# Programmer's Reference

*iPlanet Calendar Server*

**Release 5.0**

Programmer's
Reference
February 2001

# Contents

# About This Reference

This document describes the architecture of iPlanet Calendar Server 5.0, and gives detailed instructions on the use of the following two APIs and one protocol that you may use to customize your server installation:

- Calendar Server Application Program Interface (CSAPI), to modify server functionality.
- Event Notification Server (ENS) Application Program Interface, to modify publish and subscribe funtionality.
- Proxy Authentication SDK (authSDK), an external plugin to use a portal authenticaion service.
- Web Calendar Access Protocol (WCAP), to access calendar services.

In addition, this book includes a chapter on the Single Sign-on (SSO) alternate authentication scheme for single-domain instances of iPlanet Calendar Server.

Topics covered in this chapter include:

- Who Should Read This Book
- What You Need to Know
- How This Book is Organized
- Document Conventions
- Where to Find Related Information
- Where to Find This Book Online

## Who Should Read This Book

This guide is for programmers who want to customize applications in order to implement iPlanet Calendar Server 5.0.

# What You Need to Know

This book assumes that you are a programmer with a knowledge of C/C++ and that you have a general understanding of the following:

- The Internet and the World Wide Web
- Calendaring concepts
- LDAP
- RFC2445, RFC2446, RFC2447

  These RFCs describe in detail the format and definition for times, strings, parameters, etc. used in WCAP commands, unless otherwise specified.

  The RFC's may be found at the IETF web site:

  - http://www.ietf.org/rfc/rfc2445.txt
  - http://www.ietf.org/rfc/rfc2446.txt
  - http://www.ietf.org/rfc/rfc2447.txt

# How This Book is Organized

This book documents three APIs, an SDK, and a protocol inside iPlanet Calendar Server, as well as containing an overall architecture discussion of the product. For each interface there is an overview chapter, followed by a reference chapter, where available.

A list of the chapters follows:

- About This Reference (this chapter)
- Chapter 1, "Architecture Overview"

  This chapter contains an overview of the whole iPlanet Calendar Server, and a brief discussion of each API or protocol included in this Reference.

- Chapter 2, "Calendar Server API (CSAPI) Overview"

  This API allows programmers to customize server functionality in five areas: access control, authentication, calendar lookup, data format translation, and user attribute access.

- Chapter 3, "CSAPI Reference"

This chapter describes the CSAPI interfaces and their methods.There are two types of interfaces: client and server.

- Chapter 4, “Event Notification Service (ENS) Overview”

  ENS is a publish and subscribe service, which acts as a dispatcher for events and todos. The API allows you to customize or replace the default mechanism.

- Chapter 5, “Event Notification Service API Reference”

  This chapter describes the functions and their parameters that make up the ENS API.

- Chapter 6, “Proxy Authentication SDK Overview”

  This chapter discusses one of the three authentication schemes shipped with the server. This API allows you to integrate your portal service with iPlanet Calendar Server.

- Chapter 7, “Proxy Authentication SDK Reference”

  This chapter describes the five functions that make up the SDK.

- Chapter 8, “Single Sign-on Authentication”

  This chapter describes one of the three authentication mechanisms offered with the product. Single Sign-on works on single-domain installations and allows users to sign on once and use all applications in the trusted circle.

- Chapter 9, “Web Calendar Access Protocol (WCAP) Overview”

  This chapter give an introduction to the WCAP protocol. WCAP is a command based system for transmitting calendar data.

- Chapter 10, “WCAP Commands”

  Details of individual commands, and some common topics, comprise this chapter.

# Document Conventions

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

*`Italicized font`*— This is used to represent text that you enter using information that is unique to your installation (for example, variables). It is used for server paths and names and account IDs.

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

# Where to Find Related Information

In addition to this guide, these other documents are available:

- iPlanet Calendar Server Administration Guide
- iPlanet Calendar Server Installation Guide

# Where to Find This Book Online

You can find the iPlanet Calendar Server Programmer's Reference online in PDF and HTML formats. To find this book, use this URL:

`http://docs.iplanet.com/docs/manuals/calendar/ics50/pr/contents.htm`

Use the following URL to see all the Calendar Server documentation:

`http://docs.iplanet.com/docs/manuals/calendar.html`

Chapter 1

# Architecture Overview

iPlanet Calendar Server 5.0 is a powerful and flexible cross-platform solution using open Internet standards that lets service providers of all sizes host personal and group scheduling calendars for their customers.

Topics covered in this chapter are:

- What is iPlanet Calendar Server?
- What's New in Version 5.0?
- Calendar Server Services
- Group Scheduling
- Horizontal Scalability
- New Default Client UI: SHTML
- Architecture Basics
- Calendar Data
- Calendar Server API (CSAPI)
- Event Notification Service (ENS)
- Proxy Authentication SDK (authSDK)
- Single Sign-on (SSO)
- Web Calendar Access Protocol (WCAP)

# What is iPlanet Calendar Server?

iPlanet Calendar Server is a readily deployable, LDAP-based solution that lets the Internet service provider and Telecommunication service provider offer group scheduling features, as well as host personal event calendars, to their base of subscribed customers.

iPlanet Calendar Server 5.0 is a system of servers that can be configured to fit a variety of needs. It can stand in isolation as a stand-alone calendar server, or it can be configured with many instances, having the various services duplicated or split between them as was discussed earlier in this chapter. See “Horizontal Scalability,” on page 23. iPlanet Calendar Server 5.0 stores and manages calendars, calendar properties, access control information, events, todos, and alarms. It does not manage storage for user information. The minimal system requires a directory service. It makes use of plug-ins to obtain external services. It uses a directory service to perform operations such as authentication, and storage and retrieval of user preferences. iPlanet Calendar Server ships with plug-ins to perform directory services using LDAP services. You may use your own plug-ins to support non-LDAP directory services.

Figure 1-1 shows a minimal iPlanet Calendar Server (iCS) 5.0 system. It consists of a single iPlanet Calendar Server instance, a directory service, and support for event notifications.

**Figure 1-1** Minimal iPlanet Calendar Server System

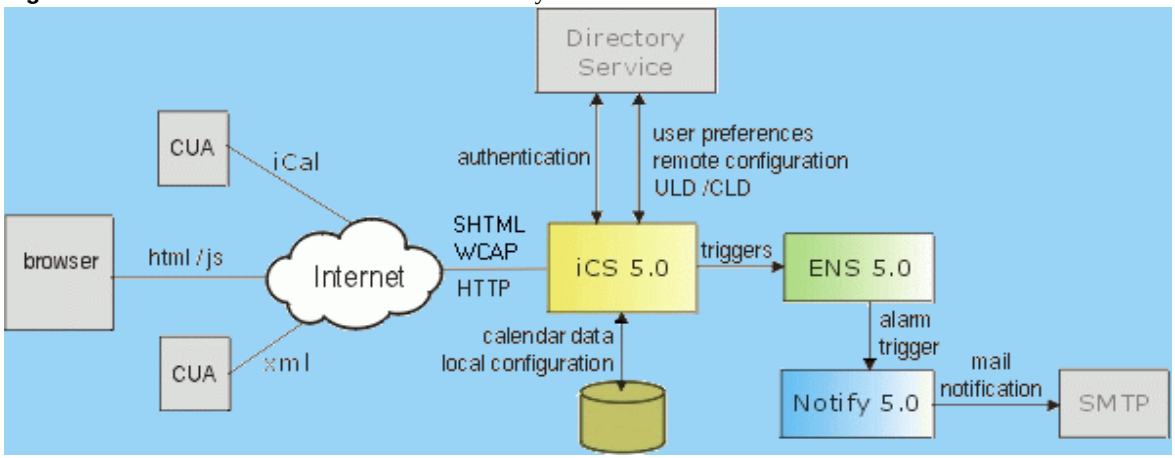


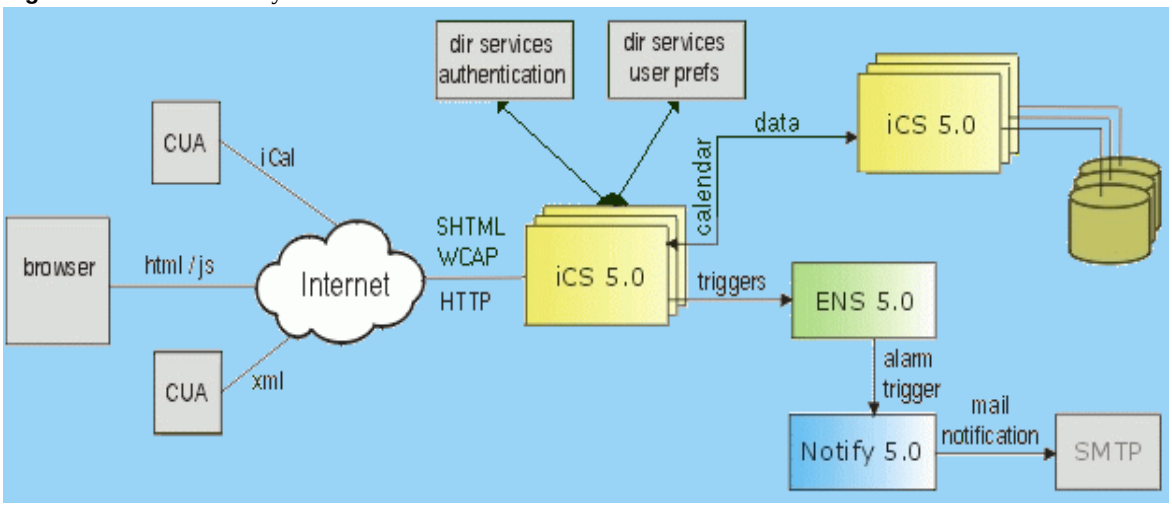

Table 1-1 is the key to the abbreviations in Figure 1-1.

**Table 1-1** iPlanet Calendar Server System Key to Terms

| | |
|---|---|
| CLD | Calendar Lookup Database |
| CUA | Calendar User Agent |
| ENS | Event Notification Server |
| iCal | iCalendar-RFC 2445 |

iPlanet Calendar Server is scalable both vertically and horizontally. On a single machine it can run in multiple processors, or it can be split up to run on multiple machines. Figure 1-2 depicts three iPlanet Calendar Server HTTP “front end” services, using three other iPlanet Calendar Server “database” services. All six of these instances can run on separate machines. The abbreviation “dwp” in this figure stands for Database Wire Protocol. For further information on this new protocol and the various configuration possibilities using horizontal scalability, see “Horizontal Scalability,” on page 23.

**Figure 1-2** Scalable System




# Summary of Features

Built from the ground up to provide native support for the emerging set of internet calendar standards, iPlanet Calendar Server 5.0 provides the following benefits:

- **Group Scheduling**–Organizers create an event, and invite attendees. Attendees accept or decline invitations. If the attendee is not on the calendar server, the group scheduling engine can send the scheduling message by email as an IMIP message (as described in RFC2447).
- **Internet Calendaring and Scheduling**–Native support for iCalendar calendaring standards ensures events are in a format that is easily shared across the internet.
- **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.
- **Massively Scalable**–iPlanet Calendar Server 5.0 is scalable both vertically and horizontally.

  Scaling vertically to the requirements of the largest service providers, it supports a hosting environment of up to several million personal event calendars. It runs on a single machine in multiple processors to take advantage of all the machine's processing power.

Scaling horizontally, it also splits up to run on several computers in many combinations. For example, there could be three Calendar Server HTTP services utilizing three other Calendar Server database services, each running on a separate machine. For further information on this new feature, see "Horizontal Scalability," on page 23.

# What's New in Version 5.0?

iPlanet Calendar Server 5.0 adds several new features:

- **Group scheduling**–In addition to keeping personal calendar information, users can now invite other calendar users to meetings who in turn may accept or decline the request. For more details on this new feature, see "Group Scheduling," on page 22.
- **Horizontal scalability**–The server can run on a single machine or its processes can be divided across multiple machines with a wide variety of possible configuration options. For more details on this new feature, see "Horizontal Scalability," on page 23.
- **New default client user interface**–Calendar Express, the bundled calendar client UI, now uses SHTML which achieves quicker browser rendering and response times as well as faster and easier customization that you can tailor to your site.
- **Migration from iPlanet Calendar Server 2.x**–A bundled utility that allows an administrator to import data from an existing iPlanet Calendar Server 2.x installation. The migration process is accomplished by running a separate program after the installation of version 5.0 has been completed successfully. For more details, see the *iPlanet Calendar Server Installation Guide.*
- **Synchronization**–iPlanet Synchronization 1.0 software is an optional application available for download (not included as part of the iPlanet Calendar Server 5.0 installation package) that allows users to synchronize their online calendar data with Palm OS devices, Windows CE devices, and ACT!, Outlook, and Palm Desktop applications.

iPlanet Calendar Server 5.0 builds on the consumer focus of iPlanet Calendar Server 2.x with the introduction of multiple-machine horizontal scalability as the design focus.This new architecture allows group scheduling capabilities using notifications. To implement notifications, iPlanet Calendar Server ships with Event Notification Service (ENS). A key benefit of this architecture is that you may customize many of the components, or even develop your own applications based on your customer's needs.

These new features required fundamental changes in some of the interfaces and tools described in this reference. Additional customization interfaces have been added to this reference. New this time is the Proxy Authentication SDK.

For continuity, you can still use WCAP, which supports all iPlanet Calendar Server 2.x data formats. For further information on WCAP, see Chapter 9, “Web Calendar Access Protocol (WCAP) Overview” and Chapter 10, “WCAP Commands”.

# Calendar Server Services

The iPlanet Calendar Server 5.0 system consists of several running processes that include multiple calendar server daemons which can run on a single machine or be divided to run on multiple machines. The specific combination of modules for a given running instance are defined in the server configuration information stored in the file `ics.conf` and can be modified using command line administration tools. The administration design supports a single service on a single machine or multiple services on multiple machines. The administrator can also issue commands remotely from a machine other than where the calendar service is running. Table 1-2 lists the Five iPlanet Calendar Server 5.0 daemons.

**Table 1-2** iPlanet Calendar Server Daemons

| | |
|---|---|
| `csadmind` | Administration service. Includes the Group Scheduling Engine (GSE), and monitors for alarms. |
| `csdwpd` | Interprocess database service. |
| `cshttpd` | HTTP service. Services SHTML and WCAP requests. |
| `csnotifyd` | Notification service. This is required in instances where the database resides. |
| `enpd` | Event Notification Service. |

## csadmind

This service provides alarm notifications, group scheduling requests, database checkpointing and deadlock detection, as well as disk usage and server response monitoring.

## cshttpd

iPlanet Calendar Server 5.0 uses HTTP as its primary transport. This service listens for HTTP commands and retrieves and returns data to the caller. For the new 5.0 user interface, commands received with the default .shtml extension returns data formatted in HTML. Alternately, for requests received with the .wcap extension, data can be formatted either as raw calendar data in standard RFC2445 iCalendar, XML, or JavaScript embedded in HTML.

## csnotifyd

The notification service (`csnotifyd`) sends calendar-based notifications of events and tasks and utilizes the Event Notification Server (ENS) as the broker for events. It subscribes to alarm events. When an alarm event occurs, it sends an SMTP message reminder to the recipients. For more information, see “Event Notification Service (ENS),” on page 34, Chapter 4, “Event Notification Service (ENS) Overview”, and Chapter 5, “Event Notification Service API Reference”.

## csdwpd

This service allows multiple machines within the same system to be linked together to form a distributed calendar store. The service can run in the background on any machine on which iPlanet Calendar Server 5.0 is installed. It acts as a service accepting requests that abide by the Database Wire Protocol (DWP) for calendaring information.

This service should be run only on a server that:

- Has a local calendar store.
- Must provide network access to its calendar data from other iPlanet Calendar Server installations.

**NOTE** This should only be done on a fast network. If the pipe (network) between the various databases is slow, it can seriously degrade overall system performance.

## enpd

This is the other half of the Event Notification Service. It acts as the broker for event alarms. It receives notifications of alarms from the `csadmind` daemon, checks for subscriptions to this event, and notifies the event's subscribers by passing the subscribed-to alarm notifications to `csnotifyd`. It also receives and stores subscriptions and cancellations of subscriptions (unsubscribe) from `csnotifyd`.

## Start Order

The iPlanet Calendar Server 5.0 daemons must be started in a specific order:

1. `enpd`–A generic event registration and notification service that can be shared by other iPlanet servers
2. `csnotifyd`–Calendar Server Notification Daemon
3. `csadmind`–Calendar Server Administration daemon (installation required on every server machine)
4. `csdwpd`–Calendar Server DataBase Daemon (only started with remote database configuration)
5. `cshttpd`–Calendar Server Daemon (at least one is required)

# Group Scheduling

In version 2.x, you could schedule events and todos on your own calendar and share your calendar with others, but with iPlanet Calendar Server 5.0, you may now make scheduling requests for other calendars. You may schedule events and invite attendees and they may accept or decline the request. When an attendee accepts or declines, the server updates the calendars of all attendees, as well as the event organizer. Attendees not in the installation's database are notified by email.

## Directory Server Services

By default, iPlanet Calendar Server supports users that are defined and maintained in an LDAP directory, such as Netscape Directory Server. iPlanet Calendar Server also supports the use of CSAPI plug-ins that you create to enable access for users defined in non-LDAP directories, such as those stored in a standard Unix authentication format, or in a Windows NT User Manager database.

If your users are already stored in an LDAP directory, the simplest solution for deploying iPlanet Calendar Server is to upgrade your directory server to Netscape Directory Server 4.12, and iPlanet Calendar Server installation will do the rest. Otherwise, you can modify your directory schema manually to allow your users to access iPlanet Calendar Server data. For more information on how to modify a directory schema for iPlanet Calendar Server, see “Installing and Configuring an LDAP Server” the *iPlanet Calendar Server Installation Guide.*

# Horizontal Scalability

Horizontal scalability may be achieved by spreading an installation over several machines. iPlanet Calendar Server consists of the daemons `cshttpd`, `csadmind`, `csdwpd`, `csnotifyd`, and `enpd`. These daemons may be run in different configurations to allow you great flexibility and scalability.

To facilitate horizontal scalability in iPlanet Calendar Server 5.0, the system employs an internal proprietary protocol, Database Wire Protocol (DWP), `csdwpd`. It was necessary to implement this protocol because the iPlanet Calendar Server 5.0 product uses a default Berkeley DB, which is not a networked database. The DWP protocol uses HTTP as it’s base. The implementation is simply an HTTP `POST` or `GET` command, with a single binary MIME part that contains serialized binary database information.

In a future release, the calendar database API will be published. It can be implemented using any database technology. DWP will not be needed by any implementation that supports a networked database.

## Configurations

To achieve horizontal scalability, you install various instances of iPlanet Calendar Server 5.0 across your machines. The basic requirements for every system are:

- Each instance must have `csadmind`.
- All of the other daemons are required to be installed at least once.

  The exception to this is the case of a single-instance installation using a local database connection. In this simple case, the `csdwpd` daemon is not necessary.

Table 1-3 indicates three possible configurations and the services you need to install for each instance. The figures that follow the table illustrate these configurations. Other configurations are possible. You must determine what combination addresses your specific needs.

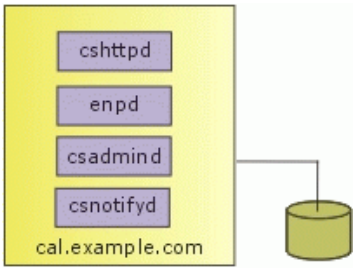
**Table 1-3** Instance Configuration of Required Services

| Instances | Required Services | | | | |
|---|---|---|---|---|---|
| | `csadmind` | `cshttpd` | `csnotifyd` | `csdwp` | `enpd` |
| Simple Single Instance installation with a local database. See Figure 1-3 that follows. | Y | Y | Y | N | Y |
| HTTP Service-only instances. (Used only in conjunction with Database Service-only instances.) | Y | Y | N | N | N |
| Examples: See boxes on left side of Figure 1-4, Network Front End, Database Back End, and Figure 1-5, Multiple Front Ends, Multiple Back Ends | | | | | |
| Database Service-only instances. (Used only in conjunction with HTTP Service-only instances.) | Y | N | Y | Y | Y |
| Examples: See boxes on right side of Figure 1-4, Network Front End, Database Back End, and Figure 1-5, Multiple Front Ends, Multiple Back Ends | | | | | |

# Simple Single Instance

This is the simplest configuration in which iPlanet Calendar Server 5.0 can run. As illustrated in Figure 1-3, it consists of `cshttpd` to handle incoming SHTML and WCAP requests, `enpd` and `csnotifyd` for event notification, and the required `csadmind`. The entire database is local.

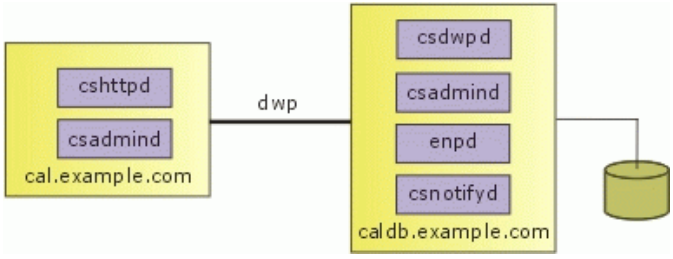
**Figure 1-3** Simple Single Instance Configuration




# Network Front End, Database Back End

In this configuration, illustrated in Figure 1-4, browsers and other clients connect to the calendar server at the HTTP Service front-end, labeled `cal.example.com` in this example. All requests for calendar data are routed to the Database Service, labeled `caldb.example.com` in this example. Notice that the front end requires only `cshttpd` and `csadmind` since it is not doing any database processing. The back end, conversely, does not need `cshttpd`, but does require `csdwpd`, `enpd`, and `csnotifyd`.

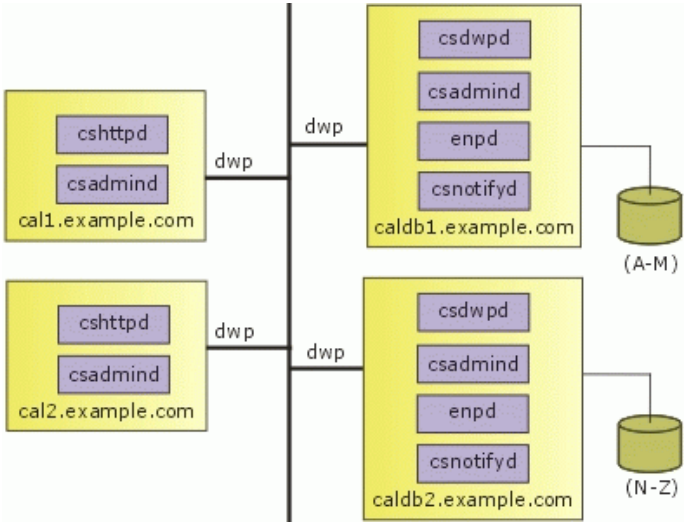
**Figure 1-4** Network Front End, Database Back End



# Multiple Front Ends, Multiple Back Ends

In this configuration, illustrated in Figure 1-5, clients are routed to one of the front end HTTP Services by some external mechanism that you provide. Please note that the session ID returned at login is valid only on the host where the login occurred. All requests for this session ID must be routed to the same host or the user will be forced to log in again. In this example, the database has also been split, with calendars `A-M` on the `caldb1.example.com` server, and calendars `N-Z` on the `caldb2.example.com` server. A CSAPI plug-in handles mapping between calendar IDs and the name of the server on which it can be found. The default CSAPI implementation provided in iPlanet Calendar Server 5.0 uses an algorithm to associate a calendar ID with a server name.

**Figure 1-5** Multiple Front Ends, Multiple Back Ends



# New Default Client UI: SHTML

iPlanet Calendar Server 5.0 no longer implements the default client UI with the WCAP protocol as it did in version 2.x. The WCAP protocol generated a mixture of HTML and JavaScript, and passed it to the client for processing. Using the new SHTML commands, the server now does all of the processing, and generates and sends only formatted output (HTML) to the client. The new SHTML commands use XML prototype definitions and XSL style-sheet templates to generate HTML. For each view and dialog displayed in the UI, there is one or more corresponding pairs of text files. Each pair consists of one .xml, and one .xsl file. You may alter or replace one or both of these files in order to customize the UI.

If you already have a custom user interface that was developed for version 2.x, you may continue to use it without change. iPlanet Calendar Server 5.0 is fully backward compatible with iPlanet Calendar Server 2.x.

WCAP remains the only way to retrieve unprocessed calendar data. Clients that need raw, unformatted calendar information should submit requests in WCAP format.

# Architecture Basics

iPlanet Calendar Server 5.0 is implemented by way of a collection of shared libraries. These shared libraries are bound in various combinations to produce the executable daemons `cshttpd`, `csdwpd`, `csadmind`, and `csnotifyd`.

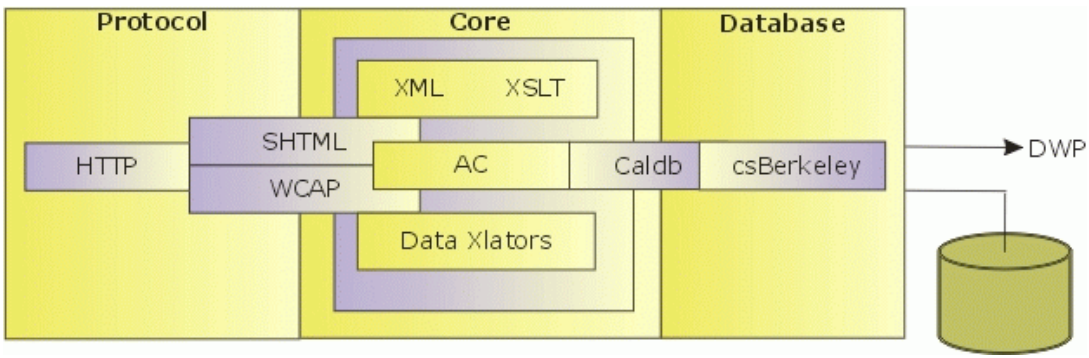
The Event Notification Service (ENS) daemon, `enpd`, is a separately installed service shipped with iPlanet Calendar Server 5.0. For further information on ENS, see "Event Notification Service (ENS)," on page 34 in this chapter, Chapter 4, and Chapter 5 of this Reference.

The shared libraries fall into three main categories, called *subsystems*:

- Protocol
- Core
- Database

Figure 1-6 represents the logical flow through these subsystems. A section follows on each of the subsystems.

**Figure 1-6** Server Architecture



## SHTML and WCAP

SHTML and WCAP are based on HTTP. Requests enter through the HTTP protocol layer. This is a minimal HTTP server implementation, streamlined to support calendar requests. Clients use either SHTML or WCAP to submit requests. WCAP is an open protocol that can perform all server commands (except for certain administrative commands). It can be used by clients that need raw, unformatted calendar information. It can also be used to obtain a JavaScript based user interface. This user interface was the only one available in iPlanet Calendar Server 2.x, but in version 5.0, it has been replaced with a new SHTML-based user interface. This new approach, based on XML and XSLT specifications, generates a user interface in response to commands. In response to an incoming request, the UI generator uses an XML specification to build a document tree with calendar and user data, subject to access control. The XSLT specification then traverses the document data tree and emits HTML. One of the benefits of this design is that it results in fewer interactions between the client and server and fewer bits being sent overall. Previously, in iPlanet Calendar Server 2.x, the system returned embedded JavaScript, which generated HTML at the client. The XML / XSLT approach generates output that is rendered faster by browsers.

For backward compatibility, the iPlanet Calendar Server 2.x design will continue to work, with WCAP requests returning a combination of HTML text and JavaScript as the default. As in iPlanet Calendar Server 2.x, commands using the .wcap extension may also request output as XML wrapped in HTML, or as iCalendar wrapped in HTML.

For a description of the WCAP protocol, see Chapter 9, “Web Calendar Access Protocol (WCAP) Overview”.

## Core

Inside the Core subsystem, other divisions include the Access Control subsystem, the UI Generator subsystem (either SHTML, using XML and XSLT, or WCAP, using data translators), and a Caldb Subsystem. CSAPI plug-ins reside in the Core.

The protocol processes the command and sends it to the Core for execution. There are four types of commands:

- **Calendar manipulation**–For Calendar manipulations requiring database actions, the protocol sends requests to the Core’s Access Control subsystem. The Access Control subsystem uses the generic Caldb subsystem to perform calendar read and write operations.

The Caldb subsystem uses a database-technology-specific subsystem to perform its operations. For this version of iPlanet Calendar Server with the Berkeley database implementation, this subsystem is capable of dealing with a local database file or making Database Wire Protocol (DWP) requests to the appropriate machine.

The Database subsystem returns data in a low-level format. The Core UI generator (either SHTML or WCAP) translates the low-level data into the desired output. Commands with the .shtml extension always default to HTML. Commands with the .wcap extension return the output format requested. WCAP remains the only open protocol the server supports. Use it to retrieve unformatted calendar data.

**NOTE** `HTML/js`, which was the default UI output for iPlanet Calendar Server 2.x, has been deprecated in favor of HTML for iPlanet Calendar Server 5.0.

- **User attributes**–Requests for user attributes go to the Core's directory service. iPlanet Calendar Server 5.0 makes use of plug-ins to obtain external services, such as directory services. The product ships with a plug-in for LDAP services. To customize your installation, you can write other plug-ins to support non-LDAP directory services.
- **Authentication**–Requests for authentication go to the Core's default LDAP directory service. iPlanet Calendar Server 5.0 ships with three different authentication options:
  - CSAPI authentication, see Chapter 2, "Calendar Server API (CSAPI) Overview". Internal authentication.
  - Proxy Authentication SDK, see Chapter 6, "Proxy Authentication SDK Overview". External plug-in authentication.
  - Single Sign-on authentication, see Chapter 8, "Single Sign-on Authentication". Applications form circles of trust in a single domain.

  An API for customization exists for CSAPI authentication and the Proxy Authentication SDK.
- **Miscellaneous**–Requests for miscellaneous services go to other Core subsystems.

## Database

iPlanet Calendar Server 5.0 uses the csBerkeley subsystem using the Berkeley DB from Sleepycat. The database API is not public.

# Calendar Data

This section describes various aspects of calendar data:

- Calendar Data Format
- Groups
- Event Feeds
- Calendar Data Exchange
- Calendar User Preferences
- Calendar Access Control
- Supported Format Encoding

## Calendar Data Format

Calendar data format is modeled after the IETF iCalendar standard RFC-2445. The Access Control layer maintains calendars, which are collections of iCalendar components. The components include events, todos, and alarms. A calendar has one primary owner, and may have other owners. User attributes are maintained by an external mechanism. The default mechanism is LDAP.

## Groups

Groups are named list of calendars. Users subscribe to calendars. A user can view and modify a subscribed calendar, subject to access control. Additionally, users can create groups. They can then work with groups of calendars, rather than having to re-specify or select a list of calendars every time they wish to view them side-by-side, or invite their owners to an event.

Groups 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 department calendar, the holidays calendar, and the “latest action video releases” calendar, all of which is called a “calendar group”.

## Event Feeds

This infrastructure makes it possible to feed real-time event data into the database using import formats such as iCalendar or XML. Event data can be fed in from calendars such as the local sports team’s season schedule, convention center schedules, concert schedules, or any schedule of events that may be of interest.

iPlanet 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 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.

iPlanet 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 maintained as a server attribute, rather than as a user or calendar attribute. iPlanet Calendar Server 5.0 has limited support for the ITIP / IMIP standards [RFC-2446, RFC-2447]. It supports ITIP methods `PUBLISH`, `REQUEST`, `REPLY`, and `CANCEL` for events.

## Calendar User Preferences

iPlanet 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_userprefs` and `set_userprefs` command descriptions in Chapter 10, “WCAP Commands”.

# Calendar Access Control

In iPlanet Calendar Server 5.0, access control is the mechanism that determines who can access a calendar when performing group scheduling. An Access Control Entry (ACE) string specifies the type of access privileges to a calendar that are granted to a user. These strings are stored in the access control calendar property `acl` (Access Control List) and collectively determine the access control of a calendar. Only users with write access to a calendar's properties can successfully change these strings. By default, only the Calendar Server administrator and the primary owner of a calendar have write access to its properties. The iPlanet Calendar Server access control model also supports the ability to act in behalf of others. For example, with this type of access granted to them, administrative assistants can invite, cancel, and reply to events on behalf of people they support.

Access control is described in more detail in the following sections:

- For more information on "Access Control Entries" in Chapter 10, "WCAP Commands".
- For more information on configuration settings, see the "*iPlanet Calendar Server Administration Guide*".

# Supported Format Encoding

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

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

You can add other formats by developing your own XSL translations for the UI views and dialogs, or, using CSAPI, you can develop a translator DLL or shared library for the WCAP protocol. (See Chapter 2, "Calendar Server API (CSAPI) Overview".)

# Calendar Server API (CSAPI)

CSAPI is a COM-like interface that allows programmers to implement customized parts of the server. Use CSAPI to modify the following areas of functionality:

- Access Control. (See “csIAccessControl,” on page 46.)
- Authentication. (See “csIAuthentication,” on page 48.)
- Calendar Lookup. (See “csICalendarLookup,” on page 53 and “csIQualifiedCalidLookup,” on page 62.)
- Data Format Translation. (See “csIDataTranslator,” on page 56.)
- User Attribute Access. (See “csIAccessControl,” on page 46.)

For example, the server’s default mechanism for authentication uses LDAP, and user preferences are stored in LDAP by default. If you have an existing infrastructure for authenticating users and saving user preferences that is not based on LDAP, use CSAPI to override the default mechanisms and use your existing authentication and directory services. In addition, there are three other customizable interfaces:

- `csIPlugin`. Upon startup, provides information to the server about your plug-in modules.
- `csICalendarServer`. Provides version information for the running server instance.
- `csIMalloc`. Memory allocation scheme.

CSAPI is described in detail in Chapter 2, “Calendar Server API (CSAPI) Overview” and in Chapter 3, “CSAPI Reference”.

# Event Notification Service (ENS)

In iPlanet Calendar Server, the primary use of the Event Notification Service (ENS) is as an alarm dispatcher, which detects events on an alarm queue and sends notifications of these events to its subscribers. The ENS API allows programmers to modify publish and subscribe functions used by iPlanet Calendar Server 5.0 to:

- Subscribe to events.
- Unsubscribe to events.
- Notify a subscriber of events.

For an overview of ENS, see Chapter 4, “Event Notification Service (ENS) Overview”.

For a detailed description of the functions in each area, see Chapter 5, “Event Notification Service API Reference”.

# Proxy Authentication SDK (authSDK)

The authSDK is one of the three tools iPlanet Calendar Server 5.0 offers for user authentication. With authSDK, you can integrate your existing portal service with iPlanet Calendar Server 5.0, thus allowing your users to access various applications without the necessity of re-authentication. The authSDK consists of five functions packaged in a DLL / shared-object library, `libicsexp10`, and a header file, `expapi.h`. These functions perform three simple tasks:

- Initialization
- Lookup
- Cleanup

In addition, two other functions allow you to use a non-standard port, and to get the authSDK version number for troubleshooting.

**NOTE** Once a connection has been established between iPlanet Calendar Server and the authSDK, the relationship set in place is one of trust. Therefore, once a user has logged in and has successfully authenticated to the authSDK, Calendar Server accepts the certificate generated by the proxy for all of its applications.

For further information on the authSDK, see Chapter 6, “Proxy Authentication SDK Overview” and Chapter 7, “Proxy Authentication SDK Reference”.

# Single Sign-on (SSO)

Single Sign-on (SSO) is one of the three authentication mechanisms offered by iPlanet Calendar Server 5.0. To use Single Sign-on, the client browser must support cookies and your server must support HTTP. Single Sign-on is independent from any other authentication mechanisms, session management, and resource access control.

With Single Sign-on, applications form circles of trust that share cookies and accept each others' user authentication. Each application can have its own verification interface, if necessary. Each verification authority, however, stores a cookie that is understood by the other applications' verification authority routines.

There are some limitations to this mechanism. The primary limitation is that each application must implement the verification protocol. Also, all trusted applications need to be in the same domain. And, finally, to switch to a different identity, the user must restart the browser because each browser session can support only one user ID.

For further information on Single Sign-on, see Chapter 8, "Single Sign-on Authentication".

# Web Calendar Access Protocol (WCAP)

WCAP is a command based system consisting of client requests and server responses for transmitting calendaring data. WCAP 2.0 returns calendaring data via HTTP. With WCAP commands, you can get, delete, and modify calendar components, user preferences, calendar properties, and other calendar information like time zones. All times, strings, parameters, etc. follow RFC2445, RFC2446, and RFC2447 specifications, unless otherwise specified.

One of the three user authentication mechanisms offered by iPlanet Calendar Server 5.0 is the WCAP default of plain-text passwords and user names. You may replace or augment the authentication mechanism that WCAP uses. See Chapter 3, "CSAPI Reference" for the section "csIAuthentication," on page 48.

WCAP supports the following client request and server response data formats:

- Calendar data in plain text format (HTML only). This is the new default format for the UI.
- Calendar data in `text/calendar` format (iCalendar).
- Calendar data in `text/xml` format. An XML-style version of iCalendar.
- Calendar data as `text/js` with embedded JavaScript objects. This was the default for the iPlanet Calendar Server2.x user interface.

Chapter 2

# Calendar Server API (CSAPI) Overview

This chapter gives an overview of the Calendar Server API (CSAPI), a set of high performance programmatic interfaces that enables you to modify or enhance the feature set of the iPlanet Calendar Server 5.0. 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
  - Thread Safe Requirement
  - Dependencies
- Using CSAPI
  - Loading CSAPI Modules
  - Plug-in Interfaces
  - Client and Server APIs
- CSAPI Samples

# CSAPI Architecture

The CSAPI is a group of shared-object runtime interfaces to Calendar Server functions. You can use plug-in CSAPI modules to manipulate server data for incoming requests and for responses. 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.

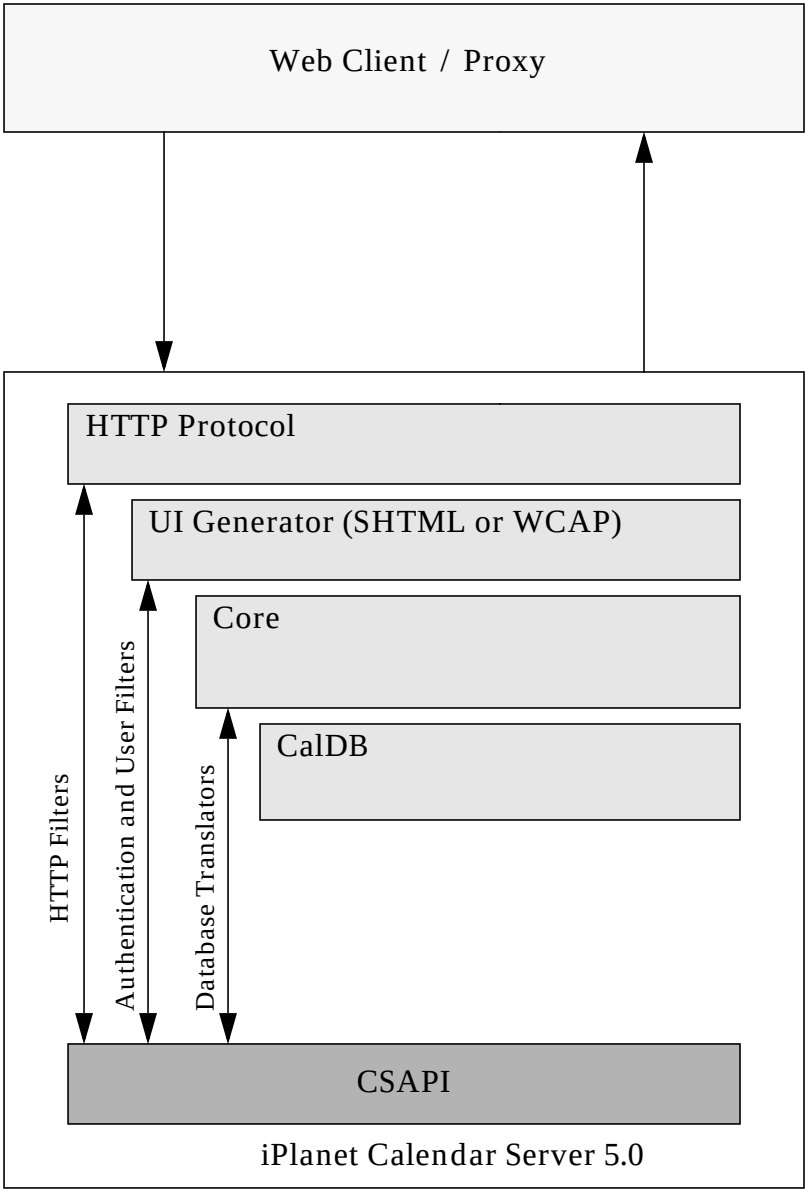
Figure 2-1, which follows, shows the relationship of CSAPI modules to other subsystems within iPlanet Calendar Server. Depending on which functional group or groups a CSAPI module supports, it can interact with one or more areas of iPlanet Calendar Server functionality, such as data formatting, authentication, and directory services.

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; see Chapter 3, “CSAPI Reference”. Each client-side interface addresses a functional area of iPlanet Calendar Server. The implementation contained in a module can either augment or override the native iPlanet Calendar Server functionality in its area.

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

The installation contains plug-ins for each of the CSAPI interfaces with the default code, which you can use as templates to create your own plug-ins, along with all supporting libraries and headers.

**Figure 2-1** CSAPI Relationship to Other Subsystems



## Thread Safe Requirement

CSAPI module plug-ins must be thread 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 at mozilla.org. For the URL, see the “Dependencies” section later in this chapter.

## Dependencies

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/projects/xpcom

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

# Using CSAPI

The following section describes how the system loads and uses the plug-ins you provide. Default plug-ins ship with the system. You may choose to augment or override any or all of them.

## Loading CSAPI Modules

iPlanet Calendar Server loads CSAPI modules from the `cal/bin/plugins` 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 `ics.conf` for the modules to be dynamically loaded at server startup. If the value of the preference `csapi.plugin.loadall` is `y`, the server loads all shared objects in the `cal/bin/plugins` directory whose names begin with the prefix `cs_`. Otherwise, if the value is `n`, various preferences exist for the various plug-ins. For more information on the preferences in `ics.conf`, see the *"iPlanet Calendar Server Administration Guide"*.

To specify the loading of a specific plug-in, `csapi.plugin.loadall` must be set to `n`. In addition, two preferences must be used: `csapi.plugin.`*`plugin name`*, with a value of `y`, and `csapi.plugin.`*`plugin name`*`.name`, with the value being the name of the plug-in. For example, for the calendar-lookup plug-in, the preferences are:

```
csapi.plugin.loadall = "n"

csapi.plugin.calendarlookup = "n"

csapi.plugin.calendarlookup.name = " "
```

Note that the `plugin name` part of the preference must match on both preferences, but that it does not have to be the same as the value of the `.name` preference. Thus, if you wanted to create a plug-in called `cs_myown_plugin`, you could call the preferences `csapi.plugin.anyname`, and `csapi.plugin.anyname.name`. The value of `csapi.plugin.anyname.name` would be "`cs_myown_plugin`".

iPlanet 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, iPlanet Calendar Server uses the function `PR_FindSymbol` to find entry points to known API implementations.

# Plug-in Interfaces

All CSAPI plug-ins 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 an object implementing the `csIPlugin` interface. This allows the server to query the plug-in for version, description, and vendor information. This interface is optional; however, it is highly recommended that you implement it so the server can ensure version control.

# Client and Server APIs

The CSAPIs fall into two categories, client and server APIs.

Table 2-1 lists the CSAPI client interfaces, which may be implemented by one or more plug-ins.

**Table 2-1** CSAPI Client APIs

| CSAPI Module Interface | Description |
|---|---|
| `csIAccessControl` | Augments or overrides the default access control mechanism. |
| `csIAuthentication` | Augments or overrides the login authentication mechanism. |
| `csICalendarLookup` | Augments or overrides the default calendar lookup mechanism. |
| `csIDataTranslator` | Augments or overrides the format translation of incoming and outgoing data. |
| `csIPlugin` | Provides version control and descriptive information about the module. |
| `csIUserAttributes` | Augments or overrides the mechanism for storing and retrieving user attributes. |
| `csIQualifiedCalidLookup` | Retrieves a calendar ID for the specified qualified URL. |

The interfaces are described in detail in Chapter 3, “CSAPI 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. Your custom plug-in can use the `QueryInterface` method to find the server interfaces, listed in Table 2-2:

**Table 2-2** CSAPI Server APIs

| 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;
}
```

# CSAPI Samples

The distribution includes sample code for three 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 2-3** CSAPI Interface Samples

| 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:<br><br>On Solaris, it uses the `pam` library to authenticate against the `local /etc/passwd` file or NIS.<br><br>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 Microsoft 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. |

Chapter 3

# CSAPI Reference

This section details the nine CSAPI interfaces each of which is an API. The APIs are divided between client and server side.

Use the APIs listed in Table 3-1 and Table 3-2 to augment or override iPlanet Calendar Server's default behavior:

**Table 3-1** Client APIs

| | |
|---|---|
| `csIAccessControl` | Augments or overrides the access control mechanism. |
| `csIAuthentication` | Augments or overrides the login authentication mechanism. |
| `csICalendarLookup` | Augments or overrides the default calendar lookup mechanism. |
| `csIDataTranslator` | Augments or overrides the format translation of incoming and outgoing data. |
| `csIPlugin` | Provides version control and descriptive information about the module. |
| `csIUserAttributes` | Augments or overrides the mechanism for storing and retrieving user attributes. |
| `csIQualifiedCalidLookup` | Retrieves a calendar ID for the specified qualified URL. |

**Table 3-2** Server APIs

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

# csIAccessControl

Implement the methods in this interface to augment or override the default access control behavior of iPlanet Calendar Server.

### Methods

The csIAccessControl interface implements two methods:

| | |
|---|---|
| `CheckAccess` | Sets access control criteria for users. |
| `Init` | Confirms that the interface was found and registered. |

### Description

Defines the types of access allowed. You must set the return code to specify whether you are using the default access control or overriding the default.

## CheckAccess

### Purpose

Sets users' calendar access.

### Syntax

```
PRUint32 CheckAccess (char * aUser,
                      char * aCalid,
                      PRInt32 * aAccessRequest,
                      PRInt32 * aAccessAllowed,
                      PRInt32 * aReturnCode) = 0;
```

### Parameters

The method has the following five parameters:

| | |
|---|---|
| `aUser` | The authenticated user making the request. For anonymous access, `pUserID` is "anonymous". |
| `aCalid` | `calid` for the calendar being accessed. |
| `aAccessRequest` | A set of bit flags representing the requested access type. |
| `aAccessAllowed` | Output parameter. A set of bit flags representing the allowed accesses. The method checks only the bits specified in aAccessRequest. |

| | |
|---|---|
| aReturnCode | On return, contains a constant that determines whether the server should continue with the default authentication procedure.<br><br>One of the following constants:<br><br>• NS_CONTINUE_DEFAULT_PROCESSING<br><br>• NS_OVERRIDE_DEFAULT_PROCESSING |

### Returns

NS_OK on success. A non-zero error code on failure.

### Description

Use this method to request access types for this user. You send in the name of the user in the aUser parameter, and the access types requested in the aAccessRequest parameter (bitmask).The system checks for only those access types specified in the aAccessRequest bitmask. The returned bitmask, aAccessAllowed, represents the user's allowed access for the types you requested.

For anonymous access, the user ID is "anonymous".

ICS_ACCESSTYPE constants (bitmaps) that define available access types are as follows:

**Table 3-3** ICS_ACCESSTYPE Constants

| Access Types | Bitmaps |
|---|---|
| ICS_ACCESSTYPE_NONE | 0x00000000 |
| ICS_ACCESSTYPE_READCOMPONENT | 0x00000001 |
| ICS_ACCESSTYPE_WRITECOMPONENT | 0x00000002 |
| ICS_ACCESSTYPE_CREATECOMPONENT | 0x00000008 |
| ICS_ACCESSTYPE_DELETECOMPONENT | 0x00000010 |
| ICS_ACCESSTYPE_READCALENDAR | 0x00000020 |
| ICS_ACCESSTYPE_WRITECALENDAR | 0x00000040 |
| ICS_ACCESSTYPE_CREATECALENDAR | 0x00000080 |
| ICS_ACCESSTYPE_DELETECALENDAR | 0x00000100 |
| ICS_ACCESSTYPE_SCHEDULE | 0x00000200 |
| ICS_ACCESSTYPE_FREEBUSY | 0x00000400 |
| ICS_ACCESSTYPE_SELF_ADMIN | 0x00000800 |

**Table 3-3** ICS_ACCESSTYPE Constants

| Access Types | Bitmaps |
|---|---|
| ICS_ACCESSTYPE_ALL | 0xFFFFFFFF |

Use this method to specify your own access control procedure. You can augment the native access control mechanism, performing your own processing first, then continuing with the default process, or you can completely replace the native access control mechanism.

## Init

### Purpose

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

### Syntax

```
PRUint32 Init (nsISupports * a Server) = 0;
```

### 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. A 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.

# csIAuthentication

All plug-ins wishing to augment or override the default authentication behavior of the calendar server must implement this interface.

### Methods

The csIAuthentication interface implements five methods:

| | |
|---|---|
| `ChangePassword` | Change a user's password. |
| `Init` | Confirms that the interface was found and registered. |
| `Logon` | Logs in a user. |
| `Logout` | Logs out a user. |
| `VerifyUserExists` | Verify a user's existence. |

### Description

Allows you to define logon, logoff, verification, and password methods that implement the authentication technique of your choice. You may replace a method and still continue to use the default for the others. Each method uses the return code parameter (`aReturnCode`) to tell the server whether to continue with the default access control process after executing the method. The return code value must be one of the following constants:

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

## ChangePassword

### Purpose

Changes the password for the specified user.

### Syntax

```
PRUint32 Init (char * aUser,
               char * aOldPassword,
               char * aNewPassword,
               PRInt32 * aReturnCode) = 0;
```

### Parameters

The method has the following four parameters:

| | |
|---|---|
| `aUser` | The user's name. |

| | |
|---|---|
| aOldPassword | The old password. |
| aNewPassword | The new password. |
| aReturnCode | On return, contains a constant that determines whether the server should continue with the default authentication procedure.<br>One of the following constants:<br>• NS_CONTINUE_DEFAULT_PROCESSING<br>• NS_OVERRIDE_DEFAULT_PROCESSING |

#### Returns

On success, NS_AUTHENTICATION_CHANGEPASSWORD_SUCCESS. On failure, NS_AUTHENTICATION_CHANGEPASSWORD_FAILURE.

#### Description

Changes the password of the specified user.

## Init

#### Purpose

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

#### Syntax

```
PRUint32 Init (nsISupports * aServer) = 0;
```

#### 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. A non-zero 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) = 0;
```

### Parameters

The method has the following three 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.<br>One of the following constants:<br>• `NS_CONTINUE_DEFAULT_PROCESSING`<br>• `NS_OVERRIDE_DEFAULT_PROCESSING` |

### Returns

On success, `NS_AUTHENTICATION_LOGON_SUCCESS`. On failure, `NS_AUTHENTICATION_LOGON_FAILURE`.

### Description

Use this method to specify your own authentication procedure on login to iPlanet 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

# Logout

### Purpose

Logout a user.

### Syntax

```
PRUint32 Init (char * aUser,PRInt32 * aReturnCode) = 0;
```

#### Parameters

The method has the following two parameters:

| | |
|---|---|
| aUser | The user ID of the user to be logged out. |
| aReturnCode | On return, contains a constant that determines whether the server should continue with the default authentication procedure.<br>One of the following constants:<br>• NS_CONTINUE_DEFAULT_PROCESSING<br>• NS_OVERRIDE_DEFAULT_PROCESSING |

#### Returns

NS_OK on success. A non-zero error code on failure.

#### Description

## VerifyUserExists

#### Purpose

Verify that the user ID is in the LDAP directory.

#### Syntax

```
PRUint32 Init (char * aUser,PRInt32 * aReturnCode) = 0;
```

#### Parameters

The method has the following two parameters:

| | |
|---|---|
| aUser | The user ID of the user to be logged out. |
| aReturnCode | On return, contains a constant that determines whether the server should continue with the default authentication procedure.<br>One of the following constants:<br>• NS_CONTINUE_DEFAULT_PROCESSING<br>• NS_OVERRIDE_DEFAULT_PROCESSING |

#### Returns

NS_OK on success. A non-zero error code on failure.

### Description

# csICalendarLookup

Implement the methods in this interface to augment or override the default calendar lookup (LDAP).

### Methods

The csICalendarLookup implements five methods:

| | |
|---|---|
| `Init` | Confirms that the interface was found and registered. |
| `QualifyCalid` | Qualifies the relative `calid`. |
| `FreeCalid` | Frees a previously allocated, fully qualified `calid`. |
| `QueryType` | Queries the type of plug-in. |
| `FreeType` | Frees a previously allocated type. |

### Description

Allows you to control calendar lookup by implementing one or more of the methods.

## Init

### Purpose

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

### Syntax

```
PRUint32 Init (nsISupports * aServer) = 0;
```

### Parameters

The method has the following parameter:

| | |
|---|---|
| `aServer` | On return, 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.

## QualifyCalid

### Purpose

Qualifies the relative calid.

### Syntax

```
PRUint32 QualifyCalid (char * aRelativeCalid,
                       char ** aQualifiedCalid,
                       PRInt32 * aReturnCode) = 0;
```

### Parameters

The method has the following parameters:

| | |
|---|---|
| `aRelativeCalid` | The relative `calid` to be qualified. |
| aQualifiedCalid | On return, contains the URL of the qualified `calid`. |
| aReturnCode | `NS_OK` if successful. Normal processing will not continue if unsuccessful. |

### Returns

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

### Description

## FreeCalid

### Purpose

Frees a previously allocated, fully qualified calendar ID (`calid`).

### Syntax

```
PRUint32 FreeCalid (char ** aQualifiedCalid,PRInt32 * aReturnCode) =
0;
```

### Parameters

The method has the following parameters:

| | |
|---|---|
| aQualifiedCalid | `calid` to free. |
| aReturnCode | `NS_OK` if successful. Normal processing will not continue if unsuccessful. |

### Returns

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

### Description

## QueryType

### Purpose

Query type of database plug-in.

### Syntax

```
PRUint32 QueryType (char * aType,PRInt32 * aReturnCode) = 0;
```

### Parameters

The method has the following parameters:

| | |
|---|---|
| `aType` | The type of CLD (Calendar Lookup Database). |
| aReturnCode | `NS_OK` if successful. Normal processing will not continue if unsuccessful. |

### Returns

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

### Description

This function retrieves a string representing the type of CLD the plugin implements.

The only supported type is “algorithmic”, which supports regular expressions.

## FreeType

### Purpose

Frees a previously allocated database plug-in type.

### Syntax

```
PRUint32 FreeType (char * aType,PRInt32 * aReturnCode) = 0;
```

### Parameters

The method has the following parameters:

| | |
|---|---|
| aType | The database plug-in type to free |
| aReturnCode | NS_OK if successful. Normal processing will not continue if unsuccessful. |

### Returns

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

### Description

Frees the string allocated in QueryType method.

# csIDataTranslator

This is the interface for data translator plug-ins. All parameters should be allocated by the plug-in.

### Methods

The csIDataTranslator interface implements three methods:

| | |
|---|---|
| GetSupportedContentTypes s | Informs the server about the content types that this database translator supports. |
| Init | Confirms that the interface was found and registered. |
| Translate | Translates calendar data to the specified MIME format. |

### 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 output format (`fmt-out`) component of a WCAP response.

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

**Table 3-4** MIME Types

| MIME Type | Description |
|---|---|
| `text/calendar` | iCalendar |
| `text/xml` | iCalendar in XML |
| `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.

## GetSupportedContentTypes

### Purpose

Gets the content type this database translator supports.

### Syntax

```
PRUint32 GetSupportedContentTypes (char ** aSupportedInContentTypes,
                                   char ** aSupportedOutContentType,
                                   char ** aPreferredInContentType,
                                   PRInt32 * aReturnCode) = 0;
```

### Parameters

The method has the following parameters:

| | |
|---|---|
| `aSupportedInContentTypes` | A list of content-types that the server can send as input to translator. An array of `NULL`-terminated strings. |

| | |
|---|---|
| aSupportedOutContentType | The content type the plug-in will convert data to. |
| aPreferredInContentType | The content type the plug-in would prefer to receive. It must be one of the supported content types passed in the first parameter. |
| aReturnCode | On return, contains a constant that determines whether the server should continue with the default authentication procedure.<br>One of the following constants:<br>• NS_CONTINUE_DEFAULT_PROCESSING<br>• NS_OVERRIDE_DEFAULT_PROCESSING |

#### Returns

NS_OK on success. A non-zero on failure.

#### Description

Get the content type this database translator supports.

## Init

#### Purpose

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

#### Syntax

```
PRUint32 Init (nsISupports * aServer) = 0;
```

#### 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,
                    PRInt32 * aReturncode) = 0;
```

### Parameters

The method has the following parameters:

| | |
|---|---|
| `aInContentType` | The incoming content type. |
| `aOutContentType` | The outgoing content type. |
| `aInBuffer` | The input data buffer. |
| `aOutBuffer` | The output data buffer. |
| `aInSize` | The input buffer size. |
| `aOutSize` | The output buffer size. |
| `aReturnCode` | On return, contains a constant that determines whether the server should continue with the default authentication procedure.<br>One of the following constants:<br>• `NS_CONTINUE_DEFAULT_PROCESSING`<br>• `NS_OVERRIDE_DEFAULT_PROCESSING` |

### Returns

`NS_OK` 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.

# csIPlugin

You should implement the methods in this interface in order to provide the server with information about your plug-in module on startup.

### Methods

The `csIPlugin` interface implements four 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. The methods return descriptive information to the server.

## GetDescription

### Purpose

Retrieve a text description of the module.

### Syntax

```
PRUint32 GetDescription (nsString& aDescription) = 0;
```

### 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 GetVendorName (nsString& aVendorName) = 0;
```

### Parameters
The method has the following parameter:

| | |
|---|---|
| aVendorName | 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) = 0;
```

### Parameters
The method has the following two 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) = 0;
```

#### 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.

# csIQualifiedCalidLookup

Implement the methods in this interface to augment or override the default method of retrieving the calendar ID of the qualified URL passed in.

#### Methods

The csICalendarLookup implements two methods:

| | |
|---|---|
| FindCalid | Returns a calendar ID for the qualified URL. |
| Init | Confirms that the interface was found and registered. |

### Description

Retrieves the calendar ID of the qualified URL passed in to it. If the `calid` is not found, the command returns an error.

## FindCalid

### Purpose

Finds the calendar ID for the URL specified.

### Syntax

```
PRUint32 FindCalid (char * pQualifiedURL,
                    char ** ppCalidOut,
                    PRInt32 * piCalidSize,
                    PRInt32 * aReturnCode) = 0;
```

### Parameters

The method has the following parameters:

| | |
|---|---|
| pQualifiedURL | The URL to search on. |
| `ppCalidOut` | A pointer to the address of the `calid` found. |
| piCalidSize | Size of the `calid` `returned`. |
| aReturnCode | On return, contains a constant that determines whether the server should continue with the default, or with the override processing.<br>One of the following constants:<br>• `NS_CONTINUE_DEFAULT_PROCESSING`<br>• `NS_OVERRIDE_DEFAULT_PROCESSING` |

### Returns

The `calid` for the qualified URL passed in.

### Description

Uses the qualified URL pointer passed to it to perform a search of the calendar ID database. If it finds a match, it returns a pointer to the address of the `calid` and an integer with the size of the `calid`.

.

## Init

### Purpose

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

### Syntax

```
PRUint32 Init (nsISupports * aServer) = 0;
```

### Parameters

The method has the following parameter:

| | |
|---|---|
| `aServer` | On return, 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

.

# csIUserAttributes

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

### Methods

The `csIUserAttributes` interface implements four methods:

| | |
|---|---|
| `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 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

```
PRint32 FreeAttribute (char * aValue, PRInt32 * aReturnCode) = 0;
```

### Parameters

The method has the following two parameters:

| | |
|---|---|
| `aValue` | The location you allocated to contain the retrieved attribute value. |
| `aReturnCode` | On return, contains a constant that determines whether the server should continue with the default, or with the override processing.<br>One of the following constants:<br>• `NS_CONTINUE_DEFAULT_PROCESSING`<br>• `NS_OVERRIDE_DEFAULT_PROCESSING` |

### 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. (See csIMalloc.)

## GetAttribute

### Purpose

Retrieve an attribute value for a user.

### Syntax

```
PRUint32 GetAttribute (char * aUser,
                       char * aKey,
                       char ** aValue,
                       PRInt32 * aReturnCode) = 0;
```

### Parameters

The method has the following four parameters:

| | |
|---|---|
| `aUser` | The name of the user. |
| `aKey` | The attribute key. |
| `aValue` | On return, this location contains a pointer to the retrieved attribute value. |
| `aReturnCode` | On return, contains a constant that determines whether the server should continue with the default, or with the override processing.<br>One of the following constants:<br>• `NS_CONTINUE_DEFAULT_PROCESSING`<br>• `NS_OVERRIDE_DEFAULT_PROCESSING` |

### 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) = 0;
```

### 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

```
RUint32 SetAttribute (char * aUser,
                      char * aKey,
                      char * aValue,
                      PRInt32 * aReturnCode) = 0;
```

### Parameters

The method has the following parameters:

| | |
|---|---|
| aUser | The name of the user. |
| aKey | The attribute key. |
| aValue | The value. |
| aReturnCode | On return, contains a constant that determines whether the server should continue with the default, or with the override processing.<br>One of the following constants:<br>• NS_CONTINUE_DEFAULT_PROCESSING<br>• NS_OVERRIDE_DEFAULT_PROCESSING |

### Returns

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

### Description

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

# csICalendarServer

Provides server version information to a plug-in module.

### Methods

The csICalendarServer interface implements two methods:

| | |
|---|---|
| GetVersion | Get the calendar server version. |
| Init | Confirms that the interface was found and registered. |

### Description

Plug-in modules can query the csICalendarServer interface to get version information about the running instance of iPlanet Calendar Server. The object is valid for the full lifetime of the client, so Init does not return a reference.

## GetVersion

### Purpose

Provide plug-in module with server version information.

### Syntax

```
PRUint32 GetVersion (PRUint32& aMajorValue,
                     PRUint32& aMinorValue) = 0;
```

### Parameters

The method has the following two 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.

### Init

#### Purpose
Confirm that the interface has been registered.

#### Syntax
```
PRUint32 Init () = 0;
```

#### Parameters
The method has no parameters.

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

#### Description
The server calls this method to confirm that the interface was found and registered successfully.

## csIMalloc

Allocates and frees memory.

#### Methods
The `csIMalloc` interface implements six 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. |
| `Init` | Confirms that the interface was found and registered. |
| `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.The object is valid for the full lifetime of the client, so `Init` does not return a reference.

## Calloc

### Purpose

Allocates, and initializes to zero, memory for a number of objects.

### Syntax

```
void* Calloc (PRUint32 aSize, PPRUint32 aNum) = 0;
```

### Parameters

The method has the following two 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) = 0;
```

### 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 to free previously allocated memory.

## FreeIf

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

### Syntax
```
PRUint32 FreeIf (void * aPtr) = 0;
```

### 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.

## Init

### Purpose
Confirm that the interface has been registered.

### Syntax
```
PRUint32 Init () = 0;
```

### Parameters
The method has no parameters.

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

### Description

The server calls this method to confirm that the interface was found and registered successfully.

## Malloc

### Purpose

Allocate a specified amount of memory.

### Syntax

```
void* Malloc (PRUint32 nBytes) = 0;
```

### 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

Reallocates memory that was previously allocated.

### Syntax

```
void* Realloc (void * aPtr,PRUint32 nBytes) = 0;
```

### 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 to reallocate memory that was previously allocated.

Chapter 4

# Event Notification Service (ENS) Overview

The Event Notification Service (ENS) is a generic publish-and-subscribe service shipped with iPlanet Calendar Server 5.0. ENS acts as a dispatcher used by iPlanet applications as a central point of collection for certain types of events that are of interest to them. Specifically, ENS accepts reports of events that can be categorized, and notifies other applications that have registered an interest in certain categories of events.

In summary, ENS is an internet service used by applications to:

- Notify whomever is interested of an event.
- Subscribe to upcoming events (that is, ask to be notified of upcoming events).
- Cancel existing subscriptions (unsubscribe).

**NOTE** iPlanet offers a generic message-oriented middleware system, distinct from ENS, called Java Message Queue (JMQ). JMQ is a Java Message Service (JMS) implementation.

For now the ENS subscriber API is the only interface which can be used to access Calendar Server events. In particular, JMS may not be used to access such events. In the future, iPlanet messaging middleware APIs will consolidate around JMS, and JMS will, therefore, become the primary method to access Calendar Server events.

This chapter contains the following topics:

- ENS Glossary

- Events
- Event References
- Event Notification Service
- Calendar Server Interaction with ENSAPI Overview
- Building and Running Custom Applications

# ENS Glossary

| | |
|---|---|
| event | A change in a resource. For instance, someone adds a new meeting to a calendar (the resource). For ENS, events are a change in the state of the alarm queue. |
| event consumer | Synonym for event subscriber. |
| event producer | Synonym for event publisher. |
| event publisher | An application that makes events known to other applications. |
| event reference | Identifies an event handled by ENS. It complies with URI syntax defined by RFC 2396. |
| event subscriber | An application that consumes events. |
| notification | Message describing an event occurrence. Sent by the event publisher, it contains a reference to the event as well as optional data used by the event consumers, but opaque to the notification service. |
| notification service | Receives subscriptions and notifications from other servers. Relays notifications to subscribers. |
| notification server | A notification service is made up of one or more server instances, each running on a separate host. |
| notify | A synonym for publish. |
| publish | Send a notification. An event publisher makes an event available to the notification service. |
| reliable event notification link (RENL) | An RENL has a publisher, a subscriber, and a unique ID, which identify notifications that are subject to acknowledgment. |
| resource | A piece of data accessed from the IP network. For example, a calendar is a resource. |

| | |
|---|---|
| resource state | The value of attributes that describe a resource. For example, a meeting time. |
| subscribe | Send a subscription. An event subscriber tells the notification service that it wants to receive notifications of a specific event. |
| subscription | Message sent by the event subscriber. Contains an event reference, a client-side request identifier, and optional access control rules. |
| unsubscribe | Cancels a subscription. An event subscriber tells the event notification service to stop relaying notifications for the specified event. |

# Events

Events are changes to the value of one or more properties of a resource, a URI represents an event. Any application that wants to know when these types of events occur registers with ENS, which identifies events in order and matches notifications with subscriptions.

# Event References

Event references identify an event handled by ENS. They use the URI syntax specified by RFC 2396.

URI syntax:

event reference:= *scheme ":" authority resource* ["?" param "=" value *("&" param "=" value)]

where:

- *scheme* is the access method, such as `http`, `imap`, `ftp`, `wcap`.

  For iPlanet Calendar Server 5.0, the ENS scheme is `enp`.

- *authority* is the DNS domain or hostname which controls access to the resource.
- *resource* is the path leading to the resource in the context of the authority. It may be composed of several path components separated by "/".
- *param* is the name of a parameter describing the state of a resource.
- *value* is its value. There can be zero or more parameter/value pairs.

The following is the URI scheme for all iPlanet Calendar Server events:

```
enp://domain.com/calendar/ics/v50/instance
```

### Example

As an example, for an iPlanet Calendar Server 5.0 user to subscribe to all event alarms with a specific unique ID, the URI would look like:

```
enp://domain.com/calendar/ics/v50/ics-hostname/alarms?type=events
&uid=XXXX
```

# Event Notification Service

The Event Notification Service is an internet service used by applications to:

- Notify
- Subscribe
- Unsubscribe

ENS runs as a daemon, enpd, along with other iPlanet Calendar Server 5.0 daemons in various calendar server configurations. For further explanation of how ENS integrates with iPlanet Calendar Server 5.0, see “Calendar Server Interaction with ENS,” on page 80 of this chapter, and “Horizontal Scalability,” on page 23 in Chapter 1, “Architecture Overview”.

## Notify

ENS notifies its subscribers of an event by sending a notification containing an event reference, optional access control rules, and optional application-specific (opaque for ENS) data. Also called “publish”; see “Publisher API,” on page 91.

There are two kinds of notifications:

- **Unsolicited notification**. Notification sent from an event publisher to a notification server. If the publisher does not know nor care about whether there are any consumers, or whether they get the notification, this request does not absolutely need to be acknowledged. However, a publisher and a subscriber, who are mutually aware of each other, may agree to set up a reliable event notification link (RENL) between themselves. In this case, once the subscriber has processed the publisher’s notification, it sends an acknowledgment notification back to the publisher.

- **Server notification.** Notification sent from a server to a subscriber as a result of a subscription. This type of notification should be acknowledged. A server notification contains the same attributes as a spontaneous notification.

## Subscribe

ENS receives a request to be notified of events. The request sent by the event subscriber is a subscription. The subscription is valid during the life of the session, or until it is cancelled (unsubscribed).

A subscription contains an event name, a client-side request identifier, and optional access control rules. See “Subscriber API,” on page 98.

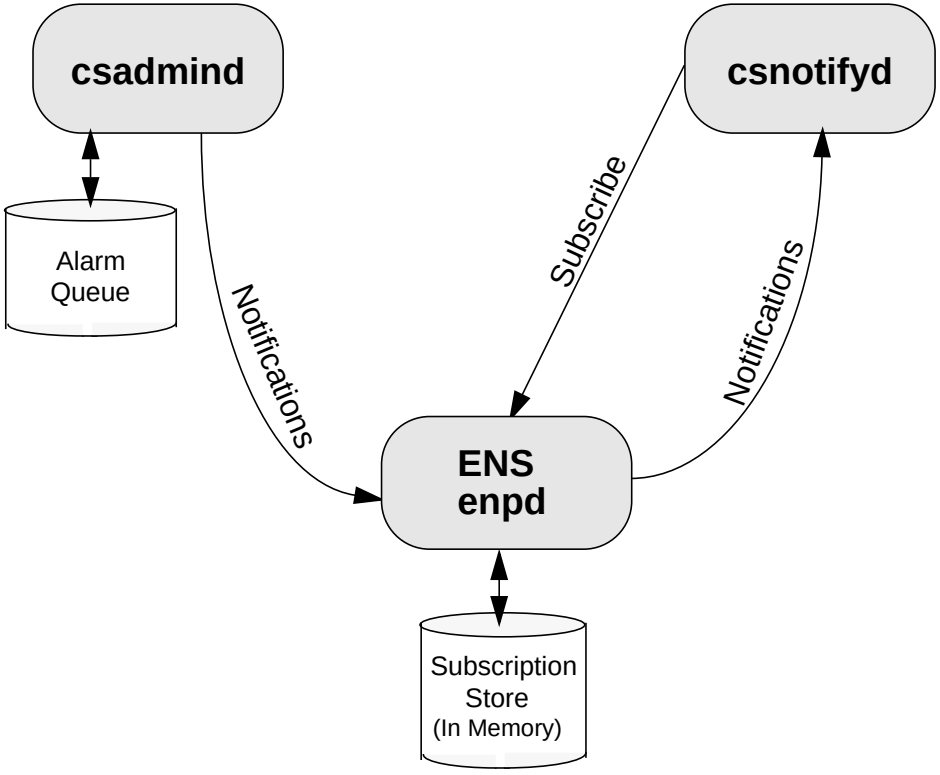
## Unsubscribe

ENS receives a request to cancel an existing subscription. The client-side request id, and the session id identify the subscription. See “Subscriber API,” on page 98.

# Calendar Server Interaction with ENS

ENS interacts with Calendar Server through the alarm queue and two daemons, `csadmind`, and `csnotifyd`. Figure 4-1 illustrates the interaction between the two Calendar Server daemons and the ENS daemon.

**Figure 4-1** ENS Overview



# Alarm Queue

ENS is an alarm dispatcher. This decouples alarm delivery from alarm generation. It also allows for the use of multiple delivery methods, such as email and wireless. csadmind detects events by sensing changes in the state of the alarm queue. The alarm queue's state changes every time an alarm is placed in the queue. An alarm is queued when an event producer generates an alarm. The following URIs represents these kind of events:

for events:
enp:///ics/eventalarm?calid=*calid*&uid=*uid*&rid=*rid*&aid=*aid*

for todos:

enp:///ics/todoalarm?calid=*calid*&uid=*uid*&rid=*rid*&aid=*aid*

where:

- *calid* is the calendar ID.
- *uid* is the event / todo ID within the calendar.
- *rid* is the recurrence id for a recurring event / todo.
- *aid* is the alarm ID within the event / todo. In case there are multiple alarms, the *aid* identifies the correct alarm.

*csadmind* dequeues the alarms and sends notifications to enpd. enpd then checks to see if anyone is subscribed to this kind of event and sends notifications to csnotifyd for any subscriptions it finds. These three daemons interacting together implement event notification.

# Daemons

iPlanet Calendar Server includes two daemons that communicate to the ENS daemon, enpd:

- csadmind

  csadmind is the publisher that submits notifications to the notification service by sending event alarms to ENS. It manages the iPlanet Calendar Server 5.0 alarm queue. It implements a scheduler, which lets it know when an alarm has to be generated. At such a point, csadmind publishes an event. ENS receives and dispatches the event notification.

To ensure alarm transfer reliability, csadmind requires acknowledgment for certain events or event types. (See “Alarm Transfer Reliability,” on page 82 in this chapter.) csadmind uses Reliable Event Notification Links (RENLs) to accomplish acknowledgment. For more about RENL’s, see the Publisher API description in the “API Overview” later in this section.

- csnotifyd

  csnotifyd is the subscriber that expresses interest in particular events (subscribes), and receives notifications about these subscribed-to events from the event notification service, and sends notice of these events and todos to its clients by email.

  Though the ability to unsubscribe is part of the ENS architecture, csnotifyd does not bother to unsubscribe to events for the following two reasons: there is no need to unsubscribe or resubscribe during normal runtime, and due to the temporary nature of the subscriptions store (it is held in memory), all subscriptions are implicitly unsubscribed when the connection to ENS is shutdown.

  It subscribes to ics://*hostname*/alarm/pop, and therefore receives all alarm notifications for the host, *hostname*. Upon receipt of an alarm notification, csnotifyd generates email messages.

# Alarm Transfer Reliability

To ensure that no alarm ever gets lost, csadmind and csnotifyd use the RENL feature of ENS for certain types of alarms. For these alarms, csadmind requests an end-to-end acknowledgment for each notification it sends, while csnotifyd, after successfully processing it, generates a notification acknowledgment for each RENL alarm notifications it receives.

For these RENL alarms, should the network, the ENS daemon, or csnotifyd fail to handle a notification, csadmind will not receive any acknowledgment, and will not remove the alarm from the alarm queue. The alarm will, therefore, be published again after a timeout.

# Example

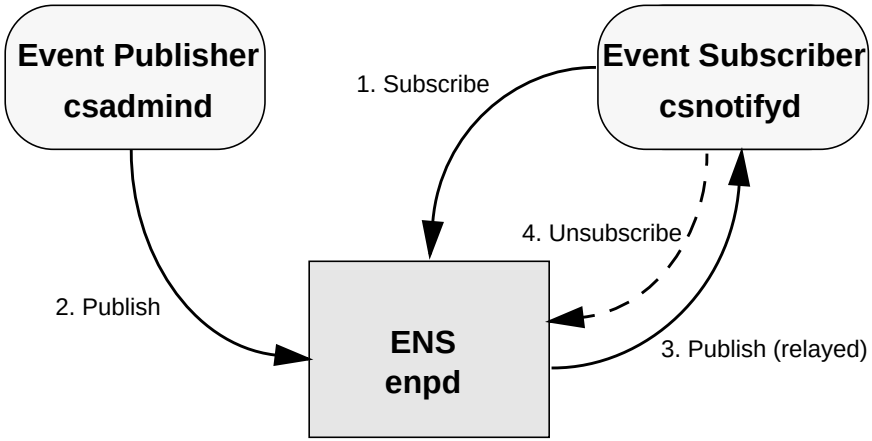
A typical ENS publish and subscribe cycle:

1. The event subscriber, csnotifyd, expresses interest in an event (subscribes).

2. The event publisher, `csadmind`, detects events and sends notification (publishes).
3. ENS publishes the event to the subscriber.
4. The event subscriber cancels interest in the event (unsubscribes). This step happens implicitly when the connection to ENS is shutdown.

Figure 4-2 illustrates this cycle and Table 4-1 provides the narrative for the figure.

**Figure 4-2** Example Event Notification Service Publish and Subscribe Cycle



# API Overview

**Table 4-1** Example Event Notification Service Publish and Subscribe Cycle

| Action | ENS Response |
|---|---|
| 1. `csnotifyd` sends a subscription request to ENS. | ENS stores the subscription in the subscriptions database. |
| 2. `csadmind` sends a notification request to ENS. | ENS queries the subscriptions database for subscriptions matching the notification. |
| 3. `csnotifyd` receives a notification from ENS. | For each matching subscription, access controls are applied and a notification request per qualifying subscription is added to the deliver queue.<br>`csnotifyd` asynchronously dequeues requests from the deliver queue and sends out notifications. |
| 4. Currently, `csnotifyd` does not bother sending cancellation requests to ENS. | Since the subscriptions store is in memory only (not in a database), all subscriptions are implicitly unsubscribed when the connection to ENS is shutdown. |

There are actually three APIs within ENS:

- Publisher API

  A publisher sends notification of a subscribed-to event to the subscriber. Optionally, the application may request acknowledgment of receipt of the notification. To do this, a Reliable Event Notification Link (RENL) is necessary. An RENL has a publisher, a subscriber, and a unique ID, which identify notifications that are subject to acknowledgment. The publisher informs the application of the receipt of an acknowledgment by invoking the `end2end_ack` callback passed to `publish_a`.

- Subscriber API

  A subscriber is a client to the notification service which expresses interest in particular events. When the notification service receives a notification about one of these events from a publisher, it relays the notification to the subscriber.

  A subscriber may also unsubscribe, which cancels an active subscription.

  To enable an RENL, the subscriber declares its existence to ENS, which then transparently generates notification acknowledgment on behalf of the subscriber application. The subscriber may revoke the RENL at any time.

- Publish and Subscribe Dispatcher API

  When an asynchronous publisher is used, ENS needs to borrow threads from a thread pool in order to invoke callbacks. The application can either choose to create its own thread pool and pass it to ENS, or it can let ENS create and manage its own thread pool. In either case, ENS creates and uses a dispatcher object to instantiate the dispatcher used (`pas_dispatcher_t`).

  GDisp (`libasync`) is the dispatcher supported for iPlanet Calendar Server 5.0.

# Publisher API Functions

The Publisher API consists of one definition and nine functions, as listed in Table 4-2.

**Table 4-2** ENS Publisher API Function

| Definition/Function | Description |
|---|---|
| `publisher_t` | Definition for a publisher. |
| `publisher_cb_t` | Generic callback function acknowledging an asynchronous call. |
| `publisher_new_a` | Creates a new asynchronous publisher. |
| `publisher_new_s` | Creates a new synchronous publisher. |
| `publish_a` | Sends an asynchronous notification to the notification service. |
| `publish_s` | Sends a synchronous notification to the notification service. |
| `publisher_delete` | Terminates a publish session. |
| `publisher_get_subscriber` | Creates a subscriber using the publisher's credentials. |
| `renl_create_publisher` | Creates an RENL, which enables the invocation of `end2end_ack`. |
| `renl_cancel_publisher` | Cancels an RENL. |

# Subscriber API Functions

The Subscriber API includes two definitions and ten functions, as listed in Table 4-3.

**Table 4-3** ENS Subscriber API Function

| Definition/Function | Description |
|---|---|
| `subscriber_t` | Definition of a subscriber. |

**Table 4-3** ENS Subscriber API Function *(Continued)*

| Definition/Function | Description |
|---|---|
| `subscription_t` | Definition of a subscription. |
| `subscriber_cb_t` | Generic callback function acknowledging an asynchronous call. |
| `subscriber_notify_cb_t` | Synchronous callback; called upon receipt of a notification. |
| `subscriber_new_a` | Creates a new asynchronous subscriber. |
| `subscriber_new_s` | Creates a new synchronous subscriber. |
| `subscribe_a` | Establishes an asynchronous subscription. |
| `unsubscribe_a` | Cancels an asynchronous subscription. |
| `subscriber_delete` | Terminates a subscriber. |
| `subscriber_get_publisher` | Creates a publisher using the subscriber's credentials. |
| `renl_create_subscriber` | Creates the subscription part of the RENL. |
| `renl_cancel_subscriber` | Cancels an RENL. |

# Publish and Subscribe Dispatcher API Functions

The Publish and Subscribe Dispatcher API includes one definition and four functions, as listed in Table 4-4.

**Table 4-4** ENS Publish and Subscribe Dispatcher API Functions

| Definition/Function | Description |
|---|---|
| `pas_dispatcher_t` | Definition of a publish and subscribe dispatcher. |
| `pas_dispatcher_new` | Creates a dispatcher. |
| `pas_dispatcher_delete` | Destroys a dispatcher created with `pas_dispatcher_new`. |
| `pas_dispatch` | Starts the dispatch loop of an event notification environment. |
| `pas_shutdown` | Stops the dispatch loop on an event notification environment started with `pas_dispatch`. |

# Building and Running Custom Applications

To assist you in building your own custom publisher and subscriber applications, the product includes sample code. (For a listing of the sample code, see “Sample Code,” on page 108 in Chapter 5, “Event Notification Service API Reference”.) This section tells you where to find the sample code, where the APIs’ include (header) files are located, and where the libraries are that you need to build and run your custom programs:

- Location of Sample Code
- Location of Include Files
- Dynamically Linked / Shared Libraries
- Your Runtime Library Path Variable

## Location of Sample Code

The product includes four simple sample programs to help you get started. The code for these samples resides in the following directory:

```
/opt/SUNWics5/cal/csapi/samples/ens
```

## Location of Include Files

The include (header) files for the publisher and subscriber APIs are: `publisher.h`, `suscriber.h`, and `pasdisp.h` (publish and subscribe dispatcher). They are located in the CSAPI `include` directory. The default `include` path is:

```
/opt/SUNWics5/cal/csapi/include
```

## Dynamically Linked/Shared Libraries

In addition, your custom code must be linked with the dynamically linked library `libens`, which implements the publisher and subscriber APIs. On some platforms all the dependencies of `libens` must be provided as part of the link directive. These dependencies, in order, are:

1. libgap
2. libcyrus

3. libyasr
4. libasync
5. libnspr3
6. libplsd4
7. libplc3

iPlanet Calendar Server uses these libraries; therefore, they are located in the server's `bin` directory. The default `libens` path is:

`/opt/SUNWics5/cal/bin`

**NOTE** For NT, in order to build publisher and subscriber applications, you also need the archive files (.lib files) corresponding to all the earlier mentioned libraries. These are located in the CSAPI library directory, `lib`. The default `lib` path is:

*drive*`:\Program Files\iPlanet\cal\csapi\lib`

# Your Runtime Library Path Variable

In order for your custom programs to find the necessary runtime libraries, which are located in the `/opt/SUNWics5/cal/bin` directory, make sure your environment's runtime library path variable includes this directory. The name of the variable is platform dependent:

- For SunOS, and Linux: LD_LIBRARY_PATH.
- For NT: PATH.
- For HPUX: SHLIB_PATH

Chapter 5

# Event Notification Service API Reference

This chapter details the ENS API; it is divided into four main sections:

- Publisher API
- Subscriber API
- Publish and Subscribe Dispatcher API
- Sample Code

## Publisher API Functions List

This chapter includes a description of the following Publisher functions, listed in Table 5-1:

**Table 5-1** ENS Publisher API Functions List

| Definition/Function | Description |
|---|---|
| `publisher_t` | Definition for a publisher. |
| `publisher_cb_t` | Generic callback function acknowledging an asynchronous call. |
| `publisher_new_a` | Creates a new asynchronous publisher. |
| `publisher_new_s` | Creates a new synchronous publisher. |
| `publish_a` | Sends an asynchronous notification to the notification service. |
| `publish_s` | Sends a synchronous notification to the notification service. |
| `publisher_delete` | Terminates a publish session. |
| `publisher_get_subscriber` | Creates a subscriber using the publisher's credentials. |

**Table 5-1** ENS Publisher API Functions List *(Continued)*

| | |
|---|---|
| `renl_create_publisher` | Creates an RENL, which enables the invocation of `end2end_ack`. |
| `renl_cancel_publisher` | Cancels an RENL. |

# Subscriber API Functions List

This chapter includes a description of following Subscriber functions, listed in Table 5-2:

**Table 5-2** ENS Subscriber API Functions List

| Definition/Function | Description |
|---|---|
| `subscriber_t` | Definition of a subscriber. |
| `subscription_t` | Definition of a subscription. |
| `subscriber_cb_t` | Generic callback function acknowledging an asynchronous call. |
| `subscriber_notify_cb_t` | Synchronous callback; called upon receipt of a notification. |
| `subscriber_new_a` | Creates a new asynchronous subscriber. |
| `subscriber_new_s` | Creates a new synchronous subscriber. |
| `subscribe_a` | Establishes an asynchronous subscription. |
| `unsubscribe_a` | Cancels an asynchronous subscription. |
| `subscriber_delete` | Terminates a subscriber. |
| `subscriber_get_publisher` | Creates a publisher using the subscriber's credentials. |
| `renl_create_subscriber` | Creates the subscription part of the RENL. |
| `renl_cancel_subscriber` | Cancels an RENL. |

# Publish and Subscribe Dispatcher Functions List

This chapter includes a description of the following Publish and Subscribe Dispatcher functions, listed in Table 5-3:

**Table 5-3** ENS Publish and Subscribe Dispatcher Functions List

| Definition/Function | Description |
|---|---|
| `pas_dispatcher_t` | Definition of a publish and subscribe dispatcher. |

**Table 5-3** ENS Publish and Subscribe Dispatcher Functions List *(Continued)*

| | |
|---|---|
| `pas_dispatcher_new` | Creates a dispatcher. |
| `pas_dispatcher_delete` | Destroys a dispatcher created with `pas_dispatcher_new`. |
| `pas_dispatch` | Starts the dispatch loop of an event notification environment. |
| `pas_shutdown` | Stops the dispatch loop on an event notification environment started with `pas_dispatch`. |

# Publisher API

The Publisher API consists of one definition and nine functions:

- `publisher_t`
- `publisher_cb_t`
- `publisher_new_a`
- `publisher_new_s`
- `publish_a`
- `publish_s`
- `publisher_delete`
- `publisher_get_subscriber`
- `renl_create_publisher`
- `renl_cancel_publisher`

## publisher_t

**Purpose.**
A publisher.

**Syntax**
```
typedef struct enc_struct publisher_t;
```

**Parameters**
None.

**Returns**
Nothing.

# publisher_cb_t

**Purpose.**
Generic callback function invoked by ENS to acknowledge an asynchronous call.

**Syntax**
```
typedef void (*publisher_cb_t) (void *arg, int rc, void *data);
```

**Parameters**

| | |
|---|---|
| `arg` | Context variable passed by the caller. |
| `rc` | The return code. |
| `data` | For an open, contains a newly created context. |

**Returns**
Nothing.

# publisher_new_a

**Purpose**
Creates a new asynchronous publisher.

**Syntax**
```
void publisher_new_a (pas_dispatcher_t *disp,
                      void *worker,
                      const char *host,
                      unsigned short port,
                      publisher_cb_t cbdone,
                      void *cbarg);
```

**Parameters**

| | |
|---|---|
| `disp` | P&S thread pool context returned by `pas_dispatcher_new`. |
| `worker` | Application worker. If not `NULL`, grouped with existing workers created by ENS to service this publisher session. Used to prevent multiple threads from accessing the publisher data at the same time. |
| `host` | Notification server host name. |
| `port` | Notification server port. |
| `cbdone` | The callback invoked when the publisher has been successfully create, or could not be created.<br>There are three Parameters to `cbdone`:<br>• `cbarg`<br>The first argument.<br>• A status code.<br>If non-zero, the publisher could not be created; value specifies cause of the failure.<br>• The new active publisher. |
| `cbarg` | First argument of `cbdone`. |

**Returns**

Nothing. It passes the new active publisher as third argument of `cbdone` callback.

# publisher_new_s

**Purpose**

Creates a new synchronous publisher.

**Syntax**

```
publisher_t *publisher_new_s (pas_dispatcher_t *disp,
                              void *worker,
                              const char *host,
                              unsigned short port);
```

### Parameters

| | |
|---|---|
| `disp` | P&S thread pool context returned by `pas_dispatcher_new` |
| `worker` | Application worker. If not `NULL`, grouped with existing workers created by ENS to service this publisher session. Used to prevent multiple threads from accessing the publisher data at the same time. |
| `host` | Notification server host name. |
| `port` | Notification server port. |

### Returns

A new active publisher (`publisher_t`).

## publish_a

### Purpose

Sends an asynchronous notification to the notification service.

### Syntax

```
void publish_a (publisher_t *publisher,
                const char *event_ref,
                const char *data,
                unsigned int datalen,
                publisher_cb_t cbdone,
                publisher_cb_t end2end_ack,
                void *cbarg,
                unsigned long timeout);
```

### Parameters

| | |
|---|---|
| `publisher_t` | The active publisher. |
| `event_ref` | The event reference. This is a URI identifying the modified resource. |
| `data` | The event data. The body of the notification message. It must be MIME object. It is opaque to the notification service, which merely relays it to the events' subscriber. |
| `datalen` | The length in bytes of the data. |
| `cbdone` | The callback invoked when the data has been accepted or deemed unacceptable by the notification service. What makes a notification acceptable depends on the protocol used. The protocol may choose to use the transport acknowledgment (TCP) or use its own acknowledgment response mechanism. |
| `end2end_ack` | The callback function invoked after acknowledgment from the consumer peer (in an RENL) has been received. Used only in the context of an RENL. |
| `cbarg` | The first argument of `cbdone` or `end2end_ack` when invoked. |
| `timeout` | The length of time to wait for an RENL to complete. |

### Returns

Nothing.

## publish_s

### Purpose

Sends a synchronous notification to the notification service.

### Syntax

```
int publish_s (publisher_t *publisher,
               const char *event_ref,
               const char *data,
               unsigned int datalen);
```

**Parameters**

| | |
|---|---|
| `publisher` | The active publisher. |
| `event_ref` | The event reference. This is a URI identifying the modified resource. |
| `data` | The event data. The body of the notification message. It must be a MIME object. It is opaque to the notification service, which relays it to the events' subscriber. |
| `datalen` | The length in bytes of the data. |

**Returns**

Zero if successful; a failure code if unsuccessful. If an RENL, the call does not return until the consumer has completely processed the notification and has successfully acknowledged it.

## publisher_delete

**Purpose**

Terminates a publish session.

**Syntax**

```
void publisher_delete (publisher_t *publisher);
```

**Parameters**

| | |
|---|---|
| `publisher` | The publisher to delete. |

**Returns**

Nothing.

## publisher_get_subscriber

**Purpose**

Creates a subscriber using the credentials of the publisher.

**Syntax**

```
struct subscriber_struct * publisher_get_subscriber(publisher_t
*publisher);
```

### Parameters

| | |
|---|---|
| `publisher` | The publisher whose credentials are used to create the subscriber. |

### Returns

The subscriber, or `NULL` if the creation failed. If the creation failed, use the subscriber_new to create the subscriber.

## renl_create_publisher

### Purpose

Declares an RENL, which enables the end2end_ack invocation. After this call returns, the end2end_ack argument is invoked when an acknowledgment notification matching the specified publisher and subscriber is received.

### Syntax

```
void renl_create_publisher (publisher_t *publisher,
                            const char *renl_id,
                            const char *subscriber,
                            publisher_cb_t cbdone,
                            void *cbarg);
```

### Parameters

| | |
|---|---|
| `publisher` | The active publisher. |
| `renl_id` | The unique RENL identifier. This allows two peers to be able to set up multiple RENL's between them. |
| `subscriber` | The authenticated identity of the peer. |
| `cbdone` | The callback invoked when the RENL is established. |
| `cbarg` | The first argument of cbdone, when invoked. |

### Returns

Nothing.

### renl_cancel_publisher

#### Purpose

This cancels an RENL. This does not prevent more notifications being sent, but should a client acknowledgment be received, the end2end_ack argument of publish will no longer be invoked. All RENL's are automatically destroyed when the publisher is deleted. Therefore, this function does not need to be called to free RENL-related memory before deleting a publisher.

#### Syntax

```
void renl_cancel_publisher (renl_t *renl);
```

#### Parameters

| | |
|---|---|
| `renl` | The RENL to cancel. |

#### Returns

Nothing.

# Subscriber API

The Subscriber API includes two definitions and ten functions:

- `subscriber_t`
- `subscription_t`
- `subscriber_cb_t`
- `subscriber_notify_cb_t`
- `subscriber_new_a`
- `subscriber_new_s`
- `subscribe_a`
- `unsubscribe_a`
- `subscriber_delete`
- `subscriber_get_publisher`
- `renl_create_subscriber`
- `renl_cancel_subscriber`

## subscriber_t

**Purpose**
A subscriber.

**Syntax**
```
typedef struct enc_struct subscriber_t;
```

**Parameters**
None.

**Returns**
Nothing.

## subscription_t

**Purpose**
A subscription.

**Syntax**
```
typedef struct subscription_struct subscription_t;
```

**Parameters**
None.

**Returns**
Nothing.

## subscriber_cb_t

**Purpose**
Generic callback function invoked by ENS to acknowledge an asynchronous call.

**Syntax**
```
typedef void (*subscriber_cb_t) (void *arg,
                                 int rc,
                                 void *data);
```

### Parameters

| | |
|---|---|
| `arg` | Context variable passed by the caller. |
| `rc` | The return code. |
| `data` | For an open, contains a newly created context. |

### Returns

Nothing

## subscriber_notify_cb_t

### Purpose

Subscriber callback; called upon receipt of a notification.

### Syntax

```
typedef void (*subscriber_notify_cb_t) (void *arg,
                                        char *event,
                                        char *data,
                                        int datalen);
```

### Parameters

| | |
|---|---|
| `arg` | Context pointer passed to subscribe (`notify_arg`). |
| `event` | The event reference (URI). The notification event reference matches the subscription, but may contain additional information called event attributes, such as a `uid`. |
| `data` | The body of the notification. A MIME object. |
| `datalen` | Length of the data. |

### Returns

Zero if successful, non-zero otherwise.

## subscriber_new_a

### Purpose

Creates a new asynchronous subscriber.

### Syntax

```
void subscriber_new_a (pas_dispatcher_t *disp,
                       void *worker,
                       const char *host,
                       unsigned short port,
                       subscriber_cb_t cbdone,
                       void *cbarg);
```

### Parameters

| | |
|---|---|
| `disp` | Thread dispatcher context returned by `pas_dispatcher_new`. |
| `worker` | Application worker. If not NULL, grouped with existing workers created by ENS to service this subscriber session. Used to prevent multiple threads from accessing the subscriber data at the same time. Only usable if the caller creates and dispatches the GDisp context. |
| `host` | Notification server host name or IP address. |
| `port` | Subscription service port number. |
| `cbdone` | The callback invoked when the subscriber session becomes active and subscriptions can be issued.<br><br>There are three parameters to `cbdone`:<br><br>• `cbarg`<br>The first argument.<br><br>• A status code.<br>If non-zero, the subscriber could not be created; value specifies cause of the failure.<br><br>• The new active subscriber (`subscriber_t`). |
| `cbarg` | First argument of `cbdone`. |

### Returns

Nothing. It passes the new active subscriber as third argument of `cbdone` callback.

## subscriber_new_s

### Purpose

Creates a new synchronous subscriber.

### Syntax

```
subscriber_t *subscriber_new_s (pas_dispatcher_t *disp,
                                const char *host,
                                unsigned short port);
```

### Parameters

| | |
|---|---|
| `disp` | Publish and subscribe dispatcher returned by `pas_dispatcher_new`. |
| `worker` | Application worker. If not NULL, grouped with existing workers created by ENS to service this publisher session. Used to prevent multiple threads from accessing the publisher data at the same time. Only usable if the caller creates and dispatches the GDisp context. |
| `host` | Notification server host name or IP address. |
| `port` | Subscription service port number. |

### Returns

A new active subscriber (`subscriber_t`).

## subscribe_a

### Purpose

Establishes an asynchronous subscription.

### Syntax

```
void subscribe_a (subscriber_t *subscriber,
                  const char *event_ref,
                  subscriber_notify_cb_t notify_cb,
                  void *notify_arg,
                  subscriber_cb_t cbdone,
                  void *cbarg):
```

**Parameters**

| | |
|---|---|
| `subscriber` | The subscriber. |
| `event_ref` | The event reference. This is a URI identifying the event's source. |
| `notify_cb` | The callback invoked upon receipt of a notification matching this subscription. |
| `notify_arg` | The first argument of `notify_arg`. May be called at any time, by any thread, while the subscription is still active. |
| `cbdone` | Called when an unsubscribe completes. It has three Parameters:<br>• `cbarg` (see below).<br>• Status code.<br>• A pointer to an opaque subscription object. |
| `cbarg` | The first argument of `cbdone`. |

**Returns**

Nothing.

## unsubscribe_a

**Purpose**

Cancels an asynchronous subscription.

**Syntax**

```
void unsubscribe_a (subscriber_t *subscriber,
                    subscription_t *subscription,
                    subscriber_cb_t cbdone,
                    void *cbarg);
```

**Parameters**

| | |
|---|---|
| `subscriber` | The disappearing subscriber. |
| `subscription` | The subscription to cancel. |
| `cbdone` | Called when an unsubscribe completes. It has three parameters:<br>• `cbarg` (see below).<br>• Status code.<br>• A pointer to an opaque subscription object. |
| `cbarg` | The first argument of `cbdone`. |

**Returns**
Nothing.

## subscriber_delete

**Purpose**
Terminates a subscriber.

**Syntax**
```
void subscriber_delete (subscriber_t *subscriber);
```

**Parameters**

| | |
|---|---|
| `subscriber` | The subscriber to delete. |

**Returns.**
Nothing

## subscriber_get_publisher

**Purpose**
Creates a publisher, using the credentials of the subscriber.

**Syntax**
```
struct publisher_struct *subscriber_get_publisher (subscriber_t
*subscriber);
```

### Parameters

| | |
|---|---|
| `subscriber` | The subscriber whose credentials are used to create the publisher. |

### Returns

The publisher, or `NULL` if creation failed. In case the creation fails, use the `publisher_new`.

## renl_create_subscriber

### Purpose

Creates the subscription part of an RENL.

### Syntax

```
renl_t *renl_create_subscriber (subscription_t *subscription,
                                const char *renl_id,
                                const char *publisher);
```

### Parameters

| | |
|---|---|
| subscription | The subscription. |
| renl_id | The unique RENL identifier. This allows two peers to be able to set up multiple RENL's between them. |
| publisher | The authenticated identity of the peer. |

### Returns

The opaque RENL object.

## renl_cancel_subscriber

### Purpose

This cancels an RENL. It does not cancel a subscription. It tells ENS not to acknowledge any more notifications received for this subscription. It destroys the RENL object, the application may no longer use this RENL. All RENLs are automatically destroyed when the subscription is canceled. Therefore, this function does not need to be called to free RENL-related memory before deleting a subscriber.

### Syntax

```
void renl_cancel_subscriber (renl_t *renl);
```

**Parameters**

| | |
|---|---|
| `renl` | The RENL to cancel. |

**Returns**
Nothing.

# Publish and Subscribe Dispatcher API

The Publish and Subscribe Dispatcher API includes one definition and four functions:

- pas_dispatcher_t
- pas_dispatcher_new
- pas_dispatcher_delete
- pas_dispatch
- pas_shutdown

**NOTE** The only thread dispatcher supported is GDisp (libasync).

## pas_dispatcher_t

**Purpose**
A publish and subscribe dispatcher.

**Syntax**

```
typedef struct pas_dispatcher_struct pas_dispatcher_t;
```

**Parameters**
None.

**Returns**
Nothing.

# pas_dispatcher_new

### Purpose
Creates or advertises a dispatcher

### Syntax
```
pas_dispatcher_t *pas_dispatcher_new (void *disp);
```

### Parameters

| | |
|---|---|
| `dispcx` | The dispatcher context. If `NULL`, to start dispatching notifications, the application must call `pas_dispatch`.<br><br>If not `NULL`, the dispatcher is a `libasync` dispatcher. |

### Returns
The dispatcher to use when creating publishers or subscribers (`pas_dispatcher_t`).

# pas_dispatcher_delete

### Purpose
Destroys a dispatcher created with `pas_dispatcher_new`.

### Syntax
```
void pas_dispatcher_delete (pas_dispatcher_t *disp);
```

### Parameters

| | |
|---|---|
| `disp` | The event notification client environment. |

### Returns
Nothing.

# pas_dispatch

### Purpose
Starts the dispatch loop of an event notification environment. It has no effect if the application uses its own thread pool.

### Syntax
```
void pas_dispatch (pas_dispatcher_t *disp);
```

**Parameters**

| | |
|---|---|
| `disp` | The new dispatcher. |

**Returns**
Nothing.

## pas_shutdown

**Purpose**
Stops the dispatch loop of an event notification environment started with `pas_dispatch`. It has no effect if an application-provided dispatcher was passed to `pas_dispatcher_new`.

**Syntax**

```
void pas_shutdown (pas_dispatcher_t *disp);
```

**Parameters**

| | |
|---|---|
| `disp` | The dispatcher context to shutdown. |

**Returns**
Nothing

# Sample Code

iPlanet Calendar Server ships with a complete ENS implementation. If you wish to customize it, you may use these APIs to do so. The following four code samples, a simple publisher and subscriber pair, and a reliable publisher and subscriber pair, illustrate how to use the API. The sample code is provided with the product in the following directory:

`/opt/SUNWics5/cal/csapi/samples/ens`

## Simple Publisher and Subscriber

This sample code pair establishes a simple interactive asynchronous publisher and subscriber.

## Publisher Code Sample

```
/*
 * Copyright 2000 by Sun Microsystems, Inc.
 * All rights reserved
 *
 * apub : simple interactive asynchronous publisher using
 *
 * Syntax:
 *   apub host port
 */

#include <stdlib.h>
#include <stdio.h>

#include "pasdisp.h"
#include "publisher.h"

static pas_dispatcher_t *disp = NULL;
static publisher_t *_publisher = NULL;
static int _shutdown = 0;

static void _read_stdin();

static void _exit_usage()
{
   printf("\nUsage:\napub host port\n");
   exit(5);
}

static void _exit_error(const char *msg)
{
   printf("%s\n", msg);
   exit(1);
}

static void _call_shutdown()
{
   _shutdown = 1;
   pas_shutdown(disp);
}

static void _open_ack(void *arg, int rc, void *enc)
{
   _publisher = (publisher_t *)enc;
   (void *)arg;
```

```
    if (!_publisher)
    {
        printf("Failed to create publisher with status %d\n", rc);
        _call_shutdown();
        return;
    }

    _read_stdin();

    return;
}

static void _publish_ack(void *arg, int rc, void *ignored)
{
    (void *)ignored;

    free(arg);

    if (rc != 0)
    {

        printf("Publish failed with status %d\n", rc);

        _call_shutdown();

        return;
    }

    _read_stdin();

    return;
}

static void _read_stdin()
{

    static char input[1024];

    printf("apub> ");

    fflush(stdout);

    while (!_shutdown)
    {

        if ( !fgets(input, sizeof(input), stdin) )
        {
            continue;

        } else {
            char *message;
            unsigned int message_len;

            input[strlen(input) - 1] = 0; /* Strip off the \n */
```

```
            if (*input == '.' && input[1] == 0)
            {
                publisher_delete(_publisher);
                _call_shutdown();
                break;
            }

            message = strdup(input);
            message_len = strlen(message);
            publish(_publisher, "enp://yoyo.com/xyz",message,
                    message_len,
                    _publish_ack, NULL, (void *)message, 0);
            return;
        }
    }
    return;
}

main(int argc, char **argv)
{
    unsigned short port = 7997;
    char host[256];

    if (argc < 2) _exit_usage();

    if (*(argv[1]) == '0')
    {
        strcpy(host, "127.0.0.1");
    } else {
        strcpy(host, argv[1]);
    }

    if (argc > 2)
    {
        port = (unsigned short)atoi(argv[2]);
    }

    disp = pas_dispatcher_new(NULL);
    if (disp == NULL) _exit_error("Can't create publisher");

    publisher_new_a(disp, NULL, host, port, _open_ack, disp);

    pas_dispatch(disp);

    _shutdown = 1;

    pas_dispatcher_delete(disp);

    exit(0);
}
```

## Subscriber Code Sample

```
/*
 * Copyright 1997 by Sun Microsystems, Inc.
 * All rights reserved
 *
 * asub : example asynchronous subscriber
 *
 * Syntax:
 *   asub host port
 */

#include <stdlib.h>
#include <stdio.h>

#include "pasdisp.h"
#include "subscriber.h"

static pas_dispatcher_t *disp = NULL;
static subscriber_t *_subscriber = NULL;
static subscription_t *_subscription = NULL;
static renl_t *_renl = NULL;

static void _exit_usage()
{
   printf("\nUsage:\nasub host port\n");
   exit(5);
}

static void _exit_error(const char *msg)
{
   printf("%s\n", msg);
   exit(1);
}

static void _subscribe_ack(void *arg, int rc, void *subscription)
{
   (void)arg;

   if (!rc)
   {
      _subscription = subscription;
      printf("Subscription successful\n");
   } else {
      printf("Subscription failed - status %d\n", rc);
      pas_shutdown(disp);
   }
}
```

```
static void _unsubscribe_ack(void *arg, int rc, void *ignored)
{
   (void *)ignored;
   (void *)arg;

   if (rc != 0)
   {
      printf("Unsubscribe failed - status %d\n", rc);
   }

   subscriber_delete(_subscriber);
   pas_shutdown(disp);
}

static int _handle_notify(void *arg, char *url, char *str, int len)
{
   (void *)arg;
   printf("[%s] %.*s\n", url, len, (str) ? str : "(null)");
   return 0;
}

static void _open_ack(void *arg, int rc, void *enc)
{
   _subscriber = (subscriber_t *)enc;

   (void *)arg;
   if (rc)
   {
      printf("Failed to create subscriber with status %d\n", rc);
      pas_shutdown(disp);
      return;
   }

   subscribe(_subscriber, "enp://yoyo.com/xyz",
             _handle_notify, NULL,
             _subscribe_ack, NULL);

   return;
}

static void _unsubscribe(int sig)
{
   (int)sig;
   unsubscribe(_subscriber, _subscription, _unsubscribe_ack, NULL);
}

main(int argc, char **argv)
{
   unsigned short port = 7997;
   char host[256];
```

```
    if (argc < 2) _exit_usage();
    if (*(argv[1]) == '0')
    {
        strcpy(host, "127.0.0.1");
    } else {
        strcpy(host, argv[1]);
    }
    if (argc > 2)
    {
        port = (unsigned short)atoi(argv[2]);
    }

    disp = pas_dispatcher_new(NULL);
    if (disp == NULL) _exit_error("Can't create publisher");

    subscriber_new_a(disp, NULL, host, port, _open_ack, NULL);

    pas_dispatch(disp);

    pas_dispatcher_delete(disp);

    exit(0);
}
```

# Reliable Publisher and Subscriber

This sample code pair establishes a reliable asynchronous publisher and subscriber.

## Reliable Publisher Sample

```
/*
 * Copyright 2000 by Sun Microsystems, Inc.
 * All rights reserved
 *
 * rpub : simple *reliable* interactive asynchronous publisher.
 *  It is designed to be used in combination with rsub,
 *  the reliable subscriber.
 *
 * Syntax:
 *   rpub host port
 */

#include <stdlib.h>
#include <stdio.h>

#include "pasdisp.h"
#include "publisher.h"
```

```
static pas_dispatcher_t *disp = NULL;
static publisher_t *_publisher = NULL;
static int _shutdown = 0;
static renl_t *_renl;

static void _read_stdin();

static void _exit_usage()
{

   printf("\nUsage:\nrpub host port\n");

   exit(5);
}

static void _exit_error(const char *msg)
{
   printf("%s\n", msg);
   exit(1);
}

static void _call_shutdown()
{
   _shutdown = 1;
   pas_shutdown(disp);
}

static void _renl_create_cb(void *arg, int rc, void *ignored)
{
   (void *)arg;
   (void *)ignored;

   if (!_publisher)
   {
      printf("Failed to create RENL - status %d\n", rc);
      _call_shutdown();
      return;
   }

   _read_stdin();

   return;
}

static void _publisher_new_cb(void *arg, int rc, void *enc)
{
   _publisher = (publisher_t *)enc;

   (void *)arg;
```

```
    if (!_publisher)
    {
        printf("Failed to create publisher - status %d\n", rc);
        _call_shutdown();
        return;
    }

    renl_create_publisher(_publisher, "renl_id", NULL,
                          _renl_create_cb,NULL);

    return;
}

static void _recv_ack(void *arg, int rc, void *ignored)
{
    (void *)ignored;

    if (rc < 0)
    {
        printf("Acknowledgment Timeout\n");
    } else if ( rc == 0) {
        printf("Acknowledgment Received\n");
    }
    fflush (stdout);

    _read_stdin();

    free(arg);

    return;
}

static void _read_stdin()
{
    static char input[1024];

    printf("rpub> ");
    fflush(stdout);
    while (!_shutdown)
    {
        if ( !fgets(input, sizeof(input), stdin) )
        {
            continue;
        } else {
            char *message;
            unsigned int message_len;

            input[strlen(input) - 1] = 0; /* Strip off the \n */
```

```
            if (*input == '.' && input[1] == 0)
            {
                publisher_delete(_publisher);
                _call_shutdown();
                break;
            }

            message = strdup(input);
            message_len = strlen(message);

            /* five seconds timeout */
            publish(_publisher, "enp://yoyo.com/xyz",
                    message, message_len,
                    NULL, _recv_ack, message, 5000);

            return;
        }
    }
    return;
}

main(int argc, char **argv)
{
    unsigned short port = 7997;
    char host[256];

    if (argc < 2) _exit_usage();
    if (*(argv[1]) == '0')
    {
        strcpy(host, "127.0.0.1");
    } else {
        strcpy(host, argv[1]);
    }
    if (argc > 2)
    {
        port = (unsigned short)atoi(argv[2]);
    }

    disp = pas_dispatcher_new(NULL);
    if (disp == NULL) _exit_error("Can't create publisher");

    publisher_new_a(disp, NULL, host, port, _publisher_new_cb,
                    NULL);

    pas_dispatch(disp);

    _shutdown = 1;

    pas_dispatcher_delete(disp);

    exit(0);
}
```

## Reliable Subscriber Sample

```
/*
 * Copyright 1997 by Sun Microsystems, Inc.
 * All rights reserved
 *
 * asub : example asynchronous subscriber
 *
 * Syntax:
 *   asub host port
 */

#include <stdlib.h>
#include <stdio.h>

#include "pasdisp.h"
#include "subscriber.h"

static pas_dispatcher_t *disp = NULL;
static subscriber_t *_subscriber = NULL;
static subscription_t *_subscription = NULL;
static renl_t *_renl = NULL;

static void _exit_usage()
{
   printf("\nUsage:\nasub host port\n");
   exit(5);
}

static void _exit_error(const char *msg)
{
   printf("%s\n", msg);
   exit(1);
}

static void _subscribe_ack(void *arg, int rc, void *subscription)
{
   (void)arg;

   if (!rc)
   {
      _subscription = subscription;
      printf("Subscription successful\n");
      _renl = renl_create_subscriber(_subscription, "renl_id",
NULL);
   } else {
      printf("Subscription failed - status %d\n", rc)
      pas_shutdown(disp);
   }
}
```

```
static void _unsubscribe_ack(void *arg, int rc, void *ignored)
{
   (void *)ignored;
   (void *)arg;

   if (rc != 0)
   {
      printf("Unsubscribe failed - status %d\n", rc);
   }

   subscriber_delete(_subscriber);
   pas_shutdown(disp);
}

static int _handle_notify(void *arg, char *url, char *str, int len)
{
   (void *)arg;
   printf("[%s] %.*s\n", url, len, (str) ? str : "(null)");
   return 0;
}

static void _open_ack(void *arg, int rc, void *enc)
{
   _subscriber = (subscriber_t *)enc;

   (void *)arg;
   if (rc)
   {
      printf("Failed to create subscriber with status %d\n", rc);
      pas_shutdown(disp);
      return;
   }

   subscribe(_subscriber, "enp://yoyo.com/xyz",_handle_notify,
             NULL,_subscribe_ack, NULL);

   return;
}

static void _unsubscribe(int sig)
{
   (int)sig;
   unsubscribe(_subscriber, _subscription, _unsubscribe_ack, NULL);
}

main(int argc, char **argv)
{
   unsigned short port = 7997;
   char host[256];
```

```
    if (argc < 2) _exit_usage();
    if (*(argv[1]) == '0')
    {
       strcpy(host, "127.0.0.1");
    } else {
       strcpy(host, argv[1]);
    }
    if (argc > 2)
    {
       port = (unsigned short)atoi(argv[2]);
    }
    disp = pas_dispatcher_new(NULL);
    if (disp == NULL) _exit_error("Can't create publisher");
    subscriber_new_a(disp, NULL, host, port, _open_ack, NULL);
    pas_dispatch(disp);
    pas_dispatcher_delete(disp);
    exit(0);
}
```

Chapter 6

# Proxy Authentication SDK Overview

This chapter describes the iPlanet Calendar Server 5.0 Proxy Authentication SDK (authSDK). It addresses the following topics:

- Who Will Use the authSDK?
- What Is the authSDK?
- Architecture
- Functions Overview

## Who Will Use the authSDK?

Programmers, whose installation has a portal service, can use authSDK to integrate the portal with Calendar Server. When a portal system authenticates a user, authSDK functions notify Calendar Server, which then allows the user access to various services without reauthentication.

## What Is the authSDK?

The authSDK consists of a DLL / shared-object that exports five functions.

The install package includes the following, located in *server root*`/bin/authsdk`:

- `libicsexp10.so/DLL`**.** The SDK library.
- `expapi.h`. Header file for API users.

# Architecture

The authSDK is pretty simple. It consists of initialization, lookup, and cleanup. Additionally, one other function,CEXP_SetHttpPort, allows the authSDK to use a non-standard port, and another,CEXP_GetVersion, gets the authSDK version number should you need to contact customer or technical support. For a complete description of the API functions, see Chapter 7, “Proxy Authentication SDK Reference”.

## Initialization

Call CEXP_Init for initialization. If you pass it LDAP information, it initializes an LDAP connection used during the lookup phase for discovering on which calendar server the user resides. This connection is set up for a threaded environment. All threads share this connection, but since the locking is done in the LDAP, it is fast enough in most environments. The connection is kept open so there is no setup / teardown cost per lookup.

Other things set up by initialization include the programmer supplied attribute to use for matching the lookup user name in LDAP queries.

## Lookup

Use the lookup function, CEXP_GenerateLoginURL, when you want to generate a new session for a user. Lookup performs no authentication. It simply uses the user name and IP address to generate an entry in the session table for them and returns a URL associated with that session. If you pass the hostname of a calendar server, the function will contact that server to generate the session. If not, it will query the LDAP server to determine the host. In order for it to generate a session without actually authenticating the user, you must provide the proxy admin’s ID and password.

## Cleanup

If you are in a threaded environment and you wish to cleanup resources such as memory, open LDAP connections, etc., use CEXP_Shutdown.

# Functions Overview

There are five functions in the SDK, as listed in Table 6-1.

**Table 6-1** Proxy Authentication SDK Functions

| Function | Description |
|---|---|
| `CEXP_GenerateLoginURL` | Generates a URL with the valid session ID. |
| `CEXP_GetVersion` | Generates the version ID string. |
| `CEXP_Init` | Initializes the SDK. |
| `CEXP_SetHttpPort` | Specify the port over which you will contact the calendar server. |
| `CEXP_Shutdown` | Performs all shutdown procedures, including freeing memory and shutting down connections. |

# Proxy Authentication SDK Reference

This chapter describes the iPlanet Calendar Server 5.0 Proxy Authentication SDK (authSDK) API. The chapter is divided into two parts: Proxy Authentication SDK Functions, and How to Use the authSDK.

## Proxy Authentication SDK Functions List

There are five authSDK functions:

- `CEXP_GenerateLoginURL`

  Generates a URL with the valid session ID.

- `CEXP_GetVersion`

  Generates the version ID string.

- `CEXP_Init`

  Initializes the SDK.

- `CEXP_SetHttpPort`

  Specifies the port over which you will contact the calendar server.

- `CEXP_Shutdown`

  Performs all shutdown procedures, including freeing memory and shutting down connections.

# Proxy Authentication SDK Functions

The authSDK consists of five functions. They are presented below in alphabetical order, not in order of use. For directions on how to use the functions, see the section, “How to Use the authSDK, in this chapter.

- `CEXP_GenerateLoginURL`
- `CEXP_GetVersion`
- `CEXP_Init`
- `CEXP_SetHttpPort`
- `CEXP_Shutdown`

## CEXP_GenerateLoginURL

### Purpose

Returns a login URL with a valid session ID for a given user.

### Syntax

```
int CEXP_GenerateLoginURL (char * pszUser,
                           char * pszClientAddress,
                           char * pszCalendarHost,
                           char * pszURL);
```

### Parameters

There are four parameters:

| | |
|---|---|
| `pszUser` | A string containing the user name. |
| `pszClientAddress` | A string containing the client host IP-address. |
| `pszCalendarHost` | A string containing the hostname (no IP-address) of the calendar server. |
| `pszURL` | A pointer to a buffer to place the URL |

### Returns

Returns `0` on success, `-1` on failure. On success, the `pszURL` buffer contains a valid URL string.

# CEXP_GetVersion

### Purpose
Gets the version ID string.

### Syntax
```
char * CEXP_GetVersion(void);
```

### Parameters
There are no parameters:

### Returns
A reference to the version ID string.

# CEXP_Init

### Purpose
Initializes the SDK.

### Syntax
```
int CEXP_Init (char * pszLdapHost,
               char * pszLdapMatchAttrib,
               char * pszLdapDN,
               unsigned int iLdapPort,
               char * pszLdapBindUser,
               char * pszLdapBindPass,
               char * pszAdminUser,
               char * pszAdminPassword);
```

### Parameters
There are eight parameters:

| | |
|---|---|
| `pszLdapHost` | A string containing the hostname of the directory server. |
| `pszLdapMatchAttrib` | A string containing the attribute name. Used to match against the user name. |
| `pszLdapDN` | A string containing the base DN to search for user records. “DN”, for Distinguished Name, is a string representation of an LDAP directory entry’s name and location. |
| `iLdapPort` | An integer specifying the directory server’s port number. |
| `pszLdapBindUser` | A string specifying the DN to bind as. |
| `pszLdapBindPass` | A string containing the password for the bind DN. |

| | |
|---|---|
| `pszAdminUser` | A string containing the Calendar Server administrator's LDAP user ID. |
| `pszAdminPassword` | A string containing the Calendar Server administrator's password. |

**Returns**

Returns `0` on success, `-1` on failure.

**Comment**

If the bindDN and password are `NULL`, anonymous searching will be attempted.

## CEXP_SetHttpPort

**Purpose**

Sets the HTTP Port used to contact the calendar server.

**Syntax**

```
void CEXP_SetHttpPort (int iHttpPort);
```

**Parameters**

There is one parameter:

| | |
|---|---|
| `iHttpPort` | An integer specifying the port. |

**Returns**

Nothing.

## CEXP_Shutdown

**Purpose**

Cleans up all global memory, shuts down connections, and other clean-up functions when the user is finished using the SDK.

**Syntax**

```
int CEXP_Shutdown (void);
```

**Parameters**

There are no parameters:

**Returns**

Returns `0` on success, `-1` on failure.

**Comments**

Call this only after all threads using the SDK complete.

# How to Use the authSDK

To implement authSDK in your installation, follow these steps:

1. Link the authSDK to your code.

   To integrate the authSDK into your existing code, simply include the `expapi.h` header file in the calling code and link with the DLL / shared-object. On some platforms you may also be required to link with other system libraries the authSDK requires.

2. Authenticate your user with your portal authentication program.

3. Call `CEXP_Init`.

   This function initializes the authSDK configuration information. This is necessary before any other authSDK function is called.

4. Optionally, call `CEXP_SetHttpPort`.

   By default, the authSDK contacts the standard HTTP port, 80. Use this function to tell the authSDK to contact a non-standard port when connecting to generate a session.

| **CAUTION** | This function is not thread safe and sets a global value. If you want to use it in a threaded environment, you must lock around this call and the `CEXP_GenerateLoginURL` call. |
|---|---|

5. Call `CEXP_GenerateLoginURL`.

   This function generates a session handle for the user and client-ip address. It returns a string, in a buffer you allocate, containing a login URL to be used when connecting to the Calendar Server. The string is a kind of token providing proof of identity. It is given to the client in the form of a cookie or URL via HTTP headers or JavaScript. The client will then connect to Calendar Server, presenting the token as proof of identity.

6. Optionally, call `CEXP_Shutdown`.

Call this function to shutdown and cleanup any resources used by the authSDK. It is not necessary to call this function in some environments (a simple CGI login, for example), but plug-ins using the API may want to reclaim resources and continue running.

## Other Tips

There are a few other things that must be done to assure success in using the AuthSDK:

- The value of `service.http.allowadminproxy` in the ics.conf file must be “yes”.
- The parameter `caladmin`, passed in the `init` method, must have the same value as `service.admin.calmaster.userid` in the `ics.conf` file.
- The parameter `calpass`, passed in the `init` method, must have the same value as `service.admin.calmaster.cred` in the `ics.conf` file.
- The two parameters `caladmin` and `calpass` must be defined in your directory service.
- If your calendar server is not listening on the default port 80, you must use the `SetHttpPort` method with the correct port value.

Chapter 8

# Single Sign-on Authentication

This chapter describes the Single Sign-on authentication mechanism included in iPlanet Calendar Server 5.0. This scheme is independent from any other authentication mechanisms, session management, and resource access control. To use it, your client must support cookies and your server must support HTTP. Single Sign-on does not impact application performance. This scheme does not require a centralized session manager. Applications manage their own sessions and may have different timeout and revocation policies.

This chapter has the following topics:

- What is Single Sign-on?
- Limitations of Single Sign-on
- Process Flow
- Implementation Requirements
- Single Sign-on Example
- Issues

## What is Single Sign-on?

Single Sign-on allows a user to sign on once and use multiple applications. These applications form a circle of trust that use each other's cookies as verification of authority so that the user does not have to sign on to each application separately.

Each application can have its own verification interface, if necessary. However, each verification authority must store a cookie that is understood by the other applications' verification authority routines. See "Cookie Information" later in this chapter.

# Limitations of Single Sign-on

Despite the benefits listed earlier, there are some limitations that might deter you from using Single Sign-on:

- Applications need to implement the verification protocol. See “Issues” later in this chapter.
- This scheme is not suitable for shared machine situations.
- Applications need to be in the same domain and have access to each other’s Single Sign-on verification URL.
- The browser must be restarted for the user to switch to a different identity.

  This is necessary because each browser session can support only one user ID.
- The client must support cookies.

# Process Flow

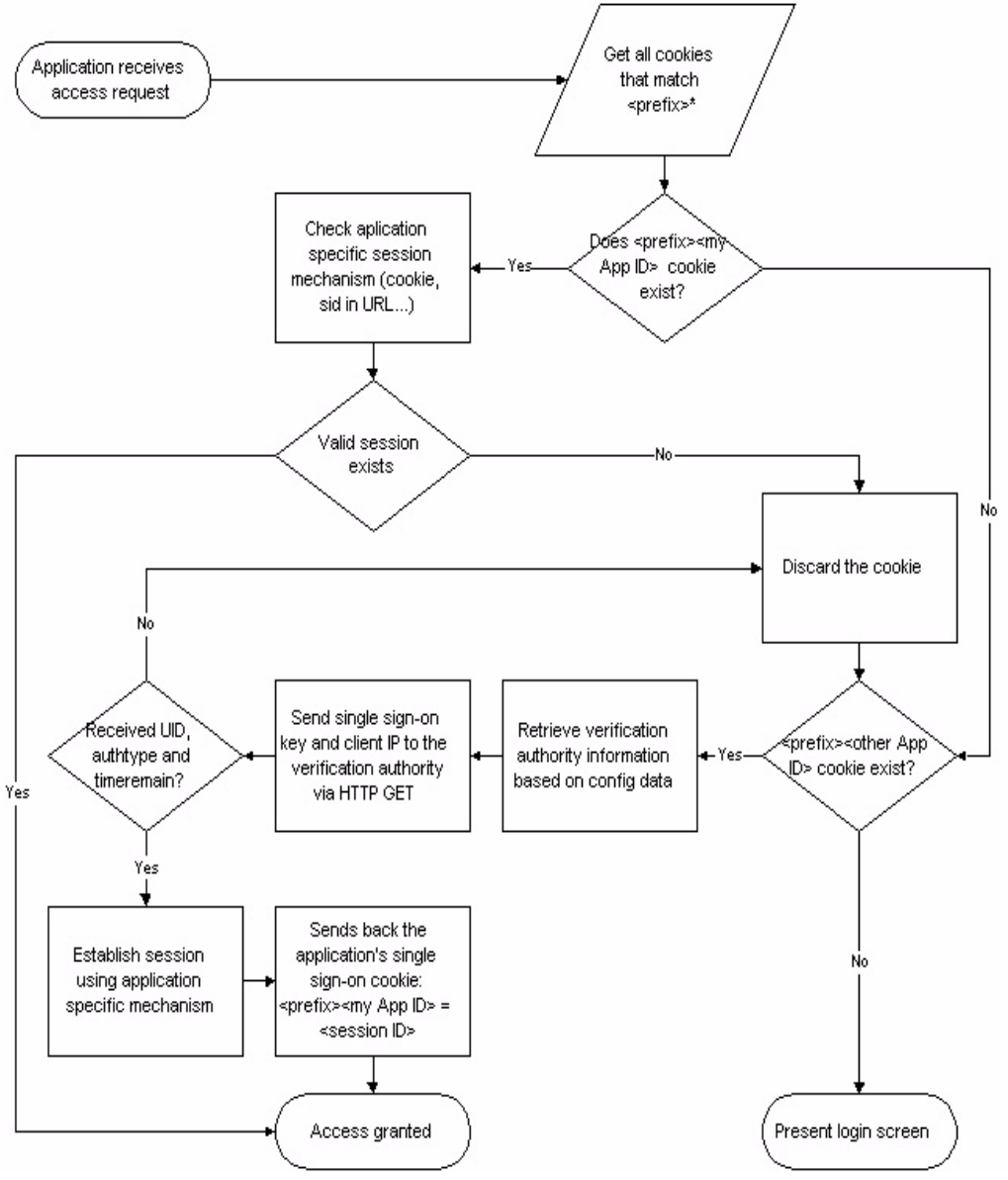
As diagrammed in the figure that follows, the flow starts with an application receiving access requests. Since a circle of trusted applications share the same prefix, the program fetches all cookies with the matching prefix.

If the application does not find one of its own cookies, it checks to see if there are cookies available from any of the other trusted applications. Once it finds a valid cookie, the program uses the information from these cookies to verify authority and establish a session. When granting access, it sends back its own application specific cookie.

If there are no cookies available with this circle’s prefix, or the available cookies are invalid, the program displays a login screen to the user.

Figure 8-1 illustrates Single Sign-on process flow.

**Figure 8-1** Single Sign-on Process Flow



# Implementation Requirements

In order to participate in this single sign-on scheme, all applications in the circle need to:

- Be able to read and write cookies.
- Support additional configuration parameters:
  - List of trusted applications.
  - Single Sign-off.
  - Prefix string.
- Implement the Single Sign-on verification protocol over HTTP.
  - Request:

    The request contains the Single Sign-on cookie in the header and has the client IP address as a parameter of the URL:

    `GET` *verificationurl* `client=`*clientIP*`HTTP/1.0`
  - Response when key is valid:

    A text/plain ASCII response with each key-value pair taking a separate line. `timeremaining` is optional.

    ```
    fquid=user id@fully qualified domain name
    authtype=plaintext | cert | ...
    timeremaining=[time left before this session time out]
    ```
  - Response when key is not valid:

    A text/plain ASCII message: `Error: user does not have a valid session.`
- Edit the configuration file, `ics.conf`, to set the configuration data for your trusted circle. See the "iPlanet Calendar Server Administration Guide" for details on the configuration file. Single Sign-on configuration parameters start with the prefix `sso`.

## Cookie Information

The cookie format is as follows:

*configurable prefix - application installation unique identifier=single sign-on key*

## Other recommended settings:

- Domain set to `.domain name`.
- Path set to `/`.
- Expires field not set.

## Trusted Applications Record

While the storage and format of additional configuration parameters are application specific, each record should contain:

- A unique identifier for this particular trusted application installation.
- A fully qualified domain name or IP address of the machine hosting the application.
- A URL that supports the Single Sign-on verification protocol.

  Example: `http://domain.com/VerifySSO?`

## Single Sign-off Parameter

The single sign-off parameter is a boolean indicating whether this application performs single sign-off. The default should be `true`.

- If `true`, the application removes all single sign-on cookies matching the trusted circle prefix when the user logs off.
- If `false`, the application removes only its own single sign-on cookie when the user logs off.

## Prefix String

This is the common string shared by all applications in the trust circle. You can form multiple circles of trust within a domain by choosing separate prefixes.

# Single Sign-on Example

This example illustrates the steps involved in completing a single sign-on cycle.

# The Example

The participating applications use `ssogrp1` as the prefix for this circle of trust. Trusted applications in this circle are: WebMail, WebCal, and HRapp.

1. John checks his mail.

   WebMail receives the login request from `jsmith` running on `208.12.60.3`. WebMail looks for cookies that have `ssogrp1` as part of their name. There is no such cookie. WebMail prompts the user to authenticate and then sends back a cookie with the prefix string, *`ikey-`*, plus this WebMail installation's unique identifier (`3fr7d`), and the single sign-on key (`l3khj5l3k9ldh`), which happens to be the session ID.

   The cookie returned is: `ssogrp13fr7d=l3khj5l3k9ldh`.

2. John then checks his calendar.

   - WebCal receives the login request from `jsmith` running on 208.12.60.3. The only `ssogrp1*` cookie WebCal found was `ssogrp13fr7d=l3khj5l3k9ldh`. WebCal retrieves the following entries in its config file:

     `3fr7d.verificationurl=http://`*`webmail host`*`/VerifySSO?`

   - WebCal issues HTTP GET:

     `GET http://`*`webmail host`*`/VerifySSO?client=208.12.60.3 HTTP/1.0`

     `Cookie: ssogrp13fr7d=l3khj5l3k9ldh`

   - WebMail verifies this is a valid key and returns:

     ```
     fquid=jsmith@example.com
     authtype=plaintext
     timeremaining=1000
     ```

   - WebCal generates a single sign-on key, `a97ads64`, for John and sends back the cookie:`ssogrp1lkj87f=a97ads64`.

3. John surfs the net for a while and then wants to check his calendar again.

   - WebCal receives log in request from `jsmith` running on `208.12.60.3`.
   - WebCal found several `ssogrp1` cookies. One of them matches WebCal's unique application ID (`lkj87f`).

     It sends back the cookie: `ssogrp1lkj87f=a97ads64`.

- Since WebCal uses session ID as the single sign-on key, it retrieves session `a97ads64` from its session database, verifies that the session is still valid and then allows the user to proceed.

4. John needs to access his company's HR application which requires certificate authentication.

   The HR application was developed in-house and was modified to support the verification protocol.

   - HRapp receives the login request from `jsmith` running on `208.12.60.3`.

     Since it requires certificate authentication, it asks the client to send the certificate.

   - HRapp does not check for the `ssogrp1*` cookies.
   - HRapp generates a session `B53P997KD` for John and sends back a cookie: `ssogrp1adf38=lka79jy5d3l3r`.

     This cookie will allow other participating applications to use HRapp as a verification authority.

5. John checks his mail again and decides to log off.

   - WebMail receives the logoff request from `jsmith` running on `208.12.60.3`.
   - In addition to invalidating John's WebMail session, it also removes all `ssogrp1*` cookies.

6. Now if John wants to access any of the applications again, he will need to log in again.

# Configuration Parameters for the Example

- Prefix: `sso.appprefix="ssogrp1"`
- Single sign-off boolean: `sso.singlesignoff="true"`
- Application information:
  - WebMail:

    `appid=3fr7d`

    `3fr7d.ip=198.93.96.111`

    `3fr7d.verificationurl=http://`*`webmail host`*`/VerifySSO?`

- WebCal:

  ```
  lkj87f.ip=198.93.78.103
  lkj87f.verificationurl=http://webcal host/VerifySSO?
  ```

- HRapp:

  ```
  adf38.ip=198.93.70.8
  adf38.verificationurl=http://hr host/VerifySSO?
  ```

# Issues

The section discusses the issues, assumptions, and requirements for four areas:

- Security
- Management
- Scalability
- Performance

## Security

- The assumption is that it is sufficiently difficult to guess the correct combination of single sign-on key and client IP address, so it's not necessary to require authentication for the verification protocol.
- If the client connections come from a proxy then the single sign-on key is the only defense. So it is extremely important that the key is very difficult to predict.
- It's the application's responsibility to generate secure single sign-on keys.
- It's also helpful to configure the application to have shorter session timeout so each single sign-on key does not stay valid for a long time.
- HTTPs could be used to protect the communications. If an application requires client certificate authentication then it should always request the certificate from the client. However, it can still act as a verification authority for others.

## Management

- Every application must maintain the trusted application list. Each application could, potentially, be using a different mechanism to store and manage that list.
- Only the application that generated it can revoke a single sign-on key. Administrators can not easily turn off a user's access to all applications.

## Scalability

Cookie number limitations:

- 300 total cookies
- 20 cookies per server or domain. Completely specified hosts and domains count as separate entities, and each has a 20 cookie limitation.

## Performance

- The verification request to other applications only happens once. Once an application session is established, the verification protocol is no longer involved.
- If the browser contains many stale cookies (cookies with an invalid single sign-on key), then logging into a new application may take a long time.

  This should not happen very often for two reasons:

  - Each application should remove its own cookie when users log out. (If the application performs single sign-off, then it removes all single sign-on cookies that have the applicable prefix.)
  - Applications should leave the `expires` field blank so the cookie is not persistent across browser sessions.

Chapter 9

# Web Calendar Access Protocol (WCAP) Overview

This chapter describes the Web Calendar Access Protocol (WCAP) 2.0, which is a high level command-based protocol that clients use to communicate with iPlanet Calendar Server 5.0. This chapter has the following sections:

- Introduction
- What's New in This Version
- Command Overview
- Command Formats

## Introduction

The default Client UI protocol for iPlanet Calendar Server 5.0 is now SHTML (see the "iPlanet Calendar Express Customization Guide" for a detailed description of the new protocol). However, you may choose to use WCAP for compatibility with your previously customized Calendar Express clients. WCAP continues to support the iPlanet Calendar Server 2.0 protocol.

WCAP is a command based system consisting of client requests and server responses for transmitting calendaring data. WCAP 2.0 returns calendaring data via HTTP. In most cases, iPlanet Calendar Server 5.0 receives data through URL-encoded arguments.

WCAP 2.0 returns output in an HTTP message. The returned content body of the HTTP messages can consist of calendar data in the following formats:

- `text/calendar` format (iCalendar).
- `text/xml` format.

This XML format follows the iCalendar DTD.

- `text/js` with embedded JavaScript objects.

  This was the default for the iPlanet Calendar Server 2.0 user interface. It has been deprecated for iPlanet Calendar Server 5.0 in favor of the .shtml approach.

WCAP 2.0 commands consist of four general categories of usage:

- User Configuration Information
- Web Calendaring Data
- Communication-sending for group scheduling
- Miscellaneous commands

# What's New in This Version

The following four commands are new to this version:

- `check_id`

  Allows administrators to check if session is still valid.

- `fetchcomponents_by_alarmrange`

  Queries for components that have alarms to trigger over a specific time period.

- `fetchcomponents_by_attendee_error`

  Queries for components that had errors while sending group scheduling messages.

- `get_freebusy`

  Returns calendar freebusy time.

Four new parameters were added to existing commands:

- `acl`

  Added to `set_calprops` to record ACE strings for access control. For more information on access control entries, see Chapter 10, "WCAP Commands.

- `alarmAudio`

  Added to `storeevents` and `storetodos` to enable audio reminders.

- `alarmFlashing`

  Added to `storeevents` and `storetodos` to enable flashing reminders.

- `alarmPopup`

  Added to `storeevents` and `storetodos` to enable popup dialog reminders.

- `compstate`

  For all fetch commands, `compstate` allows you to fetch by component state.
  `userid`

  For `get_userprefs` and `set_userprefs`, administrators can retrieve and set any user's preferences.

- `convertCalid`

  In `set_userprefs`, to allow setting the subscribed list of calendars (`icsSubscribed`), and the subscribed list of groups (`icsSet`).

- `method`

  For group scheduling, the `storeevents` and `storetodos` commands contain an additional parameter, `method`, which specifies the ITIP method.

One command has changed fundamentally in the way it works:

- `set_calprops` now works in *update* mode, rather than *replace* mode as in iPlanet Calendar Server 2.x. This means that you no longer have to specify every parameter in the command. You need only specify the parameters you wish to change. All other properties will remain unchanged by your update.

The existing parameter `tzid` has been added to the following existing commands:

- `deleteevents_by_id`
- `deletetodos_by_id`
- `fetchcomponents_by_alarmrange`
- `fetchcomponents_by_lastmod`
- `fetchcomponents_by_range`
- `fetchevents_by_id`
- `fetchtodos_by_id`

Existing parameter `tzid` has changed in two existing commands:

- `storeevents`

- `storetodos`

See also “New in iPlanet Calendar Server 5.0,” on page 162 in Chapter 10, “WCAP Commands”.

# Command Overview

Table 9-1 describes the high level list of commands supported in WCAP 2.0. For a detailed description of each command, see Chapter 10, “WCAP Commands”.

**Table 9-1** WCAP Command Overview

| WCAP Command | Description |
|---|---|
| `addlink` | Add event links from one calendar to another. |
| `change_password` | Change the user’s password. |
| `createcalendar` | Create a new calendar. |
| `deletecalendar` | Delete an existing calendar. |
| `deletecomponents_by_range` | Delete both events and todos in a calendar(s) over a specific time period. |
| `deleteevents_by_id` | Delete events given a specific calid and uid / recurrence-ID pair. |
| `deleteevents_by_range` | Delete events in a calendar(s) over a specific time period. |
| `deletetodos_by_id` | Delete todos given a specified calid and userid / recurrence-ID pair. |
| `deletetodos_by_range` | Deletes todos in a calendar(s) over a specific time period. |
| `export` | Exports a calendar to a file. |
| fetchcomponents_by_alarmrange | Queries for components that have alarms to trigger over a specific time period. |
| `fetchcomponents_by_attendee_error` | Queries for components that had errors while sending group scheduling messages. |
| `fetchcomponents_by_lastmod` | Queries for components that have changed, during the specified time range. |
| `fetchcomponents_by_range` | Queries for components over a specific time period, with filtering attributes. |
| `fetchevents_by_id` | Queries for one or more events by a unique identifier (UID, Recurrence ID, modifier). |

**Table 9-1** WCAP Command Overview *(Continued)*

| WCAP Command | Description |
|---|---|
| fetchtodos_by_id | Queries for one or more todos by a unique identifier (UID, Recurrence ID, modifier). |
| get_all_timezones | Returns all the timezones the server supports. |
| get_calprops | Returns calendar properties. |
| get_freebusy | Returns calendar freebusy time. |
| get_guids | Returns a set of random UIDs. |
| get_userprefs | Returns user preferences and some server settings. |
| import | Imports a calendar from a file to a user's calendar. |
| login | Authenticates a user and redirects to first HTML view. |
| logout | Terminates the current user's session and return to login screen. |
| ping | Administrator only: Pings the calendar server. |
| search_calprops | Searches for a calendar with the specified parameter values. |
| set_calprops | Sets calendar properties. |
| set_userprefs | Sets user preferences. |
| storeevents | Stores events that are specified in application / urlencoded manner. For storing an even by passing properties in a URL. |
| storetodos | Stores todos that are specified in the application / urlencoded manner. |
| upload_file | Uploads a file to the server. |
| version | Returns the WCAP version that the server supports. |
| write_file | Writes a file to the database. |

# Session Identifiers

For many WCAP commands, you must specify the session identifier (`id`) that is returned by the `login` command. The session identifier ensures that data is accessible only to authenticated 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. iPlanet 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. See Chapter 3, “CSAPI Reference”, for the section “csIAuthentication,” on page 48.

# Command Formats

The plug-in architecture of iPlanet 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.

The command protocol uses HTTP, and follows the standards defined by the WC3 URL specifications.

WCAP in iPlanet Calendar Server consists of JavaScript objects formatted as XML or iCalendar, communicated as HTML documents over HTTP on both the client and server side. The client supports version Internet Explorer 4.0 and above, and Navigator 4.05 and above.

## Client Request Formats

Clients submit command requests to the iPlanet Calendar Server in either Universal Resource Identifier (URI) data format, or with an HTML form.

| Command Format | Description |
|---|---|
| URI | Requests from client submitted using standard URI syntax. |
| HTML Form - `urlencoded` | Requests from client submitted using native JavaScript objects. |
| HTML Form - `text/xml` | Requests from client submitted using JavaScript objects formatted as XML. |
| HTML Form - `text/calendar` | Requests from client submitted using JavaScript objects formatted as iCalendar. |

### URI Format

Use the following format to submit a URI request:

`http://webcalenderserver/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:

`http://webcalendarserver/fetchcomponents_by_range.wcap?uid=abc;def;gh%3bi;jkl`

**NOTE** This is a change from WCAP 1.0 used in iPlanet Calendar Server 2.0. Previously, the quoted-printable equivalent for a semicolon was `=3B`.

See also “Encoded Characters,” on page 157, in Chapter 10, “WCAP Commands”.

### HTML Form

Submit a form with `method=[GET|POST]` and `action=`*command* (where *command* is the command to execute). Parameters need to be formatted as specified in the encoding.

### Client Side Event Notification

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

The above commands when used with HTTP `GET` are simply for data retrieval.

The above commands when used with HTTP `POST` are for data modifications (including creation / deletion).

## Server Response Formats

iPlanet Calendar Server responds to client requests by serving HTML containing JavaScript objects.You can configure a response format preference for a server, a user, or an individual request.

The client may request output in one of three different formats:

| Response Format | Description |
|---|---|
| HTML with JavaScript objects - text/js | Responses are in HTML and contain JavaScript objects. This is the default output format for all WCAP commands. |
| HTML with JavaScript objects - text/calendar | Responses are in HTML, containing JavaScript formatted as iCalendar objects. |
| HTML with JavaScript objects - text/xml | Responses are in HTML, containing JavaScript formatted as XML objects |

Chapter 10

# WCAP Commands

This chapter is divided into two parts: topics of common interest, and the individual WCAP commands.

## Common topics

- Access Control Entries
- Choosing a Different Language or Character Set
- Deleting Recurring Components
- Encoded Characters
- Error Handling
- Formatting of Time, Strings, Parameters, Etc.
- Freebusy Access
- New in iPlanet Calendar Server 5.0
- Output Format
- Recurrence Handling

## Commands

Table 10-1 lists the WCAP commands in alphabetical order.

**Table 10-1** WCAP Commands

| | |
|---|---|
| `addlink` | Add event links from one calendar to another. |
| `change_password` | Change the user's password. |

**Table 10-1** WCAP Commands *(Continued)*

| | |
|---|---|
| `check_id` | Check to see if the session is still valid. |
| `createcalendar` | Create a new calendar. |
| `deletecalendar` | Delete an existing calendar. |
| `deletecomponents_by_range` | Delete both events and todos in a calendar(s) over a specific time period. |
| `deleteevents_by_id` | Delete events given a specific calid and uid / recurrence-ID pair. |
| `deleteevents_by_range` | Delete events in a calendar(s) over a specific time period. |
| `deletetodos_by_id` | Delete todos given a specified calid and userid / recurrence-ID pair. |
| `deletetodos_by_range` | Deletes todos in a calendar(s) over a specific time period. |
| `export` | Exports a calendar to a file. |
| `fetchcomponents_by_alarmrange` | Queries for components over a specific time period that have alarms to trigger. |
| `fetchcomponents_by_attendee_error` | Queries for components that had errors while sending group scheduling messages. |
| `fetchcomponents_by_lastmod` | Queries for components that have changed, during the specified time range. |
| `fetchcomponents_by_range` | Queries for components over a specific time period, with filtering attributes. |
| `fetchevents_by_id` | Queries for one or more events by a unique identifier (UID, Recurrence ID, modifier). |
| `fetchtodos_by_id` | Queries for one or more todos by a unique identifier (UID, Recurrence ID, modifier). |
| `get_all_timezones` | Returns all the timezones the server supports. |
| `get_calprops` | Returns calendar properties. |
| `get_freebusy` | Returns calendar freebusy time. |
| `get_guids` | Returns a set of random UIDs. |
| `get_userprefs` | Returns user preferences and some server settings. |
| `import` | Imports a calendar from a file to a user's calendar. |
| `login` | Authenticates a user and redirects to first HTML view. |
| `logout` | Terminates the current user's session and return to login screen. |

**Table 10-1** WCAP Commands *(Continued)*

| | |
|---|---|
| `ping` | Administrator only: Pings the calendar server. |
| `search_calprops` | Searches for a calendar with the specified parameter values. |
| `set_calprops` | Sets calendar properties. |
| `set_userprefs` | Sets user preferences. |
| `storeevents` | Stores events that are specified in application / urlencoded manner. For storing an even by passing properties in a URL. |
| `storetodos` | Stores todos that are specified in the application / urlencoded manner. |
| `upload_file` | Uploads a file to the server. |
| `version` | Returns the WCAP version that the server supports. |
| `write_file` | Writes a file to the database. |

# Common Topics

This section discusses topics of common interest that apply to one or more commands. The topics are listed in alphabetical order.

## Access Control Entries

Access Control Entries (ACE strings) determine access control for calendars. There may be multiple ACE strings that apply to a calendar. Collectively, all the ACE strings that apply are called an Access Control List (ACL). As the system searches the ACL list, the first ACE encountered that either grants of denies access will be used. Thus, the ordering of an ACL is significant. ACE strings should be ordered such that the more specific ones appear before the more general ones.

Some access is “built-in”. For example, primary owners have access to everything in their calendars. The system does not need to perform access control checks for primary owners accessing their own calendars.

The `set_calprops` command uses the `acl` parameter to facilitate storing of ACE strings to a calendar. The `acl` parameter is a semicolon-separated list of ACE strings. You may set the default `acl` in the `ics.conf` file by changing the `calstore.calendar.default.acl` preference, or by using the `cscal` command line utility. See the “iPlanet Calendar Server Administration Guide” for further information on configuration settings.

Here is an example of an ACE string:

```
jdoe^c^wd^g
```

The string has four elements separated by three “^” characters. The four elements are:

1. The first element of an ACE tells who the ACE applies to.

   This could be an individual user (specified by user ID), a domain, or a class-type of user. There are four types of classes for users:

   - All users, represented by the string “@”.
   - Primary owners of a calendar, represented by the string “@@p”.
   - Owners of a calendar, represented by the string “@@o”.
   - Non-owners of a calendar, represented by the string “@@n”.

2. The second element of an ACE indicates what the ACE applies to.

The ACE can be applied to:

- The entire calendar.

  Applies to both components and calendar properties. To indicate the entire calendar, pass in the value a.

- The components of the calendar only.

  Applies to calendar components (i.e events / todos). To indicate just the components, pass in the value c.

- The calendar properties of the calendar only.

  Applies to calendar properties (i.e display name, ownerlist). To indicate calendar properties only, pass in the value p.

**3.** The third element of an ACE indicates what access values the ACE applies to.

Multiple values may be specified at the same time. To do this, the caller must pass in a string to indicate which bits to check.

Table 10-2 lists the Access Control characters used in ACE strings. The third element contains a string with one or more of the Access Control characters.

**Table 10-2** Access Control Characters

| Access Control Characters | Description |
|---|---|
| c | Grants the user act-on-behalf-of cancel access. With cancel access, a user has the right to cancel components to which attendees have been invited on behalf of the calendar's primary owner. |
| d | Grants the user delete access. |
| e | Grants the user act-on-behalf-of reply access. This grants a user the rights to accept or decline invitations on behalf of the calendar's primary owner. |
| f | Grants the user free-busy access. |
| i | Grants the user act-on-behalf-of invite access. This grants a user the right to create and modify components in which other attendees have been invited on behalf of the calendar's primary owner |
| r | Grants the user read access. |
| s | Grants the user schedule access. This means that requests can be made, replies will be accepted, and other ITIP scheduling interactions will be honored. |

**Table 10-2** Access Control Characters

| | |
|---|---|
| w | Grants the user write access. This includes adding new items, deleting items, and modifying existing items. |

For example, to grant read access, the value `r` is passed in. To grant write and delete access, the value `wd` is passed in.

**4.** The fourth element of an ACE indicates whether to grant or deny access.

The ACE can either grant or deny access.

- To grant access, set the value to `g`.
- To deny access, set the value to `d`.

### *ACE Summary*

Here is a quick summary of the order of an ACE:

who ^ flags ^ how ^ grant

Where:

- who = A string, type (str).
- flags = One of the characters `c`, `p`, or `a`.
- how = An access-string composed of one or more of the access control characters described earlier in Table 10-2.
- grant = One of the characters `g`, or `d`

### *Extended Examples*

Here are some examples of circumstances and how the ACE would be set in the `acl` parameter for `jdoe`'s calendar:

- To grant `john` read access to both components and calendar properties (`acl=john a r g`), and to grant `susan` write and delete access to components only (`acl=susan c wd g`), the entire command is:

```
set_calprops.wcap?id=${SESSIONID}&calid=jdoe&acl=john^a^r^g;
susan^c^wd^g
```

- To grant all users in a domain schedule, freebusy, and read access to components only (@*domainname* c sfr g), to grant owners to write and delete access to components only (@@o c wd g), to grant owners self-admin, schedule, freebusy, and read access to both components and calendar properties (@@o a zsfr g), to deny susan all access to both components and calendar properties (susan a zsfdwr d), and to grant read access to all users (@ c r g), the entire command is:

  ```
  set_calprops.wcap?id=${SESSIONID}&calid=jdoe&acl=@domainname^c^s
  fr^g;@@o^c^wd^g;@@o^a^zsfr^g;susan^a^zfsdwr^d;@^c^r^g
  ```

**NOTE** An administrator can override the access control of all WCAP commands if he is logged in as administrator and the server configuration preference `service.admin.calmaster.overrides.accesscontrol` is set to "yes" in the `ics.conf` file.

# Choosing a Different Language or Character Set

To insert a request for data to be returned in a language other than the system default, set either the `lang` or `charset` parameter. Note that the system default for language is now a server preference that you set in the ics.conf file. See the "Administrator's Guide" for details. The `login` command uses only the `lang` parameter.

For the `set_calprops` command, in most cases, specifying the `lang` parameter is enough. However, it may be necessary, in some instances, to use the `charset` parameter instead of the `lang` parameter. For example, if the user wants the requested data returned in a specified character set, then the user must specify it using `charset`. One possible `charset` value is: `iso-8859-1`. For more information on formatting specifications, see the RFCs referenced in "Formatting of Time, Strings, Parameters, Etc.," on page 161 in this chapter.

Please note that when the user requests data in iCalendar or XML format, data always returns in UTF-8 format, per the RFC specification. Setting `charset` will not change this.

Here is a list of the valid `lang` values:

| | |
|---|---|
| de | German |
| en | English (the default) |
| es | Spanish |

| | |
|---|---|
| fr | French |
| it | Italian |
| ja | Japanese |
| ko | Korean |
| ru | Russian |
| sv | Swedish |
| zh_CN | Chinese/Simplified Chinese |
| zh_TW | Taiwanese |

**NOTE** This does not mean that all of these languages are currently supported by the server. Please check with your iPlanet representative to find out which languages are currently supported by the server.

# Deleting Recurring Components

When you delete a component, you can (and sometimes must) specify whether or not it is recurring, and, if it is, whether to delete the recurrences as well as the original event or todo.

Use the mod parameter to choose which delete option you want:

**Table 10-3** mod Parameter Delete Options

| Value | Option |
|---|---|
| 1 | Delete this instance only. |
| 2 | Delete this and all future recurrences. |
| 3 | Delete this and all prior recurrences. |
| 4 | Delete all instances. |

For each option, the user must pass in the `rid` parameter which specifies the recurrence ID of the event, and the type of deletion to do.

To delete just the single instance of the event, the `mod` parameter should be set to `1`. For example, this URL would delete just the event that occurs on the date March 1, 2000 11:22:33 AM GMT.

`http://webcalendarserver/deleteevents_by_id.wcap?id=23423423434abc&calid=jdoe&uid=001&rid=20000301T112233Z&mod=1`

To delete the event and all future instances of the event, the `mod` parameter should be set to `2`. For example, this URL would delete the event that occurs on the date March 1, 2000 11:22:33 AM GMT and all future instances of this event (`uid 001`).

`http://jdoe/deleteevents_by_id.wcap?id=23423423434abc&calid=jdoe&uid=001&rid=20000301T112233Z&mod=2`

To delete the event and all prior instances of the event, the `mod` parameter should be set to `3`.

For example, this URL would delete the event that occurs on the date March 1, 200011:22:33 AM GMT and all prior instances of this event (`uid 001`).

`http://jdoe/deleteevents_by_id.wcap?id=23423423434abc&calid=jdoe&uid=001&rid=20000301T112233Z&mod=3`

To delete all instances of the event, the `mod` parameter should be set to `4`. For example, this URL would delete ALL instances of the event (`uid` 001).

`http://jdoe/deleteevents_by_id.wcap?id=23423423434abc&calid=jdoe&uid=001&rid=20000301T112233Z&mod=4`

# Encoded Characters

In the example, the encoded list of parameters for `cal` includes some encoded characters. Here are some examples of encoded characters:

%3D = ’=’

%26 = ’&’

%22 = ’”’

The %XX is the hexadecimal ASCII value of the character. For example, the '&' character is 26 in hex (38 in ASCII).

# Error Handling

Each call to a WCAP command that returns component data (fetch, delete, and store commands) also returns an array and an error string.

## Error String

The error string, `errno`, returns the non-zero error number for the transaction. The value is `0` if the command succeeded.

## Layer Error Number Array.

In addition to the possibility of the transaction failing, it is possible that one or more of the layers may have failed during the transaction. L*ayer* refers to one of the multiple actions or items that are being requested. For example, in `fetch_components_by_range`, you may pass in a list of calendar IDs to fetch from; in this case, each calendar is a *layer*.

The error number is specific to a layer. The index of the layer array maps to the index number of the layer passed in. Therefore, `layer_errno[3]`, refers to the third item you passed in to the fetch command.

An error in one layer does not prevent the other layers from being processed.

There are three layer error arrays:

- `layer_errno`. For fetch commands.
- `delete_layer_errno`. For delete commands.
- `store_layer_errno`. For store commands.

## Layer Count Array.

In addition, for `deleteevents_by_id` and `deletetodos_by_id`, the commands return a second array, `delete_event_count`, or `delete_todo_count` respectively. This array contains the count of successful deletions before the error occurred. This is useful if the components passed have recurrences. As with the layer error number array, the index of the layer count array maps to the index number of the layer passed in to the command.

For example, if you pass in three components to `deleteevents_by_id` and you get an error in the second event, you find the error code in `delete_layer_errno[2]`. You then look in `delete_event_count[2]` to see how many occurrences of the event were successfully deleted before the error occurred.

## Error Codes

Table 10-4 list some of the error codes returned in the error number array.

**Table 10-4** Error Names, Values, and Meanings

| Error Name | Value | Meaning |
|---|---|---|
| LOGOUT | -1 | Logout successful. |
| OK | 0 | Command successful. |
| LOGIN_FAILED | 1 | Login failed, session ID timed out. Invalid session ID |
| LOGIN_OK_DEFAULT_CALENDAR_NOT_FOUND | 2 | `login.wcap` was successful, but the default calendar for this user was not found. A new default calendar set to the userid was created. |
| DELETE_EVENTS_BY_ID_FAILED | 6 | Command failed. |
| SETCALPROPS_FAILED | 8 | Command failed. |
| FETCH_EVENTS_BY_ID_FAILED | 9 | Command failed. |
| CREATECALENDAR_FAILED | 10 | Command failed. |
| DELETECALENDAR_FAILED | 11 | Command failed. |
| ADDLINK_FAILED | 12 | Command failed. |
| FETCHBYDATERANGE_FAILED | 13 | Command failed. |
| STOREEVENTS_FAILED | 14 | Command failed. |
| STORETODOS_FAILED | 15 | Command failed. |
| DELETE_TODOS_BY_ID_FAILED | 16 | Command failed. |
| FETCH_TODOS_BY_ID_FAILED | 17 | Command failed. |
| FETCHCOMPONENTS_FAILED_BAD_TZID | 18 | Command failed to find correct TZID. Applies to `fetchcomponents_by_range`, `fetchevents_by_id`, `fetchtodos_by_id`. |
| SEARCH_CALPROPS_FAILED | 19 | Command failed. |
| GET_CALPROPS_FAILED | 20 | Command failed. |
| DELETECOMPONENTS_BY_RANGE_FAILED | 21 | Command failed. |

**Table 10-4** Error Names, Values, and Meanings *(Continued)*

| Error Name | Value | Meaning |
|---|---|---|
| DELETEEVENTS_BY_RANGE_FAILED | 22 | Command failed. |
| DELETETODOS_BY_RANGE_FAILED | 23 | Command failed. |
| GET_ALL_TIMEZONES_FAILED | 24 | Command failed. |
| CREATECALENDAR_ALREADY_EXISTS_FAILED | 25 | `createcalendar.wcap` failed; calendar with that name already exists in the database. |
| SET_USERPREFS_FAILED | 26 | Command failed. |
| CHANGE_PASSWORD_FAILED | 27 | Command failed. |
| ACCESS_DENIED_TO_CALENDAR | 28 | Command failed; user denied access to a calendar. |
| CALENDAR_DOES_NOT_EXIST | 29 | Command failed; calendar does not exist in the database. |
| ILLEGAL_CALID_NAME | 30 | `createcalendar.wcap` failed; `calid` passed in was invalid. |
| CANNOT_MODIFY_LINKED_EVENTS | 31 | `storeevents.wcap` failed; event to modify was a linked event. |
| CANNOT_MODIFY_LINKED_TODOS | 32 | `storetodos.wcap` failed; todo to modify was a linked todo. |
| CANNOT_SENT_EMAIL | 33 | Command failed; the email notification failed. Usually caused by the server not being properly configured to send email. This can occur in `storeevents`, `storetodos`, `deleteevents_by_id`, `deletetodos_by_id`. |
| CALENDAR_DISABLED | 34 | Command failed; calendar is disabled in the database. |
| WRITE_IMPORT_FAILED | 35 | Import failed when writing files to the server. |
| FETCH_BY_LAST_MODIFIED_FAILED | 36 | Command failed. |
| CAPI_NOT_SUPPORTED | 37 | Failed trying to read from CS&T calendar data. |
| CALID_NOT_SPECIFIED | 38 | Calendar ID was not specified. |
| GET_FREEBUSY_FAILED | 39 | Command failed. |
| STORE_FAILED_DOUBLE_BOOKED | 40 | `storeevents` or `storetodos` failed while attempting to store an event / todo in a time slot that was already filled. Double booking is not allowed. |
| FETCH_BY_ALARM_RANGE_FAILED | 41 | Command failed. |
| FETCH_BY_ATTENDEE_ERROR_FAILED | 42 | Command failed. |

**Table 10-4** Error Names, Values, and Meanings *(Continued)*

| Error Name | Value | Meaning |
|---|---|---|
| ATTENDEE_GROUP_EXPANSION_CLIPPED | 43 | An LDAP group being expanded was too large and exceeded the maximum number allowed in an expansion. The expansion stopped at the specified maximum limit. The maximum limit defaults to 200. To change the maximum limit, set the server configuration preference `calstore.group.attendee.maxsize`. |
| USERPREFS_ACCESS_DENIED | 44 | Either the server does not allow this administrator access to get/modify user preferences, or the requester is not an administrator. |
| NOT_ALLOWED_TO_REQUEST_PUBLISH | 45 | The requester was not an organizer of the event, and, therefore, is not allowed to edit the component using the PUBLISH or REQUEST method. |

# Formatting of Time, Strings, Parameters, Etc.

Find the exact format and definition for all times, strings, parameters, etc. by referring to RFC2445, RFC2446, and RFC2447. Unless otherwise noted, all WCAP commands follow these specifications.

The RFC's may be found at the IETF web site:

- http://www.ietf.org/rfc/rfc2445.txt
- http://www.ietf.org/rfc/rfc2446.txt
- http://www.ietf.org/rfc/rfc2447.txt

**NOTE** While there is no limit on the size of string parameters in WCAP, we recommend keeping the total parameter length to 1024 characters or less.

# Freebusy Access

Freebusy access means that the user may see scheduled time on the calendar but is not allowed to see the event details. Instead, just the word “Busy” appears by the time block. Blocks of time without any scheduled events are listed also, with the word “Free” next to them.

For example, a calendar called `jdoe` has the following events:

| | |
|---|---|
| 100:00-11:00 | first meeting |
| 12:00-1:00 | lunch |
| 3:00-4:00 | second meeting |

Suppose user `john` is given freebusy access to the calendar `jdoe`. The freebusy time for `jdoe` (from 9:00 to 6:00) would be the following:

| | | |
|---|---|---|
| 9-10 | : | Free |
| 10-11 | : | Busy |
| 11-12 | : | Free |
| 12-1 | : | Busy |
| 1-3 | : | Free |
| 3-4 | : | Busy |
| 4-6 | : | Free |

Notice that `john` does not know why the user is busy, but simply knows when the user is busy.

# New in iPlanet Calendar Server 5.0

While the WCAP commands are backwards compatible with 2.x, many changes have been made to them. One very important change is that the `set_calprops` command now works in *update* mode rather than *replace* mode. That is, you no longer have to specify every parameter in the command as in 2.x. Any parameters not specified in your `set_calprops` command remain unchanged.

This version adds several new commands and some new parameters for existing commands.

## New Commands

The new commands are:

- `check_id`

  Allows administrators to check if session is still valid.

- `fetchcomponents_by_alarmrange`

  Queries for components that have alarms to trigger over a specific time period.

- `fetchcomponents_by_attendee_error`

  Queries for components that had errors while sending group scheduling messages.

- `get_freebusy`

  Returns calendar freebusy time.

## New Parameters

The new parameters are:

- `acl`

  Added to `set_calprops` to record ACE strings for access control. See “Access Control Entries,” on page 152.

- `alarmAudio`

  Added to `storeevents` and `storetodos` to enable audio reminders.

- `alarmFlashing`

  Added to `storeevents` and `storetodos` to enable flashing reminders.

- `alarmPopup`

  Added to `storeevents` and `storetodos` to enable popup dialog reminders.

- `compstate`

  All fetch commands, except `fetchcomponents_by_attendee_error`, have the ability to fetch by component state, using the parameter `compstate`. Since all components are usually fetched with these commands, use this parameter to limit the type of components fetched.

If the parameter is not specified, the default value is ALL.

Table 10-5 lists component state values. A component state pertains either to the attendee or the organizer.

**Table 10-5** Component State Values for compstate Parameter

| Value | Organizer/ Attendee | Comment |
|---|---|---|
| REPLY-DECLINED | Attendee | Attendee has declined the meeting. |
| REPLY-ACCEPTED | Attendee | Attendee has accepted the meeting. |
| REQUEST-COMPLETED | Organizer | Meeting sent by organizer. All attendees' replies received. |
| REQUEST_NEEDS-ACTION | Attendee | Attendee has not taken action on the meeting yet. |
| REQUEST-NEEDSNOACTION | Attendee | Organizer does not require the attendee to reply. |
| REQUEST-PENDING | Organizer | Organizer's meeting sent. Not all attendees have been processed by the group scheduling engine yet. |
| REQUEST-WAITFORREPLY | Organizer | Meeting sent by organizer. Waiting for attendee replies. |
| ALL | N/A | (Default) All event and todo component states. |

- convertCalid

  In set_userprefs, to allow setting the subscribed list of calendars (icsSubscribed), and the subscribed list of groups (icsSet).

  Added to set_userprefs as a workaround to overcome the server limitation that prohibits the use of the colon character, ":".

  WCAP only checks the value of this parameter when the preference to be set is either icsSet or icsSubscribed.

- language

  Added to storeevents and storetodos to indicate which language a component is in.

- method

  For group scheduling, this parameter specifies the ITIP method used by the storeevents and storetodos commands.

- orgUID

Added to `storeevents` and `storetodos` for Palm Synchronization. It holds the user ID for the organizer of the event or the originator of the todo.

- `userid`

  Added to get_userprefs and set_userprefs so that the administrator may retrieve or set any user's preferences.

New for this version, some commands have added a parameter that previously only existed in other commands:

- `fmt-out` has been added to `set_calprops`.
- `orgEmail` has been added to `storetodos`.
- `tzid` has been added to:
  - `deleteevents_by_id`
  - `deletetodos_by_id`
  - `fetchcomponents_by_alarmrange`
  - `fetchcomponents_by_lastmod`
  - `fetchcomponents_by_range`
  - `fetchevents_by_id`
  - `fetchtodos_by_id`

  In addition, `tzid` in `storeevents` and `storetodos` has been modified. The parameter now has the same type as in the above mentioned commands, that is, it is a time zone ID string, such as "America/Los_Angeles".

# Output Format

The administrator can request the output format in three content types:

1. `text/calendar` - iCalendar
2. `text/xml` - iCalendar XML
3. `text/js` - JavaScript output

To change the output format, set `fmt-out` to the target value. If `fmt-out` is not specified, the default format of `text/js` will be returned.

### Condensed Output

The `brief` parameter allows for the printing of condensed event and todo data in JavaScript. The returned output is about half the size of the regular output and encompasses the following parameters:

**Table 10-6** Output Parameters Included in `brief` Output

| Parameters Included for Events | Parameters Included for Todos |
|---|---|
| calid | calid |
| created | completed |
| desc | created |
| dtstart | desc |
| dtend | dtstart |
| isAllDay | due |
| lastMod | isAllDay |
| linkCalid | lastMod |
| location | linkCalid |
| rid | location |
| summary | percent |
| tzid | rid |
| uid | summary |
| | tzid |
| | uid |

## Recurrence Handling

There are six parameters used to specify recurrence:

1. rrules. Semicolon-separated list of quoted recurrence-rule strings for recurring events.
2. rdates. Semicolon-separated list of ISO8601 date strings listing recurrence dates.

3. exrules. Semicolon-separated list of quoted recurrence-rule strings for dates to exclude.
4. exdates. Semicolon-separated list of ISO8601 date strings listing dates to exclude.
5. rid. ISO8601 DateTimeString giving the recurrence ID of an event.
6. mod. Numeric values 1-4. Modifier telling which instances of the event to store.
7. rchange. A boolean specifying whether or not to expand recurrences in `storecomponents`.

## 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. There are many parameters possible for recurrence rules. See RFC2445 for a complete description of the syntax.

Two very useful parameters for specifying recurrence are `freq` and `count`:

- The `freq` parameter in a rule defines the periodicity of the event, and has the following possible values:

| | |
|---|---|
| DAILY | The event recurs daily. |
| WEEKLY | The event recurs weekly. |
| MONTHLY | The event recurs monthly. |
| YEARLY | The event recurs yearly. |

- The `count` parameter in a rule defines how many times the meeting will repeat. If you do not specify the `count`, the default is the maximum number of recurrences allowed.The default maximum is 60. To change the maximum number, set the server configuration preference `calstore.recurrence.bound`.

The following example shows an `rrules` parameter that specifies two 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://webcalendarserver/storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=20000301T112233Z
&rrules="count%3D10%3Bfreq%3Ddaily";"freq%3Dweekly%3Bcount%3D4"
&dtend=20000301T112233&summary=uuuu
```

## 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 (3/31/00 11:22:33 and 5/31/00 11:22:33):

```
rdates=20000331T112233;20000531T112233
```

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

```
http://webcalendarserver/storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=20000301T112233Z
&rdates=20000331T112233;20000531T112233
&dtend=20000301T112233&summary=uuuu
```

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

## 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://webcalendarserver/storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=20000301T112233Z
&exrules="count%3D10%3Bfreq%3Ddaily";"freq%3Dweekly%3Bcount%3D4"
&rrules="count%3D100%3Bfreq%3Ddaily"&dtend=20000301T112233&summary=
uuuu
```

## 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 (3/31/00 11:22:33 and 5/31/00 11:22:33):

```
exdates=20000331T112233;20000531T112233
```

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

```
http://webcalendarserver/storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=20000301T112233Z
&exdates=20000331T112233;20000531T112233
&rrules="COUNT%3D200%3BFREQ=DAILY";dtend=20000301T112233&summary=uu
uu
```

## rid

This parameter specifies a unique recurrence date of an event or todo. Use `rid` in conjunction with the `mod` parameter to specify a range of events and todos to be modified.

For example:

```
http://webcalendarserver/storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=20000301T112233Z
&rid=20000331T112233;dtend=20000301T112233&summary=uuuu&mod=1
```

## mod

This 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:

| Value | Option |
|---|---|
| 1 | This instance only. |
| 2 | This and all future instances. |
| 3 | This and all prior instances. |
| 4 | This and all instances. |

For a non-recurring event or todo, the `rid` is `0`.

### rchange

The `rchange` parameter specifies whether recurrences are expanded in `storecomponents`. Normally, events and todo calendar components are expanded, so the parameter defaults to `1`.

However, 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://webcalendarserver/storeevents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=abcxyz&dtstart=20000201T112233Z
&rrules="byday%3Dfr%3Bfreq%3Dweekly"&summary=changed-event
&rid=20000201T112233Z&mod=2&rchange=0
```

# Commands

For all commands that have the `id` parameter (session ID), it is a required parameter. There are two exceptions to this rule. It is not required in order to grant anonymous access to a calendar, nor is it required in order to grant read access to a public calendar. In all other situations, you must provide the session ID in the `id` parameter.

**NOTE** The server supports “anonymous” as a special principal name. The anonymous user can log in with any password. It is not associated with any particular domain.

There are thirty-three WCAP commands:

- addlink
- change_password
- check_id
- createcalendar
- deletecalendar
- deletecomponents_by_range
- deleteevents_by_id
- deleteevents_by_range
- deletetodos_by_id
- deletetodos_by_range
- export
- fetchcomponents_by_alarmrange
- fetchcomponents_by_attendee_error
- fetchcomponents_by_lastmod
- fetchcomponents_by_range
- fetchevents_by_id
- fetchtodos_by_id
- get_all_timezones

- get_calprops
- get_freebusy
- get_guids
- get_userprefs
- import
- login
- logout
- ping
- search_calprops
- set_calprops
- set_userprefs
- storeevents
- storetodos
- upload_file
- version
- write_file

## addlink

### Purpose

Add links from one calendar's events or todos to another calendar.

### Parameters

Table 10-7 lists this command's five parameters:

**Table 10-7** `addlink` Parameters

| Parameter | Types | Purposes | Required | Default |
|---|---|---|---|---|
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `destCal` | string | The calendar containing the events or todos. You must have read access to this calendar. | Y | N/A |

**Table 10-7** `addlink` Parameters *(Continued)*

| Parameter | Types | Purposes | Required | Default |
|---|---|---|---|---|
| `rid` | semicolon separated list of strings. | A list of event and / or todo recurrence identifiers to which to create links. | Y | N / A |
| `srcCal` | string | The calendar receiving 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 |

### Description

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 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.

Unless the following three requirements are met, the transaction will fail:

- The source calendar and destination calendar parameters must be valid calendar ID's.
- The user must have write access to the destination calendar and read access to the source calendar.
- The `uid` and `rid` lists must have the same number of elements.

### Returns

If the transaction fails, an error of `ADDLINK_FAILED(12)` returns; otherwise, the return code is `0`.

### Example

This example shows how to add two event links to the calendar `jdoe`. The links should point to events in the `pub` calendar. The events are `1111` and `2222`, (their uids). Neither event is recurring, so their recurrence-id is `0`.

This URL executes the above transaction.

```
http://webcalendarserver/addlink.wcap?id=b5q2o8ve2rk02nv9t6&destCal
=jdoe&srcCal=pub&uid=1111;2222&rid=0;0
```

# change_password

### Purpose

Change the password of the current user. This command is deprecated in 5.0. It is here only for backward compatibility with 2.x. See the "Administrator's Guide" for details on changing a password.

### Parameters

Table 10-8 lists this command's four parameters:

**Table 10-8** `change_password` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `id` | string | The session identifier. | Y | N/A |
| `newPassword` | string | The new user password. | Y | N/A |
| `oldPassword` | string | The previous user password. | Y | N/A |
| `fmt-out` | string | The format for the returned data. | N | `text/js` |
| | | The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | | |

### Purpose

This command changes a user's password. Passwords are passed as plain text. Only users with administrative privilege may use this command unless the `service.wcap.allowchangepassword` preference is set.

You must specify the `id` parameter with the command unless the specified calendar is a public calendar.

### Returns

A failure of the command will return the error `CHANGE_PASSWORD_FAILED(27)`.

### Example

This example URL requests a password change:

```
http://webcalendarserver/change_password.wcap?id=b5q2o8ve2rk02nv9t6
&oldPassword=abc&newPassword=def
```

# check_id

### Purpose

This administrator only command allows the administrator to verify that a session is still valid.

### Parameters

Table 10-8 lists this command's four parameters:

**Table 10-9** `change_password` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `id` | string | The session identifier. | Y | N/A |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |

### Purpose

This command allows the administrator to verify that the session is still valid.

### Returns

The server returns the property `X-NSCP-WCAP-CHECK-ID`. If the value of this property is `1` the session is valid. If a zero (`0`) is returned, the session is invalid. It has either timed out or is unrecognized.

### Example

The following command returns whether the specified session is valid or not:

```
http://webcalendarserver/check_id.wcap?id=n3l0eeu6s3n3o3b8v&fmt-out
=text/calendar
```

The output returned is:

```
HTTP/1.1 200
Date: Thu, 14 Dec 2000 19:48:17 GMT
Content-type: text/calendar; charset=UTF-8
Content-length: 131
Last-modified: Thu, 14 Dec 2000 19:48:17 GMT
```

```
Pragma: no-cache
Expires: 0
Cache-Control: no-cache
Connection: Keep-Alive

BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-NSCP-WCAP-CHECK-ID:1
END:VCALENDAR lendar
```

## createcalendar

### Purpose

Create a new calendar.

### Parameters

Table 10-10 lists this command's four parameters:

**Table 10-10** `createcalendars` Parameters

| Parameter | Types | Purposes | Required | Default |
|---|---|---|---|---|
| `calid` | string | The user's `calid`, used to generate the new calendar's `calid`. | Y | N/A |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | Y | `text/js` |
| `set_calprops` | integer (`0`,`1`) | A boolean indicating whether to set the properties of the new calendar.<br>`1` = Set properties.<br>`0` = Do not set properties. | N | `0` |

### Description

Use this command to create a new calendar for the current user. To enable users who do not have administrative privileges to use this command, set the `service.wcap.allowcreatecalendars` preference in the `ics.conf` file.

### Creating a Valid Calid

The new `calid` of the created calendar is a combination of the user's `userid` and the `calid` parameter passed in. The system retrieves the `userid` by doing a lookup on the session specified with the `id` parameter. The format for the new calendar's `calid` is *`userid`*:*`calid`*. For example, if the user is `jdoe`, and the `calid` parameter is `tv`, the new calendar's calid is `jdoe:tv`.

The server will attempt to truncate `calid` parameters that are too long or contain any illegal characters. If the server is unable to truncate the `calid` parameter, the error returned is `ILLEGAL_CALID_NAME(30)`.

Valid characters for the `calid` parameter are:

- Alphabet characters (A-Z, a-z)
- Numeric characters (0-9)
- Three special characters
  - Dash (-)
  - Underscore (_)
  - Period (.)

For example, these are legal values for the `calid` parameter: `calendar1`, `calendar-1`, `calendar_1`, `calendar.1`

### Setting Calendar Properties

New in 5.0, you may set the calendar properties during creation. Pass in the `set_calprops` parameter with a value of `1`. You can then pass in any additional parameters as defined for the `set_calprops` command for setting calendar properties.

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

### Returns

The returned output shows the calendar properties (retrieved with a call to the fetchcomponents_by_range command) formatted according to the `fmt-out` value.

### Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If the newly created `calid` already exists in the database, an error code returns: `CREATECALENDAR_ALREADY_EXISTS_FAILED(25)`

### Example

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://webcalendarserver/createcalendar.wcap?id=b5q2o8ve2rk02nv9t6&
calid=newcal&set_calprops=1&name=New-Calendar&categories=business;w
ork
```

## deletecalendar

### Purpose

This command deletes a user's calendar.

### Parameters

Table 10-11 lists this command's three parameters:

**Table 10-11** `deletecalendars` Parameters

| Parameter | Types | Purposes | Required | Default |
|---|---|---|---|---|
| `calid` | string | The name of the calendar to delete. | Y | N/A |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `fmt-out` | string | The format for the returned data. | Y | `text/js` |
| | | The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | | |

### Description

Use this command to delete a user's calendar. You must pass in the `calid`, which is the name of the calendar to delete.

Only users with administrative privilege may use this command unless the `service.wcap.allowdeletecalendars` preference is set.

### Returns

The returned output is the formatted output from a call to `fetchcomponents_by_range`.

### Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If the `calid` doesn't exist in the database, the `delete_layer_errno[x]` value is set to `1`, where `x` is the calendar's index in the passed `calid` list. In addition, the `errno` variable contains the error: `CALENDAR_DOES_NOT_EXIST(29)`.

### Example

For example, sending this URL deletes the calendar named `newcal`.

`http://`*`webcalendarserver`*`/deletecalendar.wcap?id=b5q2o8ve2rk02nv9t6&calid=newcal`

## deletecomponents_by_range

### Purpose

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

### Parameters

Table 10-12 lists this command's six parameters:

**Table 10-12** `deletecomponents_by_range` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief output.<br>`0` = Complete output. | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to delete events and todos. | N | Current user's `calid`. |
| `dtend` | ISO8601 DateTime Z string (UTC) | End time and date of events or todos to be deleted.<br><br>A value of `0` means delete all events and todos up to the end of time. This value must be in coordinated universal time. | N | `0` |

**Table 10-12** `deletecomponents_by_range` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `dtstart` | ISO8601 DateTime Z string (UTC) | Start time and date of events or todos to be deleted.<br><br>A value of `0` means delete all events and todos from the beginning of time. This value must be in coordinated universal time. | N | `0` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N / A |

**Description.**
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. The range parameters, `dtstart` and `dtend`, should be specified in UTC time (the 'Z' must be on the end). Otherwise, results are unpredictable.

**Error Codes**
If the operation is successful, the error number of `0` is appended to the error string, If an error occurs while deleting from the calendar, the `delete_layer_errno[x]` value is set to `1`, where `x` is the calendar's index in the passed `calid` list. In addition, `errno` contains the error: `DELETECOMPONENTS_BY_RANGE_FAILED(21)`.

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

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

## deleteevents_by_id

**Purpose**
Deletes one or more events from a calendar by event identifier.

### Parameters

Table 10-13 lists this command's eight parameters:

**Table 10-13** `deleteevents_by_id` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | Calendar ID of event to delete. | Y | N/A |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. Required unless the calendar is public. | Y | `NULL` |
| `mod` | integer<br><br>`1,2,3,4` | A modifier indicating which recurrences to delete, or semicolon-separated list of modifiers. If a list, it must have same number of elements as `uid` list.<br><br>One of the following values:<br><br>`1` = `THISINSTANCE`<br>`2` = `THISANDFUTURE`<br>`3` = `THISANDPRIOR`<br>`4` = `THISANDALL` | Y | N/A |
| `notify` | integer<br><br>`0,1` | A boolean indicating whether or not to notify attendees of this change.<br><br>`1` = Notify attendees.<br>`0` = Do not notify attendees. | N | `0` |
| `rid` | string | Recurrence identifier of the event, or semicolon-separated list of recurrence identifiers.<br><br>If a list, it must have same number or elements as the `uid` list.<br><br>If there are no recurrences, the value is `0`. | Y | N/A |

**Table 10-13** `deleteevents_by_id` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `tzid` | time zone ID string | Default time zone to use if the `rid` parameter does not have a time zone specified.<br>For example, "America/Los_Angeles" | N | server's default time zone |
| `uid` | string | Unique Identifier of an event to be deleted, or semicolon-separated list of unique identifiers. | Y | N/A |

**Description.**
Use this command to delete the specified event or events from the specified calendar.

**Error Codes**
If the operation is successful, the error number of `0` is appended to the error string. If there was an error, the server returns two error arrays, `delete_layer_errno` and `delete_event_count`, with each element in the arrays representing the corresponding event in the `uid` list. Each element in `delete_layer_errno` indicates the error number for the corresponding event. Each element in `delete_event_count` indicated the number of occurrences successfully deleted for the corresponding event.

See also, "Error Handling" on page 158.

**Notify**
The `notify` parameter specifies whether or not to send an IMIP `CANCEL` message to the email attendees of the event. To send the cancellation message, set the `notify` value to `1`.

For example, here's a URL that sends the IMIP `CANCEL` message to all attendees of the event with `uid=001`.

```
http://webcalendarserver/deleteevents_by_id.wcap?id=3423423asdfasf&
calid=jdoe&uid=001&notify=1
```

**Recurrences**
If the `rid` parameter is passed, the command also deletes recurrences, as specified by the `mod` parameter. (See "Deleting Recurring Components" on page 156.) 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.

### Example

For example, there are two non-recurring events in the database with UIDs of `uid-EVENT1` and `uid-EVENT2`. Since 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`.

The following URL deletes the two events:

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

The resulting data would look like this:

```
HTTP/1.0 200
Date: Thu, 27 May 2000 18:40:24 GMT
Content-type: text/html; charset=iso-8859-1
Content-length: 4822
Last-modified: Thu, 27 May 2000 18:40:24 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache

<html><head><script>
function color(s) { if (s) document.bgColor=s }
var id='br6p3t6bh5po35r'
var userid='jdoe'
var calid='jdoe'
var errno=new Array()
var delete_event_errno = new Array()
var delete_event_count = new Array()
delete_event_errno[0]=0
delete_event_count[0]=1
delete_event_errno[1]=0
delete_event_count[1]=1
errno[0]=0
...
...
parent.ceCB(window.name)
</script></head></html>
```

## deleteevents_by_range

### Purpose

Delete events from a calendar in a specified range.

### Parameters

Table 10-14 lists this command's six parameters:

**Table 10-14** `deleteevents_by_range` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to delete events. | N | Current user's `calid`. |
| `dtend` | ISO8601 DateTime string | End time and date of events to be deleted.<br>A value of `0` means delete all events until the end of time. | N | `0` |
| `dtstart` | ISO8601 DateTime string | Start time and date of events to be deleted.<br>A value of `0` means delete all events from the beginning of time. | N | `0` |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

### Description.

Use this command to delete the events that fall completely within the specified range from the specified calendars. If a range is not specified (`dtstart` and `dtend`), it deletes all events from the 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 returns in the default JavaScript format.

### Error Codes

If the operation is successful, the error number of `0` is appended to the error string, `errno`. If an error occurs while deleting from the calendar, the `delete_layer_errno[x]` value is set to `1`, where `x` is the calendar's index in the passed `calid` list. In addition, the `errno` variable contains the error: `DELETEEVENTS_BY_RANGE_FAILED(22)`.

See also, "Error Handling" on page 158.

### Example

For example, assuming the user has read access to the calendars `jdoe` and `john`, the following URL would result in deleting *all* events from the calendars `jdoe` and `john`:

`http://webcalendarserver/deleteevents_by_range.wcap?id=2342347923479asdf&calid=jdoe;john&dtstart=0&dtend=0`

## deletetodos_by_id

### Purpose

Delete one or more todos from a calendar.

### Parameters

Table 10-15 lists this command's eight parameters:

**Table 10-15** `deletetodos_by_id` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | Calendar ID of the todo / todos to delete. | Y | N / A |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |

**Table 10-15** `deletetodos_by_id` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `id` | unique identifier string | The session identifier. | 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.<br><br>One of the following values:<br><br>`1` = THISINSTANCE<br>`2` = THISANDFUTURE<br>`3` = THISANDPRIOR<br>`4` = THISANDALL | Y | N/A |
| `notify` | integer (0,1) | A boolean telling whether or not to notify attendees of this change.<br><br>`1` = Notify attendees.<br>`0` = Do not notify attendees. | N | `0` |
| `rid` | string | The recurrence identifier of the todo, or a semicolon-separated list of recurrence identifiers.<br><br>If a list, it must have the same number or elements as the `uid` list.<br><br>If there are no recurrences, the value is `0`. | Y | N/A |
| `tzid` | time zone ID string | Default time zone to use if the `rid` parameter does not have a time zone specified.<br><br>For example, "America/Los_Angeles" | N | server's default time zone |
| `uid` | string | The unique identifier of a todo to be deleted, or a semicolon-separated list of unique identifiers. | Y | N/A |

**Description.**

Use this command to delete the specified todo 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 returns in the default JavaScript format.

### Error Codes

If there was an error, the server returns two error arrays, `delete_layer_errno` and `delete_todo_count`, with each element in the arrays representing the corresponding todo in the `uid` list. Each element in `delete_layer_errno` indicates the error number for the corresponding todo. Each element in `delete_todo_count` indicates the number of occurrences successfully deleted for the corresponding todo.

See also, "Error Handling" on page 158.

### Notify

The `notify` parameter specifies whether or not to send an IMIP `CANCEL` message to the email attendees of the todo. If `notify` is `1`, an email cancellation message is sent.

For example, here's a URL that sends the IMIP `CANCEL` message to all attendees of the todo with `uid=001`.

```
http://webcalendarserver/deletetodos_by_id.wcap?id=3423423asdfasf&c
alid=jdoe&uid=001&notify=1
```

### Recurrences

If the `rid` parameter is passed, the command also deletes recurrences, as specified by the `mod` parameter. See "Deleting Recurring Components" on page 156.

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.If the `rid` parameter is passed, the command also deletes recurrences, as specified by the `mod` parameter.

### Example

For example, there are two non-recurring events in the database with UIDs of `uid-TODO1` and `uid-TODO2`. Since 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`.

The following URL deletes the two events:

```
http://webcalendarserver/deletetodos_by_id.wcap?id=br6p3t6bh5po35r
  &uid=uid-TODO1;uid-TODO2&rid=0;0&mod=1;1
```

The resulting data would look like this:

```
HTTP/1.0 200
Date: Thu, 27 May 2000 18:40:24 GMT
Content-type: text/html; charset=iso-8859-1
Content-length: 4822
```

```
Last-modified: Thu, 27 May 2000 18:40:24 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache

<html><head><script>
function color(s) { if (s) document.bgColor=s }
var id='br6p3t6bh5po35r'
var userid='jdoe'
var calid='jdoe'
var errno=new Array()
var delete_todo_errno = new Array()
var delete_todo_count = new Array()
delete_todo_errno[0]=0
delete_todo_count[0]=1
delete_todo_errno[1]=0
delete_todo_count[1]=1
errno[0]=0
...
...
parent.ceCB(window.name)
</script></head></html>
```

## deletetodos_by_range

### Purpose

Delete todos in a range from a calendar.

### Parameters

Table 10-16 lists this command's six parameters:

**Table 10-16** `deletetodos_by_range` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to delete todos. | N | Current user's `calid` |
| `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` |

**Table 10-16** `deletetodos_by_range` Parameters *(Continued)*

| 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` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |

**Description.**

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.For example, the following URL would delete just the event that occurs on the date March 1, 2000 11:22:33 AM GMT.

`http://webcalendarserver/deleteevents_by_id.wcap?id=23423423434abc&calid=jdoe&uid=001&rid=20000301T112233Z&mod=1`

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.

**Error Codes**

If the operation is successful, the error number of `0` is appended to the error string. If an error occurs while deleting from the calendar, the `delete_layer_errno[x]` value is set to `1`, where `x` is the calendar's index in the passed `calid` list. In addition, the `errno` variable contains the error: `DELETETODOS_BY_RANGE_FAILED(23)`.

See also, "Error Handling" on page 158.

## export

**Purpose**

Export events and todos from a calendar to a file.

### Parameters

Table 10-17 lists this command's five parameters:

**Table 10-17** `export` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `calid` | string | A semicolon-separated list of calendar identifiers from which to export events and todos. The user must have read access to all calendars in the list. | N | Current user's `calid`. |
| `content-out` | string | Content type for output file. One of the following:<br>`text/calendar`<br>`text/xml` | Y | N/A |
| `dtend` | ISO8601 DateTime Z string. (UTC) | 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` |
| `dtstart` | ISO8601 DateTime Z string. (UTC) | 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` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

### Description.

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.

### Range

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).

### HTTP Post Examples

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

### *Example 1*

The following HTTP `POST` message exports all components of the calendars `jdoe` and `john` to an iCalendar file named `export.ica`:

```
POST
/export.wcap?id=t95qm0n0es3bo35r&calid=jdoe;john&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
```

### *Example 2*

The following HTML generates a `POST` message using the `export` command, producing files in both iCalendar and XML formats:

```
<form METHOD=POST ENCTYPE="multipart/form-data" NAME="john.ics"
ACTION="http://webcalendarserver:12345/export.wcap?id=t9u9m0eh8x5pu
9b
  &calid=jdoe;john&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="john.xml"
ACTION="http://webcalendarserver:12345/export.wcap?id=t9u9m0eh8x5pu
9b
  &calid=jdoe;john&dtstart=0&dtend=0&content-out=text/xml">
<ul>
<li>Press Export XML Now:<input type="submit" value="Export XML
now”>
</ul> </form>
```

This is the output generated:

```
HTTP/1.0 200
Date: Thu, 03 Jun 2000 22:15:52 GMT
Content-type: text/calendar
Content-disposition: attachment; filename="export.ics"
Content-length: 7004
```

```
BEGIN:VCALENDAR
METHOD:PUBLISH
VERSION:2.0
BEGIN:VEVENT
UID:tm-001
RECURRENCE-ID:20000519T010000Z
DTSTAMP:20000603T221548Z
SUMMARY:Calendar Staff
DTSTART:20000518T170000Z
DTEND:20000518T190000Z
CREATED:20000603T024254Z
LAST-MODIFIED:20000603T024254Z
PRIORITY:1
SEQ:1
GEO:37.463581;-121.897606
DESC:This is the description for event with UID = tm-001
URL:http://webcalendarserver/susan?uid=tm-001
LOCATION:Green Conference Room
STATUS:CONFIRMED
TRANSP:OPAQUE
END:VEVENT
BEGIN:VEVENT
UID:tm-001
RECURRENCE-ID:20000526T010000Z
DTSTAMP:20000603T221548Z
SUMMARY:Calendar Staff
DTSTART:20000525T170000Z
DTEND:20000525T190000Z
CREATED:20000603T024254Z
LAST-MODIFIED:20000603T024254Z
PRIORITY:1
SEQ:1
GEO:37.463581;-121.897606
DESC:This is the description for event with UID = tm-001
URL:http://webcalendarserver/susan?uid=tm-001
LOCATION:Green Conference Room
STATUS:CONFIRMED
TRANSP:OPAQUE
END:VEVENT
END:VCALENDAR
```

# fetchcomponents_by_alarmrange

**Purpose**

Retrieve calendar events and todos with alarm triggers.

### Parameters

Table 10-18 lists this command's seven parameters:

**Table 10-18** `fetchcomponents_by_alarmrange` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to retrieve components. | N | Current user's `calid`. |
| `compstate` | semicolon-separated list of component state keywords | The list of component states to fetch.<br><br>For `compstate` values, see Table 10-5 on page 164 | N | `ALL` |
| `dtend` | ISO8601 DateTime Z string | End time and date of events and todos to be returned.<br><br>A value of `0` means fetch all events. | N | `0` |
| `dtstart` | ISO8601 DateTime Z string | Start time and date of events / todos with alarms ready to go off during the specified time.<br><br>A value of `0` means fetch all events from the beginning of time. | N | `0` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N / A |
| `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` |

**Table 10-18** `fetchcomponents_by_alarmrange` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `tzid` | time zone ID string | Default time zone to use if the `rid` parameter does not have a time zone specified.<br>For example, “America/Los_Angeles” | N | server’s default time zone |

**Description.**
This command returns a list of events and todos having alarms that are about to go off during the specified time.

**Output Format**
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 JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all-day events, print out in the `eventD` array. All day events are signified by having the `isAllDay` flag turned on.

**maxResults Value**
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.

**Returns**
For each calendar specified in `calid`, the server returns the calendar's events and todos having alarms about to go off within the range specified by `dtstart` and `dtend`. Specify these in the ISO8601 DateTime Z string format.

If neither the starting nor ending date-time is specified, the server returns all events and todos with alarms, up to the specified maximum.

### Error Codes

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, `errno` is `FETCH_BY_ALARM_RANGE_FAILED(41)`.

### Example

For example, suppose there are 3 events:

- eventA: alarm on Feb 10 2000, 10:00 AM GMT
- eventB: alarm on Dec 25 2000, 12:30 PM GMT
- todoA: alarm on Jan 20 2000, 1:15 PM GMT

Here are three queries and their return values:

1. This query fetches all events and todos that have alarms.

   `http://webcalendarserver/fetchcomponents_by_alarmrange.wcap?id=abcdefg&dtstart=0&dtend=0`

   It returns eventA, eventB, and todoA.

2. This query fetches all events and todos that have alarms about to go off between 12/1/2000 and 1/31/2001.

   `http://webcalendarserver/fetchcomponents_by_alarmrange.wcap?id=abcdefg&dtstart=20001201T112233Z&dtend=20010131T112233Z`

   It returns eventB and todoA.

3. This query fetches all events and todos that have alarms to go off between 1/1/2000 and 6/1/2000.

   `http://webcalendarserver/fetchcomponents_by_alarmrange.wcap?id=abcdefg&dtstart=20000101T112233Z&dtend=20000601T112233Z`

   It returns eventA and todoA.

# fetchcomponents_by_attendee_error

# Purpose.

Fetch a list of components that had errors while sending group scheduling messages.

**Parameters.**

Table 10-19 lists this command's five parameters

**Table 10-19** `fetchcomponents_by_attendee_error` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `attendee` | string | The attendee's `calid` to search on. The command searches the calendars specified in the `calid` parameter for all errors in events for this attendee.<br><br>If this parameter is not specified, the command searches the primaryOwner's calendars for all event errors for any attendee. | N | N/A |
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = brief<br>`0` = complete | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to retrieve components. | N | Current user's `calid`. |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `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` |

**Description.**

Use this command to retrieve a list of events and todos that had errors when sending group scheduling messages. This command works almost like `fetchcomponents_by_range`.

### Output Format

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 JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all-day events, print out in the `eventD` array. All-day events are signified by having the `isAllDay` flag turned on.

### maxResults Value

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.

### Returns

For each calendar specified in `calid`, the server returns the events and todos that had errors for the specified attendee while sending group scheduling messages.

For example, if the `calid` parameter specifies calendars `cal1` and `cal2`, and the `attendee` parameter specifies `jdoe`, then both `cal1` and `cal2` would be searched for events with errors that had `jdoe` as an attendee. In the table that follows, cal1 and cal2 each have four events with associated attendees:

| cal1 Events | cal2 Events |
|---|---|
| event - 1c1 | event - 1c2 |
| attendee: `jdoe` | attendee: `john` |
| status: error | status: OK |
| event - 2c1 | event - 2c2 |
| attendee: `susan` | attendee: `jdoe` |
| status: error | status: error |
| event - 3c1 | event - 3c2 |
| attendee: `jdoe` | attendee: `susan` |
| status: OK | status: OK |

| cal1 Events | cal2 Events |
|---|---|
| event - 4c1 | event - 4c2 |
| attendee: `john` | attendee: `susan` |
| status: OK | status: error |

For attendee `jdoe`, the command returns: events 1c1 and 2c2.

### Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If the command fails for any reason, `errno` is `FETCH_BY_ATTENDEE_ERROR_FAILED(42)`.

## fetchcomponents_by_lastmod

### Purpose.

Fetch a list of components that have changed during a specified time period.

### Parameters.

Table 10-20 lists this command's seven parameters

**Table 10-20** `fetchcomponents_by_lastmode` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br>`1` = brief<br>`0` = complete | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to retrieve components. | N | Current user's `calid`. |
| `compstate` | semicolon-separated list of component state keywords | The list of component states to fetch.<br>For `compstate` values, see Table 10-5 on page 164 | N | `ALL` |
| `dtend` | ISO8601 DateTime Z string | End time and date of events to be returned.<br>A value of `0` means fetch all events. | N | `0` |

**Table 10-20** `fetchcomponents_by_lastmode` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `dtstart` | ISO8601 DateTime Z string | Start time and date of events to be returned.<br><br>A value of `0` means fetch all events from the beginning of time. | N | `0` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N / A |
| `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` |
| `tzid` | time zone ID string | Default time zone to use if the `rid` parameter does not have a time zone specified.<br><br>For example, “America / Los_Angeles” | N | server’s default time zone |

**Description.**
Use this command to retrieve a list of events and todos that have changed during a specific time period. This command works almost like `fetchcomponents_by_range`.

**Output Format**
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 JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all-day events, print out in the `eventD` array. All-day events are signified by having the `isAllDay` flag turned on.

**maxResults Value**
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.

#### Returns

For each calendar specified in `calid`, the server returns the calendar's the events and todos that changed during the range specified by `dtstart` and `dtend`. Specify these in the ISO8601 DateTime Z string format.

If neither the starting nor ending date-time is specified, the server returns all events and todos that have changed, up to the specified maximum.

#### Error Codes

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.

#### Example

For example, the calendar `jdoe` has these three events:

- eventA: last-modified on Feb 10 2000, 10:00 AM GMT.
- eventB: last-modified on Dec 25 2000, 12:30 PM GMT.
- todoA: last-modified on Jan 20 2000, 1:15 PM GMT.

Here are some queries and their return values:

`http://webcalendarserver/fetchcomponents_by_lastmod.wcap?id=jdoe&dtstart=0&dtend=0`

The above query would fetch all events and todos that have ever been modified. Thus eventA, eventB, and todoA would be returned.

`http://webcalendarserver/fetchcomponents_by_lastmod.wcap?id=jdoe&dtstart=20001201T112233Z&dtend=20000131T112233Z`

The above query would fetch all modified events and todos between 12/1/2000 and 1/31/2000. Thus eventB and todoA would be returned.

`http://webcalendarserver/fetchcomponents_by_lastmod.wcap?id=jdoe&dtstart=20000101T112233Z&dtend=20000601T112233Z`

The above query would fetch all events and todos that have been modified between 1/1/2000 and 6/1/2000. Thus eventA and todoA would be returned.

# fetchcomponents_by_range

### Purpose

Retrieve calendar events and todos.

### Parameters

Table 10-21 lists this command's seven parameters:

**Table 10-21** `fetchcomponents_by_range` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to retrieve components. | N | Current user's `calid`. |
| `compstate` | semicolon-separated list of component state keywords | The list of component states to fetch.<br><br>For `compstate` values, see Table 10-5 on page 164 | N | `ALL` |
| `dtend` | ISO8601 DateTime Z string | End time and date of events to be returned.<br><br>A value of `0` means fetch all events. | N | `0` |
| `dtstart` | ISO8601 DateTime Z string | Start time and date of events to be returned.<br><br>A value of `0` means fetch all events from the beginning of time. | N | `0` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

**Table 10-21** `fetchcomponents_by_range` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `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` |
| `tzid` | time zone ID string | Default time zone to use if the `rid` parameter does not have a time zone specified.<br>For example, “America/Los_Angeles” | N | server’s default time zone |

**Description.**
Use this command to retrieve properties, events, and todos from one or more specified calendars.

**Output Format**
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 JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all day events, print out in the `eventD` array. All day events are signified by having the `isAllDay` flag turned on.

**Returns**
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 DateTime Z string format.

If neither the starting nor ending date-time is specified, the server returns all events and todos, up to the specified maximum.

**Error Codes**
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.

**Example**
For example, to get the events for calendars `jdoe` and `susan` from Jan. 1, 2000 to Feb. 1, 2000, use the following URL:

```
http://webcalendarserver:81/fetchcomponents_by_range.wcap?id=b5q2o8
ve2rk02nv9t6&calid=jdoe
```

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

Here is the returned text:

timezoneList[0]=new TZ('GMT',
'GMT',
'GMT',
'+0000',
'+0000',
new Array())

#### Output Format

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 JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all-day events, print out in the `eventD` array. All-day events are signified by having the `isAllDay` flag turned on.

#### maxResults Value

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 command does not cap the number of returned components, and the returned JavaScript does not contain the `var maxResults` statement.

#### Error Codes

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.

#### Examples

For example to fetch the events and todos for calendars `jdoe` and `john`, from Jan. 1,2000 to Feb. 1, 2001, this URL could be used:

```
http://webcalendarserver/fetchcomponents_by_range.wcap?id=asdfasdf3
2424&calid=jdoe;john&dtstart=20000101T000000&dtend=20010101T000000
```

If you do not specify the start and end date, it returns all events and todos in the calendar, as in this example:

```
http://webcalendarserver/fetchcomponents_by_range.wcap?id=b5q2o8ve2
rk02nv9t6&calid=jdoe
```

```
HTTP/1.0 200
Date: Tue, 28 Jul 2000 22:43:01 GMT
Content-type: text/html; charset=iso-8859-1
Content-length: 5504
Last-modified: Tue, 15 Jun 2000 22:43:01 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache
```

```
<html><head><script>
function color(s) { if (s) document.bgColor=s }
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)
}
function
E(a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17,a18,a1
9,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,a1
9,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 TZ(a1,a2,a3,a4,a5,a6) {
this.tzid=a1
this.sName=a2
this.dName=a3
this.sOffset=a4
this.dOffset=a5
this.crossOver=a6
}
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()
var status_types=new Array('confirmed', 'tentative',
'cancelled')
calprops[0] = new CP('John Doe',
```

```
’jdoe’,
’’,
’’,
’0’,
’0’,
’’,
’’,
’’,
’’,
’Work Calendar for John Doe’,
’20000615T164146Z’,
’20000615T164146Z’,
’jdoe’,
new Array(’john’),
new Array(),
new Array(’business’))
event[0]=new E(’52452531427158444’,
’0’,
’jdoe’,
’20001223T233000’,
’20001224T023000’,
0,0,’jdoe event 3’,
’20000615T222725’,’20000615T222725’,
’1’,
’37.463581’,’-121.897606’,
’This is the description for event with UID =
52452531427158444’,
’http://webcalendarserver/’+’john?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’,
’20001223T130000’,
’20001223T123000’,
```

```
0,0,’jdoe event 2’,
’20000615T222725’,’20000615T222725’,
’1’,
’37.463581’,’-121.897606’,
’This is the description for event with UID =
51452531524178674’,
’http://webcalendarserver/’+’john?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’,
’20000519T010000Z’,
’jdoe’,
’20000518T170000’,
’20000518T190000’,
0,0,’Calendar Staff’,
’20000615T222725’,’
20000615T222725’,
’1’,
’37.463581’,’-121.897606’,
’This is the description for event with UID = tm-001’,
’http://webcalendarserver/’+’john?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’,
’20000526T010000Z’,
’jdoe’,
’20000525T170000’,
’20000525T190000’,
0,0,’Calendar Staff’,
’20000615T222725’,’20000615T222725’,
’1’,
’37.463581’,’-121.897606’,
’This is the description for event with UID = tm-001’,
’http://webcalendarserver/’+’john?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’,
’20000602T010000Z’,
’jdoe’,
’20000601T170000’,
’20000601T190000’,
0,0,’Calendar Staff’,
’20000615T222725’,’20000615T222725’,
’1’,
’37.463581’,’-121.897606’,
’This is the description for event with UID = tm-001’,
’http://webcalendarserver/’+’john?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',
'20000304T143000',
'20000620T222725',
3,0,'Fix the bugs',
'20000615T222725','20000615T222725',
'1',
'37.463581','-121.897606',
'This is the description for event with UID = abctodo',
'http://webcalendarserver/'+'john?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(),
'',
'')
layer_errno[0]=0
var size=5
var todosize=1
var dtstartrange="0"
```

```
var dtendrange="0"
errno[0]=0
parent.ceCB(window.name)
</script></head></html>
```

# fetchevents_by_id

### Purpose

Retrieve specific calendar events.

### Parameters

Table 10-22 lists this command's seven parameters:

**Table 10-22** `fetchevents_by_id` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | The unique identifier for the calendar from which to retrieve events. | N | Current user's `calid`. |
| `compstate` | semicolon-separated list of component state keywords | The list of component states to fetch.<br><br>For `compstate` values, see Table 10-5 on page 164 | N | `ALL` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N / A |

**Table 10-22** `fetchevents_by_id` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `mod` | integer | A modifier indicating which recurrences to retrieve. One of the following values:<br>1 = `THISINSTANCE`<br>2 = `THISANDFUTURE`<br>3 = `THISANDPRIOR`<br>4 = `THISANDALL` | N | `1`<br>`(THISINSTANCE)` |
| `rid` | ISO8601 DateTime Z string | The recurrence identifier for the event. For a nonrecurring event, set to `0`. | N | `0` |
| `tzid` | time zone ID string | Default time zone to use if the `rid` parameter does not have a time zone specified.<br>For example, “America/Los_Angeles” | N | server’s default time zone |
| `uid` | sting | The unique identifier for the event. | Y | N/A |

**Description.**
Use this command to retrieve the specified events and recurrences from the specified calendar. You must specify the `id` parameter with the command unless the specified calendar is a public calendar. The command returns recurrences as specified by the `mod` parameter. See “Recurrence Handling” on page 166

**Output Format**
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 JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all-day events, print out in the `eventD` array. All-day events are signified by having the `isAllDay` flag turned on.

**Returns**
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.

**Error Codes**
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.

# fetchtodos_by_id

### Purpose

Retrieve specific calendar todos.

### Parameters

Table 10-23 lists this command's seven parameters:

**Table 10-23** `fetchtodos_by_id` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | The unique identifier for the calendar from which to retrieve todos. | N | Current user's `calid`. |
| `compstate` | semicolon-separated list of component state keywords | The list of component states to fetch.<br><br>For `compstate` values, see Table 10-5 on page 164 | N | `ALL` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `mod` | integer | A modifier indicating which recurrences to retrieve. One of the following values:<br><br>1 = THISINSTANCE<br>2 = THISANDFUTURE<br>3 = THISANDPRIOR<br>4 = THISANDALL | N | `1`<br><br>`(THISINSTANCE)` |
| `rid` | ISO8601 DateTime Z string | The recurrence identifier for the todo. For a nonrecurring todo, set to `0`. | N | `0` |

**Table 10-23** `fetchtodos_by_id` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `tzid` | time zone ID string | Default time zone to use if the `rid` parameter does not have a time zone specified.<br>For example, “America/Los_Angeles” | N | server’s default time zone |
| `uid` | sting | The unique identifier for the todo. | Y | N/A |

#### Description

Use this command to retrieve the specified 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.

#### Output Format

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 JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all-day events, print out in the `eventD` array. All-day events are signified by having the `isAllDay` flag turned on.

#### Returns

For each calendar specified in `calid`, the server returns the calendar's todos of that calendar. If the todo has recurrences, it returns them as specified by the `rid` and `mod` parameters. See “Recurrence Handling” on page 166.

#### Error Codes

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.

## get_all_timezones

#### Purpose

Retrieve data about all timezones supported by the server.

### Parameters

Table 10-24 lists this command's four parameters:

**Table 10-24** `get_all_timezones` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `dtend` | ISO8601 DateTime Z string | End date of the crossover values to retrieve.<br><br>A value of `0` means get all crossover dates until the last known year (2087). | N | `0` |
| `dtstart` | ISO8601 DateTime Z string | Start date of crossover values to retrieve.<br><br>A value of `0` means get all crossover dates from the first known year (1987). | N | `0` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N / A |

### Description.

Use this command to retrieve data about all timezones that are supported by the server. The crossover values are defined to be the dates when the timezone enters / exits daylight savings time. The odd index dates are the beginning of daylight-savings. The even index dates are the end of daylight-savings. If the timezone does not have daylight-savings, then this value will be set to the empty-string.

### Returns

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).

The server returns data in the format specified by the `fmt-out` parameter. If you do not pass in the `fmt-out` parameter, the server uses the default JavaScript format.

### Error Codes

If there was an error in getting the timezones, the server returns the error `GET_ALL_TIMEZONES_FAILED(24)`.

### Example

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','20000404T100000Z','20001031T090000Z',
'20000402T100000Z','20001029T090000Z','20010401T100000Z',
'20011028T090000Z', '20020407T100000Z', '20021027T090000Z',
'20030406T100000Z', '20031026T090000Z', '20040404T100000Z',
'20041031T090000Z', '20050403T100000Z', '20051030T090000Z',
'20060402T100000Z', '20061029T090000Z'))
```

```
The"America/Phoenix" timezone does not have daylight-savings.  Thus
the daylight elements exactly equal the standard elements.  Also,
the crossover strings in set to the empty string.
```

```
timezoneList[23] = new TZ('America/Phoenix',
'MST',
'MST',
'-0700',
'-0700',
new Array())
```

## get_calprops

### Purpose

Retrieve calendar properties.

### Parameters

Table 10-25 lists this command's three parameters:

**Table 10-25** `get_calprops` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `calid` | semicolon-separated list of strings | A list of calendar identifiers for the calendars from which to retrieve properties. | N | Current user's `calid`. |

**Table 10-25** `get_calprops` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

**Description.**
Use this command to retrieve the calendar properties for the specified calendars.

**Returns**
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)

### Error Codes

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

If the calendar is not found, the server sets `layer_errno[x]` to the value `2` for that calendar's index.

If the fetch fails for any calendar, its error number, `errno`, is set to `GET_CALPROPS_FAILED(20)`.

### Example

In the following example, you want to retrieve the calendar properties for the calendars `jdoe`, `susan`, `pub`, `john`, and `hasdf` in that order.

The following conditions apply:

- The user has read access the calendars `jdoe`, `pub`, and `john`, but not to the calendar `susan`.
- The calendar `hasdf` does not exist in the database.

You get the following results:

- The call to `get_calprops.wcap` returns data for the calendars `jdoe, pub`, and `john,` 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(20)` value.

This is the URL and the returned data:

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

```
HTTP/1.0 200
Date: Wed, 16 Jun 2000 22:21:27 GMT
Content-type: text/html; charset=iso-8859-1
Content-length: 1624
Last-modified: Wed, 16 Jun 2000 22:21:27 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache

<html><head><script>
```

```
function color(s) { if (s) document.bgColor=s }
var id=’bu9p3eb8x5p2nm0q3’
var userid=’jsun’
var calid=’jsun’
var errno=new Array()
var timezoneList = new Array()
var calprops = 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)
}

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
}

calprops[0] = new CP('jdoe',
'John Doe',
'',
'',
'0',
'0',
'',
'',
'',
   'Work Calendar for John Doe',
```

```
'20000615T222725Z',
'20000615T222725Z',
   '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',
'20000615T222725Z',
   '20000615T222725Z',
'susan',
new Array(),
new Array(),
new Array('group','business'))
layer_errno[2]=0
calprops[3] = new CP('john',
'John Calendar Hosting Server',
'',
'',
'0',
'0',
'',
'',
'',
'John Project Calendar',
'20000615T222725Z',
'20000615T222725Z',
'susan',
new Array('smith','jones','fred','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)
```

```
new Array())
```

# get_freebusy

### Purpose

Get the freebusy information for users.

### Parameters

Table 10-26 lists this command's five parameters:

**Table 10-26** `get_freebusy` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `calid` | semicolon-separated list of strings | A list of calendar identifiers for the calendars from which to retrieve properties. | N | Current user's `calid`. |
| dtstart | ISO8601 DateTime Z string | Start time of freebusy search. | Y | N/A |
| dtend | ISO8601 DateTime Z string | End time of freebusy search. | Y | N/A |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

### Description.

This command to retrieve the freebusy information for specified users. Freebusy information tells whether or not a user's time has been scheduled. It does not indicate why the user is busy.

### Error Codes

If this command fails for any reason, `errno` is set to `GET_FREEBUSY_FAILED(39)`.

### Example

For example, a calendar called `jdoe` has the following events:

| | |
|---|---|
| 100:00-11:00 | first meeting |
| 12:00-1:00 | lunch |

3:00-4:00 second meeting

The freebusy time for `jdoe` (from 9:00 to 6:00) would be the following:

9-10 : Free

10-11 : Busy

11-12 : Free

12-1 : Busy

1-3 : Free

3-4 : Busy

4-6 : Free

The following URL generates freebusy information found in the calendar `jdoe` between May 1 2000 and July 1 2000.

The output is returned in `text/calendar` format.

```
http://webcalendarserver/get_freebusy.wcap?id=2mu95r5so0hq68ts6q3&c
alid=jsun&dtstart=20000501T112233Z&dtend=20000701T112233Z&fmt-out=t
ext/calendar
```

Here is the output:

```
HTTP/1.1 200
Date: Wed, 17 May 2000 01:23:23 GMT
Content-type: text/calendar; charset=UTF-8
Content-length: 1430
Last-modified: Wed, 17 May 2000 01:23:23 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache
Connection: Keep-Alive

BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-NSCP-CALPROPS-LAST-MODIFIED:20000517T012259Z
X-NSCP-CALPROPS-CREATED:20000517T012259Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
```

```
X-NSCP-CALPROPS-DESCRIPTION:Work Calendar for John Doe
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-OWNERS:susan
X-NSCP-CALPROPS-CATEGORIES:business
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^S^g
BEGIN:VFREEBUSY
DTSTART:20000501T112233Z
DTEND:20000701T112233Z
FREEBUSY;FBTYPE=FREE:20000501T112233Z/20000518T170000Z
FREEBUSY;FBTYPE=BUSY:20000518T170000Z/20000518T190000Z
FREEBUSY;FBTYPE=FREE:20000518T190000Z/20000525T170000Z
FREEBUSY;FBTYPE=BUSY:20000525T170000Z/20000525T190000Z
FREEBUSY;FBTYPE=FREE:20000525T190000Z/20000601T170000Z
FREEBUSY;FBTYPE=BUSY:20000601T170000Z/20000601T190000Z
FREEBUSY;FBTYPE=FREE:20000601T190000Z/20000608T170000Z
FREEBUSY;FBTYPE=BUSY:20000608T170000Z/20000608T190000Z
FREEBUSY;FBTYPE=FREE:20000608T190000Z/20000615T170000Z
FREEBUSY;FBTYPE=BUSY:20000615T170000Z/20000615T190000Z
FREEBUSY;FBTYPE=FREE:20000615T190000Z/20000622T170000Z
FREEBUSY;FBTYPE=BUSY:20000622T170000Z/20000622T190000Z
FREEBUSY;FBTYPE=FREE:20000622T190000Z/20000629T170000Z
FREEBUSY;FBTYPE=BUSY:20000629T170000Z/20000629T190000Z
FREEBUSY;FBTYPE=FREE:20000629T190000Z/20000701T112233Z
END:VFREEBUSY
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

This example shows the output when the `fmt-out` specifies `text/xml`.

```
http://webcalendarserver/get_freebusy.wcap?id=2mu95r5so0hq68ts6q3&c
alid=jdoe&dtstart=20000501T112233Z&dtend=20000701T112233Z&fmt-out=t
ext/xml
```

The output returned for `text/xml`:

```
HTTP/1.1 200
Date: Wed, 17 May 2000 01:24:58 GMT
Content-type: text/xml; charset=UTF-8
Content-length: 1868
Last-modified: Wed, 17 May 2000 01:24:58 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache
Connection: Keep-Alive
```

```
<?xml version="1.0" encoding="UTF-8"?>
<iCalendar>
<iCal version="2.0" prodid="-//iPlanet/Calendar Hosting
Server//EN" METHOD="PUBLISH">
<X-NSCP-CALPROPS-LAST-MODIFIED>20000517T012259Z</X-NSCP-CALPR
PS-LAST-MODIFIED>
<X-NSCP-CALPROPS-CREATED>20000517T012259Z</X-NSCP-CALPROPS-CREATED>
<X-NSCP-CALPROPS-READ>999</X-NSCP-CALPROPS-READ>
<X-NSCP-CALPROPS-WRITE>999</X-NSCP-CALPROPS-WRITE>
<X-NSCP-CALPROPS-DESCRIPTION>Work Calendar for John Doe
</X-NSCP-CALPROPS-DESCRIPTION>
<X-NSCP-CALPROPS-RELATIVE-CALID>jdoe</X-NSCP-CALPROPS-RELATIVE-CALI
D>
<X-NSCP-CALPROPS-NAME>John Doe</X-NSCP-CALPROPS-NAME>
<X-NSCP-CALPROPS-PRIMARY-OWNER>jdoe</X-NSCP-CALPROPS-PRIMARY-OWNER>
<X-NSCP-CALPROPS-OWNERS>susan</X-NSCP-CALPROPS-OWNERS>
<X-NSCP-CALPROPS-CATEGORIES>business</X-NSCP-CALPROPS-CATEGORIES>
<X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY>@^a^S^g</X-NSCP-CALPROPS-ACCE
SS-CONTROL-ENTRY>

<FREEBUSY>
<START>20000501T112233Z</START>
<END>20000701T112233Z</END>
<FB FBTYPE="FREE">20000501T112233Z/20000518T170000Z</FB>
<FB FBTYPE="BUSY">20000518T170000Z/20000518T190000Z</FB>
<FB FBTYPE="FREE">20000518T190000Z/20000525T170000Z</FB>
<FB FBTYPE="BUSY">20000525T170000Z/20000525T190000Z</FB>
<FB FBTYPE="FREE">20000525T190000Z/20000601T170000Z</FB>
<FB FBTYPE="BUSY">20000601T170000Z/20000601T190000Z</FB>
<FB FBTYPE="FREE">20000601T190000Z/20000608T170000Z</FB>
<FB FBTYPE="BUSY">20000608T170000Z/20000608T190000Z</FB>
<FB FBTYPE="FREE">20000608T190000Z/20000615T170000Z</FB>
<FB FBTYPE="BUSY">20000615T170000Z/20000615T190000Z</FB>
<FB FBTYPE="FREE">20000615T190000Z/20000622T170000Z</FB>
<FB FBTYPE="BUSY">20000622T170000Z/20000622T190000Z</FB>
<FB FBTYPE="FREE">20000622T190000Z/20000629T170000Z</FB>
<FB FBTYPE="BUSY">20000629T170000Z/20000629T190000Z</FB>
<FB FBTYPE="FREE">20000629T190000Z/20000701T112233Z</FB>
</FREEBUSY>
<X-NSCP-WCAP-ERRNO>0</X-NSCP-WCAP-ERRNO>
</iCal>
</iCalendar>
```

# get_guids

### Purpose

Generate a set of globally unique identifiers.

### Parameters

Table 10-27 lists this command's two parameters:

**Table 10-27** `get_guids` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `guidCount` | integer | Number of `GUID`s to return. | N | `1` |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |

### Description.

This command returns the specified number of globally unique identifiers (`GUID`s). The client need not be authenticated to call this command.

### Example

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

#### *iCalendar Format*

`http://`*`webcalendarserver`*`/get_guids.wcap?guidCount=10`
`&`**`fmt-out=text/calendar`**

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:iPlanet Calendar Server 5.0
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
```

### *XML Format*

http://*webcalendarserver*/get_guids.wcap?guidCount=10&**fmt-out=text/xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<iCalendar>
<iCal>
<X-NSCP-GUID0>fde4b537d6040000015000000c3000000<X-NSCP-GUID0>
<X-NSCP-GUID1>fde4b5379478000016000000c3000000<X-NSCP-GUID1>
<X-NSCP-GUID2>fde4b5372a6d000017000000c3000000<X-NSCP-GUID2>
<X-NSCP-GUID3>fde4b537a3550000018000000c3000000<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>
```

### *JavaScript Format*

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

http://*webcalendarserver*/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]=9ee4b537577800000100000c3000000
guid[1]=9ee4b5376a52000002000000c3000000
guid[2]=9ee4b5375a47000003000000c3000000
guid[3]=9ee4b537bf01000004000000c3000000
guid[4]=9ee4b537e05e000005000000c3000000
guid[5]=9ee4b537be5a000006000000c3000000
guid[6]=9ee4b537d544000007000000c3000000
guid[7]=9ee4b5372867000008000000c3000000
```

```
guid[8]=9ee4b5375731000009000000c3000000
guid[9]=9ee4b537830600000a000000c3000000
parent.ceCB(window.name)
</script></head></html>
```

## get_userprefs

### Purpose

Retrieve the calendar preferences for the current user.

### Parameters

Table 10-28 lists this command's two parameters:

**Table 10-28** `get_userprefs` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `userid` | string | Used only by administrators. Indicates which user's preferences to get. | N | N/A |

### Description.

This command retrieves all the calendar preferences for the current user, and the following server preferences relating to this user:

- `allowchangepassword`. Users can change the password.
- `allowcreatecalendars`. Users can create calendars.
- `allowdeletecalendars`. Users can delete calendars.
- `allowpublicwritablecalendars`. Users can have publicly writable calendars.
- `validateowners`. If set to 1, the server must validate that each owner of a calendar exists in the directory (whether the directory is LDAP or a CSAPI compatible user mechanism).

- allowsetprefs. If set to 1, allow set_userprefs.wcap to modify the user preferences.

To use the parameter `userid`, two conditions must be met. The server configuration preference `service.admin.calmaster.wcap.allowgetmodifyuserprefs` must be set to "yes" in the `ics.conf` file, and the requestor must be logged in as an administrator, using the `login.wcap` command.

See the "*Administrator's Guide*" for detailed information about server preferences.

**Example**

The following URL retrieves user preferences for the current user:

`http://`*`webcalendarserver`*`/get_userprefs.wcap?id=b5q2o8ve2rk02nv9t6`

This is the data returned:

```
<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@mailserver')
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> </script></head>
<body bgcolor=#FFFFFF>
<br><form action=set_userprefs.wcap method=post name=form>
cmd<input name=cmd>
old<input name=oldPassword>
new<input name=newPassword>
<input name=add_attrs>
<input name=del_attrs>
<input name=set_attrs>
<input name=id value='e2np3w9o0v6k0u9v9b'>
<input name=bgcolor>
<input name=security>
<input type=submit>
</form>
</body>
<script>var form=document.form
parent.cfgCB()</script>
```

# import

### Purpose

Import events and todos from a file to a calendar.

### Parameters

Table 10-29 lists this command's five parameters:

**Table 10-29** `import` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `calid` | string | Identifier of a calendar to which to import event. | Y | N / A |
| `content-in` | string | Content type of input data. One of the following values:<br>`text/calendar`<br>`text/xml` | Y | N / A |
| `dtend` | ISO8601 DateTime Z string | End time and date of the events and todos to import.<br>A value of `0` means import all components from the start date to the last date in the file. | N | `0` |

**Table 10-29** `import` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `dtstart` | ISO8601 DateTime Z string | Start time and date of events and todos to import.<br>A value of `0` means import all components from the earliest date in the file to the end date. | N | `0` |
| `id` | unique identifier string | The session identifier. Required unless the calendar is public. | Y | N/A |

**Description.**

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 file's MIME content type in the `content-in` parameter.

If you do not specify either the starting or ending date, or you pass in `0` as the value for `dtstart` and `dtend`, the command adds all events and todos in the file 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 the file containing the exported events and todos to the `POST` message. This file must be in either iCalendar (`.ics`) or XML (`.xml`) format.

**Example**

The following `POST` message imports the attached iCalendar file to the calendar `jdoe` using the `import` command (The session id is required.):

```
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:20000105T100000
DTEND:20000105T110000
DTSTAMP:20000104T120000
CREATED:20000105T110000Z
LAST-MODIFIED:20000104T120000Z
SUMMARY:Weekly QA Meeting
```

```
UID:random-uid001
END:VEVENT
BEGIN:VEVENT
DTSTART:20000106T100000
DTEND:20000106T110000
DTSTAMP:20000104T120000
CREATED:20000105T110000Z
LAST-MODIFIED:20000104T120000Z
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://webcalendarserver:12345/import.wcap?id=t95qm0n0es3bo
35r&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 a specific user.

### Parameters

Table 10-30 lists this command's five parameters:

**Table 10-30** `login` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js`<br><br>If `text/calendar` or `text/xml`, the refresh parameter is automatically set to 1. | N | `text/js` |
| `lang` | enum | The user's preferred language. | N | `NULL` |
| `password` | string | The user's password. | N | N/A |
| `refresh` | integer (0, 1) | A boolean indicating whether to return just the new session ID or everything.<br><br>1 = Return only the session identifier.<br>0 = Return everything. | N | `0` |
| `user` | string | The user's name. | N | `NULL` |

### Description.

This command logs a specific user into iPlanet Calendar Server, 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 email address. The password is also plain text.

### Authentication

Do internal authentication using either the default LDAP authentication, or your own CSAPI plug-in to link to an existing user authentication method (For more information on CSAPI authentication, see "csIAuthentication" on page 48). For more information on the Proxy Authentication SDK, see Chapter 6 for the overview and Chapter 7 for the API Reference.

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

### Example

For example, the following URL attempts to login user `jdoe`:

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

### Returns

If you do not pass in the `refresh` parameter, and `fmt-out` specifies `text/js`, the default value is 0. If `fmt-out` specifies one of the two other styles, `text/calendar` or `text/xml`, the default value is `1`.

If you specify `1` in the `refresh` parameter, the returned page contains only 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 iPlanet Calendar Server's user interface.

With the parameter set to `0`, the `login` command returns everything, including the default html file:

```
HTTP/1.0 302 OK
Date: Tue, 11 May 2000 22:38:33 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache
Content-Length: 0
Last-modified: Tue, 11 May 2000 22:38:33 GMT
Location:
http://webcalendarserver/en/main.html?id=er6en05tv6n3bv9&lang=en
 &host=http://webcalendarserver/

<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 current user's session.

### Parameters

Table 10-31 lists this command's two parameters:

**Table 10-31** `logout` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

### Description.

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://`*`webcalendarserver`*`/logout.wcap?id=bu9p3eb8x5p2nm0q3`

# ping

### Purpose

Determine whether the calendar server is active.

### Parameters

This command takes no parameters.

### Description.

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

Only users with administrative privilege can use this command.

### Returns

For this example, the administrator's `userid` and `calid` are both `adminX`.

```
HTTP/1.0 200
Date: Thu, 03 Jun 2000 21:31:42 GMT
Content-type: text/html; charset=iso-8859-1
Content-length: 190
Last-modified: Thu, 03 Jun 2000 21:31:42 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache

<html><head><script>
function color(s) { if (s) document.bgColor=s }

var id='bb5rt6eb5pu9v9w9'
var userid='adminX'
var calid='adminX'
var errno=new Array()
parent.ceCB(window.name)
</script></head></html>
```

## search_calprops

### Purpose

Search for a calendar's properties.

### Parameters

Table 10-32 lists this command's seven parameters:

**Table 10-32** `search_calprops` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `calid` | integer (0,1) | A boolean indicating whether or not to search the `calid` property.<br>`1` = Search the `calid` property<br>`0` = Do not search it. | N | `0`, unless both `primaryOwner` and `name` are `0` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `maxResults` | integer | The maximum number of results to return. | N | `200` |

**Table 10-32** `search_calprops` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `name` | integer (0,1) | A boolean indicating whether or not to search the `name` property.<br><br>`1` = Search the `calid` property<br>`0` = Do not search it. | N | `0` |
| `primaryOwner` | integer (0,1) | A boolean indicating whether or not to search the `primaryOwner` property.<br><br>`1` = Search the `primaryOwner` property.<br>`0` = Do not search it. | N | `0` |
| `searchOpts` | integer<br><br>0,1,2,3 | How to perform the search. One of the following:<br><br>`0 = CONTAINS`<br>`1 = BEGINS_WITH`<br>`2 = ENDS_WITH`<br>`3 = EXACT` | N | `0` |
| `search-string` | string | The string to search for in calendars. | Y | N / A |

**Description.**
This command searches for a calendar using the query type specified by `searchOpts`. It returns the calendar properties for all calendars where a string in the specified properties (`primaryOwner`, `calid`, `name`), matches the `search-string`, using the specified `seachOpts`, up to the specified maximum number of matches (`maxResults`).

**Search Properties**
This command searches for a matching string in one of three properties:

- `calid`. The calendar's unique identifier.
- `name`. The calendar's common name (text).
- `primaryOwner`. The calendar's primary owner.

To search for the value of a specific property, set that parameter to `1`. If both `primaryOwner` and `name` are set to `0`, `calid` defaults to `1` and the server assumes the `search-string` is a `calid`, regardless of the `calid` parameter setting.

**Search Options**
There are four search options:

- Return the calendar properties that contain the `search-string` (`CONTAINS`).
- Return the calendar properties that begin with the `search-string` (`BEGINS_WITH`).
- Return the calendar properties that ends with the `search-string` (`ENDS_WITH`).
- Return the calendar properties that exactly match the `search-string` (`EXACT`).

**Example**

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

In the returned data, the calsize variable specifies the number of calendars returned in the output.

The following data is a result of the example URL above:

```
HTTP/1.0 200
Date: Fri, 30 Apr 2000 23:49:19 GMT
Content-type: text/html; charset=iso-8859-1
Content-length: 10113
Last-modified: Fri, 30 Apr 2000 23:49:19 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache

<html><head><script>
function color(s) { if (s) document.bgColor=s }

var id='bu9p3eb8x5p2nm0q3'
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)
}
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
}
calprops[0] = new CP('hockey',
'',
'',
'',
'',
'1',
'1',
'',
'',
'',
'',
'20000615T222725Z',
'20000615T222725Z',
'susan',
new Array(),
new Array(),
new Array('Sports'))
var calsize=1
var maxReturnSize=50
errno[0]=0
parent.ceCB(window.name)
</script></head></html>
```

# set_calprops

### Purpose

Set the calendar properties of a calendar.

### Parameters

Table 10-33 lists this command's fifteen parameters:

**Table 10-33** `set_calprops` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| acl | string | A semicolon-separated list of strings specifying the new value of the access control entries. | N | ,,,, |
| `cal` | encoded string | A list of parameters to decode.<br>There can be multiple instances of this parameter. | N | N / A |
| `calid` | string | Identifier of the calendar to modify. | Y | N / A |
| `categories` | string | A semicolon-separated list of strings containing the new categories the calendar belongs to. | N | N / A |
| `charset` | string | The character set for the calendar. | N | N / A |
| `description` | string | The description of the calendar. | N | N / A |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N / A |
| `lang` | string | The language of the calendar. | N | N / A |
| `master` | string | The email contact for the calendar. | N | N / A |
| `multiple` | integer | The number of calendars for which to set these preferences. | N | `0` |
| `name` | string | The new text name of the calendar. | N | N / A |
| `owners` | string | A semicolon-separated list of strings containing he new list non-primary owners. | N | N / A |

**Table 10-33** `set_calprops` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `read` | integer | This parameter no longer has functionality in version 5.0. It remains here only for backwards compatibility with 2.x. The `acl` parameter replaces it.<br><br>The new read-access value of the calendar. The value can be one of the following:<br><br>`0 PRIVATE`<br>`1 PUBLIC`<br>`4 PRIMARY_OWNER_ONLY` | N | N / A |
| `tzid` | string | The new timezone identifier for this calendar. | N | ,,,, |
| `write` | integer | This parameter no longer has functionality in version 5.0. It remains here only for backwards compatibility with 2.x. The `acl` parameter replaces it.<br><br>The new write-access value of the calendar. The value can be one of the following:<br><br>`0 PRIVATE`<br>`1 PUBLIC`<br>`4 PRIMARY_OWNER_ONLY` | N | N / A |

**Description.**

This command only changes the values of the parameters you specify. In this way, then, it is an *update* command, rather than a *replace* command as it was in 2.x. It is not necessary to supply all parameters in the command, only the ones you wish to change. Calendar properties are special states of a calendar, which includes the calendar's name, read and write permission values (`acl` parameter), the list of owners, and the list of categories.

The `read` and `write` parameters have been deprecated in 5.0. The functionality has been replaced with the `acl` parameter. They are included for backwards compatibility with 2.x only.

Use `set_calprops` to do the following:

- Change the name of the calendar.
- Change owner of calendar.
- Change category of calendar.

- Change read permission of calendar's event.
- Change write permission of calendar's event.
- Change description of calendar.
- Change character set of calendar.
- Change language of calendar.
- Change e-mail contact of this calendar.
- Change the timezone-identifier of the calendar.

#### Single Calendar Example

Here is a sample URL that sets calendar properties: (The `calid` parameter is required.)

```
http://webcalendarserver?set_calprops.wcap?id=dfasdfzd3ds&calid=jdo
e&categories=business;meeting&name=John%39s%32Calendar
```

#### Multiple Calendars Example

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:

| 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://webcalendarserver?id=fasdfzd3ds
     &multiple=3
     &cal=calid%3Dxxxx%26description%3DX-Calendar
     &cal=calid%3Dyyyy%26description%3DY-Calendar
     &cal=calid%3Dzzzz%26description%3DZ-Calendar
```

This is the equivalent of the following three URLs:

```
http://webcalendarserver?id=fasdfzd3ds&calid=xxxx&desc=X-Calendar
http://webcalendarserver?id=fasdfzd3ds&calid=yyyy&desc=Y-Calendar
http://webcalendarserver?id=fasdfzd3ds&calid=zzzz&desc=Z-Calendar
```

In the example, notice that since the `multiple` parameter is set to `3`, there are three instances of the `cal` parameter. The value of each `cal` parameter is an encoded list of parameters and their values. The server will decode each `cal` parameter and set the properties appropriately.

#### Access Control Entries

See "Access Control Entries," on page 152, in the Common Topics section at the front of this chapter.

#### Freebusy Access

See "Freebusy Access," on page 162, in the Common Topics section at the front of this chapter.

#### Choosing a Different Language or Character Set

See "Choosing a Different Language or Character Set," on page 155, in the Common Topics section at the front of this chapter.

## set_userprefs

#### Purpose

Modify the preferences or password for a session.

#### Parameters

Table 10-34 lists this command's five parameters:

**Table 10-34** `set_userprefs` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `add_attrs` | string | Add a new preference. | N | N/A |
| `convertCalid` | integer (`0`,`1`) | When set to `1` and setting the preferences `icsSet` or `icsSubscribed`, indicates to the server to convert the character "^" to ":" when storing the `calid`.<br><br>When set to `0`, the parameter is ignored. | N | `0` |
| `del_attrs` | string | Delete an existing preference. | N | N/A |

**Table 10-34** `set_userprefs` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | Y | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `set_attrs` | string | Modify a preference value. | N | N/A |
| `userid` | string | Used only by administrators. Indicates which user's preferences to set. | N | N/A |

**Description.**

This command modifies the preferences for the current user. You may also modify the user's password through LDAP.

Use of this parameter is only necessary when setting the subscribed list of calendars (`icsSubscribed`), or the subscribed list of groups (`icsSet`). The `calid` on incoming commands must have the colon, ":", replaced with a caret, "^". For example, if the `calid` is `jdoe:personal`, then WCAP must receive it as `jdoe^personal` in order for the command to work properly.

If the value of `convertCalid` is `1`, then WCAP will convert the "^" back to a ":". If the value of the `convertCalid` is `0`, the conversion will not be done

When the administrator is logged in, and the ics.conf file preference `service.admin.calmaster.wcap.allowgetmodifyuserprefs` is set to "yes", the `userid` parameter specifies which user's preferences to set.

**Returns**

The function returns the text of `get_userprefs`.

**Examples**

For example, the following URL adds a new preference, `ceBgcolor`, to the calendar and sets it to `black`:

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

This URL deletes the calendar preference `ceBgcolor` from the user's preferences.

```
http://webcalendarserver/set_userprefs.wcap?id=b5q2o8ve2rk02nv9t6
&del_attrs=ceBgcolor
```

This URL would modify the calendar preference `ceBgcolor` to have the value `white`:

```
http://webcalendarserver/set_useprefs.wcap?id=b5q2o8ve2rk02nv9t6
&set_attrs=ceBgcolor=white
```

This URL would allow the logged-in administrator to modify the calendar preference `ceBgcolor` to have the value `black` for user `jdoe`:

```
http://webcalendarserver/set_userprefs.wcap?id=b5q2o8ve2rk02nv9t6&u
serid=jdoe&set_attrs=ceBgcolor=black
```

## storeevents

### Purpose

Add events to a calendar.

### Parameters

Table 10-35 lists this command's forty-three parameters:

**Table 10-35** `storeevents` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `alarmAudio` | ISO8601 Date Time Z string | The time at which to sound an audio alarm. | N | N / A |
| `alarmEmails` | semicolon-separated list of email addresses | Recipients of alarm notifications for the event. | N | N / A |
| `alarmFlashing` | ISO8601 Date Time Z string | The time at which to run flashing alarm. | N | N / A |
| `alarmPopup` | ISO8601 Date Time Z string | The time at which to pop up a dialog alarm. | N | N / A |
| `alarmStart` | ISO8601 DateTime Z string | The time at which to send the event alarm notification. | N | N / A |
| `attachments` | semicolon-separated list of strings | This is for iCalendar interoperability only. The strings are URLs. | N | N / A |
| `attendees` | semicolon-separated list of strings | The attendees of the event. | N | N / A |

**Table 10-35** `storeevents` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| brief | integer (`0`,`1`) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = brief<br>`0` = complete | N | `0` |
| `calid` | string | Calendar identifier in which to store the event. | Y | N / A |
| `categories` | semicolon-separated list of strings | The event categories. | N | N / A |
| `contacts` | semicolon-separated list of strings | Contacts for the event. | N | N / A |
| `desc` | string | Event purpose description. A string of any length. If not passed, `desc` is set to the `summary` value.<br><br>To include spaces in the string, use the code `%20`. | N | Value of `summary` parameter. |
| `dtend` | ISO8601 DateTime Z string | Event end time and date. | N | N / A |
| `dtstart` | ISO8601 DateTime Z string | Event start time and date. | | N / A |
| | | Not required to modify events. | N | |
| | | Required to create events. | Y | |
| `duration` | ISO8601 duration string | Event duration. If an event has both a `duration` and a `dtend`, the duration is ignored. | N | N / A |
| `exdates` | semicolon-separated list of ISO8601 DateTime Z strings | Event exclusionary recurrence dates. | N | N / A |

**Table 10-35** `storeevents` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `exrules` | semicolon-separated list of strings | Event exclusionary recurrence rules. A semicolon-separated list of recurrence-rule strings.<br><br>Each rule value must be enclosed in quotes. See “Recurrence Handling” on page 166. | N | N / A |
| fetch | integer (`0`,`1`) | A boolean indicating whether or not to fetch and return newly stored todos.<br><br>`1` = Fetch and return newly stored todos.<br>`0` = Do not fetch. | N | `0` |
| fmt-out | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `geo` | two semicolon-separated floats | Semicolon-separated string of two float numbers representing the event’s geographical location (latitude and longitude).<br><br>For example, `37.31;-123.2`. | N | `0;0` |
| `icsClass` | string | Event class.<br><br>One of the following values:<br><br>`PUBLIC`<br>`PRIVATE`<br>`CONFIDENTIAL` | N | `PRIVATE` |
| `icsUrl` | string | Event URL. | N | “” |
| `id` | unique identifier string | The session identifier. | Y | N / A |
| `isAllDay` | integer (0,1) | A boolean indicating whether or not the event lasts all day.<br><br>`1` = Lasts all day.<br>`0` = Does not last all day. | N | `0` |
| `language` | string | Language of event. (For example, “en”, “fr”, “de”) | N | N / A |
| `location` | string | Event location. | N | “” |

**Table 10-35** `storeevents` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| method | integer (1,2,4,8,16,32) | ITIP method for group scheduling.<br><br>1 = PUBLISH (organizer only uses this)<br>2 = REQUEST (organizer only uses this)<br>4 = REPLY (attendees only use this)<br>8 = CANCEL (organizer only uses this)<br>16 = MOVE<br>32 = COUNTER (attendees only use this) | N | 1(PUBLISH) |
| mod | integer | Specifies the recurrences to store / modify.<br><br>Not required for creating events.<br><br>Required to modify events.<br><br>One of the following values:<br><br>1 = THISINSTANCE<br>2 = THISANDFUTURE<br>3 = THISANDPRIOR<br>4 = THISANDALL | <br><br>N<br><br>Y | N / A |
| notify | integer (0,1) | This has been deprecated in 5.0 and remains only for 2.x compatibility. The Group Scheduling Engine (GSE) module takes care of all email notifications.<br><br>A boolean indicating whether or not to notify attendees of a change to an event.<br><br>1 = Notify attendees.<br>0 = Do not notify attendees. | N | 0 |
| orgEmail | email address | Email address of the event contact (usually the organizer). | N | N / A |
| orgUID | userid | The userid of the organizer. This is for Palm Synchronization. | N | N / A |
| priority | integer (0-9) | Event priority.<br><br>0 = lowest<br>9= highest | N | 0 |
| rchange | integer (0,1) | A boolean indicating whether or not to expand a recurring event.<br><br>1 = expand<br>0 = do not expand | N | 1 |

**Table 10-35** `storeevents` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `rdates` | semicolon-separated list of ISO8601 DateTime Z strings | Event recurrence dates. | N | N / A |
| `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 |
| `rid` | ISO8601 DateTime Z string | Event recurrence identifier. | | N / A |
| | | Not required to create events. | N | |
| | | Required to modify events. | Y | |
| `rrules` | semicolon-separated list of strings | Event recurrence rules. A semicolon-separated list of recurrence-rule strings. | N | N / A |
| | | Each rule value must be enclosed in quotes. See “Recurrence Handling” on page 166. | | |
| `seq` | integer | Event sequence number. | N | `0` |
| `status` | integer | The event status code. One of the following values: | N | N / A |
| | | `0 CONFIRMED`<br>`1 CANCELLED`<br>`2 TENTATIVE`<br>`3 NEEDS_ACTION`<br>`4 COMPLETED`<br>`5 IN_PROCESS`<br>`6 DRAFT`<br>`7 FINAL` | | |
| `summary` | string | Event summary. A string of any length. | | |
| | | Required for new events. | Y | N / A |
| | | Not required for modifying events. | N | `default summary` |
| | | To include spaces in the string, use the code `%20`. | | |

**Table 10-35** `storeevents` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| tzid | time zone ID string, such as "America/Los_Angeles" | The timezone with which to interpret all dates passed in to, or returned by, the server.<br>The default is coordinated universal time (UTC or Z format). | N | "GMT" |
| uid | string | Unique identifier of the event to be stored. | | |
| | | System generated for new events. | N | N/A |
| | | Required to modify events. | Y | default uid |

**Description.**

This command creates or modifies events with the specified attributes and stores them in the specified calendar in the database.

The command creates and stores recurrences as specified by the `rrules`, `exrules`, `rid`, `mod`, and `rchange` parameters. See "Recurrence Handling" on page 166.

When the `notify` value is `1`, it sends an IMIP `PUBLISH` message to all attendees of the event.

Use the `language` parameter to specify the language of the event. See "Choosing a Different Language or Character Set" on page 155 for a list of possible language codes.

The server does not support attachments. The `attachments` parameter exists to support iCalendar interoperability only, and is not functional.

**Returns**

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

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

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

**Error Codes**

This command cannot modify a linked event. The command will fail and return `CANNOT_MODIFY_LINKED_EVENTS(31)` in the `errno` array.

The command fails, and returns the error STORE_FAILED_DOUBLE_BOOKED(40), when it tries to store an event in a time slot that is already scheduled (double booking).

#### Required Parameters

This command creates new events and modifies existing events. There is a different set of parameter requirements for both cases:

- To create new events requires only two parameters:
  - dtstart
  - summary

  Every other parameter is optional. The server generates the uid.
- To modify existing events requires three parameters:
  - uid
  - rid
  - mod

  All other parameters are optional. If a parameter is not specified, the event will retain the previous value of the property.

#### Duration

Specify the duration in ISO8601 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).

**TIP** The ending date and time (dtend) overrides duration. If you specify both duration and dtend, the command ignores duration.

#### Example

For example, this URL would call storeevents.wcap and would result in storing an event in the calendar john,

```
http://webcalendarserver/storeevents.wcap?id=3423423asdfasf
&calid=john&dtstart=20000101T103000&dtend=20000101T113000&uid=001
&summary=new%20year%20event
```

The above example results in the following entry in an iCalendar database:

```
BEGIN:VEVENT
DTSTART:20000101T183000Z
DTEND:20000101T193000Z
UID:001
SUMMARY:new year event
END:VEVENT
```

#### Group Scheduling - Attendee Parameter

The two most important parameters for creating a group-scheduled event are `attendee` and `method`. The `attendee` parameter is a semicolon separated list of attendee entries. The attendee entry is explained in the section below.

Each attendee entry may contain several parameters, such as invitation participation status, whether attendance is required or not, etc. All such parameters are encapsulated in a syntax very similar to the ATTENDEE property defined in the iCalendar Specification (RFC 2445). Reading the entire document is recommended in order to have the necessary background information to understand the WCAP attendee syntax. There are some differences, such as, WCAP uses a different delimiter, "^", to set apart these parameters. (WCAP uses the standard iCalendar semicolon delimiter for separating attendees.)

For example, where iCalendar would have the following:

```
PARSTAT=ACCEPTED;RSVP=TRUE:mailto:abc@xyz.com
```

WCAP would format it this way:

```
PARSTAT=ACCEPTED^RSVP^=TRUE^mailto:abc@xyz.com
```

### *Examples of WCAP Attendee Entries*

If attendee A (attA) accepts an invitation, the WCAP command would contain:

```
PARTSTAT=ACCEPTED^RSVP=TRUE^attA
```

If attendee B (attB) declines an invitation, the WCAP command would contain:

```
PARTSTAT=DECLINED^RSVP=TRUE^attB
```

If the email attendee jdoe@xyz.com has not yet decided to attend and is not required to respond, the WCAP command would contain:

```
PARTSTAT=NEEDS-ACTION^RSVP=FALSE:mailto:jdoe@xyz.com
```

Table 10-36 lists the parameters in the iCalendar ATTENDEE property understood by WCAP. Most of the parameters are optional. Not all are fully supported by Calendar Server, although the information will be stored. For group scheduling, only the PARTSTAT and RSVP parameters are relevant.

**Table 10-36** iCalendar ATTENDEE parameters understood by WCAP

| Parameters | Purpose |
|---|---|
| PARTSTAT | The only required parameter. This shows the attendees participation status. |
| CUTYPE | Calendar user type. |
| MEMBER | List of groups the attendee is part of. WCAP has no understanding of these groups. |
| ROLE | Role of the attendee in this meeting. |
| RSVP | Attendee response required or not. |
| DELEGATED-TO | To whom the attendee delegates attendance. |
| DELEGATED-FROM | Attendee is a delegate for this person. |
| SENT-BY | The calendar user acting on behalf of the specified user. |
| CN | Display name of attendee. |
| DIR | Directory entry reference. |
| LANG | Language of the entry. |

### Group Scheduling - Method Parameter

The `method` parameter describes the type of message used: invitation, response, cancellation.

In an invitation, three types of messages may occur:

- An organizer invites attendees.

  When an organizer creates a meeting, there are two ways to invite people:

  - Send a `PUBLISH` message, creating or modifying a meeting, and notify the attendees. The `method` parameter is set to “1”.
  - Send a `REQUEST` message, creating or modifying a meeting, and requesting a response to the invitation from attendees. The `method` parameter is set to “2”.

  Only the organizer of the meeting can send a `PUBLISH` or `REQUEST` message.

- Attendees respond to invitation.

  An attendee sends a REPLY message, either accepting or declining the invitation. (The method parameter is set to “4”.)

- Organizer cancels the meeting.

  When an organizer cancels a meeting, attendees are notified by sending a CANCEL using one of the deleteevents commands. The method parameter is set to “8”.

**NOTE** The preferred way to handle a cancellation is to use one of the deleteevents commands, rather than storeevents.

The following set of examples demonstrates the WCAP commands for an organizer “org” to invite attendees “attA” and “attB” to a meeting. Attendee “attA” accepts the invitation; attendee “attB” declines it. The uid for the meeting is “event_u1”. The event will be created on both attendees’ calendars. Each will respond to the event on their own calendar. The response will be sent back to the organizer’s calendar by the iPlanet Calendar Server 5.0 Group Scheduling Engine.

The invitation:

```
storeevents.wcap?id=${SESSIONID of org}&calid=org&dtstart=
   20010201T200000Z&dtend=20010201T210000Z&summary=invite_attA_attB
   &method=2&attendees=PARTSTAT=ACCEPTED^RSVP=TRUE^org;PARTSTAT=
   NEEDS-ACTION^RSVP=TRUE^attA;PARTSTAT=NEEDS-ACTION^RSVP=
   TRUE^attB&fmt-out=text/xml
```

The acceptance:

```
storeevents.wcap?id=${SESSIONID ofattA}&calid=attA&uid=event_u1
   &method=4&attendees=PARTSTAT=ACCEPTED^RSVP=TRUE^attA
   &fmt-out=text/xml
```

The declined meeting:

```
storeevents.wcap?id=${SESSIONID ofattB}&calid=attB&uid=event_u1
   &method=4&attendees=PARTSTAT=DECLINED^RSVP=TRUE^attA
   &comments=I_cannot_make_it_Sorry&fmt-out=text/xml
```

## storetodos

### Purpose

Add one or more todos to a calendar.

### Parameters

Table 10-37 lists this command's the forty-four parameters:

**Table 10-37** `storetodos` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `alarmAudio` | ISO8601Date Time Z string | The time at which to sound an audio alarm. | N | N / A |
| `alarmEmails` | semicolon-separated list of email addresses | Recipients of alarm notifications for the todo. | N | N / A |
| `alarmFlashing` | ISO8601 Date Time Z string | The time at which to run a flashing alarm. | N | N / A |
| `alarmPopup` | ISO8601 Date Time Z string | The time at which to pop up a dialog alarm. | N | N / A |
| `alarmStart` | ISO8601 DateTime Z string | The time at which to send an alarm notification of the todo. | N | N / A |
| `attachments` | semicolon-separated list of strings | This parameter exists to support iCalendar interoperability only. The strings are URLs. | N | N / A |
| `attendees` | semicolon-separated list of strings | The todo attendees. | N | N / A |
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief output.<br>`0` = Complete output. | N | `0` |
| `calid` | string | Calendar identifier in which to store the todo. | Y | N / A |
| `categories` | semicolon-separated list of strings | The todo categories. | N | N / A |
| `completed` | ISO8601 DateTime Z string | Completion date of the todo. A value of `0` means the todo is not yet completed. | N | `0` |
| `contacts` | semicolon-separated list of strings | Contacts for the todo. | N | N / A |

**Table 10-37** `storetodos` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `desc` | string | Purpose of the todo. A string of any length.<br>If not passed, `desc` is set to the `summary` value.<br>To include spaces in the string, use the code `%20`. | N | Value in `summary` parameter. |
| `dtstart` | ISO8601 DateTime string | Start time and date of the todo. | | N/A |
| | | Not required to modify todos. | N | |
| | | Required to create todos. | Y | |
| `due` | ISO8601 DateTime Z string | End time and date of the todo. | N | N/A |
| `duration` | ISO8601 duration string | Todo duration. | N | N/A |
| `exdates` | semicolon-separated list of ISO8601 TimeDate Z strings | Exclusionary recurrence dates of the todo. | N | N/A |
| `exrules` | semicolon-separated list strings | Todo exclusionary recurrence rules. A semicolon-separated list of recurrence-rule strings.<br>Each rule value must be enclosed in quotes. See “Recurrence Handling” on page 166. | N | N/A |
| `fetch` | integer (`0`,`1`) | A boolean indicating whether or not to fetch and return newly stored todos.<br>`1` = Fetch and return newly stored todos.<br>`0` = Do not fetch todos. | N | `0` |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `geo` | two semicolon-separated floats | Semicolon-separated string of two float numbers representing the todo’s geographical location (latitude and longitude).<br>For example, 37.31;-123.2. | N | `0;0` |

**Table 10-37** `storetodos` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `icsClass` | string | Todo class. One of the following values:<br><br>`PUBLIC`<br>`PRIVATE`<br>`CONFIDENTIAL` | N | `PRIVATE` |
| `icsUrl` | string | Todo URL. | N | “” |
| `id` | unique identifier string | The session identifier. | Y | N / A |
| `isAllDay` | integer (`0`,`1`) | A boolean indicating whether or not it is an all day todo.<br><br>`1` = An all day todo.<br>`0` = Not an all day todo. | N | `0` |
| `language` | string | The language of the todo. (For example, “en”, “fr”, “de”.) | N | N / A |
| `location` | string | Todo location. | N | “” |
| `method` | integer (1,2,4,8,16,32) | ITIP method for group scheduling.<br><br>One of the following:<br><br>`1` = `PUBLISH`<br>`2` = `REQUEST`<br>`4` = `REPLY`<br>`8` = `CANCEL`<br>`16` = `MOVE`<br>`32` = `COUNTER` | N | `1 (PUBLISH)` |
| `mod` | integer | Specifies the recurrences to store / modify.<br><br>Not required for new todos.<br><br>Required to modify todos.<br><br>One of the following values:<br><br>`1 = THISINSTANCE`<br>`2 = THISANDFUTURE`<br>`3 = THISANDPRIOR`<br>`4 = THISANDALL` | <br><br>N<br><br>Y | N / A |

**Table 10-37** `storetodos` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `notify` | integer<br>0,1 | This parameter has been deprecated in 5.0. It remains to provide 2.x compatibility. The Group Scheduling Engine (GSE) handles sending of email notifications.<br>A boolean indicating whether or not to notify attendees of a changed todo.<br>`1` = Notify attendees of the change.<br>`0` = Do not notify attendees. | N | `0` |
| `orgEmail` | email address | The email address contact for the todo. (Usually the organizer's email.) | N | N / A |
| `orgUID` | `userid` | The `userid` of the organizer. This is for Palm Synchronization. | N | N / A |
| `percent` | integer (`0-100`) | Percentage completion of the todo. | N | `0` |
| `priority` | integer (`0-9`) | The priority of the todo.<br>`0` = Lowest priority.<br>`9`= Highest priority. | N | `0` |
| `rchange` | integer (0,1) | A boolean indicating whether or not to expand a recurring todo.<br>`1` = Expand the recurring todo.<br>`0` = Do not expand it. | N | `1` |
| `rdates` | semicolon-separated list of ISO8601 TimeDate Z strings | Recurrence dates of the todo. | N | N / A |
| `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 |
| `rid` | ISO8601 TimeDate Z string | Recurrence identifier of the todo.<br>Not required for new todos.<br>Required to modify todos. | <br>N<br>Y | N / A |
| `rrules` | semicolon-separated list of strings | Todo recurrence rules. A semicolon-separated list of recurrence-rule strings.<br>Each rule value must be enclosed in quotes. See "Recurrence Handling" on page 166. | N | N / A |

**Table 10-37** `storetodos` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `seq` | integer | Sequence number of the todo. | N | `0` |
| `status` | integer | A code for the status of the todo. One of the following values:<br>`0 CONFIRMED`<br>`1 CANCELLED`<br>`2 TENTATIVE`<br>`3 NEEDS_ACTION`<br>`4 COMPLETED`<br>`5 IN_PROCESS`<br>`6 DRAFT`<br>`7 FINAL` | N | N/A |
| `summary` | string | Todo summary. A string of any length. | | |
| | | Required for new todos. | Y | `default summary` |
| | | Not required for modifying todos. | N | N/A |
| | | To include spaces in the string, use the code `%20`. | | |
| `tzid` | time zone ID string, such as "America/Los_Angeles" | A timezone. All dates are interpreted with reference to this timezone.<br>If not passed, all dates are interpreted as being in coordinated universal time (UTC or Z format). | N | "GMT" |
| `uid` | string | Unique identifier of the todo to be stored. | | |
| | | System generated for new todos. | N | N/A |
| | | Required to modify todos. | Y | `default uid` |

**Description.**

Use this command to create and modifies todos with the specified attributes and stores them in the specified calendar in the database.

The command creates and stores recurrences as specified by `rrules`, `exrules`, `rid`, `mod`, and `rchange` parameters. See "Recurrence Handling" on page 166.

When the `notify value` is `1`, it sends an IMIP `PUBLISH` message to the email attendees of the todo.

The server does not support attachments. The `attachments` parameter exists to support iCalendar interoperability only, and is not functional.

#### `method` Parameter

For group scheduling, specify one of the following ITIP methods:

| | | |
|---|---|---|
| 1 | PUBLISH | Used only by the organizer. |
| 2 | REQUEST | Used only by the organizer. |
| 4 | REPLY | Used only by attendees. |
| 8 | CANCEL | Used only by the organizer. |
| 16 | MOVE | N / A |
| 32 | COUNTER | Used only by attendees. |

### Returns

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

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

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

### Error Codes

This command cannot modify a linked todo. The command will fail and return an error of `CANNOT_MODIFY_LINKED_TODOS(32)` in `errno`.

The command fails, and returns the error `STORE_FAILED_DOUBLE_BOOKED(40)`, when it tries to store a todo in a time slot that is already scheduled (double booking).

### Required Parameters

This command creates new todos and modifies existing todos. There is a different set of parameter requirements for both cases:

- To create new todos requires only two parameters:
  - `dtstart`
  - `summary`

  Every other parameter is optional. The server generates the `uid`.
- To modify existing todos requires three parameters:

- uid
- rid
- mod

All other parameters are optional. If a parameter is not specified, the todo will retain the previous value of the property.

#### Duration

The due date and time (due) overrides duration. If you specify both duration and due, the command ignores duration.

Specify the duration in the ISO8601 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).

## upload_file

#### Purpose

Uploads a file to the server. Used in conjunction with write_file for file import.

#### Parameters

Table 10-38 lists this command's one parameter:

**Table 10-38** upload_file Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| idt | unique identifier string | Session ID. | Y | N/A |

#### Description.

This command used in conjunction with write_file, handles file importing for iPlanet Calendar Server 2.0 compatibility. It only works through a POST command since a file is attached with it.

File import is handled in the following way through the UI:

- Upload the user's files from the user's computer to the server using the `upload_file` command. The `upload_file` command must be a `POST`.
- Take the uploaded file(s) and write them to the database using the `write_file` command.

The reason for this separation is that the UI cannot combine both parameters' values and file attachment through the JavaScript UI. Also, this separation allows for uploading multiple files to the server before actually writing them to the database.

#### upload_file Example

The data sent to `upload_file` should look exactly like `import.wcap` data. The only files that can be supported are iCalendar and XML files. Although the user can upload any file, only iCalendar and XML files can be successfully added to the database.

The returned JavaScript from `upload.wcap` looks like the following:

```
Completed sending of import HTTP/1.1 200
Date: Tue, 05 Oct 2000 23:37:51 GMT
Content-type: text/html; charset=iso-8859-1
Content-length: 301
Last-modified: Tue, 05 Oct 2000 23:37:51 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache
Connection: Keep-Alive

<html><head><script>
function color(s) { if (s) document.bgColor=s
}
var
filename='s://ns//server//msg//calendar//core//parser//test//ical1.
ics'

var uniquefilename='cf8bfa37b347000001000000ef000000'

var size=640
var errno=0
var errstr=''
</script></head>
<body onLoad=parent.uploadCB()>
</body></html>
```

### write_file Example

Notice the bold line containing the *uniquefilename* JavaScript variable in the previous example. This variable should be used as the input to the uniquefilename parameter in the write_file command.

After the UI has uploaded the appropriate files using upload_file, the UI should call write_file to actually write the file to the database.

For example, use this URL to write the previous example's upload call to the calendar jdoe. Notice that the uniquefilename parameter matches the *uniquefilename* JavaScript variable returned in upload_file.wcap.

```
http://webcalendarserver/write_file.wcap?id=abc&calid=jdoe&dtstart=
0&dtend=0&content-in=text/calendar&uniquefilename=cf8bfa37b34700000
1000000ef000000
```

To write two files to the server, put semicolons between the unique filenames. The files must be of the same content-type.

```
http://webcalendarserver/write_file.wcap?id=abc&calid=jdoe&dtstart=
0&dtend=0&content-in=text/calendar&uniquefilename=cf8bfa37b34700000
1000000ef000000;dg9cgb48c4580000010000000ab000000
```

## version

### Purpose

To get the current WCAP version.

### Parameters

Table 10-39 lists this command's one parameter:

**Table 10-39** version Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| fmt-out | string | The format for the returned data.<br>The three format types:<br>text/calendar<br>text/xml<br>text/js | N | text/js |

### Description.

This command gets the current WCAP version. (Note: this is different from the server version as well as the HTTP version.)

### Returns

The commands supports output types of iCalendar, 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.

### Example

The following examples are for each of output data types.

- JavaScript: (URL:/version.wcap)

```
<html><head><script>
function color(s) { if (s) document.bgColor=s
}
var id='0'
var userid=''
var calid=''
var errno=new Array()
var wcapversion=1.0.0
parent.ceCB(window.name)
</script></head></html>
```

- iCalendar: (URL: /version.wcap?fmt-out=text/calendar)

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//iPlanet/Calendar Hosting Server//EN
X-NSCP-WCAPVERSION:WCAP
END:VCALENDAR
```

- XML: (URL: /version.wcap?fmt-out=text/xml)

```
<?xml version="1.0" encoding="UTF-8">
<iCalendar>
<iCal>
<X-NSCP-WCAPVERSION>1.0.0</X-NSCP-WCAPVERSION>
</iCal>
</iCalendar>
```

## write_file

### Purpose

Writes a file to the database.

### Parameters

Table 10-40 lists this command's six parameters:

**Table 10-40** `write_file` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `calid` | string | Command imports file to this calendar. | Y | N/A |
| `content-in` | string | The input-content data type. | Y | N/A |
| `dtend` | ISO8601 DateTime "Z" string | The end range of events to import.<br>A value of `0` causes importing of all events up to the end of time. | N | `0` |
| `dtstart` | ISO8601 DateTime "Z" string | The start range of events to import.<br>A value of `0` causes importing of all events from the beginning of time. | N | `0` |
| `id` | unique identifier string | Session ID. | Y | N/A |
| `uniquefilename` | semicolon-separated list of strings | The filenames to import from the server. | Y | N/A |

### Description.

The `write_file` command, used in conjunction with the `upload_file` command, handles importing files. The parameters correspond exactly to those of the import command. See `upload_file` for an example and a detailed description of the file import process.

# Index

## SYMBOLS



## A

## C

## D



## E

## F

## G



## H



## I



## L



## M



## N

## O



## P



## Q



## R

## S



## T

## U



## V



## W