Programmer's Reference

iPlanet Calendar Server

Version 2.0

Service Provider Edition

Recycled and Recyclable Paper

Version 1.0

Part Number: 806-3478-10

Printed in the United States of America. 00 99 98 5 4 3 2 1

Contents

Preface

This document describes the architecture of iPlanet Calendar Server, and provides a detailed description of the Web Calendar Access Protocol (WCAP), the protocol you use to access services, as well as the Calendar Server Application Programming Interface (CSAPI) that you use to modify server functionality.

Calendar Server is a powerful and flexible cross-platform solution using open Internet standards that lets service providers of all sizes host personal scheduling calendars for their customers.

Who Should Read This Book

This guide is for programmers who want to manage and possibly customize applications in order to implement Calendar Server at your site.

What's in This Guide

- Core Architecture
- Calendar Database
- User Data
- Command Overview
- Command Formats
- Recurrence Handling
- Command Details
- Error Handling
- Examples of Command Usage

- CSAPI Architecture
- API Reference

Conventions Used in This Book

Sidebar text Sidebar text marks important information. Make sure you read the information before continuing with a task.

`Monospaced font`—This typeface is used for any text that appears on the computer screen or text that you should type. It is also used for filenames, distinguished names, functions, and examples.

`Italicized Monospaced` font—This typeface is used to represent text that you enter using information that is unique to your installation. It is used for server paths and names and account IDs.

For example, throughout this document you will see path references of the form:

`server-root/``cal`

In these situations, *`server-root`* represents the directory path in which you installed Calendar Server. For example, if you install your server in the default directory location suggested by the installation program:

- on Unix the actual path is: `/opt/SUNWicsrv/cal`
- on Windows NT the actual path is: `c:\iplanet`

Also, all paths specified in this manual are in Unix format. If you are using a Windows NT-based Calendar Server, you should assume the Windows NT equivalent file paths whenever Unix file paths are shown in this book.

What is iPlanet Calendar Server?

iPlanet Calendar Server is the first readily deployable, LDAP-based solution that lets the ISP and Telecommunication service provider host personal event calendars to their base of subscribed customers. Using iPlanet Calendar Express, a web browser-based client application, customers can access data hosted on the Calendar Server to:

- update personal schedules with their own appointments
- publish calendars for viewing by other people

Built from the ground up to provide native support for the emerging set of internet calendar standards, iPlanet Calendar Server provides the following benefits:

- **Massively Scalable.** Scaling to the requirements of the largest service providers, it supports a hosting environment of up to several million personal event calendars.
- **Low Cost of Ownership.** Native support of LDAP lets a service provider centrally manage its entire customer base in a single user directory and minimizes the costs of administering the server while also providing a platform for extending the enhanced services a provider can offer its customers.
- **Internet Calendaring and Scheduling.** Native support for iCalendar calendaring standards lets users schedule personal events in a format that is easily shared across the internet.

1

Overview of Architecture

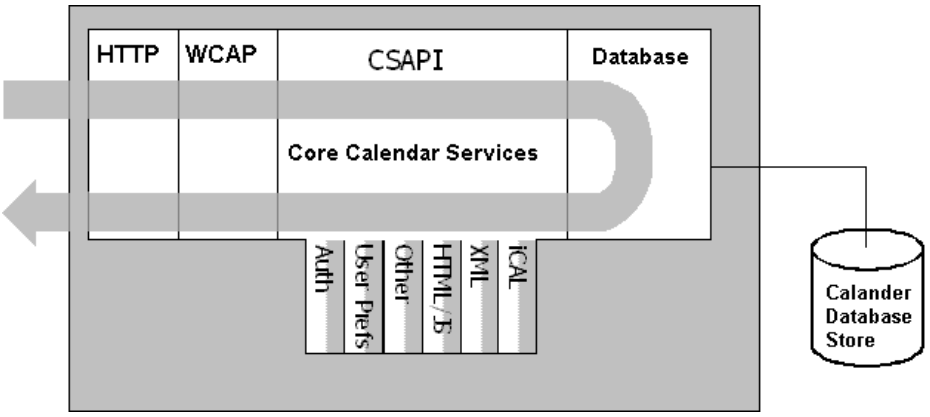
The iPlanet Calendar Server is designed for programmers who want a full web calendar solution. It is a general purpose server to support applications that need calendar data.

Core Architecture

The calendar server uses HTTP as its primary transport. It receives commands encoded in HTML and sends replies as HTML and JavaScript. The data returned from requests is not formatted for display. The embedded JavaScript generates HTML which it sends to a visible frame. The user experience is the same as if the HTML was loaded from a server. The advantage is that for many operations, the JavaScript can handle everything locally and does not need to contact the server. This provides a faster screen update for the user while lightening the load on the server.

Figure 1.1 shows the core architecture of Calendar Server. The subsystems and data flow are discussed in more detail in the sections that follow.

Figure 1.1 Calendar Server Core Architecture



HyperText Transfer Protocol (HTTP)

Commands enter through the Protocols module. This module is a minimal HTTP implementation, streamlined to support calendar requests. The requests are in Web Calendar Access Protocol (WCAP), a simple protocol for storing, retrieving, and administering calendars. The WCAP protocol is described in Chapter 2, Web Calendar Access Protocol (WCAP).

Web Calendar Access Protocol (WCAP)

The WCAP subsystem interprets a command and sends it to the Core subsystem for execution. The Core system makes data requests to the Caldb subsystem where information is stored or retrieved from the database as needed and returned in a low-level format. The Core subsystem then translates the low-level format into the format requested in the WCAP command. These results are then encoded in HTML and returned.

Calendar Server Application Program Interface (CSAPI)

CSAPI is an API you can use to add functionality to Calendar Server or to augment the way certain operations in the server are performed. You can use CSAPI to modify the following major areas of functionality:

- Authentication
- User Attribute Access
- Data Format Translation

For example, the server's default mechanism for authentication uses LDAP, and user preferences are stored in LDAP by default. You may have an existing infrastructure for authenticating users and saving user preferences which is not based on LDAP. You can use CSAPI to override the default mechanisms and use your existing authentication and directory services.

The CSAPI is described in Chapter 3, Calendar Server API (CSAPI).

Supported Format Encoding

For data being stored, the Core translates the input into the binary form that Caldb uses. Calendar Server supports the following format encodings:

- HTML/JavaScript (the default)
- XML
- iCal

You can add other formats by developing a translator DLL or shared library using CSAPI, an API for extending the calendar server.

Calendar Database

By default, Calendar Server uses BerkeleyDB as its database. This database product is provided with the distribution. With Berkeley DB, Calendar Server supports transactions and live backups, as well as a limited form of high-availability.

Calendar Data Format

The calendar data format is modeled after the IETF iCal standard [RFC-2445]. The datastore maintains **`calendars`**, which are collections of **`iCal`** components. The components include **`events`**, **`to-dos`**, **`journal entries`**, and **`alarms`**. A calendar can be owned by one or more **`users`**. User attributes are maintained by an external mechanism. By default, that mechanism is LDAP. You can use CSAPI to override the default mechanism.

Calendars have calendar sets, which allow multiple calendar sources to be aggregated into a single calendar for display purposes. A user can, for example, have a default calendar view made up of his or her own calendar, the group's calendar, the holidays calendar, and the "latest action video releases" calendar, which is called a “calendar set”.

Calendars also use links. Links are similar to calendar sets, but allow a more granular selection of events to overlay. For example, suppose that a convention center maintains a calendar of all the conventions to be held in its facilities. A user may be interested in one of the conventions, but not in all of them. In this case, the user can add an event into a personal calendar as a link to the event of interest in the convention calendar. Any change to the event in the convention center's calendar will be reflected in the linked copy.

This infrastructure makes it possible to feed real-time event data into the database using import formats such as iCal or XML. Event data can be fed in from calendars such as the San Francisco 49ers season schedule, convention center schedules, concert schedules, or any schedule of events that may be of interest.

Calendar Server provides tools for retrieving calendar data from event feeds and from the database, so that users can view and retrieve event information. The user can link or layer these events onto their own calendar views, providing rich event information that is maintained by third parties.

Calendar Data Exchange

All calendars and events can be referenced as URLs. Users can embed these links in email messages and web pages. Users can click on a link to see a monthly calendar view, a weekly view, a daily view, or a specific event. If the calendars are publicly readable, users will not be asked to log in.

Calendar Server supports server-side email alarms, which can be sent to a list of recipients. The format of the email message is completely configurable. It is currently maintained as a server attribute, rather than as a user or calendar attribute.

Currently, Calendar Server has limited support for the ITIP/IMIP standards [RFC-2446, RFC-2447]. A user can send out IMIP PUBLISH messages, and the server can import the text or calendar portion of an IMIP PUBLISH message.

User Data

The primary component of user data is the calendar set, or list of calendars that belong to a user. A user's calendar data is associated with the user through a calendar set user preference, which lists all the calendars that make up that user's calendar set.

Calendar User Preferences

Calendar Server customizes the display of calendaring information for each user according to attributes called user preferences. User preferences, as opposed to calendar preferences, refer to the user interface representation of information. User preferences include such things as email address, user name, and preferred colors to use when rendering calendar information. For a complete list of preferences, see the `get_calprops` and `set_calprops` command descriptions in Chapter 2, Web Calendar Access Protocol (WCAP).

Calendar Access Control

A calendar can have multiple owners, but it has only one primary owner. You can control access to a calendar by specifying its read access or write access as either public or private.

When a calendar is public, anyone has access for that type of action on the calendar, including an anonymous user.

- When a calendar has public read access, any user can look at data in that calendar.

- When a calendar has public write access, any user can write data to that calendar.

When a calendar is private, access is limited to the owners.

- When a calendar has private read access, only owners of the calendar can look at data from that calendar.
- When a calendar has private write access, only owners can write data to it. In addition, the following actions are limited to primary owners:
  - Adding new owners
  - Deleting the calendar
  - Changing the calendar properties

To issue commands for a calendar that is marked as private, you must provide the session identifier that was returned by the `login` command for your session. Because your identity is authenticated at login, the identifier protects the private data from being read or written to by unauthorized users. For more information on session identifiers and authentication, see the following sections:

- Session Identifiers in Chapter 2, Web Calendar Access Protocol (WCAP)
- CSAPI Authentication Interface: csIAuthentication in Chapter 3, Calendar Server API (CSAPI)

2

Web Calendar Access Protocol (WCAP)

This chapter describes the Web Calendar Access Protocol (WCAP), which is a high level command-based protocol that clients use to communicate with Netscape iPlanet Calendar Server. This chapter has the following sections:

- Command Overview
- Command Formats
- Recurrence Handling
- Command Details
- Error Handling
- Examples of Command Usage

Command Overview

This section introduces the commands supported in WCAP in each usage category, according to functionality. Detailed information about each command is given in the Command Details section later in this chapter.

WCAP commands include client requests and server responses for transmitting calendar data. WCAP commands perform three general types of function:

- Manipulation of user configuration information
- Manipulation of calendar administration information
- Manipulation of web calendar data

Table 2.1 describes the commands that are concerned with user configuration information:

Table 2.1 WCAP user configuration command overview

WCAP Command	Purpose
`login`	Authenticate a user.
`logout`	Terminate the current user's session and return to login screen.
`change_password`	Change the password for the current user.
`get_userprefs`	Retrieve calendar preferences.
`set_userprefs`	Set calendar preferences.
`version`	Get the wcap version that the calendar server supports.

Table 2.2 describes the commands that are concerned with calendar administration. These commands can only by used by an administrator with the proper privileges, and must be issued on the administrative port:

Table 2.2 WCAP administrative command overview

WCAP Command	Purpose
`shutdown`	Shut down the server.
`admin_logout`	Terminate the session of a specific user.
`get_session`	Retrieve session information for a user.
`get_all_timezones`	Retrieve data about all timezones supported by the server.
`refresh`	Refresh the server configuration.

Table 2.3 describes the commands that are concerned with manipulating calendar data:

Table 2.3 WCAP data manipulation command overview

WCAP Command	Purpose
createcalendar	Create a new calendar.
deletecalendar	Delete a calendar.
get_calprops	Retrieve calendar properties.
get_guids	Generate a set of globally unique identifiers.
set_calprops	Set calendar properties.
search_calprops	Search all calendars for specific properties.
fetchcomponents_by_range	Retrieve a list of events over a specific time period, using an attribute filter.
fetchevents_by_id	Retrieve a specific event and its recurrences.
storeevents	Store event in the database.
deleteevents_by_id	Delete events from a calendar.
deleteevents_by_range	Delete events within a range from one or more calendars.
deletecomponents_by_range	Delete events and todos with a range from one or more calendars.
deleteevents_by_range	Delete events within a range from one or more calendars.
fetchtodos_by_id	Retrieve a specific todo and its recurrences.
storetodos	Store todos in the database.
deletetodos_by_id	Delete todos from a specific calendar.
deletetodos_by_range	Delete todos within a range from one or more calendars.
addlink	Add a link from one calendar to an event or todo in another.
export	Export events and todos from a calendar to a file.
import	Import events and todos from a file to a calendar.

Session Identifiers

For most commands, you must specify the session identifier that is returned by the `login` command. For some commands this parameter is required, while for others it is required only to access a private calendar. The session identifier ensures that data is accessible only to users with the required level of privilege or ownership.

When logging into the system, a user provides authentication of identity. The default authentication mechanism uses plain-text passwords and user names. Calendar Server generates the session identifier only when authentication is successful. The identifier then serves as proof of authentication in subsequent calendaring operations.

You can customize the authentication mechanism to use a local or external authentication scheme. For more information on customization, see Chapter 2, "Web Calendar Access Protocol (WCAP)."

Command Formats

The plug-iCalin architecture of Calendar Server allows it to support multiple command formats. Both client and server can use a variety of data formats to meet various ISP needs.

Currently, the command protocol is used with HTTP, and follows the standards defined by the WC3 URL specifications. It supports SSL for encrypted communication.

WCAP in Calendar Server consist of JavaScript objects formatted as XML or iCal, communicated as HTML documents over HTTP on both the client and server side. The client supports version 3 of the major browsers running JavaScript.

Client Request Formats

Clients can submit the following data formats to Calendar Server in command requests:

Table 2.4 Supported client request formats

Command Format	Purpose
Universal Resource Identifier (URI)	Requests are made using standard URI syntax
HTML Form - encoding of `application/x-www-form-urlencoded`	Requests are made using native JavaScript objects
HTML Form - encoding of `text/xml`	Requests are made using JavaScript objects formatted as XML
HTML Form - encoding of `text/calendar`	Requests are made using JavaScript objects formatted as iCal

URI Request Format

Use the following format to submit a URI request:

```
http[s]://webcal.host.com/COMMAND?PARAM=VAL&PARAM=VAL...
```

Multiple items are delimited by semicolons. If a string contains a semicolon character, replace the semicolon with its quoted-printable equivalent, `=3B`. For example, to represent the string “gh;i” in a list of IDs, use the following:

```
uid=abc;def;gh=3bi;jkl
```

HTML Request Formats

Submit a form with `method=`*`get|post`* and `action=`*`command`*, where *`command`* is the WebCal command to execute. Format parameters as specified in the encoding.

Client-Side Event Notification

All client-side JavaScript code in the parent frame of the response page must implement a method called `Cal`*`Command`*`Callback()`, where *`Command`* is the name of the command requested. This callback is invoked when the HTML response has completed loading.

Server Response Formats

Calendar Server responds to client requests by serving HTML containing JavaScript objects. The JavaScript objects can be formatted as either of the following:

- iCal objects (output format specifier `text/calendar`)
- XML objects (output format specifier `text/xml`)

The default format for JavaScript objects in HTML responses is iCal format. A client can request that responses be in XML. You can configure a response format preference for a server or user, or specify the desired response format for an individual request.

Recurrence Handling

This section describes how you specify recurrence parameters for WCAP commands.

- Specifying Recurrences
- Storing Recurrences

Specifying Recurrences

When you modify or delete an event or todo, you can (and sometimes must) specify whether or not it is recurrent, and, if it is, whether and how to modify the recurrences as well as the original event or todo.

An event or todo can have a recurrence identifier, in the form of a `DateTime` string. Use the `rid` parameter to specify the date of a specific recurrence you want to modify or delete.

When there are multiple recurrences, you can specify whether to modify them or not, using the `mod` parameter. The `mod` parameter specifies whether to apply the changes to one or more instances of the event or todo. The `mod` values result in the following behavior:

Table 2.5

1	Modify only the specified instance of the event or todo
2	Modify the specified event or todo and all subsequent related recurrences.
3	Modify the specified event or todo and all prior related recurrences.
4	Modify all instances of the event or todo.

When deleting an event or todo by id, you must specify the `rid` and `mod` parameters. For a non-recurring event or todo, the `rid` is 0.

Note An event or todo can recur a maximum of 60 times.

Storing Recurrences

When you create and store new events or todos, you use the following rule parameters to specify exactly how to create or modify recurrences:

rid: This parameter specifies a unique recurrence date of an event or todo. If you specify this parameter when storing an event or todo, there is no expansion of the event; that is, there is only one recurrence.

For example:

```
http://myserver.mycompany.com/
storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=19990301T112233Z
&rid=19990331T112233;dtend=19990301T112233&summary=uuuu
```

rrules: The `rrules` parameter takes a semicolon-separated list of quoted recurrence rule strings. Each string represents a recurrence rule of the event. Each string must be enclosed in quotes.

This example shows an `rrules` parameter that specifies to recurrence rules:

```
rrules="count%3D10%3Bfreq%3Ddaily";"freq%3Dweekly%3Bcount%3D4"
```

(COUNT=10;FREQ=DAILY and FREQ=WEEKLY;COUNT=4 encoded)

The first rule specifies that the event is to occur daily for 10 instances. The second rule specifies that the event is to occur weekly for 4 instances.

The following example URL passes the example `rrules` parameter:

```
http://myserver.mycompany.com/
storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=19990301T112233Z
&rrules="count%3D10%3Bfreq%3Ddaily";"freq%3Dweekly%3Bcount%3D4"
&dtend=19990301T112233&summary=uuuu
```

exrules: The `exrules` parameter takes a semicolon-separated list of quoted recurrence rule strings where each rule is an excluded recurrence of the event.

For example, the following `exrules` parameter specifies a recurring event that does not recur at the times specified by the two rules:

```
exrules="count%3D10%3Bfreq%3Ddaily";"freq%3Dweekly%3Bcount%3D4"
```

(COUNT=10;FREQ=DAILY and FREQ=WEEEKLY;COUNT=4 encoded)

The first rule is for the event not to occur daily for 10 instances. The second rule is for the event not to occur weekly for 4 instances.

The following example URL passes the example `exrules` parameter:

```
http://myserver.mycompany.com/
storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=19990301T112233Z
&exrules="count%3D10%3Bfreq%3Ddaily";"freq%3Dweekly%3Bcount%3D4"
&rrules="count%3D100%3Bfreq%3Ddaily"&dtend=19990301T112233&summary=uuuu
```

rchange: The `rchange` parameter specifies whether recurrences are expanded in `storeevents` and `storetodos`. Normally, events and to do calendar components are expanded, so the parameter defaults to 1.

You might not want to expand recurrences when you are modifying multiple events. For example, suppose a meeting recurs every Friday starting Jan 1, 2000. Use the following URL to change the summary of each event after Feb. 1, 2000 to `changed-event`. This example sets the `rchange` parameter to 0, to make the modification without adding additional events:

```
http://jdoe.mycompany.com/storeevents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=abcxyz&dtstart=19990201T112233Z
&rrules="byday%3Dfr%3Bfreq%3Dweekly"&summary=changed-event
&rid=19990201T112233Z&mod=2&rchange=0
```

rdates: The `rdates` parameter takes a semicolon-separated list of date-time specifications where each date-time gives a recurrence date of the event.

For example, the following `rdates` parameter specifies a recurring event with two recurrence dates (3/31/99 11:22:33 and 5/31/99 11:22:33):

```
rdates=19990331T112233;19990531T112233
```

The following example URL passes the example `rdates` parameter:

```
http://myserver.mycompany.com/
storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=19990301T112233Z
&rdates=19990331T112233;19990531T112233
&dtend=19990301T112233&summary=uuuu
```

If you want to the change the recurrence rule after a certain date, you must set `rchange` to 1.

exdates: The `exdates` parameter takes a semicolon-separated list of date-time specifications. Each date-time represents an excluded date of the event.

For example, the following `exdates` parameter specifies a recurring event that does not occur on the two specified dates (3/31/99 11:22:33 and 5/31/99 11:22:33):

```
exdates=19990331T112233;19990531T112233
```

The following example URL passes the example `exdates` parameter:

```
http://myserver.mycompany.com/
storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=19990301T112233Z
&exdates=19990331T112233;19990531T112233
&rrules="COUNT%3D200%3BFREQ=DAILY";dtend=19990301T112233&summary=uuuu
```

Command Details

This section describes the usage and parameters for WCAP commands in alphabetical order. WCAP commands and what they do are listed and summarized briefly.

Command	Purpose
addlink	Add a link from one calendar to an event or todo in another.
admin_logout	Terminate the session of a specific user.
change_password	Change the password for the current user.
createcalendar	Create a new calendar.
deletecalendar	Delete a calendar.
deletecomponents_by_range	Delete events or todos within a range from one or more calendars.
deleteevents_by_id	Delete events or todos from a specific calendar.
deleteevents_by_range	Delete events within a range from one or more calendars.
deletetodos_by_id	Delete todos from a specific calendar.
deletetodos_by_range	Delete todos within a range from one or more calendars.
export	Export events and todos from a calendar to a file.
fetchcomponents_by_range	Retrieve properties, events, and todos within a range from one or more calendars.
fetchevents_by_id	Retrieve an event and its recurrences.
fetchtodos_by_id	Retrieve a todo and its recurrences.
get_all_timezones	Retrieve all timezones supported by the server.
get_calprops	Retrieve calendar properties.
get_guids	Generate a set of globally unique identifiers.
get_session	Retrieve session information for a user.
get_userprefs	Retrieve calendar preferences.
import	Import events and todos from a file to a calendar
login	Authenticate a user.
logout	Terminate the current user's session and return to login screen.

`ping`	Query the server to see if it is active.
`refresh`	Refresh the server configuration.
`search_calprops`	Search all calendars for specific properties.
`set_calprops`	Set the calendar properties of a calendar.
`set_userprefs`	Set calendar preferences.
`shutdown`	Shut down the server.
`storeevents`	Store events in the database.
`storetodos`	Store todos in the database.
`version`	get the wcap version that the calendar server supports.

addlink

Purpose Add links from one calendar to events or todos in another calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier. Required if the destination calendar is not public.	N	`null`
`srcCal`	string	The calendar containing the events or todos. You must have READ access to this calendar.	Y	N/A
`destCal`	string	The calendar to contain the links. You must have WRITE access to this calendar.	Y	N/A
`uid`	semicolon-separated list of strings	A list of event and/or todo identifiers to which to create links.	Y	N/A
`rid`	semicolon-separated list of strings	A list of event and/or todo recurrence identifiers to which to create links.	Y	N/A

Purpose Use this command to add links in the destination calendar to the specified events and/or todos in the source calendar. You must specify the `id` parameter with the command unless the specified destination calendar is a public calendar.

The number of items in the `uid` and `rid` lists must match exactly, with each `rid` corresponding to the `uid` in the same position in the list. If an event or todo has no recurrence ID, use 0 in the `rid` list.

admin_logout

Purpose Terminate the session of a specific user.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`user`	string	Administrative user ID (must be contained in the `service.http.admins` server preference).	Y	N/A
`password`	string	Administrative user password.	Y	N/A
`tid`	integer	The session identifier for the target session to be terminated.	N	N/A
`addr`	IP address	The IP address of the user whose session is to be terminated.	N	N/A
`userid`	string	The user ID of the user whose session is to be terminated.	N	N/A

Purpose Use this command to terminate the specified session information for the specified user. This command ends the specified session, and deletes the session instance of the user in the session table. The user is returned to the login screen. The command returns an HTML page containing the response string.

At least one of the parameters `tid`, `addr`, or `userid` must be specified.

Only users with administrative privilege can use this command. The command must be sent to the administrative port.

change_password

Purpose Change the password of the current user.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	string	The session identifier.	Y	N/A
`oldPassword`	string	The previous user password.	Y	N/A
`newPassword`	string	The new user password.	Y	N/A
`fmt-out`	string	The output content-type in which data is to be returned. One of the following: `text/calendar` `text/xml` `text/js`	N	`text/js`

Purpose Passwords are passed as plain text.

To use this function, non-administrative users must have the `service.wcap.allowchangepassword` preference set.

createcalendar

Purpose Create a new calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier.	Y	N/A
`calid`	string	A string from which to generate the ID of the new calendar. The generated ID is of the form *`username`*`:`*`calid`*.	Y	N/A

Parameter	Type	Purpose	Required	Default

`fmt-out`	string	The output content-type in which data is to be returned. One of the following:
		`text/calendar`
		`text/xml`
		`text/js`
			N	`text/js`

`set_calprops`	integer (0 or 1)	Whether to set the properties of the new calendar.	N	`0`

Purpose Use this command to create a new calendar for the current user as the child of the specified existing calendar. When you pass the `set_calprops` parameter as `1`, you can also pass any additional parameters as defined for the `set_calprops` command.

The command returns the output from a call to the `fetchcomponents_by_range` command, formatted according to the `fmt-out` value, showing the calendar properties.

This function is only available for non-administrative users if the `service.wcap.allowcreatecalendars` preference is set.

The following example URL creates a calendar with the ID `jdoe:newcal`for the user `jdoe`, sets the name to `New-Calendar`, and the categories to `business` and `work`:

```
http://myserver/createcalendar.wcap?id=b5q2o8ve2rk02nv9t6&calid=newcal
&set_calprops=1&name=New-Calendar&categories=business;work
```

For more information on calendar properties you can set, see the `set_calprops` command.

deletecalendar

Purpose Deletes a calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier.	Y	N/A
`calid`	string	The name of the calendar to delete.	Y	N/A

`fmt-out`	string	The output content-type in which data is to be returned. One of the following:
		`text/calendar`
		`text/xml`
		`text/js`
			N	`text/js`

Purpose Use this command to delete the specified calendar belonging to the current user.

This function is only available for non-administrative users if the `service.wcap.allowdeletecalendars` preference is set.

The following example URL deletes a calendar named `newcal`:

```
http://myserver/deletecalendar.wcap?id='bu9p3eb8x5p2nm0q3'&calid=newcal
```

deletecomponents_by_range

Purpose Delete events and todos in a range from a calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique id string	The session identifier. Required unless the calendar is public.	N	`null`
`dtstart`	ISO8601 DateTime string (UTC)	Start time and date of events or todos to be deleted. A value of 0 means delete all events and todos up to a specified start-time. This value must be in coordinated universal time (Z).	N	`0`
`dtend`	ISO8601 DateTime string (UTC)	End time and date of events or todos to be deleted. A value of 0 means delete all events and todos up to the latest date. This value must be in coordinated universal time (Z).	N	`0`
`calid`	string	Semicolon-separated list of calendar identifiers from which to delete events and todos. For example, `jdoe,jdoe;susan, jdoe;huey;susan`.	N	username of current user
`brief`	integer (0,1)	When 1, print out a brief version of output. When 0, print complete output of event/todo data. Applies only when output type is Javascript	N	`0`
`fmt-out`	string	The output content-type in which data is to be returned. One of the following: `text/calendar` `text/xml` `text/js`	N	`text/js`

Purpose Use this command to delete the events and todos that fall completely within the specified range from the specified calendars. If a range is not specified, it deletes all events and todos. If the `rid` parameter is passed, the command also deletes recurrences, as specified by the `mod` parameter. See “Storing Recurrences.”

When the notify value is 1, sends an IMIP cancellation message to the e-mail attendees.

For example, assuming the user has read access to the calendars jdoe and huey, the following URL deletes all events and todos from those two calendars:

```
http://deletecomponents_by_range.wcap?id=2342347923479asdf
  &calid=jdoe;huey&dtstart=0&dtend=0
```

If there was an error in deleting from jdoe, the delete_layer_errno[0] value would be set to 1. The javascript would return the error DELETECOMPONENTS_BY_RANGE_FAILED in the errno[0] variable.

deleteevents_by_id

Purpose Delete an event or events from a calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
id	unique id string	The session identifier. Required unless the calendar is public.	N	null
calid	string	Identifier of a calendar is which event is stored	Y	N/A
uid	string	Identifier of an event to be deleted, or semicolon-separated list of identifiers.	Y	N/A
rid	string	Recurrence identifier of the event, or semicolon-separated list of recurrence identifiers. If a list, must have same number or elements as uid list. If there are no recurrences, the value is 0.	Y	N/A
mod	integer 1,2,3,4	A modifier indicating which recurrences to delete, or semicolon-separated list of modifiers. If a list, must have same number or elements as uid list. One of the following values: 1 = THISINSTANCE 2 = THISANDFUTURE 3 = THISANDPRIOR 4 = THISANDALL	Y	N/A

Parameter	Type	Purpose	Required	Default
`notify`	integer 0,1	When 1, notify attendees of the change. When 0, do not notify attendees.	N	`0`
`brief`	integer (0,1)	When 1, print out a brief version of output. When 0, print complete output of event/todo data. Applies only when output type is Javascript.	N	`0`
`fmt-out`	string	The output content-type in which data is to be returned. One of the following: `text/calendar` `text/xml` `text/js`	N	`text/js`

Purpose Use this command to delete the specified event or events from the specified calendar. If the `rid` parameter is passed, the command also deletes recurrences, as specified by the `mod` parameter. See "Storing Recurrences."

When the `notify` value is 1, sends an IMIP cancellation message to the e-mail attendees of the event.

To delete multiple events, specify a semicolon-separated list for the `uid`, `rid`, and `mod` parameters. The three lists must have the same number of elements. Each list element corresponds to the same element in the other two lists.

For example, suppose there are two non-recurring events in the database with UIDs of `uid-EVENT1` and `uid-EVENT2`. Use the following URL to delete the two events:

```
http://myserver:81/deleteevents_by_id.wcap?id=br6p3t6bh5po35r
  &uid=uid-EVENT1;uid-EVENT2&rid=0;0&mod=1;1
```

Because the events are non-recurring, the `rid` value for each event is set to 0 and `mod` value for each event is set to 1.

deleteevents_by_range

Purpose Delete events in a range from a calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier. Required unless the calendar is public.	N	null
`dtstart`	ISO8601 DateTime string	Start time and date of events to be deleted. A value of 0 means delete all events up to a specified start-time.	N	0
`dtend`	ISO8601 DateTime string	End time and date of events to be deleted. A value of 0 means delete all events up to the latest time.	N	0
`calid`	string	Semicolon-separated list of calendar identifiers from which to delete events. For example, `jdoe,jdoe;susan, jdoe;huey;susan`.	N	username of current user
`brief`	integer (0,1)	When 1, print out a brief version of output. When 0, print complete output of event/todo data. Applies only when output type is Javascript	N	0
`fmt-out`	string	The output content-type in which data is to be returned. One of the following: `text/calendar` `text/xml` `text/js`	N	text/js

Purpose Use this command to delete the events that fall completely within the specified range from the specified calendars. If a range is not specified, it deletes all events. If the `rid` parameter is passed, the command also deletes recurrences, as specified by the `mod` parameter. See “Storing Recurrences.”

When the `notify` value is 1, sends an IMIP cancellation message to the e-mail attendees.

deletetodos_by_id

Purpose Delete a todo or todos from a calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier. Required unless the calendar is public.	Y	`null`
`calid`	string	Identifier of a calendar is which todo is stored	Y	N/A
`uid`	string	Identifier of a todo to be deleted, or semicolon-separated list of identifiers.	Y	N/A
`rid`	string	Recurrence identifier of the todo, or semicolon-separated list of recurrence identifiers. If a list, must have same number or elements as `uid` list. If there are no recurrences, the value is 0.	Y	N/A
`mod`	integer	A modifier indicating which recurrences to delete, or semicolon-separated list of modifiers. If a list, must have same number or elements as `uid` list. One of the following values: 1 = THISINSTANCE 2 = THISANDFUTURE 3 = THISANDPRIOR 4 = THISANDALL	Y	N/A
`notify`	integer 0,1	When 1, notify attendees of the change. When 0, do not notify attendees.	N	`0`

Parameter	Type	Purpose	Required	Default
brief	integer (0,1)	When 1, print out a brief version of output. When 0, print complete output of event/todo data. Applies only when output type is Javascript	N	0
fmt-out	string	The output content-type in which data is to be returned. One of the following: text/calendar text/xml text/js	N	text/js

Purpose Use this command to delete the specified todo from the specified calendar. If the rid parameter is passed, the command also deletes recurrences, as specified by the mod parameter. See "Storing Recurrences."

To delete multiple todos, specify a semicolon-separated list for the uid, rid, and mod parameters. The three lists must have the same number of elements. Each list element corresponds to the same element in the other two lists.

When the notify value is 1, sends an IMIP cancellation message to the e-mail attendees of the todo.

deletetodos_by_range

Purpose Delete todos in a range from a calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
id	unique id string	The session identifier. Required unless the calendar is public.	N	null
dtstart	ISO8601 DateTime string	Start time and date of todos to be deleted. A value of 0 means delete all todos up to a specified start-time.	N	0

Parameter	Type	Purpose	Required	Default
`dtend`	ISO8601 DateTime string	End time and date of todos to be deleted. A value of 0 means delete all todos up to the latest time.	N	`0`
`calid`	string	Semicolon-separated list of calendar identifiers from which to delete todos. For example, `jdoe,jdoe;susan,jdoe;huey;susan`.	N	username of current user
`brief`	integer (0,1)	When 1, print out a brief version of output. When 0, print complete output of event/todo data. Applies only when output type is Javascript	N	0
`fmt-out`	string	The output content-type in which data is to be returned. One of the following: `text/calendar` `text/xml` `text/js`	N	`text/js`

Purpose Use this command to delete the todos that fall completely within the specified range from the specified calendars. If a range is not specified, it deletes all todos. If the `rid` parameter is passed, the command also deletes recurrences, as specified by the `mod` parameter. See “Storing Recurrences.”

When the `notify` value is 1, sends an IMIP cancellation message to the e-mail attendees.

export

Purpose Export events and todos from a calendar to a file.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique id string	The session identifier. Required unless the calendar is public.	N	`null`
`calid`	semicolon-separated list of strings	A list of identifiers for calendars from which to export events and todos. The user must have read access to all calendars in the list.	N	user name of current user
`dtstart`	iCal DateTime string	Start time and date of events and todos to export. A value of 0 means export all components from the earliest date to the end date.	N	0
`dtend`	iCal DateTime string	End time and date of the events and todos to export. A value of 0 means export all components from the start date to the latest date.	N	0
`content-out`	string	Content type for output file. One of the following: `text/calendar` `text/xml`	Y	N/A

Purpose

Use this command to export events and todos from one or more specified calendars to a file. The contents of the file can later be imported to a calendar using the `import` command. The command creates a file called `export.ics` or `export.xml`, depending on the value of the `content-out` parameter.

If you do not specify either the starting or ending date, all events and todos in the calendars are added to the file. If you specify a starting and ending date, the command exports only events and todos in the calendars that fall within the time range. Specify starting and ending dates in UTC time (indicated by `Z` at the end of the datetime).

You must use this command with an HTTP `POST` (unlike other commands, which can be used with an HTTP `GET`).

The following HTTP POST message exports all component of the calendars jdoe and huey to an iCal file named export.ica:

```
POST
/export.wcap?id=t95qm0n0es3bo35r&calid=jdoe;huey&dtstart=0&dtend=0
  &content-out=text/calendar
Content-type: multipart/form-data;
boundary=---------------------------41091400621290
Content-Length: 47
-----------------------------41091400621290--
WinNT; U)
Host: jdoe:12345
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png
*/*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
```

The following HTML could be used to generate POST messages using the export command, producing files in both iCal and XML formats:

```
<form METHOD=POST ENCTYPE="multipart/form-data" NAME="Huey.ics"
ACTION="http://myserver:12345/export.wcap?id=t9u9m0eh8x5pu9b
  &calid=jdoe;huey&dtstart=0&dtend=0&content-out=text/calendar">
<ul>
<li>Press Export ICAL Now:<input type="submit" value="Export ICAL now">
</li> </ul> </form>
<form METHOD=POST ENCTYPE="multipart/form-data" NAME="Huey.xml"
ACTION="http://myserver:12345/export.wcap?id=t9u9m0eh8x5pu9b
  &calid=jdoe;huey&dtstart=0&dtend=0&content-out=text/xml">
<ul>
<li>Press Export XML Now:<input type="submit" value="Export XML now”>
</ul> </form>
```

The following is part of the output from the query, contained in the returned page:

```
BEGIN:VCALENDAR
METHOD:PUBLISH
VERSION:2.0
BEGIN:VEVENT
UID:tm-001
RECURRENCE-ID:19990519T010000Z
DTSTAMP:19990603T221548Z
SUMMARY:Calendar Staff
DTSTART:19990518T170000Z
DTEND:19990518T190000Z
CREATED:19990603T024254Z
LAST-MODIFIED:19990603T024254Z
PRIORITY:1
```

```
SEQ:1
GEO:37.463581;-121.897606
DESC:This is the description for event with UID = tm-001
URL:http://myserver.mycompany.com/susan?uid=tm-001
LOCATION:Green Conference Room
STATUS:CONFIRMED
TRANSP:OPAQUE
END:VEVENT
BEGIN:VEVENT
UID:tm-001
RECURRENCE-ID:19990526T010000Z
DTSTAMP:19990603T221548Z
SUMMARY:Calendar Staff
DTSTART:19990525T170000Z
DTEND:19990525T190000Z
CREATED:19990603T024254Z
LAST-MODIFIED:19990603T024254Z
PRIORITY:1
SEQ:1
GEO:37.463581;-121.897606
DESC:This is the description for event with UID = tm-001
URL:http://myserver.mycompany.com/susan?uid=tm-001
LOCATION:Green Conference Room
STATUS:CONFIRMED
TRANSP:OPAQUE
END:VEVENT
```

fetchcomponents_by_range

Purpose Retrieve calendar events and todos.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique id string	The session identifier. Required if the calendar is not public.	N	`null`
`dtstart`	ISO8601 DateTime string	Start time and date of events to be returned. A value of 0 means get all events up to a specified start-time.	N	`0`
`dtend`	ISO8601 DateTime string	End time and date of events to be returned. A value of 0 means get all events up to a specified end-time.	N	`0`

Parameter	Type	Purpose	Required	Default
`calid`	string	Semicolon-separated list of calendar identifiers from which to retrieve components. For example, `jdoe,jdoe;susan, jdoe;huey;susan`.	N	username of current user
`brief`	integer (0,1)	When 1, print out a brief version of output. When 0, print complete output of event/ todo data. Applies only when output type is Javascript	N	0
`fmt-out`	string	The output content-type in which data is to be returned. One of the following: `text/calendar` `text/xml` `text/js`	N	text/js
`maxResults`	integer	The maximum number of events and todos to be returned. When 0, no maximum is applied and the command returns all events and todos found.	N	0

Purpose Use this command to retrieve properties, events, and todos from one or more specified calendars. You must specify the `id` parameter with the command unless the specified calendar is a public calendar. The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data is returned in the default JavaScript format.

For each calendar specified in `calid`, the server returns the calendar's properties and the events and todos of that calendar that fall within the range specified by `dtstart` and `dtend`. Specify these in the ISO8601 Date String format. For example, to get the events for calendars `jdoe` and `susan` from Jan. 1, 1999 to Feb. 1, 1999, use the following URL:

```
http://myserver.mycompany.com:81/
fetchcomponents_by_range.wcap?id=b5q2o8ve2rk02nv9t6&calid=jdoe
```

If neither the starting nor ending date-time is specified, all events and todos of the calendar are returned, up to the specified maximum.

Note If a calendar does not have a valid timezone (`tzid`) associated with it, the GMT timezone is used.

If you specify a maximum *n*, the command returns up to the first *n* events and first *n* todos in the specified range. For example, if you specify a `maxResults` value of 75, the returned Javascript would contain the following variables

```
var maxResults=75 /* maximum cap passed in */
var size=75       /* event size is capped to 75 */
var todosize=28   /* todo size not affected since it is less than 75 */
```

If the `maxResults` parameter is set to 0 or is not passed, then the returned Javascript does not contain the `var maxResults` statement.

If the operation is successful, the error number of 0 is appended to the error string. If a calendar cannot be accessed or is missing, the error number is appended to the error string.

For an example of the returned output from this command, see "Retrieving Components from Calendars."

fetchevents_by_id

Purpose Retrieve specific calendar events.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique id string	The session identifier. Required if the calendar is not public.	N	`null`
`uid`	sting	The unique identifier for the event.	Y	N/A
`rid`	ISO8601 DateTime string	The recurrence identifier for the event. For a nonrecurring event, set to 0.	N	`0`

Parameter	Type	Purpose	Required	Default
mod	integer	A modifier indicating which recurrences to retrieve. One of the following values: 1 = THISINSTANCE 2 = THISANDFUTURE 3 = THISANDPRIOR 4 = THISANDALL	N	1
calid	string	The unique identifier for the calendar from which to retrieve events.	N	username of current user
brief	integer (0,1)	When 1, print out a brief version of output. When 0, print complete output of event/ todo data. Applies only when output type is Javascript	N	0
fmt-out	string	The output content-type in which data is to be returned. One of the following: text/calendar text/xml text/js	N	text/js

Purpose Use this command to retrieve the specified event and its recurrences from the specified calendar. You must specify the `id` parameter with the command unless the specified calendar is a public calendar. The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data is returned in the default JavaScript format.

The command returns recurrences as specified by the `mod` parameter. See "Storing Recurrences."

fetchtodos_by_id

Purpose Retrieve specific calendar todos.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique id string	The session identifier. Required if the calendar is not public.	N	`null`
`uid`	sting	The unique identifier for the todo.	Y	N/A
`rid`	ISO8601 DateTime string	The recurrence identifier for the todo. For a nonrecurring todo, set to 0.	N	`0`
`mod`	integer	A modifier indicating which recurrences to retrieve. One of the following values: 1 = THISINSTANCE 2 = THISANDFUTURE 3 = THISANDPRIOR 4 = THISANDALL	N	`1`
`calid`	string	The unique identifier for the calendar from which to retrieve todos.	N	username of current user
`brief`	integer (0,1)	When 1, print out a brief version of output. When 0, print complete output of event/ todo data. Applies only when output type is Javascript	N	`0`

`fmt-out`	string	The output content-type in which data is to be returned. One of the following:
		`text/calendar`
		`text/xml`
		`text/js`
			N	`text/js`

Purpose Use this command to retrieve the specificed todo and its recurrences from the specified calendar. You must specify the `id` parameter with the command unless the specified calendar is a public calendar. The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data is returned in the default JavaScript format.

The command returns recurrences as specified by the `mod` parameter. See "Storing Recurrences."

get_all_timezones

Purpose Retrieve data about all timezonessupported by the server.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier. Required if the calendar is not public.	Y	`null`
`dtstart`	ISO8601 DateTime string	Start date of crossover values to retrieve. A value of 0 means get all crossover dates from the first known year (1987).	N	`0`
`dtend`	ISO8601 DateTime string	End date of the crossover values to retrieve. A value of 0 means get all crossover dates until the last known year (2087).	N	`0`

`fmt-out`	string	The output content-type in which data is to be returned. One of the following:
		`text/calendar`
		`text/xml`
		`text/js`
			N	`text/js`

Purpose Use this command to retrieve data about all timezonesthat are supported by the server. The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data is returned in the default JavaScript format.

The crossover dates of a timezone are those dates on which daylight savings begins or ends; for example, 10/26/98. If you specify a range of years with the `dtstart` and `dtend` parameters, the command returns only the crossover dates for the years within the range. Otherwise, it returns all crossover dates from the first to the last known year (1987-2087).

This command returns a timezone list, which is a JavaScript array containing the identifying feature of each timezone, including another JavaScript array of the crossover dates. The following is an example of a timezone array element where crossover dates have been limited to the years from mid-1998 to 2006:

```
timezoneList[20] = new TZ('America/Los_Angeles', 'PST', 'PDT',
```

```
  '-0800', '-0700', new
Array ('19981025T090000Z', '19990404T100000Z', '19991031T090000Z',
      '20000402T100000Z', '20001029T090000Z','20010401T100000Z',
      '20011028T090000Z', '20020407T100000Z', '20021027T090000Z',
      '20030406T100000Z', '20031026T090000Z', '20040404T100000Z',
      '20041031T090000Z', '20050403T100000Z', '20051030T090000Z',
       '20060402T100000Z', '20061029T090000Z'))
```

get_calprops

Purpose Retrieve calendar properties.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique id string	The session identifier.	N	`null`
`calid`	semicolon-separated list of strings	A list of calendar identifiers for the calendars from which to retrieve properties.	N	name of current user

`fmt-out`	string	The output content-type in which data is to be returned. One of the following:
		`text/calendar`
		`text/xml`
		`text/js`
			N	`text/js`

Purpose Use this command to retrieve the calendar properties for the specified calendars. The command returns the preferences in the specified format.

The following errors can occur:

- If the calendar exists, but the user does not have READ access to it, the `layer_errno[]` array variable is set to the value 1 for that calendar's index.
- If the calendar is not found, the `layer_errno[]` array variable is set to the value 2 for that calendar's index.

- If the fetch fails for any calendar, its error number is set to the `GET_CALPROPS_FAILED` value.

The command returns a page with the following property information for the specified calendars:

- relative ID
- display name
- parent calendar ID
- timezone ID
- read-access value (`0` = cannot read, `1` = can read)
- write-access value (`0` = cannot read, `1` = can read)
- character set (if empty, default is `us-ascii`)
- language (if empty string, default is `en` = English)
- cal-master (contact information, usually email address of primary owner)
- description
- last-modified time
- created time
- primary owner
- other owner list (semicolon-separated)
- category list (semicolon-separated)

In the following example, the command tries to get the calendar properties for the calendar `jdoe`, `susan`, `pub`, `huey`, and `hasdf`, in that order.

```
http://myserver/
get_calprops.wcap?id=2mu95r5so0hq68ts6q3&calid=jdoe;susan;pub;huey;hasd
f
```

In this case, the user does not have read access to the calendar `susan`, and the calendar `hasdf` does not exist in the database.

- No data is returned for the calendar `susan`, and the `layer_errno[1]` value is set to 1 to indicate that user does not have access to that calendar.
- No data is returned for the calendar `hasdf`, and the `layer_errno[5]` value is set to 2 to indicate that the calendar does not exist.

Data is returned for the remaining calendars as follows, and the `layer_errno[]` value for these calendars' indexes is set to 0.

For an example of the JavaScript output from this command, see “Retrieving Calendar Properties.”

get_guids

Purpose Generate a set of globally unique identifiers.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
guidCount	integer	Number of GUIDs to return.	N	1
fmt-out	string	Content type of output. One of the following values: text/calendar text/xml text/js	N	text/js

Purpose Use this command to generate the specified number of globally unique identifiers (GUIDs). The client need not be authenticated to call this command.

The following example URLs and their return output show the three formats:

```
http://jdoe.mycompany.com/get_guids.wcap?guidCount=10
  &fmt-out=text/calendar
BEGIN:VCALENDAR
VERSION:2.0
PRODID:iPlanet Calendar Server 4.0
X-NSCP-GUID0:e5e4b537465600000b000000c3000000
X-NSCP-GUID1:e5e4b537d47900000c000000c3000000
X-NSCP-GUID2:e5e4b537961400000d000000c3000000
X-NSCP-GUID3:e5e4b5373d3a00000e000000c3000000
X-NSCP-GUID4:e5e4b537f31400000f000000c3000000
X-NSCP-GUID5:e5e4b5378259000010000000c3000000
X-NSCP-GUID6:e5e4b537b026000011000000c3000000
X-NSCP-GUID7:e5e4b537c263000012000000c3000000
X-NSCP-GUID8:e5e4b537241f000013000000c3000000
X-NSCP-GUID9:e5e4b537e733000014000000c3000000
END:VCALENDAR
```

```
http://jdoe.mycompany.com/get_guids.wcap?guidCount=10&fmt-out=text/xml
<?xml version="1.0" encoding="UTF-8"?>
<iCalendar>
<iCal>
<X-NSCP-GUID0>fde4b537d604000015000000c3000000<X-NSCP-GUID0>
<X-NSCP-GUID1>fde4b5379478000016000000c3000000<X-NSCP-GUID1>
<X-NSCP-GUID2>fde4b5372a6d000017000000c3000000<X-NSCP-GUID2>
<X-NSCP-GUID3>fde4b537a355000018000000c3000000<X-NSCP-GUID3>
<X-NSCP-GUID4>fde4b5377768000019000000c3000000<X-NSCP-GUID4>
```

```
<X-NSCP-GUID5>fde4b5376e2e00001a000000c3000000<X-NSCP-GUID5>
<X-NSCP-GUID6>fde4b537a07700001b000000c3000000<X-NSCP-GUID6>
<X-NSCP-GUID7>fde4b537744700001c000000c3000000<X-NSCP-GUID7>
<X-NSCP-GUID8>fde4b537ab1f00001d000000c3000000<X-NSCP-GUID8>
<X-NSCP-GUID9>fde4b5371d2200001e000000c3000000<X-NSCP-GUID9>
</iCal>
</iCalendar>
```

In the final example, the `fmt-out` parameter defaults to `text/java`:

```
http://jdoe.mycompany.com/get_guids.wcap?guidCount=10
<html><head><script>
function color(s) { if (s) document.bgColor=s }
var id='0'
var userid=''
var calid=''
var errno=new Array()
var guid=new Array()
guid[0]=9ee4b5375778000001000000c3000000
guid[1]=9ee4b5376a52000002000000c3000000
guid[2]=9ee4b5375a47000003000000c3000000
guid[3]=9ee4b537bf01000004000000c3000000
guid[4]=9ee4b537e05e000005000000c3000000
guid[5]=9ee4b537be5a000006000000c3000000
guid[6]=9ee4b537d5440000007000000c3000000
guid[7]=9ee4b5372867000008000000c3000000
guid[8]=9ee4b5375731000009000000c3000000
guid[9]=9ee4b537830600000a000000c3000000
parent.ceCB(window.name)
</script></head></html>
```

get_session

Purpose Retrieve session information for a user.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`user`	string	Administrative user ID (must be contained in the `service.http.admins` server preference).	Y	N/A
`password`	string	Administrative user password.	Y	N/A
`tid`	integer	The target session identifier for the information to be retrieved.	N	N/A
addr	IP address	The IP address of the user whose information is to be retrieved.	N	N/A
userid	string	The user ID of the user whose information is to be retrieved.	N	N/A

Purpose Use this command to retrieve the session information for the specified user and session. The command returns an HTML page containing the response string.

If one of the parameters `tid`, `addr`, or `userid` is not specified, the command returns information for all users.

Only users with administrative privilege can use this command. The command must be sent to the administrative port.

The following is an example of the response string in the returned page:

```
<html>
<!-- admin
id=927152557 userid=jdoe addr=208.12.60.179 login=19990519T152156Z
access=19990519T152156Z id=927166888 userid=admin addr=208.12.60.179
login=19990519T152203Z access=19990519T152203Z
-->
</html>
```

get_userprefs

Purpose Retrieve the calendar preferences for the current user.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier.	N	`null`

`fmt-out`	string	The output content-type in which data is to be returned. One of the following:
		`text/calendar`
		`text/xml`
		`text/js`
			N	`text/js`

Purpose Use this command to retrieve the calendar preferences for the current user. The command returns the preferences in Javascript format.

Note: It is recommended that you prefix all calendar-specific user preferences with "ce."

For an example of the output returned from this command, see “Retrieving User Preferences.”

import

Purpose Import events and todos from a file to a calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier. Required unless the calendar is public.	Y	`null`
`calid`	integer	Identifier of a calendar to which to import event.	Y	N/A
`dtstart`	iCal DateTime string	Start time and date of events and todos to import. A value of 0 means import all components from the earliest date to the end date.	N	`0`

Parameter	Type	Purpose	Required	Default
`dtend`	iCal DateTime string	End time and date of the events and todos to import. A value of 0 means import all components from the start date to the latest date.	N	`0`

`content-in`	string	Content type of input data. One of the following values:
		`text/calendar`
		`text/xml`
			Y	N/A

Purpose Use this command to import to the specified calendar events and todos that have previously been exported to a file using the `export` command. You must specify the content type of the file.

If you do not specify either the starting or ending date, all events and todos in the file are added to the specified calendar. If you specify a starting and ending date, the command imports only events and todos in the file that fall within the time range. Specify starting and ending dates in UTC time (indicated by `Z` at the end of the datetime).

You must use this command with an HTTP `POST message`(unlike other commands, which can be used with an HTTP `GET message`). You attach to the `POST` message the file containing the exported events and todos. This file must be in either iCal (`.ics`) or XML (`.xml`) format.

The following `POST` message imports the attached iCal file to the calendar `jdoe` using the `import` command:

```
POST /import.wcap?id=t95qm0n0es3bo35r&calid=jdoe&dtstart=0&dtend=0
Content-type: multipart/form-data;
boundary=---------------------------33111928916708
Content-Length: 679
-----------------------------33111928916708
Content-Disposition: form-data; name="Upload";
filename="C:\TEMP\ical1.ics"
BEGIN:VCALENDAR
BEGIN:VEVENT
DTSTART:19990105T100000
DTEND:19990105T110000
DTSTAMP:19990104T120000
CREATED:19990105T110000Z
LAST-MODIFIED:19990104T120000Z
SUMMARY:Weekly QA Meeting
```

```
UID:random-uid001
END:VEVENT
BEGIN:VEVENT
DTSTART:19990106T100000
DTEND:19990106T110000
DTSTAMP:19990104T120000
CREATED:19990105T110000Z
LAST-MODIFIED:19990104T120000Z
SUMMARY:Weekly QA Meeting 2
UID:random-uid002
END:VEVENT
END:VCALENDAR
----------------------------33111928916708--
```

The following HTML form creates such a POST message, attaching a file that the user specifies:

```
<FORM METHOD=POST ENCTYPE="multipart/form-data"
ACTION="http://myserver:12345/import.wcap?id=t95qm0n0es3bo35r
  &calid=jdoe&dtstart=0&dtend=0&content-in=text/calendar">
  <ol>
  <li>file to import:<input type="file" accept="text" name="Upload">
  </li>
  <li>Press Import Now:<input type="submit" value="Import Now"></li>
  </ol>
  </FORM>
```

login

Purpose Authenticate as a specific user.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`user`	string	The user's name.	N	`null`
`password`	string	The user's password.	N	N/A
`lang`	enum	The user's preferred language.	N	`null`

Parameter	Type	Purpose	Required	Default
`refresh`	integer (0, 1)	When 1, return only the session identifier. When 0, return the full JavaScript output.	N	0

`fmt-out`	string	The output content-type in which data is to be returned. One of the following:
		`text/calendar`
		`text/xml`
		`text/js`
		If `text/calendar` or `text/xml`, the refresh parameter is automatically set to 1.
			N	`text/js`

Purpose Use this command to log into Calendar Server as a specific user, authenticating the user to the server with a user name and password convention.

- The user name is a plain text string that uniquely identifies the user to the server. This user name could, for example, be the same as a user's e-mail address.
- The password is also plain text. In future, when the server supports SSL, the password will be encoded.

The authentication method can use either LDAP or CSAPI.

- LDAP authentication tries to authenticate a user to an LDAP server. (Currently the server supports only this method.)
- CSAPI authentication allows you to tie in an existing user authentication method. CSAPI authentication returns 1 on success, 0 on failure.

If the user fails to authenticate correctly, the login window reappears with an error noting a failure to log in.

The following is an example of a URL using this command:

```
http://myserver/login.wcap?user=jdoe&password=mypword
```

The returned page contains the session identifier in a line such as the following:

```
var id='bu9p3eb8x5p2nm0q3'
```

This is the value that you pass to many WCAP commands in order to gain access to private calendars. It is also a required parameter for certain commands, such as `logout`.

If you specify 0 for the `refresh` parameter, the command returns JavaScript output containing the location of the entry page to the Calendar Server's user interface. For example:

```
HTTP/1.0 302 OK
Date: Tue, 11 May 1999 22:38:33 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache
Content-Length: 0
Last-modified: Tue, 11 May 1999 22:38:33 GMT
Location:
http://jdoe.mycompany.com/en/main.html?id=er6en05tv6n3bv9&lang=en
 &host=http://jdoe.mycompany.com/

<html><head><script>
function color(s) { if (s) document.bgColor=s }
var id='er6en05tv6n3bv9'
var userid='jdoe'
var calid='jdoe'
var errno=new Array()
var errstr=''
</script></head>
<body bgcolor='9999CC' onLoad=parent.ceCB(window.name)>
```

logout

Purpose Terminate the session of the current user.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier.	Y	N/A

`fmt-out`	string	The output content-type in which data is to be returned. One of the following:
		`text/calendar`
		`text/xml`
		`text/js`
			N	`text/js`

Purpose This command ends the specified session of the current user, and deletes the session instance of the user in the session table. The user is returned to the login screen.

The following is an example of a URL using this command:

```
http://myserver/logout.wcap?id=bu9p3eb8x5p2nm0q3
```

ping

Purpose Determine whether the server is active.

Parameters This command takes no parameters.

Purpose This command returns a minimal HTML page to indicate that the server responded.

Only users with administrative privilege can use this command.

refresh

Purpose Refresh the server configuration.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`user`	string	The administrative user's name.	Y	`null`
`password`	string	The administrative user's password.	Y	N/A

Purpose This command causes the server to reload the `server.conf` configuration file. You can use this command to change configuration preferences while the server is running.

Only users with administrative privilege can use this command.

search_calprops

Purpose Search for a calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier. Required unless the calendar is public.	N	`null`
`search-string`	string	The string to search for in calendars.	Y	N/A
`searchOpts`	integer 0,1,2,3	How to perform the search. One of the following: 0 = CONTAINS 1 = BEGINS_WITH 2 = ENDS_WITH 3 = EXACT	N	`0`
`maxResults`	integer	The maximum number of results to return.	N	`200`
`primaryOwner`	integer (0,1)	If 1, search `primaryOwner` property.	N	`0`
`calid`	integer (0,1)	If 1, search `calid` property.	N	`0`, unless both `owner` and `name` are 1
`name`	integer (0,1)	If 1, search `name` property.	N	`0`

Purpose This command searches for a calendar using the query specified by `searchOpts`. It returns all calendars where a string in the specified properties matches the specified string in the specified way, up to the specified maximum number of matches.

Specify whether to search for the value of a specific property (`primaryOwner`, `calid`, or `name`) by setting that parameter to `1`. If both `primaryOwner` and `name` are set to `1`, `calid` defaults to `1`.

The following example URL searches the primary owner property (`primaryOwner=1`) in all calendars to see if it contains (`searchOpts=0`) the string "`susan`".

```
http://myserver/search_calprops.wcap?id=b2nehr3eq6bh5s
   &search-string=susan&primaryOwner=1&searchOpts=0&maxResults=50
```

For an example of the output from this command, see “Retrieving Calendar Properties.”

set_calprops

Purpose Set the calendar properties of a calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier. Required unless the calendar is public.	N	`null`
`calid`	string	Identifier of the calendar to modify.	Y	N/A
`name`	string	The new human-readable name of the calendar.	Y	N/A

`read`	integer	The new read-access value of the calendar. The value can be one of the following:
		`0 PRIVATE`
		`1 PUBLIC`
			Y	N/A

Parameter	Type	Purpose	Required	Default

`write`	integer	The new write-access value of the calendar. The value can be one of the following:
		`0 PRIVATE`
		`1 PUBLIC`
			Y	N/A

`categories`	semicolon-separated list of strings	The new categories list of the calendar.	Y	N/A
`owners`	semicolon-separated list of strings	The new owner list of the calendar.	Y	N/A
`description`	string	The description of the calendar.	Y	N/A
`charset`	string	The character set for the calendar.	Y	N/A
`lang`	string	The language of the calendar.	Y	N/A
`master`	string	The email contact for the calendar.	Y	N/A
`tzid`	string	The timezone for this calendar.	Y	N/A
multiple	integer	The number of calendars for which to set these preferences.	N	`0`
cal	encoded string	A list of property parameters for one calendar. Pass multiple instances of this parameter to set the properties of multiple calendars.	N	N/A

Purpose Use this command to set the properties of the specified calendar to the specified values. Calendar properties include the calendar's name, read and write permission values, the list of owners, and the list of categories.

You must pass all parameters, as the command overwrites the previous calendar properties.

- Only the primary owner of the calendar can modify the `owners` property. The primary owner of a calendar cannot be changed.
- Only owners of the calendar can modify the `read`, `write` and `categories` properties of the calendar.
- To add, delete, or modify an owner or category you must specify the entire list of new owners or categories.

For example, the following URL sets the read and write access of the calendar `jdoe` to `PUBLIC`, and specifies a categories list with two entries, `business` and `meeting`:

```
http://myserver?set_calprops.wcap?id=dfasdfzd3ds&calid=jdoe
   &read=1&categories=business;meeting&write=1
```

To set properties of several calendars at one time, set the `multiple` parameter to the number of calendars to be set, then pass a `cal` parameter for each calendar. The `cal` parameter contains an encoded string with the complete property parameter list for the identified calendar. In this string, replace all special characters with a percent character (%), followed by the hexadecimal ASCII code for the special character. ASCII hex codes for common special characters are as follows:

Table 2.6

Character	Code
=	%3D
&	%26
"	%22

For example, the following URL modifies three calendars with IDs `xxxx`, `yyyy`, and `zzzz`, setting the descriptions to `X-Calendar`, `Y-Calendar`, and `Z-Calendar`, respectively:

```
http://jdoe.mycompany.com?id=fasdfzd3ds
   &multiple=3
   &cal=calid%3Dxxxx%26description%3DX-Calendar
   &cal=calid%3Dyyyy%26description%3DY-Calendar
   &cal=calid%3Dzzzz%26description%3DZ-Calendar
```

set_userprefs

Purpose Modify the preferences or password for a session.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier. Required unless the calendar is public.	Y	`null`
`add_attrs`	string	Add a new preference.	N	N/A
`del_attrs`	string	Delete a preference.	N	N/A
`set_attrs`	string	Modify a preference value.	N	N/A
`fmt-out`	string	The output content-type in which data is to be returned. One of the following: `text/calendar` `text/xml` `text/js`	Y	`text/js`

Purpose Use this command to modify the preferences for the current user. For example, the following URL sets the value of a new preference, `ceBgcolor`, to `black`:

```
http://jdoe/set_userprefs.wcap?id=b5q2o8ve2rk02nv9t6
   &add_attrs=ceBgcolor=black
```

The function returns the result of a call to the `get_userprefs` command.

shutdown

Purpose Shut down the calendar server.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
user	string	Administrative user ID (must be contained in the service.http.admins server preference).	Y	N/A
password	string	Administrative user's password.	Y	N/A

Purpose Use this command to shut down the currently running calendar server. The command returns an HTML page containing a shutdown message.

Only users with administrative privilege can use this command. The command must be sent to the administrative port.

The server returns a minimal HTML page containing the shutdown message. For example:

```
HTTP/1.0 200
Date: Thu, 13 May 1999 01:19:25 GMT
Content-type: text/html; charset=iso-8859-1
Content-length: 483
Last-modified: Thu, 13 May 1999 01:19:25 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache

<html>
<!-- admin
Shutting down Calendar Server
-->
</html>
```

storeevents

Purpose Add events to a calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	unique ID string	The session identifier. Required unless the calendar is public.	N	`null`
`calid`	integer	Identifier of a calendar in which to store the event.	Y	N/A
`dtstart`	iCal DateTime string	Start time and date of the event.	Y	N/A
`dtend`	iCal DateTime string	End time and date of the event.	N	N/A
`summary`	string	Summary of the event. A string of any length. To include spaces in the string, use the code `%20`.	N	"default summary "
`desc`	string	Purpose of the event. A string of any length. To include spaces in the string, use the code `%20`. If not passed, description is set to the summary value.	N	value in summary field
`uid`	string	Unique identifier of the event to be stored.	N	"default uid"
`location`	string	Location of the event.	N	""

`icsClass`	string	Class of the event. One of the following values:
		`PUBLIC`
		`PRIVATE`
		`CONFIDENTIAL`
			N	PRIVATE

`icsUrl`	string	URL of the event.	N	""
`duration`	iCal duration string	Duration of the event. If an event has both a `duration` and a `dtend`, the duration is ignored.	N	N/A

Parameter	Type	Purpose	Required	Default
`status`	integer	A code for the status of the event. One of the following values: 0 CONFIRMED 1 CANCELLED 2 TENTATIVE 3 NEEDS_ACTION 4 COMPLETED 5 IN_PROCESS 6 DRAFT 7 FINAL	N	N/A
`isAllDay`	integer	If passed with a value of 1, the event is all day. If not passed or passed with a value of 0, the event is not all day.	N	`0`
`geo`	semicolon-separated string of two floats	Latitude and longitude of the geographical location of the event.	N	`0;0`
`seq`	integer	Sequence number of the event.	N	`0`
`rrules`	semicolon-separated list of quoted recurrence-rule strings	Recurrence rules of the event. Each rule value must be enclosed in quotes. See "Storing Recurrences."	N	N/A
`exrules`	semicolon-separated list of quoted recurrence-rule strings	Exclusionary recurrence rules of the event. Each rule value must be enclosed in quotes. See "Storing Recurrences."	N	N/A

Parameter	Type	Purpose	Required	Default
`rdates`	semicolon-separated list of iCal date strings	Recurrence dates of the event.	N	N/A
`exdates`	semicolon-separated list of iCal date strings	Exclusionary recurrence dates of the event.	N	N/A
`rid`	iCal DateTime string	Recurrence identifier of the event.	N	N/A
`mod`	integer	A modifier indicating which recurrences to store. One of the following values: 1 = THISINSTANCE 2 = THISANDFUTURE 3 = THISANDPRIOR 4 = THISANDALL	N	N/A
`rchange`	integer (0,1)	Whether to expand a recurring event. Expand if 1, do not exand if 0.	N	`1`
`priority`	integer (0-10)	The priority of the event. 0 is the lowest priority, and 10 is the highest.	N	`0`
`relatedTos`	semicolon-separated list of quoted strings	Other events to which this event is related.	N	N/A
`resources`	semicolon-separated list of strings	The resources associated with the event.	N	N/A
`categories`	semicolon-separated list of strings	The categories of the event.	N	N/A

Parameter	Type	Purpose	Required	Default
attendees	semicolon-separated list of strings	The attendees of the event.	N	N/A
contacts	semicolon-separated list of strings	Contacts for the event.	N	N/A
alarmEmails	semicolon-separated list of email addresses	Recipients of alarm notifications for the event.	N	N/A
alarmStart	iCal DateTime string	The time at which to send an alarm notification of the event.	N	N/A
attachments	semicolon-separated list of strings	Items to be attached to notification emails.	N	N/A
tzid	quoted string	A timezone. All dates are interpreted with reference to this timezone. If not passed, all dates are interpreted as being in coordinated universal time (UTC or Z format).	N	"GMT"
notify	integer 0,1	When 1, notify attendees of the change. When 0, do not notify attendees.	N	0

Parameter	Type	Purpose	Required	Default
`brief`	integer (0,1)	When 1, print out a brief version of output. When 0, print complete output of event/todo data. Applies only when output type is Javascript	N	0
`fmt-out`	string	The output content-type in which data is to be returned. One of the following: `text/calendar` `text/xml` `text/js`	N	`text/js`

Purpose Use this command to create an event with the specified attributes and store it in the specified calendar in the database. See "Examples of Command Usage" on page 79 for examples of usage.

The ending date and time (`dtend`) overrides the duration. If you specify both `duration` and `dtend`, the duration is ignored.

Specify the duration in the `iCal` format. For example:

- `P1Y2M3DT1H30M10S` represents a duration of 1 year, 2 months, 3 days, 1 hour, 30 minutes, 10 seconds
- `PT1H30M` represents a duration of 1 hour, 30 minutes
- `P1D` represents a duration of 1 day
- `PT15M` represents a duration of 15 minutes

Notice that the `T` in the string separates the date information (year, month, day) from the time information (hour, minute, second).

The command creates and stores recurrences as specified by the recurrence rule, `rid`, `mod`, and `rchange` parameters. See "Storing Recurrences."

When the `notify` value is `1`, sends an IMIP publish message to the e-mail attendees of the event.

If the `fmt-out` parameter is set to `text/calendar` or `text/xml`, the command returns the error value in an HTML comment; for example:

```
<!-- store_errno="0" -->
```

If the `fmt-out` parameter is set to `text/js`, the command returns the javascript from a `fetchcomponents_by_range.wcap` command on all data.

This command cannot modify a linked event.

For an example that uses all of the parameters of this command, see “Creating and Storing Events.”

storetodos

Purpose Add a todo to a calendar.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`id`	integer	The session identifier. Required unless the calendar is public.	N	`null`
`calid`	integer	Identifier of a calendar in which to store the todo.	Y	N/A
`dtstart`	iCal DateTime string	Start time and date of the event.	Y	N/A
`due`	iCal DateTime string	End time and date of the event. .	N	N/A
`completed`	iCal DateTime string	Completion date of the todo. A value of 0 means the todo is not yet completed.	N	`0`
`summary`	string	Summary of the todo. A string of any length. To include spaces in the string, use the code `%20`.	N	"default summary"

Parameter	Type	Purpose	Required	Default
description	string	Purpose of the todo. A string of any length. To include spaces in the string, use the code %20. If not passed, description is set to the summary value.	N	value in summary field
uid	string	Unique identifier of the todo to be stored.	N	"default uid"
location	string	Location of the todo.	N	""
icsClass	string	Class of the todo. One of the following values: PUBLIC PRIVATE CONFIDENTIAL	N	PRIVATE
icsUrl	string	URL of the todo.	N	""
duration	iCal duration string	Duration of the todo.	N	N/A
status	integer	A code for the status of the todo. One of the following values: 0 CONFIRMED 1 CANCELLED 2 TENTATIVE 3 NEEDS_ACTION 4 COMPLETED 5 IN_PROCESS 6 DRAFT 7 FINAL	N	N/A
isAllDay	NONE	If passed, the todo is an all-day todo. If not passed, the todo is not all day. The parameter takes no value.	N	N/A
geo	semicolon-separated string of two floats	Latitude and longitude of the geographical location of the todo.	N	0;0

Parameter	Type	Purpose	Required	Default
seq	integer	Sequence number of the todo.	N	0
percent	integer (0-100)	Percentage completion of the todo.	N	0
rrules	semicolon-separated list of quoted recurrence-rule strings	Recurrence rules of the todo.	N	N/A
exrules	semicolon-separated list of quoted recurrence-rule strings	Exclusionary recurrence rules of the todo.	N	N/A
rdates	semicolon-separated list of iCal date strings	Recurrence dates of the todo.	N	N/A
exdates	semicolon-separated list of iCal date strings	Exclusionary recurrence dates of the todo.	N	N/A
rid	iCal DateTime String	Recurrence identifier of the todo.	N	N/A
mod	integer	A modifier indicating which recurrences to store. One of the following values: 1 = THISINSTANCE 2 = THISANDFUTURE 3 = THISANDPRIOR 4 = THISANDALL	N	N/A
rchange	integer (0,1)	Whether to expand a recurring todo. Expand if 1, do not exand if 0.	N	1
priority	integer (0-10)	The priority of the todo. 0 is the lowest priority, and 10 is the highest.	N	0

Parameter	Type	Purpose	Required	Default
relatedTos	semicolon-separated list of quoted strings	Other todos to which this todo is related.	N	N/A
resources	semicolon-separated list of strings	The resources associated with the todo.	N	N/A
categories	semicolon-sepaated list of strings	The categories of the todo.	N	N/A
attendees	semicolon-separated list of strings	The attendees of the todo.	N	N/A
contacts	semicolon-separated list of strings	Contacts for the todo.	N	N/A
alarmEmails	semicolon-separated list of email addresses	Recipients of alarm notifications for the todo.	N	N/A
alarmStart	iCal DateTime string	The time at which to send an alarm notification of the todo.	N	N/A
attachments	semicolon-separated list of strings	Items to be Attached to notification mail messages.	N	N/A
tzid	quoted string	A timezone. All dates are interpreted with reference to this timezone. If not passed, all dates are interpreted as being in coordinated universal time (UTC or Z format).	N	"GMT"
notify	integer 0,1	When 1, notify attendees of the change. When 0, do not notify attendees.	N	0

Parameter	Type	Purpose	Required	Default
`brief`	integer (0,1)	When 1, print out a brief version of output. When 0, print complete output of event/todo data. Applies only when output type is Javascript	N	`0`

`fmt-out`	string	The output content-type in which data is to be returned. One of the following:
		`text/calendar`
		`text/xml`
		`text/js`
			N	`text/js`

Purpose Use this command to create a todo with the specified attributes and store it in the specified calendar in the database.

The ending date and time (`dtend`) overrides the duration. If you specify both `duration` and `dtend`, the duration is ignored.

Specify the duration in the `iCal` format. For example:

- `P1Y2M3DT1H30M10S` represents a duration of 1 year, 2 months, 3 days, 1 hour, 30 minutes, 10 seconds
- `PT1H30M` represents a duration of 1 hour, 30 minutes
- `P1D` represents a duration 1 day
- `PT15M` represents a duration of 15 minutes

Notice that the `T` in the string separates the date information (year, month, day) from the time information (hour, minute, second).

The command creates and stores recurrences as specified by the recurrence rule, `rid`, `mod`, and `rchange` parameters. See “Storing Recurrences.”

When the `notify` value is 1, sends an IMIP publish message to the e-mail attendees of the todo.

This command cannot modify a linked todo.

version

Purpose To get the WCAP version that the calendar server supports.

Parameters The command takes the following parameters:

Parameter	Type	Purpose	Required	Default
`fmt-out`	string	output content-type of date (text/calendar, text/xml, text/js)	N	`text/js`

Purpose Use this command to get the WCAP version that the server supports. (Note: this is different from the server version as well as the HTTP version.

The commands supports outpute types of iCal, XML, and JavaScript. In iCal and XML, the variable X-NSCP-WCAPVERSION contains the WCAP version number. In Javascript, the variable wcapversion returns the version number.

Error Handling

The JavaScript returned by the server in response to a command contains error numbers in a JavaScript array. Each element of the array represents a WCAP command that was issued. For example, the returned data might include the following expression:

```
var errno=new Array()
errno[0] = "0"
errno[1] = "0"
```

In this case, `errno[0]` could be the error returned on a `storecomponents` call, indicating that the store operation was successful. `errno[1]` could be the error returned on a `fetchcomponents` call, indicating that the fetch operation was successful.

Table 2.7 describes the error codes that can be returned.

Table 2.7 Error codes returned from JavaScript

Name	Value	Meaning
LOGOUT	-1	Logout successful.
OK	0	Command successful.
LOGIN_FAILED	1	Login failed.
LOGIN_OK_DEFAULT_CALENDAR_NOT_FOUND	2	Login successful but default calendar was not found.
DELETECOMPONENTS_FAILED	6	`deletecomponents.wcap` failed.
SETCALPROPS_FAILED	8	`set_calprops.wcap` failed.
FETCHBYUID_FAILED	9	`fetchcomponents_by_id.wcap` failed.
CREATECALENDAR_FAILED	10	`createcalendar.wcap` failed.
DELETECALENDAR_FAILED	11	`deletecalendar.wcap` failed.
ADDLINK_FAILED	12	`addlink.wcap` failed.
FETCHBYDATERANGE_FAILED	13	`fetchcomponents.wcap` failed.
STORECOMPONENTS_FAILED	14	`storecomponents.wcap` failed.
STORETODOS_FAILED	15	`storetodos.wcap` failed.
DELETE_TODOS_BY_UID_FAILED	16	`deletetodos.wcap` failed.

Table 2.7 Error codes returned from JavaScript

Name	Value	Meaning
FETCH_TODOS_BY_UID_FAILED	17	`fetchtodos_by_id.wcap` failed.
FETCHCOMPONENTS_FAILED_BAD_TZID	18	WCAP command failed to find a correct time zone identifier. Applies to `fetchcomponents`, `fetchtodos`, `fetchcomponents_by_id`, `fetchtodos_by_id`.
SEARCH_CALPROPS_FAILED	19	`search_calprops.wcap` failed.
GET_CALPROPS_FAILED	20	`get_calprops.wcap` failed.
DELETECOMPONENTS_BY_RANGE_FAILED	21	`deletecomponents_by_range.wcap` failed.
DELETEEVENTS_BY_RANGE_FAILED	22	`deleteevents_by_range.wcap` failed.
DELETETODOS_BY_RANGE_FAILED	23	`deletetodos_by_range.wcap` failed.
GET_ALL_time zones_FAILED	24	`get_all_time zones.wcap` failed.
CREATECALENDAR_ALREADY_EXISTS_FAILED	25	`createcalendar.wcap` failed because a calendar with the specified name already exists in the database.
SET_USERPREFS_FAILED	26	`set_userprefs.wcap` failed.
CHANGE_PASSWORD_FAILED	27	`change_password.wcap` failed.
ACCESS_DENIED_TO_CALENDAR	28	WCAP command failed because user was denied access to a specified calendar.
CALENDAR_DOES_NOT_EXIST	29	WCAP command failed because a specified calendar does not exist in database.
ILLEGAL_CALID_NAME	30	`createcalendar.wcap` failed because specified calendar ID was invalid.
CANNOT_MODIFY_LINKED_EVENTS	31	`storeevents.wcap` failed because event to modify was a linked event.

Table 2.7 Error codes returned from JavaScript

Name	Value	Meaning
CANNOT_MODIFY_LINKED_TODOS	32	`storetodos.wcap` failed because todo to modify was a linked todo.
CANNOT_SEND_EMAIL	33	WCAP command `storeevents`, `storetodos`, `deleteevents_by_id`, or `deletetodos_by_id` failed because the email notification failed. Usually means that the server is not properly configured to send email.

Multiple Error Codes

In addition to the transaction error number, each transaction may have individual subtransactions that could fail. For example, on a call to `fetchcomponents.wcap`, you can specify multiple layers from which to retrieve events. The fetch can fail for some layers, and not for others.

An additional array, `layer_errno[]`, indicates an error specific to a layer. The index of this array maps to the order in which the layers are passed to the command.

For an example of how this error array works, see the section “Retrieving Calendar Properties.”

Examples of Command Usage

This section gives examples of how to use WCAP commands to perform these common tasks.

- Creating and Storing Events
- Retrieving Components from Calendars
- Retrieving Calendar Properties
- Retrieving User Preferences

The examples show the format of returned JavaScript pages, and demonstrate multiple error codes.

Creating and Storing Events

The following URL calls `storecomponents.wcap` to create a new event and store it in `John1`'s calendar:

```
http://myserver:81/
storecomponents.wcap?id=3423423asdfasf&calid=John1&dtstart=19990101T103
000&dtend=19990101T113000&uid=001&summary=new_year_event
```

The event that is created looks like this in the database (in `iCal`):

```
BEGIN:VEVENT
DTSTART=19990101T183000Z
DTEND:19990101T193000Z
UID:001
SUMMARY:new_year_event
END:VEVENT
```

The following example shows all of the parameters of the `storeevents` command:

```
http://myserver:81/storeevents.wcap?id='bu9p3eb8x5p2nm0q3'&calid=newcal
&uid=all_params&dtstart=19990811T093000Z&dtend=19990811T163000Z
&summary=all%20params-of-storeevents%20are%20included
&location=Mountain-View&icsClass=confidential
&icsUrl= http://www.mycompany.com
&duration=PT7H&desc=read%20the%20summary&status=0&isAllDay=1
&geo=37.39;-
122.05&rrules="count%3D4%3Bfreq%3Ddaily";"count%3D4%3Bfreq%3Dweekly"
&exrules="count%3D1%3Bfreq%3Dweekly%3Binterval%3D2%3Bbyday%3DTU%2CTH"
&rdates=19991015T093000&exdates=19990813T093000&priority=10
&relatedTos=<777>&resources=board;projector&categories=MEETING
&contacts=Jane\,John&&alarmStart=19990811T092500Z
&alarmEmail=lucys@mycompany.com&orgEmail=jane@mycompany.com;
```

Retrieving Components from Calendars

Calendar Server sends an HTML page containing JavaScript code (or another format as specified) as a response to each WCAP request. The first part of every returned page is the standard HTML header, followed by a section with general information such as the session identifier and user name. After these sections, the page contains information or values specific to the command. In JavaScript, these values are assigned to variables.

For example, the following URL sends the `fetchcomponents_by_range.wcap` command:

```
http://myserver.mycompany.com:81/
fetchcomponents_by_range.wcap?id=b5q2o8ve2rk02nv9t6&calid=jdoe
```

The starting and ending date-time specifications, as well as the return data format, are allowed to default, resulting in the retrieval of all calendar properties and events, returned as JavaScript. The returned page for this command contains the following variables:

`id`	Session ID for user (general information)
`userid`	User's login name (general information)
`calid`	Calendar being returned (general information)
`size`	Number of events in current calendar
`todosize`	Number of todos of this calendar
`event`	Array of events of this calendar
`todo`	Array of todos of this calendar
`errno`	Errors in fetching
`layer_errno`	Errors in fetching an individual calendar
`layerid`	Layer name of this calendar
`dtstartrange`	Range of calendar returned. `0` means all events returned
`dtendrange`	Range of calendar returned. `0` means all events returned
`calprops`	Array of calendar properties
`timezoneList`	Array of timezone information

The following shows the complete page that is returned. The first section contains standard HTML headers, and is the same for all commands:

```
HTTP/1.0 200
Date: Tue, 15 Jun 1999 22:43:01 GMT
Content-type: text/html; charset=iso-8859-1
Content-length: 5504
Last-modified: Tue, 15 Jun 1999 22:43:01 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache
<html><head><script>
function color(s) { if (s) document.bgColor=s }
```

Every returned page also contains global information about the calendar and user:

```
var id='bb2ot68wp0t95q'
var userid='jdoe'
var calid='jdoe'
var errno=new Array()
var timezoneList = new Array()
var calprops = new Array()
var layer_errno = new Array()
function iso (p) { if (!p) return null;
   var y = p.substring (0,4)
   var m = p.substring (4,6)
   var d = p.substring (6,8)
   var h = p.substring (9,11)
   var i = p.substring (11,13)
   var s = p.substring (13,15)
   return new Date (y, m - 1, d, h, i, s, 0) }
```

The page creates functions that retrieve the information from the arrays in which it is returned, and creates those arrays:

```
function E (a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17,
    a18,a19,a20,a21,a22,a23,a24,a25,a26,a27,a28,a29,a30,a31,a32,a33){
   this.uid=a1 this.rid=a2 this.calid=a3 this.dtstart=iso(a4)
   this.dtend=iso(a5) this.status=a6 this.isAllDay=a7 this.summary=a8
   this.created=iso(a9) this.lastMod=iso(a10) this.seq=a11
   this.geo=new Array(a12,a13) this.desc=a14 this.icsUrl=a15
   this.icsClass=a16 this.location=a17 this.alarmStart=a18
   this.alarmEmails=a19 this.rrules=a20 this.rdates=a21
this.exrules=a22
   this.exdates=a23 this.tzid=a24 this.priority=a25 this.relatedTos=a26
   this.resources=a27 this.categories=a28 this.attachments=a29
   this.contacts=a30 this.attendees=a31 this.orgEmail=a32
   this.linkCalid=a33 }
function T (a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17,
    a18, a19,a20,a21,a22,a23,a24,a25,a26,a27,a28,a29,a30,a31,a32,a33,
    a34,a35) {
   this.uid=a1 this.rid=a2 this.calid=a3 this.dtstart=iso(a4)
   this.due=iso(a5) this.status=a6 this.isAllDay=a7 this.summary=a8
   this.created=iso(a9) this.lastMod=iso(a10) this.seq=a11
   this.geo=new Array(a12,a13) this.desc=a14 this.icsUrl=a15
   this.icsClass=a16 this.location=a17 this.alarmStart=a18
   this.alarmEmails=a19 this.rrules=a20 this.rdates=a21
this.exrules=a22
   this.exdates=a23 this.completed=iso(a24) this.percent=a25
   this.tzid=a26 this.priority=a27 this.relatedTos=a28
   this.resources=a29 this.categories=a30 this.attachments=a31
   this.contacts=a32 this.attendees=a33 this.orgEmail=a34
     this.linkCalid=a35 }
function CP(a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15)
  {this.calid=a1 this.name=a2 this.parent=a3 this.tzid=a4 this.read=a5
   this.write=a6 this.charset=a7 this.lang=a8 this.master=a9
```

```
   this.desc=a10 this.lastMod=iso(a11) this.created=iso(a12)
   this.primaryOwner=a13 this.owners=a14 this.categories=a15 }

var event=new Array()
var event=new Array()
var todo=new Array()
```

The information section of the page contains the actual retrieved information from the calendars in the defined array types.

```
var status_types=new Array('confirmed', 'tentative', 'cancelled')
calprops[0] = new CP('John Doe', 'jdoe', '', '', '0', '0', '', '', '',
   'Work Calendar for John Doe', '19990615T164146Z', '19990615T164146Z',
   'jdoe', new Array('susan'), new Array(), new Array('business'))
event[0]=new E ('52452531427158444', '0', 'jdoe', '19981223T233000',
   '19981224T023000', 1,0,'jdoe event 3', '19990615T222725',
   '19990615T222725', '1', '37.463581','-121.897606',
   'This is the description for event with UID = 52452531427158444',
   'http://myserver.mycompany.com/'+'susan?uid=52452531427158444',
   '', 'Green Conference Room', 0, new Array(), new Array(),
   new Array(), new Array(), new Array(),'GMT','1', new Array(),
   new Array(), new Array(), new Array(), new Array(), new Array(), '')
event[1]=new E ('51452531524178674', '0', 'jdoe', '19981223T130000',
   '19981223T123000', 1,0,'jdoe event 2', '19990615T222725',
   '19990615T222725', '1', '37.463581','-121.897606',
   'This is the description for event with UID = 51452531524178674',
   'http://myserver.mycompany.com/'+'susan?uid=51452531524178674',
   '', 'Green Conference Room', 0, new Array(), new Array(),
   new Array(), new Array(), new Array(),'GMT','1',
   new Array(), new Array(), new Array(), new Array(), new Array(),
   new Array(), '')
event[2]=new E ('tm-001', '19990519T010000Z', 'jdoe',
'19990518T170000',
   '19990518T190000', 1,0,'Calendar Staff', '19990615T222725',
   '19990615T222725', '1', '37.463581','-121.897606',
   'This is the description for event with UID = tm-001',
   'http://myserver.mycompany.com/'+'susan?uid=tm-001', '',
   'Green Conference Room', 0, new Array(), new Array(), new Array(),
   new Array(), new Array(),'GMT','1', new Array(), new Array(),
   new Array(), new Array(), new Array(), new Array(), '')
event[3]=new E ('tm-001', '19990526T010000Z', 'jdoe',
'19990525T170000',
   '19990525T190000', 1,0,'Calendar Staff', '19990615T222725',
   '19990615T222725', '1', '37.463581','-121.897606',
   'This is the description for event with UID = tm-001',
   'http://myserver.mycompany.com/'+'susan?uid=tm-001', '',
   'Green Conference Room', 0, new Array(), new Array(), new Array(),
   new Array(), new Array(),'GMT','1', new Array(), new Array(),
   new Array(), new Array(), new Array(), new Array(), '')
event[4]=new E ('tm-001', '19990602T010000Z', 'jdoe',
```

```
'19990601T170000',
   '19990601T190000', 1,0,'Calendar Staff', '19990615T222725',
   '19990615T222725', '1', '37.463581','-121.897606',
   'This is the description for event with UID = tm-001',
   'http://myserver.mycompany.com/'+'susan?uid=tm-001', '',
   'Green Conference Room', 0, new Array(), new Array(), new Array(),
   new Array(), new Array(),'GMT','1', new Array(), new Array(),
   new Array(), new Array(), new Array(), new Array(), '')

todo[0]=new T ('abctodo', '0', 'jdoe', '19990304T143000',
   '19990620T222725', 1,0,'Fix the bugs', '19990615T222725',
   '19990615T222725', '1', '37.463581','-121.897606',
   'This is the description for event with UID = abctodo',
   'http://myserver.mycompany.com/'+'susan?uid=TODO1', '',
   'Green Conference Room', 0, new Array(), new Array(), new Array(),
   new Array(), new Array(),'19700101T000000', '-1', 'GMT','1',
   new Array(), new Array(), new Array(), new Array(), new Array(),
   new Array(), '')
```

Note For Javascript output, all events that last longer than 24 hours or are all-day events are printed in an `eventD` array. For example:

```
eventD[0] = new E ('tm-001', '19990602T010000Z', 'jdoe',
'19990601T170000', '19990601T170000', 1,1,'Calendar Staff',
/* "isAllDay" flag is set to 1 (2nd '1' this line) */
'19990615T222725','19990615T222725', '1', '37.463581','-121.897606',
'This is the description for event with UID = tm-001',
'http://cal.mycompany.com/'+'jdoe?uid=tm-001', '',
'Green Conference Room', 0, new Array(), new Array(), new Array(),
new Array(), new Array(),'GMT','1', new Array(), new Array(),
new Array(), new Array(), new Array(), new Array(), '', '')
```

Finally, the page contains summary and error information:

```
layer_errno[0]=0
var size=5
var todosize=1
var layerid="jdoe"
var dtstartrange="0"
var dtendrange="0"
errno[0]=0
parent.ceCB(window.name)
</script></head></html>
```

Specifying Brief Output

The following commands that return event and todo information allow you to specify the `brief` parameter, which limits the output to about half the size:

- deletecomponents_by_range
- deleteevents_by_id
- deleteevents_by_range
- deletetodos_by_id
- deletetodos_by_range
- fetchcomponents_by_range
- fetchevents_by_id
- fetchtodos_by_id
- storeevents
- storetodos

The brief output includes information on the following parameters:

Event parameters	Todo parameters
uid	uid
rid	rid
calid	calid
dtstart	dtstart
dtend	due
isAllDay	isAllDay
summary	summary
created	created
lastMod	lastMod
desc	desc
location	location
tzid	completed
linkCalid	percent
	tzid
	linkCalid

Retrieving Calendar Properties

In this example, the following command tries to retrieve the calendar properties for the calendars `jdoe`, `susan`, `pub`, `huey`, and `hasdf` in that order.

```
http://myserver/get_calprops.wcap?id=2mu95r5so0hq68ts6q3
   &calid=jdoe;susan;pub;huey;hasdf
```

In this case, the user has read access the calendars `jdoe`, `pub`, and `huey`, but not to the calendar `susan`. The calendar `hasdf` does not exist in the database.

- The call to `get_calprops.wcap` returns data for the calendars `jdoe`, `pub`, and `huey`, and the `layer_errno[]` value for their indexes is 0.
- No data is returned for the calendar `susan`, and the `layer_errno[1]` value is set to `1`, indicating that the user does not have access to that calendar.
- No data is returned for the calendar `hasdf`, and the `layer_errno[5]` value is set to `2`, indicating that the calendar does not exist.
- If the fetch fails for any calendar, its error number is set to the `GET_CALPROPS_FAILED` value.

The command returns the following in the information section of the returned page:

```
calprops[0] = new CP('jdoe', 'John Doe', '', '', '0', '0', '', '', '',
   'Work Calendar for John Doe', '19990615T222725Z',
'19990615T222725Z',
   'jdoe', new Array('susan'), new Array(), new Array('business'))
layer_errno[0]=0  calprops[1] = new CP('susan')
layer_errno[1]=1
calprops[2] = new CP('pub', 'Public Calendar','','','1', '1','','','',
   'Public Calendar, Anyone can read and write', '19990615T222725Z',
   '19990615T222725Z', 'susan', new Array(), new Array(),
   new Array('group','business'))
layer_errno[2]=0
calprops[3] = new CP('huey', 'Huey Calendar Hosting Server',
   '', '', '0', '0', '', '', '', 'Huey Project Calendar',
   '19990615T222725Z', '19990615T222725Z', 'susan',
   new Array('smith','jones','ewok','jdoe'), new Array(),
   new Array('group','business'))
layer_errno[3]=0
calprops[4] = new CP('hasdf')
layer_errno[4]=2
errno[0]=0
```

```
parent.ceCB(window.name)
```

Retrieving User Preferences

The following URL retrieves user preferences for the current user:

```
http://myserver/get_userprefs.wcap?id=b5q2o8ve2rk02nv9t6
```

The following shows the information portion of the page returned from this command:

```
<html><head><script>
function A (name, readonly)
   {this.name=name
    this.readonly=readonly
    this.vals=new Array (A.arguments.length - 2)
    for (var i = 2; i < A.arguments.length; i++)
    this.vals[i - 2]=A.arguments[i] }
var usrattrval=new Array()
usrattrval[0]=new A('cn',true,'John Doe')
usrattrval[1]=new A('givenName',true,'John')
usrattrval[2]=new A('mail',true,'jdoe@jdoe-2.mycompany.com')
usrattrval[3]=new A('preferredlanguage',false)
usrattrval[4]=new A('sn',true,'Doe')
usrattrval[5]=new A('cebgcolor',false,'black')
usrattrval[6]=new A('ceFgcolor',false,'green')
usrattrval[7]=new A('ceBFgcolor',false,'black')
function M(name, flags, msgs, size)
   {this.name=name
    this.msgs=msgs
    this.flags=flags
    this.size=size}
var flags=new Array ('\\noinferiors', '\\hasnochildren',
                     '\\haschildren', '\\noselect')
var flag=new Array(flags.length)
flag['\\noinferiors']=1
flag['\\hasnochildren']=2
flag['\\haschildren']=4
flag['\\noselect']=8
var allowchangepassword="no"
var allowcreatecalendars="yes"
var allowdeletecalendars="yes"
var allowexpungecalendar="yes"
var errno=new Array()
errno[0]=0
var errstr=''
```

```
</script></head>
```

3

Calendar Server API (CSAPI)

This chapter describes Calendar Server API (CSAPI), a high performance programmatic interface that enables you to modify or enhance the feature set of iPlanet Calendar Server. CSAPI allows you to create very fast runtime shared objects that outperform both system executables and scripts in any language, with respect to speed, memory footprint, and load. All of these factors contribute to scalability issues in high-end systems.

This chapter has the following sections:

- CSAPI Architecture
- API Reference

CSAPI Architecture

The CSAPI is a shared-object runtime interface to Calendar Server functions. You can use plug-in CSAPI modules to manipulate server data as requests come in and as responses are made. This architecture allows the server to act as a simple gateway to data it knows nothing about. It also allows for dynamic logging and statistics tracking, external authentication schemes, user attribute manipulation, and a variety of other functions.

Figure 3.1 shows the architecture of CSAPI modules within Calendar Server. Depending on which functional group or groups an CSAPI module supports, it can interact with one or more areas of Calendar Server functionality.

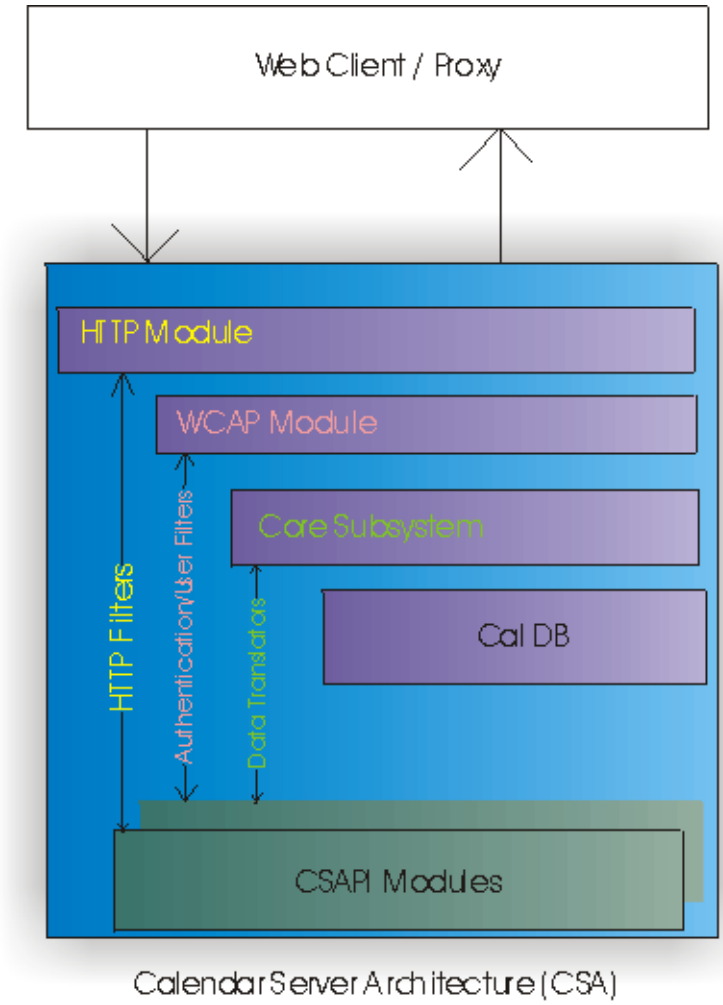


Figure 3.1 CSAPI architecture

A module is a shared object (`.so` file) on Unix, or dynamic linked library (`.dll` file) on Windows NT. Each module that you provide implements one or more of the CSAPI interfaces (or pure virtual base classes) defined in this document. Each interface addresses a functional area of Calendar Server. The implementation contained in a module can either augment or override the native Calendar Server functionality in its area.

CSAPI is a C and C++ interface for Unix and Windows NT systems. It uses Netscape Portable Runtime (NSPR), a part of the mozilla.org source code that is a platform independent API to operating system services, and XPCOM for Interface Dispatch.

For documentation on NSPR see the mozilla technical documentation site:

`http://www.mozilla.org/docs/refList/refNSPR`

For documentation on XPCOM, see:

`http://www.mozilla.org/docs/tplist/catFlow/modunote.htm`

You must use NSPR for platform-independent C data types and runtime functions in implementations that need to run on different platforms. Calendar Server uses the XPCOM C++ API (`QueryInterface`) to discover the exact interfaces a specific module implements.

A set of server-side APIs allows CSAPI modules to register for specific events, get the server's version information, and use the server's fast memory allocation mechanism.

CSAPI module plug-ins must be multithread safe, as many thousands of threads can access a module at any time. For those plug-ins that cannot be thread-aware, use simple monitors at the function-call level in the plug-in itself. For more information on NSPR threads, refer to the NSPR reference manual.

Installed Files

The installation contains plug-in shells that you can use as templates for each of the CSAPI interfaces along with all supporting libraries, headers, and build tools for compiling on all systems supported by Calendar Server. It also contains C code sample implementations of CSAPI modules.

Figure 3.2 shows the CSAPI directory structure under the Calendar Server installation directory.

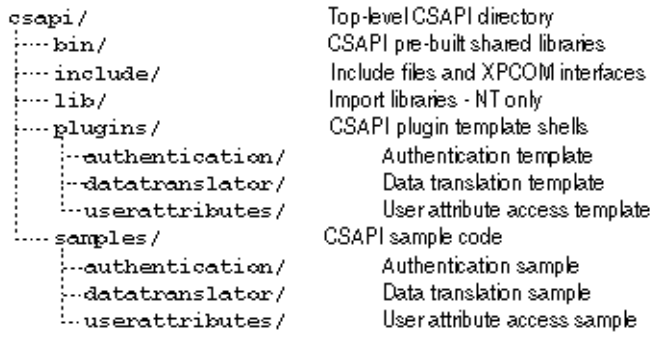


Figure 3.2 CSAPI directory structure

Loading CSAPI Modules

Calendar Server loads CSAPI modules from the *server_root*/plug-ins directory at startup and unloads them at server shutdown. All plug-in modules must reside in this directory and have filenames that are prefaced with cs_.

The server checks the file config/server.conf for the list of modules to be dynamically loaded at server startup. If the value in this file is an asterisk (*), the server loads all shared objects in the *server_root*/plug-ins directory whose names begin with the prefix cs_.

Calendar Server uses the NSPR function PR_LoadLibrary() to load the shared object at startup, the function PR_UnloadLibrary() to unload the shared image at shutdown. Once a shared object is loaded into memory, Calendar Server uses the function PR_FindSymbol() to find entry points to known API implementations.

CSAPI Module Structure

All CSAPI shared objects support one and only one exported symbol, NSGetFactory, as required by the XPCOM specification. From this entry point, Calendar Server calls the XPCOM method QueryInterface to find objects that implement the supported interfaces, to find out exactly which CSAPI modules the module supports. When it finds an interface, the server registers it and continues to look for the remaining interfaces.

A module can implement one or more of the following interfaces:

Table 3.1 CSAPI Module Interface

CSAPI Module Interface	Description
`csIplug-in`	Provides version control and descriptive information about the module.
`csIAuthentication`	Augments or overrides the login authentication mechanism.
`csIUserAttributes`	Overrides the mechanism for storing and retrieving user attributes.
`csIDataTranslator`	Overrides the format translation of incoming and outgoing data.

The interfaces are described in detail in API Reference.

All interfaces have the following initialization method that you must implement:

```
Init (nsISupports * aServer);
```

The server invokes this method immediately after it registers the interface in a newly loaded module. In the module, you can bind the parameter that the server returns, `aServer`, and use it to refer to the server instance. Use this object's `QueryInterface` method to find the following server interfaces:

Table 3.2 Server Interface

Server Interface	Description
`csICalendarServer`	Provides general server information, including version number.
`csIMalloc`	Allows access to server's memory allocation mechanism.

Server Query Example

The following example checks the version of Calendar Server. It demonstrates how to do the following:

- Bind the returned reference from the Init method.
- Query the server for an interface.
- Call a server method in that interface.

- Release the server reference.

```
NS_IMETHODIMP csDataTranslator :: Init(nsISupports * aServer)
{
nsresult res = NS_COMFALSE ;
PRUint32 min, maj;
csICalendarServer * cs;
/* QueryInterface for CalendarServer. If call succeeds, server
increments reference count */
if (aServer)
  res = aServer->QueryInterface(kICalendarServerIID,(void**)&cs);
/* If succeeded in getting reference to server, check version */
if (NS_SUCCEEDED(res)) {
  cs->GetVersion(maj,min);
  if (min > 0 && maj >= 1)
    res = NS_OK;
  else
    res = NS_COMFALSE;
/* Release this reference to the server instance */
  cs->Release();
  }
return res;
}
```

Building the CSAPI Samples

The distribution includes sample code for each of the CSAPI interfaces in the `csapi/samples` directory. You can use these files as templates in building your own CSAPI modules. The following sample modules are provided:

Table 3.3 CSAPI Modlue Sample

CSAPI Module Sample	Description
`Authentication`	This sample overrides the default login authentication mechanism, using local authentication to validate users. The sample works on Solaris and on Window NT: • On Solaris, it uses the `pam` library to authenticate against the `local /etc/passwd` file or NIS. • On Windows NT, it uses the WIN32 API `LogonUser`, which authenticates against Microsoft clients. In order for this sample to work properly on NT, the administrator must enable the privilege for users to log on via batch jobs. You can do this from within the UserManager Administrative Tool, under the Policies/ User Rights section.
`DataTranslator`	This sample overrides the default format translation of incoming and outgoing data. It shows how to convert `icalendar` data into Micrrosoft Outlook CSV format. CSV format is a simple line-oriented file, where each entry has its own line and properties are separated by commas.
`UserAttributes`	This sample overrides the default mechanism for storing and retrieving user attributes. It shows how to use the Berkeley database to store local user preferences. This code works on all supported platforms. The database is a simple table-driven key/value pair. The key for storing user preferences is a string stored as `$user.$pref`. The key must be unique.

The `samples` directory contains a makefile to create the plug-ins that Calendar Server uses. To build the samples, execute the appropriate make command for your operating system:

- In a Unix command shell, type the following:

  ```
  cd server_root/csapi/samples
  make -f Makefile
  ```

- In a DOS command shell, type the following:

  ```
  cd server_root/csapi/samples
  nmake -f makefile.win
  ```

API Reference

This section lists the interfaces that a CSAPI module can support, and the methods that can be implemented for each interface, as well as the interfaces on the server side with which modules can communicate.

This section describes the following server and CSAPI interfaces:

- Server Interface: csICalendarServer
- Server Interface: csIMalloc
- CSAPI plug-in Interface: csIplug-in
- CSAPI Authentication Interface: csIAuthentication
- CSAPI User Attribute Access Interface: csIUserAttributes
- CSAPI Data Format Translation Interface: csIDataTranslator

Server Interface: csICalendarServer

Provides server version information to a module.

Methods The `csICalendarServer` interface has the following method:

`GetVersion`	Gets the major and minor version of the server. This value must be greater than or equal to 1.0.

Description plug-in modules can query the `csICalendarServer` interface to get version information about the running instance of Calendar Server.

A reference to this object is returned by a module's initialization method, which the server calls when it loads the module. The reference is valid for as long as the plug-in module is loaded.

GetVersion

Purpose Provide plug-in module with server version information.

Syntax

```
PRUint32 GetVersion (PRUint32& aMajorValue,
                     PRUint32& aMinorValue)
```

Parameters The method has the following parameters:

aMajorValue	On return, contains the major version number.
aMinorValue	On return, contains the minor version number.

Returns NS_OK on success, non-zero error code on failure.

Description Use this method to identify the server's major and minor version number. The number is always greater than or equal to 1.0.

Server Interface: csIMalloc

Allocates and frees memory.

Methods The csIMalloc interface has the following methods:

Calloc	Allocates and initializes memory for a number of objects.
Free	Frees memory that is no longer in use.
FreeIf	Frees memory, allowing a NULL pointer.
Malloc	Allocates an amount of memory.
Realloc	Reallocates previously allocated memory.

Description plug-in modules can use this object to take advantage of the server's efficient memory allocation technique.

Calloc

Purpose Allocate and initialize memory for a number of objects.

Syntax
```
void* Calloc (PRUint32 aSize,
              PPRUint32 aNum)
```

Parameters The method has the following parameters:

aSize	The size in bytes of each object.
nNum	The number of objects.

Returns A pointer to the allocated memory on success, or NULL on failure.

Description This method allocates enough memory for the specified number of objects of the specified size, and initializes the memory to zero.

Free

Purpose Free memory previously allocated by the `Malloc` method.

Syntax
```
PRUint32 Free (void * aPtr)
```

Parameters The method has the following parameter:

`aPtr`	A pointer to the memory to be freed.

Returns `NS_OK` on success, non-zero error code on failure.

Description Use this method in the same way as its C/C++ counterpart.

FreeIf

Purpose Free memory previously allocated by the `Malloc` method, allowing a `NULL` pointer.

Syntax
```
PRUint32 FreeIf (void * aPtr)
```

Parameters The method has the following parameter:

`aPtr`	A pointer to the memory to be freed or `NULL`.

Returns `NS_OK` on success, non-zero error code on failure.

Description Frees the memory at the specified location, if `aPtr` is not `NULL`.

Malloc

Purpose Allocate a specified amount of memory.

Syntax
```
void* Malloc (PRUint32 nBytes)
```

Parameters The method has the following parameter:

`nBytes`	The size in bytes of the memory to be allocated.

Returns A pointer to the allocated memory on success, or `NULL` on failure.

Description Use this method in the same way as its C/C++ counterpart.

Realloc

Purpose Reallocate memory that was previously allocated.

Syntax

```
void* Realloc (void * aPtr,
               PRUint32 nBytes)
```

Parameters The method has the following parameter:

`aPtr`	A pointer to previously allocated memory.
`nBytes`	The size in bytes of the memory to be allocated.

Returns A pointer to the allocated memory on success, or `NULL` on failure.

Description Use this method in the same way as its C/C++ counterpart.

CSAPI plug-in Interface: csIplug-in

Implement the methods in this interface to provide server with information about the plug-in module on startup.

Methods The `csIplug-in` interface has the following methods:

`GetDescription`	Gets a textual description of what the plug-in does.
`GetVendorName`	Gets a textual description of the vendor supplying this plug-in.
`GetVersion`	Gets the major and minor version of the plug-in. This value must be greater than or equal to 1.0.
`Init`	Confirms that the interface was found and registered.

Description This interface is not required, but it is highly recommended that you implement it in each module to provide version information to the server when it loads that module. You provide methods that return descriptive information to the server.

GetDescription

Purpose Retrieve a text description of the module.

Syntax
```
PRUint32 GetDescription (NSString& aDescription)
```

Parameters The method has the following parameter:

`aDescription`	On return, contains the text description of the module.

Returns `NS_OK` on success, non-zero error code on failure.

Description Use this method to provide a text description of the module.

GetVendorName

Purpose Retrieve a text description of the vendor supplying the module.

Syntax
```
PRInt32 GetDescription (NSString& aDescription)
```

Parameters The method has the following parameter:

`aDescription`	On return, contains the text description of the vendor.

Returns `NS_OK` on success, non-zero error code on failure.

Description Use this method to identify the module's supplier.

GetVersion

Purpose Provide server with version information on startup.

Syntax
```
PRUint32 getversion (PRUint32& aMajorValue,
                     PRUint32& aMinorValue)
```

Parameters The method has the following parameters:

`aMajorValue`	On return, contains the major version number.
`aMinorValue`	On return, contains the minor version number.

Returns NS_OK on success, non-zero error code on failure.

Description Use this method to identify the module's major and minor version number. The number must be greater than or equal to 1.0.

Init

Purpose Confirm that the interface has been registered and obtains a reference to the server.

Syntax

```
PRUint32 Init (nsISupports * aServer)
```

Parameters The method has the following parameter:

`aServer` On return, this location contains a reference to the server with which the module is registered.

Returns NS_OK on success, non-zero error code on failure.

Description The server calls this method, after finding and registering the interface on module load, to confirm that the operation was successful. You can use the pointer returned in `aServer` to make calls out to the server.

CSAPI Authentication Interface: csIAuthentication

Implement the methods in this interface to augment or override the default authentication procedures when a user logs in to Calendar Server.

Methods The `csIAuthentication` interface has the following methods:

`Init` Confirms that the interface was found and registered.

`Logon` Augments or overrides the server's native authentication mechanism.

Description You define a Logon method that implements the authentication technique of your choice, then indicates whether the system should continue with the default authentication procedure.

Init

Purpose Confirm that the interface has been registered, and provides a reference to the server.

Syntax `PRUint32 Init (nsISupports * aServer)`

Parameters The method has the following parameter:

`aServer` On return, this location contains a reference to the server with which the module is registered.

Returns `NS_OK` on success, non-zero error code on failure.

Description The server calls this method after finding and registering the interface on module load, to confirm that the operation was successful. You can use the pointer returned in `aServer` to make calls out to the server.

Logon

Purpose Augment or override the authentication procedure for plain text login.

Syntax

```
PRUint32 Login (char * aUser,
                char * aPassword,
                PRInt32 * aReturnCode)
```

Parameters The method has the following parameters:

`aUser`	The user's login name.
`aPassword`	The plain text password.

`aReturnCode`	On return, contains a constant that determines whether the server should continue with the default authentication procedure.
	One of the following constants:
	`NS_CONTINUE_DEFAULT_PROCESSING`
	`NS_OVERRIDE_DEFAULT_PROCESSING`

Returns `NS_AUTHENTICATION_LOGON_SUCCESS` on success, `NS_AUTHENTICATION_LOGON_FAILURE` on failure.

Description Use this method to specify your own authentication procedure on login to Calendar Server. You can augment the native authentication mechanism, performing your own processing first, then continuing with the default process, or you can completely replace the native authentication mechanism.

Use the return code (`aReturnCode`) to tell the server whether to continue with the default authentication process after executing this method. The return code must be one of the following constant values:

`NS_CONTINUE_DEFAULT_PROCESSING`	Indicates that the server is to continue default authentication processing.
`NS_OVERRIDE_DEFAULT_PROCESSING`	Indicates that this method overrides the server's native authentication mechanism.

CSAPI User Attribute Access Interface: csIUserAttributes

Implement the methods in this interface to override the procedure for setting or retrieving user attributes.

Methods The `csIUserAttributes` interface has the following method listed here with their purposes:

`FreeAttribute`	Free the memory used to store a retrieved attribute.
`GetAttribute`	Retrieve an attribute value for a user.
`Init`	Confirm that the interface was found and registered.
`SetAttribute`	Set an attribute value for a user.

Description The User Attributes interface allows a CSAPI module to maintain or manipulate all requests coming in through the `cshttpd` daemon for setting and retrieving user attribute values. You provide methods that retrieve and set attributes using the technique of your choice.

FreeAttribute

Purpose Free the memory associated with your local attribute storage.

Syntax
```
PRUint32 FreeAttribute (char * aValue)
```

Parameters The method has the following parameter:

`aValue` The location you allocated to contain the retrieved attribute value.

Returns `NS_OK` on success, non-zero error code on failure.

Description When you retrieve the value of an attribute using the `GetAttribute` method, the value is stored at a location that you have allocated, using the memory management technique of your choice. Use the `FreeAttribute` method to free that memory when it is no longer needed, using the same memory management technique.

GetAttribute

Purpose Retrieve an attribute value for a user.

Syntax

```
PRUint32 GetAttribute (char * aUser,
                       char * aKey,
                        char ** aValue)
```

Parameters The method has the following parameters:

`aUser`	The name of the user.
`aKey`	The attribute key.
`aValue`	On return, this location contains a pointer to the retrieved attribute value.

Returns `NS_OK` on success, non-zero error code on failure.

Description Retrieves the value of the specified attribute for the specified user, and stores it at the location pointed to by `aValue`. You are responsible for allocating storage space for the returned attribute, and for freeing it (using the `FreeAttribute` method) when it is no longer needed.

Init

Purpose Confirm that the interface has been registered and obtain a reference to the server.

Syntax

```
PRUint32 Init (nsISupports * aServer)
```

Parameters The method has the following parameter:

`aServer`	On return, this location contains a reference to the server with which the module is registered.

Returns `NS_OK` on success, non-zero error code on failure.

Description The server calls this method, after finding and registering the interface on module load, to confirm that the operation was successful. You can use the pointer returned in `aServer` to make calls out to the server.

SetAttribute

Purpose Set an attribute value for a user.

Syntax
```
PRUint32 GetAttribute (char * aUser,
                       char * aKey,
                       char * aValue)
```

Parameters The method has the following parameters:

`aUser`	The name of the user.
`aKey`	The attribute key.
`aValue`	The value.

Returns `NS_OK` on success, non-zero error code on failure.

Description Sets the specified attribute for the specified user to the specified value.

CSAPI Data Format Translation Interface: csIDataTranslator

Implement the methods in this interface to translate the format of data flowing to or from the database.

Methods The `csIDataTranslator` interface has the following methods:

`GetSupportedContentType`	Informs the server about the content types that this module can translate between.
`Init`	Confirms that the interface was found and registered.
`Translate`	Implements the translation from one format to another.

Description This interface allows you to manipulate or change the HTML Body content of calendar data flowing to or from the database, or between various data translators. The data translator manipulates the "Format-Out" component of a WCAP response.

Calendar Server supports the following mime-types for translating calendar data:

Mimetype	Description
text/calendar	ICAL
text/xml	ICAL in XML
text/sxml	ICAL in SXML
text/vcalendar	VCalendar
text/js	Native javascript

A CSAPI Data Translation module registers with the server for a specific mime-type using the `GetSupportedContentType` method. The translator can request that the incoming data be provided in any of the supported mime-types.

When incoming data is in the mime-type that the module takes as input, the server passes the data to the module's `Translate` method. The translator converts the data to its supported mime-type and passes it back to the server, which proceeds to update the database.

GetSupportedContentType

Purpose Inform the server of the input and output content types of this translation module.

Syntax

```
PRUint32 GetSupportedContentType (
            char ** aSupportedInContentTypes,
            char ** aSupportedOutContentType,
            char ** aPreferredInContentType)
```

Parameters The method has the following parameters:

`aInContentTypes`	A list of content-types that the server can send as input to translator. An array of `NULL`-terminated strings.
`aOutContentType`	On return, points to the content-type this translator generates. You must allocate memory for this string.
`aPreferredInContentType`	On return, points to the content-type this translator uses as input.You must allocate memory for this string.

Returns Zero on success, non-zero on failure.

Description Use this method to inform the server what content-type the plug-in module takes as input, and what content-type it generates. You are responsible for allocating and freeing memory for all out parameters.

Init

Purpose Confirm that the interface has been registered and obtain a reference to the server.

Syntax

```
PRUint32 Init (nsISupports * aServer)
```

Parameters The method has the following parameter:

`aServer`	On return, this location contains a reference to the server with which the module is registered.

Returns `NS_OK` on success, non-zero error code on failure.

Description The server calls this method after finding and registering the interface on module load, to confirm that the operation was successful. You can use the pointer returned in `aServer` to make calls out to the server.

Translate

Purpose Implement the translation from one content type to another.

Syntax

```
PRUint32 Translate (char * aInContentType,
                    char * aOutContentType,
                    char ** aInBuffer,
                    char ** aOutBuffer,
                    PRInt32 * aInSize,
                    PRInt32 * aOutSize)
```

Parameters The method has the following parameters:

`aInContentType`	The content type of the data to be translated.
`aOutContentType`	The content type that is generated by translator.
`aInBuffer`	A pointer to the location of the incoming data.
`aOutBuffer`	On return, a pointer to the location of the translated data. You must allocate memory for this buffer.
aInSize	A pointer to the size of the input buffer in bytes.
aOutSize	On return, a pointer to the size of the output buffer in bytes.

Returns Zero on success, non-zero on failure.

Description This method retrieves content of the specified input type from the specified buffer, translates the content from its original format to the output format, stores the translated content in the specified output buffer, and stores the size of the output buffer at the specified location. You are responsible for allocating and freeing memory for all `out` parameters.

Index

F

G

H

I

L

M

O

P

R

S

T

U

W