MIRAPOINT®

Developer's Reference (XML)

Release 4.0
December 2007
Part number 010-00551bc

This manual supports Messaging Operating System (MOS) release 3.9.0 and later MOS releases until replaced by a newer edition.

Macintosh is a trademark of Apple Computer, Inc.

Windows, Outlook, Exchange, and Active Directory are trademarks of Microsoft Corporation.

Java and Solaris are trademarks of Sun Microsystems, Inc.

Linux is a registered trademark of Linus Torvalds.

All other trademarks are the property of their respective owners.

OTHER THAN ANY EXPRESS LIMITED WARRANTIES THAT MIRAPOINT PROVIDES TO YOU IN WRITING, MIRAPOINT AND MIRAPOINT'S LICENSORS PROVIDE THE SOFTWARE TO YOU "AS IS" AND EXPRESSLY DISCLAIM ALL WARRANTIES AND/OR CONDITIONS, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MIRAPOINT'S LICENSORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY (INCLUDING NEGLIGENCE OR OTHER TORT), ARISING IN ANY WAY OUT OF YOUR USE OF THE SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF DAMAGES. Mirapoint's liability shall be as limited in the License Agreement.

MIRAPOINT SOFTWARE, INC. SOFTWARE LICENSE AGREEMENT

PLEASE READ THIS SOFTWARE LICENSE AGREEMENT ("LICENSE") CAREFULLY BEFORE DOWNLOADING OR OTHERWISE USING THE SOFTWARE. BY DOWNLOADING, INSTALLING OR USING THE SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS LICENSE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, YOU ARE NOT AUTHORIZED TO DOWNLOAD OR USE THIS SOFTWARE.

1. Scope. This License governs your use of any and all computer software, any printed or electronic documentation, or other code, whether on disk, in read only memory, or on any other media (collectively, the "Mirapoint Software") provided to you as part of or with a Mirapoint Product.

2. License, not Sale, of Mirapoint Software. The Mirapoint Software is licensed, not sold, to you by MIRAPOINT SOFTWARE, INC. or its affiliate, if any ("Mirapoint"). YOU MAY OWN THE MEDIA ON WHICH THE MIRAPOINT SOFTWARE IS PROVIDED, BUT MIRAPOINT AND/OR MIRAPOINT'S LICENSOR(S) RETAIN TITLE TO THE MIRAPOINT SOFTWARE. The Mirapoint Software installed on the Mirapoint Product and any copies which this License authorizes you to make are subject to this License.

3. Permitted Uses. This License allows you to use the pre-installed Mirapoint Software exclusively on the Mirapoint Product on which the Mirapoint Software has been installed. With respect to Mirapoint Software [identified by Mirapoint as the "administrative application" that has not been pre-installed on the Mirapoint Product, this License allows you to copy, use and install such Mirapoint Software on one or more administrative workstations on which the Mirapoint Software is supported. You may make one copy of the Mirapoint Software in machine-readable form for backup purposes only, provided that such backup copy must include all copyright and other proprietary information and notices contained on the original.

4. Proprietary Rights; Restrictions on Use. You acknowledge and agree that the Mirapoint Software is copyrighted and contains materials that is protected by copyright, trademark, trade secret and other laws and international treaty provisions relating to proprietary rights. You may not remove, deface or obscure any of Mirapoint's or its suppliers' proprietary rights notices on or in the Mirapoint Software or on output generated by the Mirapoint Software. Except as permitted by applicable law and this License, you may not copy, decompile, reverse engineer, disassemble, modify, rent, lease, loan, distribute, assign, transfer, or create derivative works from the Mirapoint Software. Your rights under this License will terminate automatically without notice from Mirapoint if you fail to comply with any term(s) of this License. You acknowledge and agree that any unauthorized use, transfer, sublicensing or disclosure of the Mirapoint Software may cause irreparable injury to Mirapoint, and under such circumstances, Mirapoint shall be entitled to equitable relief, without posting bond or other security, including but not limited to, preliminary and permanent injunctive relief.

5. Disclaimer of Warranty on Mirapoint Software. You expressly acknowledge and agree that use of the Mirapoint Software is at your sole risk. Unless Mirapoint otherwise provides an express warranty with respect to the Mirapoint Software, the Mirapoint Software is provided "AS IS" and without warranty of any kind and Mirapoint and Mirapoint's licensor(s) (for the purposes of provisions 5 and 6, Mirapoint and Mirapoint's

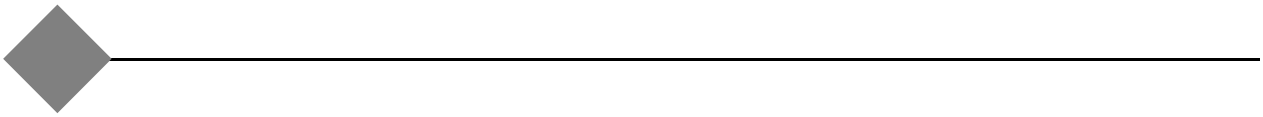
licensor(s) shall be collectively referred to as "Mirapoint") EXPRESSLY DISCLAIM ALL WARRANTIES AND/OR CONDITIONS, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN ADDITION, MIRAPOINT DOES NOT WARRANT THAT THE MIRAPOINT SOFTWARE WILL MEET YOUR REQUIREMENTS, OR THAT THE MIRAPOINT SOFTWARE WILL RUN UNINTERRUPTED OR BE ERROR-FREE, OR THAT DEFECTS IN THE MIRAPOINT SOFTWARE WILL BE CORRECTED. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES OR OTHER DISCLAIMERS, SO THE ABOVE EXCLUSION OR DISCLAIMERS MAY NOT APPLY TO YOU.

6. Limitation of Liability. UNDER NO CIRCUMSTANCES, INCLUDING NEGLIGENCE, SHALL MIRAPOINT BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO THIS LICENSE. FURTHER, IN NO EVENT SHALL MIRAPOINT'S LICENSORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES (INCLUDING BUT NOT LIMITED TO PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOSS OF USE, DATA OR PROFITS OR INTERRUPTION), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY (INCLUDING NEGLIGENCE OR OTHER TORT), ARISING IN ANY WAY OUT OF YOUR USE OF THE SOFTWARE OR THIS AGREEMENT, EVEN IF ADVISED OF THE POSSIBILITY OF DAMAGES. SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THIS LIMITATION MAY NOT APPLY TO YOU. In no event shall Mirapoint's total liability to you for all damages exceed the amount paid for this License to the Mirapoint Software.

7. Government End Users. If the Mirapoint Software is supplied to the United States Government, the Mirapoint Software and any documentation are provided with RESTRICTED RIGHTS. The Mirapoint Software is classified as "commercial computer software" and the documentation is classified as "commercial computer software documentation" or "commercial items," pursuant to DFAR Section 227.7202 or FAR Section 12.212, as applicable. Any use, modification, reproduction, display or disclosure of the Mirapoint Software or any documentation by the United States Government shall be governed by the terms of this License.

8. Miscellaneous. This License will be governed by and construed in accordance with the laws of the State of California, U.S.A., without reference to its conflict of law principles. If a court of competent jurisdiction finds any provision of this License invalid or unenforceable, that provision will be amended to achieve as nearly as possible the same economic effect as the original provision and the remainder of this License will remain in full force. Failure of a party to enforce any provision of this License shall not waive such provision or of the right to enforce such provision. This License sets forth the entire agreement between the parties with respect to your use of the Mirapoint Software and supersedes all prior or contemporaneous representations or understandings regarding such subject matter. No modification or amendment of this License will be binding unless in writing and signed by an authorized representative of Mirapoint. You will not export, reexport, divert, transfer or disclose, directly or indirectly, the Mirapoint Software, Mirapoint Products or any technical information and materials supplied under this Agreement without complying strictly with the export control laws and all legal requirements in the relevant jurisdiction, including without limitation, obtaining the prior approval of the U.S. Department of Commerce.

Contents



Tables

Preface

About This XML Guide

Mirapoint's Messaging Operating System (MOS) provides three XML interfaces for group calendar, email access, and address book. All communicate between server and client using XML encapsulated HTTP calls.

This book contains four chapters and an appendix:

- Chapter 1, Mirapoint XML Overview contains information about the different XML interfaces documented in this manual.
- Chapter 2, XML Interface to the WebCal Group Calendar for calendar.
- Chapter 3, XML Interface to the Mirapoint Message Base for email messages.
- Chapter 4, XML Interface to the WebMail Address Book for address book.
- Appendix A, Mirapoint DTDs about document types definitions (DTDs).

About Mirapoint Documentation

Documentation for all Mirapoint products is available through the Mirapoint Technical Library (MTL) on the Customer Support website:

`http://support.mirapoint.com/secure/MTL/MTL`

The MTL provides the hardware and software documentation for all supported Mirapoint releases and appliances, and the Support Knowledge Base. The Support site is accessible to all customers with a valid Support Contract. If your company has this but you need a Support login ID, email support-admin@mirapoint.com.

Getting Customer Support

If you experience problems with your system, contact Customer Support by e-mail or by telephone:

E-mail: support@mirapoint.com
Telephone: 1-877-MIRAPOINT (1-877-647-2764)

When contacting Customer Support, please be prepared with the following information about your system: MOS version (**Version** command in the CLI), the host ID (**License Hostid** command), serial number (**Model Get Serial** command), and hardware model (**Model Get Chassis** command, and label on the bezel).

Typographic Conventions

Table 1 explains what different fonts indicate in this book.

Table 1 Typefaces Used in This Book

Typeface	Use	Example
Regular	Ordinary text	The email server organizes mailboxes hierarchically.
Bold	Definitions	A **mail folder** is a container that stores messages.
Italic	Emphasis and titles	Specify *at least two* DNS servers. See the *Administrator's Guide*.
`Typewriter`	Screen display text and command names	`Enter your IP address:`
`Typewriter Bold`	Text that you must type exactly as shown	**`Dir Listdb`**
`Typewriter Italic`	Variables that you substitute and type	*`your_IP_address`*

1 Mirapoint XML Overview

Mirapoint provides several eXtensible Markup Language (XML) application programming interfaces (APIs) to allow tighter, customizable integration between Mirapoint appliances and other messaging systems, portals, or mobile devices.

These APIs can serve many functions, including calendar even synchronization, programmatic access to message and address-book data, extended user interface development, and more.

Mirapoint XML APIs Overview

Mirapoint provides XML APIs for:

- Personal and Group Calendar (including tasks)
- WebMail access to the message store
- Address Book

Calendar XML APIs

Mirapoint provides these 24 Calendar APIs, described in detail in Chapter 2, XML Interface to the WebCal Group Calendar.

Calendar APIs:

- Get/set calendar access permissions (default read/default write/read/write/free view/busy view/third-party scheduling)
- Find calendar users (matching a pattern)
- Get/set personal calendar subscriptions
- Check access permissions on a user's calendar
- View/modify a user's calendar
- Free/busy lookup on a user's calendar

Event/ToDo APIs:

- Create/update event
- Create/update to-do item
- Get events (all/by ID/by date range)

- Get to-do items (all/by ID)
- Get changed events/to-do items (by date range)
- Delete event (cancel/delete)
- Delete to-do item
- Purge calendar (by date range)

Other APIs:

- Login (simple/admin login on behalf of another user)
- Logout
- Search for patterns in events/todo items
- Get preferences
- Set preferences
- Import calendar
- Export calendar

WebMail XML APIs

Mirapoint provides these 16 WebMail APIs, described in detail in Chapter 3, XML Interface to the Mirapoint Message Base:

- Create a folder
- Delete a folder
- Get summary information (for a single folder or all folders)
- Compact a folder
- Get message headers (single, or by range of messages)
- Get content parts of a message (each part is a separate URL)
- Get original text message (RFC822 format)
- Set message flags
- Compose a new message
- Append a message to a folder
- Transfer a message to another folder
- Login
- Logout
- Search for a pattern in messages (by content values, message range, others)
- Get preferences
- Set preferences

Address Book XML APIs

Mirapoint provides these 17 Address Book APIs, described in detail in Chapter 4, XML Interface to the WebMail Address Book:

- Contacts operations (get/add/modify/delete)
- Groups of contacts operations (get/add/modify/delete)
- Get categories
- Search for patterns
- Get preferences
- Set preferences
- Import address book
- Export address book
- Others

HTTP Operations

Each XML document is retrieved through HTTP using either the HTTP 1.0 or 1.1 protocol. The information retrieved in each XML document is computed based on the path and query components of the URL. The HTTP GET operation is used to fetch documents, and the HTTP POST operation is used to perform actions.

With GET operations, all arguments are passed in the query component of the URL. For POST operations, if the argument list is short, it can also be passed in the query string. However, for longer argument lists, arguments may have to be passed in the content of the POST operation, either in a URL-encoded way, or for arguments that contain large values, in a MIME-encoded method.

XML API Conventions

In this manual, when specifying a particular GET or POST operation, both the value of the path component and the query component are described as follows:

Table 2 GET/POST Operation Convention

Operation	`GET /mc/xcal/v2/getevents.xml`	
Arguments	`eventid`	ID of the event
	`sid`	user's session ID
Result	`ok` element	
	`no` element	

The corresponding HTTP is:

```
C: GET /mc/xcal/v2/getevents.xml&eventid=123&sid=27 HTTP/1.1
C: Host: host.domain.com
C: Connection: Keep-Alive
```

```
C:
S: HTTP/1.1 200 OK
S: Date: Mon, 18 Sep 2000 11:19:24 GMT
S: Content-Type: text/xml; charset=utf-8
S:
S: <?xml version="1.0"?>
S: ...
```

Table 3 Example: POST Operation

Operation	POST /mc/xcal/v2/login.xml	
Arguments	user	user login name
	password	user's password

```
C: POST /mc/xcal/v2/login.xml HTTP/1.1
C: Host: host.domain.com
C: Connection: Keep-Alive
C: Content-Length: 85
C:
S: HTTP/1.1 100 Continue
S:
C: user=juser&password=secret99
S: HTTP/1.1 200 OK
S: ...
```

Argument Modifiers

In some commands there might be optional arguments, or there might be one of two different arguments necessary, but not both. The XML modifiers for one or more (+), zero or more (*), and zero or one (?) times are used to specify the nature of the argument. For an argument that has an OR relationship with another argument, (|) is used to signify that property. No modifier means exactly one instance of the argument is necessary.

Argument Character Encoding

All input arguments are sent using utf-8 character encoding. When arguments are sent in the query component of a URL, they must be additionally URL encoded.

Data entities that are exported by the Mirapoint server and that also might be uploaded into the server are XML encoded in the outgoing direction. They must be URL encoded by the client in the incoming direction if they are part of a URL. The client is responsible for decoding the XML encoding and performing the necessary encoding when returning the data to the server (mime multipart is recommended). For example, an email address of the form:

```
John Smith <jsmith@foobar.com>
```

appears in the XML output as:

```
John Smith &lt;jsmith@foobar.com&gt;
```

which an XML parser receiving the XML document converts back to its original form. When the client wants to give this data, it must use the first form, URL encoded if necessary.

Global XML Response Elements

This section describes the XML status element, which occurs in multiple places in these interfaces. Other XML elements are described throughout this manual in the context of their use.

status Response Elements

Some responses can occur in many GET or POST operations, including an OK response when an action is performed as specified, or a NO response when an action could not be performed. All such responses are part of a success element: `ok`, `no`, or `bad` element. Responses are localized to the locale specified during login, or the system's default locale if the locale parameter is missing.

<ok> Element

An <ok> element is included in a response to indicate that the request was successful.

Syntax

```
<!ELEMENT ok (#PCDATA)>
```

Example

```
<ok>Delete Completed</ok>
```

The <ok> element is in GMT, for example:

```
<ok>20040822T001213Z</ok>
```

<no> Element

A <no> response to an action occurs when the server could not complete the operation that the client requested.

Syntax

```
<!ELEMENT no (#PCDATA)>
```

Example

A <no> response is returned if an incorrect HTTP operation is sent, for example one of:

```
<no>HTTP GET operation not allowed</no>
<no>HTTP POST operation not allowed</no>
```

The following <no> response is returned whenever an invalid or expired session ID is passed:

```
<no>Your session has timed out</no>
```

A complete response would look as follows:

```
<?xml version="1.0"?>
<!DOCTYPE no SYSTEM "http://qa150.example.com/dtd/xml/v1/no.dtd">
```

```
<no>Authentication failed</no>
```

<bad> Element

The <bad> element is used to indicate a badly formed or illegal request.

Syntax

```
<!ELEMENT bad (#PCDATA)>
```

Example

```
<bad>System I/O Error</bad>
```

2 XML Interface to the WebCal Group Calendar

This chapter defines an XML interface to Mirapoint's WebCal group calendaring data store.

This interface provides a means of writing customized calendaring user-interface applications, using Mirapoint for calendar stores and optionally calendar email and mobile device reminders. For web applications, Mirapoint recommends using the native Mirapoint interface, for reasons of performance and ease of development.

The interface can also be used by synchronization software to synchronize the Mirapoint calendar store with other calendar applications.

The interface provides functions to:

- Add, delete, and modify events and to-do items
- Get to-do items
- Get events within a given date range
- Search for events and to-do items
- Import and export vCalendar
- Set and get user preferences
- Check for changes to records
- Subscribe to other calendars
- View other users' calendars
- Get free/busy data for users

Date and Time Representation

Times and dates are represented in the ISO-8601 short format. They are always relative to GMT unless otherwise stated. The maximum supported time is 23:11, 31st Dec 2035 and the minimum supported time is 00:01, Jan 1 1970.

Two formats of ISO-8601 strings are used, for absolute time and relative time. Unless otherwise stated, time fields are absolute time relative to GMT and must be in the format:

date "T" *time* "Z"

- *date* is of the form YYYYMMDD; that is, a four- digit year value, followed by a two-digit month value (1 through 12), followed by a two-digit day value (1 through 31, as permitted by the month).
- *time* is of the form HHMMSS; that is, a two-digit hour value followed by a two-digit minute value, followed by a two-digit second value.

Durations of time (periods) are expressed in the form:

```
"P" date-comp "T" time-comp
```

- *date-comp* is an optional list of date components.
- *time-comp* is an optional list of time components.

Both the date and time components are a decimal number followed by a letter indicating the units: "Y" for years, "M" for months, "D" for days, "W" for weeks, "H" for hours, "M" for minutes, and "S" for seconds.

For example, a duration of 2 hours and 15 minutes would be represented as:

```
PT2H15M
```

A duration of 2 weeks would be represented as:

```
P2W
```

Deleting Entries

All POST operations, except for viewother.xml, either create or modify calendar information. Once entries are set, they can be modified by setting new values. To delete all entries already set, set the data with an empty value.

For example, to set permissions, issue the following HTTP request:

```
/mc/xcal/v2/permissions.xml
```

with the following arguments, to give joe and ed read-access to your calendar:

```
cal_rdaccess_type=INCLUDE&cal_rdaccess_user=joe&cal_rdaccess_type=e
```

To remove ed, give the following arguments:

```
cal_rdaccess_type=INCLUDE&cal_rdaccess_user=joe
```

To delete all entries, give the following arguments:

```
cal_rdaccess_type=INCLUDE&cal_rdaccess_user=
```

Repeating Events

Repeating events are stored as a single event with an option for how it repeats. This rule is the "eventrrule" in the event structure (see “getevents Command” on page 29). When using this rule to calculate all the instances of a repeating event, the calculation must be performed in the timezone the event was created in.

For example, Rocky in America/Los Angeles invites Natasha in Europe/Moscow to a meeting at 5pm every Monday starting on November 11, 2006. For Natasha this is 4am November 12th in her local time. If she tries to calculate the next Monday in her local time, she gets 4am November 17th, which is incorrect. If instead she calculates in GMT and asks Rocky to do the same, they both get the same time but

all instances after the first would be on Tuesdays. If Natasha decides instead of calculating the next Monday to instead simply add 7 days (168 hours) to the start of the first instance, she gets the incorrect time the next time a daylight savings change occurs. However, if Natasha first converts the event time to Rocky's timezone (GMT start time minus 8 hours), calculates the next instances, and converts those times back to her timezone, she always gets the correct result.

A repeating event can have exceptions, which are exported through the "exceptions" option (see “getevents Command” on page 29). This option contains a list of dates in GMT. The instances on these dates are exceptions to the repeating rule. Individual events can have the "exceptiondate" field which provides the parent event uid and the date the event was originally going to occur on. This can be used to match exceptions with their parent event.

Calendaring Operations Summary

Mirapoint’s XML Group Calendaring interface provides the following operations:

Table 4 XML Group Calendaring Operations

#	Description	HTTP	Command	Result Document
1	login	POST	login.xml	<login>
2	administrative login on behalf of another user	POST	login.xml	<login>
3	logout	GET	login.xml	<status>
4	get events within a date range	GET	getevents.xml	<calendar>
5	get all to-do items	GET	gettodos.xml	<calendar>
6	get IDs of events and to-do items that have changed within a specified time window	GET	getchanges.xml	<changes>
7	get events and todos based on a search string	GET	search.xml	<calendar>
8	add/modify an event	POST	updateevent.xml	<update>
9	add/modify a to-do item	POST	updatetodo.xml	<update>
10	delete an event	POST	deleteevent.xml	<status>
11	delete a to-do item	POST	deletetodo.xml	<status>
12	delete events and todos within a date range	POST	deletedaterange.xml	<status>
13	get user preferences	GET	prefs.xml	<preferences>
14	set user preferences	POST	prefs.xml	<status>
15	get calendar access permissions	GET	permissions.xml	<permissions>
16	set calendar access permissions	POST	permissions.xml	<status>

Table 4 XML Group Calendaring Operations

#	Description	HTTP	Command	Result Document
17	get calendar subscriptions	GET	subscriptions.xml	<userlist>
18	set calendar subscriptions	POST	subscriptions.xml	<status>
19	view another user's calendar	POST	viewother.xml	<viewother>
20	find calendar user using LDAP	GET	finduser.xml	<userlist>
21	check calendar access permissions to see if a user is allowed to perform a specified operation	GET	checkperms.xml	<status>
22	perform free/busy lookups on a user's calendar	GET	freebusy.xml	<free-busyreply>
23	import in vCalendar format	POST	vcalimport.xml	<status>
24	export in vCalendar format	GET	vcalexport.xml	text/x-vCalendar MIME type
25	accept or decline a calendar event	POST	changepartstat.xml	<status>

DTD Summary

Different types of DTDs are used for different operations, which lets clients leverage XML parsing to verify that replies contain the correct type of information for the request that generated them.

Table 5 DTDs Used in XML Group Calendaring

DTD	Usage
`calendar`	In replies to calendar queries for events and to-do items.
`changes`	To retrieve list of events and to-dos which were changed or deleted.
`freebusyreply`	In replies requesting free-busy information.
`login`	In reply to login requests.
`permissions`	In replies requesting user permissions.
`preferences`	In replies requesting user preferences. Other preferences are returned in `permissions` and `subscriptions`.
`status`	For operations that just succeed or fail.
`update`	For replies to update requests on events and to-do items.
`userlist`	In replies requesting lists of users.
`viewother`	In replies to request viewing another calendar.

Each DTD is described where it is first used in this chapter.

Commands

This section describes the commands available in the Group Calendar XML interface.

changepartstat Command

The changepartstat command is used by clients to accept or decline a calendar event.

Table 6 changepartstat Command Definition

Operation	`POST /mc/xcal/v2/changepartstat.vcs`	
Arguments	`sid`	session ID
	`eventid`	ID of the event
	`stat`	This field can take the values `accept` or `decline`.
	`sendmail`	Specifies if email should be sent to notify attendees. Valid values are y and n.
	`instance?`	Specifies if one or all instances of a repeating event should be updated. Valid values are `all` and `one`. For non-repeating events, use the value `one` or do not specify this field.
Result	<status> document	

The request is sent via an HTTP POST message.

changepartstat Example

The response is a status response. See “status Response Elements” on page 17.

checkperms Command

The checkperms command lets a client test if a user is allowed to perform a particular operation on another user's calendar. An HTTP GET requests the information for a particular user and operation and a `status` doctype returns the result.

Table 7 checkperms Command Definition

Operation	`GET /mc/xcal/v2/checkperms.xml`	
Arguments	`sid`	session ID
	`user`	The login ID of the user on whose calendar the requester wishes to perform the operation.
	`scope`	The type of operation to be performed. The values of scope allowed are the same as defined in section 4.3.
Result	`<status>` doctype. <ok> means the operation is allowed, <no> means it is not, and <bad> means some input error.	

checkperms Example

The response is a status response. See “status Response Elements” on page 17.

deletedaterange Command

The deletedaterange command deletes all events within a date range.

Table 8 deletedaterange Command Definition

Operation	`POST /mc/xcal/v2/deletedaterange.xml`	
Arguments	`sid`	session ID
	`lowerbound`	start of window in ISO-8601.
	`upperbound`	end of window in ISO-8601.
	`x-mira-agent?`	(optional) A text string that identifies the client software. This string is not stored with the event record, but may be used for logging. Maximum length is 63 characters.
Result	<status> doctype	

Both `lowerbound` and `upperbound` can be `"*"`, which implies no bound. Therefore, setting both `lowerbound` and `upperbound` to `"*"` deletes the entire calendar.

deletedaterange Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE status SYSTEM "http://mail.example.com/dtd/xcal/v2/status.dtd”>
<status>
<ok>OK</ok>
</status>
```

deleteevent Command

The deleteevent command deletes an existing event from the calendar store.

Table 9 deleteevent Command Definition

Operation	`POST /mc/xcal/v2/deleteevent.xml`

Table 9 deleteevent Command Definition

Arguments	sid	session ID
	eventid	event ID
	hard?	If present and equal to "y", any events that would otherwise be moved into the "canceled" state (for group events) are deleted.
	instance?	This field applies only to repeating events. It is ignored for non-repeating events. It can have the value all or one.
	sendmail?	This field specifies if the cancel email messages should be sent to internal attendees. It can have values "y" or "n" (defaults to "y").
	extsendmail?	This field specifies if the cancel email messages should be sent to external attendees. It can have values "y" or "n" (defaults to "y").
	x-mira-agent?	(optional) A text string that identifies the client software. This string is not stored with the event record, but may be used for logging. Maximum length is 63 characters.
Result	<status> doctype	

deleteevent Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE status SYSTEM "http://mail.example.com/dtd/xcal/v2/status.dtd”>
<status>
<ok>
20040821T005101Z</ok>
</status>
```

deletetodo Command

The deletetodo command deletes an existing to-do item from the calendar store.

Table 10 deletetodo Command Definition

Operation	POST /mc/xcal/v2/deletetodo.xml	
Arguments	sid	session ID
	todoid	todo ID
	x-mira-agent?	(optional) A text string that identifies the client software. This string is not stored with the event record, but can be used for logging. Maximum length is 63 characters.
Result	<status> doctype	

deletetodo Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE status SYSTEM "http://mail.example.com/dtd/xcal/v2/status.dtd”>
<status>
<ok>OK</ok>
</status>
```

finduser Command

The finduser command lets client user software use the same user-search functionality that is available in the native Mirapoint web interface. The client provides a search string and the server performs a local search and if necessary and configured, an LDAP search for matching users.

Results are returned in a `userlist` doctype (see “subscriptions Command” on page 54). The maximum number of users returned is capped at 100 entries.

Table 11 finduser Command Definition

Operation	`GET /mc/xcal/v2/finduser.xml`	
Arguments	`sid`	session ID
	`pattern`	The search string. It must contain between 1 and 255 characters.
	`fqdn`	"yes" to fully qualify all userIDs in the top level in the users returned, "no defaults to "no".
	`urlinstance`	The URL instance to use for lookups (see the admin URL command); the used in case urlinstance parameter is not specified. The string can be 1 through 255 characters. `urlinstance` can be any filter listed by the Url List command and is represented by the class:instance name as defined in Help About Url; for example, urlinstance=groupcalendar:userlookup.
Result	`<userlist>` doctype containing a list of any matching users, together with the login ID when further referencing the users.	

finduser Example

The response is similar to the “subscriptions Example” on page 55.

freebusy Command

The freebusy command is used to request free busy information for a user's calendar.

Table 12 freebusy Command Definition

Operation	`GET /mc/xcal/v2/freebusy.xml`

Table 12 freebusy Command Definition

Arguments	`sid`	session ID
	`user`	Login ID of the user whose free-busy information is being requested.
	`dtstart`	Start of the lookup window, in ISO-8601, GMT.
	`dtend`	Stop of the lookup window, in ISO-8601, GMT.
Result	`freebusyreply` doctype containing a list of <freebusy> elements for that user.	

Table 13 freebusy DTD Fields

Name	Type	Max Length (Chars)	Default	Occurences	Description
`dtstamp`	required	255	N/A	1	The time that the element was created (that is, shortly after the lookup request was sent), in GMT.
`dtstart`	required	255	requested value	1	The time of the start of the free-busy window (that is, the value of `dtstart` provided by the client software when it sent the request), in GMT.
`dtend`	required	255	requested value	1	The time of the end of the free-busy window (that is, the value of `dtend` provided by the client software when it sent the request), in GMT.
`freebusy`	required	255	N/A	0, 1, or more	A string representing a busy period. The string is two ISO-8601 times separated by a slash ('/') character, with no spaces. The first time is the start of a busy period, the second is the end of that busy period.

freebusy Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE freebusyreply SYSTEM "http://mail.example.com/dtd/xcal/v2/
    freebusy.dtd”>
<freebusyreply>
    <ok>OK</ok>
    <vfreebusy>
    . <dtstamp>20020620T130120Z</dtstamp>
    . <dtstart>20000101T000000Z</dtstart>
    . <dtend>20101231T235900Z</dtend>
    . <freebusy>20020325T090000Z/20020325T100000Z</freebusy>
    </vfreebusy>
    <vfreebusy>
    . <dtstamp>20020620T130120Z</dtstamp>
    . <dtstart>20000101T000000Z</dtstart>
    . <dtend>20101231T235900Z</dtend>
    . <freebusy>20020416T090000Z/20020416T100000Z</freebusy>
    </vfreebusy>
    <vfreebusy>
```

```
    . <dtstamp>20020620T130120Z</dtstamp>
    . <dtstart>20000101T000000Z</dtstart>
    . <dtend>20101231T235900Z</dtend>
    . <freebusy>20020417T090000Z/20020417T100000Z</freebusy>
    </vfreebusy>
    <vfreebusy>
    . <dtstamp>20020620T130120Z</dtstamp>
    . <dtstart>20000101T000000Z</dtstart>
    . <dtend>20101231T235900Z</dtend>
    . <freebusy>20020501T090000Z/20020501T100000Z</freebusy>
    </vfreebusy>
</freebusyreply>
```

getchanges Command

The getchanges command returns a list of events and to-do items that have changed within a given time window. Fewer elements, such as the <created>, <dtstamp>, <last-modified> and <sequence> elements, are included in the <event> and <todo> elements returned.

Table 14 getchanges Command Definition

Operations	`GET /mc/xcal/v2/getchanges.xml`	
Arguments	`sid`	session ID
	`lowerbound`	start of window in ISO-8601 format
	`upperbound`	end of window in ISO-8601 format
	`fqdn`	"yes" to fully qualify all userIDs in the top level in the events returned, "no" to not. This command defaults to "no".
	`includeprivate`	"yes" to include private events in the results when viewing another user's calendar. By default, private events are not included in the result if the calendar for the session is not the same as the user that created the session (that is, the sessionid was obtained from a viewother.xml request).
	type	(optional): "*", "event" or "todo" to return all changed items, events or todos respectively. Default type is both ("*")
Result	A <changes> element containing 0 or more <changedevent> and <changedtodo> elements.	

Calendar record changes, are described using the <changes> doctype.

getchanges Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE changes SYSTEM "http://mail.example.com/dtd/xcal/v2/changes.dtd”>
<changes>
    <ok>OK</ok>
    <changedevent>
    . <eventid>e102822000</eventid>
```

```
    . <created>20020620T103400Z</created>
    . <dtstamp>20020620T124740Z</dtstamp>
    . <last-modified>20020620T112000Z</last-modified>
    . <sequence>17</sequence>
    </changedevent>
    <changedevent>
    . <eventid>e102822001</eventid>
    . <created>20020620T103400Z</created>
    . <dtstamp>20020620T124740Z</dtstamp>
    . <last-modified>20020620T110500Z</last-modified>
    . <sequence>12</sequence>
    </changedevent>
    <changedevent>
    . <eventid>e102a58400</eventid>
    . <created>20020620T101200Z</created>
    . <dtstamp>20020620T124740Z</dtstamp>
    . <last-modified>20020620T102900Z</last-modified>
    . <sequence>7</sequence>
    . <x-mira-deleted/>
    </changedevent>
    <changedtodo>
    . <todoid>t104901e00</todoid>
    . <created>20020620T121400Z</created>
    . <dtstamp>20020620T124740Z</dtstamp>
    . <last-modified>20020620T121400Z</last-modified>
    . <sequence>2</sequence>
    </changedtodo>
</changes>
```

getevents Command

The getevents command returns a list of events, or instances of repeating events, that fall within a specified date range, or that match the `eventid`. `lowerbound` and `upperbound` can both be set to `"*"`, which implies no bound to that end of the range. The request must contain either a specified `lowerbound` and `upperbound` variables, `eventid`, or both. The server returns a calendar element containing a list of all the events found (see the getevents DTD in Appendix A, Mirapoint DTDs).

If `eventid` is not specified, all events within `lowerbound` and `upperbound` are returned. Repeating event instances (including exceptions) are always specified as `<parenteventid>-<instancetime>`.

The `eventid` accepts IDs from third party calendar agents in the following forms:

```
clientid:<clientid-spec>
clientid:<clientid-spec>-<exceptiondate>
```

The first form references non-repeating events or whole series of repeating events. The second form references instances (and exceptions) of repeating events, where `<clientid-spec>` is the client ID of the repeating series parent. (Outlook uses the same client ID for parent and exceptions of repeating series). The <exceptiondate> is an ISO-8601 string that references an instance of event, as for regular Mirapoint `eventid`. The `<clientid-spec>` may be of two forms:

```
<clientid>
<clientid>:<apptid>
```

The `<apptid>` is stored in event objects and returned in the `<clientid>` element of replies to getevents.xml requests. It is ignored for event indexing though. That is, the `clientid` key used to index events is the string preceeding the colon (:) only.

When a client creates an exception to a repeating series, it may use the `clientid` form with an exception to specify the `clientid` referenced. If it uses the `clientid` form of `eventid`, the same request must not contain a `clientid` argument.

Table 15 getevents Command Definition

Operation	`GET /mc/xcal/v2/getevents.xml`	
Arguments	`sid`	session ID
	`lowerbound`	start of window in ISO-8601
	`upperbound`	end of window in ISO-8601
	`expand`	expand repeating events. Valid values are y and n. Default is n.
	`eventid?`	event ID of an event.
	`fqdn`	"yes" to fully qualify all userIDs in the top level in the events returned, "no" to not. This command defaults to "no".
	`includeprivate`	"yes" to include private events in the results when viewing another user's calendar. By default, private events are not included in the result if the calendar for the session is not the same as the user that created the session (that is, the sessionid was obtained from a viewother.xml request).
Result	A <calendar> element containing 0 or more <event> elements.	

Calendar events, or instances of repeating events are described in responses using the `event' element in a <calendar> document.

The elements of event elements are described below. There are three types of elements:

- Required elements are always present in the <event> or <todo> element.
- Optional elements appear only if they have had a value assigned to them.
- Flag elements indicate state by their presence or absence. That is, if they are present, their value is TRUE, and otherwise false. They cannot have an "undefined" value.

Table 16 on page 31 describes the fields. It includes maximum field lengths, where applicable, and default values. Defaults are the values that the Mirapoint calendar store assigns to an event or to-do item if the client specifies no value for that field.

- A default value of "n/a" means that either there is no concept of default, because the event/to-do item cannot be created without specifying a value, or that the value has no fixed default, but is always created by the system as a function of some algorithm.
- A default value of "undefined" means that the field is optional. If no value has been assigned to it, it remains in an "undefined" state and is not exported.

The "Occurrences" field indicates how many times the element might appear in an <event> or <todo> element.

All fields are XML encoded. For example, an event title of "Hi <there>" is exported as "Hi <there>".

Table 16 getevents DTD Fields

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
`eventid`	required	255	n/a	1	XML-encoded ID of the event. The event ID, assigned by the Mirapoint calendar store, is an ASCII string that uniquely identifies the event within a user's calendar.
`clientid`	optional	1024	n/a	1	Client ID of an event, if it exists.
`globalid`	required	1024	n/a	1	Globalid (server) ID of an event.
`eventtitle`	required	255	n/a	1	Title of the event. Cannot be an empty string.
`eventdesc`	optional	65531	n/a	0 or 1	Description of the event.
`eventstart`	required	255	n/a	1	Start time/date of the event, in ISO-8601 format (see "getevents Example" on page 38). For repeating events, the start time/date of the event's first instance.
`eventstop`	required	255	n/a	1	Stop time of the event, in ISO-8601 format (see "getevents Example" on page 38). For repeating events, the stop time/date of the event's first instance.
`viewablestart`	required	255	n/a	1	Date/time when an instance of the event first becomes "viewable" in a time window. For example, if an event starts at 14:00, but the `lowerbound` parameter passed to the `getevents` command starts at 15:00, the viewable start is 15:00. If a repeating event has more than one instance in the window, this value is the viewable start of the first instance that has some part of it within the window.

Table 16 getevents DTD Fields

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
`viewablestop`	required	255	n/a	1	Latest date/time within the specified time window that the first instance of the event is still viewable within the window. If an event starts at 12:00, ends at 16:00, but the `upperbound` passed to the `getevents` command is 13:00, the viewable stop is 13:00. If a repeating event has more than one instance in the window, this value is the viewable stop time of the first instance that has some part of it within the window.
`allday`	flag	n/a	FALSE	0 or 1	Present if the event is an "all-day" event ("transparent event"); one that has only a date associated with it. It does not appear as "busy" in free-busy requests. If a time window is provided that overlaps with another day, a new time is substituted starting at 7:00 AM in that timezone.
`eventpriority`	required	1	3	1	Event priority. Valid values: 1 through 5, with 1 being the highest.
`eventemaildiff`	required	4	obtained from the calendar defaults (see "prefs Command" on page 47)	1	Minutes before the event occurs to send an email reminder. Valid values: -1 (no email reminder) through 70555 minutes, in five minute increments: 5, 10, 15, ..., 70555.
`eventpagerdiff`	required	4	default is obtained from the calendar defaults (see "prefs Command" on page 47)	1	Minutes before the event occurs to send a pager email reminder. Valid values: -1 (no pager email reminder) through 70555 minutes, in five minute increments: 5, 10, 15, ..., 70555.

Table 16 getevents DTD Fields

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
`eventnotifylist`	optional	n/a	undefined	0 or 1	Email addresses to which to send notification emails when the event is created or modified.
`eventnotifyelement`	optional	total length of all email addresses cannot exceed 65531 bytes.	undefined	0, 1, or more	One email address used in `eventnotifylist`.
`eventrrule`	optional	512	undefined	0 or 1	A vCalendar 1.0 RRULE specification of a repeating event. Unless otherwise specified, events do not repeat.
`eventxdata`	optional	65531	undefined	0 or 1	General-purpose string that lets client software store its own data in an event item. The Mirapoint calendar store does not alter this field's contents.
`attendees`	optional	n/a	n/a	n/a	Attendees to a meeting.
`attendee`	optional	255	undefined	0, 1, or more	Meeting attendee. Must be references as an RFC-822 compliant form of the email address on the attendee's Mirapoint mail store. The `role` attribute of the attendee must be "CHAIR" or "REQ-PARTICIPANT". The `partstat` attribute of the attendee may be "NEEDS-ACTION", "ACCEPTED", "DECLINED", or "TENTATIVE" status. The TST attribute represents Outlook's TrackStatusTime property, used by event owners to guarantee that accept/decline emails are not read out of order.

Table 16 getevents DTD Fields

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
`extattendees`	optional	n/a	n/a	n/a	External attendees to a meeting.
`extattendee`	optional	255	undefined	0, 1, or more	External attendee to a meeting. Must be references as an RFC-822 compliant form of the email address. The `partstat` attribute of the attendee may be "NEEDS-ACTION", "ACCEPTED", "DECLINED", or “TENTATIVE”.
`resources`	optional	n/a	n/a	n/a	Resources (such as conference rooms) assigned to a meeting.
`resource`	optional	255	undefined	0, 1, or more	Resource required for a meeting (for example, projector, meeting room). Must be references as an RFC-2822 (a revised RFC-822) compliant form of the email address on the Mirapoint mail store of that resource user. (Resource calendars are created as regular users in the Mirapoint calendar server). The `partstat` attribute of the attendee may be "NEEDS-ACTION", "ACCEPTED", "DECLINED", or “TENTATIVE”.
`categories`	optional	n/a	n/a	n/a	Categories (such as personal, business, ...) assigned to a meeting.
`category`	optional	255	undefined	0, 1, or more	Category associated with a meeting.

Table 16 getevents DTD Fields

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
`created`	required	255	time the record was created required	1	Time the event/to-do record was created, in GMT, either imported via the XML interface, or the admind vCalendar interface. If undefined, defaults to the time when the record was created in the Mirapoint calendar store.
`dtstamp`	required	255	n/a	1	Time the element was created (that is, shortly after the lookup request was sent), in GMT.
`last-modified`	required	255	time the record was last modified	1	Time the event/to-do record was last modified, in GMT. Importing the event via the XML interface or the admind vCalendar interface always counts as a modification.
`sequence`	required	255	0 or previous value plus 1	1	Sequence number associated with the last change in the Mirapoint calendar store. If not specified when importing and the record already exists in the calendar store, the new sequence number is that of the existing record plus 1. If not specified when importing and the event does not already exist in the calendar store, it defaults to zero. This value can increase by more than 1 depending on how many internal operations are required to fulfill a specific request.

Table 16 getevents DTD Fields

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
`status`	required	255	undefined	0 or 1	Case-insensitive text string indicating the status of the meeting. The `status` element is present only when the event or to-do item is canceled, in which case it adopts a value of "CANCELLED". Defaults to an undefined value, which implies that the event is not canceled.
`x-mira-sendmail`	flag	n/a	TRUE	0 or 1	If present, the event is in a state where if it is changed, the Mirapoint calendar server sends change notification emails to all participants. If not present, emails are not sent.
`x-mira-deleted`	flag	n/a	FALSE	0 or 1	If present, the event was deleted from the calendar store. When present, only the `eventid`, `created`, `dtstamp`, `last-modified`, and `sequence` elements are present.
`x-mira-readonly`	flag	n/a	FALSE	0 or 1	If present, the event cannot be changed. Occurs when viewing another user's calendar or when viewing an event that comes from another calendar to which the user is subscribed.
`attach`	optional	n/a	undefined	0 or 1	An attachment to the event. Multiple attachments are allowed. The `attach` element contains only an `extref` element, which references the attachment as an entity defined in the internal DTD.
`extref`	optional	n/a	undefined	0, 1, or more	Appears only in an `attach` element. It has an "uri" attribute that references an XML entity that points to the attachment.

Table 16 getevents DTD Fields

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
`exceptions`	optional	n/a	n/a	n/a	Exceptions to a repeating event.
`exception`	optional	255	undefined	0, 1, or more	Time/date of an exception for this repeating event. Must be in ISO-8601 format (see "getevents Example" on page 38).
`timezoneoffset`	optional	255	0	0 or 1	Offset from GMT of the timezone in which the event was created. Represented as a relative time (see "Date and Time Representation" on page 19) with a minus sign prepended for offsets before GMT (for example,Los Angeles "-PT8H00M").
`timezone`	optional	255	Etc/GMT	0 or 1	Timezone the event was created in (for a list of current time zones, use the CLI **Ntp Get Knownzones** command.)
`sequence-init`	flag	255	n/a	0 or 1	If a prior updateevent.xml request specified a sequenceinit property, then it is returned here. The server does not modify this. If a sequenceinit was never specified for an event, then this element is not returned.
`client-lastmodified`	flag	255	n/a	0 or 1	If a prior updateevent.xml request specified a clientlastmodified property, then it is returned here. The server does not modify this. If a clientlastmodified was never specified, then this element is not returned. For Schedulemode Email (NFO) the server may modify this to preserve schedule semantics.

Table 16 getevents DTD Fields

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
replytime	flag	255	n/a	0 or 1	Outlook ReplyTime property, updateed when a user accepts or declines an invitation. The server stores this property and returns it in <event> tags, but does not use it for anything.
master-clientid	flag	255	n/a	0 or 1	This is the master clientid as specified by the masterclientid attribute in updateevent.xml requests. If no value was specified by updateevent for an event, this element is not returned.

getevents Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE calendar SYSTEM "http://mail.example.com/dtd/xcal/v2/calendar.dtd”>
<calendar>
<ok>OK</ok>
<event>
<eventid>e102822001</eventid>
<eventtitle>an event with 3 attendees</eventtitle>
<eventdesc>a full different event description</eventdesc>
<eventstart>20020319T000000Z</eventstart>
<eventstop>20020319T010000Z</eventstop>
<viewablestart>20020319T000000Z</viewablestart>
<viewablestop>20020319T010000Z</viewablestop>
<eventpriority>4</eventpriority>
<eventemaildiff>10</eventemaildiff>
<eventpagerdiff>10</eventpagerdiff>
<eventxdata>a full different eventxdata</eventxdata>
<created>20020620T103400Z</created>
<dtstamp>20020620T120512Z</dtstamp>
<last-modified>20020620T110500Z</last-modified>
<sequence>12</sequence>
<attendees>
<attendee role="CHAIR" partstat="ACCEPTED">t0</attendee>
<attendee role="REQ-PARTICIPANT" partstat="DECLINED">t1</attendee>
<attendee role="REQ-PARTICIPANT" partstat="ACCEPTED">t2</attendee>
<attendee role="REQ-PARTICIPANT" partstat="NEEDS-ACTION">t3</attendee>
<attendee role="CHAIR" partstat="ACCEPTED">User 0 on qa33 &lt;u0_qa33&gt;</
    attendee>
</attendees>
<extattendees>
<extattendee partstat="NEEDS-ACTION">g@m.com</extattendee>
<extattendee partstat="NEEDS-ACTION">u0</extattendee>
</extattendees>
<resources>
<resource partstat="ACCEPTED">diablo</resource>
<resource partstat="ACCEPTED">projector</resource>
</resources>
```

```
<perms scope="EVENT_RDACCESS" type="INCLUDE">
<permuser>u0_polarbear</permuser>
<permuser>u0_bustle</permuser>
</perms>
<perms scope="EVENT_WRACCESS" type="INCLUDE">
<permuser>u0_polarbear</permuser>
<permuser>u1_bustle</permuser>
</perms>
</event>
<event>
...
</event>
</calendar>
```

gettodos Command

The gettodos command returns a complete list of to-do items in a user's calendar or a single to-do item requested by `todoid`. All to-do items returned are sorted in priority order, highest priority first. If the `todoid` field is missing or empty, all to-dos are returned, otherwise just the corresponding to-do item for that `todoid` is returned.

Table 17 gettodos Command Definition

Operation	`GET /mc/xcal/v2/gettodos.xml`	
Arguments	`sid`	session ID
	`todoid?`	todo ID
Result	Result A <calendar> element containing 0 or more <todo> elements, sorted in priority order (highest priority first).	

Elements common to events and to-dos are only listed in the table of elements for the event.

Table 18 to-do Components

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
`todoid`	required	255	n/a	1	The XML-encoded ID of the to-do item. `todoid` is assigned by the Mirapoint calendar store and is an ASCII string that uniquely identifies the to-do within a user's calendar.
`todotitle`	required	255	n/a	1	The title of the to-do item. Cannot be an empty string.
`tododesc`	optional	65531	n/a	0 or 1	The description of the to-do record.
`todopriority`	required	1	3	1	The to-do priority. Valid values are 1 through 5, with 1 being the highest.

Table 18 to-do Components

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
todoxdata	optional	65531	undefined	0 or 1	A general-purpose string that lets client software store its own data in a to-do record. Mirapoint calendar store does not alter this field's contents.
created	required	n/a	n/a	1	ISO-8601 date and time when the task was created by the server.
dtstamp	required	n/a	n/a	1	ISO-8601 date and time when the task was given to the client.
last-modified	required	n/a	n/a	1	ISO-8601 date and time when the task was last modified.
sequence	required	n/a	n/a	1	The number of times that this task was modified.
due	optional	n/a	undefined	0 or 1	The time/date when the to-do is due, in ISO-8601 format.
completed	optional	n/a	undefined	0 or 1	The time/date when the to-do was completed, in ISO-8601 format.

gettodos Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE calendar SYSTEM "http://mail.example.com/dtd/xcal/v2/calendar.dtd”>
<calendar>
<ok>OK</ok>
<todo>
<todoid>t104901e00</todoid>
<todotitle>Need to buy a mirapoint server</todotitle>
<tododesc>The stuff we talked about yesterday.</tododesc>
<todopriority>1</todopriority>
<created>20020620T121400Z</created>
<dtstamp>20020620T121449Z</dtstamp>
<last-modified>20020620T121400Z</last-modified>
<sequence>2</sequence>
<due>20020629T060000Z</due>
<completed>20020630T060000Z</completed>
</todo>
<todo>
...
</todo>
</calendar>
```

All ISO-8601 time fields are expressed relative to UTC.

localelist Command

Localelist command displays all available calendar (mcal) locales on the system. No parameters are needed.

Call localelist prior to login, pick the locale of the user's choice and login including the locale name as a parameter.

Table 19 localelist Command Definition

Operation	`GET /mc/xcal/v2/localelist.xml`
Result	<localelist> document

Table 20 localelist DTD Fields

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
`default`	optional	N/A	N/A	0, or 1	Indicates that the locale containing this element is the default locale on the system.
`name`	required	1024	N/A	1	Identifies a unique, internal name for the locale. This string should be used in the locale parameter during login.
`language`	required	1024	N/A	1	Identifies a human readable form for the locale name. The string appears localized in its own locale.

localelist Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE localelist SYSTEM "http://mail.example.com/dtd/xcal/v2/
    localelist.dtd”>
<localelist>
    <locale>
    . <default>
    . <name>en_US.ISO_8859-1</name>
    . <language>English</language>
    </locale>
    <locale>
    . <name>de_DE.utf-8L3_1_0</name>
    . <language>Deutsch</language>
    </locale>
    <locale>
    . <name>it_IT.utf-8L3_1_0</name>
    . <language>Italiano</language>
    </locale>
</localelist>
```

login Command

The login action performs a login or logout operation for a user. A POST operation requests login, creating a session if authentication succeeds. A GET operation terminates a session, logging a user out.

Table 21 Login Command Definition

Operation	`POST /mc/xcal/v2/login.xml`

Table 21 Login Command Definition

Arguments	user	User's login name
	password	User's password
	locale	Session locale. If parameter is not specified, use system's default locale.
	caluser	Username of the calendar that is to be manipulated; should be used for administration only (see below).
	sid	Session ID (for logout only)
Result	<login> doctype (for logins)	
	<status> doctype (for logouts)	

In some cases, a client needs to manipulate a calendar without access to the user's password. One example is automatic synchronization of calendar data in the Mirapoint calendar store with other devices. For this reason, the login command lets users with administrator privileges only log in as themselves, and read and write another user's calendar. For example, client software needs to update user Joe with new events it detected in another of Joe's devices. The client logs into the XML interface with `user` set to the administrator's user name (for example, "administrator"), `password` set to the administrator's password and `caluser` set to "joe". The administrator then has a session open to perform any operation on Joe's calendar that Joe himself could perform.

On the login reply, the server's current time (in ISO-8601 format) is included to let synchronization clients provide useful input to the getchanges.xml command.

If the user's calendar has been the subject of a server-side repair operation, the `dumpcal` field is provided in successful login replies and contains the last repair time in ISO-8601 format.

If the login request is successful, the response of login.xml consists of all the email addresses of the logged-in user. These email addresses include the primary mail (mail attribute in LDAP) and either mailAlternateAddresses or mailLocalAddresses (whichever is present in the LDAP schema).

All the mail addresses are enclosed in the <emaillist> </emaillist> tag. The primary email address is differentiated from the other email addresses using the attribute type="primary".

Replies to login requests are made using the login DTD, which either contains some failure code, or a success code together with a session ID (contained in an <sid> element). This session ID must be used in all further requests in that session.

login Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE login SYSTEM "http://when/mc/xcal/v2/login.dtd">
<login>
    <ok>20060807T214504Z</ok>
    <sid>40ccdb885c219fca986b1a5ff04685534</sid>
    <time>20060807T214504Z</time>
```

```
    <emaillist>
    . <email type="primary">komal.kaks@dummy.com</email>
    . <email>komalkaks@dummy.com</email>
    . <email>kkaks@dummy.com </email>
    </emaillist>
</login>
```

HTTP Redirection on Login

If a login action is sent to a host on a multiple server farm, an HTTP redirection can result via a 302 (Moved Temporarily) HTTP result.

Example

```
C: POST /mc/xcal/v2/login.xml?user=joe&password=cool HTTP/1.1
C: Host: host0
C: Content-Length: 0
C:
S: HTTP/1.1 302 Moved Temporarily
S: Content-Length: 0
S: Location: http://host1/v2/login.xml?sid=23432
S:
```

permissions Command

The permissions command is used to get or set the various permissions variables for a calendar (described in section 4.3). An HTTP GET gets a <permissions> doctype with a list of the calendar permissions and a POST can be used to update the permissions.

For reading the permissions variables for a calendar:

Table 22 permissions Command Definition (Reading)

Operation	GET /mc/xcal/v2/permissions.xml	
Arguments	sid	session ID
	fqdn	"yes" to fully qualify all userIDs in the top level in the events returned, "no" to not. This parameter defaults to "no".
Result	<permissions> doctype # for preferences requests	

Permission elements are used to describe which users are allowed to perform which operations.

The <permissions> element contains a list of <perms> elements, each of which defines which users are allowed to perform a certain operation. The scope attribute

determines which operation the <perms> element refers to and can take the following values:

Table 23 Scope Attribute Values

value	operation
CAL_RDACCESS	Top-level read access to a calendar. If a user does not have such read access to a calendar, they can not read any information from that calendar.
CAL_WRACCESS	Top-level write access to a calendar. If a user does not have such write access to a calendar, they cannot write any information to that calendar, except for scheduling events in that calendar (see SCHD_WRACCESS).
DFLT_RDACCESS	When an event is created, the default read access to the event is initialized to be the same as this permissions field (see EVENT_RDACCESS). (*)
DFLT_WRACCESS	When an event is created, the default write access to the event is initialized to be the same as this permissions field (see EVENT_WRACCESS). (*)
FRBS_RDACCESS	Free-busy lookup access to a calendar. If a user does not have such access to a calendar, they cannot request free-busy information.
SCHD_WRACCESS	Scheduling write access to a calendar. For a user to be able to request a meeting with another user, the first user must have such access to that calendar, otherwise a tentative meeting cannot be scheduled in that user's calendar.
EVENT_RDACCESS	Read-access to a particular event. Defaults to the value of DFLT_RDACCESS at the time the event is created.
EVENT_WRACCESS	Write-access to a particular event. Defaults to the value of DFLT_WRACCESS at the time the event is created.
(*) The current WebCal GUI does not support thie flag.	

EVENT_RDACCESS and EVENT_WRACCESS are not allowed in GET or POST permissions.xml operations.

The `type` attribute determines what type of access control is being used for the `<perms>` element and can take two values:

- INCLUDE -- only users listed in the `<perms>` element and the owner of the calendar or event, are allowed to perform the operation.
- PUBLIC -- any user can perform that operation.

The `<permuser>` elements are used to list the users allowed to perform the operation and must contain the login ID of a valid user in the Mirapoint cluster.

For updating the permissions variables for a calendar:

Table 24 permissions Command Definition (Updating)

Operation	POST /mc/xcal/v2/permissions.xml	
Arguments	sid	session ID
	cal_rdaccess_type?	The type of access control to use for read access permissions to the calendar, one of: "PUBLIC" -- anyone can read the calendar. There must be no cal_rdaccess_user inputs. "INCLUDE" -- only users specified by the cal_rdaccess_user inputs can read the calendar. If there are no cal_rdaccess_user inputs, only the calendar owner an read it.
	cal_rdaccess_user*	Specifies a user who is allowed to read the calendar, if the type is "INCLUDE". This field cannot be used if cal_rdaccess_type is "PUBLIC".
	cal_wraccess_type?	The type of access control to use for write access permissions to the calendar, one of: "PUBLIC" -- anyone can modify the calendar. There must be no cal_wraccess_user inputs. "INCLUDE" -- only users specified by the cal_wraccess_user inputs can modify the calendar). If there are no cal_wraccess_user inputs, only the owner of the calendar can modify it.
	cal_wraccess_user*	Specifies a user who is allowed to modify the calendar, if the type is "INCLUDE". This field cannot be input if the cal_wraccess_type is "PUBLIC".
	dflt_rdaccess_type?	The default access control type to be used for read access to new events (see "updateevent Command" on page 56), one of: "PUBLIC" -- anyone can read the event. There must be no dflt_rdaccess_user inputs. "INCLUDE" -- only users specified by the dflt_rdaccess_user inputs may read the event. If there are no dflt_rdaccess_user inputs, only the owner of the calendar can read the new event.
	dflt_rdaccess_user*	Specifes a user who is allowed to read a new event by default, if the type is "INCLUDE". This field cannot be input if the cal_wraccess_type is "PUBLIC".

Table 24 permissions Command Definition (Updating)

	`dflt_wraccess_type?`	The default access control type to be used for write access to new events (see "updateevent Command" on page 56), one of: "PUBLIC" -- anyone can modify the event. There must be no `dflt_wraccess_user` inputs. "INCLUDE" -- only users specified by the `dflt_wraccess_user` inputs can modify the event. If there are no `dflt_wraccess_user` inputs, only the owner of the calendar themselves can modify the new event.
	`dflt_wraccess_user*`	Specifies a user who is allowed to modify a new event by default, if the type is "INCLUDE". This field cannot be input if the `dflt_wraccess_type` is "PUBLIC".
	`frbs_rdaccess_type?`	The type of access control to use for free-busy lookup access to the calendar, one of: "PUBLIC" -- anyone can request free-busy information from the calendar. There must be no `frbs_rdaccess_user` inputs. "INCLUDE" -- only users specified by the `frbs_rdaccess_user` inputs can be given free-busy information from the calendar. If there are no `frbs_rdaccess_user` inputs, only the owner of the calendar can request free-busy information.
	`frbs_rdaccess_user*`	Specifies a user who is allowed to request free-busy information, if the type is "INCLUDE". This field cannot be input if the `frbs_rdaccess_type` is "PUBLIC".
	`schd_wraccess_type?`	The type of access control to use for scheduling permissions to the calendar. For another user to schedule an event in the calendar, they must have such permission. This field can be: "PUBLIC" -- anyone can schedule events in the calendar. There must be no `schd_wraccess_user` inputs. "INCLUDE" -- only users specified by the `schd_wraccess_user` inputs are allowed to schedule events in the calendar. If there are no `schd_wraccess_user` inputs, no other users can schedule events in the calendar.
	`schd_wraccess_user*`	Specifies a user who is allowed to schedule meetings in the calendar, if the type is "INCLUDE". This field cannot be input of the `schd_wraccess_type` is "PUBLIC".
Result	`<status>` doctype	

The same restrictions apply to this type and user fields as in "updateevent Command" on page 56.

permissions Example

Example of a GET response:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE permissions SYSTEM "http://mail.example.com/dtd/xcal/v2/
    permissions.dtd”>
<permissions>
    <ok>OK</ok>
    <perms scope="DFLT_RDACCESS" type="PUBLIC">
    </perms>
    <perms scope="DFLT_WRACCESS" type="PUBLIC">
    </perms>
    <perms scope="CAL_RDACCESS" type="INCLUDE">
    . <permuser>u0_bustle</permuser>
    . <permuser>u1_bustle</permuser>
    </perms>
    <perms scope="CAL_WRACCESS" type="INCLUDE">
    . <permuser>u2_bustle</permuser>
    . <permuser>u3_bustle</permuser>
    </perms>
    <perms scope="FRBS_RDACCESS" type="INCLUDE">
    . <permuser>u4_bustle</permuser>
    . <permuser>u0_bustle</permuser>
    </perms>
    <perms scope="SCHD_WRACCESS" type="INCLUDE">
    . <permuser>u1_bustle</permuser>
    . <permuser>u2_bustle</permuser>
    </perms>
</permissions>
```

Example of a POST response:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE status SYSTEM "http://mail.example.com/dtd/xcal/v2/status.dtd”>
<status>
<ok>OK</ok>
</status>
```

prefs Command

The prefs commands returns the current state of the user preferences for GET operations or updates the current state of the user preferences for POST operations. When updating user preferences, a subset of the allowed elements can be used. Any preference whose elements are missing from the POST are left unchanged.

For reading the preferences:

Table 25 prefs Command Definition (For Reading Preferences)

Operation	`GET /mc/xcal/v2/prefs.xml`	
Arguments	`sid`	session ID
Result	<preferences> doctype # for preferences requests	

For updating the preferences:

Table 26 prefs Command Definition (For Updating Preferences)

Operation	`POST /mc/xcal/v2/prefs.xml`

Table 26 prefs Command Definition (For Updating Preferences)

Arguments	sid	session ID
	fullname	user's full name
	emaildiff	number of minutes before event an email reminder should be sent (default). See definition in the "'event' and 'todo' elements" section.
	pagerdiff	number of minutes before event a mobile reminder should be sent (default). See definition in the "'event' and 'todo' elements" section.
	emailsummarytime*	hour of day when email should be sent
	emailsummarytype	summaries contain today's/tomorrow's events
	pagersummarytime	hour of day when email should be sent
	pagersummarytype	summaries contain today's/tomorrow's events
	weeklysummarytime	hour of day when weekly summary should be sent
	weeklysummaryday	weekday when summary is sent
	monthlysummarytime	hour of day when monthly summary should be sent
	monthlysummaryday	day of the month when monthly summary is emailed
	emailaddr	address to which email reminders should be sent
	pageraddr	address to which mobile reminders should be sent
	dfltview	default view on login to WebCal
	daystart	start of day (hours from midnight)
	dayend	end of day (hours from midnight; must be > daystart)
	daygranularity	size of blocks into which a day is divided (in minutes)
	weekdaysep	show separator bar in weekly view of WebCal
	showeventtxt	show the description of an event with the summary of the event in short form
	showtodolist	show the todo list together with the calendar in WebCal
	weekstart	day on which a week starts (0==Sunday)

Table 26 prefs Command Definition (For Updating Preferences)

	`version`	UI version to use in WebCal (v_noframe = no frames, no javascript, v_nojs = frames, no javascript, v_all = frames and javascript)
	`timezone`	timezone
	`published?`	whether the calendar is published or not. If omitted, the published flag is left unchanged in the calendar. If present and equal to "y", the calendar becomes published, otherwise it becomes not published.
Result	`<status>` doctype for updates	

Table 27 Setting Preferences: Element Semantics, Allowed Values

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
`fullname`	required	50	empty string	1	user's fullname in UTF-8.
`emaildiff`	required	255	box default	1	See definition in the 'getevent's and 'gettodos' sections.
`pagerdiff`	required	255	box default	1	Same as `emaildiff`, but for mobile devices.
`emailsummarytime`	required	2	box default	1	Hour of the day to send daily summary emails. Allowed values: integers 0 through 23 and -1, which disables the sending of daily email summaries.
`emailsummaytype`	required	1	box default	1	Should the summary mails contains today's events or next day's events. Allowed values are 0 (send following day's events) and 1 (send today's events).
`pagersummarytime`	required	255	box default	1	Same as `emailsummarytime`, for mobile devices.
`pagersummarytype`	required	255	box default	1	Same as `emailsummarytype`, for mobile devices.
`weeklysummarytime`	required	255	box default	1	Same as `emailsummarytime`, for weekly summaries

Table 27 Setting Preferences: Element Semantics, Allowed Values

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
`weeklysummaryday`	required	255	box default	1	Day of the week on which weekly summary is emailed. Allowed values are integers 0 through 6, with Sunday = 0, Monday = 1, and so on. Selecting weekstart sends the current week's events; everything else sends the following week's events.
`monthlysummarytime`	required	255	box default	1	Same as `emailsummarytime`, for monthly summaries
`monthlysummaryday`	required	255	box default	1	Sets the day of the month when monthly summary is sent. Allowed values: 1 (first day of the month) -1 (last day of the month) -2 (day before the last day of the month) -3 (two days before the last day of the month) 28 - 20 (days of the month) Selecting 1 sends the current month's events. Selecting anything else sends the following month's events.
`emailaddr`	required	255	empty string	1	Email address to use for email reminders and daily/weekly/ monthly summaries.
`pageraddr`	required	255	empty string	1	Email address to use for mobile device reminders and daily summaries.
`dfltview`	required	1	box default	1	Default view in Mirapoint web-based UI. Possible values are 'd' (day view), 'h' (horizontal weekly view), 'w' (week view) and 'm' (month view).
`daystart`	required	2	box default	1	User's start of day. Valid values are 0 through 23, where 0 = midnight, 23 = 11pm, and so on.
`dayend`	required	2	box default	1	User's end of day (same values as daystart).

Table 27 Setting Preferences: Element Semantics, Allowed Values

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
`daygranularity`	required	2	box default	1	In daily view, the size of the units that a day is divided into. Allowed values are 15, 30, and 60 (minutes).
`weekdaysep`	required	1	box default	1	Shows bars between days in the daily view. Allowed values are "y" (yes) and "n" (no).
`showeventtxt`	required	1	box default	1	Show the description of an event with the title in daily and weekly views. Allowed values are "y" (yes) and "n" (no).
`showtodolist`	required	1	box default	1	Show the to-do list in the main calendar page in the Mirapoint web-based UI. Allowed values are "y" (yes) and "n" (no).
`weekstart`	required	1	0	1	Day of the week on which the user's week starts. Allowed values are 0 through 6, where Sunday = 0, Monday = 1, and so on.
`version`	required	255	v_nojs	1	UI version. v_nojs = frames but no Javascript, v_noframe = no frames and no Javascript (simple mode), v_all = Javascript and frames.
`timezone`	required	255	box default	1	Timezone of the user. For a list of time zones, use the CLI **Ntp Get Knownzones** command.
`published`	optional	N/A	n	0 or 1	If present and set to “y” (yes), the calendar is published (and can be subscribed to).

- replyopt can have the following values:
  `(dontinclude|includeinline|includeattach)`
- includesig can have the following values:
  `(yes|no)`
- usetrash can have the following values:
  `(yes|no)`
- version can have the following values:
  `(noframes|nojavascript|javascript)`
- charset can have values as given in the preferences element after request.
- timezone can have values listed by the CLI **Ntp Get Knownzones** command.

Example: GET Response

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE preferences SYSTEM "http://mail.example.com/dtd/xcal/v2/
    preferences.dtd”>
<preferences>
    <ok>OK</ok>
    <fullname></fullname>
    <emaildiff>5</emaildiff>
    <pagerdiff>-1</pagerdiff>
    <emailsummarytime>-1</emailsummarytime>
    <emailsummarytype>1</emailsummarytype>
    <pagersummarytime>-1</pagersummarytime>
    <pagersummarytype>1</pagersummarytype>

    <weeklysummarytime>-1</weeklysummarytime>
    <weeklysummaryday>0</weeklysummaryday>
    <monthlysummarytime>-1</monthlysummarytime>
    <monthlysummaryday>1</monthlysummaryday>
    <emailaddr></emailaddr>
    <pageraddr></pageraddr>
    <dfltview>w</dfltview>
    <daystart>8</daystart>
    <dayend>18</dayend>
    <daygranularity>60</daygranularity>
    <weekdaysep>y</weekdaysep>
    <showeventtxt>n</showeventtxt>
    <showtodolist>y</showtodolist>
    <weekstart>0</weekstart>
    <version>v_nojs</version>
    <timezone>Etc/GMT</timezone>
    <published>n</published>
</preferences>
```

Example: POST Response

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE status SYSTEM "http://mail.example.com/dtd/xcal/v2/status.dtd”>
<status>
<ok>OK</ok>
</status>
```

search Command

The search command searches for events and to-do items in the calendar store and returns any events whose description or title contain one or more of the keywords supplied in the search string.

Table 28 search Command Definition

Operation	`GET /mc/xcal/v2/search.xml`	
Arguments	`sid`	session id
	`searchstr`	search string. Its length can be between 0 through 255 characters.
	`fqdn`	"yes" to fully qualify all userIDs in the top level in the events returned, "no" to not. This command defaults to "no".

Table 28 search Command Definition

Result	A <calendar> element containing 0 or more <event> and/or <todo> elements that match the search string.

search example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE calendar SYSTEM "http://mail.example.com/dtd/xcal/v2/calendar.dtd”>
<calendar>
<ok>OK</ok>
<event>
...
</event>
<event>
    <eventid>e102a58401</eventid>
    <eventtitle>a partial event title</eventtitle>
    <eventdesc>a partial event description</eventdesc>
    <eventstart>20020325T070000Z</eventstart>
    <eventstop>20020325T080000Z</eventstop>
    <viewablestart>20020325T070000Z</viewablestart>
    <viewablestop>20020325T080000Z</viewablestop>
    <allday/>
    <eventpriority>5</eventpriority>
    <eventemaildiff>5</eventemaildiff>
    <eventpagerdiff>5</eventpagerdiff>
    <eventnotifylist>
    . <eventnotifyelement>g@m.com</eventnotifyelement>
    . <eventnotifyelement>u2_bustle</eventnotifyelement>
    </eventnotifylist>
    <eventrrule>D1 20020501T000000Z</eventrrule>
    <eventxdata>a partial eventxdata</eventxdata>
    <created>20020620T135600Z</created>
    <dtstamp>20020620T135658Z</dtstamp>
    <last-modified>20020620T135600Z</last-modified>
    <sequence>3</sequence>
    <x-mira-sendmail/>
    <attendees>
    . <attendee role="CHAIR" partstat="ACCEPTED">t0</attendee>
    . <attendee role="REQ-PARTICIPANT" partstat="NEEDS-ACTION">t1</attendee>
    . <attendee role="REQ-PARTICIPANT-" partstat="NEEDS-ACTION">t2</attendee>
    </attendees>
    <extattendees>
    . <extattendee partstat="NEEDS-ACTION">g@m.com</extattendee>
    . <extattendee partstat="NEEDS-ACTION">t4_hwtest29</extattendee>
    </extattendees>
    <resources>
    . <resource partstat="ACCEPTED">Diablo_cr</resource>
    . <resource partstat="ACCEPTED">projector</resource>
    </resources>
    <perms scope="EVENT_RDACCESS" type="INCLUDE">
    . <permuser>u0_polarbear</permuser>
    . <permuser>u1_bustle</permuser>
    </perms>
    <perms scope="EVENT_WRACCESS" type="INCLUDE">
    . <permuser>u1_polarbear</permuser>
    . <permuser>u1_bustle</permuser>
    </perms>
</event>
<todo>
    <todoid>t104901e00</todoid>
    <todotitle>Need to buy a mirapoint server</todotitle>
    <tododesc>The stuff we talked about yesterday.</tododesc>
```

```
    <todopriority>5</todopriority>
    <created>20020620T121400Z</created>
    <dtstamp>20020620T135658Z</dtstamp>
    <last-modified>20020620T121400Z</last-modified>
    <sequence>2</sequence>
    <due>20020629T060000Z</due>
    <completed>20020630T060000Z</completed>
</todo>
</calendar>
```

subscriptions Command

The subscriptions command is used for editing the list of calendars to which the user is subscribed.

When a user subscribes to a calendar (which must first be published), all events to which the user has read access automatically appear in their calendar (that is, are returned by the getevents command) in read-only mode.

A GET of the page returns a `<userlist>` doctype with a list of the users to which the user is subscribed and a POST of the page updates that list:

Table 29 subscriptions Command Definition (Retrieve)

Operation	`GET /mc/xcal/v2/subscriptions.xml`	
Arguments	`sid`	session ID
	`fqdn`	"yes" to fully qualify all userIDs in the top level in the events returned, "no" to not. This defaults to "no".
Result	`<userlist>` doctype	

The XML interface to the calendar lets queries be made that return a list of users (for example, LDAP queries to find a user based on a search string, list of subscribed calendars). The `userlist` doctype and elements are used to represent such a list of users. It is described below.

To update the subscription list:

Table 30 subscriptions Command Definition (Update)

Operation	`POST /mc/xcal/v2/subscriptions.xml`	
Arguments	`sid`	session ID
	`subscribeto*`	A user whose calendar the calendar should be subscribed to. One of these input values must be present for each calendar that is to be subscribed to. The values used must be the userids found in <userlist> doctypes (that is, obtained using the `finduser` command). Maximum length per instance is 255 characters.
Result	**`<status>`** doctype	

The calendars listed by the `subscribeto` inputs in the POST described above overwrite any existing subscription list. A client deletes an entry from the list by first doing a GET operation to get the subscription list, removing the entry and then

POSTing the list back to the server. A single empty subscribeto field causes all subscribeto entries to be deleted in the subscription list.

The <user> element contains a user-viewable form of the username, which can be either the user's login ID or an RFC-822 compliant phrasal form of their email address. The `userid` attribute is a string that must be used by the client whenever referring to one of the listed users in a request to the server.

subscriptions Example

Example of a GET response:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE userlist SYSTEM "http://mail.example.com/dtd/xcal/v2/userlist.dtd”>
<userlist>
    <ok>OK</ok>
    <users>
    . <user userid="u1_belldandy">&quot;User 1 on belldandy,
    u1_bbelldandy&quot; &lt;u1_belldandy&gt;</user>
    . <user userid="u1_bench350">&quot;User 1 on bench350, u1_bench350&quot;
    &lt;u1_bench350&gt;</user>
    . <user userid="u1_bustle">&quot;User 1 on bustle, u1_bustle&quot;
    &lt;u1_bustle&gt;</user>
    . <user userid="u1_chrysos">&quot;User 1 on chrysos, u1_chrysos&quot;
    &lt;u1_chrysos&gt;</user>
    . <user userid="u1_polarbear">&quot;User 1 on polarbear, u1_polarbear&quot;
    &lt;u1_polarbear&gt;</user>
    . <user userid="u1_qa121">&quot;User 1 on qa121, u1_qa121&quot;
    &lt;u1_qa121&gt;</user>
    . <user userid="u1_webclient25">&quot;User 1 on
    webclient25,u1_webclient25&quot; &lt;u1_webclient25&gt;</user>
    </users>
</userlist>
```

Example: POST Response

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE status SYSTEM "http://mail.example.com/dtd/xcal/v2/status.dtd”>
<status>
<ok>OK</ok>
</status>
```

time Command

The time command displays the server's current time in GMT.

Table 31 time Command Definition

Operation	`GET /mc/xcal/v2/time.xml`
Result	<time> document

time Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE version SYSTEM "http://mail.example.com/dtd/xcal/v2/status.dtd">
<status>
    <ok>20040821T052225Z</status>
</status>
```

updateevent Command

The updateevent command creates a new, or updates an existing, event in the calendar store.

If the `eventemaildiff` or `eventpagerdiff` elements are omitted, values are taken from the calendar's defaults. For details of field defaults and sizes, see the "getevents Command" on page 29. `eventid` is an optional field.

To create an exception, specify <parenteventid>-<exceptiondate> as the `eventid`.

- If `eventid` is not provided, `eventtitle` is required, and all other fields are optional.
- If `eventid` is provided, all fields are optional (although passing no fields does not change the event), and only those provided replace the existing values. To erase a value, supply a replacement value or an empty value as follows (an empty string is referred to as a string of zero length):
  a. `eventtitle` can be cleared by setting it to an empty string only if `eventid` is specified. `eventtitle` can be cleared but never be deleted from the event.
  b. `allday`, `priority`, `eventemaildiff`, `eventpagerdiff`, `start`, `stop`, and `sendmail` can be replaced only by specifying one of their pre-defined values.
  c. `eventdesc`, `eventnotifylist`, `rrule`, `eventxdata`, `attach`: An empty string deletes the entry from the event. If no entry is given, the entry remains unchanged in the event. However given an empty string, `attach` deletes all attachments. To delete just one attachment, use `attachdelete` instead.
  d. `x-mira-agent`: A log entry is written out if this entry is provided.
  e. `attendee`, `extattendee`, `resource`: A list composed of a single empty value causes all entries to be deleted in the event. Any empty entry among other non-empty entries is an error condition. If no entries are given, the entries remain unchanged in the event.
  f. `rdaccess_type`, `rd_access_user`, `wraccess_type`, `wr_access_user`:
  If the type field is set to PUBLIC, no user entries must be present.
  If the type field is INCLUDE, at least one corresponding (for example, rd or wr) user entry must be present. A list composed of a single empty user entry deletes all existing user entries from the event. Any empty user entry among other non-empty user entries is an error condition. If no user entries are given, the entries remain unchanged in the event.

Table 32 updateevent Command Description

Operation	POST /mc/xcal/v2/updateevent.xml	
Arguments	sid	session ID
	eventid?	event ID; can be omitted only for a new event being created. In that case, eventtitle is not required.
	clientid	(optional) Client ID of the event. The maximum number of chars allowed is 1024.
	eventtitle	Title; Optional if eventid is provided. If provided with eventid, it replaces the existing eventtitle value.
	eventdesc?	Description of the event.
	allday?	"y" if this is an all-day event.
	priority?	1 to 5, 1 is highest priority.
	eventnotifylist?	list of email addresses to which notification should be sent. Maximum length = 65531
	rrule?	a vCalendar RRULE string; "D", "W", "MP", "MD", "YM" types are supported. The "YD" type is not supported. Weekly repeating events may occur on any combination of weekdays. Monthly repeating events may only occur once per month. Events can repeat until a specific date or repeat forever (or until the maximum supported time).
	start	start date, in ISO-8601.
	stop	stop date, in ISO-8601.
	eventemaildiff	number of minutes before event reminder is sent. -1 means that no reminder is desired. See definition in the "'event' and 'todo' elements" section.
	eventpagerdiff	Same as eventemaildiff, but for mobile devices. See definition in the "'event' and 'todo' elements" section.
	eventxdata?	client side data.
	attendee*	the userid (see "subscriptions Command" on page 54) of an attendee that is invited to a meeting. There can be multiple instances of this value in the POST.

Table 32 updateevent Command Description

	extattendee*	the (RFC-822 compliant) email address of an external attendee that is invited to a meeting. There can be multiple instances of this value in the POST. If extsendmail=y, all external attendees receive email containing a URL that they can use to accept or decline the invitation.
	resource*	the userid (see "subscriptions Command" on page 54) of a resource that is scheduled for a meeting. There can be multiple instances of this value in the POST.
	eventlocation?	user-supplied information about location, stored in the user's database and in all attendees' databases for the event. Specifying an empty eventlocation field deletes the location from pre-existing events. Not specifying this field leaves data on the server unchanged. Only the event owner can change this field.
	category*	The name of a category associated with a meeting. If a case-insensitive match of a category specified matches one that exists on the Mirapoint system, that is used; otherwise a new category with that name is created. A new category created in this way defaults to using the calendar default ACLs. The color of the category is undefined. If the event being updated already contains a list of associated categories, it is replaced with the list provided by the updateevent.xml request.
	sendmail?	If present and equal to "y", the server sends out notification emails to all attendee's specified by the attendee inputs. This flag is stored in the event record so subsequent changes through other interfaces also cause mail to be sent. The state of this flag is viewable in the event DTD and via the native Mirapoint web interface.
	extsendmail?	If present and equal to "y", the server sends out notification emails to external attendees. This flag affects the current operation only.

Table 32 updateevent Command Description

	rdaccess_type?	The type of access control to a user for read access permissions to the event, one of: "PUBLIC" -- anyone with read access to the calendar can read the event. There must be no rdaccess_user inputs. "INCLUDE" -- only users specified by the rdaccess_user inputs and who have read access to the calendar can read the event. If there are no rdaccess_user inputs, only the owner of the event can read it. "DEFAULT" -- default calendar permissions. If there are categories associated with the event, for an operation to be performed on the event, ACL rules for at least one of the categories must be satisfied. If no categories are associated with the event, the calendar default ACLs are used. If this is a new event and no access is specified, "DEFAULT" is used.
	rdaccess_user*	The userid (see "subscriptions Command" on page 54) of a user who is allowed to read the event (if they have read access to the calendar).
	wraccess_type?	The type of access control to a user for write access permissions to the event, one of: "PUBLIC" -- anyone can write the event who has write access to the calendar. There must be no wraccess_user inputs. "INCLUDE" -- only users specified by the wraccess_user inputs and who have write access to the calendar can write to the event. If there are no wraccess_user inputs, only the owner of the event can modify it. "DEFAULT" -- default calendar permissions. If this is a new event and no access is specified, "DEFAULT" is used. For a description of how "DEFAULT" ACLs are handled, see rdaccess_type.
	wraccess_user*	The userid (see "subscriptions Command" on page 54) of a user allowed to write to the event (if they have write access to the calendar).
	attach*	Multiple attachments can be added when creating an event.
	attachdelete?	Specifies filename(s) of attachment(s) to remove from the event. Attach is processed afterwards to ensure proper replacement of attachments.
	x-mira-agent?	(optional) A text string that identifies the client software. This string is not stored with the event record, but can be used for logging. Maximum length is 63 characters.

Table 32 updateevent Command Description

	attendeefallback?	(optional) Affects what happens when meetings cannot be schedules with some of the specified attendees. These values are accepted: none - Operation fails, the current default. discard - Unschedulable attendees are discarded. external - Unschedulable attendees go external.
	timezone?	(optional) Specifies the timezone the event is to be created in (for relevance, see "Repeating Events" on page 20). The value must be one of the current time zone values listed by the CLI **Ntp Get Knownzones** command. If this parameter is absent, the event is created in the timezone in the user's preferences.
	status?	for group events, sets the event's status: accepted (status=accept) or declined (status=decline)
	metadata*	contains the name of the opaque data variable (maxlen = 255) and the value element (maxlen = 1024). To set the data, the name should be separated by a colon from the value, as in for:bar. The server stores and preserves all metadata fields for the client.
	sequence	Allows client to set sequence number attribute. This is the same sequence number present in iCal/iTIP/iMIP messages as the SEQUENCE property. Integer >= 0.
	sequenceinit	A property Outlook uses to generate sequence numbers for owner events. The server stores this as opaque data and returns it in <event> elements. Integer >= 0.
	clientlastmodified	Allows client to set an event timestamp. This value is stored in the event record and returned by the <event> element replying to getevents.xml (see below). ISO-8601 date/time.
	replytime	Outlook ReplyTime property, updated when a user accepts or declines an invitation. The server stores this property and returns it in <event>, but does not actually use it. ISO-8601 date/time.
	masterclientid	Clientid of master (parent) record of a repeating series. Stored as opaque data in event records and not used for server scheduling logic. Event lookups cannot be done on masterclientid. Used by SynQ for meetings where an exception arrives before repeating series. ASCII string.
	<update> doctype	

To create a new event, the event ID field should be missing. An empty eventid is considered an error.

The eventxdata field is for client application use. This element's data is stored with the event, and is retrievable from "event" elements, but is not processed by the server.

If the specified eventid is the parent eventid (i.e. no -<date> is present), the entire series is modified. If instance is set to "all", all exceptions are deleted. Otherwise the exceptions are preserved as specified in the "Changing Calendar Repeating Event Whole Series" protocol spec.

The only event attributes that attendees can change (that is, users other than the owner and a user who accessed the calendar with write permissions using the viewother.xml command) are: category, eventemaildiff, eventpagerdiff, rdaccess_type, rdaccess_user, wraccess_type, and wraccess_user.

updateevent Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE login SYSTEM "file:///mc/xcal/v2/update.dtd">
<update>
    <ok>20040101T120030Z</ok>
    <gid>ef00bar</gid>
    <last-modified>20030804T230812</last-modified>
    <event>
    <eventid>e102822001</eventid>
    <eventtitle>an event with 3 attendees</eventtitle>
    <eventdesc>a full different event description</eventdesc>
    <eventstart>20020319T000000Z</eventstart>
    <eventstop>20020319T010000Z</eventstop>
    <viewablestart>20020319T000000Z</viewablestart>
    <viewablestop>20020319T010000Z</viewablestop>
    <eventpriority>4</eventpriority>
    <eventemaildiff>10</eventemaildiff>
    <eventpagerdiff>10</eventpagerdiff>
    <eventxdata>a full different eventxdata</eventxdata>
    <created>20020620T103400Z</created>
    <dtstamp>20020620T120512Z</dtstamp>
    <last-modified>20020620T110500Z</last-modified>
    <sequence>12</sequence>
    <attendees>
    <attendee role="CHAIR" partstat="ACCEPTED">t0</attendee>
    <attendee role="REQ-PARTICIPANT" partstat="DECLINED">t1</attendee>
    <attendee role="REQ-PARTICIPANT" partstat="ACCEPTED">t2</attendee>
    <attendee role="REQ-PARTICIPANT" partstat="NEEDS-ACTION">t3</attendee>
    <attendee role="REQ-PARTICIPANT" partstat="NEEDS-ACTION">"grpcal Test User
    4 on hwtest29, t4_hwtest29" <t4_hwtest29></attendee>
    </attendees>
    <extattendees>
    <extattendee partstat="NEEDS-ACTION">g@m.com</extattendee>
    <extattendee partstat="NEEDS-ACTION">u0</extattendee>
    </extattendees>
    <resources>
    <resource partstat="ACCEPTED">diablo</resource>
    <resource partstat="ACCEPTED">projector</resource>
    </resources>
    <perms scope="EVENT_RDACCESS" type="INCLUDE">
    <permuser>u0_polarbear</permuser>
    <permuser>u0_bustle</permuser>
    </perms>
    <perms scope="EVENT_WRACCESS" type="INCLUDE">
    <permuser>u0_polarbear</permuser>
    <permuser>u1_bustle</permuser>
```

```
    </perms>
    <metadata>
    <name>eventstatus</name>
    <value>Tentative</value>
    </metadata>
    </event>
</update>
```

The <gid> element is used to communicate the ID of the event or to-do item that is being created or updated. This is useful when creating a new event or to-do item, because when specifying it, the event ID is as yet unknown. <last-modified> is the last modified time of the event that was changed/added.

updatetodo Command

The updatetodo command creates, or updates an existing, to-do item in the calendar store.

Table 33 updatetodo Command Definition

Operation	`POST /mc/xcal/v2/updatetodo.xml`	
Arguments	`sid`	session ID
	`todoid`	todo ID; can be omitted only for a new to-do being created. In that case, `todotitle` is not required.
	`todotitle`	Title; Optional if `todoid` is provided. If provided with `todoid`, it replaces the existing `todotitle` value.
	`tododesc?`	
	`priority?`	1 through 5, 1 is highest priority.
	`todoxdata?`	A general-purpose string that lets client software store its own data in the to-do item. The Mirapoint calendar store does not alter this field's contents.
	`due?`	(optional) ISO 8601 time that the to-do item is due. If not present, the to-do item has no due date.
	`completed?`	(optional) ISO 8601 time that the to-do item is completed. If not present, the to-do item has no completed date.
	`x-mira-agent?`	(optional) A text string that identifies the client software. This string is not stored with the event record, but can be used for logging.
Result	<update> doctype	

- If `todoid` is not provided, `todotitle` is required, and all other fields are optional. `todotitle` can never be completely removed from the to-do item.
- If `todoidid` is provided, all fields are optional (although passing no fields do not change the to-do), and only those provided replace the existing values. To

erase a value, you must supply a replacement value or an empty value (an empty string is referred to as a string of zero length):

a. `todotitle` can be cleared by setting it to an empty string only if a `todoid` is specified. `todotitle` can be cleared but never removed from the to-do.
b. `priority` can be replaced only by specifying one of its pre-defined values.
c. `tododesc`, `todoxdata`, `due`, `completed`: An empty string deletes the entry from the event. If no entry is given, the entry remains unchanged in the event.
d. `x-mira-agent`: A log entry is written out if this entry is provided.

updatetodo Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE update SYSTEM "http://when/mc/xcal/v2/update.dtd">
<update>
    <ok>OK</ok>
    <gid>tf00bar</gid>
    . <last-modified>20030804T230812</last-modified>
    <todo>
    . <todoid>t104901e00</todoid>
    . <todotitle>Need to buy a mirapoint server</todotitle>
    . <tododesc>The stuff we talked about yesterday.</tododesc>
    . <todopriority>1</todopriority>
    . <created>20020620T121400Z</created>
    . <dtstamp>20020620T121449Z</dtstamp>
    . <last-modified>20020620T121400Z</last-modified>
    . <sequence>2</sequence>
    . <due>20020629T060000Z</due>
    . <completed>20020630T060000Z</completed>
    </todo>
</update>
```

vcalexport Command

The vcalexport command is used to export calendar information in vCalendar format.

Table 34 vcalexport Command Definition

Operation	`GET /mc/xcal/v2/vcalexport.vcs`	
Arguments	`sid`	session ID
	`lowerbound`	start of window in ISO-8601
	`upperbound`	end of window in ISO-8601
Result	text/x-vCalendar MIME type	

The request is sent via an HTTP GET message. If either `lowerbound` or `upperbound` are equal to `"*"`, that bound is treated as unbounded. So lowerbound=*&upperbound=* results in the whole calendar being exported.

vcalexport Example

The response is of text/x-vCalendar MIME type.

vcalimport Command

The vcalimport command is used to import calendar information in vCalendar format.

Table 35 vcalimport Command Definition

Operation	POST /mc/xcal/v2/vcalimport.xml	
Arguments	sid	session ID
	vcvcal	vCalendar data
Result	<status> doctype	

Data is imported by making an HTTP POST request to the server; the Content-Type MUST be "multipart/form-data" and the HTTP message body must be formatted as such. For example:

```
POST /mc/xcal/v2/vcalimport.xml HTTP/1.0
Connection: Keep-Alive
Host: chrysos
Content-type: multipart/form-data; =----------------------------
    93253262126603226552182772 8
Content-Length: 953

    ----------------------------93253262126603226552182772 8
    Content-Disposition: form-data; name="sessionid"

    861701f76aed4586246b96a2d99c9420
    ----------------------------93253262126603226552182772 8
    Content-Disposition: form-data; name="vcupload"

    Import
    ----------------------------93253262126603226552182772 8\r
    Content-Disposition: form-data; name="vcvcal"; filename="test.vcs"

    BEGIN:VCALENDAR
    PRODID:-//Mirapoint Calendar
    VERSION:1.0
    BEGIN:VEVENT
    UID:20010807T1500Z-0-b@chrysos.mirapoint.com

[ etc ... ]

END:VCALENDAR

    ----------------------------93253262126603226552182772 8--
```

vcalimport example

The response is a status response.

version Command

The version command displays the XML protocl version and the MOS version.

Table 36 version Command Definition

Operation	GET /mc/xcal/v2/version.xml
Result	<version> document

Table 37 version DTD Fields

Name	Type	Max Length (Chars)	Default	Occur-rences	Description
interface	required	128	N/A	1	The XML version of the Mirapoint Group Calendar XML API. It consists of two numbers, comma seperated. The first is the major number. The second is the revision number that increases each time the protocol changes. The version should not be treated as a float point number (for example, 2.10 comes after 2.9).
mos	required	128	N/A	1	Identifies the MOS version (for example, 3.8.4)

version Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE version SYSTEM "http://mail.example.com/dtd/xcal/v2/version.dtd">
<version>
    <interface>2.0</interface>
    <mos>3.4.1.18</mos>
</version>
```

viewother Command

The viewother command is used to view and/or modify a calendar other than that of the user currently logged in. In this section, the user performing the "viewother" request is the "requesting user" and the user whose calendar is being requested is the "user".

Because the user and requesting user calendars are not necessarily hosted by the same server, a two-stage transaction is required for security reasons:

1. The requesting user performs a POST request to the requesting user's server. The server performs permissions validation and, if the requesting user is allowed to read the user's calendar, a URL is returned in an HTTP 302 redirect message (this URL will point to the server hosting the user's calendar). The client software follows this redirect. A GET to the URL provided in the redirect message "logs on" the requesting user to the user's calendar.
2. The user's calendar server validates that the request it received is a authentic by confirming details with the requesting user's server. Once the request is validated, the user's server creates a new session to be used when viewing the user's calendar. The new sessionid is returned in a `viewother` doctype (described below).

The old sessionid used by the requesting user can still be used to access the requesting user's calendar, but the new one must be used to view the user's calendar.

Event and to-do data exported or edited via the viewother.xml command are subject to the ACLs of the events and the categories of the events (if any). Event

access rights of "DEFAULT" mean "apply category ACLs if available, otherwise use the calendar defaults". MOS 3.6 does not support categories for to-do items.

Table 38 viewother Command Definition

Operation	`POST /mc/xcal/v2/viewother.xml`	
Arguments	`sid`	session ID
	`user`	The login ID of the user whose calendar we wish to view.
Result	HTTP 3.0.2 redirect message, containing a URL that when used in a GET request connects to the user's calendar host. This second GET operation returns a `<viewother>` doctype.	

All viewother fields are the same as for the login response, except for `x-mira-readonly` which is specific to the `viewother` command. If this field is present in the viewother doc returned, the requesting user is only allowed read access to the user's calendar and any attempts to modify it in this session fail.

viewother Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE viewother SYSTEM "http://mail.example.com/dtd/xcal/v2/
    viewother.dtd”>
<viewother>
    <ok>OK</ok>
    <sid>a996446bda98d40ad4e7414e93bf41c6</sid>
    <x-mira-readonly/>
<emaillist>
    <email type="primary">komal.kaks@dummy.com</email>
    <email>komalkaks@dummy.com</email>
    <email>kkaks@dummy.com</email>
</emaillist>
</viewother>
```

3 XML Interface to the Mirapoint Message Base

The Mirapoint XML interface to the Mirapoint Message Store uses the HTTP protocol to retrieve XML documents from Mirapoint systems representing the requested data in XML document format.

The kinds of information available to application developers are similar to what can be accessed via the IMAP protocol. Unlike IMAP, the XML interface provides access to message data without having to parse MIME data or decode BASE64 transmission encodings. This simplifies applications such as voicemail or streaming multimedia.

In addition to accessing a Mirapoint Message Base, application developers can use this interface to send email through the Mirapoint system. This interface alleviates application developers from having to know details of SMTP, MIME, attachments, and transmission encodings.

Finally, this interface provides access to WebMail preferences.

Message Addressing

For operations that operate over several messages, there are three techniques used to specify message ranges: specifying msgids, uids, or msgid ranges. You can use any of the three techniques, but two techniques cannot be used together.

Specifying msgids

A msgid is the ordinal value of a message in a mailbox, where the first msgid is 1. When specifying individual msgids, the argument name is "msgids", and the query component looks as follows:

Example, to specify msgids 33, 44, and 100:

```
msgids=33&msgids=44&msgids=100
```

Specifying uids

A uid is an unique value assigned a message in a mailbox. Uids for message in a mailbox are returned via the index command. The query component looks as follows to request uids 2353, 5666, and 9953:

Example, to specify uids 2353, 5666, and 9953:

```
uids=2353&uids=5666&uids=9953
```

Specifying msgid Ranges

To specify ranges of msgids to be used in an operation, the msgranges argument is used. The first msgranges value specified is the start of a message range, the 2nd msgranges value is the end of the first range, the 3rd value is the start of the 2nd range, the 4th value is the end of the 2nd range, and so on. There is a special value "*" which means the last msgid of the mailbox.

Example, for msgid 3 thru 30 and 45 thru 55:

```
msgranges=3&msgranges=30&msgranges=45&msgranges=55
```

Example, for msgid 1 to the last msgid:

```
msgranges=1&msgranges=*
```

Commands

This section describes the commands available in the Mirapoint Message Base XML interface.

append Command

The append command appends a message to a mailbox. The message is in RFC822 text format.

Table 39 append Command Definition

Operation	POST /wm/xml/v1/append.xml	
Arguments	sessionid	session ID
	mailbox	mailbox name
	rfc822msg	me
Result	BAD element	
	NO element	
	OK element	

body Command

The body command returns a set of xlinks that point to the components of a particular message. The first link returned points to the raw message contents, and the subsequent links point to the remaining MIME body parts. The document type is returned if known, otherwise the xlink:type attribute is empty. If empty, the document type can be deduced from the extension of the document name. This version provides no support for multipart/related MIME attachments.

Table 40 body Command Definition

Operation	GET /wm/xml/v1/body.xml

Table 40 body Command Definition

Arguments	sessionid	session ID
	mailbox	mailbox name
	msgid	message ID (or...)
	uid	message user ID
Result	BAD element	
	NO element	
	Body element	

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
<bad>Invalid mailbox name</bad>
<bad>Invalid unique message id</bad>
<bad>Invalid message id</bad>
<bad>Missing uid or msgid value</bad>
```

Body Element

The body elements returns a set of links that point to the contents of the message. These links often do not return XML documents, rather documents whose type is specified by the xlink:type attribute.

The first bodypart in a body element is always a link to the RFC822 message. The second bodypart element contains the first body part of the message, with subsequent bodypart elements following.

Example

In this example, the message contains five body parts of various type and subtype:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE body SYSTEM "http://qa18.example.com/dtd/xml/v1/body.dtd">
<body>
<bodypart type="TEXT/PLAIN" href="http://qa150.example.com/wm/xml/v1/
    rfc822.txt?sessionid=2cea02183474ec897573d4188c0a0a70&amp;mailbox=attachme
    nts&amp;uid=1">RFC822 message</bodypart>
<bodypart type="APPLICATION/OCTET-STREAM" href="http://qa150.example.com/wm/
    mail/genimage/
    zz.c?sessionid=2cea02183474ec897573d4188c0a0a70&amp;uid=1&amp;off=1530&amp
    ;len=854&amp;enc=0&amp;type=APPLICATION&amp;sub=OCTET-STREAM">zz.c</
    bodypart>
<bodypart type="APPLICATION/MSWORD" href="http://qa150.example.com/wm/mail/
    genimage/
    Readme.doc?sessionid=2cea02183474ec897573d4188c0a0a70&amp;uid=1&amp;off=25
    17&amp;len=65252&amp;enc=1&amp;type=APPLICATION&amp;sub=MSWORD">Readme.doc
    </bodypart>
<bodypart type="IMAGE/GIF" href="http://qa150.example.com/wm/mail/genimage/
    bfly2.gif?sessionid=2cea02183474ec897573d4188c0a0a70&amp;uid=1&amp;off=678
    92&amp;len=2374&amp;enc=1&amp;type=IMAGE&amp;sub=GIF">bfly2.gif</bodypart>
<bodypart type="TEXT/HTML" href="http://qa150.example.com/wm/mail/genimage/
    addr.html?sessionid=2cea02183474ec897573d4188c0a0a70&amp;uid=1&amp;msgid=1
    &amp;off=70389&amp;len=13970&amp;enc=1&amp;type=TEXT&amp;sub=HTML">addr.ht
    ml</bodypart>
```

```
<bodypart type="TEXT/PLAIN" href="http://qa150.example.com/wm/mail/genimage/
    New+Text+Document.txt?sessionid=2cea02183474ec897573d4188c0a0a70&amp;uid=1
    &amp;off=84493&amp;len=1134&amp;enc=0&amp;type=TEXT&amp;sub=PLAIN">
    New+Text+Document.txt</bodypart>
<bodypart type="TEXT/HTML" href="http://qa150.example.com/wm/mail/genimage/
    gb2312.html?sessionid=2cea02183474ec897573d4188c0a0a70&amp;uid=1&amp;msgid
    =1&amp;off=85752&amp;len=3680&amp;enc=1&amp;type=TEXT&amp;sub=HTML">gb2312
    .html</bodypart>
</body>
```

bodystructure Command

The bodystructure command is used to fetch the body structure from the host for a particular message based on its message id or uid. The body that is returned is actually base64 encoded due to the presence of special characters such as parentheses and symbols like ">" and "<". So if body structure has to be used in the form which will be returned as imap returns, the return value of this command needs to be decoded from its base64 encoding.

This command is supposed to be used internally by WebMail.

Table 41 bodystructure Command Definition

Operation	`GET /wm/xml/v1/bodystructure.xml`	
Arguments	`sessionid`	ID of a valid session on this host
	`mailbox`	name of the mailbox where this message resides
	`msgid`	message ID of message (or...)
	`uid`	message user ID
Result	`OK element`	
	`Structure element`	

Result

OK Element - if parameters passed to the system are correct. Even though the msgid might not exist or mbox might not be found, the returned value will be OK only.

Structure Element - This might be empty if there is no body structure associated with the parameters passed to the command.

Possible Errors

```
<no>XML Invalid Session</no>
<no>XML IO Error</no>
```

Example: Response Returned from Server

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE bodystructure SYSTEM "http://qa18/dtd/xml/v1/bodystructure.dtd">
<bodystructure>
    <structure>KCJURVhUIiAiUExBSU4iICgiQ0hBUlNFVCIgInVzLWFzY2lpIikgTklMIE5JTCA
    iNOJJVCIgMjMwIDggTklMIE5JTCBOSUwp</structure>
</bodystructure>
```

Structure Element -Body Structure

The structure element returns the base64 encoded data for the associated body type structure.

compose Command

The compose command performs the operations needed to send a message with attachments. Because of the large nature of the arguments of this command, Mirapoint recommends that MIME encoded messages be used to provide arguments for this action.

Table 42 compose Command Definition

Operation	`POST /wm/xml/v1/compose.xml`	
Arguments	`sessionid`	session ID
	`op`	mailbox operation (send | draft)
	`to`	To field recipients
	`cc`	Cc field recipients
	`bcc`	Bcc field recipients
	`subject?`	message subject
	`message?`	message body
	`attach*`	attachment name
	`savesent+`	save in Sent folder (on)
	`includesig+`	include signature in message (on)
	`msg_priority+`	message priority
Result	`BAD element`	
	`OK element`	
	`NO element`	

Notes

- to, cc, and bcc can include multiple comma-separated recipients
- savesent is meaningless if composing a draft document
- savesent and includesig should have the value "on" when used. In a browser, they would be equivalent to checkboxes
- msg_priority can be any value from one through three or one through five, depending on system configuration

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
<no>Folder already exists</no>
```

```
<no>SMTP is not running</no>
<no>Error composing document</no>
<no>Address book timed out</no>
<no>Missing file in Attachment: field</no>
<no>Missing or empty file in Attachment: field</no>
<no>Attachment already exists</no>
<no>Mail host timed out</no>
<no>No recipients specified</no>
<bad>Invalid op value</bad>
<bad>Use a comma (,) between addresses</bad>
<bad>Too many recipients</bad>
```

Example

```
C: POST /wm/xml/v1/compose.xml HTTP/1.1
C: Host: host1
C: Content-Length: 14342
C: Content-Type: multipart/form-data; boundary=xxx
C:
C: --xxx
C: Content-Disposition: form-data; name="sessionid"
C:
C: 14324
C: --xxx
C: Content-Disposition: form-data; name="op"
C:
C: send
C: --xxx
C: Content-Disposition: form-data; name="to"
C:
C: Ben Franklin <ben@domain.com>
C: --xxx
C: Content-Disposition: form-data; name="to"
C:
C: George Washington <george@domain.com>
C: --xxx
C: Content-Disposition: form-data; name="subject"
C:
C: How about lunch
C: --xxx
C: Content-Disposition: form-data; name="attach"
C:
C: menu.html
C: --xxx
C: Content-Disposition: form-data; name="attach";
C:
C: menu.gif
C: --xxx
C: Content-Disposition: form-data; name="menu.html"; filename="menu.html"
C: Content-Type: text/html
C:
C: <html>
C: ...
C: </html>
C: --xxx
C: Content-Disposition: form-data; name="menu.gif"; filename="menu.gif"
C: Content-Type: image/gif
C:
C: GIF BINARY DATA
C: ...
C: --xxx
C: Content-Disposition: form-data; name="message"
C:
C: What about lunch today?
```

```
C: --xxx
C: Content-Disposition: form-data; name="savesent"
C:
C: on
C: --xxx--
```

Sample Response

```
<?xml version="1.0"?>
<!DOCTYPE ok SYSTEM "http://qa150.example.com/dtd/xml/v1/ok.dtd">
<ok>Successfully composed document</ok>
```

expunge Command

The expunge command expunges the specified mailbox. The action may fail if the user doesn't have write access to the mailbox.

Table 43 expunge Command Definition

Operation	POST /wm/xml/v1/expunge.xml	
Arguments	sessionid	session ID
	mailbox	mailbox name
Result	BAD element	
	OK element	
	NO element	

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
<no>Folder not expunged</no>
<no>Permission denied</no>
<bad>Folder does not exist</bad>
```

Example: Valid Response

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE ok SYSTEM "http://qa18.example.com/dtd/xml/v1/ok.dtd">
<ok>Folder expunged</ok>
```

index Command

The index command returns message summary information for zero or more messages in a mailbox.

Table 44 index Command Definition

Operation	GET /wm/xml/v1/index.xml

Table 44 index Command Definition

Arguments	sessionid	session ID
	mailbox	mailbox name
	msgids*	message ID (or...)
	uids*	message user ID (or...)
	msgranges*	msgid range
Result	BAD element	
	NO element	
	Index-List element	

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
<bad>Folder does not exist</bad>
<bad>Invalid unique message id</bad>
<bad>Invalid message id</bad>
<bad>Invalid message range</bad>
<bad>Operation is not supported on folder</bad>
```

Index Element

The index element returns attributes associated with a message. This includes the From, Subject, and Date fields. Also returned are the flags associated with a message.

Example

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE indexlist SYSTEM "http://qa18.example.com/dtd/xml/v1/indexlist.dtd">
    <indexlist>
    . <index>
    . . <msgid>1</msgid>
    . . <uid>1</uid>
    . . <date>20010719135959</date>
    . . <subject>ALERT! qa18.example.com (00902773acea)</subject>
    . . <size>2193</size>
    . . <from>qa18.example.com Monitor System
    &lt;Administrator@qa18.example.com&gt;</from>
    . . <tolist>
    . . .<to>administrator@qa18.example.com</to>
    . . </tolist>
    . . <cclist>
    . . . <cc>Joe Bob &lt;joe&gt;</cc>
    . . . <cc>Billy Joe &lt;billy&gt;</cc>
    . . </cclist>
    . . <seen/>
    . </index>
    . <index>
    . . <msgid>2</msgid>
    . . <uid>3</uid>
    . . <date>20010720002053</date>
    . . <subject>Message Log Email</subject>
    . . <size>595</size>
```

```
. . <from>qa18.example.com Monitor System
&lt;Administrator@qa18.example.com&gt;</from>
. . <tolist>
. . . <to>administrator@qa18.example.com</to>
. . . <to>other@example.com</to>
. . </tolist>
. . <deleted/>
. . <seen/>
. </index>
</indexlist>
```

- msgid - message ID
- uid - uid of message
- date - received date (in YYYYMMDDHHMMSS)
- subject - message subject
- size - size of message (in bytes)
- from - sender of message
- tolist - list of To: field recipients
- cclist - list of Cc: field recipients
- bcclist - list of Bcc: field recipients
- deleted - deleted flag
- seen - seen flag
- flagged - flagged flag
- answered - answered flag
- draft - draft flag

login Command

The login command performs login and logout operations for a user. Even though some users might not require a password to login, the password entry must be present in the URI and contain an empty string or the "Missing password" error document will be returned.

Table 45 login Command Definition

Operation	`POST /wm/xml/v1/login.xml`	
Arguments	`user`	User's login name
	`password`	User's password
Result	`NO element`	
	`SID element`	

Possible Errors

```
<no>System I/O error</no>
<no>Missing user entry</no>
<no>Invalid user</no>
<no>Authentication failed</no>
```

```
<no>Missing password</no>
<no>Unable to open mailbox</no>
<no>Can't contact LDAP</no>
```

There is no logout.xml. For logout, you can use login.xml as a GET operation, with session ID.

Example: Valid Response

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE sid SYSTEM "http://qa150.example.com/dtd/xml/v1/login.dtd">
<sid>41aa01f377bdac9bd5a564fbac741750</sid>
```

SID Element - Session ID

The SID element returns the session ID after a successful login.

HTTP Redirection On Login

If login action is sent to a host on a multiple server farm, an HTTP redirection can result via a 302 (Moved Temporarily) HTTP result.

Example

```
C: POST /wm/xml/v1/login.xml?user=joe&password=cool HTTP/1.1
C: Host: host0
C: Content-Length: 0
C:
S: HTTP/1.1 302 Moved Temporarily
S: Content-Length: 0
S: Location: http://host1/wm/xml/v1/login.xml?sessionid=23432
S:
```

mailbox Command

The mailbox command performs mailbox add and delete operations on a user's mailbox.

Table 46 mailbox Command Definition

Operation	`POST /wm/xml/v1/mailbox.xml`	
Arguments	`sessionid`	session ID
	`mailbox`	mailbox name
	`op`	mailbox operation (add | delete)
Result	`BAD element`	
	`OK element`	
	`NO element`	

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
<bad>Operation is not supported on folder/bad>
<bad>Invalid mailbox name</bad>
```

```
<bad>Permission denied</bad>
<no>Folder already exists</no>
<no>Folder not deleted</no>
<no>Folder does not exist</no>
<no>Invalid argument</no>
```

Examples: Valid Responses

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE ok SYSTEM "http://qa18.example.com/dtd/xml/v1/ok.dtd">
<ok>Folder added</ok>
```

or

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE ok SYSTEM "http://qa18.example.com/dtd/xml/v1/ok.dtd">
<ok>Folder deleted</ok>
```

mailboxlist Command

The mailboxlist command returns a user's list of mailboxes. For each mailbox, its name, number of message, and number of unread messages are returned.

Table 47 mailboxlist Command Definition

Operation	`GET /wm/xml/v1/mailboxlist.xml`	
Arguments	`sessionid`	session ID
	`counts`	sends message and unread counts (yes | no) (optional)
Result	`NO element`	
	`Mailboxlist element`	

If `counts` is `no` or omitted, <unread> and <count> are not transmitted.

Possible errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
```

Example

```
http://mira/wm/xml/v1/mailboxlist.xml?sessionid=sID&counts=yes
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE mailboxlist SYSTEM "http://qa150.example.com/dtd/xml/v1/
    mailboxlist.dtd">
    <mailboxlist>
    . <mailbox>
    . . <name>Inbox</name>
    . . <unread>358</unread>
    . . <count>378</count>
    . </mailbox>
    . <mailbox>
    . . <name>attachments</name>
    . . <unread>6</unread>
    . . <count>6</count>
    . </mailbox>
    </mailboxlist>
```

Mailboxlist Element

The mailboxlist element returns a list of mailbox names and their attributes.

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
```

Example

```
<mailboxlist>
    <mailbox>
    . <name>INBOX</name>
    . <unread>4</unread>
    . <count>100</count>
    </mailbox>
    <mailbox>
    . <name>Draft</name>
    . <unread>0</unread>
    . <count>3</count>
    </mailbox>
    <mailbox>
    . <name>Sent</name>
    . <unread>2</unread>
    . <count>34</count>
    </mailbox>
</mailboxlist>
```

- name - the name of the mailbox
- unread - number of unread messages
- count - number of the message in the mailbox

preferences Command (GET)

The preferences get command returns the current state of the user's mail preferences.

Table 48 preferences Command (GET) Definition

Operation	`GET /wm/xml/v1/prefs.xml`	
Arguments	`sessionid`	session ID
Result	`BAD element`	
	`NO element`	
	`Preference element`	

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
<no>Unable to open preferences</no>
```

Example

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!DOCTYPE preferences SYSTEM "http://qa150.example.com/dtd/xml/v1/
    preferences.dtd”>
<preferences>
    <fullname>George Harrison</fullname>
    <email>george@example.com</email>
    <replyto>george@example.com</replyto>
    <messagecnt>
    . <value>25</value>
    . <default>20</default>
    </messagecnt>
    <composewidth>
    . <value>60</value>
    . <default>62</default>
    </composewidth>
    <composeheight>
    . <value>20</value>
    . <default>15</default>
    </composeheight>
    <sentfolder>Sent</sentfolder>
    <savesent>
    . <value>yes</value>
    . <default>yes</default>
    . <optionlist>
    . . <option>no</option>
    . . <option>yes</option>
    . </optionlist>
    </savesent>
    <replyopt>
    . <value>includeinline</value>
    . <default>dontinclude</default>
    . <optionlist>
    . . <option>dontinclude</option>
    . . <option>includeinline</option>
    . . <option>includeattach</option>
    . </optionlist>
    </replyopt>
    <signature>This is my signature
    </signature>
    <includesig>
    . <value>no</value>
    . <default>no</default>
    . <optionlist>
    . . <option>no</option>
    . . <option>yes</option>
    . </optionlist>
    </includesig>
    <version>
    . <value>javascript</value>
    . <default>nojavascript</default>
    </version>
    <timezone>Pacific/Apia</timezone>
    <charset>
    . <value>utf-8</value>
    . <default>utf-8</default>
    <optionlist>
    <option>EUC-KR</option>
    <option>Big5</option>
    <option>GB2312</option>
    <option>UTF-8</option>
    <option>ISO-2022-JP</option>
    <option>TIS-620</option>
    </optionlist>
    </charset>
    <draftfolder>Draft</draftfolder>
```

```
    <junkfolder>Junk Folder</junkfolder>
    <trashfolder>Trash</trashfolder>
    <usetrash>
    <value>yes</value>
    <default>no</default>
    <optionlist>
    <option>no</option>
    <option>yes</option>
    </optionlist>
    </usetrash>
</preferences>
```

preferences Command (POST)

The preferences post command sets the specified values for a user's mail preferences. Only the specified values are set by the action.

Table 49 preferences Command (POST) Definition

Operation	`POST /wm/xml/v1/prefs.xml`	
Arguments	`sessionid`	session ID
	`fullname+`	user's full name
	`email+`	user's email address
	`reply-to+`	reply-to value
	`header+`	number of messages to display in WebMail TOC
	`composewidth+`	width, in characters, of the Compose message field
	`composeheight+`	height, in lines, of the Compose message field
	`sentfolder+`	Sent folder name
	`replyopt+`	default replying option for WebMail
	`signature+`	signature for outgoing mail
	`includesig+`	default compose option for including signature
	`version+`	UI version of WebMail
	`timezone+`	user's timezone
	`charset+`	default outgoing message charset
	`draftfolder+`	name of the draft folder
	`junkfolder+`	name of the junk mail folder
	`trashfolder+`	name of the trash mail folder
	`usetrash+`	deleted messages are moved to trashfolder
Result	`BAD element`	
	`NO element`	
	`OK element`	

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
<no>Preferences could not be saved</no>
<no>Unable to open preferences</no>
<bad>Folder already exists</bad>
<bad>Folder name too long.</bad>
<bad>Invalid fullname value</bad>
<bad>Invalid email value</bad>
<bad>Invalid reply-to value</bad>
<bad>Invalid messagecnt value</bad>
<bad>Invalid composewidth value</bad>
<bad>Invalid composeheight value</bad>
<bad>Invalid sentfolder value</bad>
<bad>Invalid savesent value</bad>
<bad>Invalid replyopt value</bad>
<bad>Invalid signature value</bad>
<bad>Invalid includesig value</bad>
<bad>Invalid version value</bad>
<bad>Invalid timezone value</bad>
<bad>Invalid charset value</bad>
<bad>Invalid draftfolder value</bad>
<bad>Invalid junkfolder value</bad>
<bad>Invalid trashfolder value</bad>
<bad>Invalid usetrash value</bad>
```

Example of Valid Operation

```
<?xml version="1.0"?>
<!DOCTYPE ok SYSTEM "http://qa150.example.com/dtd/xml/v1/ok.dtd">
<ok>Preferences saved</ok>
```

RFC822 Command

The rfc822 command returns the raw RFC822 message contents of a message. Because the RFC822 contents is of type text/plain, the result of this command is not an XML document, but a plain text document.

Table 50 RFC822 Command Definition

Operation	`GET /wm/xml/v1/rfc822.txt`	
Arguments	`sessionid`	session ID
	`mailbox`	mailbox name
	`msgid`	message ID (or...)
	`uid`	message user ID
Result	`BAD element`	
	`NO element`	
	`RFC822 element`	

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
<bad>Folder does not exist</bad>
```

```
<bad>Invalid unique message id</bad>
<bad>Invalid message id</bad>
<bad>Missing uid or msgid value</bad>
```

Example

```
Return-Path: <Administrator@qa18.example.com>
Received: (from Administrator@localhost)
by qa18.example.com (Mirapoint)
id AAA00001;
Tue, 17 Jul 2001 14:12:52 GMT
Date: Tue, 17 Jul 2001 14:12:52 GMT
Message-Id: <200107171412.AAA00001@qa18.example.com>
From: "qa18.example.com Monitor System" <Administrator@qa18.example.com>
Subject: ALERT! qa18.example.com (00902773acea)
To: administrator@qa18.example.com

Version 3.0.0.17004-gberthet-1-200106191822

The following conditions are outstanding:
SYSTEM.SMTPL: SMTP service is not running
Contact information: ...
---End of Example---
```

search Command

The search command returns an index list of messages that match a search query.

Table 51 search Command Definition

Operation	`GET /wm/xml/v1/search.xml`	
Arguments	`sessionid`	session ID
	`mailbox`	mailbox name
	`fromsearch+`	search in from field
	`subjectsearch+`	search in subject field
	`toccsearch+`	search in to/cc field
	`bodysearch+`	search in body field
	`largersearch+`	match message larger than value in kilobytes
	`smallersearch+`	match message smaller than value in kilobytes
	`unreadsearch`	search unread only (on)
	`msgids*`	search in message ID (or...)
	`uids*`	search in message uid (or...)
	`msgranges*`	search in msgid range
Result	`BAD element`	
	`NO element`	
	`Index-List element`	

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
<bad>Invalid mailbox name</bad>
<bad>Invalid unique message id</bad>
<bad>Invalid message id</bad>
<bad>Invalid message range</bad>
```

Example: When No Messages Match Search Criteria

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE indexlist SYSTEM "http://qa150.example.com/dtd/xml/v1/
    indexlist.dtd”>
<indexlist>
</indexlist>
```

setflags Command

The setflags command sets the flags for a given set of messages. The flags that can be set are the seen and deleted flags.

Table 52 setflags Command Definition

Operation	POST /wm/xml/v1/setflags.xml	
Arguments	sessionid	session ID
	mailbox	mailbox name
	op	operation (read } unread | delete | undelete)
	msgids*	message ID (or...)
	uids*	message user ID (or...)
	msgranges*	message ID range
Result	BAD element	
	OK element	
	NO element	

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
<bad>Invalid operation</bad>
<bad>Invalid mailbox name</bad>
<bad>Invalid unique message id</bad>
<bad>Invalid message id</bad>
<bad>Invalid message range</bad>
<bad>Folder does not exist</bad>
```

Example: Valid Response

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE ok SYSTEM "http://qa18.example.com/dtd/xml/v1/ok.dtd">
<ok>Successfully set message flags</ok>
```

status Command

The status command returns the same mailbox information as the mailbox list command, but only for the specified mailbox. Instead of returning a mailbox list element, a single mailbox element is returned.

Table 53 status Command Definition

Operation	GET /wm/xml/v1/status.xml	
Arguments	sessionid	session ID
	mailbox	mailbox name
Result	BAD element	
	NO element	
	Mailboxlist element	

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
<bad>Invalid mailbox name</bad>
```

Example

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE status SYSTEM "http://qa18.example.com/dtd/xml/v1/status.dtd">
    <status>
    . <mailbox>
    . . <name>inbox</name>
    . . <unread>178</unread>
    . . <count>178</count>
    . . <quota_used>579550504667942688<quota_used>
    . . <quota_limit>584047382535077888<quota_limit>
    . </mailbox>
    </status>
```

Status Element

The status element returns a single mailbox name and its attributes.

- name - name of the mailbox
- unread - number of unread messages
- count - number of the message in the mailbox
- quota_used - quota (in bytes) used up by the user.
- quota_limit - quota (in bytes) assigned to the user.

transfer Command

The transfer command moves or copies message from a source mailbox to a destination mailbox. When the move operation is specified, the messages in the

source mailbox have the deleted flags set, and are not removed until the expunge action is executed.

Table 54 transfer Command Definition

Operation	POST /wm/xml/v1/transfer.xml	
Arguments	sessionid	session ID
	mailbox	mailbox name
	dstmbox	destination mailbox name
	msgids*	message ID (or...)
	uids*	message uid (or...)
	msgranges*	msgid range
	op	mailbox operation (move | copy)
Result	BAD element	
	OK element	
	NO element	

Possible Errors

```
<no>System I/O error</no>
<no>Your session has timed out</no>
<bad>Folder does not exist</bad>
<bad>Invalid unique message id</bad>
<bad>Invalid message id</bad>
<bad>Invalid message range</bad>
```

Example

```
<?xml version="1.0"?>
<!DOCTYPE ok SYSTEM "http://qa150.example.com/dtd/xml/v1/ok.dtd">
<ok>Transfer succeeded</ok>
```

4 XML Interface to the WebMail Address Book

The Mirapoint XML interface to the WebMail Address Book lets customers write their own address book user-interface applications, and is helpful when developing synchronization tools, such as Mirapoint's SynQ add-in for Microsoft Outlook.

The interface provides functions to:

- Retrieve contacts and groups from the address book, using criteria such as uids, range of indexes, categories, and so on
- Get the list of categories
- Create new contacts and new groups
- Delete existing contacts and groups
- Import or export address books in LDIF format
- Search the address book

The XML address book operations are described in "Commands" on page 88.

Command Parameters

In the description of the XML operations, these parameters are common:

`sid`

Session ID obtained after a successful login operation.

`ref`

The ID of the contact on which the operation is performed. For operations that return lists of objects, there might be several occurrences of this parameter. The value corresponds to the contact's unique ID (uuid).

`group/category`

Contacts are referred to using their uuid. For groups and categories, it is enough to use the name. This parameter is used either to operate on a given group or category, or to restrict the retrieval of contacts.

`letter`

Similar to `group` or `category`, `letter` can be used to restrict the retrieval to contacts that begin with a given letter. It can be used with `category` or `group`.

`offset/count`

Instead of using multiple instances of ref parameters, a set of objects on which an operation is performed can be specified as a range of contact indexes. This is valid when accessing a sorted list of contacts in a category (including the "all" category), in a group, or in a set of contacts starting with the same letter. The parameter offset contains the first index, it defaults to the first record available in the given set. It is an error if the offset is out of bounds. The parameter count contains the maximum count of records to be returned. These parameters are optional but if supplied, must be both supplied. count can be set to * to denote all the available contacts starting at index offset.

The special value * can be used for parameters that can be multiple, such as `ref`, `group`, or `category`. It denotes all the existing elements (all contacts, all groups, all categories).

Some other parameters, like `charset` or `language`, accept enumerated values depending on the system configuration. To know exactly which values are available for a given parameter, use the XML operation `prefs`.

Commands

An address book XML operation has the same meaning and result whether accessed with a POST or GET method, except for the `import` operation for which one of the parameters is a file.

An operation either completely succeeds or completely fails. For instance, when deleting a list of contacts, the deletion occurs only if all the given IDs are valid. When modifying a contact, the contact is modified only if each value supplied is valid for the given field.

The following descriptions indicate if each parameter is single or multiple and mandatory or optional. Optional parameters are usually single. If the operation name is plural, such as `get_contacts`, `del_groups`, the operation can accept a multiple `ref` or `group` parameter. If the operation name is singular, such as `add_contact` or `mod_group`, the parameter is mandatory and single. `Add` operations can accept a wildcard parameter (such as *ref*=*) but `delete` operations cannot.

The sessionid parameter `sid` is always mandatory and single.

category Commands

This section describes the category commands get_categories, add_category, mod_category, del_categories, and get_letter_categories, followed by the category DTD.

get_categories Command

The get_categories command returns the list of categories.

Table 55 get_categories Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/get_categories.xml`

Table 55 get_categories Command Definition

Arguments	`sid`	session ID (required)
	`category`	category name (multiple)

Result

The `categories` element is returned with the description of the existing categories.

add_category Command

The add_category command creates a new category.

Table 56 add_category Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/add_category.xml`	
Arguments	`sid`	session ID (required)
	`category`	name of the category to create (required)

Result

A `category` element is returned if the category was successfully created. If not, a `status` element is returned to indicate the failure of the request.

mod_category Command

The mod_category command modifies an existing category. A modification is either a change of name, the addition of a contact to a category (in which case the contact is removed from its current category) or the removal of a contact (which is equivalent to the addition to the unfiled category). The unfiled category cannot be renamed.

Table 57 mod_category Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/mod_category.xml`	
Arguments	`sid`	session ID (required)
	`category`	name of the category to rename (required)
	`modtype`	rename, add, or remove (required)
	`newname`	new name of the category
	`ref`	contact ID (multiple)

Result

A `category` element is returned if the category was successfully modified. If not, a `status` element is returned to indicate the failure of the request.

del_categories Command

The del_categories command deletes one or more categories. It is an error to delete the unfiled category, in which case no category at all is deleted. Using * as the value of `category` will delete all categories except *Unfiled*.

Table 58 del_categories Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/del_categories.xml`	
Arguments	`sid`	session ID (required)
	`category`	name of the category to delete (multiple)

Result

A `status` element is returned to indicate the success or failure of the request.

get_letter_categories Command

The get_letter_categories command lets a user know which letters have contacts in the address book, and how many contacts. It can be restricted to a given category.

The first letter in a contact name is the first character in the string composed by the last name, first name, nickname and email address, in this order.

The term `letter` encloses "A" to "Z" (same as "a" to "z"), "0" to "1" (for the ten digits), and eventually other characters.

Table 59 get_letter_categories Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/get_letter_categories.xml`	
Arguments	`sid`	session ID (required)
	`category`	count only contacts in this category (optional)

Result

A `letters` element is returned with the list of letters and their contact counts.

contact Commands

This section describes the contact commands get_contacts, add_contact, mod_contact, and del_contacts, followed by the contact DTD.

get_contacts Command

The get_contacts command returns a list of contacts.

Table 60 get_contacts Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/get_contacts.xml`

Table 60 get_contacts Command Definition

Arguments	sid	session ID (required)
	ref	contact ID (multiple)
	group	group name (optional)
	category	category name (optional)
	letter	contact's name first letter (optional)
	offset	first contact index (optional)
	count	maximum count (optional)
	lastmodtime	last modification time (optional)
	detail	full or partial (optional; default is full)
	showdeleted	true or false (optional)

Result

The list of contacts is returned into a persons element. In case of failure, the reason is given with a status element. The contacts can be selected using their unique ID, supplied with the parameters ref, and/or with the group and/or the category they belong to and/or with their first letter. The selection can further be restricted using the two parameters offset and count.

The lastmodtime parameter is used to retrieve only the contacts that were modified after a given date. The format of this parameter is: YearMonthDayTHourMinutesSecondsZ (format: %Y%m%dT%H%M%SZ).

Depending on the value of detail, which defaults to full, a limited set of field values is returned per contact, enough to get a contact's summary and to use this command again to retrieve all the fields (with detail unset or set to full) using the same operation, if needed. If the value is partial and lastmodtime was supplied, only the fields that were modified are returned.

For the category element, the index of the contact's category is given. If the contact belongs to the *Unfiled* category, the attribute unfiled is present.

The element that contains the primary phone number contains the attribute primary. The element primaryphone contains the name of the field that holds the primary phone number.

If showdeleted is supplied with the value true, the uuids of contacts that were deleted (eventually since lastmodtime only) are returned using deleted-person elements. The first time it is used, it also notifies the address book to start tracking the deleted contacts.

```
<persons count="1">
    <person ref="123-2344-34455">
    . <category unfiled="yes">3</category>
    . <cn>Joe T. User</cn>
    . <surname>User</surname>
    . <givenname>Joe</givenname>
    . <displayname>Joe User</displayname>
```

```
    . <mail>juser@example.com</mail>
    . <telephonenumber primary="yes">+1 408.720.9999</teleph
    . <primaryphone>telephonenumber</primaryphone>
    </person>
</persons>
```

add_contact Command

The add_contact command creates a new contact. The field names are the name of the elements that a `person` element can contain (see the contact DTD in Appendix A, Mirapoint DTDs"). They are all optional but at least one of *surname*, *givenname*, *nickname*, and *mail* must have a non-empty value. If the contact conflicts with an entry in the address book, the value of `conflict` indicates how to resolve the conflict, either by generating an error (using the `status` element), replacing the existing entry, or creating a duplicate entry (and ignoring the conflict).

Table 61 add_contact Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/add_contact.xml`	
Arguments	`sid`	session ID (required)
	`fieldname`	field value
	`conflict`	discard, error, replace, or ignore (default)

Result

If the contact was successfully created, the corresponding `person` element is returned. If not, a `status` element indicates the failure.

mod_contact

The mod_contact command modifies an existing contact. A new value for a field is given using the field name as a parameter. The field names are the name of the elements that a `person` element can contain (see the contact DTD in Appendix A, Mirapoint DTDs").

Table 62 mod_contact Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/mod_contact.xml`	
Arguments	`sid`	session ID (required)
	`ref`	contact ID (required)
	`fieldname`	field value

Result

If the contact was successfully modified, the corresponding `person` element is returned. If not, a `status` element is returned.

```
<person ref="123-2344-34455">
    <category>2</category>
    <cn>Joe T. User</cn>
    <surname>user</surname>
    <givenname>Joe</givenname>
```

```
    <displayname>Joe User</displayname>
    <mail>juser@example.com</mail>
    <telephonenumber primary>+1 408.720.9999</telephonenumbe
    <primaryphone>telephonenumber</primaryphone>
</person>
```

del_contacts Command

The del_contacts command deletes one or more contacts. An optional category can be supplied to restrict the deletion to contacts that belong to this category.

Table 63 del_contacts Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/del_contacts.xml`	
Arguments	`sid`	session ID (required)
	`ref`	contact ID (multiple)
	`category`	category restricting deletion (optional)

Result

A `status` element indicates the result of the operation.

group Commands

This section describes the group commands get_groups, add_group, mod_group, and del_groups, followed by the group DTD.

get_groups Command

The get_groups command returns a list of groups. The members of a group are given in the output of this operation but can also be retrieved using `get_contacts` on page 90.

Table 64 get_groups Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/get_groups.xml`	
Arguments	`sid`	session ID (required)
	`group`	group name (multiple)

Result

The list of groups is returned as a `groupsofnames` element. An error in the operation is signified with a `status` element.

add_group Command

The add_group command creates a new group.

Table 65 add_group Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/add_group.xml`

Table 65 add_group Command Definition

Arguments	`sid`	session ID (required)
	`group`	group name (required)

Result

A `groupofnames` element is returned if the group has been successfully created. In case of failure, the reason is given in a **status** element.

mod_group Command

The mod_group command modifies an existing group. The type of modification can be:

- `rename` to rename the group
- `add` to add members
- `remove` to remove members

One command performs only one type of modification. For `add` or `remove`, there must be at least one `ref`. For `remove`, it can be `*`.

Table 66 mod_group Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/mod_group.xml`	
Arguments	`sid`	session ID (required)
	`group`	current group name (required)
	`modtype`	rename, add, or remove (required)
	`newname`	new name of the group
	`ref`	contact ID (multiple)

Result

If the group was successfully modified, the corresponding `groupofnames` element is returned. A `status` element is returned to indicate the failure of the request.

del_groups Command

The del_groups command deletes one or more groups.

Table 67 del_groups Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/del_groups.xml`	
Arguments	`sid`	session ID (required)
	`group`	group name (multiple)

Result

A `status` element is returned to indicate the success or failure of the request.

import/export Commands

This section describes the export and import commands.

export Command

The export command exports the address book, or part of it, in LDIF or CSV format, either as an XML document or as raw data with corresponding HTTP headers.

Table 68 export Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/export.xml`
	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/export.ldif`
	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/export.csv`	

Arguments	`sid`	session ID (required)
	`format`	ldif or csv (required only for export.xml)
	`charset`	character set of the file (defaults to UTF-8)
	`language`	language of the CSV file (default depends on locale)
	`category`	export only the contents of one category (defaults to all)
	`cdata`	if true (default), export as a CDATA section in an XML document; if false, export as a plan LDIF or CSV file

Result

The xml format is:

```
<export file="addrbook.ldif">
    ... raw data in <![CDATA[ - ]]> section ...
</export>
```

The LDIF and CSV formats are:

```
Content-Type: x-application/octet-stream; name=addrbook.ld
Content-Disposition: inline; filename=addrbook.ldif
Content-Length: 1234

... raw data ...
```

import Command

The import command imports the contents of an LDIF or CSV file into the address book. Conflicts can be:

- `ignore`–contacts are created and groups are replaced
- `discard`–new conflicting records are ignored

- `replace`–existing conflicting contacts and/or groups are deleted and the new records are created

Table 69 import Command Definition

Operation	`POST /cgi-bin/addrbook.cgi/xab/v1/import.xml`	
Arguments	`sid`	session ID (required)
	`format`	ldif or csv (required)
	`file`	data to import (required)
	`charset`	character set of the file (defaults to UTF-8)
	`language`	language of the CSV file
	`category`	category the file is imported into (defaults to unfiled)
	`conflict`	discard, error, replace, or ignore (default)

Result

The `import` element indicates the number of records found in the file and the number of contacts and groups actually created.

```
<import file="mirapoint.ldif">
    <recordcount>1004</recordcount>
    <personcount>1000</personcount>
    <groupcount>4</groupcount>
</import>
```

preferences Command

This section describes the preferences command.

prefs Command

The prefs command lets an automatic application query the XML interface for the list of accepted values for a given parameter in the XML operations previously described. The value used as a default, if any, can also be indicated.

Table 70 prefs Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/prefs.xml`	
Arguments	`sid`	session ID (required)
	`option`	option name (optional, multiple)

If `option` is not supplied, all the parameters are described. If supplied, it can have these values:

`charset`

The list of character sets that the application recognizes. The default is UTF-8.

`language`

The application know how to import CSV files produced by localized versions of Outlook for these languages.

`format`

The format of address book export: currently LDIF and CSV.

`fieldname`

The names of contact's fields.

`conflict`

The accepted values for this parameter.

`detail`

The accepted values for this parameter.

`modtype`

The accepted values for this parameter.

`primaryphone`

The accepted values for this parameter and for the corresponding contact field.

Result

```
<preferences>
    <charset>
    . <default>utf-8</default>
    . <optionlist>
    . . <option>big5</option>
    . . <option>iso-8859-1</option>
    . . <option>shift_jis</option>
    . . <option>us-ascii</option>
    . . <option>euc-jp</option>
    . . <option>euc-kr</option>
    . . <option>gb2312</option>
    . . <option>iso-2022-jp</option>
    . . <option>iso-2022-kr</option>
    . . <option>utf-8</option>
    . </optionlist>
    </charset>
</preferences>
```

search Command

This section describes the search command.

search Command

The search command searches the address book for contacts whose fields match given patterns.

The search can also apply to groups, to find groups with a given name or containing a specific member. `memberref` is the only parameter that is not a pattern: if supplied, it must contain the exact uuid of a contact.

The contacts returned match all the supplied patterns, like a logical AND. Same with the groups if `grouppattern` and `memberref` are both supplied.

Table 71 search Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/search.xml`	
Arguments	`sid`	session ID (required)
	`namepattern`	pattern for all the name fields
	`phonepattern`	pattern for all the phone number fields
	`addresspattern`	pattern for all the address fields
	`otherpattern`	pattern for all other fields
	`grouppattern`	pattern for group names
	`memberref`	contact ID for group search
	`fieldname``pattern`	pattern for the *fieldname* field
	`detail`	full or partial (optional)

Result

The search results are returned inside a `searchresult` element that contains a `search` element that lists the patterns that were supplied for the search, a list of `person` elements, for each matching contact, inside an optional `persons` element and zero or more `groupofnames` elements, for each matching group inside an optional `groupsofnames` element.

If the parameter `detail` has the value `partial` (which is its default), contacts are returned with a limited set of fields (enough to display a contact summary and access the remaining fields with get_contacts).

version Command

This section describes the version command.

version Command

The version command lets a client query for the server's MOS and XML versions. The command does not take any argument, and does not require a valid session ID.

Table 72 version Command Definition

Operation	`GET|POST /cgi-bin/addrbook.cgi/xab/v1/version.xml`	
Arguments	`none`	

Result

```
<version>
    <interface>XML version</interface>
    <mos>MOS version</mos>
</version>
```

A

Mirapoint DTDs

This appendix contains the DTD for each XML API in this manual.

Status DTD

The status DTD is used for replies that need to indicate success or failure, only. Each XML subsystem has a different status DTD. See below for details.

WebCal Group Calendar DTD

WebCal Group Calendar has twelve DTDs, each presented in a section below.

calendar.dtd

```
<!ELEMENT calendar (no | bad | (ok, (event | todo)*))>

<!ELEMENT event (eventid, globalid, clientid?, eventtitle, eventdesc?,
        eventstart, eventstop, viewablestart, viewablestop,
        allday?, eventpriority, eventemaildiff, eventpagerdiff,
        eventnotifylist?, eventrrule?, eventxdata?, created, dtstamp,
        last-modified, sequence, status?, x-mira-sendmail?,
        x-mira-readonly?, attach*, attendees?, extattendees?,
        resources?, perms*, exceptions?, timezoneoffset?, timezone?)>

<!ELEMENT eventid (#PCDATA)>
<!ELEMENT globalid (#PCDATA)>
<!ELEMENT clientid (#PCDATA)>
<!ELEMENT eventtitle (#PCDATA)>
<!ELEMENT eventdesc (#PCDATA)>
<!ELEMENT eventstart (#PCDATA)>
<!ELEMENT eventstop (#PCDATA)>
<!ELEMENT viewablestart (#PCDATA)>
<!ELEMENT viewablestop (#PCDATA)>
<!ELEMENT allday EMPTY>
<!ELEMENT eventpriority (#PCDATA)>
<!ELEMENT eventemaildiff (#PCDATA)>
<!ELEMENT eventpagerdiff (#PCDATA)>
<!ELEMENT eventnotifylist (eventnotifyelement*)>
<!ELEMENT eventrrule (#PCDATA)>
<!ELEMENT eventxdata (#PCDATA)>
<!ELEMENT attendees (attendee*)>
<!ELEMENT extattendees (extattendee*)>
<!ELEMENT resources (resource*)>
<!ELEMENT created (#PCDATA)>
<!ELEMENT dtstamp (#PCDATA)>
<!ELEMENT last-modified (#PCDATA)>
```

```
<!ELEMENT sequence (#PCDATA)>
<!ELEMENT status (#PCDATA)>
<!ELEMENT x-mira-sendmail EMPTY>
<!ELEMENT x-mira-deleted EMPTY>
<!ELEMENT x-mira-readonly EMPTY>
<!ELEMENT attach (extref)>
        <!ATTLIST attach name CDATA #REQUIRED>

<!ELEMENT eventnotifyelement (#PCDATA)>

<!ELEMENT attendee (#PCDATA)>
        <!ATTLIST attendee role CDATA #REQUIRED>
        <!ATTLIST attendee partstat CDATA #REQUIRED>

<!ELEMENT extattendee (#PCDATA)>
        <!ATTLIST extattendee partstat CDATA #REQUIRED>

<!ELEMENT resource (#PCDATA)>
        <!ATTLIST resource partstat CDATA #REQUIRED>

<!ELEMENT extref EMPTY>
        <!ATTLIST extref uri CDATA #REQUIRED>

<!ELEMENT todo (todoid, todotitle, tododesc?, todopriority,
        todoxdata?, created, dtstamp, last-modified, sequence,
        x-mira-readonly?, due?, completed?)>

<!ELEMENT todoid (#PCDATA)>
<!ELEMENT todotitle (#PCDATA)>
<!ELEMENT tododesc (#PCDATA)>
<!ELEMENT todopriority (#PCDATA)>
<!ELEMENT todoxdata (#PCDATA)>
<!ELEMENT due (#PCDATA)>
<!ELEMENT completed (#PCDATA)>

<!ELEMENT perms (permuser*)>
        <!ATTLIST perms scope CDATA #REQUIRED>
        <!ATTLIST perms type CDATA #REQUIRED>

<!ELEMENT permuser (#PCDATA)>

<!ELEMENT exceptions (exception+)>

<!ELEMENT exception (#PCDATA)>
    <!ATTLIST exception deleted (true | false) #IMPLIED>

<!ELEMENT eventlocation (#PCDATA)>

<!ELEMENT timezoneoffset (#PCDATA)>

<!ELEMENT timezone (#PCDATA)>

<!ELEMENT sequence-init (#PCDATA)>

<!ELEMENT client-lastmodified (#PCDATA)>

<!ELEMENT replytime (#PCDATA)>

<!ELEMENT master-clientid (#PCDATA)>

<!ELEMENT metadata (#PCDATA)>
        <!ATTLIST metadata name CDATA #REQUIRED>
        <!ATTLIST metadata value CDATA #REQUIRED>
```

```
<!ELEMENT ok (#PCDATA)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT bad (#PCDATA)>
```

changes.dtd

```
<!ELEMENT changes (no | bad |
           (ok, changedevent*, changedtodo*))>

<!ELEMENT changedevent (eventid, created, dtstamp,
           last-modified, sequence, x-mira-deleted?)>

<!ELEMENT changedtodo (todoid, created, dtstamp,
           last-modified, sequence, x-mira-deleted?)>

<!ELEMENT eventid (#PCDATA)>
<!ELEMENT todoid (#PCDATA)>
<!ELEMENT created (#PCDATA)>
<!ELEMENT dtstamp (#PCDATA)>
<!ELEMENT last-modified (#PCDATA)>
<!ELEMENT sequence (#PCDATA)>
<!ELEMENT x-mira-deleted EMPTY>

<!ELEMENT ok (#PCDATA)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT bad (#PCDATA)>
```

freebusy.dtd

```
<!ELEMENT freebusyreply (no | bad | (ok, vfreebusy))>

<!ELEMENT vfreebusy (dtstamp, dtstart, dtend, freebusy*)>

<!ELEMENT dtstamp (#PCDATA)>
<!ELEMENT dtstart (#PCDATA)>
<!ELEMENT dtend (#PCDATA)>
<!ELEMENT freebusy (#PCDATA)>

<!ELEMENT ok (#PCDATA)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT bad (#PCDATA)>
```

localelist.dtd

```
<!ELEMENT localelist (locale+)>

<!ELEMENT locale (default?, name, language)>
<!ELEMENT default EMPTY>
<!ELEMENT name (#PCDATA)>
<!ELEMENT language (#PCDATA)>
```

login.dtd

```
<!ELEMENT login (no | bad | (ok, sid, time, dumpcal?, emaillist))>

<!ELEMENT sid (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT dumpcal (#PCDATA)>
<!ELEMENT emaillist (email*)>
<!ELEMENT email (#PCDATA)>
        <!ATTLIST email type CDATA #IMPLIED>
```

```
<!ELEMENT ok (#PCDATA)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT bad (#PCDATA)>
```

permissions.dtd

```
<!ELEMENT permissions (no | (ok, (perms+)))>

<!ELEMENT perms (permuser*)>
        <!ATTLIST perms scope CDATA #REQUIRED>
        <!ATTLIST perms type (INCLUDE | PUBLIC) "INCLUDE">

<!ELEMENT permuser (#PCDATA)>

<!ELEMENT ok (#PCDATA)>
<!ELEMENT no (#PCDATA)>
```

preferences.dtd

```
<!ELEMENT preferences (no | bad |
         (ok, fullname, emaildiff, pagerdiff,
          emailsummarytime, emailsummarytype, pagersummarytime,
          pagersummarytype, weeklysummarytime, weeklysummaryday,
          monthlysummarytime, monthlysummaryday,
          emailaddr, pageraddr, dfltview,
          daystart, dayend, daygranularity, weekdaysep,
          showeventtxt, showtodolist, weekstart, version,
          timezone, published))>

<!ELEMENT fullname (#PCDATA)>
<!ELEMENT emaildiff (#PCDATA)>
<!ELEMENT pagerdiff (#PCDATA)>
<!ELEMENT emailsummarytime (#PCDATA)>
<!ELEMENT emailsummarytype (#PCDATA)>
<!ELEMENT pagersummarytime (#PCDATA)>
<!ELEMENT pagersummarytype (#PCDATA)>
<!ELEMENT weeklysummarytime     (#PCDATA)>
<!ELEMENT weeklysummaryday (#PCDATA)>
<!ELEMENT monthlysummarytime (#PCDATA)>
<!ELEMENT monthlysummaryday (#PCDATA)>
<!ELEMENT emailaddr (#PCDATA)>
<!ELEMENT pageraddr (#PCDATA)>
<!ELEMENT dfltview (#PCDATA)>
<!ELEMENT daystart (#PCDATA)>
<!ELEMENT dayend (#PCDATA)>
<!ELEMENT daygranularity (#PCDATA)>
<!ELEMENT weekdaysep (#PCDATA)>
<!ELEMENT showeventtxt (#PCDATA)>
<!ELEMENT showtodolist (#PCDATA)>
<!ELEMENT weekstart (#PCDATA)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT timezone (#PCDATA)>
<!ELEMENT published (#PCDATA)>

<!ELEMENT ok (#PCDATA)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT bad (#PCDATA)>
```

status.dtd

```
<!ELEMENT status (ok | no | bad)>
```

```
<!ELEMENT ok (#PCDATA)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT bad (#PCDATA)>
```

update.dtd

```
<!ELEMENT update (no | bad | (ok, gid, last-modified, (event | todo)))>

<!ELEMENT event (eventid, globalid, clientid?, eventtitle, eventdesc?,
        eventstart, eventstop, viewablestart, viewablestop,
        allday?, eventpriority, eventemaildiff,
        eventpagerdiff, eventnotifylist?, eventrrule?,
        eventxdata?, created, dtstamp, last-modified, sequence,
        status?, x-mira-sendmail?, x-mira-readonly?, attach*,
        attendees?, extattendees?, resources?, perms*, exceptions?,
        timezoneoffset?, timezone?)>

<!ELEMENT todo (todoid, todotitle, tododesc?, todopriority,
        todoxdata?, created, dtstamp, last-modified, sequence,
        x-mira-readonly?, due?, completed?)>

<!ELEMENT gid (#PCDATA)>
<!ELEMENT last-modified (#PCDATA)>

<!ELEMENT ok (#PCDATA)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT bad (#PCDATA)>
```

userlist.dtd

```
<!ELEMENT userlist (no | bad | (ok, users*))>

<!ELEMENT users (user*)>
<!ELEMENT user (#PCDATA)>
        <!ATTLIST user userid CDATA #REQUIRED>

<!ELEMENT ok (#PCDATA)>
<!ELEMENT no (#PCDATA)>
<!ELEMENT bad (#PCDATA)>
```

version.dtd

```
<!ELEMENT version (interface,mos,synq?)>

<!ELEMENT interface (#PCDATA)>
<!ELEMENT mos (#PCDATA)>
<!ELEMENT synq (#PCDATA)>
```

viewother.dtd

```
<!ELEMENT viewother (no | bad |
         (ok, sid, x-mira-readonly?, dumpcal?, emaillist))>

<!ELEMENT sid (#PCDATA)>
<!ELEMENT x-mira-readonly EMPTY>
<!ELEMENT dumpcal (#PCDATA)>
<!ELEMENT emaillist (email*)>
<!ELEMENT email (#PCDATA)>
        <!ATTLIST email type CDATA #IMPLIED>

<!ELEMENT ok (#PCDATA)>
```

```
<!ELEMENT no (#PCDATA)>
<!ELEMENT bad (#PCDATA)>
```

Mirapoint Message Base DTDs

WebMail and message store have twelve DTDs, each presented in a section below.

bad.dtd

```
<!ELEMENT bad (#PCDATA)>
```

body.dtd

```
<!ELEMENT body (bodypart*)>
<!ELEMENT bodypart (#PCDATA)>
<!ATTLIST bodypart
    xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
        xlink:href CDATA #REQUIRED
        xlink:type CDATA #IMPLIED >
```

bodystructure.dtd

```
<!ELEMENT bodystructure (bodystructure)>
  <!ELEMENT bodystructure (structure)>
  <!ELEMENT structure (#PCDATA)>
```

imap_genlist.dtd

```
<!ELEMENT mailboxlist (mailbox+)>
  <!ELEMENT mailbox (name, unread?, count?)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT unread (#PCDATA)>
  <!ELEMENT count (#PCDATA)>
```

indexlist.dtd

```
<!ELEMENT indexlist (index*)>
<!ELEMENT index (msgid, uid, date, subject?, size, priority,
        deleted?, attachment?, seen?, flagged?, answered?, draft?,
        from?, tolist?, cclist?, bcclist?)>
<!ELEMENT msgid (#PCDATA)>
<!ELEMENT uid (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT size (#PCDATA)>
<!ELEMENT priority (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT tolist (to*)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT cclist (cc*)>
<!ELEMENT cc (#PCDATA)>
<!ELEMENT bcclist (bcc*)>
<!ELEMENT bcc (#PCDATA)>
<!ELEMENT deleted EMPTY>
<!ELEMENT attachment EMPTY>
<!ELEMENT seen EMPTY>
<!ELEMENT flagged EMPTY>
<!ELEMENT answered EMPTY>
<!ELEMENT draft EMPTY>
```

login.dtd

```
<!ELEMENT sid (#PCDATA)>
```

mailboxlist.dtd

```
<!ELEMENT mailboxlist (mailbox+)>
  <!ELEMENT mailbox (name, unread, count)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT unread (#PCDATA)>
  <!ELEMENT count (#PCDATA)>
```

no.dtd

```
<!ELEMENT no (#PCDATA)>
```

ok.dtd

```
<!ELEMENT ok (#PCDATA)>
```

preferences.dtd

```
<!ELEMENT preferences (fullname, email, replyto,
                       messagecnt, composewidth, composeheight,
                       sentfolder, savesent, replyopt, signature,
                       includesig, version, timezone, charset,
                       draftfolder, junkfolder, trashfolder, usetrash)>
<!ELEMENT fullname (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT replyto (#PCDATA)>
<!ELEMENT messagecnt (value, default)>
<!ELEMENT composewidth (value, default)>
<!ELEMENT composeheight (value, default)>
<!ELEMENT sentfolder (#PCDATA)>
<!ELEMENT savesent (value, default, optionlist)>
<!ELEMENT replyopt (value, default, optionlist)>
<!ELEMENT signature (#PCDATA)>
<!ELEMENT includesig (value, default, optionlist)>
<!ELEMENT version (value, default, optionlist)>
<!ELEMENT timezone (#PCDATA)>
<!ELEMENT charset (value, default, optionlist)>
<!ELEMENT draftfolder (#PCDATA)>
<!ELEMENT junkfolder (#PCDATA)>
<!ELEMENT trashfolder (#PCDATA)>
<!ELEMENT usetrash (value, default, optionlist)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT default (#PCDATA)>
<!ELEMENT optionlist (option*)>
<!ELEMENT option (#PCDATA)>
```

rfc822.dtd

```
<!ELEMENT rfc822 (#PCDATA)>
```

status.dtd

```
<!ELEMENT status (mailbox+)>
  <!ELEMENT mailbox (name, unread, count, quota_used, quota_limit)>
  <!ELEMENT name   (#PCDATA)>
  <!ELEMENT unread (#PCDATA)>
```

```
<!ELEMENT count  (#PCDATA)>
<!ELEMENT quota_used  (#PCDATA)>
<!ELEMENT quota_limit  (#PCDATA)>
```

Address Book DTD

The status DTD is included in the single Address Book DTD.

addrbook.dtd

```
<!-- Contacts: -->
<!ELEMENT persons (person | deleted-person)*>
<!ATTLIST persons count CDATA #REQUIRED>

<!ELEMENT person (category |
                  cn |
                  surname |
                  givenname |
                  nickname |
                  displayname |
                  mail |
                  postaladdress |
                  locality |
                  state |
                  postalcode |
                  country |
                  organization |
                  organizationalunit |
                  title |
                  homeurl |
                  telephonenumber |
                  homephone |
                  mobile |
                  pager |
                  facsimiletelephonenumber |
                  primaryphone |
                  anniversaryday |
                  anniversarymonth |
                  anniversaryyear |
                  birthday |
                  birthmonth |
                  birthyear |
                  description |
                  uuid |
                  lastmodtime)*>
<!ATTLIST person ref CDATA #REQUIRED>

<!ELEMENT cn (#PCDATA)>
<!ELEMENT surname (#PCDATA)>
<!ELEMENT givenname (#PCDATA)>
<!ELEMENT nickname (#PCDATA)>
<!ELEMENT displayname (#PCDATA)>
<!ELEMENT mail (#PCDATA)>
<!ELEMENT postaladdress (#PCDATA)>
<!ELEMENT locality (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT postalcode (#PCDATA)>
<!ELEMENT country (#PCDATA)>
<!ELEMENT organization (#PCDATA)>
<!ELEMENT organizationalunit (#PCDATA)>
<!ELEMENT title (#PCDATA)>
```

```
<!ELEMENT homeurl (#PCDATA)>
<!ELEMENT telephonenumber (#PCDATA)>
<!ATTLIST telephonenumber primary (primary|yes|no) #IMPLIED>
<!ELEMENT homephone (#PCDATA)>
<!ATTLIST homephone primary (primary|yes|no) #IMPLIED>
<!ELEMENT mobile (#PCDATA)>
<!ATTLIST mobile primary (primary|yes|no) #IMPLIED>
<!ELEMENT pager (#PCDATA)>
<!ATTLIST pager primary (primary|yes|no) #IMPLIED>
<!ELEMENT facsimiletelephonenumber (#PCDATA)>
<!ATTLIST facsimiletelephonenumber primary (primary|yes|no) #IMPLIED>
<!ELEMENT primaryphone (#PCDATA|default|optionlist)*>
<!ELEMENT anniversaryday (#PCDATA)>
<!ELEMENT anniversarymonth (#PCDATA)>
<!ELEMENT anniversaryyear (#PCDATA)>
<!ELEMENT birthday (#PCDATA)>
<!ELEMENT birthmonth (#PCDATA)>
<!ELEMENT birthyear (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT uuid (#PCDATA)>
<!ELEMENT lastmodtime (#PCDATA)>

<!ELEMENT deleted-person EMPTY>
<!ATTLIST deleted-person ref CDATA #REQUIRED>

<!-- Groups: -->
<!ELEMENT groupsofnames (groupofnames)*>
<!ATTLIST groupsofnames count CDATA #REQUIRED>

<!ELEMENT groupofnames (cn , member*)>
<!ATTLIST groupofnames ref   CDATA #REQUIRED
                       count CDATA #REQUIRED>

<!ELEMENT member (#PCDATA)>
<!ATTLIST member ref CDATA #REQUIRED>

<!-- Categories: -->
<!ELEMENT categories (category)*>
<!ATTLIST categories count CDATA #REQUIRED>

<!ELEMENT category (#PCDATA | cn | index | count)*>
<!ATTLIST category unfiled (unfiled|yes|no) #IMPLIED>

<!ELEMENT index (#PCDATA)>
<!ELEMENT count (#PCDATA)>

<!-- Letter Categories: -->
<!ELEMENT letters (letter)*>
<!ATTLIST letters count CDATA #REQUIRED>

<!ELEMENT letter (cn | count)*>

<!-- Import Result: -->
<!ELEMENT import (recordcount|personcount|groupcount)*>
<!ATTLIST import file CDATA #REQUIRED>

<!ELEMENT recordcount (#PCDATA)>
<!ELEMENT personcount (#PCDATA)>
<!ELEMENT groupcount  (#PCDATA)>

<!-- Export Result: -->
<!ELEMENT export (#PCDATA)>
<!ATTLIST export file CDATA #REQUIRED>
```

```
<!-- Search Results: -->
<!ELEMENT searchresult (search,(persons|groupsofnames)*)>
<!ATTLIST searchresult resultcount CDATA #IMPLIED
                       personcount CDATA #IMPLIED
                       groupcount  CDATA #IMPLIED>

<!ELEMENT search (namepattern|mailpattern|phonepattern|addresspattern|
                  otherpattern|
                  grouppattern|memberref)*>

<!ELEMENT namepattern      (#PCDATA)>
<!ELEMENT mailpattern      (#PCDATA)>
<!ELEMENT phonepattern     (#PCDATA)>
<!ELEMENT addresspattern   (#PCDATA)>
<!ELEMENT otherpattern     (#PCDATA)>
<!ELEMENT grouppattern     (#PCDATA)>
<!ELEMENT memberref        (#PCDATA)>

<!-- Options/Preferences: -->
<!ELEMENT preferences (charset|language|format|fieldname|
                       conflict|detail|modtype|csvfield|
                       primaryphone)*>

<!ELEMENT charset   (default|optionlist)*>
<!ELEMENT language  (optionlist)*>
<!ELEMENT format    (optionlist)*>
<!ELEMENT fieldname (optionlist)*>
<!ELEMENT conflict  (optionlist)*>
<!ELEMENT detail    (optionlist)*>
<!ELEMENT modtype   (optionlist)*>
<!ELEMENT csvfield  (optionlist)*>

<!ELEMENT optionlist (option)*>
<!ELEMENT option     (#PCDATA)>

<!ELEMENT default    (#PCDATA)>

<!-- Status: -->
<!ELEMENT ok  (#PCDATA)>
<!ELEMENT no  (#PCDATA)>
<!ELEMENT bad (#PCDATA)>

<!-- SessionID: -->
<!ELEMENT sid (#PCDATA)>

<!-- Version: -->
<!ELEMENT version (interface,mos)>
<!ELEMENT interface (#PCDATA)>
<!ELEMENT mos (#PCDATA)>
```