IMAP Extensions Working Group
Internet Draft

B. Leiba
IBM T.J. Watson Research Center
A. Melnikov
Isode Limited

Expires April 2005
October 2004

IMAP4 LIST Command Extensions
draft-ietf-imapext-list-extensions-10.txt

Status of this Document

A revised version of this draft document will be submitted to the RFC editor as an Proposed Standard for the Internet Community. Discussion and suggestions for improvement are requested, and should be sent to ietf-imapext@imc.org.

This documents obsoletes RFC 3348 and updates RFC 2193.

Abstract

IMAP4 has two commands for listing mailboxes: LIST and LSUB. As we have added extensions that have required specialized lists (see [MboxRefer] for an example) we have had to expand the number of list commands, since each extension must add its function to both LIST and LSUB, and these commands are not, as they are defined, extensible. If we've needed the extensions to work together, we've had to add a set of commands to mix the different options, the set increasing in size with each new extension. This document describes an extension to the base LIST command that will allow these additions to be done

with mutually compatible options to the LIST command, avoiding the exponential increase in specialized list commands.

1. Conventions used in this document

In examples, "C:" indicates lines sent by a client that is connected to a server. "S:" indicates lines sent by the server to the client.

The words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" are used in this document as specified in RFC 2119 [Keywords].

The term "canonical LIST pattern" refers to the canonical pattern constructed internally by the server from the reference and mailbox name arguments (Section 6.3.8 of [IMAP4]). The [IMAP4] LIST command returns only mailboxes that match the canonical LIST pattern.

Other terms are introduced where they are referenced for the first time.

<<Other editorial comments/questions are enclosed like this.>>

2. Introduction and overview

The extensions to the LIST command is accomplished by amending the syntax to allow options to be specified. The list of options replaces the several commands that are currently used to mix and match the information requested. The new syntax is backward-compatible, with no ambiguity: the new syntax is being used if one of the following conditions is true:

1). if the first word after the command name begins with a parenthesis ("LIST selection options");
2). if the second word after the command name begins with a parenthesis ("multiple mailbox patterns");
3). if the LIST command has more than 2 parameters ("LIST return options");

Otherwise the original syntax is used.

By adding options to the LIST command, we are announcing the intent to phase out and eventually to deprecate the RLIST and RLSUB commands described in [MboxRefer]. We are also defining the mechanism to request extended mailbox information, such as is described in the "Child Mailbox Extension" [ChildMbox]. The base LSUB command is not deprecated by this extension; rather, this extension adds a way to obtain subscription information with more options, with those server implementations that support it. Clients that simply need a list of subscribed mailboxes, as provided by the LSUB command, SHOULD continue to use that command.

This document defines an IMAP4 extension that is identified by the

capability string "X-DRAFT-W10-LISTEXT" <<Note to the RFC Editor: the capability name will change upon publication as an RFC>>. The X-DRAFT-W10-LISTEXT extension makes the following changes to the IMAP4 protocol, which are described in more details in sections 3 and 4 of this document:

a. defines new syntax for LIST command options.
b. extend LIST to allow for multiple mailbox patterns.
c. adds LIST command selection options: SUBSCRIBED, REMOTE and MATCHPARENT.
d. adds LIST command return options: SUBSCRIBED and CHILDREN.
e. adds new mailbox attributes: "\NonExistent", "\PlaceHolder", "\Subscribed", "\Remote", "\HasSubMailboxes", "\HasChildren" and "\HasNoChildren".

3. Extended LIST Command

This extension updates the syntax of the LIST command to allow for multiple mailbox patterns to be specified, if they are enclosed in parantheses. A mailbox name match a list of mailbox patterns if it matches at least one mailbox pattern. Note that if a mailbox name matches multiple mailbox patterns from the list, the server should return only a single LIST response.

Note that the non-extended LIST command is required to treat an empty ("" string) mailbox name argument as a special request to return the hierarchy delimiter and the root name of the name given in the reference parameter (as per [IMAP4]). However ANY extended LIST command (extended in any of 3 ways specified in section 2, or any combination of therof) MUST NOT treat the empty mailbox name as such special request and any regular processing described in this document applies. In particular, if an extended LIST command has multiple mailbox names and one (or more) of them is the empty string, the empty string MUST be ignored for the purpose of matching. <<Open issue: what if one of the mailbox patterns is invalid for the server? Does it have to reject the LIST command with BAD response, or should it just ignore invalid patterns?>>

The LIST command syntax is also extended in two additional ways: by adding a parenthesized list of command options between the command name and the reference name (LIST selection options) and an optional list of options at the end that control what kind of information should be returned (LIST return options). See the formal syntax in section 6 for specific details.

A LIST selection option tells the server which mailbox names should be selected by the LIST operation. The server should return information about all mailbox names that match any of the "canonical LIST pattern" (as described above) and satisfy additional selection criteria (if any) specified by the LIST selection options. Let's call any such mailbox name a "matched mailbox name".

When multiple selection options are specified, the server must return information about mailbox names that satisfy every selection option, unless a description of a particular specified option prescribes special rules. An example of an option prescribing special rules is the MATCHPARENT selection option described later in this section.
We will use the term "selection criteria" when referring collectively to all selection options specified in a LIST command.

A LIST return option controls which information is returned for each matched mailbox name. Note that return options MUST NOT cause the server to report information about additional mailbox names. If the client has not specified any return option, only information about attributes should be returned by the server. (Of course the server is allowed to include any other information at will)

Both selection and return command options will be defined in this document and in approved extension documents; each option will be enabled by a capability string (one capability may enable multiple options), and a client MUST NOT send an option for which the server has not advertised support. A server MUST respond to options it does not recognize with a BAD response. The client SHOULD NOT specify any option more than once, however if the client does this, the server MUST act as if it recieved the option only once.

This extension is identified by the capability string "X-DRAFT-W10-LISTEXT" <<Note to the RFC editor: please update upon publication as above>>, and support for it is a prerequisite for any future extensions that require specialized forms of the LIST command. Such extensions MUST refer to this document and MUST add their function through command options as described herein.
Note that extensions that don't require support for an extended LIST command, but use extended LIST responses (see below), don't need to advertise the "X-DRAFT-W10-LISTEXT" capability string. <<Note to the RFC Editor: fix upon publication as above>>

This extension also defines extensions to the LIST response, allowing a series of extended fields at the end, a parenthesized list of tagged data (also referred to as "extended data item"). The first element of an extended field is a tag, which identifies type of the data. Tags MUST be registered with IANA, as described in section 8.5 of this document. An example of such extended set might be

```
((tablecloth (("fringe" "lacy")("color" "white")))(X-Sample
"text"))
```

or...

```
((tablecloth ("fringe" "lacy"))(X-Sample "text" "and even more text"))
```

See the formal grammar, below, for the full syntactic details.
The server MAY return data in the extended fields that was not solicited by the client. The client MUST ignore all extended fields it doesn't recognize.

The X-DRAFT-W10-LISTEXT <<Note to the RFC Editor: fix upon publication>> capability also defines several new mailbox attributes.

The "\PlaceHolder" attribute indicates that the designated mailbox does not meet the selection criteria of the given LIST command, but that it has one or more child mailbox that might (unspecified whether any, all, or none match the canonical LIST pattern).
The LSUB command indicates this condition by using the "\NoSelect" attribute, but the LIST (SUBSCRIBED) command MUST NOT do that, since "\NoSelect" retains its original meaning here. Further, the "\PlaceHolder" attribute is more general, in that it can be used with any extended set of selection criteria.

The "\HasSubMailboxes" attribute indicates that the designated mailbox meets the selection criteria of the given LIST command and also has one or more child mailbox that might (unspecified whether any, all, or none match the canonical LIST pattern).

The "\Placeholder" and the "\HasSubMailboxes" attributes MUST only be returned when the client has specified the MATCHPARENT selection option.

When the MATCHPARENT selection option was specified by the client, the absence of both \PlaceHolder and \HasSubMailboxes means that the mailbox meets the selection criteria, but doesn't have any children that also meets the selection criteria and doesn't match the canonical LIST pattern. However, absence of both \PlaceHolder and \HasSubMailboxes doesn't tell whether there are any children that meet the selection criteria and match the canonical LIST pattern.

The "\PlaceHolder" and the "\HasSubMailboxes" attributes are mutually exclusive.

Examples 8 and 10 in section 5 demonstrates the difference between "\Placeholder"/""\HasSubMailboxes" and "\HasChildren" attributes.

The "\NonExistent" attribute indicates that a mailbox does not actually exist. Note that this attribute is not meaningful by itself, as mailboxes that match the canonical LIST pattern but don't exist must not be returned unless one of the two conditions listed below is also satisfied:

a) the mailbox also satisfies the selection criteria

b) the mailbox has at least one child mailbox that satisfies the selection criteria, but doesn't match the canonical LIST pattern.

In practice this means that the "\NonExistent" attribute is usually returned

with one or more of "\PlaceHolder"/"\HasSubMailboxes", "\Subscribed", "\Remote" (see their description below).

The "\NonExistent" attribute implies "\NoSelect". The "\NonExistent" attribute MUST be supported and MUST be accurately computed.

The following table summarizes when "\NonExistent", "\PlaceHolder" or "\HasSubMailboxes" attributes are to be returned (Note that "\PlaceHolder" or "\HasSubMailboxes" attributes are only returned when MATCHPARENT selection option is specified):

exists	meets the selection criteria	has a child that meets the selection criteria	returned LISTEXT attributes
no	no	no	no LIST response returned
yes	no	no	no LIST response returned
no	yes	no	(\NonExistent)
yes	yes	no	()
no	no	yes	(\NonExistent \PlaceHolder)
yes	no	yes	(\PlaceHolder)
no	yes	yes	(\NonExistent \HasSubMailboxes)
yes	yes	yes	(\HasSubMailboxes)

The selection options defined in this specification are

SUBSCRIBED - causes the LIST command to list subscribed names, rather than the existing mailboxes. This will often be a subset of the actual mailboxes. It's also possible for this list to contain the names of mailboxes that don't exist. In any case, the list MUST include exactly those mailbox names that match the canonical list pattern and are subscribed to. This option is intended to supplement the LSUB command.
Of particular note are the mailbox attributes as returned by this option, compared with what is returned by LSUB. With the latter, the attributes returned may not reflect the actual attribute status on the mailbox name, and the \NoSelect attribute has a special meaning (it indicates that this mailbox is not, itself, subscribed, but that it has child mailboxes that are). With the SUBSCRIBED selection option described here, the attributes are accurate, complete, and have no special meanings.
"LSUB" and "LIST (SUBSCRIBED)" are, thus, not the same thing, and some servers must do significant extra work to respond to "LIST (SUBSCRIBED)". Because of this, clients SHOULD continue to use "LSUB" unless they specifically want the additional

information offered by "LIST (SUBSCRIBED)".

This option defines a new mailbox attribute, "\Subscribed", that indicates that a mailbox name is subscribed to. The "\Subscribed" attribute MUST be supported and MUST be accurately computed when the SUBSCRIBED selection option is specified.

Note that the SUBSCRIBED selection option implies the SUBSCRIBED return option (see below).

REMOTE - causes the LIST command to show remote mailboxes as well as local ones, as described in [MboxRefer]. This option is intended to replace the RLIST command and, in conjunction with the SUBSCRIBED selection option, the RLSUB command.

This option defines a new mailbox attribute, "\Remote", that indicates that a mailbox is a remote mailbox. The "\Remote" attribute MUST be accurately computed when the REMOTE option is specified.

Note that a server implementation that doesn't support any remote mailboxes is compliant with this specification as long as it accepts and ignores the REMOTE selection option.

MATCHPARENT - when this option is specified, the "\Placeholder" and the "\HasSubMailboxes" attributes MUST be accurate (see their description above). This might force the server to return information about parent mailboxes that don't match other selection options, but have some submailboxes that do.

Note 1: In order for a parent mailbox to be returned, it still has to match the canonical LIST pattern.

Note 2: When calculating "\Placeholder"/"\HasSubMailboxes" attributes, it doesn't matter if the submailbox matches the canonical LIST pattern or not. See also example 9 in section 5.

The MATCHPARENT option MUST NOT occur as the only selection option, as it only makes sense when other selection options are also used. The server MUST return BAD tagged response in such case.

Note that even if MATCHPARENT option is specified, the client MUST still be able to handle a case when a "\PlaceHolder"/ "\HasSubMailboxes" is returned and there are no submailboxes that meet the selection criteria of the given LIST command, as they can be deleted/renamed after the LIST response was sent, but before the client had a chance to access them.

The return options defined in this specification are

SUBSCRIBED - causes the LIST command to return subscription

state for all matching mailbox names. The "\Subscribed" attribute MUST be supported and MUST be accurately computed when the SUBSCRIBED return option is specified.

CHILDREN - Requests mailbox child information as originally proposed in [ChildMbox]. See section 4, below, for details. This option MUST be accepted by all servers, however it MAY be ignored.

3.1. General principles for returning LIST responses

This section outlines several principles that can be used by server implementations of this document to decide if a LIST response should be returned, as well as how many responses and what kind of information they may contain.

1) Exactly one LIST response should be returned for each mailbox name which matches the canonical LIST pattern.
   Server implementors must not assume that clients will be able to assemble mailbox attributes and other information returned in multiple LIST responses.

2) There are only two reasons for including a matching mailbox name in the responses to the LIST command (Note that the server is allowed to return unsolicited responses at any time. Such responses are not governed by this rule):

   a) the mailbox name also satisfies the selection criteria;

   b) the mailbox name doesn't satisfy the selection criteria, but it has at least one child mailbox name that satisfies the selection criteria and that doesn't match the canonical LIST pattern.
      For more information on this case see the \PlaceHolder attribute description in Section 3. Note that the "\Placeholder" attribute can only be returned when the MATCHPARENT selection option is specified.

3) Attributes returned in the same LIST response must be treated additively. For example the following response

   S: * LIST (\Subscribed \NonExistent) "/" "Fruit/Peach"

   means that the "Fruit/Peach" mailbox doesn't exist, but it is subscribed.

3.2. Additional requirements on LISTEXT clients

All clients that support this extension MUST treat an attribute with a stronger meaning, as implying any attribute that can be inferred from it.

For example, the client must treat presence of the \NoInferiors attribute as if the \HasNoChildren attribute was also sent by the server.

The following table summarizes inference rules described in section 3.

returned attribute	implied attribute
\NoInferiors	\HasNoChildren
\NonExistent	\NoSelect

4. The CHILDREN return Option

The CHILDREN return option implements the Child Mailbox Extension, originally proposed by Mike Gahrns and Raymond Cheng, of Microsoft Corporation. Most of the information in this section is taken directly from their original specification [ChildMbox]. The CHILDREN return option is simply an indication that the client wants this information; a server MAY provide it even if the option is not specified, or MAY ignore the option entirely.
Many IMAP4 [IMAP4] clients present to the user a hierarchical view of the mailboxes that a user has access to. Rather than initially presenting to the user the entire mailbox hierarchy, it is often preferable to show to the user a collapsed outline list of the mailbox hierarchy (particularly if there is a large number of mailboxes). The user can then expand the collapsed outline hierarchy as needed. It is common to include within the collapsed hierarchy a visual clue (such as a ''+'') to indicate that there are child mailboxes under a particular mailbox. When the visual clue is clicked the hierarchy list is expanded to show the child mailboxes.
The CHILDREN return option provides a mechanism for a client to efficiently determine if a particular mailbox has children, without issuing a LIST "" * or a LIST "" % for each mailbox name.
The CHILDREN return option defines two new attributes that MAY be returned within a LIST response: \HasChildren and \HasNoChildren.
While these attributes MAY be returned in response to any LIST command, the CHILDREN return option is provided to indicate that the client particularly wants this information. If the CHILDREN return option is present, the server SHOULD return these attributes even if their computation is expensive.

\HasChildren - The presence of this attribute indicates that the mailbox has child mailboxes.
A server SHOULD NOT set this attribute if there are child mailboxes, and the user does not have permissions to access any of them. In this case, \HasNoChildren SHOULD be used.
In many cases, however, a server may not be able to efficiently compute whether a user has access to all child mailboxes. As

such a client MUST be prepared to accept the \HasChildren attribute as a hint. That is, a mailbox MAY be flagged with the \HasChildren attribute, but no child mailboxes will appear in the LIST response.

\HasNoChildren - The presence of this attribute indicates that the mailbox has NO child mailboxes that are accessible to the currently authenticated user.

In some instances a server that supports the LISTEXT extension might not be able to determine whether a mailbox has children. For example it may have difficulty determining whether there are child mailboxes when LISTing mailboxes while operating in a particular namespace.
In these cases, a server MAY exclude both the \HasChildren and \HasNoChildren attributes in the LIST response. As such, a client can not make any assumptions about whether a mailbox has children based upon the absence of a single attribute. In particular, some servers may not be able to combine the SUBSCRIBED selection option and CHILDREN return option. Such servers MUST honour the SUBSCRIBED selection option, and they will simply ignore the CHILDREN return option if both are requested. It is an error for the server to return both a \HasChildren and a \HasNoChildren attribute in a LIST response.

Note: the \HasNoChildren attribute should not be confused with the IMAP4 [IMAP4] defined attribute \NoInferiors which indicates that no child mailboxes exist now and none can be created in the future.

5. Examples

Example 1:

The first example shows the complete local hierarchy that will be used for the other examples.

```
C: A01 LIST "" "*"
S: * LIST (\Marked \NoInferiors) "/" "inbox"
S: * LIST () "/" "Fruit"
S: * LIST () "/" "Fruit/Apple"
S: * LIST () "/" "Fruit/Banana"
S: * LIST () "/" "Tofu"
S: * LIST () "/" "Vegetable"
S: * LIST () "/" "Vegetable/Broccoli"
S: * LIST () "/" "Vegetable/Corn"
S: A01 OK done
```

Example 2:

In the next example, we'll see the subscribed mailboxes. This is similar to, but not equivalent with, <LSUB "" "*">. Note that the mailbox

called "Fruit/Peach" is subscribed to, but does not actually exist (perhaps it was deleted while still subscribed). The "Fruit" mailbox is not subscribed to, but it has two subscribed children. The "Vegetable" mailbox is subscribed and has two children, one of them is subscribed as well.

```
C: A02 LIST (SUBSCRIBED) "" "*"
S: * LIST (\Marked \NoInferiors \Subscribed) "/" "inbox"
S: * LIST (\Subscribed) "/" "Fruit/Banana"
S: * LIST (\Subscribed \NonExistent) "/" "Fruit/Peach"
S: * LIST (\Subscribed) "/" "Vegetable/Broccoli"
S: A02 OK done
```

Example 3:

The next example shows the use of the CHILDREN option. The client, without having to list the second level of hierarchy, now knows which of the top-level mailboxes have submailboxes (children) and which do not. Note that it's not necessary for the server to return the \HasNoChildren attribute for the inbox, because the \NoInferiors attribute already implies that, and has a stronger meaning.

```
C: A03 LIST () "" "%" RETURN (CHILDREN)
S: * LIST (\Marked \NoInferiors) "/" "inbox"
S: * LIST (\HasChildren) "/" "Fruit"
S: * LIST (\HasNoChildren) "/" "Tofu"
S: * LIST (\HasChildren) "/" "Vegetable"
S: A03 OK done
```

Example 4:

In this example we see more mailboxes, which reside on another server to which we may obtain referrals. This is similar to the command <RLIST "" "%">. Note that in the case of the remote mailboxes, the server might or might not be able to include CHILDREN information; it includes it if it can, and omits it if it can't.

```
C: A04 LIST (REMOTE) "" "%" RETURN (CHILDREN)
S: * LIST (\Marked \NoInferiors) "/" "inbox"
S: * LIST (\HasChildren) "/" "Fruit"
S: * LIST (\HasNoChildren) "/" "Tofu"
S: * LIST (\HasChildren) "/" "Vegetable"
S: * LIST (\Remote) "/" "Bread"
S: * LIST (\HasChildren \Remote) "/" "Meat"
S: A04 OK done
```

Example 5:

The following example also requests the server to include mailboxes, which reside on another server. The server returns information about all mailboxes which are subscribed. This is similar to the command

<RLSUB "" "%">. We also see the use of two selection options.

```
C: A05 LIST (REMOTE SUBSCRIBED) "" "*"
S: * LIST (\Marked \NoInferiors \Subscribed) "/" "inbox"
S: * LIST (\Subscribed) "/" "Fruit/Banana"
S: * LIST (\Subscribed \NonExistent) "/" "Fruit/Peach"
S: * LIST (\Subscribed) "/" "Vegetable"
S: * LIST (\Subscribed) "/" "Vegetable/Broccoli"
S: * LIST (\Remote \Subscribed) "/" "Bread"
S: A05 OK done
```

Example 6:

The following example requests the server to include mailboxes, which reside on another server. The server is requested to return subscription information for all returned mailboxes. This is different from the example above.

Note that the output of this command is not a superset of the output in the previous example, as it doesn't include LIST response for the non-existent "Fruit/Peach".

```
C: A06 LIST (REMOTE) "" "*" RETURN (SUBSCRIBED)
S: * LIST (\Marked \NoInferiors \Subscribed) "/" "inbox"
S: * LIST () "/" "Fruit"
S: * LIST () "/" "Fruit/Apple"
S: * LIST (\Subscribed) "/" "Fruit/Banana"
S: * LIST () "/" "Tofu"
S: * LIST (\Subscribed) "/" "Vegetable"
S: * LIST (\Subscribed) "/" "Vegetable/Broccoli"
S: * LIST () "/" "Vegetable/Corn"
S: * LIST (\Remote \Subscribed) "/" "Bread"
S: * LIST (\Remote) "/" "Meat"
S: A06 OK done
```

Example 7:

In the following example the client has specified multiple mailbox patterns. Note that this example doesn't use the mailbox hierarchy used in the previous examples.

```
C: BBB LIST "" ("INBOX" "Drafts" "Sent/%")
S: * LIST () "/" "INBOX"
S: * LIST (\NoInferiors) "/" "Drafts"
S: * LIST () "/" "Sent/March2004"
S: * LIST (\Marked) "/" "Sent/December2003"
S: * LIST () "/" "Sent/August2004"
S: BBB OK done
```

Example 8:

The following example demonstates the difference between

\HasChildren and \PlaceHolder/\SubMailboxes.

Let's assume there is the following hierarchy:

```
 C: C01 LIST "" "*"
 S: * LIST (\Marked \NoInferiors) "/" "inbox"
 S: * LIST () "/" "Foo"
 S: * LIST () "/" "Foo/Bar"
 S: * LIST () "/" "Foo/Baz"
 S: * LIST () "/" "Moo"
 S: C01 OK done
```

If the client asks RETURN (CHILDREN) it will get:

```
 C: CA3 LIST "" "%" RETURN (CHILDREN)
 S: * LIST (\Marked \NoInferiors) "/" "inbox"
 S: * LIST (\HasChildren) "/" "Foo"
 S: * LIST (\HasNoChildren) "/" "Moo"
 S: CA3 OK done
```

A). Let's also assume that the mailbox "Foo/Baz" is the only subscribed mailbox. Then

```
 C: C02 LIST (SUBSCRIBED) "" "*"
 S: * LIST (\Subscribed) "/" "Foo/Baz"
 S: C02 OK done
```

Now, if the client issues <LIST (SUBSCRIBED) "" "%">, the server will return no mailboxes (as the mailboxes "Moo", "Foo" and "Inbox" are NOT subscribed). However, if the client issues:

```
 C: C04 LIST (SUBSCRIBED MATCHPARENT) "" "%"
 S: * LIST (\PlaceHolder) "/" "Foo"
 S: C04 OK done
```

i.e. the mailbox "Foo" is not subscribed, but it has a child that is.

A1). If the mailbox "Foo" would have been subscribed instead, the last command would return:

```
 S: * LIST (\HasSubMailboxes \Subscribed) "/" "Foo"
```

or even

```
 S: * LIST (\HasSubMailboxes \HasChildren \Subscribed) "/" "Foo"
```

A2). If we assume instead that the mailbox "Foo" is not part of the original hierarchy and is not subscribed, the last command will return

```
 S: * LIST (\PlaceHolder \NonExistent) "/" "Foo"
```

B). Now, let's assume that no mailbox is subscribed. In this case the command <LIST (SUBSCRIBED MATCHPARENT) "" "%"> will return no responses, as there are no subscribed children (even though "Foo" has children).

C). And finally, let's assume that the mailboxes "Foo" and "Moo" are subscribed. In this case the command:

```
C: LIST (SUBSCRIBED MATCHPARENT) "" "%" RETURN (CHILDREN)
```

will return:

```
S: * LIST (\HasChildren \Subscribed) "/" "Foo"
S: * LIST (\HasNoChildren \Subscribed) "/" "Moo"
```

Which means that the mailbox "Foo" has children, but none of them is subscribed.

Example 9:

The following example demonstrates that the calculation of \PlaceHolder (or \HasSubMailboxes) attributes is not affected by the fact that children mailboxes match the canonical LIST pattern.

Let's assume there is the following hierarchy:

```
C: D01 LIST "" "*"
S: * LIST (\Marked \NoInferiors) "/" "inbox"
S: * LIST () "/" "foo2"
S: * LIST () "/" "foo2/bar1"
S: * LIST () "/" "foo2/bar2"
S: * LIST () "/" "baz2"
S: * LIST () "/" "baz2/bar2"
S: * LIST () "/" "baz2/bar22"
S: * LIST () "/" "baz2/bar222"
S: * LIST () "/" "eps2"
S: * LIST () "/" "eps2/mamba"
S: * LIST () "/" "quux2/bar2"
S: D01 OK done
```

And that the following mailboxes are subscribed:

```
C: D02 LIST (SUBSCRIBED) "" "*"
S: * LIST (\Subscribed) "/" "foo2/bar1"
S: * LIST (\Subscribed) "/" "foo2/bar2"
S: * LIST (\Subscribed) "/" "baz2/bar2"
S: * LIST (\Subscribed) "/" "baz2/bar22"
S: * LIST (\Subscribed) "/" "baz2/bar222"
S: * LIST (\Subscribed) "/" "eps2"
S: * LIST (\Subscribed) "/" "eps2/mamba"
S: * LIST (\Subscribed) "/" "quux2/bar2"
S: D02 OK done
```

The client issues the following command first:

```
C: D03 LIST (MATCHPARENT SUBSCRIBED) "" "*2"
S: * LIST (\PlaceHolder) "/" "foo2"
S: * LIST (\Subscribed) "/" "foo2/bar2"
S: * LIST (\PlaceHolder) "/" "baz2"
S: * LIST (\Subscribed) "/" "baz2/bar2"
S: * LIST (\Subscribed) "/" "baz2/bar22"
S: * LIST (\Subscribed) "/" "baz2/bar222"
S: * LIST (\HasSubMailboxes \Subscribed) "/" "eps2"
S: * LIST (\Subscribed) "/" "quux2/bar2"
S: D03 OK done
```

and the server may also include

```
S: * LIST (\PlaceHolder \NonExistent) "/" "quux2"
```

The \PlaceHolder attribute is returned for mailboxes "foo2" and "baz2" (and the \HasSubMailboxes is returned for the mailbox "eps2"), because all of them have subscribed children, even though for the mailbox "foo2" only one of the two subscribed children match the pattern, for the mailbox "baz2" all the subscribed children match the pattern and for the mailbox "eps2" none of the subscribed children match the pattern.

Note that if the client issues

```
C: D03 LIST (MATCHPARENT SUBSCRIBED) "" "*"
S: * LIST (\PlaceHolder) "/" "foo2"
S: * LIST (\Subscribed) "/" "foo2/bar1"
S: * LIST (\Subscribed) "/" "foo2/bar2"
S: * LIST (\PlaceHolder) "/" "baz2"
S: * LIST (\Subscribed) "/" "baz2/bar2"
S: * LIST (\Subscribed) "/" "baz2/bar22"
S: * LIST (\Subscribed) "/" "baz2/bar222"
S: * LIST (\HasSubMailboxes \Subscribed) "/" "eps2"
S: * LIST (\Subscribed) "/" "eps2/mamba"
S: * LIST (\Subscribed) "/" "quux2/bar2"
S: D03 OK done
```

the mailboxes "foo2", "baz2" and "eps2" still have the same \PlaceHolder/\HasSubMailboxes attribute, even though this information is redundant and the client can determine it by itself.

Example 10:

The following example shows usage of multiple mailbox patterns. It also demonstrates that \HasSubmailboxes/\PlaceHolder attributes don't necessarily imply \HasChildren.

```
C: a1 LIST "" ("foo" "foo/*")
```

```
S: * LIST () "/" foo
S: a1 OK done

C: a2 LIST (SUBSCRIBED) "" "foo/*"
S: * LIST (\Subscribed \NonExistent) "/" foo/bar
S: a2 OK done

C: a3 LIST (SUBSCRIBED MATCHPARENT) "" foo RETURN (CHILDREN)
S: * LIST (\Placeholder \HasNoChildren) "/" foo
S: a3 OK done
```

6. Formal Syntax

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in [ABNF]. Terms not defined here are taken from [IMAP4]. "vendor-token" is defined in [ACAP].

```
child-mbox-flag    = "\HasChildren" / "\HasNoChildren"
                     ; attributes for CHILDREN return option, at most one
                     ; possible per LIST response

list               = "LIST" [SP list-select-opts] SP mailbox SP mbox-or-pat
                     [SP list-return-opts]

list-select-opts   = "(" [*(list-select-mod-opt SP) list-select-base-opt
                         *(SP list-select-opt)] ")"
                     ; list selection options, e.g. REMOTE

list-return-opts   = "RETURN" SP "(" [return-option *(SP return-option)] ")"
                     ; list return options, e.g. CHILDREN

mparent-mbox-flag  = "\PlaceHolder" / "\HasSubMailboxes"
                     ; attributes for MATCHPARENT selection option,
                     ; at most one possible per LIST response

mailbox-list       = "(" [mbx-list-flags] ")" SP
                     (DQUOTE QUOTED-CHAR DQUOTE / nil) SP mailbox
                     [SP mbox-list-extended]

mbox-list-extended = "(" [mbox-list-extended-item
                     *(SP mbox-list-extended-item)] ")"

mbox-list-extended-item = "(" mbox-list-extended-item-data ")"

mbox-list-extended-item-data =  mbox-list-extended-item-tag SP nstring-list

mbox-list-extended-item-tag  = astring
                     ; The content MUST conform to either "eitem-vendor-tag"
or
                     ; "eitem-standard-tag" ABNF productions.
                     ; A tag registration template is described in section
```

```
                      ; 8.5 of this document.

mbox-or-pat        = list-mailbox / patterns

patterns           = "(" list-mailbox *(SP list-mailbox) ")"

eitem-vendor-tag   = vendor-tag
                     ; a vendor specific tag for extended list data

eitem-standard-tag = atom
                     ; a tag for extended list data defined in a Standard
                     ; Track or Experimental RFC.

nstring-list       = nstring /
                     "(" [nstring-list *(SP nstring-list)] ")"
                     ;; a recursive list definition

mbox-list-oflag    = child-mbox-flag / mparent-mbox-flag / "\NonExistent" /
                     / "\Subscribed" / "\Remote"

list-select-opt    = list-sel-mod-opt / list-sel-base-opt
                     ; An option registration template is described in
                     ; section 8.3 of this document.

list-sel-base-opt  = "SUBSCRIBED" / "REMOTE" / option-extension
                     ; options that can be used by themselves

list-sel-mod-opt   = "MATCHPARENT" / option-extension
                     ; options that require a list-select-base-opt
                     ; to also be present

return-option      = "SUBSCRIBED" / "CHILDREN" /
                     option-extension

option-extension   = option-vendor-tag / option-standard-tag

option-vendor-tag  = vendor-tag
                     ; a vendor specific option

option-standard-tag= atom
                     ; an option defined in a Standard Track or
                     ; Experimental RFC

vendor-tag         = vendor-token "-" atom
```

7. Security Considerations

This document describes syntactic changes to the specification of the IMAP4 commands LIST, LSUB, RLIST, and RLSUB, and the modified LIST command has the same security considerations as those commands. They are described in [IMAP4] and [MboxRefer].

The Child Mailbox Extension provides a client a more efficient means of determining whether a particular mailbox has children. If a mailbox has children, but the currently authenticated user does not have access to any of them, the server SHOULD respond with a \HasNoChildren attribute. In many cases, however, a server may not be able to efficiently compute whether a user has access to all child mailboxes. If such a server responds with a \HasChildren attribute, when in fact the currently authenticated user does not have access to any child mailboxes, potentially more information is conveyed about the mailbox than intended. In most situations this will not be a security concern, because if information regarding whether a mailbox has children is considered sensitive, a user would not be granted access to that mailbox in the first place.

8. IANA Considerations

8.1. Guidelines for IANA

It is requested that IANA creates two new registries for LISTEXT options and LISTEXT extended response data. The templates and the initial registrations are detailed below.

8.2. Registration procedure and Change control

Registration of a LISTEXT option is done by filling in the template in section 8.3 and sending it via electronic mail to <iana@iana.org>. Registration of a LISTEXT extended data item is done by filling in the template in section 8.5 and sending it via electronic mail to <iana@iana.org>.
IANA has the right to reject obviously bogus registrations, but will perform no review of claims made in the registration form.

A LISTEXT option/extended data item name that starts with "V-" is reserved for vendor specific options/extended data items. All options, whether they are vendor specific or global, should be registered with IANA. If a LISTEXT extended data item is returned as a result of requesting a particular LISTEXT option, the name of the option SHOULD be used as the name of the LISTEXT extended data item.

Each vendor specific options/extended data item MUST start with their vendor-token ("vendor prefix"). The vendor-token MUST be registered with IANA, using the [ACAP] vendor subtree registry.

Standard LISTEXT option/extended data item names are case insensitive. If the vendor prefix is omitted from a vendor specific LISTEXT option/extended data item name, the rest is case insensitive. The vendor prefix itself is not case-sensitive, as it might contain non-ASCII

characters.

While the registration procedures do not require it, authors of LISTEXT options/extended data items are encouraged to seek community review and comment whenever that is feasible. Authors may seek community review by posting a specification of their proposed mechanism as an Internet-Draft. LISTEXT options/extended data items intended for widespread use should be standardized through the normal IETF process, when appropriate.

Comments on registered LISTEXT options/extended response data should first be sent to the "owner" of the mechanism and/or to the IMAPEXT WG mailing list.
Submitters of comments may, after a reasonable attempt to contact the owner, request IANA to attach their comment to the registration itself. If IANA approves of this, the comment will be made accessible in conjunction with the registration LISTEXT options/extended response data itself.

Once a LISTEXT registration has been published by IANA, the author may request a change to its definition. The change request follows the same procedure as the registration request.

The owner of a LISTEXT registration may pass responsibility for the registered option/extended data item to another person or agency by informing IANA; this can be done without discussion or review.

The IESG may reassign responsibility for a LISTEXT option/extended data item.
The most common case of this will be to enable changes to be made to mechanisms where the author of the registration has died, moved out of contact or is otherwise unable to make changes that are important to the community.

LISTEXT registrations may not be deleted; mechanisms which are no longer believed appropriate for use can be declared OBSOLETE by a change to their "intended use" field; such LISTEXT options/extended data items will be clearly marked in the lists published by IANA.

The IESG is considered to be the owner of all LISTEXT options/extended data items
which are on the IETF standards track.

8.3. Registration template for LISTEXT options

```
To: iana@iana.org
Subject: Registration of LISTEXT option X

LISTEXT option name:

LISTEXT option type: (One of SELECTION or RETURN)
```

Implied return options(s), if the option type is SELECTION: (zero or more)

LISTEXT option description:

Published specification (optional, recommended):

Security considerations:

Intended usage:
(One of COMMON, LIMITED USE or OBSOLETE)

Person & email address to contact for further information:

Owner/Change controller:

(Any other information that the author deems interesting may be added below this line.)

8.4. Initial LISTEXT option registrations

It is requested that the LISTEXT option registry is being populated with the following entries:

1)

To: iana@iana.org
Subject: Registration of LISTEXT option SUBSCRIBED

LISTEXT option name: SUBSCRIBED

LISTEXT option type: SELECTION

Implied return options(s): SUBSCRIBED

LISTEXT option description: Causes the LIST command to list subscribed mailboxes, rather than the actual mailboxes.

Published specification : this RFC, section 3.

Security considerations: this RFC, section 7.

Intended usage: COMMON

Person & email address to contact for further information:
Alexey Melnikov <Alexey.Melnikov@isode.com>

Owner/Change controller: IESG <iesg@ietf.org>

2)

To: iana@iana.org

Subject: Registration of LISTEXT option REMOTE

LISTEXT option name: REMOTE

LISTEXT option type: SELECTION

Implied return options(s): (none)

LISTEXT option description: causes the LIST command to return remote mailboxes as well as local ones, as described in RFC 2193.

Published specification : this RFC, section 3.

Security considerations: this RFC, section 7.

Intended usage: COMMON

Person & email address to contact for further information:
Alexey Melnikov <Alexey.Melnikov@isode.com>

Owner/Change controller: IESG <iesg@ietf.org>

3)

To: iana@iana.org
Subject: Registration of LISTEXT option SUBSCRIBED

LISTEXT option name: SUBSCRIBED

LISTEXT option type: RETURN

LISTEXT option description: Causes the LIST command to return subscription state.

Published specification : this RFC, section 3.

Security considerations: this RFC, section 7.

Intended usage: COMMON

Person & email address to contact for further information:
Alexey Melnikov <Alexey.Melnikov@isode.com>

Owner/Change controller: IESG <iesg@ietf.org>

4)

To: iana@iana.org
Subject: Registration of LISTEXT option MATCHPARENT

LISTEXT option name: MATCHPARENT

LISTEXT option type: SELECTION

Implied return options(s): (none)

LISTEXT option description: Requests that \Placeholder/
\HasSubMailboxes attributes are to be returned.

Published specification : this RFC, sections 3.

Published specification : this RFC

Security considerations: this RFC, section 7.

Intended usage: COMMON

Person & email address to contact for further information:
Alexey Melnikov <Alexey.Melnikov@isode.com>

Owner/Change controller: IESG <iesg@ietf.org>

5)

To: iana@iana.org
Subject: Registration of LISTEXT option CHILDREN

LISTEXT option name: CHILDREN

LISTEXT option type: RETURN

LISTEXT option description: Requests mailbox child information.

Published specification : this RFC, sections 3 and 4.

Published specification : this RFC

Security considerations: this RFC, section 7.

Intended usage: COMMON

Person & email address to contact for further information:
Alexey Melnikov <Alexey.Melnikov@isode.com>

Owner/Change controller: IESG <iesg@ietf.org>

8.5. Registration template for LISTEXT extended data item

To: iana@iana.org
Subject: Registration of LISTEXT extended data item X

LISTEXT extended data item tag:

LISTEXT extended data item description:

Which LISTEXT option(s) (and their types) causes this extended data item to be returned (if any):

Published specification (optional, recommended):

Security considerations:

Intended usage:
(One of COMMON, LIMITED USE or OBSOLETE)

Person & email address to contact for further information:

Owner/Change controller:

(Any other information that the author deems interesting may be added below this line.)

9. References

9.1. Normative References

[Keywords] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, Harvard University, March 1997.

[ABNF] Crocker, D., and Overell, P. "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[IMAP4] Crispin, M., "Internet Message Access Protocol - Version 4rev1", RFC 3501, University of Washington, March 2003.

[MboxRefer] Gahrns, M., "IMAP4 Mailbox Referrals", RFC 2193, Microsoft Corporation, September 1997.

[ChildMbox] Gahrns, M. & Cheng, R., "IMAP4 Child Mailbox Extension", RFC 3348, Microsoft Corporation, July 2002.

[ACAP] Newman, C. and J. Myers, "ACAP -- Application Configuration Access Protocol", RFC 2244, November 1997.

10. Acknowledgements

Mike Gahrns and Raymond Cheng of Microsoft Corporation originally devised the Child Mailbox Extension and proposed it in 1997; the idea, as well as most of the text in section 4, is theirs.

This document is the result of discussions on the IMAP4 and IMAPEXT mailing lists and is meant to reflect consensus of those groups. In particular, Mark Crispin, Philip Guenther, Cyrus Daboo, Timo Sirainen, Ken Murchison, Rob Siemborski, Steve Hole, Arnt Gulbrandsen, Larry

Greenfield, Dave Cridland and Pete Maclean were active participants in this discussion or made suggestions to this document.

11. Author's Address

Barry Leiba
IBM T.J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532
Phone: 1-914-784-7941
Email: leiba@watson.ibm.com

Alexey Melnikov
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex
TW12 2BX, UK

Email: Alexey.Melnikov@isode.com
Home page: http://www.melnikov.ca/

12. IPR Disclosure Acknowledgement

By submitting this Internet-Draft, we certify that any applicable patent or other IPR claims of which I am aware have been disclosed, and any of which we become aware will be disclosed, in accordance with RFC 3668.

13. Intellectual Property

copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

14. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.