IMAP Extensions Working Group
Internet-Draft
Obsoletes: 3348 (if approved)
Updates: 2193 (if approved)
Intended status: Standards Track
Expires: March 25, 2007

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

September 21, 2006

IMAP4 LIST Command Extensions
draft-ietf-imapext-list-extensions-18

Status of this Memo

Copyright Notice

Abstract

IMAP4 has two commands for listing mailboxes: LIST and LSUB. As we have added extensions, such as Mailbox Referrals, that have required specialized lists 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.

Note

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

This document obsoletes RFC 3348 and updates RFC 2193.

Table of Contents

1. Conventions used in this document 5

2. Introduction and overview 6

3. Extended LIST Command 8
3.1. Initial list of selection options 10
3.2. Initial list of return options 12
3.3. General principles for returning LIST responses 12
3.4. Additional requirements on LIST-EXTENDED clients 13
3.5. CHILDINFO extended data item 13

4. The CHILDREN return Option 16

5. Examples 18

6. Formal Syntax 25

7. Internationalization Considerations 29

8. Security Considerations 30

9. IANA Considerations 31
9.1. Guidelines for IANA 31
9.2. Registration procedure and Change control 31
9.3. Registration template for LIST-EXTENDED options 32
9.4. Initial LIST-EXTENDED option registrations 33
9.5. Registration template for LIST-EXTENDED extended data item 35
9.6. Initial LIST-EXTENDED extended data item registrations . . . 36

10. Acknowledgements 37

11. References 38
11.1. Normative References 38
11.2. informative References 38

Authors' Addresses 39
Intellectual Property and Copyright Statements 40

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 [Kwds].

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.

2. Introduction and overview

The LIST command is extended by amending the syntax to allow options and multiple patterns 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 [MBRef]. We are also defining the mechanism to request extended mailbox information, such as is described in the "Child Mailbox Extension" [CMbox]. 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 "LIST-EXTENDED". The LIST-EXTENDED extension makes the following changes to the IMAP4 protocol, which are described in more detail in Section 3 and Section 4:

a. defines new syntax for LIST command options.

b. extends LIST to allow for multiple mailbox patterns.

c. adds LIST command selection options: SUBSCRIBED, REMOTE and RECURSIVEMATCH.

d. adds LIST command return options: SUBSCRIBED and CHILDREN.

e. adds new mailbox attributes: "\NonExistent", "\Subscribed", "\Remote", "\HasChildren" and "\HasNoChildren".

f. adds CHILDINFO extended data item.

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 parentheses. A mailbox name matches a list of mailbox patterns if it matches at least one mailbox pattern. 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 thereof) 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.

Some servers might restrict which patterns are allowed in a LIST command. If a server doesn't accept a particular pattern, it MUST silently ignore it.

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 RECURSIVEMATCH 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 received the option only once. The order in which options are specified by the client is not significant.

In general, each selection option except for RECURSIVEMATCH will have a corresponding return option. The REMOTE selection option is an anomaly in this regard, and does not have a corresponding return option. That is because it expands, rather than restricts, the set of mailboxes that are returned. Future extensions to this specification should keep parallelism in mind, and define a pair of corresponding options.

This extension is identified by the capability string "LIST-EXTENDED", 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 "LIST-EXTENDED" capability string.

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 9.5 of this document. An example of such extended set might be

tablecloth (("edge" "lacy") ("color" "red"))) (X-Sample "text"))

or...

tablecloth ("edge" "lacy")) (X-Sample "text" "more text"))

See the formal syntax, in Section 6, for the full syntactic details. The server MUST NOT return any extended data item, unless the client has expressed its ability to support extended LIST responses, for example by using an extended LIST command. The server MAY return

data in the extended fields that was not directly solicited by the client in the corresponding LIST command. For example, the client can enable extra extended fields by using another IMAP extension that make use of the extended LIST responses. The client MUST ignore all extended fields it doesn't recognize.

The LIST-EXTENDED capability also defines several new mailbox attributes.

The "\NonExistent" attribute indicates that a mailbox name does not refer to an existing mailbox. Note that this attribute is not meaningful by itself, as mailbox names 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 name also satisfies the selection criteria (for example, it is subscribed and the "SUBSCRIBED" selection option has been specified)

b. "RECURSIVEMATCH" has been specified, and the mailbox name has at least one descendant mailbox name that does not match the LIST pattern and does match the selection criteria.

In practice this means that the "\NonExistent" attribute is usually returned with one or more of "\Subscribed", "\Remote", "\HasChildren" or the CHILDINFO extended data item (see their description below).

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

3.1. Initial list of selection options

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 second special meaning (it indicates that this mailbox is not, itself, subscribed, but that it has descendant 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 [MBRef]. 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.

The REMOTE selection option has no interaction with other options. Its effect is to tell the server to apply the other options, if any, to remote mailboxes, in addition to local ones. In particular, it has no interaction with RECURSIVEMATCH (see below). A request for (REMOTE RECURSIVEMATCH) is invalid, because a request for (RECURSIVEMATCH) is. A request for (REMOTE RECURSIVEMATCH SUBSCRIBED) is asking for all subscribed mailboxes, both local and remote.

RECURSIVEMATCH - this option forces the server to return information about parent mailboxes that don't match other selection options, but have some submailboxes that do. Information about children is returned in the CHILDINFO extended data item, as described in Section 3.5.

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

Note 2: When returning the CHILDINFO extended data item, it doesn't matter if the submailbox matches the canonical LIST

pattern or not. See also example 9 in Section 5.

The RECURSIVEMATCH option MUST NOT occur as the only selection option (nor only with REMOTE), 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 the RECURSIVEMATCH option is specified, the client MUST still be able to handle a case when a CHILDINFO extended data item is returned and there are no submailboxes that meet the selection criteria of the subsequent LIST command, as they can be deleted/renamed after the LIST response was sent, but before the client had a chance to access them.

3.2. Initial list of return options

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. Further, all mailbox flags MUST be accurately computed (this differs from the behaviour of the LSUB command).

CHILDREN - Requests mailbox child information as originally proposed in [CMbox]. See Section 4, below, for details. This option MUST be supported by all servers.

3.3. 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 descendant mailbox name that satisfies the selection criteria and that doesn't match the canonical LIST pattern.
For more information on this case see the CHILDINFO extended data item described in Section 3.5. Note that the CHILDINFO extended data item can only be returned when the RECURSIVEMATCH 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.

3.4. Additional requirements on LIST-EXTENDED 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

3.5. CHILDINFO extended data item

The CHILDINFO extended data item MUST NOT be returned unless the client has specified the RECURSIVEMATCH selection option.

The CHILDINFO extended data item in a LIST response describes the

selection criteria that has caused it to be returned and indicates that the mailbox has at least one descendant mailbox that matches the selection criteria.

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 CHILDINFO extended data item is more general, in that it can be used with any extended set of selection criteria.

Note: Some servers allow for mailboxes to exist without requiring their parent to exist. For example, a mailbox "Customers/ABC" can exist while the mailbox "Customers" does not. As CHILDINFO extended data item is not allowed if the RECURSIVEMATCH selection option is not specified, such servers SHOULD use the "\NonExistent \HasChildren" attribute pair to signal to the client that there is a descendant mailbox that matches the selection criteria. See example 11 in Section 5.

The returned selection criteria allow the client to distinguish a solicited response from an unsolicited one, as well as to distinguish among solicited responses caused by multiple pipelined LIST commands that specify different criteria.

Servers SHOULD ONLY return a non-matching mailbox name along with CHILDINFO if at least one matching child is not also being returned. That is, servers SHOULD suppress redundant CHILDINFO responses.

Examples 8 and 10 in Section 5 demonstrate the difference between present CHILDINFO extended data item and the "\HasChildren" attribute.

The following table summarizes interaction between the "\NonExistent" attribute and CHILDINFO (the first collumn describes if the parent mailbox exists):

exists	meets the selection criteria	has a child that meets the selection criteria	returned LIST-EXTENDED attributes and CHILDINFO
no	no	no	no LIST response returned
yes	no	no	no LIST response returned
no	yes	no	(\NonExistent <attr>)
yes	yes	no	(<attr>)
no	no	yes	(\NonExistent) + CHILDINFO
yes	no	yes	() + CHILDINFO
no	yes	yes	(\NonExistent <attr>) + CHILDINFO
yes	yes	yes	(<attr>) + CHILDINFO

where <attr> is one or more attributes that correspond to the selection criteria, for example for the SUBSCRIBED option the <attr> is \Subscribed.

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 [CMbox]. 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.

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 MUST 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 MUST 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 any child mailbox. Note that even though the \HasChildren attribute for a mailbox must be correct at the time of processing of the mailbox, a client must be prepared to deal with a situation when a mailbox is marked with the \HasChildren attribute, but no child mailbox appears in the response to the LIST command. This might happen, for example, due to children mailboxes beig deleted or made inaccessible to the user (using access control) by another client before the server is able to list them.

\HasNoChildren

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

It is an error for the server to return both a \HasChildren and a \HasNoChildren attribute in the same 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

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
```

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"
S: * LIST (\Subscribed) "/" "Vegetable/Broccoli"
S: A02 OK done
```

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
```

4: In this example we see more mailboxes that reside on another server. This is similar to the command <RLIST "" "%">.

```
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
```

5: The following example also requests the server to include mailboxes that 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
```

6: The following example requests the server to include mailboxes that reside on another server. The server is asked 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
```

7: In the following example the client has specified multiple mailbox patterns. Note that this example does not 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
```

8: The following example demonstrates the difference between the \HasChildren attribute and the CHILDINFO extended data item.

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 this:

```
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 we get this result:

```
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 this:

```
C: C04 LIST (SUBSCRIBED RECURSIVEMATCH) "" "%"
S: * LIST () "/" "Foo" ("CHILDINFO" ("SUBSCRIBED"))
S: C04 OK done
```

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

A1) If the mailbox "Foo" had also been subscribed, the last command would return this:

```
C: C04 LIST (SUBSCRIBED RECURSIVEMATCH) "" "%"
S: * LIST (\Subscribed) "/" "Foo" ("CHILDINFO" ("SUBSCRIBED"))
S: C04 OK done
```

or even this:

```
C: C04 LIST (SUBSCRIBED RECURSIVEMATCH) "" "%"
S: * LIST (\Subscribed \HasChildren) "/" "Foo" ("CHILDINFO"
   ("SUBSCRIBED"))
S: C04 OK done
```

A2) If we assume instead that the mailbox "Foo" is not part of the original hierarchy and is not subscribed, the last command will give this result:

```
C: C04 LIST (SUBSCRIBED RECURSIVEMATCH) "" "%"
S: * LIST (\NonExistent) "/" "Foo" ("CHILDINFO" ("SUBSCRIBED"))
S: C04 OK done
```

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

C) And finally, suppose that only the mailboxes "Foo" and "Moo" are subscribed. In that case we see this result:

```
C: C04 LIST (SUBSCRIBED RECURSIVEMATCH) "" "%" RETURN (CHILDREN)
S: * LIST (\HasChildren \Subscribed) "/" "Foo"
S: * LIST (\HasNoChildren \Subscribed) "/" "Moo"
S: C04 OK done
```

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

9: The following example demonstrates that the CHILDINFO extended data item is returned whether children mailboxes match the canonical LIST pattern or not.

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 () "/" "qux2/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) "/" "qux2/bar2"
S: D02 OK done
```

The client issues the following command first:

```
C: D03 LIST (RECURSIVEMATCH SUBSCRIBED) "" "*2"
S: * LIST () "/" "foo2" ("CHILDINFO" ("SUBSCRIBED"))
S: * LIST (\Subscribed) "/" "foo2/bar2"
S: * LIST () "/" "baz2" ("CHILDINFO" ("SUBSCRIBED"))
S: * LIST (\Subscribed) "/" "baz2/bar2"
S: * LIST (\Subscribed) "/" "baz2/bar22"
S: * LIST (\Subscribed) "/" "baz2/bar222"
S: * LIST (\Subscribed) "/" "eps2" ("CHILDINFO" ("SUBSCRIBED"))
S: * LIST (\Subscribed) "/" "qux2/bar2"
S: D03 OK done
```

and the server may also include

```
S: * LIST (\NonExistent) "/" "qux2" ("CHILDINFO" ("SUBSCRIBED"))
```

The CHILDINFO extended data item is returned for mailboxes "foo2", "baz2" and "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 matches the pattern.

Note that if the client issues

```
C: D03 LIST (RECURSIVEMATCH SUBSCRIBED) "" "*"
S: * LIST () "/" "foo2" ("CHILDINFO" ("SUBSCRIBED"))
S: * LIST (\Subscribed) "/" "foo2/bar1"
S: * LIST (\Subscribed) "/" "foo2/bar2"
S: * LIST () "/" "baz2" ("CHILDINFO" ("SUBSCRIBED"))
S: * LIST (\Subscribed) "/" "baz2/bar2"
S: * LIST (\Subscribed) "/" "baz2/bar22"
S: * LIST (\Subscribed) "/" "baz2/bar222"
S: * LIST (\Subscribed) "/" "eps2" ("CHILDINFO" ("SUBSCRIBED"))
S: * LIST (\Subscribed) "/" "eps2/mamba"
S: * LIST (\Subscribed) "/" "qux2/bar2"
S: D03 OK done
```

the LIST responses for mailboxes "foo2", "baz2" and "eps2" still have the CHILDINFO extended data item, even though this information is redundant and the client can determine it by itself.

10: The following example shows usage of multiple mailbox patterns. It also demonstrates that the presence of the CHILDINFO extended data item doesn'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 RECURSIVEMATCH) "" foo RETURN (CHILDREN)
S: * LIST (\HasNoChildren) "/" foo ("CHILDINFO" ("SUBSCRIBED"))
S: a3 OK done
```

11: The following example shows how a server that supports missing mailbox hierarchy elements can signal to a client that didn't specify the RECURSIVEMATH selection option that there is a child mailbox that matches the selection criteria.

```
C: a1 LIST (REMOTE) "" *
S: * LIST () "/" music/rock
S: * LIST (\Remote) "/" also/jazz
S: a1 OK done

C: a2 LIST () "" %
S: * LIST (\NonExistent \HasChildren) "/" music
S: a2 OK done

C: a3 LIST (REMOTE) "" %
S: * LIST (\NonExistent \HasChildren) "/" music
S: * LIST (\NonExistent \HasChildren) "/" also
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]. In particular, note that the version of "mailbox-list" below, which defines the payload of the LIST response, updates the version defined in the IMAP specification. It is pointed to by "mailbox-data", which is defined in [IMAP4].

"vendor-token" is defined in [ACAP]. Note that this normative reference to ACAP will be an issue in moving this spec forward, since it introduces a dependency on ACAP. The definitions of "vendor-token" and of the IANA registry must eventually go somewhere else, in a document that can be moved forward on the standards track independently of ACAP.

```
childinfo-extended-item =  "CHILDINFO" SP "("
            list-select-base-opt-quoted
            *(SP list-select-base-opt-quoted) ")"
            ; Extended data item (mbox-list-extended-item)
            ; returned when the RECURSIVEMATCH
            ; selection option is specified.
            ; Note 1: the CHILDINFO tag can be returned
            ; with and without surrounding quotes, as per
            ; mbox-list-extended-item-tag production.
            ; Note 2: The selection options are always returned
            ; quoted, unlike their specification in
            ; the extended LIST command.


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


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


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


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

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


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


   list-select-base-opt-quoted =  DQUOTE list-select-base-opt DQUOTE


   list-select-independent-opt =  "REMOTE" / option-extension
               ; options that do not syntactically interact with
               ; other options


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


   list-select-opt =  list-select-base-opt / list-select-independent-opt
               / list-select-mod-opt
               ; An option registration template is described in
               ; Section 9.3 of this document.


   list-select-opts =  "(" [
                 (*(list-select-opt SP) list-select-base-opt
                  *(SP list-select-opt))
               / (list-select-independent-opt
                  *(SP list-select-independent-opt))
               ] ")"
               ; Any number of options may be in any order.
               ; If a list-select-mod-opt appears, then a
               ; list-select-base-opt must also appear.
               ; This allows these:
               ; ()
               ; (REMOTE)
               ; (SUBSCRIBED)
               ; (SUBSCRIBED REMOTE)
               ; (SUBSCRIBED RECURSIVEMATCH)
               ; (SUBSCRIBED REMOTE RECURSIVEMATCH)
               ; But does NOT allow these:
               ; (RECURSIVEMATCH)
               ; (REMOTE RECURSIVEMATCH)
```

```
   mailbox-list =  "(" [mbx-list-flags] ")" SP
               (DQUOTE QUOTED-CHAR DQUOTE / nil) SP mailbox
               [SP mbox-list-extended]
               ; This is the list information pointed to by the ABNF
               ; item "mailbox-data", which is defined in [IMAP4]


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


   mbox-list-extended-item =  mbox-list-extended-item-tag SP
               tagged-ext-val


   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 this
               ; document in Section 9.5.


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


   mbox-or-pat =  list-mailbox / patterns


   option-extension =  (option-standard-tag / option-vendor-tag)
               [SP option-value]


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


   option-val-comp =  astring /
               option-val-comp *(SP option-val-comp) /
               "(" option-val-comp ")"


   option-value =  "(" option-val-comp ")"
```

```
   option-vendor-tag =  vendor-token "-" atom
               ; a vendor specific option, non-standard


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


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


   tagged-ext-comp =  astring /
               tagged-ext-comp *(SP tagged-ext-comp) /
               "(" tagged-ext-comp ")"
               ; Extensions that follow this general
               ; syntax should use nstring instead of
               ; astring when appropriate in the context
               ; of the extension.
               ; Note that a message set or a "number"
               ; can always be represented as an "atom".
               ; An URL should be represented as
               ; a "quoted" string.

   tagged-ext-simple =  sequence-set / number

   tagged-ext-val =  tagged-ext-simple /
               "(" [tagged-ext-comp] ")"
```

7. Internationalization Considerations

The LIST command selection option types defined in this specification involve simple tests of mailbox properties. However, future extensions to LIST-EXTENDED may define selection options that do more sophisticated tests. In the case of a test that requires matching text, in the presence of the COMPARATOR [I18N] extension, the active comparator must be used to do comparisons. Such LIST-EXTENDED extensions MUST indicate in their specification the interaction with the COMPARATOR [I18N] extension.

8. 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 [MBRef].

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 any child mailbox. 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.

The CHILDINFO extended data item has the same security considerations as the \HasChildren attribute described above.

9. IANA Considerations

9.1. Guidelines for IANA

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

9.2. Registration procedure and Change control

Registration of a LIST-EXTENDED option is done by filling in the template in Section 9.3 and sending it via electronic mail to iana@iana.org. Registration of a LIST-EXTENDED extended data item is done by filling in the template in Section 9.5 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 LIST-EXTENDED 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 LIST-EXTENDED extended data item is returned as a result of requesting a particular LIST-EXTENDED option, the name of the option SHOULD be used as the name of the LIST-EXTENDED 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 LIST-EXTENDED option/extended data item names are case insensitive. If the vendor prefix is omitted from a vendor specific LIST-EXTENDED 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 LIST-EXTENDED 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. LIST-EXTENDED options/extended data items intended for widespread use should be standardized through the normal IETF process, when appropriate.

Comments on registered LIST-EXTENDED 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 LIST-EXTENDED options/ extended response data itself.

Once a LIST-EXTENDED 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 LIST-EXTENDED 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 LIST-EXTENDED 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.

LIST-EXTENDED 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 LIST-EXTENDED options/ extended data items will be clearly marked in the lists published by IANA.

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

9.3. Registration template for LIST-EXTENDED options

To: iana@iana.org
Subject: Registration of LIST-EXTENDED option X

LIST-EXTENDED option name:

LIST-EXTENDED option type: (One of SELECTION or RETURN)

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

LIST-EXTENDED option description:

Published specification (optional, recommended):

Security considerations:

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

Person and email address to contact for further information:

Owner/Change controller:

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

9.4. Initial LIST-EXTENDED option registrations

It is requested that the LIST-EXTENDED option registry be populated with the following entries:

1. To: iana@iana.org
   Subject: Registration of LIST-EXTENDED option SUBSCRIBED

   LIST-EXTENDED option name: SUBSCRIBED

   LIST-EXTENDED option type: SELECTION

   Implied return options(s): SUBSCRIBED

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

   Published specification : XXXX, Section 3.

   Security considerations: XXXX, Section 8.

   Intended usage: COMMON

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

   Owner/Change controller: iesg@ietf.org

2. To: iana@iana.org
   Subject: Registration of LIST-EXTENDED option REMOTE

   LIST-EXTENDED option name: REMOTE

   LIST-EXTENDED option type: SELECTION

   Implied return options(s): (none)

   LIST-EXTENDED option description: causes the LIST command to return remote mailboxes as well as local ones, as described in

RFC 2193.

Published specification : XXXX, Section 3.

Security considerations: XXXX, Section 8.

Intended usage: COMMON

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

Owner/Change controller: iesg@ietf.org

3. To: iana@iana.org
Subject: Registration of LIST-EXTENDED option SUBSCRIBED

LIST-EXTENDED option name: SUBSCRIBED

LIST-EXTENDED option type: RETURN

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

Published specification : XXXX, Section 3.

Security considerations: XXXX, Section 8.

Intended usage: COMMON

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

Owner/Change controller: iesg@ietf.org

4. To: iana@iana.org
Subject: Registration of LIST-EXTENDED option RECURSIVEMATCH

LIST-EXTENDED option name: RECURSIVEMATCH

LIST-EXTENDED option type: SELECTION

Implied return options(s): (none)

LIST-EXTENDED option description: Requests that CHILDINFO

extended data item (childinfo-extended-item) is to be returned.

Published specification : XXXX, Section 3.

Security considerations: XXXX, Section 8.

Intended usage: COMMON

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

Owner/Change controller: iesg@ietf.org

5. To: iana@iana.org
Subject: Registration of LIST-EXTENDED option CHILDREN

LIST-EXTENDED option name: CHILDREN

LIST-EXTENDED option type: RETURN

LIST-EXTENDED option description: Requests mailbox child information.

Published specification : XXXX, Section 3 and Section 4.

Security considerations: XXXX, Section 8.

Intended usage: COMMON

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

Owner/Change controller: iesg@ietf.org

9.5. Registration template for LIST-EXTENDED extended data item

To: iana@iana.org
Subject: Registration of X LIST-EXTENDED extended data item

LIST-EXTENDED extended data item tag:

LIST-EXTENDED extended data item description:

Which LIST-EXTENDED 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 and email address to contact for further information:

Owner/Change controller:

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

9.6. Initial LIST-EXTENDED extended data item registrations

It is requested that the LIST-EXTENDED extended data item registry be populated with the following entries:

1. To: iana@iana.org
   Subject: Registration of CHILDINFO LIST-EXTENDED extended data item

   LIST-EXTENDED extended data item tag: CHILDINFO

   LIST-EXTENDED extended data item description: The CHILDINFO extended data item describes the selection criteria that has caused it to be returned and indicates that the mailbox has one or more child mailbox that match the selection criteria.

   Which LIST-EXTENDED option(s) (and their types) causes this extended data item to be returned (if any): RECURSIVEMATCH selection option

   Published specification : XXXX, Section 3.5.

   Security considerations: XXXX, Section 8.

   Intended usage: COMMON

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

   Owner/Change controller: iesg@ietf.org

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 those discussions or made suggestions to this document.

11. References

11.1. Normative References

[ABNF] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

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

[I18N] Newman, C. and A. Gulbrandsen, "ACAP -- Application Configuration Access Protocol", draft-ietf-imapext-i18n (work in progress), February 2006.

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

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

[MBRef] Gahrns, M., "IMAP4 Mailbox Referrals", RFC 2193, September 1997.

11.2. informative References

[CMbox] Gahrns, M. and R. Cheng, "", RFC 3348, July 2002.

Authors' Addresses

Barry Leiba
IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532
US

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
URI: http://www.melnikov.ca/

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).