Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 14, 2016

I. Radu, Ed.

October 12, 2015

Advanced Groupware Access Protocol
draft-iulian-advanced-groupware-access-protocol-12

Abstract

The Advanced Groupware Access Protocol, (AGAP) allows a client to access and store electronic mail messages, contacts, events, files, and configurations on a server. The electronic mail messages can be grouped in folders. AGAP also provides the capability for an offline client to resynchronize with the server.

AGAP does not specify a means of posting electronic mail messages; this function is handled by a mail transfer protocol such as SMTP [RFC2821] . It also does not specify a means for exchanging messages with contacts that are reported as being online; this function is handled by an instant messaging protocol such as XMPP [RFC3921] .

AGAP includes the following operations for electronic mail messages: creating, deleting, renaming, moving and coping mail folders; checking for new messages; permanently removing messages; moving and coping messages between folders; fetching information about a message; setting and clearing tags for messages; searching in messages; retrieving only a part of a message; marking messages as SPAM; deleting attachments from a message.

AGAP includes the following operations to manipulate the contacts: creating, deleting, moving, coping, tagging, and searching contacts; checking if a contact is online; fetching information about a contact.

AGAP includes the following operations related to the use of the events: creating, deleting, moving, coping and tagging events in calendar; fetching events details; searching for events.

All items are read and written in format XML encoded UTF-8 [RFC3629] and each item is identified by a unique alphanumeric identifier.

AGAP is designed to support access only to a single server per connection. It is also designed to balance the volume of text exchanged between the server and clients and its readability by humans for debugging.

Table of Contents

1. How to Read This Document

1.1. Organization of This Document

This document is written from the point of view of someone implementing an AGAP client or server, and also from the point of view of a server administrator. The protocol overview (chapter 2) presents all aspects related to a correct implementation (like the maximum length of a command or response line, charset used). The material in chapter 3 through 5 provides the states in which can be a connection at a moment, respectively what commands are valid in each state and their valid responses. Chapter 6 makes a summary of the return codes for each command. The implementers find in chapter 7 samples of conversations so that they can test the compliance of their applications with this standard.

1.2. Conventions Used in This Document

Document conventions are noted in this chapter. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT","MAY", and "OPTIONAL" in this document are to be interpreted as described in 'Key words for use in RFCs to Indicate Requirement Levels' [RFC2119] . The word "CAN" (not "MAY") is used to refer to a possible circumstance or situation, as opposed to an optional facility of the protocol.

"User" is used to refer to a human user. "Client" refers to the software being run by the user. "Server" refers to the software responding to the client requests. In examples, "C:" and "S:" indicate lines sent by the client and server respectively. "Connection" refers to the entire sequence of client/server interaction from the initial establishment of the network connection until its termination. "Conversation" is an exchange of commands and responses between the client and the server. "Account" defines all folders and their content that can be accessed from Authenticated State. All references to characters order is according to the UTF-8 [RFC3629] specification.

2. Protocol Overview

2.1. Charset Used for Commands and Responses

All data exchanged between the server and the client is done using strings encoded UTF-8 [RFC3629] . If the server or client send a string incorrect encoded then the other side can close immediately the connection.

2.2. Maximal Length of a Command or Response Line

A command or response consists of a line of text that has a maximal length of 1024 characters (including line end). A line of text is ended with the character LF (0x0A). There can be optionally a CR character (0x0D) before the LF character. If the server or client sends a line with a length greater of 1024 then the other side can close immediately the connection.

2.3. Numbers in Commands and Responses

The numbers that are used in commands are signed integers on 32 bits. The valid values are between -2,147,483,648 and 2,147,483,647.

2.4. Regular Expressions in Commands

Following is a resume of all regular expression rules that CAN be used by the commands defined in this standard:

```
       Logical operators:
   XY       X followed by Y
   X|Y      Either X or Y

       Predefined character class:
   .        Any character (does not match line terminators)

       Characters:
   x        The character x
   \\       The backslash character
   \xhh     The character with hexadecimal value 0xhh
   \uhhhh   The character with hexadecimal value 0xhhhh
   \t       The tab character ('\x09')
   \n       The newline (line feed) character ('\x0A')
   \r       The carriage-return character ('\x0D')

       Character classes:
   [abc]     a, b, or c (simple class)
   [^abc]    Any character except a, b, or c (negation)
   [a-zA-Z]  a through z or A through Z, inclusive (range)

       Boundary matchers:
   ^        The beginning of a line
   $        The end of a line
   \b       A word boundary
   \B       A non-word boundary

       Greedy quantifiers:
   X?       X, once or not at all
   X*       X, zero or more times
   X+       X, one or more times
   X{n}     X, exactly n times
   X{n,}    X, at least n times
   X{n,m}   X, at least n but not more than m times

       Reluctant quantifiers:
   X??      X, once or not at all
   X*?      X, zero or more times
   X+?      X, one or more times
   X{n}?    X, exactly n times
   X{n,}?   X, at least n times
   X{n,m}?  X, at least n but not more than m times
```

Figure 1

2.5. Unique Identification Numbers (UID)

The length of an UID is between 1 and 32 characters.

The UIDs MUST to be unique only between items from the same folder.

The characters accepted for building an UID are only all 26 Latin letters (A-Z) in lowercase and uppercase and all 10 Latin digits (0-9). An UID is case sensitive and it is the same for each connections, except after server change it and announce the change by changing the ECID assigned to the corresponding folder.

Any new item MUST have a bigger UID as all other existing items in the selected folder. The sorting is made according UTF-8 [RFC3629] (digits before letters and uppercase letters before the lowercase letters - 0..9A..Za..z). A shorter UID is before a longer one (9234 before 02345) and any zero (0) before a number is take into account by the server when two UIDs are compared.

We get an approximately maximum number of 4.50+e+17 unique combinations for 32 characters long UIDs. We get a maximum number of 3381098545 unique combinations for 8 characters long UIDs.

2.6. Folder Change Identification Numbers (FCID)

An FCID has the same format as a normal UID and each new value of an FCID is bigger as the precedent one (as is described for UIDs). An FCID is changed for a folder when the structure of the folder is changed (subfolders are added or removed). The FCID of a folder is not changed if it is changed the (sub)child of one of its children.

2.7. Entries Change Identification Numbers (ECID)

An ECID has the same format as a normal UID and each new value of an ECID is bigger as the precedent one (as is described for UIDs). If the last ECID has already had the biggest valid UID value then its new value can be the first valid UID value. An ECID is changed for a folder when items ware added or removed. Or when the server had changed the UIDs assigned to the items. This can be necessary if, for example, there is a new item and already last valid UID was assigned to an other item. The new UIDs must keep the items in the same order as before the renumbering.

2.8. Representation of Text and Binary Content in XML Bodies

Binary content must be encoded using the BASE64 [RFC4648] method and the corresponding tag must have the ENCODED attribute set to "base64" (if it is not assumed as default encoding method). All BASE64

encoded content found in one line must have a length divisible by 4. The server can refuse an encoded content having a length not divisible by 4 in a line.

A text content can be passed as it is (UTF-8 [RFC3629]) or it can be encoded using the BASE64 [RFC4648] method. The corresponding tag must have the ENCODED attribute set to "utf-8", in case of plain text, and to "base64", if the content was encoded using the BASE64 method.

2.9. SRV record

An SRV record (Service record) defines the location in the DNS (Domain Name System) of a server providing a specified service. It is defined in RFC 2782 [RFC2782] . A non-secured port is searched with _agap and a secured port with _agaps as _service name.

An SRV record has the form:

```
_service._proto.name ttl IN SRV priority weight port target
```

- o service - the symbolic name of the desired service;
- o proto - the transport protocol of the desired service (TCP or UDP);
- o name - the domain name for which this record is valid;
- o ttl - standard DNS time to live field (usually 86400);
- o priority - the priority of the target host (lower value means more preferred);
- o weight - a relative weight for records with the same priority;
- o port - the TCP or UDP port on which the service is present;
- o target - the canonical hostname of the machine providing the service.

The following textual item can be used to specify the location of an AGAP service:

```
_agap._tcp.example.com. 86400 IN SRV 0 5 143 agapserver.example.com.
```

The following textual item can be used to specify the location of a secured AGAP port (via SSL):

```
_agaps._tcp.example.com. 86400 IN SRV 0 5 993 agapserver.example.com.
```

2.10. Folders

2.10.1. Naming

All folder names are case sensitive and they are encoded according to UTF-8 [RFC3629] .

A backslash (\) does not escape the character after it (it has no special meaning).

For building a folder name, the user CAN use all UTF-8 [RFC3629] characters with a value bigger then 0x1f (white space is the first allowed character), but with the exception of the slash (/ 9x2F), back slash (\ 0x5C), multiplication sign (* 0x2A), and question mark (? 0x3F).

The following folder names are also not accepted: '.', and '..'.

2.10.2. Hierarchy

None of the reserved folders can have subfolders, exception makes TRASH that must store also deleted folders and FILESHARE that holds ordinary files.

The character used for delimiting path levels is the slash (/). A path that starts with '/' represents an absolute path. All other are relative to currently selected folder (with SLCT).

If there is no folder currently selected then the client MUST use only absolute paths. It is recommended for a client to use always absolute paths.

2.10.3. Folder Types

The following folder types are defined by this standard:

o addresses - ADDR - holds addresses of locations;

o audio notes - AUDN - holds notes with an attached audio;

o bookmarks - BKMK - holds an URL with an attached note;

o calendar - CALE - holds events;

o configuration - CONF - holds user accounts configuration (the client is free to store all information it needs for providing

roaming);

- contacts - ADBK - holds contact information;
- files - FILE - holds files that have no special meanings for the server;
- filter - FILT - holds the definition of a filter (it does not allow subfolders);
- folder - FOLD - contains only subfolders;
- links - SLNK - holds links to actual items.
- locations - ADDR - contains addresses of locations;
- Java application - JAPP - holds jars that can be launched by the client;
- journal - JRNL - holds journal items;
- message - MESG - holds email messages;
- notes - NOTE - holds short texts;
- radio - RDIO - holds links to radio stations streaming via http;
- tasks - TASK - holds tasks;
- timezone - TLOC - holds timezones;
- versioned documents - DOCU - holds multiple versions of a document;
- weather - WLOC - holds a location for providing weather forecasts for it.

Each of these types allow for subfolders in them.

2.10.4. Reserved Folders

All the following reserved folders are located in the root of the user's account:

- CALENDAR - CALE - holds the main calendar of the user (PUBLIC);
- CONFIGURATION - CONF - holds account configuration;

- CONTACT - ADBK - holds the main contact list (PUBLIC);
- DRAFT - MESG - holds templates for email messages;
- FILESHARE - FILE - holds shared files (PUBLIC);
- INBOX - MESG - holds all new email messages (PUBLIC);
- JOURNAL - JRNL - holds the main journal (PUBLIC);
- JUNK - MESG - holds all email messages marked as SPAM or VIRUSED by user or the server;
- NOTE - NOTE - holds short texts (PUBLIC);
- OUTBOX - MESG - holds all email messages that wait to be sent;
- SENT - MESG - holds copy of sent email messages;
- TASK - TASK - holds the main tasks list (PUBLIC);
- TRASH - MESG - holds all deleted email messages;

A client can use different names for these folders when display them so that the client application can use localization and standard or customized names for them. If this is the case, then the user cannot create a folder, in the root of his account, with the same name as the real (reserved) name of the folder.

2.11. Tags

2.11.1. Syntax

The client can set tags for folder and folder items. The tags of a folder are reported by the STAT command and can be read with GFTG. The tags of an item can be get with GTAG. The tags of a server can be set with SFTG, and the tags for folder items with STAG.

The format of a tag is a name optionally followed by the equal sign (=) and a value. Each time a tag is set, the new value replace the old one. All tags that have no value assigned are returned only as names. Assigning an empty value to a tag makes it to return a name followed by the equal sign and no value. Setting a tag without a value for an item which previously had the same tag with a value makes the tag to lose its value and to be returned as name only (without the equal sign).

The characters accepted for building a TAG are only all 26 Latin

letters (A-Z) in uppercase, all 10 Latin digits (0-9) and the minus sign (-). A TAG is case insensitive. Its length is between 1 and 32 characters.

The characters accepted for a TAG value are only all 26 Latin letters (A-Z) in lowercase and uppercase, all 10 Latin digits (0-9), plus the minus (-), underscore (_) and dot (.) characters. A TAG value is case sensitive. Its length is between 1 and 32 characters.

The server returns always the TAG names in uppercase, even if the client set them using a lowercase version. The server should convert silently any lowercase character in a TAG name (sent by client) to its corresponding uppercase character.

2.11.2. Reserved Tag Names

The following tag names have a meaning set by this standard for folders:

o HIDDEN - this folder must not be advertised to the user in GUI;

o FIX-TAGS - the tags of this folder cannot be changed with FTGS;

Implicit the folder's items can be read only by its owner.

The following tag names have a meaning set by this standard for messages:

o ANSWERED - a replay was sent for this item;

o FORWARDED - the message was forwarded by the user;

o SEEN - the item was sow by the user;

o SPAM - this email message is marked as spam;

2.12. Folder's Access Rights and Access Control List (ACL)

An access right is represented by a letter between a and z, respectivelly between A and Z. A minus '-' sign means no rights. A lowercase letter represents a different right then its uppercase version.

This document defines following rights:

o a - user can change his and others ACLs with AACL when they already exists, otherwise the user can change only his/hers ACL rights;

- A - user can add/remove ACL accounts with AACL or DACL and change existing ACLs with AACL;

- d - user can delete items from folder with DELE or FDEL, can move items with MOVE and MVFC or can replace an item with RPLC;

- D - user can delete the folder or items from it with DELF, DELE, or FDEL and can move the folder with MOVE, MOVF and MVFC;

- F - user can create folders in this folder with MAKE;

- l - user can list the folder's subfolders with LIST and LSTX;

- L - user can create links to folder's items in SLNK folders;

- r - user can read folder's items with FRTR, RETR and RETC or find them with FIND or FCNT;

- R - user can rename the folder with NAME;

- t - user can (un)tag items with FTAG and STAG;

- T - user can (un)tag the folder with SFTG;

- w - user can add/replace items to the folder with COPY, CPFC, CPYF, FCPY, FMOV, FSTO, MOVE, MOVF, MVFC, RPLC, and STOR functions.

The ACL of a folder is made from pairs of rights and an account pattern. By default the ACL of a folder cannot be changed by user and an user have no rights. The rights for an user are the union of all rights found for matching account patterns. For the state STORING and PRESENCE, the rights are checked only at the access moment.

The account pattern is a regexp as defined in chapter 2.4 Regular Expressions in Commands .

2.13. The Responses for Each Type of Folder

2.13.1. Format and Conventions

All responses are in XML format. The tags and their attributes names are written only in uppercase. The values for attributes only in lowercase. The exception are header items for a message. The tags keep the case from the message.

The content is encoded in UTF-8 [RFC3629] format.

Each type of folder returns its items in a different format.

Each tag written in uppercase must to be send as it is, each tag written in lowercase will be replaced with the right value at the time of generation.

Each tag that have a question mark will be present only once if it is the case and without the question mark.

Each tag that have a star will be present, possible many times, only if it is the case and without the star.

If a command is correct but the server cannot execute it because of an internal error, then the server returns the code 401.

2.13.2. Response for Audio Note Folders

A response holding the content of an audio note has the following structure:

```
<AUDIO-NOTE>
    <SUBJECT>...</SUBJECT>
    <CONTENT type="text/..." encoded="utf-8|base64">...</CONTENT>
    <AUDIO>...</AUDIO>
    <CATEGORY color='rrggbb'>a category</CATEGORY>?
</AUDIO-NOTE>
```

Figure 2

Note: the subject can be any short text. The content can be encoded UTF-8 or BASE64. Implicit is content encoded in utf-8. The type can be any subtype of 'text/*'. Implicit is 'text/plain'. It is recommended to be used only 'text/plain' and 'text/html'. The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color.

Note: the audio represents the bytes of a linear PCM encoding of one channel having a frame rate of 8000 and 8 bits per sample, signed and in little endian order. These bytes are encoded BASE64.

Example:

```
   <AUDIO-NOTE>
       <SUBJECT>Important speaker recorded</SUBJECT>
       <CONTENT type="text/plain" encoded="utf-8">
           Listen the audio.</CONTENT>
       <AUDIO>AAD//...</AUDIO>
   </AUDIO-NOTE>
```

Figure 3

2.13.3. **Response for Bookmark Folders**

A response holding the content of a bookmark has the following structure:

```
   <BOOKMARK>
       <SUBJECT>...</SUBJECT>
       <URL>https?://...</URL>
       <CONTENT type="text/..." encoded="utf-8|base64">...</CONTENT>
       <CATEGORY color='rrggbb'>a category</CATEGORY>?
   </BOOKMARK>
```

Figure 4

Note: the subject can be any short text. The URL must start with http:// or https://. The content can be encoded UTF-8 or BASE64. Implicit is content encoded in utf-8. The type can be any subtype of 'text/*'. Implicit is 'text/plain'. It is recommended to be used only 'text/plain' and 'text/html'. The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color.

Example:

```
<BOOKMARK>
    <SUBJECT>The best search engine</SUBJECT>
    <URL>http://www.google.com/</URL>
    <CONTENT type="text/plain" encoded="utf-8">
        Use it every day.</CONTENT>
</BOOKMARK>
```

Figure 5

2.13.4. Response for Calendar Folders

A calendar entry represents a non repetitive or repetitive action who took a specified amount of time. The format for a timestamp is: yyyy-mm-dd hh:mm:ss. All times are UTC/GMT times according Representation of dates and times [ISO.8601.1988]. The DESCRIPTION must be BASE-64 encoded if there are empty lines present. Implicitly the description is in UTF-8, but if it is BASE-64 encoded then there must be present 'encoded="base64"'. The implicit type of description is text/plain, but it can be also text/html.

The VERSION of a calendar entry is used when there are sent updates. Each new version must have a higher number.

The CONTENT has an implicit type of text/plain.

The list of weekdays are:

- MO - monday
- TU - tuesday
- WE - wednesday
- TH - thursday
- FR - friday
- SA - satuerday
- SU - sunday

The LOCATION is a two fields record, with a TAB character (UTF-8 code 9) delimiting the two fields. It is represented bellow as \t.

The ALARM represents the timestamp in UTC when the alarm must be triggered.

A response holding the content of a calendar entry has the following

structure:

```
<CALENDAR>
    <UID>a_unique_string</UID>
    <VERSION>version_as_a_number</VERSION>
    <AUTHOR>author@domain</AUTHOR>
    <SUBJECT>a subject</SUBJECT>
    <STATUS>DECLINED|NEEDS-ACTION|ACCEPTED|another status</STATUS>
    <FREE_BUSY>FREE|TENTATIVE|BUSY|OUT OF OFFICE|other</FREE_BUSY>
    <LOCATION>location name\tlocation address</LOCATION>
    <LOCATION-MAX-CAPACITY>max-available</LOCATION-MAX-CAPACITY>
    <LOCATION-USED-CAPACITY>occupated</LOCATION-USED-CAPACITY>
    <START>yyyy-mm-dd hh:mm:ss</START>
    <END>yyyy-mm-dd hh:mm:ss</END>
    <CONTENT type="text/..." encoded="utf-8|base64">
             description</CONTENT>
    <ATTENDEE attending="required|optional"
    answer="not-invited|not-answered|rejected|attempting|accepted">
             attendee_as_name_or_email</ATTENDEE>...
    <RESOURCE>resource_as_name_or_email</RESOURCE>...
    <CATEGORY>a category</CATEGORY>...
    <ALARM signal='audio|popup|email' email='email@address,...'>
           yyyy-mm-dd hh:mm:ss</ALARM>
    <JOURNAL>path_to_jrnl_folder</JOURNAL>...
    <REPEAT-RULES>?
        <DAYS>weekday,..</DAYS>?
    </REPEAT-RULES>
    <CATEGORY color='rrggbb'>a category</CATEGORY>?
</CALENDAR>
```

Figure 6

The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color.

Example:

```
   <CALENDAR>
       <UID>CALENDAR-UIDx1234:author@domain</UID>
       <VERSION>1</VERSION>
       <AUTHOR>author@domain</AUTHOR>
       <SUBJECT>To improve the draft for AGAP</SUBJECT>
       <STATUS>ACCEPTED</STATUS>
       <FREE_BUSY>BUSY</FREE_BUSY>
       <LOCATION>Tower Office    Landstrasse, Wien</LOCATION>
       <START>2014-02-15 10:40:00</START>
       <END>2014-03-25 15:59:59</END>
       <CONTENT type="text/plain" encoded="utf-8">
           Improved AGAP</CONTENT>
       <ATTENDEE attending="required" answer="accepted">
           iulian.radu@gmx.at</ATTENDEE>
       <RESOURCE>PC</RESOURCE>
       <CATEGORY>IETF_Drafts</CATEGORY>
       <ALARM type='popup'>-300</ALARM>
       <JOURNAL>/journal/projectX</JOURNAL>
       <REPEAT-RULES>
           <DAYS>MO,TU,WE,TH,FR</DAYS>
       </REPEAT-RULES>
   </CALENDAR>
```

Figure 7

2.13.5. **Response for Configuration Folders**

A response holding the configuration has the following structure:

```
   <CONFIGURATION>
       <name>value</name>...
   </CONFIGURATION>
```

Figure 8

Example:

```
   <CONFIGURATION>
       <CHECK-EACH-MIN>10</CHECK-EACH-MIN>
       <QUOTA>1024</QUOTA>
   </CONFIGURATION>
```

Figure 9

2.13.6. Response for Contact Folders

A response holding the contact information has the following structure:

```
 <CONTACT>
     <SUBJECT>a title</SUBJECT>
     <SALES-LEAD>|Lead|Qualified|Disqualified|
     Opportunity|Customer|Former</SALES-LEAD>?
     <TITLE>Dr.|Prof.|...</TITLE>?
     <NAME>full name</NAME>
     <FOLLOWUP>yyyymmddHHMMSS+tztz</FOLLOWUP>?
     <CATEGORY color='rrggbb'>a category</CATEGORY>?
     <GENDER>m|f|</GENDER>?
     <BIRTHDAY>yyyy-mm-dd</BIRTHDAY>?
     <PHOTO type='image/...'>photo encoded base64</PHOTO>?
     <JOURNAL>/path/to/journal</JOURNAL>?
     <CALENDAR>/path/to/calendar</CALENDAR>?
     <COMPANY>company name</COMPANY>?
     <JOB-TITLE>job title</JOB-TITLE>?
     <WORK-DAYS>working day,...</WORK-DAYS>?
     <WORK-HOURS>hh:mm hh:mm hh:mm hh:mm</WORK-HOURS>?
     <WORK-ADDRESS>company address</WORK-ADDRESS>*
     <WORK-PHONE>company phone number</WORK-PHONE>*
     <FAX>fax</FAX>?
     <WORK-EMAIL>name &lt;email&gt;</WORK-EMAIL>*
     <WORK-WEB name='name'>https?://...</WORK-WEB>*
     <DEPUTY-NAME>assistent name</DEPUTY-NAME>?
     <DEPUTY-PHONE>assisten phone number</DEPUTY-PHONE>?
     <HOME-ADDRESS>address at home</HOME-ADDRESS>*
     <HOME-PHONE>phone number at home</HOME-PHONE>*
     <HOME-EMAIL>name &lt;email&gt;</HOME-EMAIL>*
     <HOME-WEB name='name'>https?://...</HOME-WEB>*
     <MESSAGE type="text/..." encoded="utf-8|base64">a message</MESSAGE>
     <CONTENT type="text/..." encoded="utf-8|base64">a note</CONTENT>
 </CONTACT>
```

Figure 10

GENDER has m for male, f for female and nothing for unknown. BIRTHDAY has the date in UTC. The list of weekdays are:

o MO - monday

o TU - tuesday

- WE - wednesday
- TH - thursday
- FR - friday
- SA - satuerday
- SU - sunday

Example:

```
<CONTACT>
    <SUBJECT>Iulian Radu</SUBJECT>
    <TITLE>DI</TITLE>
    <NAME>Iulian Radu</NAME>
    <GENDER>m</GENDER>
    <PHOTO type='image/jpeg'>ABCDEFGHIJ==</PHOTO>
    <WORK-DAYS>MO,TU,WE,TH,FR</WORK-DAYS>
    <WORK-HOURS>09:00 12:00 13:00 17:00</WORK-HOURS>
    <WORK-EMAIL>Iulian Radu &lt;iulian.radu@gmx.at&gt;</WORK-EMAIL>
    <CONTENT type="text/plain" encoded="utf-8">author</CONTENT>
</CONTACT>
```

Figure 11

2.13.7. Response for File Folders

A response holding the content of a file has the following structure:

```
<FILE>
    <SUBJECT>filename</SUBJECT>
    <TYPE>mime/type</TYPE>
    <SIZE>size</SIZE>
    <AUTHOR>author@domain</AUTHOR>?
    <DESCRIPTION encoded="utf-8|base64">a short text</DESCRIPTION>
    <CONTENT encoded="utf-8|base64">content</CONTENT>
    <CATEGORY color='rrggbb'>a category</CATEGORY>?
</FILE>
```

Figure 12

The valid encodings type are: utf-8 and base64. The default encoding is utf-8. The size is in bytes. The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two

digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color.

Example:

```
<FILE>
    <SUBJECT>Example.txt</SUBJECT>
    <TYPE>text/plain</TYPE>
    <SIZE>6</SIZE>
    <DESCRIPTION>my first example</DESCRIPTION>
    <CONTENT encoded="base64">c3VyZS4=</CONTENT>
</FILE>
```

Figure 13

2.13.8. Response for Filter Folders

An ruleOp can be: AND, OR, NOT, UID, TAG, HAS, IS, WILDCARD, REGEXP or NEW. The value associated to ruleOp is specified as an XML text node. The HAS has one attribute: PATH. The IS, WILDCARD and REGEXP tags have two attributes: PATH and OP. Their values are set as for a filter command (see chapter 4.3 Syntax of a Filter for more information). The tag RULES group all its rules in an AND group.

There must be assigned at least one folder and must be present at least a rule. Optionally can be gived a description using ABOUT tag. Cannot be assigned as folders for being searched folders of the following types: FILT and FOLD.

A response holding the content of a file has the following structure:

```
   <FILTER>
       <ABOUT>...</ABOUT>?
       <FOLDERS>
           <FOLDER>...</FOLDER>...
       </FOLDERS>
       <RULES>
           <ruleOp>...</ruleOp>...
       </RULES>
   </FILTER>
```

Figure 14

Example:

```
   <FILTER>
       <ABOUT>A sample FILT filter.</ABOUT>
       <FOLDERS>
           <FOLDER>/INBOX</FOLDER>
           <FOLDER>/Spam</FOLDER>
       </FOLDERS>
       <RULES>
           <OR>
               <IS path="/MESSAGE/HEADER/subject" op="=">Viagra</IS>
               <AND>
                   <UID>UIDx1234:UIDx4321</UID>
                   <TAG>SPAM</TAG>
               </AND>
           </OR>
       </RULES>
   </FILTER>
```

Figure 15

2.13.9. Response for Location Folders

A response holding the content of a location has the following structure:

```
   <LOCATION>
       <SUBJECT>a subject</SUBJECT>
       <ADDRESS>the full address, inclusive country</ADDRESS>
       <CAPACITY>maximum-capacity|0</CAPACITY>
       <CALENDAR>associated-calendar</CALENDAR>
       <CONTENT encoded='utf-8|base64' type='text/...'>
       a description</CONTENT>
       <CATEGORY color='rrggbb'>a category</CATEGORY>?
       <LAT>lat.itude</LAT>
       <LNG>lon.gitude</LNG>
   </LOCATION>
```

Figure 16

The default encoding for CONTENT is utf-8. The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color. maximum-capacity is 0 if unknown.

Example:

```
   <LOCATION>
       <SUBJECT>Main Office</SUBJECT>
       <ADDRESS>Landstrasse 1, Wien, Austria</ADDRESS>
       <CONTENT encoded='utf-8' type='text/html'>The address of
       the main office.</CONTENT>
       <CATEGORY>Client</CATEGORY>
   </LOCATION>
```

Figure 17

2.13.10. **Response for Internet Radio Folders**

A response holding the content of a radio has the following structure:

```
<RADIO>
    <SUBJECT>...</SUBJECT>
    <URL>https?://...</URL>
    <TYPE>audio/...</TYPE>
    <GENRE>...</GENRE>
    <CONTENT type="text/..." encoded="utf-8|base64">...</CONTENT>
    <CATEGORY color='rrggbb'>a category</CATEGORY>?
</RADIO>
```

Figure 18

Note: the subject can be any short text. The URL must start with http:// or https://. The content can be encoded UTF-8 or BASE64. Implicit is content encoded in utf-8. The type can be any subtype of 'text/*'. Implicit is 'text/plain'. It is recommended to be used only 'text/plain' and 'text/html'. The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color.

Example:

```
<RADIO>
    <SUBJECT>Hits24</SUBJECT>
    <URL>http://streaming208.radionomy.com:80/Hits-24</URL>
    <TYPE>audio/mpeg</TYPE>
    <GENRE>infos pistes pop rock top40</GENRE>
    <CONTENT type="text/plain" encoded="utf-8">
        Use it every day.</CONTENT>
    <CATEGORY color='rrggbb'>a category</CATEGORY>?
</RADIO>
```

Figure 19

2.13.11. Response for Japp Folders

A response holding the content of a Japp has the following structure:

```
<JAPP>
    <SUBJECT>JAPP's name</SUBJECT>
    <ICON type="image/mime-type">icon</ICON>
    <JAR>jar file name</JAR>
    <MAIN-CLASS>main-class</MAIN-CLASS>
    <SIZE>size</SIZE>
    <DESCRIPTION encoded="utf-8|base64">a short text</DESCRIPTION>
    <CONTENT>content of the jar file</CONTENT>
    <CATEGORY color='rrggbb'>a category</CATEGORY>?
</JAPP>
```

Figure 20

The default encoding for DESCRIPTION is utf-8. The ICON and CONTENT are assumed to be encoded base64. The content is a jar file having as main class the class defined in the tag MAIN-CLASS. The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color.

Example:

```
<JAPP>
    <SUBJECT>Sokoban</SUBJECT>
    <ICON type="image/png">AnIcon==</ICON>
    <JAR>Sokoban.jar</JAR>
    <MAIN-CLASS>japp.Sokoban</MAIN-CLASS>
    <SIZE>9</SIZE>
    <DESCRIPTION>a short text</DESCRIPTION>
    <CONTENT>ABCDEFGHIJ==</CONTENT>
</JAPP>
```

Figure 21

The DESCRIPTION tag do not accept an encoded attribute and the text must not have in it empty lines. If the text has empty lines then the server must refuse to accept this XML.

2.13.12. Response for Journal Folders

A journal entry represents a non repetitive action who took no time. The format for a timestamp is: yyyy-mm-dd hh:mm:ss. All times are UTC/GMT times according Representation of dates and times [ISO.8601.1988]. The DESCRIPTION must be BASE-64 encoded if there

are empty lines present. Implicitly the description is in UTF-8, but if it is BASE-64 encoded then there must be present 'encoded="base64"'. The implicit type of description is text/plain, but it can be also text/html. A response holding the content of a journal entry has the following structure:

```
<JOURNAL>
    <TIMESTAMP>yyyy-mm-dd hh:mm:ss</TIMESTAMP>
    <AUTHOR>author@domain</AUTHOR>
    <SUBJECT>a summary</SUBJECT>
    <CONTENT type="text/..." encoded="utf-8|base64">
        a description</CONTENT>
    <ATTENDEE>attendee_as_name_or_email<ATTENDEE>...
    <CATEGORY color='rrggbb'>a category</CATEGORY>?
</JOURNAL>
```

Figure 22

The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color.

Example:

```
<JOURNAL>
    <TIMESTAMP>2011-06-07 13:52:38</TIMESTAMP>
    <AUTHOR>user@example.com</AUTHOR>
    <SUBJECT>The AGAP was updated</SUBJECT>
    <CONTENT type="text/html">A new version of AGAP
        was uploaded to IETF</CONTENT>
    <ATTENDEE>secretary@ietf.com</ATTENDEE>
    <ATTENDEE>Iulian Radu
        &lt;iulian.radu@gmx.at&gt;</ATTENDEE>
</JOURNAL>
```

Figure 23

2.13.13. **Response for Message Folders**

A response holding the content of a message has the following structure:

```
<MESSAGE>
    <HEADER>
        <header-item-once>value</header-item-once>...
        <header-item-multi>value 1
 value 2
 ...
 value n...</header-item-multi>...
    </HEADER>
    <TEXT? encoded="utf-8|base64">main text</TEXT>
    <HTML? encoded="utf-8|base64">main html</HTML>
    <ATTACHMENT-{id}*>
        <HEADER>
            ...
        </HEADER>
        <BODY encoded="utf-8|base64">
            ...
        </BODY>
    </ATTACHMENT-{id}>...
</MESSAGE>
```

Figure 24

The first attachment id has value 1.

The id of on item tag shows the order of the items in the original message.

The default content encoding is utf-8. It is assumed that the content for TEXT and HTML is encoded in UTF-8 when the ENCODED attribut has the value base64.

The items in the header of the main message and attachments are the same with the one from the email message.

There can be at most 2,147,483,647 attachments defined and their numbers must be sequential starting with 1.

Example:

```
<MESSAGE>
    <HEADER>
        <from>example@no-spam.com</from>
        <to>example@example.com</to>
        <received>
            <item>
from mail.yahoo.com by example.com; Tue, 16 Mar 2010 12:14:24 +0100
            </item>
            <item>
from no-spam.com by mail.yahoo.com; Mon, 15 Mar 2010 11:13:23 +0100
            </item>
        </received>
        <content-type>multipart/mixed; boundary="XYZ"</content-type>
        <subject>A basic example</subject>
    </HEADER>
    <TEXT>Please see the attachments.</TEXT>
    <HTML>
&lt;b&gt;Please&lt;/b&gt; see the &lt;u&gt;attachments&lt;/u&gt;.
    </HTML>
    <ATTACHMENT-1/>
      <HEADER>
        <content-type>text/plain</content-type>
      </HEADER>
      <BODY encoded="utf-8">See the picture.</BODY>
    </ATTACHMENT-1>
    <ATTACHMENT-2>
      <HEADER>
        <content-type>image/jpeg</content-type>
        <content-transfer-encoding>base64</content-transfer-encoding>
      </HEADER>
      <BODY encoded="base64">c3VyZS4=</BODY>
    </ATTACHMENT-2>
</MESSAGE>
```

Figure 25

The previous example corresponds to a message with the following structure:

- o multipart/mixed
   - * multipart/alternative
      - + text/plain
      - + text/html

   * text/plain

   * image/jpeg

2.13.14. Response for Note Folders

A response holding the content of the note has the following structure:

```
<NOTE>
    <SUBJECT>a title</SUBJECT>
    <CONTENT type="text/..." encoded="utf-8|base64">a note</CONTENT>
    <FOLLOWUP>yyyymmddHHMMSS+tztz</FOLLOWUP>?
    <CATEGORY color='rrggbb'>a category</CATEGORY>?
</NOTE>
```

Figure 26

Note: the subject can be any short text. The content can be encoded UTF-8 or BASE64. Implicit is content encoded in utf-8. The type can be any subtype of 'text/*'. Implicit is 'text/plain'. It is recommended to be used only 'text/plain' and 'text/html'. The followup represents when this note should be read again (checked). The fields of followup means:

- yyyy - the year
- mm - the year's month as a number (01 to 12)
- dd - the month's day (01 to 31)
- HH - the day's hour (00 to 23)
- MM - the hour's minute (00 to 59)
- SS - the minut's second (00 to 59)
- +tztz - the timezone as four digits (-1200 to +1300)

The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color.

Example:

```
<NOTE>
    <SUBJECT>Important!</SUBJECT>
    <CONTENT type="text/plain" encoded="utf-8">
        To review the code.</CONTENT>
</NOTE>
```

Figure 27

2.13.15. Response for Symbolic Link Folders

This type of folder allows for specifying a link to an item found in an other folder. It cannot indicate an item found in an other SLNK folder.

Example:

```
<SYMBOLIC-LINK>
    <PATH>/Reports</PATH>
    <UID>UIDx0012345</UID>
</SYMBOLIC-LINK>
```

Figure 28

2.13.16. Response for Task Folders

A task entry represents a non repetitive action who took a specified amount of time and which can have subtasks associated. The format for a timestamp is: yyyy-mm-dd hh:mm:ss. All times are UTC/GMT times according Representation of dates and times [ISO.8601.1988]. The DESCRIPTION must be BASE-64 encoded if there are empty lines present. Implicitly the description is in UTF-8, but if it is BASE-64 encoded then there must be present 'encoded="base64"'. The implicit type of description is text/plain, but it can be also text/html.

The VERSION of a task is used when there are sent updates. Each new version must have a higher number.

The CONTENT has an implicit type of text/plain.

The PRIORITY 0 means that there was no priority assigned and the higher the number the higer is the priority. The EFFORT means how difficult it is to implement this. The BENEFIT is how high is the ROI after implementing and using this. The POINTS is a value based on PRIORITY, EFFORT and BENEFIT and it is intendet to offer a mean for ordering the tasks. The higher the number the quicker should be the task implemented. The START-DAYS and DUE-DAYS says in which days of the week the task must start or end. The list is made from two

letter days names delimited with comma. The list of weekdays are:

- MO - monday
- TU - tuesday
- WE - wednesday
- TH - thursday
- FR - friday
- SA - satuerday
- SU - sunday

The GLUE defines the realtion between this task and its subtasks. If there is a delay present then it is applied between the connected ends of the two tasks. If the new timestamp do not meet the requirements of START-DAYS or END-DAYS then the user should be asked for deciding the new dates.

The ALARM represents the timestamp in UTC when the alarm must be triggered.

A response holding the content of a task entry has the following structure:

```
<TASK>
    <UID>a_unique_string</UID>
    <VERSION>version_as_a_number</VERSION>
    <AUTHOR>author@domain</AUTHOR>
    <SUBJECT>a subject</SUBJECT>
    <STATUS>NOT-ASSIGNED|ASSIGNED|ACCEPTED|DECLINED|ON-HOLD|
    WAITING-FOR-INFO|IN-PROGRESS|READY-FOR-REVIEW|IN-REVIEW|
    APPROVED|DONE|another status</STATUS>
    <LOCATION>a location as string</LOCATION>
    <LOCATION-MAX-CAPACITY>capacity</LOCATION-MAX-CAPACITY>
    <START-EARLIEST>yyyy-mm-dd hh:mm:ss</START-EARLIEST>?
    <START-LATEST>yyyy-mm-dd hh:mm:ss</START-LATEST>?
    <START-DAYS>weekday,..</START-DAYS>?
    <START>yyyy-mm-dd hh:mm:ss</START>
    <DUE-EARLIEST>yyyy-mm-dd hh:mm:ss</DUE-EARLIEST>?
    <DUE-LATEST>yyyy-mm-dd hh:mm:ss</DUE-LATEST>?
    <DUE-DAYS>weekday,..</DUE-DAYS>?
    <DUE>yyyy-mm-dd hh:mm:ss</DUE>
    <PRIORITY>priority_as_a_number_btwn_0_and_100</PRIORITY>
    <PERCENT>percent_as_a_number_between_0_and_100</PERCENT>
    <EFFORT>effort_as_a_number_between_0_and_100</EFFORT>
    <BENEFIT>benefit_as_a_number_between_0_and_100</BENEFIT>
    <POINTS>points_as_a_number_between_0_and_10000000</POINTS>
    <CONTENT type="text/..." encoded="utf-8|base64">
             description</CONTENT>
    <ATTENDEE>attendee_as_name_or_email</ATTENDEE>...
    <RESOURCE>resource_as_name_or_email</RESOURCE>...
    <CATEGORY>a category</CATEGORY>...
    <PARENT>TASK_UID_parent_task</PARENT>?
    <SUBTASK>TASK_UID_subtask</SUBTASK>...
    <GLUE link="start-start|start-due|due-start|due-due"
         delay="+-seconds">UID_(sub)task UID_(sub)task</GLUE>...
    <ALARM signal='audio|popup|email' email='email@address,...'>
           yyyy-mm-dd hh:mm:ss</ALARM>
    <JOURNAL>path_to_jrnl_folder</JOURNAL>...
    <CATEGORY color='rrggbb'>a category</CATEGORY>?
</TASK>
```

Figure 29

The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color.

Example:

```
   <TASK>
       <UID>TASK-UIDx1234:author@domain</UID>
       <VERSION>1</VERSION>
       <AUTHOR>author@domain</AUTHOR>
       <SUBJECT>To improve the draft for AGAP</SUBJECT>
       <STATUS>IN-PROGRESS</STATUS>
       <LOCATION>Wien</LOCATION>
       <LOCATION-MAX-CAPACITY>1000</LOCATION-MAX-CAPACITY>
       <START-EARLIEST>2014-01-01 09:00:00</START-EARLIEST>
       <START-LATEST>2014-06-01 09:00:00</START-LATEST>
       <START-DAYS>MO,TU,WE,TH,FR</START-DAYS>
       <START>2014-02-15 10:40:00</START>
       <DUE-LATEST>2014-06-30 17:59:59</DUE-LATEST>
       <DUE>2014-03-25 15:59:59</DUE>
       <PRIORITY>80</PRIORITY>
       <PERCENT>15</PERCENT>
       <EFFORT>50</EFFORT>
       <BENEFIT>100</BENEFIT>
       <POINTS>400000</POINTS>
       <CONTENT type="text/plain" encoded="utf-8">
           Improved AGAP</CONTENT>
       <ATTENDEE>iulian.radu@gmx.at</ATTENDEE>
       <RESOURCE>PC</RESOURCE>
       <CATEGORY>IETF_Drafts</CATEGORY>
       <GLUE type="due-start" delay="86400">UIDNextDraftVersion</GLUE>
       <ALARM type='popup'>-300</ALARM>
       <JOURNAL>/journal/projectX</JOURNAL>
   </TASK>
```

Figure 30

There is defined a virtual property DURATION which returns the difference in seconds between DUE and START.

2.13.17. Response for Timezone Folders

This type of folder allows for specifying a time zone and a title for it.

Example:

```
<TIME-LOCATION>
    <SUBJECT>Vienna, Austria</SUBJECT>
    <TIMEZONE>Europe/Vienna</TIMEZONE>
    <CATEGORY color='rrggbb'>a category</CATEGORY>?
</TIME-LOCATION>
```

Figure 31

The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color.

2.13.18. Response for Versioned Documents Folders

A response holding the content of a versioned document has the following structure:

```
<DOCUMENT>
    <SUBJECT>a title</SUBJECT>
    <VERSION>version</VERSION>?
    <PERCENT>percent_as_a_number_between_0_and_100</PERCENT>
    <AUTHOR>author@domain</AUTHOR>?
    <CREATED>creation_date_time_utc</CREATED>?
    <CONTENT type="text/..." encoded="utf-8|base64">a note</CONTENT>
    <FOLLOWUP>yyyymmddHHMMSS+tztz</FOLLOWUP>?
    <CATEGORY color='rrggbb'>a category</CATEGORY>?
</DOCUMENT>
```

Figure 32

Note: the subject can be any short text. The content can be encoded UTF-8 or BASE64. Implicit is content encoded in utf-8. The type can be any subtype of 'text/*'. Implicit is 'text/plain'. It is recommended to be used only 'text/plain' and 'text/html'. The followup represents when this note should be read again (checked). The fields of followup means:

- o yyyy - the year

- o mm - the year's month as a number (01 to 12)

- o dd - the month's day (01 to 31)

- HH - the day's hour (00 to 23)
- MM - the hour's minute (00 to 59)
- SS - the minut's second (00 to 59)
- +tztz - the timezone as four digits (-1200 to +1300)

The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color. The percent indicate how much from the document is done. The created tag holds the date when the file was created. The format for the date is: YYYY-MM-DD hh:mm:ss. If the created tag is missing, then the server will use its current timestamp at the saving time. If the version is missing and the file is with STOR saved, then the version will be 1. If it is saved with RPLC, then the version will be the next arithmetical version.

Example:

```
<DOCUMENT>
    <SUBJECT>Important!</SUBJECT>
    <VERSION>2</VERSION>
    <PERCENT>90</PERCENT>
    <CREATED>2015-09-30 18:20</CREATED>
    <CONTENT type="text/plain" encoded="utf-8">
        To review the code.</CONTENT>
</DOCUMENT>
```

Figure 33

2.13.19. **Response for Weather Folders**

This type of folder allows for specifying a location and a title for it.

Example:

```
   <WEATHER>
       <SUBJECT>Vienna, Austria</SUBJECT>
       <LOCATION>Vienna</LOCATION>
       <CATEGORY color='rrggbb'>a category</CATEGORY>?
       ...
   </WEATHER>
```

Figure 34

The color of a category is optional and it is defined as a string having only ASCII hex digits, with the first two digits representing the red component, the middle two digits representing the green component, and the last two digits representing the blue component. All components are represented in hexadecimal. The client can use this value (if present) or can ignore it and use its own color.

3. States

3.1. Not-authenticated State

This is the default state when a new connection is made to the server. The client becomes a welcome message.

From this state the client can use the command 'AUTH mechanism' to move in the 'Pre-authentication State'. This is the only other state in which the server can go.

The client can use the command 'STLS' for commuting in the encrypted mode of the channel. After STLS the server remains in the 'Not-authenticated State'. There is no command for switching back to clear-text communication.

The client can use the command 'SGZP' for commuting in the compressed mode of the channel. After SGZP the server remains in the 'Not-authenticated State'. There is no command for switching back to not-compressed communication.

A client can use at the same time the both modes (encrypted and compressed).

The client can use the command 'QUIT' for terminating the connection.

For finding what extensions are installed in server, the client can use the 'CAPA' command.

3.2. Pre-authentication State

This is the state where a client authenticate itself and move to the 'Authenticated State' or returns to the 'Not-authenticated State'.

This standard defines only one method for AUTH: PLAIN. Following is a description of the commands flow used by this authentication mechanism.

The client must send a 'USER account' followed by a 'PASS password' (if the server confirms the acceptance of the account name). If the pair account and password is accepted then the server move to the state 'Authenticated State' and the folder INBOX is selected by server. If this folder does not exist then the server moves in the 'Not-Selected State' and the client must to select an existing folder for operating with this account. If this pair is rejected then the server returns to the 'Not-authenticated State'. That means that the client must to send a new 'AUTH mechanism' for trying a new authentication.

The client can use the command 'QUIT' for terminating the connection.

A client can enter into this state only after a successful 'AUTH' command in 'Not-authenticated State'.

3.3. Authenticated (and Selected) State

This is the state from which a client operates with the content of an account.

From this state the client can use the command 'EXIT' to move in the 'Not-authenticated State'. After an unsuccessful SLCT, the server goes in 'Not-selected State'.

The client can use the command 'QUIT' for terminating the connection.

Check the following chapter for finding which commands can be performed from this state.

A client can enter into this state only after a successful authentication in the 'Pre-authenticated State' or after a successful 'SLCT' command in the 'Authenticated State' or 'Not-selected State'.

3.4. (Authenticated but) Not-selected State

This is the state from which a client must to select a folder for performing further operations.

From this state the client must use the command 'SLCT' to select a folder and to move in the 'Authenticated State'. This is the only other state in which the server can go.

The client CAN use the command 'LIST' for finding valid folder names that eventually CAN be selected with 'SLCT' command.

The client CAN use the command 'QUIT' for terminating the connection.

A client CAN enter into this state only after an unsuccessful 'SLCT' command or if the INBOX folder does not exists and it cannot be selected automatically after a successful authentication.

3.5. Presence State

This is the state in which a client can only ask information about the presence of an user/account.

In this state the client can use only the command 'PGET' to ask for presence information of an account (inclusive finding when a meeting can be scheduled) and the command 'QUIT' for terminating the connection.

3.6. Storing State

This is the state in which a client can only add items (for example: messages, events) in an account which it is not his/her.

In this state the client can use only the command 'FSTO' to find and store the item into a folder of specified type from specified user and the command 'QUIT' for terminating the connection.

4. Commands

4.1. Semantic and Syntax

Each command has its name from 4 letters and it is matched case-insensitive.

Each command is separated by its arguments by a 0x20 character. Also, each argument is separated from its adjacent arguments by a 0x20 character.

The minimal response has only the return code without any text.

A list of elements is enclosed between parentheses (round brackets).

4.2. Syntax of a Tag List

A tag list is used by the following commands: FTAG, GTAG, SFTG and STAG.

A tag list defines what action to be done with its tags.

Syntax: ACTION TAG TAG=VALUE ...

ACTION:

- = - set only these tags;
- + - add these tags
- - - delete these tags.

Note: When it is used the delete tags action and for a tag is set a value then the tag is deleted only if the current value match the value found in the delete command. If in delete command is specified a value for a tag which actually has no value set then this tag is not deleted. If in delete command is specified only the name of a tag without a value and the tag has a value assigned then the tag is deleted.

Example:

```
C: STAG UIDx1234 = SEEN SPAM=YES
C: STAG UIDx1234 + SEEN FLAG=RED OWNER=RAI
C: STAG UIDx1234 - FLAG JUNK OWNER=JOHN SEEN=
```

Figure 35

Note: After this three commands we have only the following tags: SEEN SPAM=YES JUNK OWNER=RAI.

4.3. Syntax of a Filter

4.3.1. Syntax of a Filter for a Command

A filter of this type is used by the following commands: FCPY, FMOV, FDEL, FTAG, FRTR and FIND.

A filter defines rules for matching items. It is defined as lines with rules and it is ended by an empty line.

The keywords of the filter are case insensitive matched (ex.: UID and

Uid are the same).

A rule must be completely defined in the same line (exception are grouping, AND, OR, and NOT rules).

Accepted rules:

- ( ) - grouping for AND and OR;

- AND - all following rules are with AND bonded (until the end of the current group). It is the implicit rule when the first rule is not an AND or an OR;

- OR - all following rules are with OR bonded (until the end of the current group);

- NOT - invert the result of the following rule;

- UID uid - one UID;

- UID uid_begin_range:uid_end_range - inclusive range (uid_end_range is optional and if it missing then it is assumed the maximum valid UID: 32 of lower-case letter z);

- PATH path - the path where is located the item (path is written between ' and ' can be escaped with \');

- TAG tag_name - a tag of an item;

- TAG tag_name=tag_value - an item's tag with a value (tag_value is the complete value);

- HAS field_path - check if exists a field in content (as XML);

- IS field_path op string - a field from the content (as XML) with an exact matched text (string is written between ' and ' can be escaped with \'); op can be: <, <=, =, !=, >=, >;

- WILDCARD field_path op wildcard_string - a field from the content (as XML) with a case-insensitive wildcard expression matched text (widlcard_string is written between ' and ' can be escaped with \'); op can be: =, !=; the widlcard_string can match only a part of the content. In a wildcard_string a '?' matches none or one character and a '*' matches zero or more characters.

- REGEXP field_path op regexp_string - a field from the content (as XML) with a regular expression matched text (regexp_string is written between ' and ' can be escaped with \'); op can be: =, !=;

the regexp_string can match only a part of the content.

- o NEW - it is true if an item is marked as new; after a new item was reported or retrieved (by FIND, FRTR, RETC, RETR or a filter folder) it will be marked as no longer being new and it will not be matched by a new search for new items.

The field_path is a PATH as it is returned by RETR and must point to a not binary end leaf. It contains only tag names separated with /. It is accepted as instead of the first tag to have a * (star) for searching in items of different types but having a common tag (like CATEGORY). Example: /MESSAGE/HEADER/subject, /MESSAGE/HEADER/received, /MESSAGE/TEXT, /MESSAGE/HTML, /MESSAGE/ATTACHMENT-1/HEADER/type, /MESSAGE/ATTACHMENT-1/BODY, /*/CATEGORY. There is an exception, for FILT folder types the path /FILTER/FOLDERS returns the list of folders with a folder path per line and the path /FILTER/FOLDERS/FOLDER is invalid.

Searching for a TAG without associating and a value to it will match all items having this tag, even if it have values set for it.

It can be searched only in the body of attachments that have a content type of type 'text/*'.

Example 1: These filters find all messages with the UID between UIDx0001:UIDx1000 and that were seen and marked as being spam or having a virus (the AND is redundant in the second case). Both filter definitions are equivalent.

```
C: UID UIDx0001:UIDx1000 ( OR TAG SPAM TAG HAS=VIRUS ) TAG SEEN
C: UID UIDx0001:UIDx1000 (AND ( OR TAG SPAM TAG HAS=VIRUS ) TAG SEEN)
```

Figure 36

Example 2:

```
C: AND and1 and2 OR and3or1 and3or2 OR and3or3 and3or4
C: AND and1 and2 OR and3or1 and3or2 AND and3or3and1 and3or3and2
C: OR or1 or2 AND or3and1 or3and2 AND or3and3 or3and4
C: OR or1 or2 AND or3and1 or3and2 OR or3and3or1 or3and3or2
C: AND and1 and2 (OR and3or1 and3or2) and4 and5
C: OR or1 or2 (AND or3and1 or3and2) or4 or5
```

Figure 37

Example 3:

```
C: IS /MESSAGE/HEADER/subject = 'From University'
C: REGEXP /MESSAGE/HEADER/FROM != '[^0-9]+@example\.com$'
```

Figure 38

4.3.2. Syntax of a Filter for a FILT Folder

A filter of this type is used by the following command: STOR.

A filter defines rules for matching the different messages from different folders. It is defined as an XML with target folders and rules.

The keywords of the filter are case sensitive matched (ex.: UID and Uid are not the same). They are always lowercase.

Accepted rules:

- AND - all its items must be matched;
- OR - at least one of its items must be matched;
- NOT - invert the result of its child rule;
- UID uid - one UID;
- UID uid_begin_range:uid_end_range - inclusive range;
- TAG tag_name - a tag;
- TAG tag_name=tag_value - a tag with a value (tag_value is the complete value);
- HAS field_path - check if exists a field in content (as XML);

- IS field_path op string - a field from the content (as XML) with an exact matched text (string is written between ' and ' can be escaped with \'); op can be: <, <=, =, !=, >=, >;

- WILDCARD field_path op wildcard_string - a field from the content (as XML) with a case-insensitive wildcard expression matched text (wildcard_string is written between ' and ' can be escaped with \'); op can be: =, !=; the wildcard_string can match only a part of the content. In a wildcard_string a '?' matches zero or one characters and a '*' matches zero or more characters.

- REGEXP field_path op regex_string - a field from the content (as XML) with a regular expression matched text (regex_string is written between ' and ' can be escaped with \'); op can be: =, !=; the regex_string can match only a part of the content.

The field_path is a PATH as it is returned by RETR and must point to a not binary end leaf. It contains only tag names separated with /. Example: /MESSAGE/HEADER/subject, /MESSAGE/HEADER/received, /MESSAGE/TEXT, /MESSAGE/HTML, /MESSAGE/ATTACHMENT-1/BODY. There is an exception, for FILT folder types the path /FILTER/FOLDERS returns the list of folders with a folder path per line and the path /FILTER/FOLDERS/FOLDER is invalid.

Searching for a TAG without associating and a value to it will match all items that have this tag even if it have values set for it (the empty string is also considered matched).

The following two examples corresponds to the two examples from the previous chapter:

```
<FILTER>
    <FOLDERS>
        <FOLDER>/INBOX</FOLDER>
    </FOLDERS>
    <RULES>
        <AND>
            <UID>UIDx0001:UIDx0010</UID>
            <OR>
                <TAG>SPAM</TAG>
                <TAG>HAS=VIRUS</TAG>
            </OR>
            <TAG>SEEN</TAG>
        </AND>
    </RULES>
</FILTER>
```

Figure 39

Example 2:

```
<FILTER>
  <FOLDERS>
    <FOLDER>/INBOX</FOLDER>
  </FOLDERS>
  <RULES>
    <OR>
      <IS path="/MESSAGE/HEADER/subject" op="=">From University</IS>
      <REGEXP path="/MESSAGE/HEADER/FROM" op="!=">
         [^0-9]+@example\.com$</REGEXP>
    </OR>
  </RULES>
</FILTER>
```

Figure 40

4.4. The Welcome Message - not-authenticated state

Results: 200 401 410 531

Result 200 - the client is accepted for sending commands;

Result 401 - there was an internal error;

Result 410 - too many connections;

Result 531 - the client is rejected permanently.

Description: When a client connects to the server it receives a welcome message. This message begins with a response code that shows if the client is accepted for sending commands.

Examples:

```
S: 200 Welcome localhost [127.0.0.1]
```

Figure 41

```
S: 401 Internal error, please contact our administrator
```

Figure 42

```
S: 410 Sorry, too many connections, please retry later
```

Figure 43

```
S: 531 Your hostname/IP (localhost:127.0.0.1) is blacklisted
```

Figure 44

4.5. Command QUIT - all states

Name: quit

Arguments: none

Result: 200

Description: The QUIT command close the connection between the client and server.

Example:

```
C: QUIT
S: 200 OK Bye
```

Figure 45

4.6. Command AUTH mechanism - not-authenticated state

Name: authenticate

Argument: mechanism

Results: 200 510 511

Result 200 - the mechanism is known and accepted.

Result 510 - unknown command.

Result 511 - the mechanism is unknown/unsupported.

Description: Choose an authentication method. The name of the mechanism can contain only latin letters (A-Z), digits (0-9), the signs minus (-) and underscore (_). It is case insensitive. All supported mechanisms must to be advertised in CAPA's list of capabilities as AUTH-MechanismNameInUpperCase.

The PLAIN Authentication Mechanism: the client send the username and password in clear text using the commands USER and PASS.

The MD5 and SHA1 Authentication Mechanisms: the server send an additional line starting with a dot and providing a prefix that will gone be used by the client to send back to the server an MD5 or SHA-1 value computed on the string build from this prefix and user's password. This prefix can have between 1 and 256 characters. Allowed characters are any UTF-8 characters having a code bigger the decimal value 31 (first valid character is space). The initial dot is not part of the prefix. The client send the username and the computed hash using the commands USER and HASH.

The PRESENCE Authentication Mechanism: this mechanism is used by a client to query the presence of an user having an account on the server. If the server knows to forward the request to other servers (in case that requested account in not local) then it can return the answer from the remote server. The client send the username and password in clear text using the commands USER and PASS. For an anonymous access the server can accept as username anonymous and as password the email address of the connecting user. Once the username and password are accepted, the server enters in the presence state and the client can execute only the commands PGET and QUIT.

The PRESENCE-MD5 and PRESENCE-SHA1 Authentication Mechanisms: these mechanisms are working similar with MD5 and SHA1 authentication mechanisms, only that move the server in the same status as PRESENCE authentication mechanism.

The STORING Authentication Mechanism: this mechanism is used by a client to store items in accounts which are not his/hers. The client send the username and password in clear text using the commands USER and PASS. For an anonymous access the server can accept as username anonymous and as password the email address of the connecting user. Once the username and password are accepted, the server enters in the storing state and the client can execute only the commands FSTO and QUIT.

The STORING-MD5 and STORING-SHA1 Authentication Mechanisms: these mechanisms are working similar with MD5 and SHA1 authentication mechanisms, only that move the server in the same status as STORING authentication mechanism.

Examples:

```
C: AUTH PLAIN
S: 200 OK Send USER
C: USER anonymous
S: 200 OK Send PASS
C: PASS email@example.com
S: 200 OK User anonymous authenticated
```

Figure 46

```
C: AUTH MD5
S: .Hash prefix ... for user's password
S: 200 OK Send USER
C: USER anonymous
S: 200 OK Send HASH
C: HASH 79054025255fb1a26e4bc422aef54eb1
S: 200 OK Authenticated
```

Figure 47

```
   C: AUTH SHA1
   S: .Hash prefix ... for user's password
   S: 200 OK Send USER
   C: USER anonymous
   S: 200 OK Send HASH
   C: HASH de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b4
   S: 200 OK Authenticated
```

Figure 48

```
   C: AUTH PRESENCE
   S: 200 OK Send USER
   C: USER anonymous
   S: 200 OK Send PASS
   C: PASS email@example.com
   S: 200 OK User anonymous authenticated
```

Figure 49

```
   C: AUTH PRESENCE-MD5
   S: .Hash prefix ... for user's password
   S: 200 OK Send USER
   C: USER anonymous
   S: 200 OK Send HASH
   C: HASH 79054025255fb1a26e4bc422aef54eb1
   S: 200 OK User anonymous authenticated
```

Figure 50

```
   C: AUTH PRESENCE-SHA1
   S: .Hash prefix ... for user's password
   S: 200 OK Send USER
   C: USER anonymous
   S: 200 OK Send HASH
   C: HASH de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b4
   S: 200 OK User anonymous authenticated
```

Figure 51

```
   C: AUTH STORING
   S: 200 OK Send USER
   C: USER anonymous
   S: 200 OK Send PASS
   C: PASS email@example.com
   S: 200 OK User anonymous authenticated
```

Figure 52

```
   C: AUTH STORING-MD5
   S: .Hash prefix ... for user's password
   S: 200 OK Send USER
   C: USER anonymous
   S: 200 OK Send HASH
   C: HASH 79054025255fb1a26e4bc422aef54eb1
   S: 200 OK User anonymous authenticated
```

Figure 53

```
   C: AUTH STORING-SHA1
   S: .Hash prefix ... for user's password
   S: 200 OK Send USER
   C: USER anonymous
   S: 200 OK Send HASH
   C: HASH de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b4
   S: 200 OK User anonymous authenticated
```

Figure 54

```
   C: AUTH
   S: 510 UNKNOWN command
```

Figure 55

```
   C: AUTH unknown
   S: 511 UNKNWON method
```

Figure 56

4.7. Command CAPA - not-authenticated state

Name: capabilities

Arguments: none

Result: 200

Description: Ask for the parts of this standards or extensions supported by the server.

Following is a list with all capabilities defined and covered by this document. If the server do no present an item from the following list then the client must assume that the sever is unable to do the associated operations of the missing item.

- ADBK - contact information;
- AUTH-PLAIN - suport plain authentication;
- AUTH-PRESENCE - suport authentication for asking about presence;
- AUTH-PRESENCE-MD5 - suport authentication for asking about presence;
- AUTH-PRESENCE-SHA1 - suport authentication for asking about presence;
- AUTH-MD5 - suport MD5 authentication;
- AUTH-SHA1 - suport SHA1 authentication;
- AUTH-STORING - suport authentication for receiving items;
- AUTH-STORING-MD5 - suport authentication for receiving items;
- AUTH-STORING-SHA1 - suport authentication for receiving items;
- CALE - events;
- CHNG - list the FCID of all folders;
- CONF - user accounts configuration;
- FACL - ACL for folders;
- FILE - storing files;
- FILT - definition of a filter;
- STOR - accepts storing from external sources;
- JAPP - Java applications;

- JRNL - journal items;

- MESG - email messages;

- NOTE - short texts;

- PNFO - presence;

- SPWD spwdFlags - server accepts compression;

- SGZP - server accepts compression;

- STLS - server can encrypt the communication channel;

- TASK - tasks.

spwdFlags indicate the format for the password. After each flag can be specified a number indicating the minimum amount of characters which must correspond to this flag. If a flag is present then it is assumed that at least one character must correspond to it. The order of flags did not impose to keep that order in the new password. The following SPWD flag are defined by this draft:

a - a lowercase latin letter (a-z)

A - a uppercase latin letter (A-Z)

- - a digit (0-9)

. - not a latin letter or digit (not a-z, A-Z, nor 0-9)

Example:

```
C: CAPA
S: .GZIP
S: .TLS
S: .SPWD A1a7-1.1
S: .Extension1
S: .Extension.2 argument1
S: .Extension-3 argument1 argument2
S: 200 OK CAPA completed
```

Figure 57

4.8. Command SGZP - not-authenticated state

Name: start using GZip

Arguments: none

Results: 200 510

Result 200 - the communication is now compressed.

Result 510 - unknown/unsupported command.

Description: Change the communication in compressed mode using GZIP [RFC1952] as compression method. If this command is executed from the compression mode then it simply returns a 200 response code. The response to this command is using still the not-compressed mode of the channel. The compression becomes effective only after a 200 response line was send by the server.

Note: With GZIP the data is compressed using the LZ77 algorithm and Huffman coding. Starting using this mode is like starting to write clear texts into a GZIP format archive and reading texts from a GZIP format archive. The compression is used both by the client and the server and they start to use it with the next line they send after the 200 response line received from the server.

Examples:

```
C: SGZP
S: 200 OK Using GZIP
```

Figure 58

```
C: SGZP
S: 510 UNKNOWN command
```

Figure 59

4.9. Command STLS - not-authenticated state

Name: start using TLS

Arguments: none

Results: 200 510

Result 200 - the communication is now encrypted.

Result 510 - unknown/unsupported command.

Description: Change the communication in mode TLS. If this command is executed from the encrypted mode then it simply returns a 200 response code. The response to this command is using still the not-encrypted mode of the channel. The encryption becomes effective only after a 200 response line was send by the server.

Examples:

```
C: STLS
S: 200 OK Using TLS
```

Figure 60

```
C: STLS
S: 510 UNKNOWN command
```

Figure 61

4.10. Command HASH - pre-authenticated state (MD5 and SHA1)

Name: hash

Argument: hash_code

Result: 200 510 511 512

Result 200 - the pair user/hash was successfully authenticated.

Result 510 - unknown/unsupported command.

Result 511 - invalid hash.

Result 512 - first send USER and then HASH.

Description: Send the hash code associated to the previous authentication method (MD5 or SHA1), previous USER and provided prefix.

Examples:

```
   C: AUTH MD5
   S: .prefix is-here!
   S: 200 OK Send USER
   C: USER account
   S: 200 OK Send HASH
   C: HASH 79054025255fb1a26e4bc422aef54eb1
   S: 200 OK Authenticated
```

Figure 62

```
   C: USER account
   S: 200 OK Send HASH
   C: HASH
   S: 510 UNKNOWN command
```

Figure 63

```
   C: USER account
   S: 200 OK Send HASH
   C: HASH 79054025255fb1a26e4bc422aef54eb1
   S: 511 WRONG user/hash pair
```

Figure 64

```
   C: AUTH SHA1
   S: .prefix is-here!
   S: 200 OK Send USER
   C: HASH de9f2c7fd25e1b3afad3e85a0bd17d9b100db4b3
   S: 512 EXPECTED USER
```

Figure 65

4.11. Command PASS - pre-authenticated state (PLAIN)

Name: password

Argument: password

Result: 200 510 511 512

Result 200 - the pair user/password was successfully authenticated.

Result 510 - unknown/unsupported command.

Result 511 - invalid password.

Result 512 - first send USER and then PASS.

Description: Send the password associated to the previous USER.

Examples:

```
C: USER account
S: 200 OK Send PASS
C: PASS password
S: 200 OK Authenticated
```

Figure 66

```
C: USER account
S: 200 OK Send PASS
C: PASS
S: 510 UNKNOWN command
```

Figure 67

```
C: USER account
S: 200 OK Send PASS
C: PASS password
S: 511 WRONG user/password pair
```

Figure 68

```
C: AUTH PLAIN
S: 200 OK AUTH completed
C: PASS password
S: 512 EXPECTED USER
```

Figure 69

4.12. Command USER - pre-authenticated state (PLAIN, MD5 and SHA1)

Name: user

Argument: account

Result: 200

Result: 200 510 511

Result 200 - the user is accepted and expecting the password.

Result 510 - unknown/unsupported command.

Result 511 - invalid account.

Description: Send an account name for authentication and authorization.

Examples:

```
C: AUTH PLAIN
S: 200 OK Send USER
C: USER account
S: 200 OK Send PASS
```

Figure 70

```
C: AUTH PLAIN
S: 200 OK Send USER
C: USER
S: 510 UNKNOWN command
```

Figure 71

```
C: AUTH PLAIN
S: 200 OK Send USER
C: USER account
S: 511 INVALID username
```

Figure 72

4.13. Command AACL - authenticated state

Name: add a ACL for selected folder

Arguments: rights account

Result: 200 510 511 541

Result 200 - the command was successful.

Result 510 - unknown/unsupported command.

Result 511 - the rights are incorrect or the account is missing.

Result 541 - the user does not have enough rights to change the ACL rights.

Description: Add a new ACL to the current list of ACLs for selected folder or replace the old rights if exists an item for this account.

Examples:

```
C: AACL ADR *@mydomain.com
S: 200 OK The ACL was successfully added
C: AACL R *@mydomain.com
S: 200 OK The ACL was successfully replaced
```

Figure 73

```
C: AACL
S: 510 UNKNOWN command
```

Figure 74

```
C: AACL GR user@domain.com
S: 511 UNKNOWN right G
```

Figure 75

4.14. Command APBL - authenticated state

Name: set currently selected folder as public folder for its type

Arguments: none

Result: 200 410 510

Result 200 - the command was successful.

Result 410 - for the moment the selected folder cannot be added to the list of public folders.

Result 510 - unknown/unsupported command.

Description: Add currently selected folder to the list of public folders.

Note: If there was already set a folder for this type then the previously folder is removed from the list.

Examples:

```
C: APBL
S: 200 OK Folder /MyCalendar was made PUBLIC for CALE
```

Figure 76

```
C: APBL
S: 410 Please retry to add it later
```

Figure 77

```
C: APBL
S: 510 UNKNOWN command
```

Figure 78

4.15. Command CHNG - authenticated and not-selected state

Name: report the FCID (Folder Change ID) for all folders

Arguments: path?

Result: 200 510 511

Result 200 - the command was successful.

Result 510 - unknown/unsupported command.

Result 511 - the path is invalid.

Description: Return a list with the FCID of all folders or of the specified path.

Note: For no path in the list is included and the root folder for all other folders as slash ('/').

Examples:

```
C: CHNG
S: .0BIH /
S: .0009 /Temporary
S: .0001 /Temporary/1980
S: .0BIG /INBOX
S: .0123 /ARCHIVE
S: .0003 /ARCHIVE/2010
S: .0003 /ARCHIVE/2011
S: .00aA /ARCHIVE/2010/OLD
S: 200 OK CHNG completed
C: CHNG /Temporary/1980
S: .0001 /Temporary/1980
S: 200 OK CHNG completed
```

Figure 79

Note: A change in /ARCHIVE/2010 will change the FCID of /ARCHIVE/2010, but not the FCID of /ARCHIVE nor /.

```
C: CHNG
S: 510 UNKNOWN command
```

Figure 80

```
C: CHNG /no/path
S: 511 UNKNOWN path
```

Figure 81

4.16. Command COPY - authenticated state

Name: copy item

Arguments: UID_source path_destination_folder

Result: 200 510 511

Result 200 - the copy was successful.

Result 510 - unknown/unsupported command.

Result 511 - unknown uid, invalid destination folder or path not absolute.

Result 541 - the user does not have enough rights to read items from source or write in destination folder.

Description: Copy an item from currently selected folder into another folder (by UID).

Note: For copying a folder the client must use CPYF or CPFC.

Examples:

```
C: COPY UIDx1234 /ARCHIVE_FOLDER/TODAY
S: 200 OK COPY completed
```

Figure 82

```
C: COPY
S: 510 UNKNOWN command
```

Figure 83

```
C: COPY UIDx1234 ARCHIVE_FOLDER/TODAY
S: 511 INVALID UID
C: COPY MSGx1234 ARCHIVE_FOLDER/1970
S: 511 INVALID Destination
```

Figure 84

4.17. Command CPFC - authenticated state

Name: copy folder content

Argument: path_destination_folder

Result: 200 510 511

Result 200 - the copy was successful.

Result 510 - unknown/unsupported command.

Result 511 - invalid destination folder, destination is not an absolute path or destination does not exists.

Result 541 - the user does not have enough rights to read items from source or write in destination folder.

Description: Copy the non-folder content of a folder into another folder.

Examples (in TODAY are copied only the messages from INBOX):

```
C: SLCT /INBOX
S: 200 Selected /INBOX
C: CPFC /ARCHIVE_FOLDER/TODAY
S: 200 OK CPFC completed (100 items)
```

Figure 85

```
C: CPFC
S: 510 UNKNOWN command
```

Figure 86

```
C: CPFC NoAbsolutePath
S: 511 INVALID Destination
C: CPFC /IDoNotExist
S: 511 Destination folder not found
```

Figure 87

4.18. Command CPYF - authenticated state

Name: copy folder and its content and subfolders

Argument: path_destination_folder/new_folder_name

Result: 200 510 511

Result 200 - the copy was successful.

Result 510 - unknown/unsupported command.

Result 511 - invalid destination folder, destination is not an absolute path or destination exists.

Result 541 - the user does not have enough rights to read items from source or to create the destination folder.

Description: Copy a folder together with its content and subfolders into another new folder.

Examples:

```
   C: SLCT /INBOX
   S: 200 Selected /INBOX
   C: CPYF /ARCHIVE_FOLDER/TODAY
   S: 200 OK CPYF completed (100 items, 5 subfolders)
```

Figure 88

```
   C: CPYF
   S: 510 UNKNOWN command
```

Figure 89

```
   C: CPYF NoAbsolutePath
   S: 511 INVALID destination
   C: CPYF /IAlreadyExist
   S: 511 Destination folder exists
```

Figure 90

4.19. Command DACL - authenticated state

Name: delete an ACL from selected folder

Argument: account

Result: 200 220 510 511

Result 200 - the command was successful and the account was removed from ACL.

Result 220 - the command was successful, but the account was not found in ACL.

Result 510 - unknown/unsupported command.

Result 511 - the account is missing or incorrect.

Result 541 - the user does not have enough rights to change the ACL rights.

Description: Delete an ACL from currently selected folder ACLs.

Examples:

```
C: DACL *@mydomain.com
S: 200 OK The ACL was successfully deleted
```

Figure 91

```
C: DACL user@domain.com
S: 220 Entry not found
```

Figure 92

```
C: DACL
S: 510 UNKNOWN command
```

Figure 93

```
C: DACL @domain.com
S: 511 UNKNOWN item format
```

Figure 94

4.20. Command DELE - authenticated state

Name: delete item

Argument: UID path?

Result: 200 510 511

Result 200 - the item was successfully deleted.

Result 510 - unknown/unsupported command.

Result 511 - unknown uid.

Result 541 - the user does not have enough rights to delete the item.

Description: Delete an item by uid or a value based on path in this uid.

Note: It cannot be undone.

Examples:

```
C: DELE UIDx1234
S: 200 OK Message deleted
```

Figure 95

```
C: DELE
S: 510 UNKNOWN command
```

Figure 96

```
C: DELE 1234
S: 511 INVALID UID
```

Figure 97

4.21. Command DELF - authenticated state

Name: delete folder

Arguments: none

Result: 200 510 511

Result 200 - the folder was successfully deleted.

Result 510 - unknown/unsupported command.

Result 511 - no folder was selected or the user do not have the ACL right to delete from currently selected folder.

Result 541 - the user does not have enough rights to delete the folder.

Description: Delete currently selected folder and all its content and subfolders. If the operation is successful then after it no folder is selected.

Note: It cannot be undone.

Examples:

```
C: DELF
S: 200 OK Folder '/delete/me' was deleted
```

Figure 98

```
C: DELF
S: 510 UNKNOWN command
```

Figure 99

```
C: DELF
S: 511 Please select first a folder
C: DELF
S: 511 /INBOX cannot be deleted
```

Figure 100

4.22. Command DPBL - authenticated state

Name: remove currently selected folder from the list of public folders

Arguments: none

Result: 200 220 410 510

Result 200 - the command was successful and currently selected folder was removed from the list of public folders.

Result 220 - the command was successful, but currently selected folder was not in the list of public folders.

Result 410 - for the moment the selected folder cannot be removed from the list of public folders.

Result 510 - unknown/unsupported command.

Description: Remove currently selected folder from the list of public folders (if it is already there).

Examples:

```
C: DPBL
S: 200 OK Folder /MyCalendar is no longer public
```

Figure 101

```
C: DPBL
S: 220 OK Folder /MyCalendar was not in the list
```

Figure 102

```
C: DPBL
S: 410 Please retry to remove it later
```

Figure 103

```
C: DPBL
S: 510 UNKNOWN command
```

Figure 104

4.23. Command EXIT - authenticated state

Name: exit

Arguments: none

Result: 200

Description: Return the server to the Not-authenticated State.

Example:

```
C: EXIT
S: 200 OK EXIT completed
```

Figure 105

4.24. Command FCNT - authenticated state

Name: find items and returns how many items matched the filter

Argument: filter*

Result: 110 200 220 511

Result 110 - the client can send the filter.

Result 200 - the find was successful.

Result 220 - no item matching the filter was found.

Result 511 - wrong filter.

Result 541 - the user does not have enough rights to read items.

Description: Search for items only from currently selected folder (no subfolders) that correspond to a filter and return the number of matched items. If the search is done for a filter folder then the server does not expect any filter and apply the current filter (if any). If there is no filter in the filter folder then it is returned 0 and the 220 code. If there is no match for the filter then it is returned 0 and the 220 code.

Note: For not FILT folders, the filter is delivered after the acceptance of the command. An empty filter matches all items from that folder.

Examples:

```
C: SLCT /MESG-Folder
C: FCNT
S: 110 OK SEND filter definition (end it with an empty line)
C: UID UIDx0001:UIDx9000 TAG SPAM
C:
S: .3
S: 200 OK FCNT completed (3 matches)
C: SLCT /FILT-Folder
C: FCNT
S: .3
S: 200 OK FCNT completed (3 matches)
```

Figure 106

```
C: FCNT
S: 110 OK SEND filter definition (end it with an empty line)
C: UID UIDx0001:UIDx9000 TAG SPAM
C:
S: .0
S: 220 OK FCNT completed (no matches)
```

Figure 107

```
C: FCNT
S: 110 OK SEND filter definition (end it with an empty line)
C: LATER
C:
S: 511 INVALID filter definition
```

Figure 108

4.25. Command FCPY - authenticated state

Name: find and copy items

Arguments: path_destination_folder filter*

Result: 110 200 210 220 510 511 541

Result 110 - the client can send the filter.

Result 200 - the find and copy was successful for all found UIDs.

Result 210 - the find and copy was successful but not for all found UIDs.

Result 220 - no item matching the filter was found.

Result 510 - unknown/unsupported command.

Result 511 - invalid destination folder or wrong filter.

Result 541 - the user does not have enough rights to read items from source or write in destination folder.

Description: Search for items only in currently selected folder (no subfolders) that correspond to a filter and copy them to a new folder. The tags are also copied. If there is no match for the filter then it is returned a 200 code.

Note: The filter is delivered after the acceptance of the command

(response code 110).

Examples:

```
C: FCPY /ARCHIVE/SPAM
S: 110 OK SEND filter definition (end it with an empty line)
C: UID 00000001:00001000 AND TAG SPAM
C:
S: 200 OK FCPY completed (10 matches)
```

Figure 109

```
C: FCPY /ARCHIVE/SPAM
S: 110 OK SEND filter definition (end it with an empty line)
C: UID 00000001:00001000 AND TAG SPAM
C:
S: 210 OK FCPY completed (8 from 10 were copied - out of space)
```

Figure 110

```
C: FCPY /ARCHIVE/SPAM
S: 110 OK SEND filter definition (end it with an empty line)
C: UID UIDx0001:UIDx9000 TAG SPAM
C:
S: 220 OK FCPY completed (no matches)
```

Figure 111

```
C: FCPY
S: 510 UNKNOWN command
```

Figure 112

```
   C: FCPY MISSING
   S: 511 INVALID folder or path not absolute
   C: FCPY SEND
   S: 110 OK SEND filter definition (end it with an empty line)
   C: LATER
   C:
   S: 511 INVALID filter definition
```

Figure 113

4.26. Command FDEL - authenticated state

Name: find and delete items

Argument: path? filter*

Result: 110 200 210 220 511

Result 110 - the client can send the filter.

Result 200 - the find and delete was successful for all found UIDs.

Result 210 - the find and delete was successful but not for all found UIDs.

Result 220 - no item matching the filter was found or the path was not found in any found item.

Result 511 - wrong filter (inclusive empty filter) or no ACL right to delete.

Result 541 - the user does not have enough rights to delete items.

Description: Search for items only in currently selected folder (no subfolders) that correspond to a filter and delete them (no copy in TRASH) or a value based on path in founded uids. If there is no match for the filter then it is returned a 200 code.

Note: The filter is delivered after the acceptance of the command (response code 110). No filter removes all items.

Examples:

```
   C: FDEL
   S: 110 OK SEND filter definition (end it with an empty line)
   C: UID 00000001:00001000 AND TAG SPAM
   C:
   S: 200 OK FDEL completed (10 matches)
   C: FDEL /MESSAGE/ATTACHMENT-1
   S: 110 OK SEND filter definition (end it with an empty line)
   C: UID 00000001:00001000 AND TAG SPAM
   C:
   S: 200 OK FDEL completed (all 10 attachments deleted)
```

Figure 114

```
   C: FDEL
   S: 110 OK SEND filter definition (end it with an empty line)
   C: UID 00000001:00001000 AND TAG SPAM
   C:
   S: 210 OK FDEL completed (only 8 from 10 matches were deleted)
   C: FDEL /MESSAGE/ATTACHMENT-1
   S: 110 OK SEND filter definition (end it with an empty line)
   C: UID 00000001:00001000 AND TAG SPAM
   C:
   S: 210 OK FDEL completed (only 8 from 10 attachments were deleted)
```

Figure 115

```
   C: FDEL
   S: 110 OK SEND filter definition (end it with an empty line)
   C: UID UIDx0001:UIDx9000 TAG SPAM
   C:
   S: 220 OK FDEL completed (no matches)
   C: FDEL /MESSAGE/ATTACHMENT-1
   S: 110 OK SEND filter definition (end it with an empty line)
   C: UID UIDx0001:UIDx9000 TAG SPAM
   C:
   S: 220 OK FDEL completed (no attachments found in 10 items)
```

Figure 116

```
   C: FDEL
   S: 110 OK SEND filter definition (end it with an empty line)
   C: LATER
   C:
   S: 511 INVALID filter definition
```

Figure 117

4.27. Command FIND - authenticated state

Name: find items

Argument: filter*

Result: 110 200 220 511

Result 110 - the client can send the filter.

Result 200 - the find was successful.

Result 220 - no item matching the filter was found.

Result 511 - wrong filter.

Result 541 - the user does not have enough rights to read items.

Description: Search for items only from currently selected folder (no subfolders) that correspond to a filter and return their UIDs. If the search is done for a filter folder then the server does not expect any filter and apply the current filter (if any). If there is no filter in the filter folder then it is returned only the return code. The answer consists of the UIDs and, for a filter folder, they are followed by a 0x20 character and the absolute path for which are the corresponding UID. If there is no match for the filter then it is returned a 220 code.

Note: For not FILT folders, the filter is delivered after the acceptance of the command. An empty filter matches all items from that folder.

Examples:

```
   C: SLCT /MESG-Folder
   C: FIND
   S: 110 OK SEND filter definition (end it with an empty line)
   C: UID UIDx0001:UIDx9000 TAG SPAM
   C:
   S: .UIDx1234
   S: .UIDx1235
   S: .UIDx2340
   S: 200 OK FIND completed (3 matches)
   C: SLCT /FILT-Folder
   C: FIND
   S: .UIDx1234 /INBOX
   S: .UIDx1234 /Trash
   S: .UIDx1235 /Trash
   S: 200 OK FIND completed (3 matches)
```

Figure 118

```
   C: FIND
   S: 110 OK SEND filter definition (end it with an empty line)
   C: UID UIDx0001:UIDx9000 TAG SPAM
   C:
   S: 220 OK FIND completed (no matches)
```

Figure 119

```
   C: FIND
   S: 110 OK SEND filter definition (end it with an empty line)
   C: LATER
   C:
   S: 511 INVALID filter definition
```

Figure 120

4.28. Command FMOV - authenticated state

Name: find and move

Arguments: path_destination_folder filter*

Result: 110 200 210 220 510 511 541

Result 110 - the client can send the filter.

Result 200 - the find and move was successful for all found UIDs.

Result 210 - the find and move was successful but not for all found UIDs.

Result 220 - no item matching the filter was found.

Result 510 - unknown/unsupported command.

Result 511 - invalid destination folder, wrong filter, or no right to move.

Result 541 - the user does not have enough rights to read items from source or write in destination folder.

Description: Search for items only from currently selected folder (no subfolders) that correspond to a filter and move them to a new folder. The tags are also moved. If there is no match for the filter then it is returned a 200 code.

Note: The filter is delivered after the acceptance of the command (response code 110).

Examples:

```
C: FMOV /ARCHIVE/SPAM
S: 110 OK SEND filter definition (end it with an empty line)
C: UID 00000001:00001000 AND TAG SPAM
C:
S: 200 OK FMOV completed (10 matches)
```

Figure 121

```
C: FMOV /ARCHIVE/SPAM
S: 110 OK SEND filter definition (end it with an empty line)
C: UID 00000001:00001000 AND TAG SPAM
C:
S: 210 OK FMOV completed (8 from 10 moved - out of space)
```

Figure 122

```
C: FMOV /ARCHIVE/SPAM
S: 110 OK SEND filter definition (end it with an empty line)
C: UID UIDx0001:UIDx9000 TAG SPAM
C:
S: 220 OK FMOV completed (no matches)
```

Figure 123

```
C: FMOV
S: 510 UNKNOWN command
```

Figure 124

```
C: FMOV MISSING
S: 511 INVALID folder or not absolute path
C: FMOV /SEND
S: 110 OK SEND filter definition (end it with an empty line)
C: LATER
C:
S: 511 INVALID filter definition
```

Figure 125

4.29. Command FRTR - authenticated state

Name: find items and retrieve fields

Argument: filter* part*

Result: 110 200 220 511

Result 110 - the client can send the filter and the items parts.

Result 200 - the find was successful.

Result 220 - no item matching the filter was found.

Result 511 - wrong filter or items parts.

Result 541 - the user does not have enough rights to read items.

Description: Search for items only from currently selected folder (no subfolders) that correspond to a filter and return their UIDs together with the requested parts from them. If the search is done for a filter folder then the server does not expect any filter and apply the current filter (if any). If there is no filter in the

filter folder then it is returned only the return code. If there are no parts specified then only the UIDs are returned. Each requested part becomes a number starting with 1 and being assigned in the same order as the fields. It is defined a special part named '#' which returns all tags associated to a UID. Each tag is returned on its own line prefixed with the corresponding starting number. It is possible to look only for a tag by adding its name after '#' (like #SEEN) and if it exists then it is returned its name, an equal sign and its value, if it has any value. It is possible to look for a tag having a value by adding its name, equal sign and the searched value after '#' (like #MYTAG=myValue) and if it exists with the given value then it is returned its name, an equal sign and its value.

Note: The number 0 is reserved for the UID. The answer consists of the UIDs and, for a filter folder, they are followed by a 0x20 character and the absolute path for which are the corresponding UID. If the item is marked as new then the UID is prefixed with a multiplication sign (*). The item is then marked as no longer being new. If there is no match for the filter then it is returned a 220 code.

Note: For not FILT folders, the filter is delivered after the acceptance of the command. An empty filter matches all items from that folder.

Examples: The first message has only a value in To, the second has two, and the last one none. The second message has no subject.

```
C: SLCT /MESG-Folder
C: FRTR
S: 110 OK SEND filter&parts definition (end each with an empty line)
C: UID UIDx0001:UIDx9000 TAG SPAM
C:
C: /MESSAGE/HEADER/subject
C: /MESSAGE/HEADER/from
C: /MESSAGE/HEADER/to
C: #
C:
S: .0 UIDx1234
S: .1 Not so important
S: .2 contact@win.com
S: .3 you@example.com
S: .0 *UIDx1235
S: .2 spam@ultimate-spam.com
S: .3 you@example.com
S: .3 your_boss@example.com
S: .0 UIDx2340
```

```
   S: .1 Please respond
   S: .2 office@example.com
   S: .4 SEEN
   S: .4 EXPIRED=NO
   S: 200 OK FRTR completed (3 matches)
   C: FRTR
   S: 110 OK SEND filter&parts definition (end each with an empty line)
   C: UID UIDx0001:UIDx9000 TAG SPAM
   C:
   C:
   S: .0 UIDx1234
   S: .0 UIDx1235
   S: .0 UIDx2340
   S: 200 OK FRTR completed (3 matches)
   C: SLCT /FILT-Folder
   C: FRTR
   S: 110 OK SEND the parts definition (end it with an empty line)
   C: /MESSAGE/HEADER/subject
   C: /MESSAGE/HEADER/from
   C:
   S: .0 UIDx1234 /INBOX
   S: .1 Please respond
   S: .2 office@example.com
   S: .0 UIDx1234 /Trash
   S: .1 Very urgent
   S: .2 spam@ultimate-spam.com
   S: .0 *UIDx1235 /Trash
   S: .1 Not so important
   S: .2 contact@win.com
   S: 200 OK FRTR completed (3 matches)
```

Figure 126

```
   C: FRTR
   S: 110 OK SEND filter&parts definition (end each with an empty line)
   C: UID UIDx0001:UIDx9000 TAG SPAM
   C:
   C: /MESSAGE/HEADER
   C:
   S: 220 OK FRTR completed (no matches)
```

Figure 127

```
   C: FRTR
   S: 110 OK SEND filter&parts definition (end each with an empty line)
   C: LATER
   C:
   S: 511 INVALID filter definition
   C: FRTR
   S: 110 OK SEND filter&parts definition (end each with an empty line)
   C: UID UIDx0001:UIDx9000
   C:
   C: INVALID/PATH
   C:
   S: 511 INVALID part definition
```

Figure 128

4.30. **Command FSTO - storing state**

Name: find and write an item into a folder with a specified type from a certain user

Arguments: FolderType Account

Result: 110 200 410 510 511 541

Result 110 - the requested folder was found and the client can send the item.

Result 200 - the item was successfully stored or there was no content sent by the client.

Result 410 - if the item cannot be stored.

Result 510 - unknown/unsupported command.

Result 511 - invalid folder type, unknown account, the data is not a valid XML or its schema does not correspond to the type of the destination folder.

Result 541 - the user does not have enough rights to write items.

Description: Locate a folder with a specified type in an user account for receiving items from other users and store there the item sent by the client. It behaves like STOR.

Note: Do not send a message content using CDATA as it can hold empty lines and an empty line means for the server the end of the message to be stored.

Examples:

```
C: FSTO MESG kontakt@agap.at
S: 110 Send the message ended with an empty line
C: <MESSAGE><HEADER>...</HEADER><TEXT>...</TEXT></MESSAGE>
C:
S: 200 OK Message stored with UID UIDx1234 into INBOX
```

Figure 129

```
C: FSTO
S: 510 UNKNOWN command
C: FSTO MESG
S: 510 UNKNOWN command
```

Figure 130

```
C: FSTO -1 kontakt@agap.at
S: 511 INVALID folder type
C: FSTO MESG nouser@agap.at
S: 511 UNKNOWN account name
```

Figure 131

```
C: FSTO MESG kontakt@agap.at
S: 541 Not enough rights. Please contact your administrator.
```

Figure 132

4.31. Command FTAG - authenticated state

Name: find and tag items

Arguments: tag_list filter*

Result: 110 200 210 220 510 511

Result 110 - the client can send the filter.

Result 200 - the find and set of tag(s) was successful for all found UIDs.

Result 210 - the find and set of tag(s) was successful but not for all found UIDs.

Result 220 - no item matching the filter was found.

Result 510 - unknown/unsupported command.

Result 511 - invalid tag list, wrong filter, or no right to tag.

Result 541 - the user does not have enough rights to tag items.

Description: Search for items only from currently selected folder (no subfolders) that correspond to a filter and change their tags. If there is no match for the filter then it is returned a 200 code.

Note: The filter is delivered after the acceptance of the command (response code 110).

Examples:

```
C: FTAG + SEEN
S: 110 OK SEND filter definition (end it with an empty line)
C: UID 00000001:00001000 AND NEW
C:
S: 200 OK FTAG completed (10 matches)
```

Figure 133

```
C: FTAG + SEEN
S: 110 OK SEND filter definition (end it with an empty line)
C: UID 00000001:00001000 AND NEW
C:
S: 210 OK FTAG completed (only 8 from 10 matches taged)
```

Figure 134

```
C: FTAG + SEEN
S: 110 OK SEND filter definition (end it with an empty line)
C: UID UIDx0001:UIDx9000 TAG SPAM
C:
S: 220 OK FIND completed (no matches)
```

Figure 135

```
C: FTAG
S: 510 UNKNOWN command
```

Figure 136

```
C: FTAG SEEN
S: 511 INVALID tag list
C: FTAG + SEEN
S: 110 OK SEND filter definition (end it with an empty line)
C: LATER
C:
S: 511 INVALID filter definition
```

Figure 137

4.32. Command GACL - authenticated state

Name: get all ACLs for selected folder

Arguments: none

Result: 200

Result 200 - the command was successful.

Description: It returns a list with all defined ACLs for currently selected folder. If the user do not have the ACL right C then it receives only the entries matching his account.

Examples:

```
C: GACL
S: 200 OK No ACL were defined
C: GACL
S: .CR anonymous
S: .RAD *@mydomain.com
S: .A partner@extdomain.com
S: 200 OK The ACL list for anonymous
C: GACL
S: .A partner@extdomain.com
S: 200 OK The ACL list for partner
```

Figure 138

4.33. Command GFTG - authenticated state

Name: get tags of currently selected folder

Arguments: none

Result: 200 510 511

Result 200 - the tags for UID were successful displayed.

Result 510 - unknown/unsupported command.

Description: Return the tags associated to currently selected folder.

Examples:

```
C: GFTG
S: .SYNC
S: .HIDDEN=NO
S: 200 OK GFTG completed
```

Figure 139

```
C: GFTG
S: 510 UNKNOWN command
```

Figure 140

4.34. Command GPBL - authenticated and not-selected state

Name: get the list of public folders

Arguments: none

Results: 200 510

Result 200 - the list was successful delivered (even if it is empty).

Result 510 - unknown command.

Description: Return the list of all folders declared PUBLIC together with their type.

Note: There should be only one folder for each type.

Examples:

```
   C: GPBL
   S: .MESG /INBOX
   S: .CALE /CALENDAR
   S: .ADBK /Public/CONTACT
   S: .FILE /Project X/Public Files
   S: .JRNL /JOURNAL
   S: .TASK /Project X/Tasks
   S: 200 OK GPBL completed
```

Figure 141

```
   C: GPBL
   S: 510 UNKNOWN command
```

Figure 142

4.35. Command GTAG - authenticated state

Name: get tag of an item

Arguments: UID

Result: 200 510 511

Result 200 - the tags for UID were successful displayed.

Result 510 - unknown/unsupported command.

Result 511 - invalid UID.

Description: Return the tags associated to an item.

Examples:

```
   C: GTAG UIDx1000
   S: .SEEN
   S: .SPAM
   S: 200 OK GTAG completed
   C: GTAG UIDx1001
   S: 200 OK GTAG completed
```

Figure 143

```
C: GTAG
S: 510 UNKNOWN command
```

Figure 144

```
C: GTAG -1
S: 511 INVALID UID
```

Figure 145

4.36. Command LINK - authenticated

Name: add symbolic link to an other folder

Arguments: name, path (on a new line)

Results: 110 200 511 541

Result 110 - the server expect the path.

Result 200 - the link was successful created.

Result 513 - the path does not exists or the name is already used for a subfolder.

Result 541 - the user does not have enough rights to create the link.

Description: Creates in currently slected folder a new folder pointing to an other folder. By deleting this new created folder, the original folder is not created.

Path': It must be an absolute (begins with /) path. The slash sign (/) is used to delimit folders in the hierarchy. The server can return 511 if it founds '.' or '..' in path or '/' in the new folder name.

Examples:

```
C: LINK WORK 2010-JUN
S: 110 Send path
C: /WORK/YEAR-2010/JUN
S: 200 OK LINK completed
```

Figure 146

```
C: LINK WORK 2010-JUN
S: 110 Send path
C: YEAR-2010/JUN
S: 511 ERROR path is not absolute
C: LINK WORK 2010-JUN
S: 110 Send path
C: /WORK/YEAR-2010/JUN
S: 511 ERROR The folder '/WORK/YEAR-2010/JUN' does not exist
C: LINK /WORK/2010-JUN
S: 511 ERROR new name contains invalid characters
```

Figure 147

4.37. Command LIST - authenticated and not-selected state

Name: list folders

Arguments: path/filter?

Results: 200 220 511 541

Result 200 - the list was successful delivered.

Result 220 - the list it is empty.

Result 511 - filter is invalid, the specified path (that has no wildcard) does not exist, or the specified path before last folder name (which has an wildcard) does not exist.

Result 541 - the user does not have enough rights to list the folders. (it is returned only for version without arguments)

Description: List all folders that correspond to the filter (if it is provided), otherwise all direct children of currently selected folder together with their types. All returned folder names are prefixed with the type of the corresponding folder (as it is used by the MAKE command) followed by a white space and the absolute path to the folder.

Filter's path': It is a relative (does not begins with /) or an

absolute (begins with /) path. The slash sign (/) is used to delimit folders in the hierarchy. There can be only a star (*) and must to be located in the name of the last folder. Or two stars which must be the last characters of the filter and means that each folder matching the filter is listed together with all its direct and indirect subfolders. The server can return 511 if it founds '.' or '..' as folder names or '\' in the filter definition.

Examples:

```
C: LIST
S: .MESG YESTERDAY
S: .MESG YEAR-2000
S: 200 OK LIST completed (2 matches)
C: LIST /*
S: .FOLD /
S: .MESG /INBOX
S: .MESG /TRASH
S: .CALE /CALENDAR
S: 200 OK LIST completed (4 matches)
C: LIST YEAR-2010/J*
S: .MESG /WORK/YEAR-2010/JUN
S: .MESG /WORK/YEAR-2010/JUL
S: 200 OK LIST completed (2 matches)
```

Figure 148

```
C: LIST /archive*
S: 220 OK LIST completed (0 matches)
```

Figure 149

```
C: LIST */*
S: 511 ERROR path filter can contain only one * in last folder name
C: LIST /ARCHIVE/2000
S: 511 ERROR The specified folder does not exist
C: LIST /ARCHIVE/2000/Documents *.doc
S: 511 ERROR The folder '/ARCHIVE/2000' does not exist
```

Figure 150

4.38. Command LSTX - authenticated and not-selected state

Name: list folders

Arguments: path/filter?

Results: 200 220 511 541

Result 200 - the list was successful delivered.

Result 220 - the list it is empty.

Result 511 - filter is invalid, the specified path (that has no wildcard) does not exist, or the specified path before last folder name (which has an wildcard) does not exist.

Result 541 - the user does not have enough rights to list the subfolders. (it is only for the version without arguments)

Description: List all folders that correspond to the filter (if it is provided), otherwise all direct children of currently selected folder together with their types. All returned folder names are prefixed with the type of the corresponding folder (as it is used by the MAKE command) followed by a flag indicating some information about that folder, its FCID (Section 2.6) and ECID (Section 2.7), the number of total items, the number of seen items, the effective rights of authorized user for that directory or - if there are no rights assigned to the user, and the absolute path to that folder. For a FILT folder as total is returned 1 if there is a filter defined and 0 if there is no filter defined. Between each arguments are exactly only one white space.

Note: If a folder does not have a ECID, it must to return always the same value, which must be in a valid UID format.

Filter's path': It is a relative (does not begins with /) or an absolute (begins with /) path. The slash sign (/) is used to delimit folders in the hierarchy. There can be only a star (*) and must to be located in the name of the last folder. Or two stars which must be the last characters of the filter and means that each folder matching the filter is listed together with all its direct and indirect subfolders. The server can return 511 if it founds '.' or '..' as folder names or '\' in the filter definition.

The flag is a single char and can be only a digit (0-9) or a latin letter in lower (a-z) or upper case (A-Z) case-sensitive. This document defines the following flags:

- 0 - it have no subfolders and no items; it is a normal folder;
- 1 - it have no subfolders, but it have items; it is a normal folder;
- 2 - it have subfolders, but no items; it is a normal folder;
- 3 - it have subfolders and items; it is a normal folder;
- 4 - it have no subfolders and no items; it is a link to a folder;
- 5 - it have no subfolders, but it have items; it is a link to a folder;
- 6 - it have subfolders, but no items; it is a link to a folder;
- 7 - it have subfolders and items; it is a link to a folder;

Examples:

```
C: LSTX
S: .MESG 1 FCIDx012 ECIDx001 7 2 aAdDFlLrRtTw YESTERDAY
S: .MESG 3 FCIDx123 ECIDx001 2 0 aAdDFlLrRtTw YEAR-2000
S: 200 OK LSTX completed (2 matches)
C: LSTX /*
S: .FOLD 2 FCIDx041 ECIDx000 0 0 aAdDFlLrRtTw /
S: .MESG 2 FCIDx321 ECIDx001 0 3 aAdDFlLrRtTw /INBOX
S: .MESG 0 FCIDx001 ECIDx001 0 0 adDFlLrRtTw /TRASH
S: .CALE 0 FCIDx222 ECIDx001 3 0 aAdDFlLrRtTw /CALENDAR
S: 200 OK LSTX completed (4 matches)
C: LSTX YEAR-2010/J*
S: .MESG 1 FCIDx009 ECIDx001 0 0 - /WORK/YEAR-2010/JUN
S: .MESG 2 FCIDx309 ECIDx001 0 0 FlLrRtT /WORK/YEAR-2010/JUL
S: 200 OK LSTX completed (2 matches)
```

Figure 151

```
C: LSTX /archive*
S: 220 OK LSTX completed (0 matches)
```

Figure 152

```
C: LSTX */*
S: 511 ERROR path filter can contain only one * in last folder name
C: LSTX /ARCHIVE/2000
S: 511 ERROR The specified folder does not exist
C: LSTX /ARCHIVE/2000/Documents *.doc
S: 511 ERROR The folder '/ARCHIVE/2000' does not exist
```

Figure 153

4.39. Command MAKE - authenticated and not-selected state

Name: make folder

Arguments: type path

Result: 200 510 511

Result 200 - the folder was successfully created.

Result 510 - unknown/unsupported command.

Result 511 - invalid path, unknown/unsupported type or the parent.

Result 541 - the user does not have enough rights to create subfolders in path.

Description: Create a folder of a certain type.

Types: They are case insensitive

- ADBK - it holds contacts information;
- ADDR - it holds addresses;
- AUDN - it holds audio notes;
- BKMK - it holds bookmarks for URLs;
- CALE - it holds calendar events;
- CONF - it holds user's settings for roaming.
- FILE - it holds normal folders and files;
- FILT - it holds the results of a filter defined by the user (there can be only one filter per folder);

- FOLD - it contains only subfolders;
- JAPP - it holds applications;
- JRNL - it holds a journal;
- MESG - it holds messages;
- NOTE - it holds user's notes;
- SLNK - it holds links to items found in other folders;
- TASK - it holds tasks;
- TLOC - it holds time zones for different locations;
- WLOC - it holds weather for different locations;

Note: If it requires parents that does not exist then the server will not create them for the client but it will return a 511 response code.

Examples:

```
C: MAKE MESG /ARCHIVE/2010
S: 200 OK Folder created
```

Figure 154

```
C: MAKE
S: 510 UNKNOWN command
```

Figure 155

```
C: MAKE 1234
S: 511 ERROR Missing folder name
C: MAKE new 1234
S: 511 ERROR Unknown folder type
C: MAKE MESG /INBOX/1234
S: 511 ERROR The parent folder does not accept subfolders.
```

Figure 156

4.40. Command MOVE - authenticated state

Name: move item

Arguments: UID_source path_destination_folder

Result: 200 510 511

Result 200 - the move was successful.

Result 510 - unknown/unsupported command.

Result 511 - unknown uid, invalid destination folder or no ACL right to move items.

Result 541 - the user does not have enough rights to read or delete items from source or write in destination folder.

Description: Move an item into another folder (by UID).

Note: For moving a folder the client must use MOVF or MVFC.

Examples:

```
C: MOVE UIDx1234 ARCHIVE_FOLDER/TODAY
S: 200 OK MOVE completed
```

Figure 157

```
C: MOVE
S: 510 UNKNOWN command
```

Figure 158

```
C: MOVE UIDx1234 ARCHIVE_FOLDER/TODAY
S: 511 INVALID UID
C: MOVE MSGx1234 ARCHIVE_FOLDER/1970
S: 511 INVALID Destination
```

Figure 159

4.41. Command MOVF - authenticated state

Name: move folder and its content and subfolders

Argument: path_destination_folder/new_folder_name

Result: 200 510 511 541

Result 200 - the move was successful.

Result 510 - unknown/unsupported command.

Result 511 - invalid destination folder, destination is not an absolute path or destination exists.

Result 541 - the user does not have enough rights to read or delete items from source or write in destination folder.

Description: Move folder together with its content and subfolders into another new folder.

Examples:

```
C: SLCT /INBOX
S: 200 Selected /INBOX
C: MOVF /ARCHIVE_FOLDER/TODAY
S: 200 OK MOVF completed (100 items, 5 subfolders)
```

Figure 160

```
C: MOVF
S: 510 UNKNOWN command
```

Figure 161

```
C: MOVF NotAnAbsolutePath
S: 511 INVALID destination
C: MOVF /IAlreadyExist
S: 511 Destination folder exists
```

Figure 162

4.42. Command MVFC - authenticated state

Name: move folder content

Argument: path_destination_folder

Result: 200 510 511 541

Result 200 - the move was successful.

Result 510 - unknown/unsupported command.

Result 511 - invalid destination folder, destination is not an absolute path or destination does not exists.

Result 541 - the user does not have enough rights to read or delete items from source or write in destination folder.

Description: Move only the non-folder content of a folder into another folder.

Examples:

```
C: SLCT /INBOX
S: 200 Selected /INBOX
C: MVFC /ARCHIVE_FOLDER/TODAY
S: 200 OK MVFC completed (100 items)
```

Figure 163

```
C: MVFC
S: 510 UNKNOWN command
```

Figure 164

```
C: MVFC NotAnAbsolutePath
S: 511 INVALID destination
C: MVFC /IDoNotExist
S: 511 Destination folder not found
```

Figure 165

4.43. Command NAME - authenticated state

Name: rename folder

Arguments: new_name

Results: 200 510 511

Result 200 - the rename was successful.

Result 510 - unknown/unsupported command.

Result 511 - invalid new_name or trying to rename a folder name without having the ACL right to rename the folder.

Result 541 - the user does not have enough rights to rename it.

Description: Rename a folder. The currently selected folder remains selected even if the name was changed.

Note: The new_name does not hold any path hierarchy.

Examples:

```
C: SLCT /ARCHIVE/2001
S: 200 OK
C: NAME OLD-2001
S: 200 OK NAME completed
```

Figure 166

```
C: NAME
S: 510 UNKNOWN command
```

Figure 167

```
   C: SLCT /INBOX
   S: 200 OK
   C: NAME InBox
   S: 511 ERROR The folder cannot be renamed (reserved name)
   C: NAME /A/new-folder
   S: 511 ERROR The argument must not be a path
```

Figure 168

4.44. Command NOOP - authenticated state

Name: noop

Arguments: none

Result: 200

Description: It does nothing (eventually announce what changes was done in current folder).

Example:

```
   C: NOOP
   S: 200 OK NOOP completed
```

Figure 169

4.45. Command PGET - authenticated, not-selected and presence state

Name: fetch presence information

Arguments: USER user|BUSY user|FREE user|UID uid user

Result: 110 200 510 511

Result 110 - the client can send the list of timestamps and locations.

Result 200 - the presence information for the user were found and returned or the uid was found and returned.

Result 510 - unknown/unsupported command or not found uid.

Result 511 - unknown/unsupported/missing arguments.

Description: Fetch information about an user current availability, checks when an user is busy or free and returns the event items.

Note: If there is present an argument USER then the user wants to obtain information about the status of an other user or himself. If the server does not know how to obtain the information about this user then it returns an UNKNOWN as argument. Otherways can return a list with all set texts. After an HERE and AWAY is present when was this set, for an IDLE is the timestamp corresponding to the starting point of idle period. In answer can be present only one of these three. For a PSET FOR is returned an HERE for a PGET USER.

Note: If there is present an argument BUSY or FREE then it is expected a list with timestamp periods and locations for which to be returned the list of busy, respectively free time frames. The end of list is marked by sending an empty line to server. Each list line has three fields: a start and inclusive end timestamp and a location. Between each argument is exactly only one space character (0x20). A timestamp has the format: YYYY-MM-DD hh:mm:ss and represents the time in UTC. The meaning of the timestamp fields could be found in Date and Time on the Internet: Timestamps [RFC3339]. A location is a string or the star character (*) for matching any location. The command returns a list with all available busy or free time frames for specified period and location. If there cannot be found any busy or free time frame then no list is returned, but only a 200 return code. The location name is compared which what was stored by the user, so there someone can write Vienna and someone else Wien, so the two of them cannot be found with a list having only one line and specifing a location. The answer is made of pairs of start and inclusive end timestamps followed by the UID correspondig to the event associated with this time period. In case there is no time period specified a 511 error is returned by server.

Note: If there is present an argument UID then it is returned its associated content only if it is of type VEVENT.

Examples:

```
   C: PGET USER user@example.com
   S: .IDLE 2011-05-27 18:00:00 +1000
   S: .STATUS Today I am doing HomeOffice
   S: .AT Headquarter, 1010 Vienna, Austria
   S: 200 OK USER found
   C: PGET USER user2@example.com
   S: .HERE 2011-05-27 18:00:00 +1000
   S: .STATUS Today I am in Austria
   S: .AT Headquarter, 1010 Vienna, Austria
   S: 200 OK USER found
   C: PGET USER user3@example.com
   S: .AWAY 2011-05-27 18:00:00 +1000
   S: 200 OK USER found
   C: PGET USER user@domain.com
   S: .UNKNOWN
   S: 200 OK USER not found
   C: PGET USER user@domain.com
   S: .UNKNOWN
   S: 200 OK USER not found

   C: PGET BUSY user@domain.com
   S: 110 Send the time periods and location ended with an empty line
   C: 2012-01-01 09:00:00 2012-01-03 17:30:00 Vienna/AT
   C: 2012-01-01 09:00:00 2012-01-03 17:30:00 Wien
   C:
   S: .2012-01-01 10:00:00 2012-01-02 23:59:59 UIDx1234
   S: .2012-01-03 14:00:00 2012-01-03 16:30:00 UIDx1289
   S: 200 OK 2 BUSY time periods found
   C: PGET FREE user@domain.com
   S: 110 Send the time periods and location ended with an empty line
   C: 2012-01-01 09:00:00 2012-01-03 17:30:00 Vienna/AT
   C: 2012-01-01 09:00:00 2012-01-03 17:30:00 Wien
   C:
   S: .2012-01-01 09:00:00 2012-01-01 09:59:59 UIDx234
   S: .2012-01-03 00:00:00 2012-01-03 13:59:59 UIDx345
   S: .2012-01-03 16:30:01 2012-01-03 17:30:00 UIDx456
   S: 200 OK 3 FREE time periods found

   C: PGET UID UIDx1234 user@domain.com
   S: .<VEVENT>
   S: .<UID>20110531T114600Z-123456@agap.at</UID>
   S: .<DTSTAMP>2011-05-31T12:10:00Z</DTSTAMP>
   S: .<DTSTART>2011-06-07T18:00:00Z</DTSTART>
   S: .<DTEND>2011-06-07T24:00:00Z</DTEND>
   S: .<SUMMARY>AGAP RFC Party</SUMMARY>
   S: .<DESCRIPTION>Celebration of a new revision!
   S: .0.4</DESCRIPTION>
   S: .</VEVENT>
```

Figure 170

```
   C: PGET USER user@domain.com
   S: 510 Presence is not supported
   C: PGET UID uid user
   S: 510 Uid not found
```

Figure 171

```
   C: PGET
   S: 511 Missing argument
   C: PGET WRONG
   S: 511 UNKNOWN argument WRONG
   C: PGET USER
   S: 511 Missing argument
   C: PGET BUSY user
   C:
   S: 511 There must be defined at least one time period and location
   C: PGET BUSY user
   C: yyyy-01-01 09:00:00 2012-01-03 17:30:00 *
   C:
   S: 511 Invalid time period definition
   C: PGET BUSY user
   C: yyyy-01-01 09:00:00 2012-01-03 17:30:00
   C:
   S: 511 Missing location
   C: PGET UID user
   S: 511 Missing uid
```

Figure 172

4.46. Command PSET - authenticated and not-selected state

Name: announce presence

Arguments: HERE|AWAY|FOR number unit_of_time|IDLE number unit_of_time|STATUS text|AT text

Result: 200 510 511

Result 200 - the presence of the user was updated or requested information returned.

Result 510 - unknown/unsupported command.

Result 511 - unknown/unsupported/missing arguments.

Description: Announce that the logged user is still online, eventually since when is idle. A QUIT command or disconnection means that the user is no longer online. In state PRESENCE only the "USER user" arguments are accepted.

If there is present an argument HERE then that means the user is no longer idle. As there is no information for how long this status is valid, then it will remain valid until comes a new command which change it.

If there is present an argument AWAY then that means the user is no longer connected. As there is no information for how long this status is valid, then it will remain valid until comes a new command which change it.

If there is present an argument FOR then that means the user is suppose to be considered present for the given amount of time. The client is expected to send its updated status in this period. If comes no command to change the state in the specified time, then after this period the presence will be reported as unknown. The arguments are as by IDLE.

If there is present an argument IDLE then that means the user is idle and the arguments must be: IDLE number unit_of_time. The number must be a positive number and the unit_of_time must be one of the following: sec, min, hour, day, year. As there is no information for how long this status is valid, then it will remain valid until comes a new command which change it.

If there is present an argument STATUS then the user want to change the text that is associated with its presence online. After STATUS can follow any text. The text ends to the end of line. If there is no text then any previous text is deleted and no text is displayed as status text.

If there is present an argument AT then that user want to change the text that is associated with its present location. After AT can follow any text. The text ends to the end of line. If there is no text then any previous text is deleted and no text is displayed as location text.

Examples:

```
C: PSET HERE
S: 200 OK You are online and not idle
C: PSET AWAY
S: 200 OK You are no more online
C: PSET FOR 5 min
S: 200 OK You are now online. Expecting an update in 5 minutes.
C: PSET IDLE 5 min
S: 200 OK You are idle since 5 minutes
C: PSET STATUS Today I am doing HomeOffice
S: 200 OK You changed your status text
C: PSET AT Headquarter, 1010 Vienna, Austria
S: 200 OK You changed your location text
```

Figure 173

```
C: PSET HERE
S: 510 Unknown command
```

Figure 174

```
C: PSET
S: 511 Missing argument
C: PSET WRONG
S: 511 UNKNOWN argument WRONG
C: PSET AT
S: 511 Missing argument
```

Figure 175

4.47. Command PSHA - authenticated and not-selected state

Name: set a listening point of a client for push announcements

Arguments: ip port sec filter?

Result: 200 410 510 511

Result 200 - the listening point was registered.

Result 410 - listening point is unreachable or ip is different from current connection.

Result 510 - unknown/unsupported command.

Result 511 - unknown/unsupported/missing arguments.

Description: Register an IP and port on client for announcing the client when there are new items received. If there was already a record for this listening point then the old information are replaced with the new one. The server will not send notification for AGAP folder of the following types: FILT, FOLD. When it is noticed a change in a monitored folder then the server will send a line to registered listening point holding the new FCID of that folder and the path of changed folder.

The ip and port are on client. The server can check if the client is accessible on this port and if the ip is the same which the one from which comes the current connection. If these checks fails then it returns an 410.

The sec indicate how long should the server send notification on this listening point. It must be a positive value, otherwise the server answers with 511.

The filter is a set of AGAP folder types delimited with a white space or comma. The server will gone send notifications only for changes in folders of this type. Even if there are no notification send for FILT or FOLD they are accepted in the filter. If the filter is missing then there are sent notification for all folder types except FILT and FOLD.

Examples:

```
C: PSHA 192.168.100.4 12345 900 MESG,CALE
S: 200 OK We notify you for 15 min
```

Figure 176

The notification send from the server to the client on registered listening point for changes in two AGAP folders:

```
S: FCIDx1234 /INBOX/new messages folder
S: FCIDx2345 /SPAM
```

Figure 177

```
   C: PSHA 192.168.100.4 12345 900
   S: 410 The ip is not same as current connection
```

Figure 178

```
   C: PSHA 192.168.100.4 12345 900
   S: 510 Unknown command
```

Figure 179

```
   C: PSHA 192.168.100.4 12345
   S: 511 Missing timespan
   C: PSHA 192.168.100.4 12345 0
   S: 511 The notification timespan is invalid
   C: PSHA 192.168.100.4
   S: 511 Missing arguments
```

Figure 180

4.48. Command PSHD - authenticated and not-selected state

Name: set a listening point of a client for push announcements

Arguments: ip port

Result: 200 510 511

Result 200 - the listening point was unregistered.

Result 510 - unknown/unsupported command.

Result 511 - unknown/unsupported/missing arguments.

Description: Unregister an IP and port on client for announcing the client when there are new items received.

Examples:

```
   C: PSHD 192.168.100.4 12345
   S: 200 OK We removed it
```

Figure 181

```
   C: PSHD 192.168.100.4 12345 900
   S: 510 Unknown command
```

Figure 182

```
   C: PSHD 192.168.100.4 0
   S: 511 Invalid port
   C: PSHD 192.168.100.4 12345 900
   S: 511 Too many arguments
```

Figure 183

4.49. Command RETC - authenticated state (MESG and FILE folder types)

Name: retrieve

Argument: UID

Results: 200 510 511

Result 200 - the message was found.

Result 510 - unknown/unsupported command.

Result 511 - invalid UID or folder is not of type MESG or FILE.

Result 541 - the user does not have enough rights to read the item.

Description: Fetch from server information about the message, a file or a japp item with the given UID. Each line of answer is prefixed with a dot that it is not part of the returned object. RETC returns for an attachment its name and size in bytes instead of its content as RETR. RETC do not returns for a file or japp its content as RETR did.

Examples:

```
   C: RETC UIDx1234
   S: .<MESSAGE><HEADER>...</HEADER>
   S: .<ATTACHMENT-1>
   S: .<NAME>cid:A12</NAME>
   S: .<SIZE>123456</SIZE>
   S: .</ATTACHMENT-1>
   S: .</MESSAGE>
   S: 200 OK RETC completed
```

Figure 184

```
   C: RETC UIDx1234
   S: .<FILE>
   S: .<NAME>file-test.txt</NAME>
   S: .<SIZE>123456</SIZE>
   S: .</MESSAGE>
   S: 200 OK RETC completed
```

Figure 185

```
   C: RETC
   S: 510 UNKNOWN command
```

Figure 186

```
   C: RETC WrongUID
   S: 511 INVALID UID
   C: RETC UIDx1234
   S: 511 RETC is allowed only for MESG or FILE folders
```

Figure 187

4.50. Command RETR - authenticated state

Name: retrieve

Arguments for a FILT folder: none

Arguments for other types: UID part?

Results: 200 510 511

Result 200 - the item was found or filter content was delivered.

Result 510 - unknown/unsupported command.

Result 511 - invalid UID or part name.

Result 541 - the user does not have enough rights to read the item.

Description: Fetch from server the item with the given UID. For a filter folder, it must be called without an UID and it returns the content of the filter. Each line of answer is prefixed with a dot that it is not part of the returned object.

Part: It is a PATH as it is returned by RETR or RETC and must point to a not binary end leaf. It contains only tag names separated with /. Example: /MESSAGE/HEADER/subject, /MESSAGE/HEADER/received, /MESSAGE/HTML, /MESSAGE/ATTACHMENT-1/BODY. For an item in the header with a multivalue are returned each value on its own line.

Examples:

```
C: RETR UIDx1234
S: .<MESSAGE><HEADER>...</HEADER><TEXT>...</TEXT></MESSAGE>
S: 200 OK RETR completed
C: RETR UIDx1234 /MESSAGE/HEADER/subject
S: .Message's subject
S: 200 OK RETR completed
C: RETR UIDx1234 /MESSAGE/HEADER/received
S: .from s0001.srv.example.com [10.11.12.13] by mx.example.com
S: . (Postfix) with ESMTP id 01234567890 for <user@example.com>;
S: . Thu, 19 Nov 2009 01\:02\:03 +0100 (CET)
S: . by userpc (192.168.192.168) id 20091119010204A;
S: . Thu, 19 Nov 2009 01\:02\:04 +0100 (CET)
S: 200 OK RETR completed
C: RETR
S: .<FILTER>
S: .<FOLDERS><FOLDER>/Spam</FOLDER></FOLDERS>
S: .<RULES></RULES>
S: .</FILTER>
S: 200 OK RETR completed
```

Figure 188

```
C: RETR
S: 510 UNKNOWN command (only FILT folders do not needs arguments)
```

Figure 189

```
C: RETR WrongUID
S: 511 INVALID UID
C: RETR UIDx1234 ABC
S: 511 UNKNOWN part name
C: RETR UIDx1234
S: 511 RETR with UID is not allowed for a FILT folder
```

Figure 190

4.51. Command RPLC - authenticated state

Name: replace an item

Arguments: UID (utf-8|base64|del path)?

Result: 110 200 410 511

Result 110 - the client can send the item.

Result 200 - the item was successfully stored, the old uid (if present) was removed or there was no content sent by the client.

Result 210 - the new item was stored but the old one was not deleted.

Result 410 - if the item cannot be stored.

Result 511 - if the data is not a valid XML or its schema does not correspond to the type of the destination folder.

Result 541 - the user does not have enough rights to write or delete items.

Description: Store a new item/filter into a folder and remove the item with provided UID if it is present. If it is written a new filter into a FILT folder, then the previous filter is deleted. If the new filter has an invalid XML structure or cannot be saved then the folder remains with the old filter (if any). The server can send a 410 or 511 respons before the empty line is send, so the client must check after each sended line of content if the server had rejected the content. If the old item cannot be removed the new item remais saved and a 210 return code instead of a 200 return code is produced.

If there is present a path then the value of the corresponding path is changed (for utf-8 and base64) or removed (for del). The utf-8 or base64 announce how is the new content delivered from client and an empty line marks the end of the new content. If this type of replace is supported than the server returns a 110 before waiting for the new content. This command is the atomic equivalent of STOR followed by DELE UID. The del do not returns a 110 but 200 if the path was found and successfully removed.

Note: Do not send a message content using CDATA as it can hold empty lines and an empty line means for the server the end of the message to be stored.

Examples:

```
C: RPLC UIDx1212
S: 110 Send the message ended with an empty line
C: <MESSAGE><HEADER>...</HEADER><TEXT>...</TEXT></MESSAGE>
C:
S: 200 OK Message stored (UID is UIDx1234), UIDx1212 removed
C: RPLC UIDx1212
S: 110 Send the message ended with an empty line
C:
S: 200 OK Message not stored and UIDx1212 not removed
C: RPLC UIDx1212 utf-8 /MESSAGE/TEXT
S: 110 Send the message ended with an empty line
C: New text
C:
S: 200 OK Message stored as UIDx1212, TEXT replaced
C: RPLC UIDx1212 del /MESSAGE/ATTACHMENT-1
S: 200 OK Attachment-1 was removed
```

Figure 191

```
C: RPLC UIDx1212
S: 110 Send the message ended with an empty line
C: <MESSAGE><HEADER>...</HEADER><TEXT>...</TEXT></MESSAGE>
C:
S: 210 OK Message stored as UIDx1234, UIDx1212 not removed
```

Figure 192

```
C: RPLC UIDx1212
S: 110 Send the message ended with an empty line
C: <MESSAGE><HEADER>...</HEADER><HTML>...</HTML></MESSAGE>
C:
S: 410 Not enough space, UIDx1212 not removed
```

Figure 193

```
C: RPLC UIDx1212
S: 110 Send the message ended with an empty line
C: msg
C:
S: 511 Cannot store it, the message has an incorrect format
C: RPLC
S: 511 Missing UID
```

Figure 194

4.52. Command SFTG - authenticated state

Name: set the tags of currently selected folder

Arguments: tag_list

Result: 200 210 410 510 511 541

Result 200 - all tags for current folder were successful set.

Result 210 - not all tags for current folder were successful set.

Result 410 - for the moment the flags cannot be saved.

Result 510 - unknown/unsupported command.

Result 511 - invalid tag list.

Result 541 - the tag FIX-TAGS does not allow to change the already set tags or the user does not have enough rights to tag the folder.

Description: Set or delete tags of currently selected folder. The FCID of the folder is increased to mark the change.

Note: Setting the tag FIX-TAGS makes the tags of currently selected folder unchangable after this command. The tags can only be changed on the server.

Note: A return code 210 is returned even when no flag could be set if

the tag list is correct.

Examples:

```
C: SFTG + SYNC
S: 200 OK SFTG completed
```

Figure 195

```
C: SFTG - SYNC TEST
S: 210 OK SFTG did not removed TEST
C: SFTG - TEST
S: 210 OK SFTG did not removed TEST
```

Figure 196

```
C: SFTG + SYNC
S: 410 Please retry to set them later
```

Figure 197

```
C: SFTG
S: 510 UNKNOWN command
C: SFTG + SYNC
S: 510 Please select a folder first
```

Figure 198

```
C: SFTG SYNC
S: 511 INVALID tag list
```

Figure 199

```
C: SFTG + SYNC
S: 541 Tag FIX-TAGS prevents the change of tags
```

Figure 200

4.53. Command SLCT - authenticated and not-selected state

Name: select a folder

Argument: path

Result: 200 510 511

Result 200 - the folder was successfully selected.

Result 510 - unknown/unsupported command.

Result 511 - unknown path or '/'.

Description: Select a folder. If the selection was not successful then no folder remains selected and the server switch in the 'Not-selected State'.

Examples:

```
C: SLCT /INBOX
S: 200 OK Folder selected
C: SLCT ARCHIVE/2000
S: 200 OK Folder selected
```

Figure 201

```
C: SLCT
S: 510 UNKNOWN command
```

Figure 202

```
C: SLCT 1234
S: 511 INVALID folder
C: SLCT /
S: 511 You cannot select /
```

Figure 203

4.54. Command SPWD - authenticated and not-selected state

Name: change password

Argument: old-password new-password

Result: 110 200 511 541

Result 110 - the client can send the passwords.

Result 200 - the password was successfully changed.

Result 511 - the old password was wrong or the new password did not respect the rules imposed for new passwords.

Result 541 - the user cannot change the password.

Description: If the old-password corresponds with current password then the new-password will be replace the old one.

Note: Th return code 541 is returned and if the server cannot change the password because it can only read or validate it.

Examples:

```
C: SPWD
S: 110 OK SEND old and new passwords
C: old-password
C: new-password
S: 200 OK SPWD changed the password
```

Figure 204

```
C: SPWD
S: 110 OK SEND old and new passwords
C: old-password
C: new-password
S: 511 Old password don't match
```

Figure 205

```
C: SPWD
S: 541 Server cannot change password on your behalf
C: SPWD
S: 110 OK SEND old and new passwords
C: old-password
C: new-password
S: 541 Server cannot change password on your behalf
```

Figure 206

4.55. Command STAG - authenticated state

Name: set tags of items

Arguments: UID tag_list

Result: 200 410 510 511

Result 200 - the tags for UID were successful set.

Result 410 - for the moment the flags cannot be saved.

Result 510 - unknown/unsupported command.

Result 511 - invalid UID.

Result 541 - the user does not have enough rights to tag the item.

Description: Set or delete tags associated to an item.

Examples:

```
C: STAG UIDx1000 + SEEN SYNC DAY=2012
S: 200 OK STAG completed
```

Figure 207

```
C: STAG UIDx1000 + SEEN SYNC DAY=2012
S: 410 Please retry to set them later
```

Figure 208

```
   C: STAG
   S: 510 UNKNOWN command
```

Figure 209

```
   C: STAG -1
   S: 511 INVALID UID
   C: STAG 0x1 + SEEN
   S: 511 UID not found
   C: STAG UIDx1234 SEEN
   S: 511 INVALID tag list (missing operator)
```

Figure 210

4.56. Command STAT - authenticated state

Name: status

Arguments: none

Result: 200 512

Result 200 - the status of the folder was successfully delivered.

Result 512 - no folder is selected.

Description: Return the absolute path of currently selected folder (PATH), its type (TYPE), its FCID, its ECID (only for folders with can hold items), the tags (TAGS), the number of items holded and seen in this folder. In case of a FILT folder is returned 1 if there is a filter definition and 0 if there is no filter defined. A change in a folder means that its structure was changed or there was a change of its items (like new messages, an event was canceled and automatically removed from a calender). If current folder is a link to an other folder, it is also LINK returned.

Examples:

```
C: STAT
S: .PATH /INBOX
S: .TYPE MESG
S: .FCID 1
S: .TAGS RESERVED
S: .TOTAL 10
S: .SEEN 2
S: .LINK
S: 200 OK Folder status displayed
```

Figure 211

```
C: STAT
S: 512 ERROR First select a folder
```

Figure 212

4.57. Command STOR - authenticated state

Name: store

Arguments: none

Result: 110 200 410 511

Result 110 - the client can send the item.

Result 200 - the item was successfully stored or there was no content sent by the client.

Result 410 - if the item cannot be stored.

Result 511 - if the data is not a valid XML or its schema does not correspond to the type of the destination folder.

Result 541 - the user does not have enough rights to write items.

Description: Store a new item/filter into a folder. If it is written a new filter into a FILT folder, then the previous filter is deleted. If the new filter has an invalid XML structure or cannot be saved then the folder remains with the old filter (if any). The server can send a 410 or 511 respons before the empty line is send, so the client must check after each sended line of content if the server had rejected the content.

Note: Do not send a message content using CDATA as it can hold empty lines and an empty line means for the server the end of the message

to be stored.

Examples:

```
C: STOR
S: 110 Send the message ended with an empty line
C: <MESSAGE><HEADER>...</HEADER><TEXT>...</TEXT></MESSAGE>
C:
S: 200 OK Message stored (UID is UIDx1234)
C: STOR
S: 110 Send the message ended with an empty line
C:
S: 200 OK Message not stored as it was empty
```

Figure 213

```
C: STOR
S: 110 Send the message ended with an empty line
C: <MESSAGE><HEADER>...</HEADER><HTML>...</HTML></MESSAGE>
C:
S: 410 Cannot store it, not enough space
```

Figure 214

```
C: STOR
S: 110 Send the message ended with an empty line
C: msg
C:
S: 511 Cannot store it, the message has an incorrect format
```

Figure 215

4.58. **Command SUID - authenticated state**

Name: last UID returned by STOR, COPY, MOVE or RPLC

Arguments: none

Results: 200 511

Result 200 - the command was accepted and eventually an UID was returned.

Result 511 - the command is not accepted in the actual state.

Description: This command returns the last UID generated by a STOR, COPY, MOVE or RPLC command in the currently selected folder since it was last time selected. Selecting an other folder or leaving the actual state makes to forget last generated UID. By selecting a folder, storing an item and then reselecting the same folder makes the UID to be forgot. If there is no UID stored then is returned a 200 without any line holding an UID. A failling STOR, COPY, MOVE or RPLC also makes no UID to be remembered.

Examples:

```
C: SLCT /Mbox
S: 200 Selected /Mbox
C: SUID
S: 200 OK There was no STOR since last SLCT
C: STOR
...
S: 200 OK STOR completed with UID UIDx1234
C: SUID
S: .UIDx1234
S: 200 OK SUID completed; found UIDx1234
C: SLCT /Mbox
S: 200 Selected /Mbox
C: SUID
S: 200 OK There was no STOR since last SLCT
C: STOR
...
S: 511 ERROR STOR ompleted with error
C: SUID
S: 200 OK There was no successfull STOR since last SLCT
```

Figure 216

```
C: SUID
S: 511 ERROR SUID is not accepted in this state
```

Figure 217

5. Responses

5.1. Semantic and Syntax

The Response-Code element is a 3-digit integer result code of the attempt to understand and satisfy the request. These codes are fully defined in the following section.

After the Response-Code, can follow a 0x20 character and then a Reason-Phrase intended to give a short textual description of the returned code. The Response-Code is intended for automatic use. The Reason-Phrase is intended for humane persons that debug the connection.

The first digit of the Response-Code defines the class of response. The last two digits do not have any categorization role. There are 4 values for the first digit:

- 1xx: Informational - Server waits for request continuation or send unrequested data;

- 2xx: Success - The action was successfully executed;

- 4xx: Server Error - The server failed to perform the request, retry later;

- 5xx: Server Error - The server failed to perform the request, permanent error;

There are commands that return a multi-line response. These are: CAPA, FIND, GTAG, LIST, RETR, and STAT. In this cases, the response code is at the beginning of the last line of the response. All other lines start with a dot (.).

5.2. 1xx Informational

5.2.1. 100 Reserved

Reserved.

5.2.2. 110 Continue

The client SHOULD continue sending the rest of this request. This response informs the client that the server accepted the initial part of the request and it is waiting for the next part of the request. The server sends a final response after the request has been completely received and processed.

5.3. 2xx Success

5.3.1. 200 OK

The request was successfully processed.

5.3.2. 210 Partial OK

The request was successfully applied for at least one item but not for all requested items. (see FCPY, FDEL, FMOV, and FTAG)

5.3.3. 220 Nothing to do

The request was successful, but none of the arguments were found. (see DACL, DPBL)

5.4. 4xx Temporary Server Error

5.4.1. 400 Reserved

Reserved.

5.4.2. 401 Internal Error

The request could not be processed because it was an internal error (ex.: something is wrong configured).

5.4.3. 410 Retry later

The operation must to be retried later. This return code is used when the data cannot be stored because there was an error (ex.: not enough space on disk) or the operation faild now but can succeed later (ex.: the client listen on proposed IP and port).

5.5. 5xx Permanent Server Error

5.5.1. 500 Reserved

Reserved.

5.5.2. 510 Unknown Command

The request could not be processed because this command is unknown or its syntax is wrong.

5.5.3. 511 Invalid Parameter Format

The request could not be processed because the command has an invalid parameter.

This answer can be returned even in the case when more than one 0x20 character were present between command and its arguments or between arguments.

5.5.4. 512 Out of order

This command has a valid syntax but must to be send after other command required by the logic of the server. (Ex.: PASS after USER in Pre-authenticated State.)

5.5.5. 521 Not found

This command has a valid syntax but the searched argument does not exist or cannot be accessed.

5.5.6. 531 Banned

The client is not allowed to interact with the server. (Ex.: the client's IP is blacklisted.)

5.5.7. 541 Not enough rights

The client is not allowed to do the command because of insufficient rights. If it had enough rights then the command would have been successful. (Ex.: the client cannot store a message with FSTO.)

6. All Possible Response Codes for All Commands

6.1. Not-authenticated State

The Welcome Message: 200 401 410 531

AUTH: 510 511

AUTH mechanism: 200 511

CAPA: 200

QUIT: 200

SGZP: 200 510

STLS: 200 510

other: 510

6.2. Pre-authenticating State (PLAIN method)

PASS: 510 511 512

PASS password: 200 511 512

QUIT: 200

USER: 510 511

USER account: 200 511

other: 510

6.3. Pre-authenticating State (MD5 and SHA1 methods)

HASH: 510 511 512

HASH hashcode: 200 511 512

QUIT: 200

USER: 510 511

USER account: 200 511

other: 510

6.4. Authenticated State

AACL: 510

AACL arguments: 200 511 541

APBL: 200 410

CHNG: 200 510

CHNG arguments: 200 511

COPY: 510 511

COPY arguments: 200 511 541

CPFC: 510 511

CPFC arguments: 200 511 541

CPYF: 510 511

CPYF arguments: 200 511 541

DACL: 510

DACL arguments: 200 220 511 541

DPBL: 200 220 410

DELE: 510 511

DELE arguments: 200 511 541

DELF: 510 511

DELF arguments: 200 511 541

EXIT: 200

FCNT: 110

FCNT arguments: 200 220 511 541

FCPY: 110

FCPY arguments: 200 210 220 511 541

FDEL: 110

FDEL arguments: 200 210 220 511 541

FIND: 110

FIND arguments: 200 220 511 541

FMOV: 110

FMOV arguments: 200 210 511 541

FRTR: 110

FRTR arguments: 200 220 511 541

FTAG: 510

FTAG arguments: 110 200 210 220 511 541

GACL: 200

GFTG: 200

GPBL: 200

GTAG: 510

GTAG arguments: 200 511

LINK: 510

LINK arguments: 110 200 511 541

LIST: 200 220

LIST arguments: 200 220 511 541

LSTX: 200 220

LSTX arguments: 200 220 511 541

MAKE: 510 511

MAKE arguments: 200 511 541

MOVE: 510 511

MOVE arguments: 200 511 541

MOVF: 510 511

MOVF arguments: 200 511 541

MVFC: 510 511

MVFC arguments: 200 511 541

NAME: 510 511

NAME arguments: 200 511 541

NOOP: 200

PGET: 510 511

PGET arguments: 110 200 510 511

PSET: 510 511

PSET arguments: 200 510 511

PSHA: 510 511

PSHA arguments: 200 410 510 511

PSHD: 510 511

PSHD arguments: 200 510 511

QUIT: 200

RETC: 510

RETC arguments: 200 511 541

RETR: 510 511

RETR arguments: 200 511 541

RPLC: 510 511

RPLC argument: 110 200 210 410 511 541

SLCT: 510 511

SLCT arguments: 200 511

SFTG: 510 511

SFTG arguments: 200 410 511 541

SPWD: 110 200 511 541

STAG: 510 511

STAG arguments: 200 410 511 541

STAT: 200 512

STOR: 110 200 410 511 541

SUID: 200 511

other: 510

6.5. Not-selected State

CHNG: 200 510

CHNG arguments: 200 511

GPBL: 200

LIST: 200

LIST arguments: 200 511

PSET: 510 511

PSET arguments: 200 510 511

SLCT: 510 511

SLCT arguments: 200 511

SPWD: 110 200 511 541

other: 510

6.6. Presence State

PGET: 510 511

PGET arguments: 200 510 511

QUIT: 200

other: 510

6.7. Storing State

FSTO: 510

FSTO arguments: 110 200 410 510 511 541

QUIT: 200

other: 510

7. Example of Conversations

7.1. Successful connection and authentication

```
   S: 200 Welcome
   C: AUTH PLAIN
   S: 200 OK Send USER
   C: USER account
   S: 200 OK Send PASS
   C: PASS password
   S: 200 OK Authenticated
   C: STAT
   S: .PATH /INBOX
   S: .TYPE MESG
   S: .TAGS RESERVED
   S: .TOTAL 10
   S: 200 OK Folder status displayed
```

Figure 218

```
   S: 200 Welcome
   C: AUTH MD5
   S: .Use this as prefix, please!
   S: 200 OK Send USER
   C: USER account
   S: 200 OK Send HASH
   C: HASH 79054025255fb1a26e4bc422aef54eb1
   S: 200 OK Authenticated
   C: STAT
   S: .PATH /INBOX
   S: .TYPE MESG
   S: .TAGS RESERVED
   S: .TOTAL 10
   S: 200 OK Folder status displayed
```

Figure 219

7.2. Successful connection but unsuccessful authentication

```
   S: 200 Welcome
   C: AUTH PLAIN
   S: 200 OK Send USER
   C: USER account
   S: 200 OK Send PASS
   C: PASS password
   S: 511 WRONG user/password pair
```

Figure 220

7.3. Connection refused

```
   S: 531 Your IP is blacklisted
```

Figure 221

```
   S: 410 Too many connections, please retry later
```

Figure 222

```
   S: 401 Internal error, the server has an error in its configuration
```

Figure 223

7.4. Find what folders are available with messages

```
   C: LIST /*
   S: .MESG /INBOX
   S: .MESG /TRASH
   S: .CALE /CALENDAR
   S: 200 OK LIST completed (3 matches)
```

Figure 224

7.5. Find all items available in a folder

```
   C: SLCT /INBOX
   S: 200 OK Folder selected
   C: FIND
   S: 110 OK SEND filter definition (end it with an empty line)
   C:
   S: .UIDx1230
   S: .UIDx1231
   S: .UIDx1234
   S: .UIDx1235
   S: .UIDx2340
   S: 200 OK FIND completed (5 matches)
```

Figure 225

7.6. Retrieve a message

```
   C: SLCT /INBOX
   S: 200 OK Folder selected
   C: FIND
   S: 110 OK SEND filter definition (end it with an empty line)
   C: NEW IS /HEADER/subject = 'Newsletter from Example.com'
   C:
   S: .UIDx1234
   S: .UIDx1235
   S: .UIDx2340
   S: 200 OK FIND completed (3 matches)
   C: RETR UIDx1234
   S: .<MESSAGE><HEADER>
   S: .<from>HCCP&lt;news@example.com&gt;</from>
   S: .<to>newsletter@localhost.localdomain</to>
   S: .<subject>Newsletter from Example.com</subject>
   S: .</HEADER>
   S: .<TEXT>This is your weekly newsletter.</TEXT>
   S: .</MESSAGE>
   S: 200 OK RETR completed
```

Figure 226

7.7. Store a message

```
   C: SLCT /OUTBOX
   S: 200 OK Folder selected
   C: STOR
   S: 110 Send the message ended with an empty line
   C: <MESSAGE><HEADER>
   C: <from>HCCP&lt;news@example.com&gt;</from>
   C: <to>newsletter@localhost.localdomain</to>
   C: <subject>HCCP Newsletter</subject>
   C: </HEADER>
   C: <TEXT>This is your weekly newsletter.</TEXT>
   C: </MESSAGE>
   C:
   S: 200 OK Message stored (UID is UIDx1234)
```

Figure 227

7.8. Mark messages as SPAM and move them in a new folder

```
   C: STAG UIDx1000 + SPAM
   S: 200 OK STAG completed
   C: MAKE MESG /Archive-SPAM
   S: 200 OK Folder created
   C: FMOV /Archive-SPAM
   S: 110 OK SEND filter definition (end it with an empty line)
   C: TAG SPAM
   C:
   S: 200 OK FMOV completed (19 matches)
```

Figure 228

7.9. Create a filter folder, find the matching items of the filter and read its filter definition

```
   C: MAKE FILT /New-messages
   S: 200 OK Folder created
   C: STOR
   S: 110 Send the filter content ended with an empty line
   C: <FILTER>
   C: <FOLDERS><FOLDER>/INBOX</FOLDER></FOLDERS>
   C: <RULES>
   C: <AND><NOT><TAG>SEEN</TAG></NOT></AND>
   C: </RULES>
   C: </FILTER>
   C:
   S: 200 OK Filter stored
   C: SLCT /New-messages
   S: 200 OK Folder selected
   C: FIND
   S: .UIDx1234 /INBOX
   S: .UIDx1234 /Trash
   S: .UIDx1235 /Trash
   S: 200 OK FIND completed (3 matches)
   C: RETR
   S: .<FILTER>
   S: .<FOLDERS><FOLDER>/INBOX</FOLDER></FOLDERS>
   S: .<RULES><NOT><TAG>SEEN</TAG></NOT></RULES>
   S: .</FILTER>
   S: 200 OK RETR completed
```

Figure 229

7.10. Create a folder and rename it

```
   C: MAKE MESG /My/NewFolder
   S: 200 OK Folder created
   C: NOOP
   S: 200 NOOP OK
   C: SLCT /My/NewFolder
   S: 200 OK Selected /My/NewFolder
   C: NAME AFolder
   S: 200 OK /My/NewFolder --> /My/AFolder
```

Figure 230

7.11. Find the status for a folder

```
   C: LIST /*
   S: .MESG /INBOX
   S: .MESG /TRASH
   S: .CALE /CALENDAR
   S: 200 OK LIST completed (3 matches)
   C: SLCT /INBOX
   S: 200 OK SELECT completed
   C: STAT
   S: .PATH /INBOX
   S: .TYPE MESG
   S: .TAGS RESERVED
   S: .TOTAL 10
   S: 200 OK Folder status displayed
```

Figure 231

7.12. Set and check the tags of a message

```
   C: STAG UIDx1000 + SEEN
   S: 200 OK STAG completed
   C: GTAG UIDx1000
   S: .SPAM
   S: .FLAG=RED
   S: .SEEN
   S: 200 OK GTAG completed
```

Figure 232

7.13. Find messages that can be SPAM and delete them

```
   C: FTAG + SPAM
   S: 110 OK SEND filter definition (end it with an empty line)
   C: REGEXP header/subject = '[Vv][i1]agra'
   C:
   S: 200 OK FTAG completed (10 matches)
   C: FDEL
   S: 110 OK SEND filter definition (end it with an empty line)
   C: UID 00000001:00001000 AND TAG SPAM
   C:
   S: 200 OK FDEL completed (10 matches)
```

Figure 233

7.14. Connect for a short period

```
   C: PSET FOR 15 min
   S: 200 Nice to se you back
   C: PSET AT At Home
   S: 200 So you are there
   C: PSET STATUS Today I am doing HomeOffice
   S: 200 So kind to share your thoughts with us
   C: PGET USER coworker
   S: .AWAY 2011-05-27 18:00:00 +1000
   S: 200 Sorry, your buddy is not here
   C: PSET AWAY
   S: 200 Oh, so soon
```

Figure 234

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <http://www.rfc-editor.org/info/rfc3552>.

8.2. Informative References

[ISO.8601.1988] International Organization for Standardization, "Data elements and interchange formats - Information interchange - Representation of dates and times", ISO Standard 8601, June 1988.

[RFC1952] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, DOI 10.17487/RFC1952, May 1996, <http://www.rfc-editor.org/info/rfc1952>.

[RFC2426] Dawson, F. and T. Howes, "vCard MIME Directory Profile", RFC 2426, DOI 10.17487/RFC2426, September 1998, <http://www.rfc-editor.org/info/rfc2426>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <http://www.rfc-editor.org/info/rfc2782>.

[RFC2821] Klensin, J., Ed., "Simple Mail Transfer Protocol", RFC 2821, DOI 10.17487/RFC2821, April 2001, <http://www.rfc-editor.org/info/rfc2821>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <http://www.rfc-editor.org/info/rfc3339>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

[RFC3921] Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 3921, DOI 10.17487/RFC3921, October 2004, <http://www.rfc-editor.org/info/rfc3921>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <http://www.rfc-editor.org/info/rfc4648>.

[RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <http://www.rfc-editor.org/info/rfc5545>.

Author's Address

Iulian Radu (editor)

Email: iulian.radu@gmx.at