 0.

4.81 accept-async [80] (9) Recommended

```
accept-async [80] ( request-list : ARRAY INT32 )
        -> ( );
```

This call advises the server that the client wants to receive the asynchronous messages listed in `request-list`. The server must send these messages to the client when applicable, but may also send other types of messages if it so desires. The list of currently requested asynchronous messages may be retrieved using the `query-async` call (see Section 4.82 [query-async], page 95).

Don't forget that message type 12 is personal, group and global text messages. Most users will not want these turned off.

Example:

```
1 80 2 { 7 9 }
=1
```

This example tells the server that the client wants to receive asynchronous messages when the database is being synched (message 7) and when someone logs in (message 9).

Error codes

If the client requests a message that the server does not send, the server will reply with an error message saying which message number it does not support. The call will succeed anyway. This mechanism is useful for clients that want new versions of some messages, but need to be compatible with older servers.

`unknown-async`
: At least one of the numbers in `request-list` is not the number of an async message the server knows about. The `error-status` indicates the first offending number.

  Please note that a bug in lyskomd 1.9.0 prevented the server from sending this error message (frankly, we simply forgot about it.)

`long-array`
: The `request-list` was too long. Servers should always accept a `request-list` that contains a lot more asynchronous messages than the server sends, so that it can deal with newer clients. This error message should only be returned if the client tries to trigger a buffer overrun.

4.82 query-async [81] (9) Recommended

```
query-async [81] ( )
        -> ( ARRAY INT32 );
```

This call queries the server for which asynchronous messages the client is receiving. Note that the client may not be able to turn off all messages returned in this list since the server may consider some messages to be mandatory. Also note that the client may still receive messages that are not listed in the result of this call. Even though those messages are turned off, the server may decide to send them under certain circumstances.

Example:

```
1 81
=1 7 { 0 5 7 9 11 12 13 }
```

In this example the client is receiving seven types of asynchronous messages: messages about new articles, changed names, database synching, new logins, rejected connections, personal messages and logouts. This particular set was the default for new connections to lyskomd 1.9 servers. See Chapter 5 [Asynchronous Messages], page 113, for the currently recommended list of asynchronous messages that servers should preselect.

Error codes

This call always succeeds.

4.83 user-active [82] (9) Recommended

```
user-active [82] ( )
        -> ( );
```

This call simply notifies the server that the user is active. The server uses the time of the last user-active call to calculate how long a user has been idle.

The client should send this to the server every time the user actively does something LysKOM-related, such as reads a texts, writes on a comment, gives a command, de-iconifies the LysKOM window, et c. However, the call should not be issued more than twice per minute, to avoid excessive network and server load.

Error codes

This call always succeeds.

4.84 who-is-on-dynamic [83] (9) Recommended

```
who-is-on-dynamic [83] (( want-visible      :   BOOL;
                          want-invisible    :   BOOL;
                          active-last       :   INT32 ))
        -> ( ARRAY Dynamic-Session-Info );
```

This call returns a list of information about sessions. Only sessions with the desired visibility and activeness are returned.

If `want-visible` is true then information about visible sessions is returned. If `want-invisible` is true then information about invisible sessions is returned. If they are both true sessions will be included in the answer regardless of their visibility status.

Sessions where no-one is logged in are considered invisible, and the `invisible` flag is set in the corresponding `Dynamic-Session-Info` that is returned.

If `active-last` is zero then the result is a list of all sessions with the proper visibility. If `active-last` is nonzero then only sessions that have issued an `user-active` call within the last `active-last` seconds are included in the list. Sessions that have never issued an `user-active` call are always included (if they have the proper visibility).

Error codes

This call always succeeds.

4.85 get-static-session-info [84] (9) Recommended

```
get-static-session-info [84] ( session-no : Session-No )
        -> ( Static-Session-Info );
```

This call returns information about session number `session-no`. The returned information cannot change until the session number is reused (and that cannot happen until the server is shut down), so clients are encouraged to cache the information.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-session`
: The session `session-no` does not exist.

4.86 get-collate-table [85] (10) Recommended

```
get-collate-table [85] ( )
        -> ( HOLLERITH );
```

This call returns the collate table being used by the server to match names. If index A and index B in the string are the same character, characters A and B are considered equivalent. An empty collate table indicates that the server considers all characters different.

Currently, the lyskomd server only deals with 8-bit characters. Clients should be prepared for collate tables of any length. Characters whose code are greater than the length of the collate table should be considered to be unique.

Error codes

This call always succeeds.

4.87 create-text [86] (10) Recommended

```
create-text [86] (( text        :   HOLLERITH;
                    misc-info   :   ARRAY Misc-Info;
                    aux-items   :   ARRAY Aux-Item-Input ))
        -> ( Text-No );
```

Creates a new text with contents from `text` and recipients etc. defined by `misc-info` (see Section 1.4 [The Misc-Info List], page 4). In addition to the result, the server will send an asynchronous message to all members of any of the recipients of the new text. It is not defined whether this messages comes before or after the reply to the create-text message. Clients should be prepared for either situation.

The text up to the first line feed is considered to be the subject line. The remaining text is the message body. Although messages with only a subject are valid, clients should avoid letting users create such messages.

The items in `aux-items` are attached to the new text.

The only Misc-Info items valid for this call are `recpt`, `cc-recpt`, `bcc-recpt` (introduced in protocol version 10), `comm-to` and `footn-to`.

Error codes

`login-first`
: Login required before issuing this call.

`string-too-long`
: The string `text` is longer than the maximum length of a message.

`temporary-failure`
: The text could not be created at the moment.

`no-such-text`
: Attempt to comment or footnote a non-existent or secret text.

`not-author`
: Attempt to footnote a text authored by someone else.

`footnote-limit`
: Attempt to footnote a text with the maximum number of footnotes already set.

`comment-limit`
: Attempt to comment a text with the maximum number of comments already set.

`index-out-of-range`
: Attempt to create a text failed because we reached the maximum number of texts permitted.

`access-denied`
: Attempt to send a text to a conference failed because no access to the conference or any super conference that will accept a text.

`illegal-misc`
: Invalid misc-info list. A recipient is listed more than once or there is an unknown misc item in the misc-info list.

`illegal-aux-item`
: One of the aux-items in `aux-items` is illegal. The tag might be out of range, the item not applicable to texts or whatever

`aux-item-permission`
: One of the items looks valid but could not be created anyway.

`long-array`
: Too many Misc-Info items or aux-items were specified.

4.88 create-anonymous-text [87] (10) Recommended

```
create-anonymous-text [87] (( text      : HOLLERITH;
                              misc-info : ARRAY Misc-Info;
                              aux-items : ARRAY Aux-Item-Input ))
        -> ( Text-No );
```

Similar to `create-text` (see Section 4.87 [create-text], page 97), but the text is created the author field set to zero. Not even the server has a record of who created the text.

The original intended use for this call was for importing texts from other sources, such as WWW, FTP or Gopher, but some clients include explicit support for sending anonymous texts to a server.

It is only possible to send anonymous texts to a conference with the right flag bit set.

The only Misc-Info items valid for this call in the `misc-info` array are `recpt`, `cc-recpt`, `bcc-recpt` (introduced in protocol version 10), `comm-to` and `footn-to`.

Error codes

`login-first`
: Login required before issuing this call.

`string-too-long`
: The string `text` is longer than the maximum length of a message.

`temporary-failure`
: The text could not be created at the moment.

`no-such-text`
: Attempt to comment or footnote a non-existent or secret text.

`not-author`
: Attempt to footnote a text authored by someone else.

`footnote-limit`
: Attempt to footnote a text with the maximum number of footnotes already set.

`comment-limit`
: Attempt to comment a text with the maximum number of comments already set.

`access-denied`
: Attempt to send a text to a conference failed because no access to the conference or any super conference that will accept a text.

`anonymous-rejected`
: Attempt to send an anonymous text to a conference that does not accept anonymous texts.

`illegal-misc`
: Invalid misc-info list. A recipient is listed more than once or there is an unknown misc item in the misc-info list.

`illegal-aux-item`
: One of the aux-items in `aux-items` is illegal. The tag might be out of range, the item not applicable to texts or whatever

`aux-item-permission`
: One of the items looks valid but could not be created anyway.

4.89 create-conf [88] (10) Recommended

```
create-conf [88] (( name        :   HOLLERITH;
                    type        :   Any-Conf-Type;
                    aux-items   :   ARRAY Aux-Item-Input ))
            -> ( Conf-No );
```

This call is used to create new conferences. `name` is the name of the new conference and `type` is its type. If successful, the call returns the conference number of the newly created conference. The list `aux-items` contains the aux items to attach to the conference.

To use this call the session must have logged in as a user with privileges to create conferences (see Section 1.6 [Security], page 8).

Error codes

`login-first`
: Login required before issuing this call.

`permission-denied`
: The server does not allow everyone to create a conference and user does not have the `create-conf` bit set. May also be an attempt to create a conference with the `letterbox` bit set.

`conference-exists`
: A conference named `name` already exists.

`bad-name` `name` contains invalid characters.

`string-to-long`
: `name` is too long to be used as a conference name.

`secret-public`
: The conference type has the `secret` bit set, but the `rd-prot` bit is cleared.

`illegal-aux-item`
: One of the aux-items in `aux-items` is illegal. The tag might be out of range, the item not applicable to conferences or whatever

`aux-item-permission`
: One of the items looks valid but could not be created anyway.

4.90 create-person [89] (10) Recommended

```
create-person [89] (( name      :   HOLLERITH;
                      passwd    :   HOLLERITH;
                      flags     :   Personal-Flags;
                      aux-items :   ARRAY Aux-Item-Input ))
              -> ( Pers-No );
```

This call requests that the server create a new person with the name and password given as arguments. To create a person the session must be logged in as a person with sufficient

privileges. The list `aux-items` contains the aux items that are to be attached to the new person's mailbox conference. The person flags are set to `flags`.

The new person will be a member of exactly one conference: the associated mailbox. That membership will have priority 255 and (of course) position 0. All flags of the membership will be 0.

Unlike call number 5, this call does not do an automatic login.

Error codes

`login-first`
: The session is not logged in and the server does not allow person creation before logging in.

`permission-denied`
: The server does not allow everyone to create person and the person currently logged on does not have the `create-pers` bit set.

`person-exists`
: There is already a person named `name`.

`invalid-password`
: The string `passwd` is not a valid password.

`illegal-aux-item`
: One of the aux-items in `aux-items` is illegal. The tag might be out of range, the item not applicable to conferences or mailboxes or whatever.

`aux-item-permission`
: One of the items looks valid but could not be created anyway.

4.91 get-text-stat [90] (10) Recommended

```
get-text-stat [90] ( text-no : Text-No )
        -> ( Text-Stat );
```

Get information about text number `text-no`. The text-stat contains information about the size of the text, its recipients, comments, author and more.

Error codes

`no-such-text`
: The text `text-no` does not exist, or no read access. This error code will also be used when attempting to fetch texts without logging in first. (There are some texts that are readable without logging in: the motd-of-lyskom (see Section 4.42 [set-motd-of-lyskom], page 72), and texts with the `world-readable` aux item set on them.)

`text-zero`
: Attempt to retrieve text number 0.

4.92 get-conf-stat [91] (10) Recommended

```
get-conf-stat [91] ( conf-no : Conf-No )
        -> ( Conference );
```

This call retrieves the conference data structure for conference number `conf-no`.

Error codes

`undefined-conference`
: The conference `conf-no` does not exist or is secret.

4.93 modify-text-info [92] (10) Recommended

```
modify-text-info [92] (( text   :   Text-No;
                         delete :   ARRAY Aux-No;
                         add    :   ARRAY Aux-Item-Input ))
        -> ( );
```

This call deletes the aux-items listed in `delete` from the text `text` and then adds the ones listed in `add` to the text. Either list may be empty, and the call is guaranteed to either completely fail or completely succeed.

Error codes

`login-first`
: Login required before issuing this call.

`no-such-text`
: The text `text` does not exist or is secret.

`aux-item-permission`
: No permission to delete one or more of the items in `delete`, or not enough permissions to add one or more of the items in `add`.

`illegal-aux-item`
: One of the items in `add` is illegal for some reason.

4.94 modify-conf-info [93] (10) Recommended

```
modify-conf-info [93] (( conf   :   Conf-No;
                         delete :   ARRAY Aux-No;
                         add    :   ARRAY Aux-Item-Input ))
        -> ( );
```

This call deleted the aux-items listed in `delete` from the conference `conf` and then adds the ones listed in `add` to the conference. Either list may be empty, and the call is guaranteed to either completely fail or completely succeed.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: The conference `conf` does not exist or is secret.

`aux-item-permission`
: No permission to delete one or more of the items in `delete`, or not enough permissions to add one or more of the items in `add`.

`illegal-aux-item`
: One of the items in `add` is illegal for some reason.

4.95 get-info [94] (10) Recommended

```
get-info [94] ( )
        -> ( Info );
```

This call returns the `Info` structure for the server (see [Info], page 26). Clients should call this in order to find out which conferences are used for presentations and such.

It can be issued without logging in.

Error codes

This call always succeeds.

4.96 modify-system-info [95] (10) Recommended

```
modify-system-info [95] (( items-to-delete : ARRAY Aux-No;
                           items-to-add    : ARRAY Aux-Item-Input ))
          -> ( );
```

This call modifies the aux-item list of the server information (which can be retrieved using `get-info` (see Section 4.95 [get-info], page 103).) It only succeeds when issued by a person with the admin bit set and privileges enabled.

The items in `items-to-delete` are removed, and the items in `items-to-add` are added. This call is atomic; either all deletions or additions succeeded, or none of them is made.

Error codes

`login-first`
: Login requires before issuing this call.

`permission-denied`
: Admin bit not set or privileges not enabled.

`illegal-aux-item`
: Attempt to create an invalid aux item.

`aux-item-permission`
: Attempt to delete an undeletable item or create an uncreateable item.

4.97 query-predefined-aux-items [96] (10) Recommended

```
query-predefined-aux-items [96] ( )
        -> ( ARRAY INT32 );
```

Returns the list of aux-items that have specific definitions in the server. These items are the only items within the restricted tag ranges that can be created. The meanings of the various item types are defined in this document; see Chapter 7 [Aux-Item Types], page 123.

Error codes

This call always succeeds.

4.98 set-expire [97] (10) Experimental

```
set-expire [97] (( conf-no  :   Conf-No;
                   expire   :   Garb-Nice ))
        -> ( );
```

This call sets the `expire` field of the conference `conf-no` to `expire`. This call can only be issued by the conference's supervisor or a privileged user.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: The conference `conf-no` does not exist or is secret.

`permission-denied`
: Not supervisor of conference `conf-no` and not privileged enough to complete the call anyway.

4.99 query-read-texts [98] (10) Recommended

```
query-read-texts [98] (( person     : Pers-No;
                         conference : Conf-No ))
        -> ( Membership );
```

This call is used to find the number of unread texts in a conference. The data it returns is actually a membership structure which specifies which texts have been read. It is up to the client to transform the data to a more usable form. `person` is the person being queried and `conference` is the conference in question.

Calling `query-read-texts` does not require the session to be logged in.

Example:

```
1 98 6 1
=1 4 32 5 11 12 7 93 1 193 1 1 20 133
   3 { 135 136 137 } 5 43 8 3 12 7 93 1 193 1 01000000
```

This example finds the read texts for user 6 in conference 1. The returned data indicates that conference 1 is the fifth conference on the users' membership list ('4'; remember that the `position` starts its count at 0), user last read the conference on Monday July 12th, 1993 at 11:05:32 ('`32 5 11 12 7 93 1 193 1`'), that it is the membership in conference number 1 ('1'), that the person has assigned priority 20 to the conference ('20') and that all articles up to and including local number 133 ('133') plus articles 135, 136 and 137 ('`3 { 135 136 137 }`') have been read. The membership was added by person 5 ('5') at Monday July 12th, 1993 at 03:08:43 ('`43 8 3 12 7 93 1 193 1`') and it is passive ('01000000').

Error codes

`undefined-person`
: `person` does not exist, or no access to person.

`undefined-conference`
: Conference `conference` does not exist, or is secret.

`conference-zero`
: `conference` is zero.

`not-member`
: `person` is not a member of `conference` or insufficient privileges to find out if `person` is a member.

4.100 get-membership [99] (10) Recommended

```
get-membership [99] (( person          : Pers-No;
                       first           : INT16;
                       no-of-confs     : INT16;
                       want-read-texts : BOOL ))
                -> ( ARRAY Membership );
```

This call retrieves the membership record for a list of conferences for a single person. `person` is the person whose memberships are to be retrieved. `first` is the first position in the membership list to retrieve, numbered from 0 and up. `no-of-confs` is the number of membership records to retrieve. If `want-read-texts` is '0', the server will not send the contents of the `read-texts` array of the memberships. (The size will be transmitted, but a single asterisk ('*') will be sent instead of the array itself.)

The server will return a membership list that is shorter than `no-of-confs` if `no-of-confs + first` is larger than the number of conferences the person is a member of. Elements of the member list that the person requesting the list does not have sufficient privileges to see may be cleared. Cleared elements simply have all fields set to zero.

Example:

```
1 99 5 0 3 1
=1 2 { 0 49 14 17 13 8 91 5 255 1 5 255 0 0 * 5
   49 14 17 13 8 91 5 255 1 00000000
   1 20 14 22 17 6 97 4 197 1 6 100 2 0 * 5
   49 14 17 13 8 91 5 255 1 00000000 }
2 99 5 0 1 1
=2 1 { 0 49 14 17 13 8 91 5 255 1 5 255 0 0 * 5
   49 14 17 13 8 91 5 255 1 00000000 }
3 99 5 1 4 1
=3 1 { 1 20 14 22 17 6 97 4 197 1 6 100 2 0 * 5
   49 14 17 13 8 91 5 255 1 00000000 }
```

In this example we retrieve the memberships of person 5. The first call asks for three memberships, starting with number 0. Since this person is only a member of two conferences, the list returned only contains two memberships. The next two calls retrieve a single membership each. The first by asking for only one, and the second by asking for four memberships, starting with number 1.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-person`
: The person `person` does not exist.

`undefined-conference`
: The conference `person` does not exist or is secret.

`index-out-of-range`
: `first` is higher than the index of the last conference in the person's membership list.

4.101 add-member [100] (10) Recommended

```
add-member [100] (( conf-no  : Conf-No;
                    pers-no  : Pers-No;
                    priority : INT8;
                    where    : INT16;
                    type     : Membership-Type ))
          -> ( );
```

Make the person `pers-no` a member of conference `conf-no`. The membership priority is set to `priority` and its position in the membership list is set to `where`.

The membership flags are set to `type`. If the current user is adding a user he isn't supervisor of, the `invitation` bit of `type` is automatically set by the server.

This call can be used to change the priority, position and flags of a conference in the person's membership list if the person is already a member of the conference. The person doing this must either be a supervisor of the affected person, or have sufficient privileges enabled.

Example:

```
1 99 119 0 10 0
=1 1 { 49 14 17 13 8 91 5 255 1 119 255 0 0 * 119 00001111 }
1 100 1 119 250 0 10000000
=1
1 100 119 119 251 1 00000000
=1
1 99 119 0 10 0
=1 2 { 52 30 14 11 5 96 2 162 1 1 250 0 0 * 119 00000000
       49 14 17 13 8 91 5 251 1 119 255 0 0 * 10000000 }
```

This example makes person 119 (me) a member of conference number 1 and changes the priority and some flags of the preexisting membership in conference 119. The priority is set to 250 and the conference is placed first in the membership list. The first and last calls of the example show the membership list for person 119 before and after the calls.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: Conference `conf-no` does not exist or is secret.

`conference-zero`
: `conf-no` is zero.

`undefined-person`
: Person `pers-no` does not exist

`access-denied`
: Not enough permissions or privileges to add members to `conf-no` or to change privileges, position or type of a preexisting membership.

`permission-denied`
: Person `pers-no` is already a member of conference `conf-no`, but the person logged on is not a supervisor and does not have enough privileges to change the priorities of person `pers-no`.

4.102 get-members [101] (10) Recommended

```
get-members [101] (( conf            : Conf-No;
                     first           : INT16;
                     no-of-members   : INT16 ))
          -> ( ARRAY Member );
```

This call returns a list of members of the conference `conf`. `first` is the first index in the membership to return, numbered from zero and up. `no-of-members` is the maximum number of members to return.

Some of the elements of the result may be cleared if the person requesting the information does not have sufficient privileges to see the contents. Cleared elements simply have all fields set to zero.

Example:

```
1 101 1 0 100
=1 4 { 7 7 00000000 8 8 00000000 9 8 00000000
   10 10 00000000 }
1 101 6 0 100
=1 4 { 5 5 01000000 7 7 01000000 9 8 10000000
   10 10 00000000 }
1 101 6 2 2
=1 2 { 9 8 10000000 10 10 00000000 }
```

In this example the client first requests the first 100 members in conference 1. The second request is for the first 100 members of conference 6. The last request is for members 2 and 3 in conference 6.

Error codes

`undefined-conference`
: The conference `conf` does not exist or is secret.

`index-out-of-range`
: `first` is higher than the number of members in `conf`.

4.103 set-membership-type [102] (10) Recommended

```
set-membership-type [102] (( pers       : Pers-No;
                             conf       : Conf-No;
                             type       : Membership-Type ))
        -> ( );
```

This call modifies the type of a membership. The person `pers` membership in conference `conf` is affected. The server may impose arbitrary restrictions on how the membership type may be changed. Typically it will only be possible to clear the `invitation` bit. It is possible that the server will not permit the `secret` bit to be set. Attempting to set a membership type that does not agree with the server's restrictions will result in an error.

Error codes

`login-first`
: Login required before issuing this call.

`conference-zero`
: `conf` is zero.

`undefined-conference`
: The conference `conf` does not exist or is secret or the person `pers` does not exist or is secret.

`permission-denied`
: Insufficient permissions to change the membership of `pers`.

`not-member`
: Person `pers` is not a member of conference `conf`.

`invalid-membership-type`
: The requested membership type `type` was not compatible with restrictions set on the server or on the conference `conf`

4.104 local-to-global [103] (10) Recommended

```
local-to-global [103] (( conf-no              : Conf-No;
                         first-local-no       : Local-Text-No;
                         no-of-existing-texts : INT32 ))
        -> ( Text-Mapping );
```

This call retrieves information that makes it possible to convert `no-of-existing-texts` existing local text numbers starting at `first-local-no` to global text numbers, provided that there are that many local texts.

The `conf-no` parameter specifies which conference to look up local numbers in. `first-local-no` is the first number that the client is interested in. `no-of-existing-texts` is the maximum number of texts the client wants information about. Legal values for `no-of-existing-texts` are 1-255 (inclusive).

The server will return a sparse or dense Text-Mapping depending on the how many deleted texts there are after `first-local-no`.

Example:

```
1 103 93 1 5
=1 1 7 1 1 1 6 { 1003 1005 1009 1029 0 1034 }
2 103 93 1 6
=2 1 63 1 0 6 {
   1 1003
   2 1005
   3 1009
   4 1029
   6 1034
   62 1302 }
3 103 93 50 10
=3 50 70 0 0 2 {
   62 1302
   69 1006 }
```

The above example shows three calls to `local-to-global`. (Extra newlines have been inserted in the result of the two final calls to make the result more readable.)

The first call requests information about the first five existing texts in conference 93. The result contains information about texts in the range 1-7 (including the lower limit, but not the upper), and there are more texts. The server uses the dense form of the `Text-Mapping`. As can be seen from the result, they have local text numbers 1, 2, 3, 4 and 6. The global text number corresponding to local text number 5 is sent as 0, indicating that it doesn't exist.

In the second call, the client requests the same information, but one additional text. The result looks dramatically different, since the next existing text in this example has local text number 62. The result contains information about texts in the range 1-63 (including

the lower limit, and excluding the upper), and there are more texts. The server of course uses the sparse form of the `Text-Mapping`.

The final call shows what happens when `first-local-no` doesn't exist. The result contains information about texts in the range 50-70 (including the lower limit and excluding the upper); only local text number 62 and 69 actually exists in that range. 69 is the highest local text number.

(Note that local text number 69 corresponds to global text number 1006, which is lower than 1302. Situations like this often occurs when `add-recipient` is used.)

Error codes

`login-first`
: Login required before issuing this call.

`long-array`
: `no-of-existing-texts` was larger than 255.

`conf-zero`
: `conf-no` was set to 0.

`local-text-zero`
: `first-local-no` was set to 0.

`undef-conf`
: The conference does not exist, or the client is not allowed to know that it exists.

`access-denied`
: The conference exists, but the client is not allowed to retrieve information about the texts in the conference.

`no-such-local-text`
: `first-local-no` is greater than the highest local text number that ever existed in the conference.

4.105 map-created-texts [104] (10) Recommended

```
map-created-texts [104] (( author               : Pers-No;
                           first-local-no       : Local-Text-No;
                           no-of-existing-texts : INT32 ))
          -> ( Text-Mapping );
```

Return text numbers for existing texts that `author` has written.

Just as each conference has a mapping from local text numbers to global text numbers, each person has a mapping from the N:th text written by him to the global text number. This function can be used to retrieve part of that mapping.

More information and examples may be found in Section 4.104 [local-to-global], page 109.

Error codes

`login-first`
: Login required before issuing this call.

`long-array`
: `no-of-existing-texts` was larger than 255.

`conf-zero`
: `author` was set to 0.

`local-text-zero`
: `first-local-no` was set to 0.

`undef-pers`
: The conference does not exist, or the client is not allowed to know that it exists.

`no-such-local-text`
: `first-local-no` is greater than the highest local text number that ever existed in the conference.

`access-denied`
: The conference exists, but the client is not allowed to retrieve information about the texts in the conference.

4.106 set-keep-commented [105] (10) Experimental

```
set-keep-commented [105] (( conf-no          :   Conf-No;
                            keep-commented   :   Garb-Nice ))
        -> ( );
```

Sets the `keep-commented` field of the conference `conf-no` to `keep-commented`. This call can only be issued by a conference's supervisor or a privileged user.

Error codes

`login-first`
: Login required before issuing this call.

`undefined-conference`
: The conference `conf-no` does not exist or is secret.

`permission-denied`
: Not supervisor of conference `conf-no` and not privileged enough to complete the call anyway.

4.107 set-pers-flags [106] (10) Recommended

```
set-pers-flags [106] (( pers-no     :   Pers-No;
                        flags       :   Personal-Flags ))
        -> ( );
```

Set the flags field of person `pers-no` to `flags`. This call can only be issued by the person supervisor or a privileged user.

Error codes

`login-first`
: Login required before issuing this call.

`conference-zero`
: The `pers-no` parameter is zero.

`undefined-person`
: The person does not exist, or the session does not have permission to know about the person.

`permission-denied`
: Person exists, but the session does not have permission to change the flags.

5 Asynchronous Messages

Asynchronous messages are information messages sent from the server to the client. Most of them are used to inform the client about changes to the database (such as when a new text is created), so that the clients don't have to poll the server for new information. Some have other uses.

The messages with status "O" are included here for historical purposes only. Servers are not required to handle them, and are encouraged to reject them if a client uses it as an argument to `accept-async` (see Section 4.81 [accept-async], page 95).

Clients can select which messages to receive by issuing an `accept-async` call (see Section 4.81 [accept-async], page 95). They can find out which messages are being sent by issuing the `query-async` call (see Section 4.82 [query-async], page 95). Note that the server can send other messages as well. For example, a broadcast message from a person with admin bits set may get through even if the client has not requested broadcast messages.

When a connection is opened some messages are selected by default. Clients should use the `accept-async` call to select which messages they want. Servers are encouraged to preselect the `async-new-text-old`, `async-new-name`, `async-sync-db`, `async-leave-conf`, `async-login`, `async-rejected-connection`, `async-send-message` and `async-logout` messages. These correspond to the useful messages that were sent prior to the introduction of `accept-async` (see Section 4.81 [accept-async], page 95).

An asynchronous message is sent as a colon immediately followed by the number of message parameters, the message number and the message parameters. For example, message number 5 could be sent as

```
:3 5 119 11HDavid Byers 13HDavid C Byers
```

The parameters of each message are listed in the same format as server calls.

5.1 async-new-text-old (1) Obsolete (10)

```
async-new-text-old [0] (( text-no     :   Text-No;
                          text-stat   :   Text-Stat-Old ));
```

This message is sent when a text is created. The text number of the text is sent in `text-no` and the text stat in `text-stat`. This message is sent to all logged-in members of any recipient of the text.

In protocol version 10 this call has been superseded by Section 5.14 [async-new-text], page 116.

5.2 async-i-am-off (1) Obsolete

```
async-i-am-off [1] ( person  :   Pers-No );
```

This message was sent when `person` logged off. It has been replaced by `async-logout` (see Section 5.12 [async-logout], page 115), since this asynchronous message could not differentiate between sessions if the same person was logged in more than once.

5.3 async-i-am-on-obsolete (1) Obsolete

```
async-i-am-on-obsolete [2] (( person          :   Pers-No;
                              conference      :   Conf-No;
                              what-am-i-doing :   HOLLERITH ));
```

This message was sent when `person` changed his `what-i-am-doing` string to `what-am-i-doing` or his working conference to `conference`.

It has been replaced by call number 6, `async-i-am-on` (see Section 5.5 [async-i-am-on], page 114), since this asynchronous message could not differentiate between sessions if the same person was logged in more than once.

5.4 async-new-name (1) Recommended

```
async-new-name [5] (( conf-no     :   Conf-No;
                      old-name    :   HOLLERITH;
                      new-name    :   HOLLERITH ));
```

This message is sent when a person or conference changes names. The conference whose name is being changed is sent in `conf-no`, the old name in `old-name` and the new name in `new-name`.

5.5 async-i-am-on (1) Recommended

```
async-i-am-on [6] ( info :   Who-Info );
```

This message is sent when a session's working conference, `what-i-am-doing` string (see Section 4.5 [change-what-i-am-doing], page 43) or username changes. The new information is sent in `info`.

5.6 async-sync-db (1) Recommended

```
async-sync-db [7] ( );
```

This message is sent once just before the server blocks to save its database and once just after it blocks. There is no good way to tell the difference between the two cases.

5.7 async-leave-conf (1) Recommended

```
async-leave-conf [8] ( conf-no    :   Conf-No );
```

This message is sent to a user when the user's membership in the working conference is removed for any reason. The conference the user is being removed from is sent in `conf-no`.

Earlier versions of the LysKOM Protocol A specifications stated that this message was only sent if the membership was "revoked forcefully". The exact meaning of that phrasing was never specified. The lyskomd implementation has probably always followed the current version of the specification, which means that this message is sent whatever the reason for the removal is.

Possible reasons include:

- The session issued a `sub-member` call (see Section 4.16 [sub-member], page 50).
- Some other session issued a `sub-member` call (see Section 4.16 [sub-member], page 50).

- The conference was deleted.
- The person was deleted.

This message is not sent if a membership is transformed from an active to a passive membership.

5.8 async-login (1) Recommended

```
async-login [9] (( pers-no    :   Pers-No;
                   session-no :   Session-No ));
```

This message is sent when someone logs in. The identity of the person logging in is sent in `pers-no`, and the session number in `session-no`.

5.9 async-broadcast (1) Obsolete

```
async-broadcast [10] (( sender    :   Pers-No;
                        message   :   HOLLERITH ));
```

This message has been superseded by `async-send-message` which is more flexible (see Section 5.11 [async-send-message], page 115). It used to be sent when the administrator (`sender`) broadcasted a string (`message`) to all LysKOM users, but is no longer used.

5.10 async-rejected-connection (1) Recommended

```
async-rejected-connection [11] ( );
```

This message is sent when someone fails to log in because the maximum number of allowed connections has been reached. Some clients may take this as a signal to log out. Administrators should take it as a signal to allow more connections.

5.11 async-send-message (1) Recommended

```
async-send-message [12] (( recipient  :   Conf-No;
                           sender     :   Pers-No;
                           message    :   HOLLERITH ));
```

This message is sent when someone (the `sender`) sends a message string (the `message`). The recipient of the message is sent in `recipient`. If it is zero, then the message was sent to all connections. If it is a conference, then the message is being sent to all logged-in members of that conference. If it is a mailbox then the message is personal and is only sent to members of the mailbox conference.

5.12 async-logout (1) Recommended

```
async-logout [13] (( pers-no      :   Pers-No;
                     session-no   :   Session-No ));
```

This message is sent when someone logs out. `pers-no` is the person logging out and `session-no` is the session in which the person is logging out. This message might also be sent when a session disconnects, even if there is nobody logged on in the session.

5.13 async-deleted-text (10) Recommended

```
async-deleted-text [14] (( text-no    :   Text-No;
                           text-stat  :   Text-Stat ));
```

This message is sent when a text is deleted and the currently logged-in person is a member of one of the recipients. The text number being deleted is sent in `text-no` and the text stat in `text-stat`.

5.14 async-new-text (10) Recommended

```
async-new-text [15] (( text-no    :   Text-No;
                       text-stat  :   Text-Stat ));
```

This message indicates that a new text has been created. The text has number `text-no`, and the text stat is `text-stat`. The message is sent to all logged-in members of any recipient of the text.

5.15 async-new-recipient (10) Recommended

```
async-new-recipient [16] (( text-no     :   Text-No;
                            conf-no     :   Conf-No;
                            type        :   Info-Type ));
```

This message indicates that a new recipient has been added to text `text-no`. The recipient added is `conf-no` and the type of recipient is indicated by `type`. This message is sent to all recipients of the text that are permitted to know about the new recipient.

5.16 async-sub-recipient (10) Recommended

```
async-sub-recipient [17] (( text-no     :   Text-No;
                            conf-no     :   Conf-No;
                            type        :   Info-Type ));
```

This message indicates that a recipient has been removed from text `text-no`. The recipient removed is `conf-no` and the type of recipient is indicated by `type`. This message is sent to everybody that were recipients of the text and that were permitted to know about the recipient.

5.17 async-new-membership (10) Recommended

```
async-new-membership [18] (( pers-no    :   Pers-No;
                             conf-no    :   Conf-No ));
```

This message indicates that the membership for `pers-no` in conference `conf-no` has been added. This message is currently sent only to `pers-no`, but that may change in the future.

See also Section 5.7 [async-leave-conf], page 114.

6 Error Codes

Normal errors are sent in reply to syntactically correct calls to the server. The client should accept any error code in response to any call, even if the error code in question is not listed in the description of the call, and even if the error code in question is not defined in the protocol specification yet.

This table lists the currently defined error codes together with a short explanation. The explanation given below is the default semantics for the error code. It can be updated by the descriptions found for a specific call.

See Section 2.4 [Client-Server Dialog], page 15, for more information about error responses, including the syntax of the error response.

`no-error (0)`
: No error has occurred. `error-status` is undefined. This should never happen, but it might.

`not-implemented (2)`
: The call has not been implemented yet. `error-status` is undefined.

`obsolete-call (3)`
: The call is obsolete and no longer implemented. `error-status` is undefined.

`invalid-password (4)`
: Attempt to set a password containing illegal characters, or to use an incorrect password.

`string-too-long (5)`
: A string was too long (see descriptions of each call.) `error-status` indicates the maximum string length.

`login-first (6)`
: Login is required before issuing the call. `error-status` is undefined.

`login-disallowed (7)`
: The system is in single-user mode. You need to be privileged to log in despite this. `error-status` is undefined.

`conference-zero (8)`
: Attempt to use conference number 0. `error-status` is undefined.

`undefined-conference (9)`
: Attempt to access a non-existent or secret conference. `error-status` contains the conference number in question.

`undefined-person (10)`
: Attempt to access a non-existent or secret person. `error-status` contains the person number in question.

`access-denied (11)`
: No read/write access to something. This might be returned in response to an attempt to create a text, when the recipient conference and its super conferences are read-only, or when attempting to add a member to a conference without enough permission to do so. `error-status` indicates the object to which we didn't have enough permissions to.

`permission-denied (12)`
: Not enough permissions to do something. The exact meaning of this response depends on the call. `error-status` indicated the object for which permission was lacking, or zero.

`not-member (13)`
: The call requires the caller to be a member of some conference that the caller is not a member of. `error-status` indicates the conference in question.

`no-such-text (14)`
: Attempt to access a text that either does not exist or is secret in some way. `error-status` indicates the text number in question.

`text-zero (15)`
: Attempt to use text number 0. `error-status` is undefined.

`no-such-local-text (16)`
: Attempt to access a text using a local text number that does not represent an existing text. `error-status` indicates the offending number.

`local-text-zero (17)`
: Attempt to use local text number zero. `error-status` is undefined.

`bad-name (18)`
: Attempt to use a name that's too long, too short or contains invalid characters. `error-status` is undefined.

`index-out-of-range (19)`
: Attempt to use a number that's out of range. The range and meaning of the numbers depends on the call issued. `error-status` is undefined unless stated otherwise in the call documentation.

`conference-exists (20)`
: Attempt to create a conference or person with a name that's already occupied. `error-status` is undefined.

`person-exists (21)`
: Attempt to create a person with a name that's already occupied. `error-status` is undefined. This error code is probably not used, but you never know for sure.

`secret-public (22)`
: Attempt to give a conference a type with `secret` bit set and the `rd-prot` bit unset. This is an error since such a conference type is inconsistent. `error-status` is undefined.

`letterbox (23)`
: Attempt to change the `letterbox` flag of a conference. `error-status` indicates the conference number.

`ldb-error (24)`
: Database is corrupted. `error-status` is an internal code.

`illegal-misc (25)`
: Attempt to create an illegal misc item. `error-status` contains the index of the illegal item.

`illegal-info-type (26)`
: Attempt to use a Misc-Info type (or Info-Type value) that the server knows nothing about. `error-status` is the type.

`already-recipient (27)`
: Attempt to add a recipient that is already a recipient of the same type. `error-status` contains the recipient that already is.

`already-comment (28)`
: Attempt to add a comment to a text twice over. `error-status` contains the text number of the text that already is a comment.

`already-footnote (29)`
: Attempt to add a footnote to a text twice over. `error-status` contains the text number of the text that already is a footnote.

`not-recipient (30)`
: Attempt to remove a recipient that isn't really a recipient. `error-status` contains the conference number in question.

`not-comment (31)`
: Attempt to remove a comment link that does not exist. `error-status` contains the text number that isn't a comment.

`not-footnote (32)`
: Attempt to remove a footnote link that does not exist. `error-status` contains the text number that isn't a footnote.

`recipient-limit (33)`
: Attempt to add a recipient to a text that already has the maximum number of recipients. `error-status` is the text that has the maximum number of recipients.

`comment-limit (34)`
: Attempt to add a comment to a text that already has the maximum number of comments. `error-status` is the text with the maximum number of comments.

`footnote-limit (35)`
: Attempt to add a footnote to a text that already has the maximum number of footnote. `error-status` is the text with the maximum number of footnotes.

`mark-limit (36)`
: Attempt to add a mark to a text that already has the maximum number of marks. `error-status` is the text with the maximum number of marks.

`not-author (37)`
: Attempt to manipulate a text in a way that required the user to be the author of the text, when not in fact the author. `error-status` contains the text number in question.

`no-connect (38)`
: Currently unused.

`out-of-memory (39)`
: The server ran out of memory.

`server-is-crazy (40)`
: Currently unused.

`client-is-crazy (41)`
: The client used an illegal call sequence, such as calling `set-client-version` more than once.

`undefined-session (42)`
: Attempt to access a session that does not exist. `error-status` contains the offending session number.

`regexp-error (43)`
: Error using a regexp. The regexp may be invalid or the server unable to compile it for other reasons. `error-status` is undefined.

`not-marked (44)`
: Attempt to manipulate a text in a way that requires the text to be marked, when in fact it is not marked. `error-status` indicates the text in question.

`temporary-failure (45)`
: Temporary failure. Try again later. `error-status` is undefined.

`long-array (46)`
: An array sent to the server was too long. `error-status` is undefined.

`anonymous-rejected (47)`
: Attempt to send an anonymous text to a conference that does not accept anonymous texts. `error-status` is undefined.

`illegal-aux-item (48)`
: Attempt to create an invalid aux-item. Probably the tag or data are invalid. `error-status` contains the index in the aux-item list where the invalid item appears.

`aux-item-permission (49)`
: Attempt to manipulate an aux-item without enough permissions. This response is sent when attempting to delete an item set by someone else or an item that can't be deleted, and when attempting to create an item without permissions to do so. `error-status` contains the index at which the item appears in the aux-item list sent to the server.

`unknown-async (50)`
: Sent in response to a request for an asynchronous message the server does not send. The call succeeds, but this is sent as a warning to the client. `error-status` contains the message type the server did not understand.

`internal-error (51)`
: The server has encountered a possibly recoverable internal error. `error-status` is undefined.

`feature-disabled (52)`
: Attempt to use a feature that has been explicitly disabled in the server. `error-status` is undefined.

`message-not-sent (53)`
: Attempt to send an asynchronous message failed for some reason. Perhaps the recipient is not accepting messages at the moment or there are no viable recipients for a group message. `error-status` is undefined.

`invalid-membership-type (54)`
: A requested membership type was not compatible with restrictions set on the server or on a specific conference. `error-status` is undefined unless specifically mentioned in the documentation for a specific call.

7 Aux-Item Types

Some of the aux-items below (mostly the ones that begin with "mx-") are used by mail importers.

‘`content-type [1] (text)`’
: Specifies the content type of a text. Data is a valid MIME type or one of the special LysKOM types (see Chapter 9 [LysKOM Content Types], page 133).

  This item may only be set by the author of a text. The inherit, secret and hide-owner bits are cleared. Only one content-type item can be created per creator.

‘`fast-reply [2] (text)`’
: Data is a string that constitutes a brief comment to the text. This comment should be displayed immediately after the text body.

  An item of this type will never be inherited, can always be deleted, is never anonymous and is never secret.

‘`cross-reference [3] (text, conference, letterbox)`’
: Data is a cross-reference to something else. The contents consist of a letter, a number, and optionally a space and a descriptive text. The letter must be one of T, C or P. T specifies that the cross-reference points to a text; C that it points to a conference; and P that it points to a person. The number is the id of the target of the cross reference. The descriptive text is simply that, a text that describes the cross-reference. For example, "T15 Check this out!" is a cross reference to text 15 with a description that reads "Check this out!", and "T17" is a cross reference without a description.

  The inherit bit is automatically cleared and the item can always be deleted.

‘`no-comments [4] (text)`’
: When this item is set, the author requests that nobody comments the text. This is advisory only; it is still possible to write comments, but clients should advise the user that this is contrary to the author’s wishes. Data should be empty.

  This item may only be set by the author. The secret, hide-creator and inherit bits are automatically cleared.

‘`personal-comment [5] (text)`’
: When this item is set, the author requests only personal comments. This is advisory only; it is still possible to create regular comments, but clients should advise the user that the author prefers a personal comment. Data should be empty.

  This item may only be set by the author. The secret, hide-creator and inherit bits are automatically cleared.

‘`request-confirmation [6] (text)`’
: The author requests that everyone who reads the text confirms having done so by creating read-confirmation items on the text. Clients should ask users if

they wish to confirm having read the text when it is displayed. Data should be empty.

The hide-creator, secret and inherit bits are automatically cleared.

‘`read-confirm [7] (text)`’
: This item can be taken as confirmation that the item creator has read the text to which the item is attached. Clients should never ever create this item without an explicit confirmation from the user that the text has indeed been read.

  The hide-creator, secret and inherit bits are automatically cleared. Once created an item of this type cannot be deleted.

‘`redirect [8] (conference, letterbox)`’
: This item indicates that texts should not be sent to the conference, but be directed to some other target instead. Clients should notify users that attempt to send texts to the conference of the redirect and offer to send the text to the target of the redirect instead. A typical use of this item would be a user that does not read LysKOM very often and would like to advise other users to send e-mail instead.

  Data is PROTOCOL:ADDRESS where PROTOCOL is either "E-mail" or "LysKOM", and ADDRESS is either an e-mail address or a LysKOM conference number. Hopefully we'll be able to replace this with a forwarding mechanism later.

  This item can only be set by the conference supervisor or in the case of a mailbox, the person attached to the mailbox. The hide-creator and secret bits are cleared automatically. Only one redirect can be specified.

‘`x-face [9] (conference, letterbox, server)`’
: Data is the face of the person in compface format. Cool, innit?

  This item can only be set by the conference supervisor or in the case of a mailbox, the person attached to the mailbox. The hide-creator and secret bits are cleared automatically.

‘`alternate-name [10] (text, conference, letterbox)`’
: Data is a string that the client may use as an alternate to the name of a conference or the subject of a text. Note that the server does not match against this name when performing name lookups. Clients should only display alternate names created by the user currently logged on.

  The inherit flag is automatically cleared.

‘`pgp-signature [11] (text)`’
: Data is a PGP signature of the text. The signature should be the equivalent of what ‘`pgp -sba`’ in PGP 2.6.2 generates.

  The secret, hide-creator and inherit bits are automatically cleared. Signatures cannot be deleted once they have been created.

‘`pgp-public-key [12] (letterbox)`’
: Data is the public key of the person. It is desirable that the public key contains a userid of the format "LysKOM <p*n*@*server*>+", where *n* is the number of the

person in the LysKOM server specified in *server*. This rule is currently not enforced.

This item can only be set by the person himself. The hide-creator, secret and inherit bits are automatically cleared.

'`e-mail-address [13] (conference, letterbox, server)`'
Data is an RFC 822-style email address. When set on a mailbox, it should be the email address of the person. If the person has multiple email addresses he may set several e-mail-address aux-items.

The meaning of this aux-item when set on a conference that isn't a mailbox is vague. For a conference that is used as to import a mailing list this should be the email address of the list. For other conferences we haven't really defined a sensible use.

When this aux-item is set on the server it should contain the email address of the administrator (or administrators.)

This aux-item can only be set by the supervisor of a conference or the server administrator. The creator cannot be hidden.

'`faq-text [14] (conference, letterbox, server)`'
Data is a decimal text number, which is a FAQ for the conference, letterbox or server. Creating an item of this type automatically causes creation of a faq-for-conf item.

This item can only be set by the supervisor or server administrator. The hide-creator, secret, and inherit bits are automatically cleared.

'`creating-software [15] (text)`'
Data is the name and version number of the client that created the text. This aux-item can only be set by the author of the text. Once set, it cannot be removed or changed. A typical value would be '`elisp-client 0.47.3`'. Setting the creating-software aux-item is optional.

The data should be the client name, a space, and the client version used in the `set-client-version` (see Section 4.70 [set-client-version], page 88) call. The server may enforce this restriction.

'`mx-author [16] (text)`'
Data is a string containing the name of the author of an imported e-mail, extracted from the `From` header. This aux-item may be missing, if the mail address in the `From` header consists of just the `addr-spec` (see the next aux-item).

Clients should display this instead of the actual author of the text (which will be an importer ID) even if an mx-from aux-item is not present.

Sample contents: `Joe Q. Public` which may come from a `From` header containing `"Joe Q. Public" <john.q.public@example.com>`.

'`mx-from [17] (text)`'
Data is the proper e-mail address (called `addr-spec` in the mail standards) extracted from the `From` header of an imported e-mail.

Clients should display this address together with the `mx-author`, preferably inside angles. If `mx-author` is not present, this address should be shown anyway. It can also be used by clients to construct an address for personal (e-mail) replies to an imported message.

Sample contents: `john.q.public@example.com` which may come from a `From` header containing `john.q.public@example.com` or something like `"Joe Q. Public" <john.q.public@example.com>`.

‘`mx-reply-to [18] (text)`’
Data is the proper e-mail address (called `addr-spec` in the mail standards) extracted from the `Reply-To` header of an imported e-mail. Clients should use this for constructing replies to imported messages.

‘`mx-to [19] (text)`’
Data is a single e-mail address from an email `To` header. Multiple `mx-to` items may be present when multiple recipients are specified in the header. Clients should display these items along with the normal LysKOM recipient headers.

Sample contents: Both `john.q.public@example.com` and `"Joe Q. Public" <john.q.public@example.com>` are valid.

‘`mx-cc [20] (text)`’
Same as `mx-to`, but applies to the `CC` header rather than the `To` header.

‘`mx-date [21] (text)`’
Data is the date and time from the `Date` header of an imported email. Its format is "YYYY-MM-DD hh:mm:ss TZ". YYYY is the year the message was sent, MM is the month, DD is the day, hh is the hour, mm is the minute and ss is the second. This date and time are given in the timezone where the message was sent. TZ is the timezone the date is valid for. It must be of the form "+hhmm" or "-hhmm", where hh is the number of hours offset from UTC and mm is the number of minutes offset. Symbolic timezones are not permitted. The timezone specification is recommended but optional, since it is not always available.

Clients should display this information as the date and time a text was written, since the imported text will have been created at a later time. The date and time when the message was imported would then be displayed elsewhere or not at all.

‘`mx-message-id [22] (text)`’
Data is the `Message-ID` header of an imported e-mail, with whitespace and comments removed. The Message-ID should contain the surrounding angles.

‘`mx-in-reply-to [23] (text)`’
Data is a string containing one item of the same form as the mx-message-id item described above. This is the Message-ID of another mail the current text is a comment to.

Hopefully, this information comes from the `In-Reply-To` header of the imported e-mail, but it could also have been picked from the end of the `References` header line.

If the text really comments more than one other text directly, it is allowed to attach more than one `mx-in-reply-to` items to it.

‘`mx-misc [24] (text)`’
: Data is a string that contains all of the headers of an imported email, including `Subject`, and including those that are redundantly stored in other aux-items. The headers are concatenated with `"\n"`. In other words, this item contains all headers of an imported e-mail as they appear in the message.

  Clients are encouraged to provide a command to display this information.

‘`mx-allow-filter [25] (conference, letterbox)`’
: This aux-item has been declared obsolete. It was intended to supply the importer with information on how to filter incoming messages based on regular expressions matching header lines.

‘`mx-reject-forward [26] (conference, letterbox)`’
: This aux-item has been declared obsolete. It was intended to supplement mx-allow-filter by telling where rejected mails should be sent.

‘`notify-comments [27] (letterbox)`’
: Data is a decimal text number that the user is interested in. Clients should monitor this text for unread comments and present these to the user in some convenient manner. This is typically used by users that want to read comments to some text of theirs as soon as they arrive, rather than in the normal reading order.

  This item can only be set by the owner of the letterbox. No flags are forced or cleared.

‘`faq-for-conf [28] (text)`’
: Data is a decimal number specifying the conference a certain text is a FAQ for. The special number zero denotes that the text is a FAQ for the entire system. Items of this kind can only be created by the LysKOM server itself. Texts with this item are protected from garbage collection.

‘`recommended-conf [29] (server)`’
: Data is a decimal number specifying a conference that new members should automatically be added to, optionally followed by a space and a recommended priority, optionally followed by a space and a membership type. In the future, additional data may be defined; clients should be prepared to accept and ignore a space and any trailing data that may follow the membership type.

  A few examples might clarify what the data may look like:

  `1` Conference number 1.

  `2 32` Conference number 2, with priority 32.

  `3 250 11100000`
  : Conference number 3, with priority 250. The membership should be secret, passive and have the invitation bit set.

  `4 253 01000000 garbage`
  : Conference number 4, with priority 253. The membership should be passive. The client should ignore the trailing garbage; it is reserved

for future extensions. Note that clients are not allowed to create aux-items of this format, but they should be prepared to handle them correctly.

This is a recommendation only; it is up to the client that creates a new person to also add him to the conferences that are specified via `recommended-conf`.

'`allowed-content-type [30] (conference, letterbox, server)`'

Data is a non-negative decimal priority number, followed by a space, followed by a LysKOM content type glob pattern. Clients should send texts to a conference only if the content-type matches any of the `allowed-content-type` glob patterns of that conference.

If the conference doesn't have any `allowed-content-type`, the `allowed-content-type` items of the server should be used. If the server also has no `allowed-content-type` aux-items, it should be interpreted as if a single `allowed-content-type` aux-item with the value '`1 text/plain`' exists.

If there are `allowed-content-type` aux-items with different priority numbers, it is a hint to the client about which content-type is most desirable. Content-types that matches a lower priority number are preferred.

As an example, consider a conference with the following four `allowed-content-type` aux-items:

```
1 text/plain
2 text/x-kom-basic
2 text/enriched
3 text/*
```

These aux-items taken together means that '`text/plain`' is preferred, that '`text/x-kom-basic`' and '`text/enriched`' can be used if there is a reason why '`text/plain`' is inadequate, and that any text type (such as '`text/html`') is acceptable. Other content types, such as '`x-kom/user-area`', should not be used.

The server does not currently enforce the above restriction on the content type of new texts. This mechanism is currently a hint to the client (or to the author of a new text). This may change in the future, if experience shows that it is desirable to have the server enforce the content type.

'`canonical-name [31] (server)`'

This should be set to the official domain name of the server, optionally followed by a colon and the port number. The port number should only be used if a non-standard port is used. Examples: '`kom.lysator.liu.se:8300`', '`kom.lysator.liu.se`'. The intent is that the `canonical-name` should be globally unique among all LysKOM servers. Only a single aux-item of this type can be set on the server. This aux-item can be used by clients that wish to do certain things only when connected to a specific server.

'`mx-list-name [32] (conference)`'

This item should only be used if the main purpose of the conference is to import a single external mailing list. The data should be the email address that is used to post to the list, such as '`bug-lyskom@lysator.liu.se`'.

‘`send-comments-to [33] (letterbox)`’

Normally, when a comment is created, the LysKOM client adds the author of the commented text to the recipient list if he isn't a member of any of the recipients of the comment. By setting this aux-item on the letterbox, a person can request different treatment.

Data is a decimal integer (a conference number) optionally followed by a space character and a decimal number (a recipient type). In the future, additional data may be defined; clients must be prepared to accept and ignore a space and any trailing data that may follow the recipient type. Clients must be prepared to accept items of this type that contain only the conference number.

Comments should be sent to the specified conference number instead of to the author. The value ‘`0`’ is special and means that the comment should not be sent to any special conference, even though that means that the author of the commented text will never see the comment. (The value ‘`0`’ is useful for mail importers and other similar robots.)

The recipient type is the type of recipient to add. If the recipient type is missing, clients should either assume recipient type zero (`recpt`) or ask the user. Legal recipient types are ‘`0`’ for `recpt`, ‘`1`’ for `cc-recpt` and ‘`15`’ for `bcc-recpt` (these are the values used in `Info-Type`). Additional recipient types may be added in the future. Clients should treat unknown or invalid recipient types as if they were missing (i.e. treat them as zero or ask the user).

See Section A.2.4 [Recipients of comments], page 143, for more information about how the client should select the conferences that a comment should be sent to.

This aux-item can only be set by the supervisor of the person.

‘`world-readable [34] (text)`’

Texts that has this aux-item set on them can be read by everybody. It is not even necessary to log in.

Only the author of the text can set this item. The data must be the empty string. The hide-creator, secret, dont-garb and inherit bits are automatically cleared.

‘`mx-mime-belongs-to [10100] (text)`’

Data is a decimal text number that this text is an attachment to. Most likely, the current text is also a comment (or perhaps a footnote) to the text mentioned in the aux-item. A client can use this aux-item to alter the display format of the text (stating that this is an attachment, not a normal comment).

‘`mx-mime-part-in [10101] (text)`’

Data is a decimal text number of a text that is an attachment to the current one. In other words: this is the converse of mx-mime-belongs-to. A client can use this aux-item to know which comments to mark as attachments; the remaining comments are assumed to be normal.

‘`mx-mime-misc [10102] (text)`’

Data is a string that contains all of the MIME headers for the current text. It is set by the importer. The fields are concatenated with `"\n"`.

Clients are encouraged to provide a command to display this.

‘`mx-envelope-sender [10103] (text)`’
: Data is the envelope sender of an imported text. The mail server is supposed to pass this information to the importer, for inclusion here.

‘`mx-mime-file-name [10104] (text)`’
: Data is the file name of an attachment. Most likely, the importer gets this information from a `name` parameter on a `Content-Type` MIME header line.

  Clients are encouraged to use this file name as the default file name when the user chooses to save the text.

See also Appendix C [Some Client-specific Aux-Item Types], page 149, for information about some non-standardized aux-item types.

8 Name Expansion

Names in LysKOM can be expanded according to two rules, regexp matching or KOM conventions.

8.1 Regexp Matching

This type of expansion, used by the `re-z-lookup` call and its predecessors (see Section 4.75 [re-z-lookup], page 91) simply matches `ed`(1) style regular expressions to names in the database to find the list of matching names. The matching is case sensitive.

8.2 KOM Conventions

This type of matching is a little more complicated. Patterns consist of words and parenthesized expressions, and contain implicit wildcards. The `lyskomd` program implements an approximation of theses conventions. Since `lyskomd` is the trendsetter, these semantics are good enough.

The rules are simple. Any parenthesized expressions are removed from the pattern and the names being checked for matches. Then the words of the pattern are examined from beginning to end, and if every pattern word matches the prefix of the corresponding word in the name, the name matches the pattern.

For example “L D” matches “LysKOM (client, server and protocol) Discussion (and) Ideas”, but not “LysKOM Protocol Discussion”.

The matching is case insensitive. Character case is converted according to a collate table in the server. The collate table can be retrieved from the server with the `get-collate-table` call (see Section 4.86 [get-collate-table], page 97).

The current collate table simply maps ISO 8859-1 uppercase and lowercase letters to equivalents, and also considered braces and suchlike equivalent according to swascii rules.

9 LysKOM Content Types

LysKOM defines a few special content types for texts. They are all described in this chapter. In addition to these, clients must support '`text/plain`', should support '`text/enriched`' and are encouraged to support '`text/html`'.

Lines are separated by a single linefeed (ASCII 10) character. The linefeed character is used as a line separator, not as a line terminator. In other words, there should be no linefeed after the last line.

9.1 Reformattable Text

This type of content corresponds to the mime type '`text/x-kom-basic`'. It is raw text that can be reformatted by the client without ill effects, but that can be legibly displayed on a text terminal without formatting.

- Lines must be no longer than 70 characters.
- Each line, except the last, is terminated by a single newline character.
- Two newline characters in succession signal the end of the paragraph.
- There must be no whitespace or newlines after the last character.
- The end of the last line also signals the end of the paragraph.

The following rules apply when reformatting:

- The indentation of the first line of a paragraph is to be applied to all lines in the paragraph.
- If the first line of a paragraph matches `">+ *"` then the string that matched that regexp is to be prefixed to all lines of the paragraph.

This content type was previously erroneously called '`x-kom/basic`', but as far as we know no client ever created texts that were labeled with that name. Please use the new name '`text/x-kom-basic`' instead.

Historical note: version 0.46.1 and earlier of the elisp client created texts labeled with '`x-kom/text`'. Clients may treat that content type as '`text/x-kom-basic`' for improved interoperability.

9.2 The User Area

This content type indicates that the article contains a user area. See Chapter 10 [The User Area], page 135.

10 The User Area

The user area is a regular text that is used to store client-specific information in the server. Most clients use this to store settings a user has made that are specific to a particular server. There are also provisions to store settings that are shared between clients.

The user-area is divided into several sub-blocks. The `common` block is shared by all clients, and its formats and contents are dictated by this protocol specification. Clients may also create one or more other blocks.

In normal texts, everything up to the first line feed is considered to be the subject line. The user area is an exception: it does not contain any subject line.

Each block is encoded as a `HOLLERITH` string. A table of contents is also encoded as a `HOLLERITH` string and added first in the user area. There must be at last one space between each string, and there might be spaces at the beginning and/or end of the user-area. Thus, a user-area made up of three blocks will contain four `HOLLERITH`-encoded strings, each separated by at least one space, and maybe with extra spaces at the beginning or end of the text.

The table of contents consists of one `HOLLERITH`-encoded string for each block in the user-area. Each string contains the name of the corresponding block. There is at least one space between each string, and there may be spaces before the first string and after the last string. Clients must never create two or more blocks with the same name.

This format ensures that clients can copy or read past other clients' blocks without knowing their structure.

Example:

```
13H7Hblock-a 1Hb 4Hasdf 5H hjkl
```

In the above example, there are two blocks: `block-a` and `b`. `block-a` contains four bytes, `asdf`, while `b` contains five bytes: `hjkl` (note the leading space). Below are a few other ways to encode the same user-area:

```
14H 7Hblock-a 1Hb 4Hasdf 5H hjkl
16H 7Hblock-a  1Hb   4Hasdf  5H hjkl
13H1Hb 7Hblock-a 5H hjkl 4Hasdf
```

The first two examples embed extra (redundant) spaces, and the last swaps the order of the two blocks.

The following block names have been defined:

`common` The common block shared by all clients. The format of the common block is described below.

`elisp` The block created by the Emacs lisp client. The format is completely undocumented, but you'll need a lisp reader to parse it.

`WWW-kom` The block created by the web gateway WWW-kom. It has the same syntax as the common block, but the keys and values are not documented.

`rkom` Used by Anders Magnusson <ragge@ludd.luth.se>.

If you're writing a client that uses the user-area, please let us know what you name your client's block.

10.1 The Common Block

This defines the structure of the common block. The common block contains a list of variable settings. Each variable setting consists of a name, some whitespace, and a value. Settings are separated by a line feed character. The values can be of several different types (such as integers, strings, booleans, lists of stuff) but they are all encoded as HOLLERITHs. The reason for this is to simplify for clients that need to ignore the value, and so it is possible to add new value types without confusing old clients.

The grammar below defines the syntax of the common block.

```
common-block    :   settings
settings        :   settings setting
                |   /* empty */
setting         :   variable ' ' value '\n'
variable        :   [A-Za-z-_0-9]+
value           :   HOLLERITH
```

The values contain structure within the HOLLERITH. The following grammar defines the datatypes that are currently used. Clients must be prepared to ignore values that have other datatypes.

```
boolean         :   1 | 0
integer         :   -?[0-9]+
string-list     :   string-list ' ' HOLLERITH
                |   HOLLERITH
```

Currently the following variables are used, but more may be added, and clients must cope with variables they know nothing of in the common block. (Doing so is easy, as all values are encoded as HOLLERITHs.)

`created-texts-are-read`
: True if the user wants texts s/he creates to be marked as read automatically. `boolean`.

`dashed-lines`
: True if the user wants dashed lines around the text body when it's displayed. `boolean`.

`presence-messages`
: True if the user wants messages about people logging in and logging out of LysKOM. `boolean`.

`print-number-of-unread-on-entrance`
: True if the user wants to see the number of unread texts when entering a conference. `boolean`.

`read-depth-first`
: True if the user wants to read text in depth-first order. `boolean`.

`reading-puts-comments-in-pointers-last`
: True if the user wants the client to display comment links after the text body. `boolean`.

`confirm-multiple-recipients`
: True if the user wants the client to ask for confirmation before sending a text to many conferences. `boolean`.

`default-mark`
: The default mark to set on marked texts. `integer`.

`language`
: Preferred languages, in priority order. `string-list`.

  This contains a list of ISO 639 language codes. ISO 639-2 should be used for languages that have a 2-character language code. For other languages, ISO 639-1 should be used. `http://lcweb.loc.gov/standards/iso639-2/langcodes.html` contains a list of the current language codes.

  The client should configure its user interface to use the first language in the list that it supports. Example: a client that supports English and Swedish would use:

  - Swedish if the list is '`2Hfr 2Hsv 2Hen`'
  - English if the list is '`2Hen 2Hsv 2Hfr`'
  - either if the list is '`2Hes 2Hfr`'

11 Membership visibility

The details of the security model isn't properly documented. Much information can be extracted from various parts of the protocol specification, but there is no complete overview. In time, this chapter may become such an overview. For now, it only defines when a membership is visible.

The various requests that return information about a membership can be grouped into three categories:

- Category 1: given a conference, list (some of) the members.
  - `get-members` (see Section 4.102 [get-members], page 107)
  - `get-members-old` (see Section 4.49 [get-members-old], page 76)

  These requests fail if the conference is secret and you don't have access to it, even if a membership should be visible to you according to the rules below.
- Category 2: given a person, list (some of) the conferences that the person is a member of.
  - `get-membership` (see Section 4.100 [get-membership], page 105)
  - `get-membership-old` (see Section 4.47 [get-membership-old], page 74)
  - `get-unread-confs` (see Section 4.53 [get-unread-confs], page 78)

  These requests fail if the person is secret and you don't have access to it, even if a membership should be visible to you according to the rules below.
- Category 3: given a person and a conference, return the membership.
  - `query-read-texts` (see Section 4.99 [query-read-texts], page 104)
  - `query-read-texts-old` (see Section 4.10 [query-read-texts-old], page 46)

  These requests follows the rules below exactly.

The rules for membership visibility are given below. In the explanation, the variables C, P and V are used like this:

- a conference C
- a person P who is a member of the conference C
- a viewer V who wants to find out about P:s membership in C.

The following flags also influences how much V can see:

- the `unread-is-secret` flag of P
- the `secret` flag of P:s membership in C

The following rules determines if a memberhip is visible to V. The first rule that matches is used.

- If V is supervisor of P, then the membership is visible. Both the `unread-is-secret` and `secret` flags are ignored.
- If V is supervisor of C, then the membership is visible. The `secret` flag is ignored. The `unread-is-secret` flag is honored.
- If V is allowed to see both P and C, and if `secret` is unset, then the membership is visible. The `unread-is-secret` flag is honored.
- Otherwise, V is not allowed to se the membership.

Appendix A Writing Clients (informative)

This appendix is not really part of the protocol specification, but it contains some information that may be useful for client writers.

A.1 Common Commands

Most clients will implement certain commands. This main purpose of this section is to get client writers started on some of these commands, and to answer some questions that seem to come up over and over again.

A.1.1 What do I have unread

Each person has a membership list containing the conferences the person is a member of. Each element is an object of type Membership. Among other things it contains the number of the conference, the priority of the membership, when the person most recently marked a text as read in the conference, and which texts the person has read.

The list of read texts consists of two parts: a local text number called `last-text-read` and a list of local text numbers called `read-texts`. The person has marked all texts up to and including `last-text-read` as read, and also the texts listed in `read-texts`. All other texts in the conference are unread. Clients can use either the `query-read-texts` or `get-membership` calls to get membership data.

The standard procedure for finding out which texts are unread is the following:

1. Call `get-unread-confs` for the person. This returns a list of conferences in which the person may have unread texts. This call may return conferences that do not contain any unread texts, but it will never forget to return a conference that does contain an unread text.
2. Call `query-read-texts` for each conference returned in the previous step. This will return the membership data for all the conferences that may contain unread texts.
3. Call `get-uconf-stat` for each conference returned in the first step. The conference status will be needed shortly. Repeat the following steps for each conference.
4. Compare the highest existing local number in the conference (from the conference status) with the `last-text-read` field for the corresponding membership. If the highest existing local text is higher than `last-text-read`, the conference may contain unread texts.
5. Get part of the local to global map from the conference starting at `last-text-read` and ending at the highest existing local number. Every local number in the map that is not read according to the membership data and that has a mapping to a global number is an unread text. You might say that you remove the read texts from the map to get the unread texts.

Take care not to call get-map or get-membership too much since they tend to be expensive operations. Use `get-unread-confs` and `query-read-texts` to minimize the work. Another point to remember is that the server will send asynchronous messages with information about new texts. Clients need to listen to these messages.

A.2 Client Conventions

There are certain conventions that most clients follow, and that users expect. These are not part of the protocol and are subject to change. In particular those conventions that address deficiencies in the protocol will go away when the protocol is updated to correct these deficiencies.

A.2.1 Text formatting

Traditionally the only clients for LysKOM were text-based and only displayed texts exactly as they were stored in the server. Although there are a number of clients now that can wrap lines automatically, texts should still be stored in preformatted style, suitable for display in a monospaced font.

If the client accepts texts from the user and then reformats them, such as a client with an editor with a variable-width font, it should ensure that it follows the following simple rules:

- Lines should be no longer than 72 characters.
- Lines are terminated with a single newline character.
- Paragraphs are separated by two newlines in succession.
- There are no empty lines at the end of the text.

Clients that include editors but do not alter the text before sending it to the server should attempt to ensure that texts confirm to the above conventions.

The same conventions apply to messages sent with the `send-message` call (see Section 4.54 [send-message], page 79).

A.2.2 Content type specification

This convention is understood by all popular clients. If the first line is one of a few predefined strings, then this string specifies the type of text. Currently only the strings "html:" and "enriched:" are supported, specifying '`text/html`' and '`text/enriched`' respectively.

Starting with protocol version 10, this ugly workaround is obsolete. Use aux-items to specify content type instead.

A.2.3 Client-side name expansion

When a client needs to prompt the user for a conference (or person), it could offer the user several ways of specifying the conference:

- By conference number
- By name, using the `lookup-z-name` call (see Section 4.77 [lookup-z-name], page 92).
- By name, using the `re-z-lookup` call (see Section 4.75 [re-z-lookup], page 91).
- By name, transforming the string to a case insensitive regular expression, and then using the `re-z-lookup` call.

It is beyond the scope of this document to specify how the client interacts with the user.

The client should offer the user all the methods of specifying a conference listed above. At the very least, the two first methods should be supported.

A.2.4 Recipients of comments

When a user writes a comment, the user should have full control over which conferences and/or letterboxes the comment is sent to. By default, the recipient list should be all `recpt` recipients. `cc-recpt` and `bcc-recpt` recipients should not be included.

Conferences in the recipient list that have the `original` bit set, and a non-zero `super-conf`, should be replaced by the `super-conf`.

If the author of the commented text isn't a member of any of the new recipients, the client should check if the author has a `send-comments-to` aux item. If it is '`0`', nothing should be done. Otherwise, `send-comments-to` should be a valid conference number, and that conference should be added as a `recpt` recipient. If there is no `send-comments-to` aux-item, the user should be prompted to add the author of the commented text as a `recpt` or `cc-recpt` recipient.

If the author of the new comment is not a member of any of the recipients, the client should check if the author has a `send-comments-to` aux item. If it is '`0`' nothing should be done. Otherwise, `send-comments-to` should be a valid conference number, and the user should be prompted to add that conference as a `recpt` or `cc-recpt` recipient. If there is no `send-comments-to`, or `send-comments-to` is invalid, the user should be prompted to add his letterbox as a `recpt` or `cc-recpt` recipient.

If a recipient is present more than once, all duplicates should be removed.

A.2.5 Order of misc-info groups

The ordering of the groups of `Misc-Info` items are not defined by this protocol. However, when creating a new text, clients are recommended to use this order:

- any `footn-to` items
- any `comm-to` items
- any `recpt` items
- any `cc-recpt` items
- any `bcc-recpt` items

This is the order that the elisp-client normally uses, and users have come to expect that order.

Appendix B Importing and Exporting E-Mail (informative)

E-mail import has been implemented using various programs since the first LysKOM server became operational. Protocol version 10 introduces a lot of aux-items, a large part of which are intended for use by mail importers to enhance the functionality. As of this moment, there is one mail importer (`komimportmail`) that is designed to take full advantage of all the new aux-items.

E-mail export has never been used seriously. The first person to design and implement an exporter gets to rewrite this appendix based on his or her experiences.

B.1 Importing e-mail

The main job of the mail importer is to figure out where to deliver mail, how to handle MIME coding and/or structure and how to deal with threading. During this, it creates one or more texts and a lot of aux-items.

B.1.1 Recipients

Although a mail message contains `To` and `CC` headers, they are not really useful when importing as it is the envelope recipients, not the header recipients, that should be used. To understand this, consider a mail where the `To` header contains a personal mail address. The mail is received using a tool like `procmail` and forwarded to the LysKOM importer. The envelope address will be correct, but the `To` header will still contain the personal address.

The `komimportmail` importer uses addresses like "*number*`@`*server*", where *number* is the number of the recipient and *server* is the mail domain reserved for the LysKOM importer. For backwards compatibility with earlier importers, it is allowed to prepend a "p" before the number. Instead of the number, `komimportmail` can accept a name, as long as the name can be resolved to exactly one conference or letterbox. Before looking up the name, any underscore or period is translated into a space.

Care should be taken when a mail is received more than once. This can happen if a mail is addressed to more than one address. For example, assume that a mail is sent to `john.q.public@example.com` and `sven.svensson@exempel.se`. Two different mail servers handle the two recipients, but both eventually decide to forward the mail to the LysKOM importer (but for different conferences). The LysKOM importer will receive the mail twice, with different envelope recipients.

A solution is to keep a database containing a mapping from `Message-ID` to LysKOM text number for imported messages. If a message is seen more than once, the message is not imported. Instead, recipients are added to the existing text.

On the other hand, that will introduce a security hole, where a person who knows the `Message-ID` of an interesting imported mail can add himself or some open conference as a recipient. Perhaps the importer should check for matching contents before adding recipients.

The importer needs to be careful not to deliver messages to conferences that do not allow messages, even though the server might not complain.

For mail delivery to work for any conference, the importer has to use a privileged person, or it will be unable to deliver mail to secret conferences. A potential problem is that this

leaks secret information from the server. For the time being, the `komimportmail` importer avoids this problem by using an unprivileged person and requiring the members of secret conferences to invite the importer if they want e-mail import to work.

B.1.2 Threading

The importer should do its best to thread messages. When the importer sees a new message it needs to look at the `In-Reply-To` header to see what the message is a reply to. If the `In-Reply-To` header does not exist, or if it exists but does not contain a valid Message-ID, the last valid Message-ID of a `References` header may be used instead.

If the Message-ID of a previously imported e-mail is found, the new text should be made a comment of the replied-to text.

If the Message-ID is of the form defined below (see [Message-ID], page 147), and it refers to a text exported from this server, the new text should be made a comment of the replied-to text.

This means that the importer will probably have to maintain its own database of imported texts that maps the message ID to the text number in the LysKOM database. There is no other way to find the text number for a particular imported text. Fortunately, this is exactly the same database we need to solve the multiple reception problem described above.

It has been noted that messages on some mailing lists arrive in peculiar order, with replies before the original messages. Perhaps this is due to moderation. A smart importer should be prepared to handle this, by adding a comment link when the original message eventually arrives.

One possible solution is to add a new kind of entry to the Message-ID database, mapping a Message-ID to a list of text numbers that should become comments to the message when it is imported.

B.1.3 MIME issues

An importer should try to handle e-mail messages containing MIME appendices as smart as possible. As the current LysKOM model lacks hierarchical structuring inside articles, appendices should probably be imported as comments or footnotes to the main message.

One would think that it is easy to convert the hierarchical MIME structure to a corresponding LysKOM comment tree. However, this would require creating empty interior nodes to attach some comments to.

Therefore, the `komimportmail` importer currently uses a rather naive algorithm: All leaf parts are found. The first one gets to be the main text, and the rest are included as comments to it.

Appendices encoded with Base64 or Quoted-Printable should be decoded.

When creating aux-items like `mx-author`, text coded using the method in RFC 2047 should be decoded.

B.2 Exporting e-mail

As of this writing, an experimental e-mail exporter exists, but it is a fairly recent creation. The author of this document knows very little about how it works, so this section contains very little information.

B.2.1 Message-ID

A standard for Message-ID creation has been established. The general format is:

text-no`.`*port*`.`*exporter*`.`*randomness*`@`*server*

The different parts are explained below:

text-no The text number of the text that was exported.

port
server The `canonical-name` aux item defines a unique name for the LysKOM system that the text was exported from. The *server* and *port* fields are set from it. If no port is specified in the `canonical-name`, the *port* part is set to the empty string.

exporter The name of the software that exported the text.

randomness
A string that ensures that the Message-ID is unique even if the same text is exported several times. This could contain a random string, a sequence number, or something else that makes the Message-ID unique.

Appendix C Some Client-specific Aux-Item Types (informative)

This appendix contains some contributed information from client writers about some of the aux-item types that they use. This information may be updated at any time by the client writers. How much the client writer cares about backward compatibility when they make changes to these aux-item types may be beyond the control of the authors of the Protocol A specification.

If any of the aux-item types defined here becomes widely used by different clients, they should probably be standardized and moved to the Chapter 7 [Aux-Item Types], page 123 chapter.

‘`elisp-client-read-faq [10000] (letterbox)`’
: This item indicates FAQs that the user has read. Data is a decimal number indicating the conference for which a FAQ has been read, followed by a space character and a second decimal number indicating the text number of the FAQ. In the future, additional data may be defined; clients should be prepared to accept and ignore a space and any trailing data that may follow the second number.

  Note that aux-items of this type should always be secret since they may contain information about texts of conferences that are not publicly visible.

  A few examples might clarify what the data may look like:

  `459 11215` FAQ in text 11215 for conference 459 has been read.

  `459 11215 garbage`
  : FAQ in text 11215 has been read. Clients must ignore the trailing garbage. Note that clients are not allowed to create aux-items of this format, but they should be prepared to handle them correctly.

  This aux-item is specific to the elisp client. Other clients are not required to use or understand this item type.

‘`elisp-client-rejected-recommendation [10001] (letterbox)`’
: This item indicates a rejected membership recommendation. Data is a decimal number indicating the conference for which the recommendation was rejected. In the future, additional data may be defined; clients should be prepared to accept and ignore a space and any trailing data that may follow the second number.

  Note that aux-items of this type should always be secret since future extensions may contain sensitive data.

  A few examples might clarify what the data may look like:

  `459` A recommendation for conference 459 has been rejected.

  `459 garbage`
  : A recommendation for conference 459 has been rejected. Clients must ignore the trailing garbage. Note that clients are not allowed to create aux-items of this format, but they should be prepared to handle them correctly.

This aux-item is specific to the elisp client. Other clients are not required to use or understand this item type.

Appendix D Future changes (speculative)

While useful and stable, this protocol is far from perfect. Here is a short list of things the current developers would like to change in future versions of the protocol. The list is not sorted.

All changes will be made in a backwards compatible way. Clients will still be able to use the old requests.

- Cryptography! LysKOM is sometimes used for sensitive information. It is unacceptable that everything is sent in the clear. Should we use TLS? This area needs further study. (This is bug 133.)
- The information contained in the `Misc-Info` array should be integrated into the `Text-Stat`. There should be one array of recipients, one of texts that comments this text, and so on. This would make the `Misc-Info` type obsolete. (This is bug 134.)
- The aux-items can be very large. It was a mistake to include them in the `Text-Stat`. They should probably be separated from the `Text-Stat`; retrieving the `Text-Stat` should only indicate which aux-items that exists. (This is bug 135.)
- There is too few asynchronous messages. There are many situations where something can change without a client noticing it.

  There is more than one way to fix this, and it is not known which way is the best. (This is bug 93.)
- The super conference is used for two purposes: to indicate where followups of original conferences should be sent, and to indicate where articles created by persons that are not permitted submitters should be sent. Occasionally one wants to send followups to one conference and unauthorized original articles to another conference.

  One way to fix this is to remove the `original` bit and introduce a `followups-to` field in the `Conference`. Doing this in a backwards-compatible way requires some thought... (This is bug 136.)
- The security system is too complex, yet unable to do many useful things, and should be rethought. (This is bug 137.)
- The `last-text-read` and `read-texts` fields of a `Membership` is a very poor way to store information about what is read. Consider the case where somebody has read everything up to text 100, and text 102-2002. Using the current protocol specification more than 2000 numbers must be transmitted to convey that information. A much more attractive solution would be to send a list of ranges of read texts. (This is bug 52.)
- The `mark-as-read` (see Section 4.28 [mark-as-read], page 60) call is a reasonably good way to mark a single text as read (but one could argue that it should take a single `Text-No` and an `ARRAY Conf-No` as arguments), but there is no easy way to undo such an operation. You can use `set-last-read` (see Section 4.78 [set-last-read], page 92) followed by a number of `mark-as-read` calls to get the desired effect, but it would be nice to have a `mark-as-unread` call. (This is bug 53.)
- If the client issues several request without waiting for a reply, the replies will nevertheless always arrive in the same order as the requests were sent in. If a threaded version of the LysKOM server is ever made, it could possibly process two requests from the same

client at the same time, and the answers could be returned in any order. However, the current clients are probably not written so that they can handle such reordering.

In the future, a new call might be added, so that a client can give the server explicit permission to reorder the replies. The client would then have to rely on the `ref-no` to match each reply to the corresponding call. (This is bug 138.)

For more information about potential future changes, see Bugzilla @ Lysator.

Appendix E Protocol Version History (informative)

E.1 Protocol version 10 (first implemented in lyskomd 2.0.0)

Error codes

The error codes are now documented. Several error codes were changed to more sane values while documenting the new behavior.

New Server Calls

These new calls have status Recommended.

- 85=get-collate-table
- 86=create-text
- 87=create-anonymous-text
- 88=create-conf
- 89=create-person
- 90=get-text-stat
- 91=get-conf-stat
- 92=modify-text-info
- 93=modify-conf-info
- 94=get-info
- 95=modify-system-info
- 96=query-predefined-aux-items
- 98=query-read-texts
- 99=get-membership
- 100=add-member
- 101=get-members
- 102=set-membership-type
- 103=local-to-global
- 104=map-created-texts
- 106=set-pers-flags

These new calls have status Experimental.

- 97=set-expire
- 105=set-keep-commented

Name changes

Old name	New name
5=create-person	5=create-person-old
9=query-read-texts	9=query-read-texts-old
10=create-conf	10=create-conf-old
13=get-conf-stat-old	13=get-conf-stat-older
14=add-member	14=add-member-old
26=get-text-stat	26=get-text-stat-old

28=create-text	28=create-text-old
36=get-info	36=get-info-old
46=get-membership	46=get-membership-old
48=get-members	48=get-members-old
50=get-conf-stat	50=get-conf-stat-old
59=create-anonymous-text	59=create-anonymous-text-old

Status change

The following calls have change status from Experimental to Recommended.

- 58=get-last-text
- 77=set-last-read
- 78=get-uconf-stat

The following calls have changed status from Recommended or Experimental to Obsolete.

- 5=create-person-old
- 9=query-read-texts-old
- 10=create-conf-old
- 14=add-member-old
- 26=get-text-stat-old
- 28=create-text-old
- 36=get-info-old
- 46=get-membership-old
- 47=get-created-texts
- 48=get-members-old
- 50=get-conf-stat-old
- 59=create-anonymous-text-old

New and Modified Structures

- Aux-Item
- Aux-Item-Input
- Conference
- Info
- Member
- Membership
- Membership-Type
- Misc-Info
- Text-Stat

Renamed Asynchronous Messages

A ‘`async-`’ prefix has been added to the name of all asynchronous messages. In addition, 0=new-text has been renamed to 0=async-new-text-old, and it is now considered obsolete. Clients should use 80=accept-async to listen to 15=async-new-text instead.

New Asynchronous Messages

- 14=async-deleted-text
- 15=async-new-text
- 16=async-new-recipient
- 17=async-sub-recipient
- 18=async-new-membership

Notes

- Since protocol version 9 setting a priority of zero for a conference was supposed to indicate passive membership in a conference. It was largely up to the client to implement this. True passive memberships have been introduced in this protocol version through the Membership-type extension to the Membership type. In order to maintain compatibility with clients that interpret priority 0 as passive membership, the old calls `add-member-old` (see Section 4.15 [add-member-old], page 49) and `get-membership-old` (see Section 4.47 [get-membership-old], page 74) perform magic, translating between priorities and membership types. The magic is documented with each call.

E.2 Protocol version 9 (first implemented in lyskomd 1.9.0)

New functionality

- The server shall now reply with error `not-implemented` when a client attempts to use an unimplemented call. This feature requires that the client uses newline as call terminator.

Added Commands

- 79=set-info: Can change server information.
- 80=accept-async: Can select asynchronous messages to receive.
- 81=query-async: Can query which messages are being send.
- 82=user-active
- 83=who-is-on-dynamic
- 84=get-static-session-info

Changed names

- `change-conference` was previously called `pepsi`. The name was changed, but not the functionality.

Status change

- 63=`who-is-on-ident` is now considered obsolete.
- 64=`get-session-info-ident` is now considered obsolete.

E.3 Protocol version 8 (first implemented in lyskomd 1.8.0)

Added Functionality

- 30=add-recipient: Can change `recpt` to `cc-recpt` and vice versa.
- 21=set-conf-type: Accepts `Conf-Type` and `Extended-Conf-Type`.

- 10=create-conf: Accepts `Conf-Type` and `Extended-Conf-Type`.

New Commands

- 77=set-last-read
- 78=get-uconf-stat

E.4 Protocol version 7 (first implemented in lyskomd 1.7.0)

Added Functionality

- 53=send-message: Recipient can be a conference or a person.

New Commands

- 74=re-z-lookup
- 75=get-version-info
- 76=lookup-z-name

Other

- The asynchronous message 1=i-am-off has been removed

E.5 Protocol Version 6 (first implemented in lyskomd 1.4.0)

New Calls

- 67=lookup-person
- 68=lookup-conf
- 69=set-client-version
- 70=get-client-name
- 71=get-client-version
- 72=mark-text
- 73=unmark-text

E.6 Protocol Version 5 (first implemented in lyskomd 1.3.0)

New Calls

- 65=re-lookup-person
- 66=re-lookup-conf

E.7 Protocol Version 4 (first implemented in lyskomd 1.1.1)

New Calls

- 62=login
- 63=who-is-on-ident
- 64=get-session-info-ident

E.8 Protocol Version 3 (first implemented in lyskomd 1.1.0)

New Calls

- 61=find-previous-text-no
- 60=find-next-text-no
- 59=create-anonymous-text
- 58=get-last-text

E.9 Protocol Version 2 (first implemented in lyskomd 0.30.0)

New Calls

- 57=set-user-area

E.10 Protocol Version 1 (first implemented in lyskomd 0.29.2)

New Calls All calls from 0–56.

Appendix F Document Edition History (informative)

10.7: 2002-11-03

Fixed errors: The description of common block of the user area was just plain wrong. See Chapter 10 [The User Area], page 135, for the updated specification.

The documentation for no-of-created-texts in the `Person` structure was wrong. See [Person], page 28.

The documentation for the privilege bits `create-conf` and `create-pers` stated that they are by default on. In fact, they are by default ignored, but off. See Section 1.6 [Security], page 8.

The example for `re-z-lookup` (see Section 4.75 [re-z-lookup], page 91) was wrong.

An example for `lookup-z-name` (see Section 4.77 [lookup-z-name], page 92) was wrong.

Protocol change: Added a new `"language"` value to the common block of the user area. See Chapter 10 [The User Area], page 135.

New aux-items: `world-readable` [34]. See Chapter 7 [Aux-Item Types], page 123.

`elisp-client-read-faq` [10000] and `elisp-client-rejected-recommendation` [10001]. See Appendix C [Some Client-specific Aux-Item Types], page 149.

Modified aux-item: `send-comments-to` [33] may now also contain an optional recipient type. See Chapter 7 [Aux-Item Types], page 123.

Previously undocumented stuff: The rules for when a membership is visible are now documented. See Chapter 11 [Membership visibility], page 139.

Documented the `unread-is-secret` flag of `Person`. See [Personal-Flags], page 28.

The documentation for `get-person-stat-old` was improved (see Section 4.7 [get-person-stat-old], page 44).

The Message-ID of exported texts is now documented. See [Message-ID], page 147.

A client should offer to add the author as a recipient of a text he is created if he isn't a member of any of the recipients. See Section A.2.4 [Recipients of comments], page 143.

There should be no linefeed after the last line of a text. See Chapter 9 [LysKOM Content Types], page 133. See Section 9.1 [Reformattable Text (text/x-kom-basic)], page 133.

Added a recommendation for how groups of `Misc-Info` items should be sorted. See Section A.2.5 [Order of misc-info groups], page 143.

Editorial changes: Several spelling errors, typos, et c were fixed.

Distributed with lyskomd 2.0.7.

10.6: 2002-03-29

Fixed errors: The format of the user area was plain wrong. The examples for `query-read-texts` and `get-membership` were wrong. The `idle-time` field is only affected by the `user-active` request, not by all activity (see Section 3.14 [Session Information], page 36).

Protocol change: Expanded the documentation of `super-conf`. See Section 3.5 [Conference Status Types], page 23. Simplified the rules for `super-conf`; a setting of 0 no longer means anything. A new section contains a summary of how a client should decied where to send a comment, and it contains the new rules. Section A.2.4 [Recipients of comments], page 143. Most clients already implemented the new rules; the old rules were overly complex.

New aux-item: `send-comments-to` [33].

Modified aux-item: `faq-text` [14] may now be set on a letterbox.

Previously undocumented stuff: Expanded the documentation of `permitted-submitters`. See Section 4.20 [set-permitted-submitters], page 54.

Clarify that a person is a supervisor of himself, except for the `set-supervisor` call; see Section 1.2 [Conferences], page 3.

Document the `change-name` privilege bit; see Section 1.6 [Security], page 8.

Document that `last-login` is also updated on logout; see Section 3.9 [Person Status Types], page 28.

Clarify that the address part of `redirect` [8] is a conference number.

Compatibility info: Mention that the elisp client used to enter texts as '`x-kom/text`' instead of '`text/x-kom-basic`'.

Administrativia: Updated "Future changes" (see Appendix D [Future changes], page 151), and mention that we now use Bugzilla to keep track of bugs and feature requests.

Editorial changes: Added some of Texinfo markup and cross references. Several spelling errors, typos, et c were fixed.

Distributed with lyskomd 2.0.6.

10.5: 2001-09-30

A new section (see Section A.2.3 [Client-side name expansion], page 142) in the informative Client Conventions appendix was written. Distributed with lyskomd 2.0.5.

10.4: 2001-05-24

Fixed errors: Fixed the description of the `old-pwd` argument to `set-passwd` (see Section 4.9 [set-passwd], page 45). The content-type '`x-kom/basic`' was renamed to '`text/x-kom-basic`'.

Previously undocumented stuff: Documented several previously undocumented aspects of the protocol: the hello string used during connection establishment, the `items-to-delete` and `items-to-add` arguments of `modify-system-info`, the `type` argument of `add-member` (including the fact that the invitation membership flag is automatically set in some circumstances), that a client can issue many requests at once (and that the replies are sent back in order).

New stuff: Registered the "rkom" user-area block.

New aux-items: The following predefined aux-items were added: `canonical-name` [31], `mx-list-name` [32] `mx-mime-belongs-to` [10100], `mx-mime-part-in` [10101], `mx-mime-misc` [10102], `mx-envelope-sender` [10103] and `mx-mime-file-name` [10104]. Those aux-items with a number higher than 10000 were previously documented in this document, but they now have the status of predefined aux-items.

Editorial changes: Lots of editorial changes needed to publish an online version on the web at http://www.lysator.liu.se/lyskom/protocol/. Many minor syntax errors corrected, and much missing Texinfo markup added. The document now makes heavy use of Texinfo macros. Due to bugs in the current version of '`texinfo.tex`' this file cannot currently be typeset using TEX. (All bugs are reported to the Texinfo maintainers.)

`get-membership-old`, `get-membership` and `login` used to take an argument of type `BITSTRING(`*a-single-field*`)`. The type has now been changed to `BOOL` instead, to simplify the description. The encoding of the protocol is unchanged.

The indices are now joined into a single index. More terms have been added to it. Many sections of text were moved around to make it easier to find information. Some non-normative information was moved to appendices. Removed a few non-normative empty sections.

This edition was published on the web; no lyskomd release was imminent when the edition was finished. Future editions will also be published on the web; most of them will likely be published at the same time as a lyskomd release is made. They will continue to be included in the lyskomd releases.

10.3: 2000-09-09

Several aux-items can be set on letterboxes and not only conferences. A few can now be set on the server. The `allowed-content-type` and `recommended-conf` aux-items were added. The `mx-allow-filter` and `mx-reject-forward` aux-items are marked obsolete. Text regarding mail import was improved, based on actual experience in writing an email importer. Reserved a range of aux-items for komimportmail. Several minor corrections and clarifications made. Distributed with lyskomd 2.0.4.

10.2: 1999-07-23

Some typos and other minor errors were fixed. Distributed with lyskomd 2.0.2.

10.1: 1999-07-12

Call `sub-comment` was incorrectly marked obsolete. This has been corrected. Regexps are case sensitive. The Info-Type enumeration was introduced in the description of the protocol. (Previous versions of the protocol had broken definitions of `add-recipient`, `async-new-recipient` and `async-sub-recipient`.) Distributed with lyskomd 2.0.1.

10.0: 1999-06-27

The specification was translated to English and converted to Texinfo by David Byers. Protocol version 10. Distributed with lyskomd 2.0.0. Note: this edition incorrectly marked the `sub-comment` call as obsolete, and stated that regexp

lookup was case insensitive. Both statements were wrong, and has since been fixed.

9.0: 1996-08-04
: Protocol version 9. Distributed with lyskomd 1.9.0.

8.0: 1995-11-10
: Protocol version 8. Distributed with lyskomd 1.8.0.

7.1: 1995-01-08.
: Protocol and document revision history were added by Per Cederqvist. Outline mode was used to make the document more manageable. This version was distributed with lyskomd 1.7.1.

7.0: 1994-12-31.
: The first specification with a version number. All calls that had been added since 1991-06-25 were documented. Pell and Per Cederqvist did the deed. This version was distributed with lyskomd 1.7.0.

1993-05-19.
: Linus Tolke wrote comments for some calls that were without comments.

1992-07-06.
: Linus Tolke converted the document to ISO 8859-1.

1991-08-12.
: Per Cederqvist started using version control for documentation.

1991-06-25.
: Lars Aronsson documented the protocol that was in use at the time.

Index

H

I

L

M

P

Q

R

S

T

U

V

W