Programmer's Reference Manual

Sun™ ONE Calendar Server

Version 6.0

816-6711-10
December 2003

Copyright © 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.
Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.sun.com/patents and one or more additional patents or pending patent applications in the U.S. and in other countries.
THIS PRODUCT CONTAINS CONFIDENTIAL INFORMATION AND TRADE SECRETS OF SUN MICROSYSTEMS, INC. USE, DISCLOSURE OR REPRODUCTION IS PROHIBITED WITHOUT THE PRIOR EXPRESS WRITTEN PERMISSION OF SUN MICROSYSTEMS, INC.
U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.
This distribution may include materials developed by third parties.
Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and in other countries, exclusively licensed through X/Open Company, Ltd.
Sun, Sun Microsystems, the Sun logo, Java, Solaris, JDK, Java Naming and Directory Interface, JavaMail, JavaHelp, J2SE, iPlanet, the Duke logo, the Java Coffee Cup logo, the Solaris logo, the SunTone Certified logo and the Sun ONE logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.
All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc.
Legato and the Legato logo are registered trademarks, and Legato NetWorker, are trademarks or registered trademarks of Legato Systems, Inc. The Netscape Communications Corp logo is a trademark or registered trademark of Netscape Communications Corporation.
The OPEN LOOK and Sun(TM) Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.
Products covered by and information contained in this service manual are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.
DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright © 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés.
Sun Microsystems, Inc. détient les droits de propriété intellectuels relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plus des brevets américains listés à l'adresse http://www.sun.com/ patents et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats - Unis et dans les autres pays.
CE PRODUIT CONTIENT DES INFORMATIONS CONFIDENTIELLES ET DES SECRETS COMMERCIAUX DE SUN MICROSYSTEMS, INC. SON UTILISATION, SA DIVULGATION ET SA REPRODUCTION SONT INTERDITES SANS L AUTORISATION EXPRESSE, ECRITE ET PREALABLE DE SUN MICROSYSTEMS, INC.
Cette distribution peut comprendre des composants développés par des tierces parties.
Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.
Sun, Sun Microsystems, le logo Sun, Java, Solaris, JDK, Java Naming and Directory Interface, JavaMail, JavaHelp, J2SE, iPlanet, le logo Duke, le logo Java Coffee Cup, le logo Solaris, le logo SunTone Certified et le logo Sun[tm] ONE sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.
Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.
Le logo Netscape Communications Corp est une marque de fabrique ou une marque déposée de Netscape Communications Corporation.
L'interface d'utilisation graphique OPEN LOOK et Sun(TM) a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une license non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui, en outre, se conforment aux licences écrites de Sun.
Les produits qui font l'objet de ce manuel d'entretien et les informations qu'il contient sont regis par la legislation americaine en matiere de controle des exportations et peuvent etre soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucleaires, des missiles, des armes biologiques et chimiques ou du nucleaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou reexportations vers des pays sous embargo des Etats-Unis, ou vers des entites figurant sur les listes d'exclusion d'exportation americaines, y compris, mais de maniere non exclusive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une facon directe ou indirecte, aux exportations des produits ou des services qui sont regi par la legislation americaine en matiere de controle des exportations et la liste de ressortissants specifiquement designes, sont rigoureusement interdites.
LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFACON.

Contents

About This Reference **9**

Who Should Read This Book 9
What You Need to Know 10
How This Book is Organized 10
Document Conventions 11
Monospaced Font 11
Italicized Font 11
Square or Straight Brackets 12
Platform-specific Syntax 12
Where to Find Related Information 12
Where to Find This Book Online 12
Related Third-Party Web Site References 13

Chapter 1 Calendar Server API (CSAPI) Overview **15**
CSAPI Architecture 16
Thread Safe Requirement 17
Dependencies 18
Using CSAPI 18
Loading CSAPI Modules 18
Plug-in Interfaces 19
Client and Server APIs 20
CSAPI Samples 22

Chapter 2 CSAPI Reference **23**
csIAccessControl 23
CheckAccess 24
Init 25

csIAuthentication 26
ChangePassword 27
Init 28
Logon 28
Logout 29
VerifyUserExists 30
csICalendarLookup 30
DeleteHostnameForCalid 32
FreeCalid 32
FreeType 33
GetHostnameForCalid 33
Init 34
QualifyCalid 34
QueryType 35
SetHostnameForCalid 36
csIDataTranslator 36
GetSupportedContentTypes 37
Init 38
Translate 39
csIPlugin 39
GetDescription 40
GetVendorName 41
GetVersion 41
Init 42
csIQualifiedCalidLookup 42
FindCalid 43
Init 43
csIUserAttributes 44
FreeAttribute 44
GetAttribute 45
Init 46
SetAttribute 47
csICalendarServer 47
GetVersion 48
Init 48
csIMalloc 49
Calloc 49
Free 50
FreeIf 50
Init 51
Malloc 51
Realloc 52

Chapter 3 Proxy Authentication SDK Overview 53
Who Will Use the authSDK? 53
What Is the authSDK? 53
Architecture 54
Initialization 54
Lookup 54
Cleanup 54
Functions Overview 55

Chapter 4 Proxy Authentication SDK Reference 57
Proxy Authentication SDK Functions 57
CEXP_GenerateLoginURL 58
CEXP_GetVersion 58
CEXP_Init 59
CEXP_SetHttpPort 60
CEXP_Shutdown 60
How to Use the authSDK 60
Other Tips 61

Chapter 5 Web Calendar Access Protocol (WCAP) Overview 63
Introduction 63
Command Overview 64
Session Identifiers 66
Hosted (Virtual) Domain Mode 66
Command Formats 67
Client Request Formats 67
Server Response Formats 68

Chapter 6 WCAP Commands 69
Common Topics List 69
WCAP Commands List 70
Common Topics 72
Access Control Entries 72
Application IDs (appid parameter) 76
Changing Language or Character Set 77
Encoded Characters 79
Error Handling 79
Fetching Component Data 84
Fetching Component State Data 84
Fetching Deleted Data 85
Fetching Recurrence Data 85
Formatting Standards 86

Freebusy Calendars 86
Freebusy Calculation for Private Events 88
Group Scheduling 88
Output Format 92
Recurring Components – Overview 92
Recurring Components – Creating, Modifying 93
Recurring Components – Deleting 97
Recurring Components – Fetching 98
Time Zones 99
Updating Parameter Values 100
X-Tokens 101
WCAP Commands 104
addlink 105
change_password 106
check_id 107
createcalendar 108
deletecalendar 110
deletecomponents_by_range 112
deleteevents_by_id 113
deleteevents_by_range 116
deletetodos_by_id 117
deletetodos_by_range 120
export 121
fetchcomponents_by_alarmrange 124
fetchcomponents_by_attendee_error 132
fetchcomponents_by_lastmod 135
fetchcomponents_by_range 138
fetch_deletedcomponents 148
fetchevents_by_id 154
fetchtodos_by_id 157
get_all_timezones 163
get_calprops 167
get_freebusy 170
get_guids 173
gettime 174
get_userprefs 176
import 179
login 182
logout 184
ping 184
search_calprops 185
set_calprops 188
set_userprefs 192

storeevents 193
storetodos 202
verifyevents_by_ids 209
verifytodos_by_ids 211
version 213

Glossary 215

Index 221

About This Reference

This document of gives detailed instructions on the use of the following Sun™ Open Net Environment (Sun ONE) Calendar Server APIs and protocol that you may use to customize your server installation:

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

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
- Related Third-Party Web Site References

Who Should Read This Book

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

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
- RFC 2445, RFC 2446, RFC 2447

  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 Sun ONE 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, "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 2, "CSAPI Reference"

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

- Chapter 3, "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 Sun ONE Calendar Server.

- Chapter 4, “Proxy Authentication SDK Reference”

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

- Chapter 5, “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 6, “WCAP Commands”

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

- Glossary

Document Conventions

Monospaced Font

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

Italicized Font

Italicized font 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.

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

`cal_svr_base``/...`

The Calendar Server Base (*`cal_svr_base`*) represents the directory path in which you install the server. The default value of the *`cal_svr_base`* is `/opt/SUNWics5/cal`.

Square or Straight Brackets

Square (or straight) brackets [] are used to enclose optional parameters.

Platform-specific Syntax

All paths specified in this manual are in UNIX® format.

Where to Find Related Information

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

- *Sun ONE Messaging and Collaboration Event Notification Service Manual*
- *Sun ONE Calendar Server Administrator's Guide*
- *Sun ONE Calendar Server Installation Guide*
- *Sun ONE Messaging and Collaboration Schema Reference Manual*

Where to Find This Book Online

You can find the Calendar Server Programmer's Manual online in PDF and HTML formats. To find this book and all the Sun ONE Calendar Server documentation, use this URL:

`http://docs.sun.com/coll/S1_CalendarServer_60`

Related Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

NOTE Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Chapter 1

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 Sun™ ONE Calendar Server. CSAPI allows you to create very fast runtime shared objects that outperform both system executables and scripts in any language, with respect to speed, memory footprint, and load. All of these factors contribute to scalability issues in high-end systems.

This chapter has the following sections:

- CSAPI Architecture
  - 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.

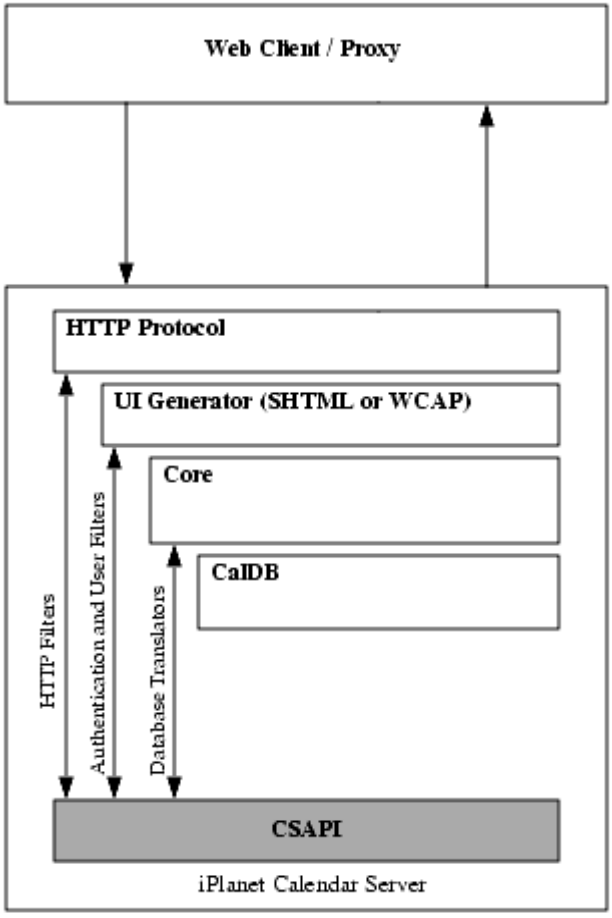
Figure 1-1, which follows, shows the relationship of CSAPI modules to other subsystems within Calendar Server. Depending on which functional group or groups a CSAPI module supports, it can interact with one or more areas of Calendar Server functionality, such as data formatting, authentication, and directory 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 2, "CSAPI Reference." Each client-side interface addresses a functional area of Calendar Server. The implementation contained in a module can either augment or override the native Calendar Server functionality in its area.

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 1-1 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 Netscape™ Portable Runtime (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/projects/nspr/reference/html/index.html`

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. 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

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 *Sun ONE Calendar Server Administrator's 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`".

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

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.

Plug-in Version Numbers

Each default plug-in interface can have a different version number. Version numbers increment by a whole number when the API is updated by Sun Microsystems, Inc. All custom plug-ins must use the methods in the current version of the default plug-in API.

If you created a custom plug-in based on an earlier default plug-in version, you must update your custom plug-in to use the new version of all updated methods, and you must increment its version number to reflect the current version number of the default plug-in.

The system will not load plug-ins with version numbers below the current default version number. Your plug-in must have a version number greater than the current default plug-in, but less than the next whole number. For example, if you are writing, or have created in the past, a custom plug-in for `csIDatabaseLookup`, which is currently at version 2.0, your plug-in version number must be greater than 2.0 and less than 3.0.

Table 1-1 lists the current version of each plug-in API.

Table 1-1 Current Plug-in API Versions

Plug-in API	Current Version of Default Plug-in
csIAccessControl	1.0
csIAuthentication	1.0
csICalendarLookup	2.0
csIDataTranslator	2.0
csIPlugin	1.0
csIQualifiedCalidLookup	1.0
csIUserAttributes	2.0
csICalendarServer	1.0
csIMalloc	1.0

Client and Server APIs

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

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

Table 1-2 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.
csIQualifiedCalidLookup	Retrieves a calendar ID for the specified qualified URL.
csIUserAttributes	Augments or overrides the mechanism for storing and retrieving user attributes.

The interfaces are described in detail in Chapter 2, “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 1-3:

Table 1-3 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 1-4 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: On Solaris, it uses the `pam` library to authenticate against the `local /etc/passwd` file or NIS. On Windows NT, it uses the WIN32 API `LogonUser`, which authenticates against Microsoft clients. In order for this sample to work properly on NT, the administrator must enable the privilege for users to log on via batch jobs. You can do this from within the UserManager Administrative Tool, under the Policies/User Rights section.
`DataTranslator`	This sample overrides the default format translation of incoming and outgoing data. It shows how to convert `icalendar` data into 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 2

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 2-1 and Table 2-2 to augment or override Sun™ ONE Calendar Server's default behavior:

Table 2-1 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 2-2 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 Sun ONE 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. One of the following constants: NS_CONTINUE_DEFAULT_PROCESSING 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:

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
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. One of the following constants: • `NS_CONTINUE_DEFAULT_PROCESSING` • `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. One of the following constants: • NS_CONTINUE_DEFAULT_PROCESSING • 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 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. One of the following constants: • NS_CONTINUE_DEFAULT_PROCESSING • NS_OVERRIDE_DEFAULT_PROCESSING

Returns

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

Description

None.

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. One of the following constants: • `NS_CONTINUE_DEFAULT_PROCESSING` • `NS_OVERRIDE_DEFAULT_PROCESSING`

Returns

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

Description

None.

csICalendarLookup

Implement the methods in this interface to augment or override the default calendar lookup (LDAP). There are two types of calendar lookup provided in Sun ONE Calendar Server: algorithmic or LDAP. LDAP is the default type. Some of the methods in this plug-in are used only by the LDAP-type lookup.

The table that follows lists the methods for this plug-in, tells which type of lookup it supports: Algorithmic (Alg), or LDAP, and gives a description of the method.

Methods

The csICalendarLookup implements the following methods:

Method	Type	Description
DeleteHostnameForCalid	LDAP only	Removes the host name entry from the calendar lookup database for a specific calendar.
FreeCalid	both Alg and LDAP	Frees a previously allocated, fully qualified calid.
FreeType	both Alg and LDAP	Frees a previously allocated type.
GetHostnameForCalid	LDAP only	Gets the hostname from the calendar lookup database associated with the calendar specified by the calid
Init	both Alg and LDAP	Confirms that the interface was found and registered.
QualifyCalid	both Alg and LDAP	Returns the name of the qualified calid.
QueryType	both Alg and LDAP	Queries the type of plug-in.
SetHostnameForCalid	LDAP only	Sets the host name for a calendar user associated with a calid being created.

Description

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

piReturnCode Values

There are four possible return codes for the piReturnCode parameter:

- Cannot connect to LDAP server
- Insufficient privileges to modify LDAP entry
- Invalid credentials to connect to LDAP
- No value for back end host

DeleteHostnameForCalid

Purpose

Removes the hostname in LDAP associated with the specified `calid`.

Syntax

```
PRInt32 GetHostnameForCalid(char* psCalid, PRInt32 *piReturnCode) = 0;
```

Parameters

The following are the parameters and definitions for this method:

`psCalid`	`calid` for which the hostname is requested.
`piReturnCode`	`0`= successful, non-zero indicates failure.

Returns

Returns zero for success; a non-zero code for failure. See piReturnCode Values.

Description

Removes the hostname associated with the specified calendar in the calendar lookup database. This is a no-op for the algorithmic implementation.

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
Used by both implementations of lookup.

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. Used by both implementations.

GetHostnameForCalid

Purpose
Retrieves the hostname associated with the specified calid.

Syntax
```
PRInt32 GetHostnameForCalid(char* psCalid, char** ppsHost, PRInt32
                            *piReturnCode) = 0;
```

Parameters
The following are the parameters and definitions for this method:

psCalid	calid for which the hostname is requested.
ppsHost	The hostname to be returned.
piReturnCode	0= successful, non-zero indicates failure.

Returns

Returns zero for success; a non-zero code for failure. See piReturnCode Values.

Description

Gets the hostname associated with a calendar in the calendar lookup database. This is a no-op for the algorithmic implementation.

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* psCalid, char** ppsQualifiedCalid, PRInt32
                      *piReturnCode) = 0;
```

Parameters

The method has the following parameters:

psCalid	The calid to be qualified.
ppsQualifiedCalid	On return, contains the URL of the qualified calid.
piReturnCode	0= successful, non-zero indicates failure.

Returns

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

Description

Returns the name of the qualified calid. The qualified calid format is:

dwp://*back-end host*[:port]/psCalid

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.

SetHostnameForCalid

Purpose

Not currently used. Sets the host name for a calendar user.

Syntax

```
PRInt32 SetHostnameForCalid(char* psCalid, char* ppsHost, PRInt32
                            *piReturnCode) = 0;
```

Parameters

This method uses the following parameters:

`psCalid`	`calid` for which the hostname is to be set.
`ppsHost`	The hostname to be set.
`piReturnCode`	`0`= successful, non-zero indicates failure.

Returns

Zero for success; non-zero for failure. See

Description

Sets the hostname for a `calid` that is about to be created. This is a no-op for the algorithmic implementation.

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`	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.

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

MIME Type	Description
`text/calendar`	iCalendar
`text/xml`	iCalendar in XML

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. One of the following constants: NS_CONTINUE_DEFAULT_PROCESSING 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.
	One of the following constants:
	`NS_CONTINUE_DEFAULT_PROCESSING`
	`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. For a list of the current version numbers for each plug-in API, seeTable 1-1 Current Plug-in API Versions in Chapter 1, “Calendar Server API (CSAPI) Overview.”.
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. One of the following constants: NS_CONTINUE_DEFAULT_PROCESSING 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. One of the following constants: • NS_CONTINUE_DEFAULT_PROCESSING • 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. One of the following constants: • NS_CONTINUE_DEFAULT_PROCESSING • 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

```
PRUint32 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. One of the following constants: • NS_CONTINUE_DEFAULT_PROCESSING • 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 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 3

Proxy Authentication SDK Overview

This chapter describes the Calendar Server 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 Sun ONE Calendar Server. When a portal system authenticates a user, authSDK functions notify Sun ONE 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 *cal_svr_base*/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 4, "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 administrator'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 3-1.

Table 3-1 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 Calendar Server 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

There are five authSDK functions. They are presented below in alphabetical order, not in order of use:

- 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.

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 Sun ONE Calendar Server administrator’s LDAP user ID.
`pszAdminPassword`	A string containing the Sun ONE Calendar Server administrator’s password.

Returns

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

Comment

If the bind DN (`pszLdapBindUser`) and password (`pszLdapBindPass`) 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 Sun ONE 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 Sun ONE 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 5

Web Calendar Access Protocol (WCAP) Overview

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

- Introduction
- Command Overview
- Command Formats

Introduction

The default client UI protocol for Calendar Server is an SHTML protocol, and is private. To retrieve calendar data in `text/calendar` and `text/xml` formats, use WCAP commands with the `fmt-out` parameter set to `text/calendar` or `text/xml`.

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

WCAP 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.
- `text/xml` format.

NOTE In order to start the Calendar Server user interface (Calendar Express), you must specify `fmt-out=text/html` in the `login` command. This is the only instance of this format type.

WCAP commands consist of four general categories of usage:

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

Command Overview

Table 5-1 describes the high level list of commands supported in WCAP. For a detailed description of each command, see Chapter 6, "WCAP Commands."

Table 5-1 WCAP Command Overview

WCAP Command	Description
`addlink`	Add event links from one calendar to another.
`change_password`	Change the user's password.
`check_id`	Administrator only: Check if user's session ID is 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.

Table 5-1 WCAP Command Overview *(Continued)*

WCAP Command	Description
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.
fetch_deletedcomponents	Queries the deletelog database for deleted components.
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.
gettime	Returns the server times for the requested calids.
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.
verifyevents_by_ids	Fetches events and returns the uid/rid of events not in the database.
verifytodos_by_ids	Fetches todos and returns the uid/rid of events not in the database.

Table 5-1 WCAP Command Overview *(Continued)*

WCAP Command	Description
version	Returns the WCAP version that the server supports.

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. 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 2, "CSAPI Reference," for the section "csIAccessControl."

Hosted (Virtual) Domain Mode

If your LDAP is set up to use hosted (virtual) domains, all WCAP commands you issue must have fully qualified user IDs and calendar IDs (calid), for example jdoe@example.com.

In order to be in hosted domain mode, the following ics.conf parameter must be set as shown:

```
service.virtualdomain.support="y"
```

See your Calendar Server administrator if you do not know whether you are using hosted domains.

The following two example WCAP commands demonstrate the difference between calid values for non-hosted domain mode and hosted domain mode.

Not in hosted domain mode:

```
http://webcalendarserver/get_userprefs.wcap?id=b5q2o8ve2rk02nv9t6&
calid=jdoe&fmt-out=text/calendar
```

In hosted domain mode:

```
http://webcalendarserver/get_userprefs.wcap?id=b5q2o8ve2rk02nv9t6&
calid=jdoe@example.com&fmt-out=text/calendar
```

Command Formats

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

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

WCAP in Calendar Server consists of JavaScript objects formatted as XML or iCalendar, communicated as HTML documents over HTTP on both the client and server side. Refer to the *Sun ONE Calendar Server 6.0 Release Notes* for recommended browser versions for client interfaces.

NOTE There is a limit to the number of characters that may be passed in for each parameter. The limit per parameter is 1024 characters.

Client Request Formats

Clients submit command requests to the Calendar Server in either Universal Resource Identifier (URI) data format, or with one of three HTML forms.

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`

See also “Common Topics List” on page 69, in Chapter 6, “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.

NOTE The maximum length for WCAP parameters is 1024 characters.

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

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 either text / calendar or text / xml formats. The following table gives a brief description of each response format:

Response Format	Description
HTML with JavaScript objects - text/calendar	Responses are in HTML, containing JavaScript formatted as iCalendar objects. This is the default format.
HTML with JavaScript objects - text/xml	Responses are in HTML, containing JavaScript formatted as XML objects

Chapter 6

WCAP Commands

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

Common Topics List

There are certain issues that span multiple commands. These issues are called common topics. The table that follows lists and contains links to these topics:

Access Control Entries

Application IDs (appid parameter)

Changing Language or Character Set

Encoded Characters

Error Handling

Fetching Component Data

Fetching Component State Data

Fetching Deleted Data

Fetching Recurrence Data

Formatting Standards

Freebusy Calendars

Freebusy Calculation for Private Events

Group Scheduling

Output Format

Recurring Components – Overview

Recurring Components – Creating, Modifying

Recurring Components – Deleting

Recurring Components – Fetching

Time Zones

Updating Parameter Values

X-Tokens

WCAP Commands List

WCAP commands accept various parameters, which are defined for each command in the "WCAP Commands" section. Unless otherwise noted, the maximum length value for any parameter accepted by WCAP commands is 1024 characters. While no input length checking is performed by WCAP, any parameter value longer than 1024 can produce unpredictable results.

Table 6-1 lists the WCAP commands in alphabetical order.

Table 6-1 WCAP Commands

`addlink`	Add event links from one calendar to another.
`change_password`	Change the user's password.
`check_id`	Administrator only: Check if user's session ID is 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.
`fetch_deletedcomponents`	Queries the deletelog.db for deleted components that match the criteria.
`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.

Table 6-1 WCAP Commands *(Continued)*

get_calprops	Returns calendar properties.
get_freebusy	Returns calendar freebusy time.
get_guids	Returns a set of random UIDs.
gettime	Gets the server time for the requested calendars.
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 event by passing properties in a URL.
storetodos	Stores todos that are specified in the application/urlencoded manner.
verifyevents_by_ids	Fetches events and returns the uid/rid of events not in the database.
verifytodos_by_ids	Fetches todos and returns the uid/rid of events not in the database.
version	Returns the WCAP version that the server supports.

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 single calendar. Collectively, all the ACE strings that apply to a calendar are called an Access Control List (ACL). As the system searches the ACL list for a calendar, 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 *Sun ONE Calendar Server Administrator’s Guide* for further information on configuration settings.

The `get_userprefs` command will fail if any node does not have “allow anyone” access rights for reading and searching. For example, the following LDAP modify record for an ACI entry gives the correct privileges to make the command work correctly.

```
dn: o=sesta.com
changetype: modify
add: aci
aci:  (target="ldap:///o=sesta.com")(targetattr="*")(version 3.0;
acl "Allow anyone"; allow (read,search) userdn = "ldap:///anyone";)
```

Due to a limitation on the user interface, do not add ACE strings for more than 75 users per calendar.

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 6-2 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 6-2 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.

Table 6-2 Access Control Characters

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.
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 6-2.
- 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 a calendar (*@domainname* a sfr g), to grant owners 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^a^sfr^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.

Mapping User Interface Operations to ACLs

Table 6-3 shows the ACLs necessary to achieve the desired user interface operation.

Table 6-3 Mapping User Interface Operations to ACLs

User Interface Operation	ACL Required	Example	Description
Delete Events and Todos	Modify Events and Todos + Delete Components or Delete Calendar	c^d^g or, a^d^g	To delete events or todos, you need modify permission, and either delete components or delete calendar permission.
Freebusy	Freebusy Components or Freebusy Calendar	c^f^g a^f^g	To view a freebusy representation of a calendar (the events and todos), you need freebusy components or freebusy calendar permission.

Table 6-3 Mapping User Interface Operations to ACLs

User Interface Operation	ACL Required	Example	Description
Modify Events and Todos	Read Events and Todos + Write Components or Write Calendar	c^w^g a^w^g	To modify components of a calendar (events and todos), you need read permission, and either write components or write calendar permission.
Read Events on a Calendar	Read Calendar	a^r^g	To read components, you must have read calendar permission. Note that read components permission (c^r^g) will not work.
Schedule (Invite)	Schedule Calendar	a^s^g	To invite someone, you need schedule calendar permission.
Subscribe	Read Properties	p^r^g	To subscribe to a calendar, you must have read properties permission.

Application IDs (appid parameter)

The following WCAP commands accept the `appid` parameter:

- `deletecomponents_by_range` – (ENS notifications not yet implemented)
- `deleteevents_by_id`
- `deleteevents_by_range` – (ENS notifications not yet implemented)
- `deletetodos_by_id`
- `deletetodos_by_range` – (ENS notifications not yet implemented)
- `import` – (ENS notifications not yet implemented)
- `storeevents`
- `storetodos`

This WCAP command parameter is used to set the value of an X-Token that ENS returns with notifications.

Applications passing this parameter in with the appropriate WCAP command can detect which ENS notifications they originated by checking the value of the X-Token `X-NSCP-COMPONENT-SOURCE`. Note that this X-Token is not returned by WCAP commands, only ENS notifications.

This parameter is a runtime parameter. That is, nothing is stored in the database.

If appid is present, the Event Notification Service (ENS) returns the value of appid as the value of the X-Token X-NSCP-COMPONENT-SOURCE. If the appid parameter is missing, ENS assigns one of the standard values to the X-Token (WCAP, CALENDAR EXPRESS, ADMIN). (Note that ADMIN is not yet implemented.)

Table 6-4 shows the effect of the presence of the appid parameter on the value of the X-Token X-NSCP-COMPONENT-SOURCE. For more information about ENS, see the *Sun ONE Messaging and Collaboration Event Notification Service Manual*.

Table 6-4 Presence of appid and Value of X-Token X-NSCP-COMPONENT-SOURCE

appid Present?	Value of X-Token X-NSCP-COMPONENT-SOURCE (with Request Origin)
no	WCAP (default) CALENDAR EXPRESS (from UI) ADMIN (from Admin tools) – Not yet implemented
yes	Value of appid

NOTE For the Calendar Server, ENS notifications are only returned for some of the commands (as noted earlier). The other commands will be implemented in a later release.

Changing 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 *Sun ONE Calendar Server 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 Standards" on page 86 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 Sun ONE representative to find out which languages are currently supported by the server.

For example, enter the following if you want to insert an event into the calendar:

```
storeevents.wcap?id=${SESSIONID}&calid=id&summary=summary&
location=location&desc=desc&charset=euc-jp
```

As another example, suppose that the location value is two Japanese characters whose unicode values are \u3068\u30889. In this case, the location value is %A4%C8%A4%E9. Note that all non-ASCII characters should be URL-encoded according to the charset parameter, which in this case is euc-jp. The following command is an example of same data sent in Shift_JIS:

```
storeevents.wcap?id=${SESSIONID}&calid=id&summary=summary&
location=location&desc=desc&charset=Shift_JIS
```

In the above example, the *location* value is %82%C6%82%E7.

WCAP uses the value of the `charset` parameter to convert the data from the URL-encoded value into UTF-8 before storing it into the database. It is stored internally in UTF-8.

The `charset` parameter in this command have the same role as in the `storeevents.wcap` because the `set_calprops` command takes non-ASCII data. The `charset` parameter in this command does not have any other special meaning.

If `charset` is not specified, WCAP expects the data to be URL-encoded in UTF-8.

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 error number.

Error String

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

Error Codes

Table 6-5 list some of the error codes returned in the error number array.

Table 6-5 Error Names, Values, and Meanings

Error Name	Value	Meaning
LOGOUT	-1	Logout successful.

Table 6-5 Error Names, Values, and Meanings *(Continued)*

Error Name	Value	Meaning
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.
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.

Table 6-5 Error Names, Values, and Meanings *(Continued)*

Error Name	Value	Meaning
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	If double booking is not allowed in this calendar, `storeevents` or `storetodos` will fail with this error when attempting to store an event/todo in a time slot that was already filled.
FETCH_BY_ALARM_RANGE_FAILED	41	Command failed.
FETCH_BY_ATTENDEE_ERROR_FAILED	42	Command failed.
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.

Table 6-5 Error Names, Values, and Meanings *(Continued)*

Error Name	Value	Meaning
INSUFFICIENT_PARAMETERS	46	The caller tried to invoke verifyevents_by_ids, or verifytodos_by_ids with insufficient arguments (mismatched number of uids and rids).
MUSTBEOWNER_OPERATION	47	The user needs to be an owner or co-owner of the calendar in questions to complete this operation. (Probably related to private or confidential component.)
GETTIME_FAILED	55	Get server time failed.
FETCH_DELETEDCOMPONENTS_FAILED	56	Fetch deleted components failed.
FETCH_DELETEDCOMPONENTS_PARTIAL_RESULT	57	Success but partial result.
WCAP_NO_SUCH_FORMAT	58	Returned in any of the commands when supplied fmt-out is not a supported format.
COMPONENT_NOT_FOUND	59	Returned when a fetch or delete is attempted that does not exist.
BAD_ARGUMENTS	60	Currently used when attendee or organizer specified does not have a valid email address.
GET_USERPREFS_FAILED	61	`get_userprefs.wcap` failed. The following error conditions will return error code 61: • ldap access denied • no results found • ldap limit exceeded • ldap connection failed
WCAP_MODIFY_NO_EVENT	62	`storeevents.wcap` issued with `storeytype` set to 2 (WCAP_STORE_TYPE_MODIFY) and the event doesn't exist.
WCAP_CREATE_EXISTS	63	`storeevents.wcap` issued with `storetype` set to 1 (WCAP_STORE_TYPE_CREATE) and the event already exists.
WCAP_MODIFY_CANT_MAKE_COPY	64	`storevents.wcap` issued and copy of event failed during processing.
STORE_FAILED_RECUR_SKIP	65	One instance of a recurring event skips over another
STORE_FAILED_RECUR_SAMEDAY	66	Two instances of a recurring event can't occur on the same day

Table 6-5 Error Names, Values, and Meanings *(Continued)*

Error Name	Value	Meaning
BAD_ORG_ARGUMENTS	67	Bad organizer arguments. `orgCalid` or `orgEmail` must be passed if any other org parameter is sent. That is, `orgUID` can't be sent alone on a `storeevents` or a `storetodos` command if it is trying about to "create" the event or task. Note, if no org information is passed, the organizer defaults to the `calid` being passed with the command.
LDAP_ERROR	69	For `get_calprops`, when there is an error is getting LDAP derived token values (`X-S1CS-CALPROPS-FB-INCLUDE`, `X-S1CS-CALPROPS-COMMON-NAME`).
GET_INVITE_COUNT_FAILED	70	Error in getting invite count (for `get_calprops.wcap` and `fetchcomponents_by_range.wcap` commands)
LIST_FAILED	71	`list.wcap` failed
LIST_SUBSCRIBED_FAILED	72	`list_subscribed.wcap` failed
SUBSCRIBE_FAILED	73	`subscribe.wcap` failed
UNSUBSCRIBE_FAILED	74	`unsubscribe.wcap` failed
ANONYMOUS_NOT_ALLOWED	75	Command cannot be executed as anonymous. Used only for `list.wcap`, `list_subscribed.wcap`, `subscribe.wcap`, and `unsubscribe.wcap` commands.
ACCESS_DENIED	76	Generated f a non-admin user tries to read or set the calendar-owned list or the calendar-subscribed list of some other user, or if the option is not turned on in the server
CDWP_ERR_MAX_CONNECTION_REACHED	11000	Maximum connections to back end database reached. As connections are freed up users will be allowed to connect to back end.
CDWP_ERR_CANNOT_CONNECT	11001	Cannot connect to back end. Back end machine may be down or DWP server is not up and running.
CDWP_ERR_CANNOT_RESOLVE_CALENDAR	11002	Front end cannot resolve calendar to a particular back end.
CDWP_ERR_BAD_DATA	11003	Bad data received from DWP connection.
CDWP_ERR_DWPHOST_CTX_DOES_NOT_EXIST	11004	For the back end host, context doesn't exist in the context table. Solution is to add back end host to ics.conf and restart front end.
CDWP_ERR_HOSTNAME_NOT_RESOLVABLE	11005	DNS/NIS/files or hostname resolver is not set up properly or machine does not exist.

Fetching Component Data

The component_type parameter directs WCAP to return either only events, only todos, or both events and todos. The keyword arguments, respectively, are: event, todo, or all. The parameter is not required. Its default is all, returning both events and todos. If an unrecognized value is passed in, the default value is used.

This parameter is found in all the fetchcomponents_by_* commands.

In addition, deleted components can be retrieved from the deletelog.db in the calendar store using the fetch_deletedcomponents command.

Fetching Component State Data

All fetch commands, except fetchcomponents_by_attendee_error, have the ability to fetch by component state, using the parameter compstate. The default (compstate=ALL) is to fetch all component states, Use this parameter to limit the type of components fetched.

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

Table 6-6 lists component state values. A component state pertains either to the attendee or the organizer.

Table 6-6 Component State Values for compstate Parameter

Value	Organizer/ Attendee	Comment
REPLY-DECLINED	Attendee	The event or todo is an invitation from another user and the current user has declined the invitation.
REPLY-ACCEPTED	Attendee	The event or todo is an invitation from another user and the current user has accepted the invitation.
REQUEST-COMPLETED	Organizer	The event or todo is an invitation from the current user to other invitees, and all invities have replied.
REQUEST_NEEDS-ACTION	Attendee	The event or todo is an invitation from another user and the current user has not replied to it yet.
REQUEST-NEEDSNOACTION	Attendee	The event or todo is an invitation from another user and the current user is not required to reply.
REQUEST-PENDING	Organizer	The event or todo is an invitation from the current user to other invitees, and is currently in the process of sending out invitations.
REQUEST-WAITFORREPLY	Organizer	The event or todo is an invitation from the current user to other invitees, and is currently awaiting replies from all invitees.

Table 6-6 Component State Values for `compstate` Parameter *(Continued)*

Value	Organizer/ Attendee	Comment
ALL	N/A	(Default) All event and todo component states.

Fetching Deleted Data

If you have deleted component data and need to reconstruct it, you can use the `fetch_deletedcomponents` command, which causes the system to process the Delete Log Database (ics50deletelog.db in the csdb directory). However, it is not possible to recreate the entire component data since not all of it is logged.

When non-recurring components are deleted, the server removes it from the component database and writes it to the Delete Log database.

When individual instances of a recurring event or task are deleted, the server writes each deleted instance to the Delete Log database. When all instances of the recurring event or todo are deleted, the server deletes the master entry for the component from the component database and writes it to the Delete Log database. A master entry in the Delete Log database contains the `rrules`, `exrules`, and `exdates` recurrence parameters.

In a single `fetch_deletedcomponents` command, either individual instances can be retrieved, or the master entry with its exceptions, but not both.

For more information on the Delete Log database, see the *Sun ONE Calendar Server Administrator's Guide.*

Fetching Recurrence Data

The `compressed` parameter allows you to retrieve a reduced amount of recurrence data. The parameter defaults (`compressed=0`) to the compressed format, which returns data without the `rrule`, `rdate`, `exrule`, and `exdate` properties as the default. To receive all the recurrence data back from the following commands, use `compressed=1`.

This parameter is used by all the `fetchcomponents_by_*` commands, the `fetchevents_by_id` and `fetchtodos_by_id` commands, and the `store*` commands.

NOTE This parameter has been deprecated for the current release and may be removed from the future releases.

Formatting Standards

Find the exact format and definition for all times, strings, parameters, etc. by referring to RFC 2445, RFC 2446, and RFC 2447. 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 As a reminder, the maximum parameter value length is 1024 characters.

For more information on time zones, see "Time Zones,"which follows later in this section.

Freebusy Calendars

Calendars can be displayed in freebusy format instead of showing details of scheduled events and todos. Freebusy calendars are used to facilitate scheduling of events or todos in the event and todo creation dialogs in the user interface. They can also be used by calendar owners to prevent other users from viewing the details of their calendars.

This can be accomplished in two separate ways that are not mutually exclusive:

- Freebusy access rights can be granted to users in the calendar's properties.
- Freebusy access can be assigned to the calendar as a whole using the calendar property `fbinclude`.

  If `fbinclude` is set to `0`, it overrides any access rights granted to users. That is, if `fbinclude=0`, then the calendar will not be included in freebusy calculations, no matter what the `acl` parameter specifies.

  If the `fbinclude` is set to `1`, which allows the use of the calendar for freebusy calculations, then the `acl` access rights will be used to determine if the user can see the details of the calendar, the freebusy representation only, or neither.

In freebusy calendars, instead of event or todo details, 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

If user `john` has only freebusy access to the calendar `jdoe`, user `john` will get only a freebusy version of the calendar. The freebusy time for `jdoe` (from 9:00 to 6:00) would appear as 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 the details of why the user is busy, but simply knows when the user is busy.

The `get_freebusy` command allows selection of which calendars to use in the calculation in two ways: using the `calid` parameter, or the `mail` parameter. One of the parameters is required, but not both. If both are present, the `calid` value is used.

When an email address is passed in using the `mail` parameter, all calendars, for which this user is the primary owner and which have `fbinclude=1` in their calendar properties settings, will be included in the freebusy calculation.

When the `calid` parameter is used, specific calendars can be named rather than using all of owners calendars. Note that the `calid` parameter can take an email address (by specifying `mailto:`*`ref822address`*, where *`rfc822address`* is any valid email address that maps to a single calendar in the local LDAP directory).

The command returns the busy data only. The rest of the time slots are presumed to be free.

Freebusy Calculation for Private Events

When a freebusy calendar rendition is requested using the `get_freebusy` command, private events and todos will be included or excluded depending on the value of the `transparent` parameter stored with the events and todos when they were created or modified.

Using the `store*` commands, set the event or todo `transparent` parameter to `1` to keep the event truly private, and set it to `0` to allow it to be included in the freebusy calculation.

As opposed to regular events, all-day events (`isAllDay` parameter is set to `1`) default to private and opaque (`transparent=1`). For the all-day event to be included in the freebusy calculation, the `transparent` parameter must be set to `0`.

Group Scheduling

When you are using the `storeevents` command for scheduling a group event, there are two parameters that are required:

- Attendee Parameter
- Method Parameter

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. (However, 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
```

An attendee in an existing meeting can be marked for deletion by assigning `X-NSCP-WCAP-ATTENDEE-DELETE` to `PARTSTAT`. For example, if you want to delete attendee `jdoe`, the `attendee` parameter of the `storeevents` command would contain the following:

```
PARTSTAT=X-NSCP-WCAP-ATTENDEE-DELETE^jdoe
```

Table 6-7 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 6-7 iCalendar ATTENDEE parameters understood by WCAP

Parameters	Purpose
PARTSTAT	The only required parameter. This shows the attendees participation status.

Table 6-7 iCalendar ATTENDEE parameters understood by WCAP

Parameters	Purpose
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.

In addition, WCAP allows the optional use of an additional parameter, SENT-STATUS, which is specific to Calendar Server and is not part of the iCalendar specification. Possible values for SENT-STATUS are: NOT-SENT, and SENT-SUCCEEDED. The default is NOT-SENT. The Group Scheduling Engine inside Calendar Server will not process an attendee with a SENT-STATUS value of SENT-SUCCEEDED.

Method Parameter

The method parameter describes the type of message used: invitation, response, cancellation.

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.

NOTE Even though the method parameter has a default value, it is a required parameter if you are trying to do anything other than PUBLISH. Leaving the parameter off the storeevents or storetodos commands will cause the default (PUBLISH) to be the presumed action.

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. The method parameter is set to "1". Note that nothing is sent to the attendees.
  - 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 Calendar Server Group Scheduling Engine.

The following is an example of an invitation:

```
storeevents.wcap?id=${SESSIONID of org}&calid=org&dtstart=
   20020201T200200Z&dtend=20020201T210000Z&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 following is an example of 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 following is an example of a 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
```

Output Format

WCAP commands can request the output format in two content types:

1. `text/calendar` - iCalendar
2. `text/xml` - iCalendar XML

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

Recurring Components – Overview

Recurrence handling occurs as follows:

- A recurring series of events or todos has a master entry plus entries for exceptions.
- Changing the `rrules` of a single event or todo is not supported; that is, when `rrules` are modified for one date of a recurring chain, the whole chain is deleted and recreated.
- Changing `dtstart` of a recurring series entry is not supported.

- Inserting a rid that was not part of the original rule is not supported.
- Multiple rrules for any component are not supported.

Recurring Components – Creating, Modifying

The following parameters are used with the storeevents and storetodos commands to create and modify components:

- rrules - Semicolon-separated list of quoted recurrence-rule strings for recurring events.
- rdates – Semicolon-separated list of ISO 8601 date strings listing recurrence dates.
- exrules – Semicolon-separated list of quoted recurrence-rule strings for dates to exclude.
- exdates – Semicolon-separated list of ISO 8601 date strings listing dates to exclude.
- rid – ISO 8601 DateTimeString giving the recurrence ID of an event.
- mod – Modifier telling which instances of the event to store.
- rchange – A boolean specifying whether or not to replace the rrules parameter. If rchange=1, the store commands map all mod settings (2-4) are mapped to 4. This means that you can not change an rrule for only some of the components in a recurring series. To change the rrules parameter, all components in the recurring series must be changed.

NOTE The rrules, rdates, exrules, and exdates parameters always function in replace mode. That is, no matter what the replace parameter value is set to, the values passed in always replace the parameter value, rather being appended to it.

This means you can not have multiple rrules for the same component.

For more information on the replace parameter, see Updating Parameter Values.

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 RFC 2445 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.

NOTE Using the storeevents.wcap command to create an event with only exdate or rdate values, without specifying a rrule results in no events being created. The same behavior can be observed with the storetodos.wcap command.

The following example shows an rrules parameter that specifies the event is to occur daily for 10 instances:

```
rrules="count%3D10%3Bfreq%3Ddaily" (COUNT=10;FREQ=DAILY)
```

The following example URL passes the example rrules parameter:

```
http://webcalendarserver/storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=20020301T112233Z
&rrules="count%3D10%3Bfreq%3Ddaily"&dtend=20020301T112233&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 (3/31/02 11:22:33 and 5/31/02 11:22:33):

```
rdates=20020331T112233;20020531T112233
```

The following example URL passes the example rdates parameter:

```
http://webcalendarserver/storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=20020301T112233Z
&rdates=20020331T112233;20020531T112233
&dtend=20020301T112233&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=20020301T112233Z
&exrules="count%3D10%3Bfreq%3Ddaily";"freq%3Dweekly%3Bcount%3D4"
&rrules="count%3D100%3Bfreq%3Ddaily"&dtend=20020301T112233&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 (3/31/02 11:22:33 and 5/31/02 11:22:33):

```
exdates=20020331T112233;20020531T112233
```

The following example URL passes the example exdates parameter:

```
http://webcalendarserver/storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=20020301T112233Z
&exdates=20020331T112233;20020531T112233
&rrules="COUNT%3D200%3BFREQ=DAILY";dtend=20020301T112233&summary=uuuu
```

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=20020301T112233Z
&rid=20020331T112233;dtend=20020301T112233&summary=uuuu&mod=1
```

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

mod

When modifying recurring components, this parameter specifies whether to apply the changes to one or more instances of the event or todo. The following settings are mapped, but currently only the value 4 is used.

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

When creating or modifying a recurring component, if changing rrules is allowed (rchange is set to 1), the system assumes a setting of 4, which causes the entire series of events or todos to be deleted and rewritten. If 2 or 3 are specified when trying to change rules or start times, it will be mapped to 4 anyway. If 2 0r 3 is specified for changing a summary or description, the setting will be honored, but no exceptions will be created for errors. A setting of mod=1 is not valid.

rchange

The rchange parameter specifies whether recurrences are expanded in storeevents, and storetodos. Normally, events and todos calendar components are not expanded, so the parameter defaults to 0.

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, 2002. Use the following URL to change the summary of each event after Feb. 1, 2002 to changed-event.

The following 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=20020201T112233Z
&rrules="byday%3Dfr%3Bfreq%3Dweekly"&summary=changed-event
&rid=20020201T112233Z&mod=2&rchange=0
```

Recurring Components – Deleting

When you delete a recurring component, specify the recurrence ID and whether to delete the recurrences as well as the original event or todo.

Use the mod parameter to choose which option you want:

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

A setting of 2 will delete only as many instances as exist on the server.

Examples Using deleteevents_by_id

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, 2002 11:22:33 AM GMT.

```
http://webcalendarserver/deleteevents_by_id.wcap?id=23423423434abc&calid=j
doe&uid=001&rid=20020301T112233Z&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, 2002 11:22:33 AM GMT and all future instances of this event (uid 001).

```
http://webcalendarserver/deleteevents_by_id.wcap?id=23423423434abc&
calid=jdoe&uid=001&rid=20020301T112233Z&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, 200211:22:33 AM GMT and all prior instances of this event (uid 001).

```
http://webcalendarserver/deleteevents_by_id.wcap?id=23423423434abc&
calid=jdoe&uid=001&rid=20020301T112233Z&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://webcalendarserver/deleteevents_by_id.wcap?id=23423423434abc&
calid=jdoe&uid=001&rid=20020301T112233Z&mod=4
```

Recurring Components – Fetching

The following parameters are found in the fetchcomponents_by_* commands, and the fetchevents_by_id and fetchtodos_by_id commands:

- compressed – A boolean specifying whether to return all of the recurring entry's data, or to exclude the following parameters:rrules, rdate, exrule, exdate.

  Note that this parameter is deprecated in the current release and may be removed from future releases.

- recurring – A boolean parameter specifying whether or not to return all components in compressed form (master entry and exceptions). This parameter is also present in the fetch_deletedcomponents command.

Time Zones

To support a universal standard, Calendar Server uses the date and time strings in Greenwich Mean Time (GMT) or Coordinated Universal Time (UTC), called Zulu time. The server stores and returns all date and time strings from the database in Zulu time. WCAP converts the Zulu times to the appropriate time zone settings depending upon the value of the tzid and tzidout parameters.

The tzid parameter is used for date and time strings passed in with the dtstart, dtend, and rid parameters, which are not already in Zulu time. WCAP uses the value of the tzid parameter to calculate the Zulu time. If the tzid parameter is not passed in, the server's default time zone is used to calculate Zulu time.

For commands that return events and todos, the data will be returned in Zulu time, unless the tzidout parameter is passed in. In this case the Zulu time is translated into the time zone specified in the tzidout parameter.

For example, if the fetch_components_by range command specifies a date range of 20020506T100000 to 20020507T100000, with a tzid=America/Los_Angeles, WCAP will translate that to Zulu time for database lookup. If the tzidout parameter was also passed in (for our example, tzidout=America/New_York), then the resulting output would be translated to that time zone and returned. If the tzidout parameter is missing, the component data is returned in Zulu time.

The tzidout parameter can be used with the storeevents and storetodos command when the fetch parameter is set to 1 (fetch=1).

The time zones information is kept in a plain text file (timezones.ics) in VTIMEZONE format.

The server never uses the system time zone information to calculate the current date and time. It uses the time elapsed in seconds since the Epoch (00:00:00 UTC, January 1, 1970) to calculate current date and time. Then depending on the user's time zone settings, the date is displayed to reflect the correct time zone.

The following commands use both the tzid and tzidout parameters:

- fetchcomponents_by_alarmrange
- fetchcomponents_by_lastmod
- fetchcomponents_by_range

- fetchevents_by_id
- fetchtodos_by_id
- storeevents
- storetodos

In addition, the following commands use the tzid parameter (but not the tzidout):

- deleteevents_by_id
- deletetodos_by_id
- get_freebusy
- set_calprops

Updating Parameter Values

Two commands, storeevents and storetodos, allow you to update (replace, append, or delete) parameter values. When updating current values for a component, you can either replace the current values with the new ones being passed in, append the new values to the current values, or pass in empty parameter values to delete the parameter.

The ability to append parameter values applies only to parameters that can accommodate multiple values (that is, parameters that use semicolon-separated values, such as the attendees parameter). The default is to append (replace=0) the new values to the current values. If you want to replace the current values with the new values being passed in, include the replace parameter in the command, with the value set to 1 (replace=1). If you do not include the replace parameter in the command, the system assumes the default setting (replace=0) and appends the new values to the old values.

With one exception, the recurrence and alarm parameters can only be replaced, not appended. Specifically, the parameters are: rrules, rdates, exrules, exdates, alarmAudio, alarmDescription,alarmFlashing, alarmPopup, and alarmStart. The alarmEmails parameter is the only one that allows multiple values, and thus is the only exception. Therefore you can append an alarm email recipient to the list by specifying replace=0.

In all cases, a parameter can be deleted by passing in an empty parameter and setting replace to 1. For example, to delete the alarmPopup parameter, pass in the following: alarmPopup=&replace=1. Using replace=1 can also be used to delete string fields. For example, to delete the description, you would use desc=&replace=1.

X-Tokens

Table 6-8 lists the X-tokens returned by WCAP commands.

Table 6-8 X-Tokens Returned by WCAP Commands

Token Name	Type	WCAP Command	Description
X-NSCP-WCAP-PREF-	varies by preference	get_userprefs	User preferences: cn, givenName, preferredLanguage, sn, nswcalCALID, cefontSizeDelta, ceTableSpacing, ceColorSet, ceBgcolor, ceCalList, ceAgendaList, ceAgendaTZMode_Personal, ceCursorColor
X-NSCP-WCAP-SERVER-PREF-	varies by preference	get_userprefs	Server preferences: allowchangepassword, allowcreatecalendars allowdeletecalendars
X-NSCP-GUID	GUID integer	get_guids	Globally unique identifier.
X-NSCP-WCAP-SESSION-ID	string	check_id	Session identifier of logged-in user
X-NSCP-WCAP-USER-ID	string		user ID of logged-in user
X-NSCP-WCAP-CALENDAR-ID	string		calendar ID of logged-in user
X-NSCP-WCAP-ERRNO	integer	all	Error number generated from WCAP command
X-NSCP-REQUEST-STATUS	string	deleteevents_by_range, deletetodos_by_range	Error message string returned from WCAP commands.
X-NSCP-WCAPVERSION	string		WCAP version the server currently supports (current version is 3.0)
X-NSCP-CALPROPS-RELATIVE-CALID	string	get_freebusy, all fetch commands	Calendar identifier of user.
X-NSCP_CALPROPS-LAST-MODIFIED	string	all fetch commands	Date string of last modification to properties.
X-NSCP-CALPROPS-CREATED	string	all fetch commands	Date the component was created.
X-NSCP-CALPROPS-READ	integer	all fetch commands	Permission to read properties.
X-NSCP-CALPROPS-WRITE	integer	all fetch commands	Permission to write properties.
X-NSCP-CALPROPS-DESCRIPTION	string	all fetch commands	Calendar description.
X-NSCP-CALPROPS-PARENT-CALID	string	all fetch commands	Parent calid if present.

Table 6-8 X-Tokens Returned by WCAP Commands

Token Name	Type	WCAP Command	Description
X-NSCP-CALPROPS-CALMASTER	string	all fetch commands	Calmaster
X-NSCP-CALPROPS-NAME	string	all fetch commands	Name of calendar.
X-NSCP-CALPROPS-CHARSET	string	all fetch commands	Character Set used by calendar.
X-NSCP-CALPROPS-LANGUAGE	string	all fetch commands	Language used by calendar.
X-NSCP-CALPROPS-PRIMARY-OWNER	string	all fetch commands	Calendar's primary owner.
X-NSCP-CALPROPS-TZID	string	all fetch commands	Time date string for calendar creation.
X-NSCP-CALPROPS-OWNERS	string	all fetch commands	Owners for this calendar.
X-NSCP-CALPROPS-CHILDREN	string	all fetch commands	Child calendars if present.
X-NSCP-CALPROPS-CATEGORIES	string	all fetch commands	
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY	string	all fetch commands	ACL for this calendar.
X-NSCP-CALPROPS-RESOURCE	string	all fetch commands	Resource rather than a regular calendar.
X-NSCP-CHARSET	string	all fetch commands	Character Set.
X-NSCP-LANGUAGE	string	all fetch commands	Returns the lang string (for example, en for English)
X-NSCP-LINK-CALID	string	all fetch commands	Calendar ID.
X-NSCP-ONGOING	integer	all fetch commands	
X-NSCP-TOMBSTONE	integer	all fetch commands	Dead events (those in the past).
X-NSCP-ORGANIZER-EMAIL	string	all fetch commands	Organizer's email address.
X-NSCP-DTSTART-TZID	string	all fetch commands	Start time string.
X-NSCP-DTEND-TZID	string	all fetch commands	End time string.
X-NSCP-DUE-TZID	string	all fetch commands	Due date time string.
X-NSCP-GSE-COMPONENT-STATE	string	all fetch commands	Component state.
X-NSCP-GSE-COMMENT	string	all fetch commands	Comment.
X-NSCP-ORGANIZER-UID	string	all fetch commands	Organizer's uid.
X-NSCP-ORGANIZER-SENT-BY-UID	string	all fetch commands	uid for who created the component for the organizer.
X-NSCP-ATTENDEE-GSE-STATUS	integer	all fetch commands	Attendee status for group scheduled components.

Table 6-8 X-Tokens Returned by WCAP Commands

Token Name	Type	WCAP Command	Description
X-S1CS-CALID	string	all fetch commands	Calendar ID.
X-S1CS-EMAIL	string	all fetch commands	Email address.
X-S1CS-RECURRENCE-COUNT	integer	all fetch commands	Count of the number of components in the recurring series.
X-S1CS-RECURRENCE-UNTIL	string	all fetch commands	Date string for end date.
X-S1CS-RECURRENCE-RDATELIST	string	all fetch commands	List of recurrence dates.
X-S1CS-RECURRENCE-EXDATELIST	string	all fetch commands	Exception list for recurring master.
X-S1CS-ATTENDEE-RDATELIST	string	all fetch commands	Attendees for each date.
X-S1CS-ATTENDEE-EXDATELIST	string	all fetch commands	Attendee exceptions for each date.
X-NSCP-COMPONENT-SOURCE	string	all fetch commands	
X-NSCP-TRIGGERED_BY	string	all fetch commands	
X-S1CS-EXPORTVERSION	string	all fetch commands	Export version.
X-S1CS-TZID-ALIAS	string	all fetch commands	Alias for time zone.

WCAP Commands

For all commands that allow 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.

Unless otherwise specified, the maximum length parameter value for WCAP commands is 1024 characters. For example, you can’t have a uid longer than 1024 characters. No length checking is done by WCAP on incoming parameter values, but a longer value can cause unpredictable results.

The following is a list of the available 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
fetch_deletedcomponents
fetchcomponents_by_range
get_all_timezones
get_calprops
get_freebusy
get_guids
gettime
get_userprefs
import
login
logout
ping
search_calprops
set_calprops
set_userprefs
storeevents
storetodos
verifyevents_by_ids

fetchevents_by_id
fetchtodos_by_id
verifytodos_by_ids
version

addlink

Purpose

Add links from one calendar's events or todos to another calendar.

Parameters

Table 6-9 lists addlink parameters:

Table 6-9 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
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. It is here only for backward compatibility. See the "Administrator's Guide" for details on changing a password.

Parameters

Table 6-10 lists change_password parameters:

Table 6-10 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. The three format types: text/calendar text/xml	N	text/calendar

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 6-10 lists `check_id` parameters:

Table 6-11 `check_id` Parameters

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

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

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 2002 19:48:17 GMT
Content-type: text/calendar; charset=UTF-8
Content-length: 131
Last-modified: Thu, 14 Dec 2002 19:48:17 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache
Connection: Keep-Alive

BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-WCAP-CHECK-ID:1
END:VCALENDAR
```

createcalendar

Purpose

Create a new calendar.

Parameters

Table 6-12 lists createcalendar parameters:

Table 6-12 createcalendar Parameter

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. The three format types: text/calendar text/xml	Y	text/calendar
set_calprops	integer (0,1)	A boolean indicating whether to set the properties of the new calendar. 1 = Set properties. 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

You can 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.

Note that at calendar creation, if you do not specify calendar properties, the defaults set in the ics.conf file will be used.

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;work
```

deletecalendar

Purpose

This command deletes a user's calendar.

Parameters

Table 6-13 lists deletecalendar parameters:

Table 6-13 deletecalendars Parameter

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. The three format types: text/calendar text/xml	Y	text/calendar

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.

NOTE ENS notifications for `appid` are not yet implemented.

Parameters

Table 6-14 lists `deletecomponents_by_range` parameters:

Table 6-14 `deletecomponents_by_range` Parameters

Parameter	Type	Purpose	Required	Default
`appid`	string	A runtime parameter (not stored in the database) that specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the *SunONE Messaging and Collaboration Event Notification Service Manual*.	N	N/A
`calid`	string	Semicolon-separated list of calendar identifiers from which to delete events and todos.	N	Current user's `calid`.
`dtend`	ISO 8601 DateTime Z string (UTC)	End time and date of events or todos to be deleted. A value of `0` means delete all events and todos up to the end of time. This value must be in coordinated universal time.	N	`0`
`dtstart`	ISO 8601 DateTime Z string (UTC)	Start time and date of events or todos to be deleted. 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.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`id`	unique identifier string	The session identifier.	Y	N/A
`smtp`	integer (0, 1)	Send email invitation to user without calendar. 0—No 1—Yes	N	1

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 6-15 lists `deleteevents_by_id` parameters:

Table 6-15 `deleteevents_by_id` Parameters

Parameter	Type	Purpose	Required	Default
`appid`	string	A runtime parameter (not stored in the database) that specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the *SunONE Messaging and Collaboration Event Notification Service Manual*.	N	N/A
`calid`	string	Calendar identifier of event to delete.	Y	N/A

`fmt-out`	string	The format for the returned data.
		The two format types:
		`text/calendar`
		`text/xml`
			N	text/ calendar

Table 6-15 `deleteevents_by_id` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`id`	unique identifier string	The session identifier. Required unless the calendar is public.	Y	`NULL`

`mod`	integer
	`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.
		One of the following values:
		`1` = `THISINSTANCE`
		`2` = `THISANDFUTURE`
		`3` = `THISANDPRIOR`
		`4` = `THISANDALL`
			Y	N/A

`notify`	integer `0`,`1`	A boolean indicating whether or not to notify attendees of this change. `1` = Notify attendees. `0` = Do not notify attendees.	N	`0`
`rid`	string	Recurrence identifier of the event, or semicolon-separated list of recurrence identifiers. If a list, it must have same number of elements as the `uid` list. If there are no recurrences, the value is `0`.	Y	N/A
`smtp`	integer (0, 1)	Send email invitation to user without calendar. 0—No 1—Yes	N	1
`tzid`	time zone ID string	Default time zone to use if the `rid` parameter does not have a time zone specified. 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 the `uid` does not exist, the server returns error code `59`. See also, “Error Handling.”

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 "Recurring Components – Deleting.".) 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=0;0&fmt-out-text/calendar
```

The resulting data would look like this:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
BEGIN:VEVENT
UID:uid-EVENT1
REQUEST-STATUS:2.0;Success.  Delete successful.
END:VEVENT
BEGIN:VEVENT
UID:uid-EVENT2
REQUEST-STATUS:2.0;Success.  Delete successful.
END:VEVENT
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

deleteevents_by_range

Purpose

Delete events from a calendar in a specified range.

NOTE ENS notifications for `appid` are not yet implemented.

Parameters

Table 6-16 lists `deleteevents_by_range` parameters:

Table 6-16 `deleteevents_by_range` Parameters

Parameter	Type	Purpose	Required	Default
`appid`	string	A runtime parameter (not stored in the database) that specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the *SunONE Messaging and Collaboration Event Notification Service Manual*.	N	N/A
`calid`	string	Semicolon-separated list of calendar identifiers from which to delete events.	N	Current user's `calid`.
`dtend`	ISO 8601 DateTime string	End time and date of events to be deleted. A value of `0` means delete all events until the end of time.	N	`0`
`dtstart`	ISO 8601 DateTime string	Start time and date of events to be deleted. A value of `0` means delete all events from the beginning of time.	N	`0`

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`id`	unique identifier string	The session identifier.	Y	N/A
`smtp`	integer (0, 1)	Send email invitation to user without calendar. 0—No 1—Yes	N	1

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 `text/calendar` format.

Error Codes

If the operation is successful, the error number of `0` is appended to the error string, `errno`. If the operation is not successful, the `errno` variable contains the error: `DELETEEVENTS_BY_RANGE_FAILED(22)`.

See also, "Error Handling."

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 6-17 lists `deletetodos_by_id` parameters:

Table 6-17 `deletetodos_by_id` Parameters

Parameter	Type	Purpose	Required	Default
`appid`	string	A runtime parameter (not stored in the database) that specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the *SunONE Messaging and Collaboration Event Notification Service Manual*.	N	N/A

Table 6-17 `deletetodos_by_id` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`calid`	string	Calendar identifier of the todos to delete.	Y	N/A

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`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 of elements as `uid` list. One of the following values: `1` = THISINSTANCE `2` = THISANDFUTURE `3` = THISANDPRIOR `4` = THISANDALL	Y	N/A
`notify`	integer (0,1)	A boolean telling whether or not to notify attendees of this change. `1` = Notify attendees. `0` = Do not notify attendees.	N	`0`
`rid`	string	The recurrence identifier of the todo, or a semicolon-separated list of recurrence identifiers. If a list, it must have the same number of elements as the `uid` list. If there are no recurrences, the value is `0`.	Y	N/A
`tzid`	time zone ID string	Default time zone to use if `dtstart`, or `dtend` parameters do not have a time zone specified. 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 `text/calendar` format.

Error Codes

If the uid does not exist, returns error 59.

See also, "Error Handling."

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&calid=jd
oe&uid=001&notify=1
```

Recurrences

If the rid parameter is passed, the command also deletes recurrences, as specified by the mod parameter. See "Recurring Components – Deleting."

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 todos in the database with UIDs of uid-TODO1 and uid-TODO2. Since the todos are non-recurring, the rid value for each todo is set to 0 and mod value for each todo is set to 1.

The following URL deletes the two todos:

```
http://webcalendarserver/deletetodos_by_id.wcap?id=br6p3t6bh5po35r
  &uid=uid-TODO1;uid-TODO2&rid=0;0&mod=1;1&fmt-out=text/calendar
```

The resulting data would look like this:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
BEGIN:VTODO
UID:uid-TODO1
REQUEST-STATUS:2.0;Success.  Delete successful.
END:VTODO
BEGIN:VTODO
UID:uid-TODO2
```

```
REQUEST-STATUS:2.0;Success.  Delete successful.
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

deletetodos_by_range

Purpose

Delete todos in a range from a calendar.

NOTE ENS notifications for `appid` are not yet implemented.

Parameters

Table 6-18 lists `deletetodos_by_range` parameters:

Table 6-18 `deletetodos_by_range` Parameters

Parameter	Type	Purpose	Required	Default
`appid`	string	A runtime parameter (not stored in the database) that specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the *SunONE Messaging and Collaboration Event Notification Service Manual*.	N	N/A
`calid`	string	Semicolon-separated list of calendar identifiers from which to delete todos.	N	Current user's `calid`
`dtstart`	ISO 8601 DateTime string	Start time and date of todos to be deleted. A value of `0` means delete all todos from the beginning of time.	N	`0`
`dtend`	ISO 8601 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.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

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 todo that occurs on the date March 1, 2002 11:22:33 AM GMT.

```
http://webcalendarserver/deletetodos_by_id.wcap?id=23423423434abc&calid=jdoe&uid=001&rid=20020301T112233Z&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 `text/calendar`.

Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If an error occurs, the `errno` variable contains the error: `DELETETODOS_BY_RANGE_FAILED(23)`.

See also, "Error Handling."

export

Purpose

Export events and todos from a calendar to a file.

Parameters

Table 6-19 lists `export` parameters:

Table 6-19 `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:
		`text/calendar`
		`text/xml`
			Y	N/A

`dtend`	ISO 8601 DateTime 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`

Table 6-19 `export` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`dtstart`	ISO 8601 DateTime 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.	N	`NULL`

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=t9u9m0eh8x5pu9b
  &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=t9u9m0eh8x5pu9b
  &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 2002 22:15:52 GMT
Content-type: text/calendar
Content-disposition: attachment; filename="export.ics"
Content-length: 7004

BEGIN:VCALENDAR
METHOD:PUBLISH
VERSION:6.0
BEGIN:VEVENT
UID:tm-001
RECURRENCE-ID:20020519T010000Z
DTSTAMP:20020603T221548Z
SUMMARY:Calendar Staff
DTSTART:20020518T170000Z
DTEND:20020518T190000Z
CREATED:20020603T024254Z
LAST-MODIFIED:20020603T024254Z
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:20020526T010000Z
DTSTAMP:20020603T221548Z
SUMMARY:Calendar Staff
DTSTART:20020525T170000Z
DTEND:20020525T190000Z
CREATED:20020603T024254Z
LAST-MODIFIED:20020603T024254Z
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 6-20 lists `fetchcomponents_by_alarmrange` parameters:

Table 6-20 `fetchcomponents_by_alarmrange` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	A semicolon-separated list of calendar identifiers.	N	Current user's `calid`.

Table 6-20 `fetchcomponents_by_alarmrange` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`component-type`	keyword (`event`, `todo`, `all`)	Indicates which components to return: `event` returns only events `todo` returns only todos (tasks) `all` returns both events and todos If an invalid value is passed in, the system assumes `all`.	N	`all`

`compressed`	integer (`0`,`1`)	This parameter is deprecated in this release and may be deleted in future releases.
		
		For `compressed=0`, returns less data. Specifically, it does not return the following parameters:
		`rrules`
		`rdate`
		`exrule`
		`exdate`
		
		For `compressed=1`, all recurrence data is returned.
			N	`0`

`compstate`	semicolon-separated list of component state keywords	The list of component states to fetch. For `compstate` values, see Table 6-6 on page 84.	N	ALL
`dtend`	ISO 8601 DateTime string	End time and date of events and todos to be returned. A value of `0` means fetch all events.	N	`0`
`dtstart`	ISO 8601 DateTime string	Start time and date of events/todos with alarms ready to go off during the specified time. A value of `0` means fetch all events from the beginning of time.	N	`0`
`emailorcalid`	integer (`0`, `1`)	`0` = The `calid` is returned in the cal-address part of the ATTENDEE and ORGANIZER properties. `1` = The email address is returned in the cal-address part of the ATTENDEE and ORGANIZER properties. The X-Token X-S1CS-CALID contains the `calid` value.	N	`0`

`fmt-out`	string	The format for the returned data.
		
		The three format types:
		
		`text/calendar`
		`text/xml`
			N	`text/calendar`

Table 6-20 `fetchcomponents_by_alarmrange` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`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`
`recurring`	integer (`0`, `1`)	Whether to return all components in compressed form (1) or not (0). (Compressed form has master entry plus exceptions.)	N	0 (not compressed)
`relativealarm`	integer (0, 4)	Return the alarm as relative or absolute. 0—Return alarm values as absolute. 4—Return alarms as originally created.	N	0 (absolute)
`tzid`	time zone ID string	If `dtstart` and `dtend` parameters are not already in Zulu time, the time zone to use for translating them to Zulu time. For example, “America/Los_Angeles”	N	server’s default time zone
`tzidout`	time zone ID string	Time zone to report returned data in.	N	Zulu time

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 `text/calendar` format.

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 data 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 data 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`.

If the times specified in the `dtstart` and `dtend` parameters is not Zulu time, the system uses the time zone specified in the `tzid` parameter to translate the times into Zulu time for data retrieval. If the `tzid` parameter is missing, the system uses the server's default time zone.

The system uses the `tzidout` parameter to determine what time zone to translate retrieved data into before returning it. If the `tzidout` parameter is missing, the system returns the data in Zulu time.

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 Dec. 25, 2001, 12:30 PM GMT
- eventB: alarm on Feb. 10, 2002, 10:00 AM GMT
- todoA: alarm on Jan. 20, 2002, 1:15 PM GMT

Here are two queries and their return values:

Example 1

This query fetches all events and todos that have alarms about to go off between Dec. 1, 2001 and Jan. 31, 2002.

```
http://webcalendarserver/fetchcomponents_by_alarmrange.wcap?id=abcdefg&dts
tart=20011201T112233Z&dtend=20020131T112233Z&fmt-out=text/calendar
```

It returns eventA and todoA:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
```

```
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3c11625900005ffe00000011000010b7
DTSTAMP:20011208T011139Z
SUMMARY:eventA
DTSTART:20011225T133000Z
DTEND:20011225T143000Z
CREATED:20011208T004409Z
LAST-MODIFIED:20011208T010857Z
PRIORITY:0
SEQUENCE:4
ORGANIZER;SENT-BY="jdoe@sesta.com"
;X-NSCP-ORGANIZER-UID=jdoe
;X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:CONFIRMED
TRANSP:OPAQUE
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL
;PARTSTAT=ACCEPTED;CN="JOHN SMITH"
;RSVP=TRUE
;X-NSCP-ATTENDEE-GSE-STATUS=2
:jdoe
X-NSCP-ORIGINAL-DTSTART:20020210T190000Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20011225T123000Z
ATTENDEE:MAILTO:jsmith@company22.com
```

```
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;X-NSCP-GSE-COMMENT="REQUEST-COMPLETED":
31074
END:VEVENT
BEGIN:VTODO
UID:3c1162e200207ff600000015000010b7
DTSTAMP:20011208T011139Z
SUMMARY:todoA
DTSTART:20011208T004626Z
DUE:20020120T141500Z
CREATED:20011208T004626Z
LAST-MODIFIED:20011208T011000Z
PRIORITY:0
SEQUENCE:3
PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="jdoe@sesta.com"
;X-NSCP-ORGANIZER-UID=jdoe
;X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20011208T004626Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020120T131500Z
ATTENDEE:MAILTO:jdoe@sesta.com
END:VALARM
X-NSCP-DUE-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":65538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Example 2

This query fetches all events and todos that have alarms to go off between Jan. 1, 2002 and June 1, 2002.

```
http://webcalendarserver/fetchcomponents_by_alarmrange.wcap?id=abcdefg&dts
tart=20020101T000000Z&dtend=20020601T000000Z&fmt-out=
text/calendar
```

It returns eventB and todoA:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3c1162b3000051c300000013000010b7
DTSTAMP:20011208T011645Z
SUMMARY:eventB
DTSTART:20020210T110000Z
DTEND:20020210T120020Z
CREATED:20011208T004539Z
LAST-MODIFIED:20011208T011638Z
PRIORITY:0
SEQUENCE:4
ORGANIZER;SENT-BY="jdoe@sesta.com"
;X-NSCP-ORGANIZER-UID=jdoe
;X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
```

```
STATUS:CONFIRMED
TRANSP:OPAQUE
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL
;PARTSTAT=ACCEPTED;CN="John Smith"

;RSVP=TRUE
;X-NSCP-ATTENDEE-GSE-STATUS=2:jsmith
X-NSCP-ORIGINAL-DTSTART:20021225T213000Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020210T100000Z
ATTENDEE:MAILTO:jsmith@company22.com
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles

X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="REQUEST-COMPLETED":131074
END:VEVENT
BEGIN:VTODO
UID:3c1162e200207ff600000015000010b7
DTSTAMP:20011208T011645Z
SUMMARY:todoA
DTSTART:20011208T004626Z
DUE:20020120T141500Z
CREATED:20011208T004626Z
LAST-MODIFIED:20011208T011000Z
PRIORITY:0
SEQUENCE:3

PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="jdoe@sesta.com"
;X-NSCP-ORGANIZER-UID=jdoe
;X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20011208T004626Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020120T131500Z
ATTENDEE:MAILTO:jdoe@sesta.com
```

```
END:VALARM
X-NSCP-DUE-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0

X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":
5538

END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

fetchcomponents_by_attendee_error

Purpose.
Fetch a list of components that had errors while sending group scheduling messages.

Parameters.
Table 6-21 lists `fetchcomponents_by_attendee_error` parameters

Table 6-21 `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. If this parameter is not specified, the command searches the primaryOwner's calendars for all event errors for any attendee.	N	N/A
`calid`	string	A semicolon-separated list of calendar identifiers.	N	Current user's `calid`.
`component-type`	keyword (`event`, `todo`, `all`)	Indicates which components to return: `event` returns only events `todo` returns only todos (tasks) `all` returns both events and todos If an invalid value is passed in, the system assumes `all`.	N	`all`

Table 6-21 `fetchcomponents_by_attendee_error` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default

`compressed`	integer (`0`,`1`)	This parameter is deprecated in this release and may be deleted in future releases.
		For `compressed=0`, returns less data. Specifically, it does not return the following parameters:
		`rrules`
		`rdate`
		`exrule`
		`exdate`
		For `compressed=1`, all recurrence data is returned.
			N	`0`

`emailorcalid`	integer (`0`, `1`)	`0` = The `calid` is returned in the cal-address part of the ATTENDEE and ORGANIZER properties. `1` = The email address is returned in the cal-address part of the ATTENDEE and ORGANIZER properties. The X-Token X-S1CS-CALID contains the `calid` value.	N	`0`

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`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`
`recurring`	integer (`0`, `1`)	Whether to return all components in compressed form (1) or not (0). (Compressed form has master entry plus exceptions.)	N	0 (not compressed)
`relativealarm`	integer (`0`, `4`)	Return the alarm as relative or absolute. 0—Return alarm values as absolute. 4—Return alarms as originally created.	N	0 (absolute)
`tzidout`	standard time zone string	The time zone returned data is translated to.	N	Returns data in Zulu time

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 `text/calendar` format.

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 data 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 data 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.

The system uses the tzidout parameter to determine what time zone to translate retrieved data into before returning it. If the tzidout parameter is missing, the system returns the data in Zulu time.

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 6-22 lists fetchcomponents_by_lastmod parameters

Table 6-22 fetchcomponents_by_lastmod Parameters

Parameter	Type	Purpose	Required	Default
calid	string	A semicolon-separated list of calendar identifiers.	N	Current user's calid.
component-type	keyword (event, todo, all)	Indicates which components to return: event returns only events todo returns only todos (tasks) all returns both events and todos If an invalid value is passed in, the system assumes all.	N	all

Table 6-22 `fetchcomponents_by_lastmod` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default

`compressed`	integer (0,1)	This parameter is deprecated in this release and may be deleted in future releases.
		For `compressed=0`, returns less data. Specifically, it does not return the following parameters:
		`rrules`
		`rdate`
		`exrule`
		`exdate`
		For `compressed=1`, all recurrence data is returned.
			N	`0`

`compstate`	semicolon-separated list of component state keywords	The list of component states to fetch. For `compstate` values, see Table 6-6 on page 84.	N	ALL
`dtend`	ISO 8601 DateTime Z string	End time and date of events to be returned. A value of `0` means fetch all events.	N	`0`
`dtstart`	ISO 8601 DateTime Z string	Start time and date of events to be returned. A value of `0` means fetch all events from the beginning of time.	N	`0`
`emailorcalid`	integer (0, 1)	`0` = The `calid` is returned in the cal-address part of the ATTENDEE and ORGANIZER properties. `1` = The email address is returned in the cal-address part of the ATTENDEE and ORGANIZER properties. The X-Token X-S1CS-CALID contains the `calid` value.	N	`0`

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`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`
`recurring`	integer (0, 1)	Whether to return all components in compressed form (1) or not (0). (Compressed form has master entry plus exceptions.)	N	0 (not compressed)

Table 6-22 `fetchcomponents_by_lastmod` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`relativealarm`	integer (`0`, `4`)	Return the alarm as relative or absolute. `0`—Return alarm values as absolute. `4`—Return alarms as originally created.	N	0 (absolute)
`tzid`	time zone ID string	Time zone to use if `dtstart`, or `dtend` parameters are not in Zulu time. For example, “America/Los_Angeles”	N	server’s default time zone
`tzidout`	time zone ID string	Time zone to report returned data in.	N	Zulu time

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 `text/calendar` format.

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 data 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 data 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`.

If the times specified in the `dtstart` and `dtend` parameters is not Zulu time, the system uses the time zone specified in the `tzid` parameter to translate the times into Zulu time for data retrieval. If the `tzid` parameter is missing, the system uses the server’s default time zone.

The system uses the `tzidout` parameter to determine what time zone to translate retrieved data into before returning it. If the `tzidout` parameter is missing, the system returns the data in Zulu time.

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, 2002, 10:00 AM GMT.
- eventB: last-modified on Dec. 25, 2001, 12:30 PM GMT.
- todoA: last-modified on Jan. 20, 2002, 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=20011201T112233Z&dtend=20020131T112233Z
```

The above query would fetch all modified events and todos between 12/1/2001 and 1/31/2002. Thus eventB and todoA would be returned.

```
http://webcalendarserver/fetchcomponents_by_lastmod.wcap?id=jdoe
&dtstart=20020101T112233Z&dtend=20020601T112233Z
```

The above query would fetch all events and todos that have been modified between 1/1/2002 and 6/1/2002. Thus eventA and todoA would be returned.

fetchcomponents_by_range

Purpose

Retrieve calendar events and todos.

Parameters

Table 6-23 lists fetchcomponents_by_range parameters:

Table 6-23 fetchcomponents_by_range Parameters

Parameter	Type	Purpose	Required	Default
calid	string	A semicolon-separated list of calendar identifiers.	N	Current user's calid.
component-type	keyword (event, todo, all)	Indicates which components to return: event returns only events todo returns only todos (tasks) all returns both events and todos If an invalid value is passed in, the system assumes all.	N	all
compressed	integer (0,1)	This parameter is deprecated in this release and may be deleted in future releases. For compressed=0, returns less data. Specifically, it does not return the following parameters: rrules rdate exrule exdate For compressed=1, all recurrence data is returned.	N	0
compstate	semicolon-separated list of component state keywords	The list of component states to fetch. For compstate values, see Table 6-6 on page 84.	N	ALL
dtend	ISO 8601 DateTime Z string	End time and date of events to be returned. A value of 0 means fetch all events.	N	0
dtstart	ISO 8601 DateTime Z string	Start time and date of events to be returned. A value of 0 means fetch all events from the beginning of time.	N	0
emailorcalid	integer (0, 1)	0 = The calid is returned in the cal-address part of the ATTENDEE and ORGANIZER properties. 1 = The email address is returned in the cal-address part of the ATTENDEE and ORGANIZER properties. The X-Token X-S1CS-CALID contains the calid value.	N	0

Table 6-23 `fetchcomponents_by_range` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`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`
`recurring`	integer (`0`, `1`)	Whether to return all components in compressed form (1) or not (0). (Compressed form has master entry plus exceptions.)	N	0 (not compressed)
`relativealarm`	integer (0, 4)	Return the alarm as relative or absolute. 0—Return alarm values as absolute. 4—Return alarms as originally created.	N	0 (absolute)
`tzid`	time zone ID string	Default time zone to use if `dtstart`, or `dtend` parameters are not in Zulu time. For example, “America/Los_Angeles”	N	server’s default time zone
`tzidout`	time zone ID string	Time zone to report returned data in.	N	Zulu time

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 `text/calendar` format.

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`.

Tasks returned by this command:

- Tasks due within the date range
- Overdue tasks

- Tasks completed within the date range
- Tasks that are never due but have a start date within the range

If the times specified in the `dtstart` and `dtend` parameters is not Zulu time, the system uses the time zone specified in the `tzid` parameter to translate the times into Zulu time for data retrieval. If the `tzid` parameter is missing, the system uses the server's default time zone.

The system uses the `tzidout` parameter to determine what time zone to translate retrieved data into before returning it. If the `tzidout` parameter is missing, the system returns the data in Zulu time.

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.

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 `text/calendar` format.

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 data 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 data 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.

Example

Example 1

This example fetches components for the current user from Dec. 1, 2001 to Jan. 31, 2002, using the following URL:

```
http://webcalendarserver/fetchcomponents_by_range.wcap?id=bes6bbe2mu98uw9&
dtstart=20011201T000000Z&dtend=20020131T000000Z&fmt-out=text/calendar
```

It returns one event and one todo for this period:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3c11625900005ffe00000011000010b7
DTSTAMP:20011208T015014Z
SUMMARY:eventA
DTSTART:20011225T133000Z
DTEND:20011225T143000Z
CREATED:20011208T004409Z
LAST-MODIFIED:20011208T010857Z
PRIORITY:0
SEQUENCE:4
ORGANIZER;SENT-BY="jdoe@sesta.com"
;X-NSCP-ORGANIZER-UID=jdoe
;X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
```

```
STATUS:CONFIRMED
TRANSP:OPAQUE
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED;
CN="John Smith";RSVP=TRUE;X-NSCP-ATTENDEE-GSE-STATUS=2:jsmith
X-NSCP-ORIGINAL-DTSTART:20020210T190000Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20011225T123000Z
ATTENDEE:MAILTO:jsmith@company22.com
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;X-NSCP-GSE-COMMENT="REQUEST-COMPLETED":
31074
END:VEVENT
BEGIN:VTODO
UID:3c1162e200207ff600000015000010b7
DTSTAMP:20011208T015014Z
SUMMARY:todoA
DTSTART:20011208T004626Z
DUE:20020120T141500Z
CREATED:20011208T004626Z
LAST-MODIFIED:20011208T011000Z
PRIORITY:0
SEQUENCE:3
PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="jdoe@sesta.com"
;X-NSCP-ORGANIZER-UID=jdoe
;X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20011208T004626Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020120T131500Z
ATTENDEE:MAILTO:jsmith@company22.com
END:VALARM
X-NSCP-DUE-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":
```

```
5538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Example 2

The second example fetches all components for calendars jdoe and susan between Dec. 1, 2001 to Jan. 31, 2002.

```
http://webcalendarserver/fetchcomponents_by_range.wcap?id=bes6bbe2mu98uw9&
calid=jdoe;susan&dtstart=20020101T000000Z&dtend=20020202T000000Z&fmt-out=t
ext/calendar
```

The following events and todos are returned:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3c1162b3000051c300000013000010b7
DTSTAMP:20011208T011645Z
SUMMARY:Joe's event
DTSTART:20020110T110000Z
DTEND:20020110T120020Z
```

```
CREATED:20011208T004539Z
LAST-MODIFIED:20011208T011638Z
PRIORITY:0
SEQUENCE:4
ORGANIZER;SENT-BY="jdoe@sesta.com";
X-NSCP-ORGANIZER-UID=jdoe;
X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:CONFIRMED
TRANSP:OPAQUE
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED;
CN="John Smith";RSVP=TRUE;X-NSCP-ATTENDEE-GSE-STATUS=2:jsmith
X-NSCP-ORIGINAL-DTSTART:20021225T213000Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020210T100000Z
ATTENDEE:MAILTO:jsmith@company22.com
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="REQUEST-COMPLETED":31074
END:VEVENT
BEGIN:VTODO
UID:3c1162e200207ff600000015000010b7
DTSTAMP:20011208T011645Z
SUMMARY:Joe’s Todo
DTSTART:20011208T004626Z
DUE:20020120T141500Z
CREATED:20011208T004626Z
LAST-MODIFIED:20011208T011000Z
PRIORITY:0
SEQUENCE:3
PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="jdoe@sesta.com";
X-NSCP-ORGANIZER-UID=jdoe;
X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20011208T004626Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020120T131500Z
```

```
ATTENDEE:MAILTO:jdoe@sesta.com
END:VALARM
X-NSCP-DUE-TZID:America/Los_AngelesX-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:
X-NSCP-ORGANIZR-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COPONENT-STATE;
X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":6538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:19700101T000000Z
X-NSCP-CALPROPS-CREATED:19700101T000000Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:susan
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20011010T001050Z
X-NSCP-CALPROPS-CREATED:20000929T180436Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:susan
X-NSCP-CALPROPS-NAME:default
X-NSCP-CALPROPS-PRIMARY-OWNER:susan
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^aÍ»X-NSCP-CALPROPS-ACCESS-CONTROL-E
NTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:fred^a^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:fred^c^^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3c1162b3000051c300000013000010b7
DTSTAMP:20011208T011645Z
SUMMARY: Susan’s event
DTSTART:20020110T110000Z
DTEND:20020110T120020Z
CREATED:20011208T004539Z
```

```
LAST-MODIFIED:20011208T011638Z
PRIORITY:0
SEQUENCE:4
ORGANIZER;SENT-BY="susan@sesta.com";
X-NSCP-ORGANIZER-UID=susan;
X-NSCP-ORGANIZER-SENT-BY-UID=susan:susan
STATUS:CONFIRMED
TRANSP:OPAQUE
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL;
PARTSTAT=ACCEPTED;CN="Mary Anderson";RSVP=TRUE;
X-NSCP-ATTENDEE-GSE-STATUS=2:marya
X-NSCP-ORIGINAL-DTSTART:20021225T213000Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020210T100000Z
ATTENDEE:MAILTO:marya@company22.com
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:susan@seata.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="REQUEST-COMPLETED":131074
END:VEVENT
BEGIN:VTODO
UID:3c1162e200207ff600000015000010b7
DTSTAMP:20011208T011645Z
SUMMARY:susan's todo
DTSTART:20011208T004626Z
DUE:20020120T141500Z
CREATED:20011208T004626Z
LAST-MODIFIED:20011208T011000Z
PRIORITY:0
SEQUENCE:3
PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="susan@sesta.com";
X-NSCP-ORGANIZER-UID=crowe;
X-NSCP-ORGANIZER-SENT-BY-UID=susan:susa
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20011208T004626Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020120T131500Z
```

```
ATTENDEE:MAILTO:susan@sesta.com
END:VALARM
X-NSCP-DUE-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:mailto:susan@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":65538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

fetch_deletedcomponents

Purpose.

Returns a list of deleted components from the `deletelog.db` for a specified time period.

Parameters.

Table 6-22 lists `fetch_deletedcomponents` parameters

Table 6-24 `fetch_deletedcomponents` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	A semicolon-separated list of calendar identifiers.	N	Current user's `calid`.
`component-type`	keyword (`event`, `todo`, `all`)	Indicates which components to return: `event` returns only events `todo` returns only todos (tasks) `all` returns both events and todos If an invalid value is passed in, the system assumes `all`.	N	`all`
`dtend`	ISO 8601 DateTime Z string	End time and date of events to be returned. A value of `0` means fetch all deleted components in the deletelog database until the end of time. The return value has the server time that will be used in the next fetch.	N	`0`
`dtstart`	ISO 8601 DateTime Z string	Start time and date of events to be returned. A value of `0` means fetch all deleted components in the deletelog database from the beginning of time.	N	`0`

Table 6-24 `fetch_deletedcomponents` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`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`
`recurring`	integer (`0`, `1`)	Whether to return all components in compressed form (1) or not (0). For `recurring=0`, it does not return the master entry. For `recurring=1`, it does not return the expanded instances of the recurring components; it returns the master entry plus any exceptions. If all the instances of the recurring series have been deleted, it returns `dtStart`, `dtEnd`, `rrules`, `rdate`, `exrule`, `exdate`, and `uid`.	N	0 (not compressed)
`tzid`	time zone ID string	Time zone to use if `dtstart`, or `dtend` parameters are not in Zulu time. For example, “America/Los_Angeles”	N	server’s default time zone
`tzidout`	time zone ID string	Time zone to report returned data in.	N	Zulu time

Description.

Use this command to retrieve a list of events and todos that have been deleted during a specific time period. For recurring format components, this command should be used in conjunction with `fetchcomponents_by_lastmod` in order to return recurring instances that are still active.

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 `text/calendar` format.

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 data 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 data does not contain the `var maxResults` statement.

Returns

When this command is called in compressed mode (with `recurring =1`), the query interface goes through the Delete Log database and returns all the non-repeating entries and the master components deleted that match the criteria. This pass ignores the recurring instances that are stored in the database. This will not return any master entries associated with the deleted recurring instances that are still active. Those active master entries will be returned using the `fetchcomponents_by_lastmod` command. If all the instances in a recurring chain are deleted, the master component will return `dtstart`, `dtend`, `rrules`, `rdate`, `exrule`, `exdate` and `uid`.

When the command is called in expanded mode (with `recurring=0`), the query interface goes through the Delete Log database and returns all instances of recurring components. Specifically, it does not return the master component.

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.

The following failure codes can be returned:

- `X-NSCP-WCAP-ERRNO:1` - Session ID timed out or Invalid session ID
- `X-NSCP-WCAP-ERRNO:28` - Command failed; user denied access to a calendar
- `X-NSCP-WCAP-ERRNO:29` - Command failed; calendar does not exist in the database
- `X-NSCP-WCAP-ERRNO:56` - Fetch deleted components failed
- `X-NSCP-WCAP-ERRNO:57` - Success but partial result

Example

Fetching Deleted Components (recurring defaults to 0)

```
http://webcalendarserver/fetch_deletedcomponents.wcap?d=8sh8ubh2rbl08u&fmt
-out=text/calendar&calid=jdoe
```

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-NSCP-CALPROPS-LAST-MODIFIED:20030110T222754Z
X-NSCP-CALPROPS-CREATED:20030110T221814Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:john doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-OWNERS:""
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^wdeic^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^sf^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3e224e5b000041c6000000010000664b
DTSTAMP:20030113T055314Z
DTSTART:20030114T060000Z
DTEND:20030114T070000Z
LAST-MODIFIED:20030113T052800Z
X-NSCP-TRIGGERED_BY:jdoe
END:VEVENT
BEGIN:VTODO
UID:3e2254bd000041c600000001000066eb
DTSTAMP:20030113T055517Z
DTSTART:20030113T055509Z
DUE:20030114T060000Z
LAST-MODIFIED:20030113T055513Z
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Fetching Deleted Components (In recurring format, recurring defaults to 0)

```
http://webcalendarserver/fetch_deletedcomponents.wcap?d=8sh8ubh2rbl08u&fmt
-out=text/calendar&calid=jdoe
```

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-NSCP-CALPROPS-LAST-MODIFIED:20030110T222754Z
X-NSCP-CALPROPS-CREATED:20030110T221814Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:john doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-OWNERS:""
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^wdeic^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^sf^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3e224e5b000041c6000000010000664b
DTSTAMP:20030113T055314Z
DTSTART:20030114T060000Z
DTEND:20030114T070000Z
LAST-MODIFIED:20030113T052800Z
X-NSCP-TRIGGERED_BY:jdoe
END:VEVENT
```

Fetching Deleted Components (In recurring format, recurring=1)

```
http://webcalendarserver/fetch_deletedcomponents.wcap?d=8sh8ubh2rbl08u&fmt
-out=text/calendar&calid=jdoe&recurring=1
```

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-NSCP-CALPROPS-LAST-MODIFIED:20030110T222754Z
X-NSCP-CALPROPS-CREATED:20030110T221814Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
```

```
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:john doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-OWNERS:""
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^wdeic^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^sf^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3e224e5b000041c6000000010000664b
DTSTAMP:20030113T055314Z
DTSTART:20030114T060000Z
DTEND:20030114T070000Z
LAST-MODIFIED:20030113T052800Z
X-NSCP-TRIGGERED_BY:jdoe
END:VEVENT
BEGIN:VEVENT
UID:3e225538000027810000000003000066eb
DTSTAMP:20030113T055758Z
DTSTART:20030114T060000Z
DTEND:20030114T070000Z
LAST-MODIFIED:20030113T055721Z
RRULE:FREQ=WEEKLY;INTERVAL=1;WKST=SU;COUNT=5
X-NSCP-TRIGGERED_BY:jdoe
END:VEVENT
BEGIN:VEVENT
UID:3e2255ed00000ff60000000a000066eb
DTSTAMP:20030113T060117Z
DTSTART:20030114T060000Z
DTEND:20030114T070000Z
LAST-MODIFIED:20030113T060107Z
EXDATE:20030116T060000Z
EXDATE:20030116T060000Z
EXDATE:20030116T060000Z
RRULE:FREQ=DAILY;INTERVAL=1;WKST=SU;COUNT=5
X-NSCP-TRIGGERED_BY:jdoe
END:VEVENT
BEGIN:VTODO
UID:3e2254bd000041c6000000001000066eb
DTSTAMP:20030113T055517Z
DTSTART:20030113T055509Z
DUE:20030114T060000Z
```

```
LAST-MODIFIED:20030113T055513Z
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

fetchevents_by_id

Purpose

Retrieve specific calendar events.

Parameters

Table 6-25 lists `fetchevents_by_id` parameters:

Table 6-25 `fetchevents_by_id` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	A semicolon-separated list of calendar identifiers.	N	Current user's `calid`.

`compressed`	integer (`0`,`1`)	This parameter has been deprecated for this release and may be removed from future versions.
		For `compressed=0`, returns less data. Specifically, it does not return the following parameters:
		`rrules`
		`rdate`
		`exrule`
		`exdate`
		For `compressed=1`, all recurrence data is returned.
			N	`0`

`compstate`	semicolon-separated list of component state keywords	The list of component states to fetch. For `compstate` values, see Table 6-6 on page 84.	N	`ALL`
`emailorcalid`	integer (`0`, `1`)	`0` = The `calid` is returned in the cal-address part of the ATTENDEE and ORGANIZER properties. `1` = The email address is returned in the cal-address part of the ATTENDEE and ORGANIZER properties. The X-Token X-S1CS-CALID contains the `calid` value.	N	`0`

Table 6-25 `fetchevents_by_id` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`id`	unique identifier string	The session identifier.	Y	N/A
`mod`	integer	A modifier indicating which recurrences to retrieve. One of the following values: 1 = `THISINSTANCE` 2 = `THISANDFUTURE` 3 = `THISANDPRIOR` 4 = `THISANDALL`	N	`1` (`THISINSTANCE`)
`recurring`	integer (`0`, `1`)	Whether to return all components in compressed form (1) or not (0). (Compressed form has master entry plus exceptions.)	N	0 (not compressed)
`relativealarm`	integer (0, 1, 2, 3, 4)	Return the alarm as relative or absolute. 0—Return alarm values as absolute. 1—Return alarm values as relative. 2—Return event alarms relative / todos alarm absolute. 3—Return event alarms absolute / todos alarm relative. 4—Return alarms as originally created.	N	0 (absolute)
`rid`	ISO 8601 DateTime string	The recurrence identifier for the event. For a nonrecurring event, set to `0`.	N	`0`
`tzid`	time zone ID string	Time zone to use if the `rid` parameter is not in Zulu time. For example, “America/Los_Angeles”	N	server’s default time zone
`tzidout`	time zone ID string	Time zone that returned data should be translated to.	N	Returns data in Zulu time
`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 “Recurring Components – Overview.”

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 format.

Returns

The system uses the `tzidout` parameter to determine what time zone to translate retrieved data into before returning it. If the `tzidout` parameter is missing, the system returns the data in Zulu time.

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

This query retrieves an event with a specific id.

```
http://webcalendarserver/fetchevents_by_id.wcap?id=bes6bbe2mu98uw9&calid=j
doe&uid=3c11625900005ffe00000011000010b7
&fmt-out=text/calendar
```

It returns one event:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
```

```
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3c11625900005ffe00000011000010b7
DTSTAMP:20011208T015845Z
SUMMARY:eventA
DTSTART:20011225T133000Z
DTEND:20011225T143000Z
CREATED:20011208T004409Z
LAST-MODIFIED:20011208T010857Z
PRIORITY:0
SEQUENCE:4
ORGANIZER;SENT-BY="jdoe@sesta.com";
X-NSCP-ORGANIZER-UID=jdoe;
X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:CONFIRMED
TRANSP:OPAQUE
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL;
PARTSTAT=ACCEPTED;CN="John Smith";RSVP=TRUE;
X-NSCP-ATTENDEE-GSE-STATUS=2:jsmith
X-NSCP-ORIGINAL-DTSTART:20020210T190000Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20011225T123000Z
ATTENDEE:MAILTO:jdoe@sesta.com
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="REQUEST-COMPLETED":31074
END:VEVENT
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

fetchtodos_by_id

Purpose

Retrieve specific calendar todos.

Parameters

Table 6-26 lists `fetchtodos_by_id` parameters:

Table 6-26 `fetchtodos_by_id` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	A semicolon-separated list of calendar identifiers.	N	Current user's `calid`.

`compressed`	integer (`0`,`1`)	This parameter has been deprecated for the current release and may be removed from future releases.
		For `compressed=0`, returns less data. Specifically, it does not return the following parameters:
		`rrules`
		`rdate`
		`exrule`
		`exdate`
		For `compressed=1`, all recurrence data is returned.
			N	`0`

`compstate`	semicolon-separated list of component state keywords	The list of component states to fetch. For `compstate` values, see Table 6-6 on page 84.	N	`ALL`
`emailorcalid`	integer (`0`, `1`)	`0` = The `calid` is returned in the cal-address part of the ATTENDEE and ORGANIZER properties. `1` = The email address is returned in the cal-address part of the ATTENDEE and ORGANIZER properties. The X-Token X-S1CS-CALID contains the `calid` value.	N	`0`

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`id`	unique identifier string	The session identifier.	Y	N/A
`mod`	integer	A modifier indicating which recurrences to retrieve. One of the following values: 1 = THISINSTANCE 2 = THISANDFUTURE 3 = THISANDPRIOR 4 = THISANDALL	N	`1` (`THISINSTANCE`)
`recurring`	integer (`0`, `1`)	Whether to return all components in compressed form (1) or not (0). (Compressed form has master entry plus exceptions.)	N	0 (not compressed)

Table 6-26 `fetchtodos_by_id` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`relativealarm`	integer (0, 1, 2, 3, 4)	Return the alarm as relative or absolute. 0—Return alarm values as absolute. 1—Return alarm values as relative. 2—Return event alarms relative / todos alarm absolute. 3—Return event alarms absolute / todos alarm relative. 4—Return alarms as originally created.	N	0 (absolute)
`rid`	ISO 8601 DateTime string	The recurrence identifier for the todo. For a nonrecurring todo, set to `0`.	N	`0`
`tzid`	time zone ID string	Time zone to use if the `rid` parameter is not in Zulu time. For example, “America/Los_Angeles”	N	server’s default time zone
`tzidout`	time zone ID string	Time zone the returned data should be translated to.	N	Returns data in Zulu time
`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 `text/calendar` format.

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 “Recurring Components – Overview.”

If the times specified in the `rid` parameter is not Zulu time, the system uses the time zone specified in the `tzid` parameter to translate the times into Zulu time for data retrieval. If the `tzid` parameter is missing, the system uses the server’s default time zone.

The system uses the `tzidout` parameter to determine what time zone to translate retrieved data into before returning it. If the `tzidout` parameter is missing, the system returns the data in Zulu time.

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, a todo "weekly todo B" that's due weekly at 5:00pm starting on Feb 1, 2002 and ending on Mar 1, 2002.

Example 1

This query fetches just the first todo, on Feb 1, 2002, because the recurrence ID of the first item is specified (`rid=20020201T170000Z`) but no modifier is specified, so it defaults to 1 (`THISINSTANCE`):

```
http://webcalendarserver/fetchtodos_by_id.wcap?
id=n3o3m05sx9v6t98t8u2p&uid=3c15309d000037020020021400003189&
rid=20020201T170000Z&fmt-out=text/calendar
```

The following output is generated:

```
BEGIN:VCALENDARPRODID:-//SunONE/Calendar Hosting Server//EN

METHOD:PUBLIS
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VTODO
```

```
UID:3c15309d000037020020021400003189
RECURRENCE-ID:20020201T170000Z
DTSTAMP:20011210T222131Z
SUMMARY:weekly todo B
DTSTART:20020201T170000Z
DUE:20020201T170000Z
CREATED:20011210T220101Z
LAST-MODIFIED:20011210T220101Z
PRIORITY:0
SEQUENCE:0
PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="jdoe@sesta.com";
X-NSCP-ORGANIZER-UID=jdoe;
X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20020201T170000Z
X-NSCP-LANGUAGE:en
X-NSCP-DUE-TZID:Europe/London
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":65538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Example 2

This query fetches the last two recurrences by specifying the recurrence ID of the second to last recurrence on Feb. 22, 2002 (`rid=20020222T170000Z`) and a modifier of 2 (`mod=2`) which means `THISANDFUTURE` recurrences:

```
http://webcalendarserver/fetchtodos_by_id.wcap?
id=n3o3m05sx9v6t98tu2p&uid=3c15309d000037020020021400003189&
rid=20020222T170000Z&mod=2&fmt-out=text/calendar
```

The results of the query are as follows:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
```

```
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VTODO
UID:3c15309d000037020020021400003189
RECURRENCE-ID:20020222T170000Z
DTSTAMP:20011210T222757Z
SUMMARY:weekly todo B
DTSTART:20020222T170000Z
DUE:20020222T170000Z
CREATED:20011210T220101Z
LAST-MODIFIED:20011210T220101Z
PRIORITY:0
SEQUENCE:0
PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="jdoe@sesta.com";
X-NSCP-ORGANIZER-UID=jdoe;
X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20020222T170000Z
X-NSCP-LANGUAGE:en
X-NSCP-DUE-TZID:Europe/London
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":65538
END:VTODO
BEGIN:VTODO
UID:3c15309d000037020020021400003189
RECURRENCE-ID:20020301T170000Z
```

```
DTSTAMP:20011210T222757Z
SUMMARY:weekly todo B
DTSTART:20020301T170000Z
DUE:20020301T170000Z
CREATED:20011210T220101Z
LAST-MODIFIED:20011210T220101Z
PRIORITY:0
SEQUENCE:0
PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="jode@sesta.com";
X-NSCP-ORGANIZER-UID=jdoe;
X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20020301T170000Z
X-NSCP-LANGUAGE:en
X-NSCP-DUE-TZID:Europe/London
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":65538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

get_all_timezones

Purpose

Retrieve data about all timezones supported by the server.

Parameters

Table 6-27 lists get_all_timezones parameters:

Table 6-27 get_all_timezones Parameters

Parameter	Type	Purpose	Required	Default
dtend	ISO 8601 DateTime Z string	End date of the crossover values to retrieve. A value of 0 means get all crossover dates until the last known year (2087).	N	0
dtstart	ISO 8601 DateTime Z string	Start date of crossover values to retrieve. A value of 0 means get all crossover dates from the first known year (1987).	N	0

Table 6-27 `get_all_timezones` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`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 `text/calendar` format.

Error Codes
If there was an error in getting the timezones, the server returns the error `GET_ALL_TIMEZONES_FAILED(24)`.

Example
The first example shows the command output. The second example is a crossover array.

Example 1

This query gets all time zones.

```
http://calendarserver/get_all_timezones.wcap?
id=2m2ns6w9x9h2mr6p3b&fmt-out=text/calendar
```

This is the result of the query:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:19700101T000000Z
X-NSCP-CALPROPS-CREATED:19700101T000000Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:default
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^wdeic^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^sf^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VTIMEZONE
TZID:Africa/Amman
BEGIN:STANDARD
DTSTART:19950920T000000
TZOFFSETFROM:+0300
TZOFFSETTO:+0200
TZNAME:EEST
RRULE:FREQ=YEARLY;BYDAY=-1FR;BYMONTH=9
END:STANDARD
BEGIN:DAYLIGHT
DTSTART:19930420T000000
TZOFFSETFROM:+0200
TZOFFSETTO:+0300
TZNAME:EEDT
RRULE:FREQ=YEARLY;BYDAY=-1FR;BYMONTH=4
END:DAYLIGHT
END:VTIMEZONE
BEGIN:VTIMEZONE
TZID:Africa/Cairo
BEGIN:STANDARD
DTSTART:19950924T000000
TZOFFSETFROM:+0300
TZOFFSETTO:+0200
TZNAME:EEST
COMMENT:this is a comment
RRULE:FREQ=YEARLY;BYDAY=-1FR;BYMONTH=9
END:STANDARD
BEGIN:DAYLIGHT
```

```
DTSTART:19950420T000000
TZOFFSETFROM:+0200
TZOFFSETTO:+0300
TZNAME:EEDT
RRULE:FREQ=YEARLY;BYDAY=-1FR;BYMONTH=4
END:DAYLIGHT
END:VTIMEZONE
BEGIN:VTIMEZONE
```

...(other time zones omitted to conserve space)

```
BEGIN:VTIMEZONE
TZID:Pacific/Tongatapu
BEGIN:STANDARD
DTSTART:19970101T000000
TZOFFSETFROM:+1300
TZOFFSETTO:+1300
TZNAME:TOT
TZNAME:PHOT
END:STANDARD
END:VTIMEZONE
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Example 2

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','20020404T100000Z','20021031T090000Z',
'20020402T100000Z','20021029T090000Z','20020401T100000Z',
'20021028T090000Z', '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 are 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 6-28 lists `get_calprops` parameters:

Table 6-28 `get_calprops` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	semicolon-separated list of strings	Semicolon-separated list of calendar identifiers from which to retrieve properties.	N	Current user's `calid`.

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`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 write, `1` = can write)
- 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, `errno` is set to `ACCESS_DENIED_TO_CALENDAR` (28).

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`, `jsmith`, and `susan`, in that order.

This is the URL:

```
http://webcalendarserver/get_calprops.wcap?id=2mu95r5so0hq68ts6q3
&calid=jdoe;jsmith;susan&fmt-out=text/calendar
```

This is the returned data:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
```

```
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20011115T234638Z
X-NSCP-CALPROPS-CREATED:20011115T234638Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jsmith
X-NSCP-CALPROPS-NAME:James Smith
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jsmith
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^wdeic^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^sf^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20011130T002458Z
X-NSCP-CALPROPS-CREATED:20010914T015956Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
```

```
X-NSCP-CALPROPS-RELATIVE-CALID:susan
X-NSCP-CALPROPS-NAME:Susan Anderson
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:susan
X-NSCP-CALPROPS-OWNERS:jdoe
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^rsf^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

get_freebusy

Purpose

Get the freebusy information for users.

Parameters

Table 6-29 lists `get_freebusy` parameters:

Table 6-29 `get_freebusy` Parameters

Parameter	Type	Purpose	Required	Default
`busyonly`	Integer (`0`, `1`)	`0` = return both busy and free periods `1` = return only busy periods	N	`1`
`calid`	string	The calendar identifier.	Y	Current user's default calendar.
`dtstart`	ISO 8601 DateTime Z string	Start time of freebusy search.	Y	N/A
`dtend`	ISO 8601 DateTime Z string	End time of freebusy search.	Y	N/A
`duration`	Integer	Freebusy duration time in number of days.	N	60 or Default taken from `ics.conf`

`fmt-out`	string	The format for the returned data.
		Format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`
				Default taken from `ics.conf`

Table 6-29 `get_freebusy` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`freebusybegin`	integer	Offset in number of days from the value of `ics.conf` setting `service.wcap.freebusybegin`. Backs off the date range by the value of this parameter. For example, a value of 30 would start the freebusy range 30 days before the current time found in the `ics.conf` parameter.	N	Default `ics.conf` value is 30.
`freebusyend`	integer	Offset in number of days from the value of `ics.conf` setting `service.wcap.freebusyend` to calculate the end of the freebusy range. Extends the date found in the `ics.conf` parameter. For example, a value of 30 would put the end date 30 days beyond the current setting.	N	Default `ics.conf` value is 30 days.
`id`	unique identifier string	The session identifier.	Y	N/A
`tzid`	timezone-ID string	Default time zone to use if `dtstart`, or `dtend` parameters do not have a time zone specified. For example, “America/Los_Angeles”	N	Server’s default timezone.
`tzidout`	time zone ID string	Time zone to report returned data in.	N	Zulu time

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.

The component freebusy time will be calculated for a time period that can be specified in one of three ways:

- `duration` parameter

  The default value of the `duration` parameter is taken from the `ics.conf` setting `service.wcap.freebusyduration`. The standard default is `60` days. This is the default taken if none of the time period parameters are passed in.

- `dtstart` and `dtend` parameters

  The absolute start and end times to use for this freebusy calculation. There is no default for these parameters.

- `freebusybegin` and `freebusyend` parameters.

  The relative beginning and end times to include in the freebusy calculation. If these are specified with no value, the default is `30` for each.

If conflicting parameters are passed in, the `duration` parameter overrides the other two types.

For further information about how freebusy calendars are specified, see Freebusy Calendars and Freebusy Calculation for Private Events.

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:

10:00-11:00	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 2002 and July 1 2002.

The output is returned in `text/calendar` format.

```
http://webcalendarserver/get_freebusy.wcap?id=2mu95r5so0hq68ts6q3&calid=js
un&dtstart=20020501T112233Z&dtend=20020701T112233Z&fmt-out=text/calendar
```

Here is the output:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20010517T012259Z
X-NSCP-CALPROPS-CREATED:20010517T012259Z
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:20020501T112233Z
DTEND:20020701T112233Z
FREEBUSY;FBTYPE=FREE:20020501T112233Z/20020518T170000Z
FREEBUSY;FBTYPE=BUSY:20020518T170000Z/20020518T190000Z
FREEBUSY;FBTYPE=FREE:20020518T190000Z/20020525T170000Z
FREEBUSY;FBTYPE=BUSY:20020525T170000Z/20020525T190000Z
FREEBUSY;FBTYPE=FREE:20020525T190000Z/20020601T170000Z
FREEBUSY;FBTYPE=BUSY:20020601T170000Z/20020601T190000Z
FREEBUSY;FBTYPE=FREE:20020601T190000Z/20020608T170000Z
FREEBUSY;FBTYPE=BUSY:20020608T170000Z/20020608T190000Z
FREEBUSY;FBTYPE=FREE:20020608T190000Z/20020615T170000Z
FREEBUSY;FBTYPE=BUSY:20020615T170000Z/20020615T190000Z
FREEBUSY;FBTYPE=FREE:20020615T190000Z/20020622T170000Z
FREEBUSY;FBTYPE=BUSY:20020622T170000Z/20020622T190000Z
FREEBUSY;FBTYPE=FREE:20020622T190000Z/20020629T170000Z
FREEBUSY;FBTYPE=BUSY:20020629T170000Z/20020629T190000Z
FREEBUSY;FBTYPE=FREE:20020629T190000Z/20020701T112233Z
END:VFREEBUSY
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

get_guids

Purpose

Generate a set of globally unique identifiers.

Parameters

Table 6-30 lists get_guids parameters:

Table 6-30 get_guids Parameters

Parameter	Type	Purpose	Required	Default
guidCount	integer	Number of GUIDs to return.	N	1
fmt-out	string	The format for the returned data. The three format types: text/calendar text/xml	N	text/calendar

Description.

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

Example

http://*webcalendarserver*/get_guids.wcap?guidCount=10
&fmt-out=text/calendar

```
BEGIN:VCALENDAR
VERSION:6.0
PRODID:SunONE Calendar Server 6.0
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:e5e4b537b0260000110000000c3000000
X-NSCP-GUID7:e5e4b537c263000012002002c3000000
X-NSCP-GUID8:e5e4b537241f000013000000c3000000
X-NSCP-GUID9:e5e4b537e7330000140000000c3000000
END:VCALENDAR
```

gettime

Purpose

Gets the server time for the requested calendars.

Parameters

Table 6-29 lists `gettime` parameters:

Table 6-31 `gettime` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	semicolon-separated list of strings	A list of calendar identifiers.	N	Current user's calid.

`fmt-out`	string	The format for the returned data.
		Format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`id`	unique identifier string	The session identifier.	Y	N/A
`tzidout`	time zone ID string	Time zone to report returned data in.	N	Zulu time

Description

Calendars must have given read permission to the user requesting the server time. Returns the server time of the server where the calendar is stored.

Error Codes

- `X-NSCP-WCAP-ERRNO:1` - Session ID timed out or Invalid session ID
- `X-NSCP-WCAP-ERRNO:28` - Command failed; user denied access to a calendar
- `X-NSCP-WCAP-ERRNO:29` - Command failed; calendar does not exist in the database
- `X-NSCP-WCAP-ERRNO:55` - Get Server time Failed

Example

Valid session with tzidout

`http://webcalendarserver/gettime.wcap?id=br6e8vx9ek02n2ow9&calid=jdoe&tzidout=America/Los_Angeles`

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
VERSION:2.0
X-NSCP-WCAPTIME:20021021T082743
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

get_userprefs

Purpose

Retrieve the calendar preferences for the current user.

Parameters

Table 6-32 lists `get_userprefs` parameters:

Table 6-32 `get_userprefs` Parameters

Parameter	Type	Purpose	Required	Default

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`id`	unique identifier string	The session identifier.	Y	N/A
`userid`	string	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.

See the "*Administrator's Guide*" for detailed information about server preferences.

Note that all nodes under the `basedn` must be set to allow anyone read and search access rights in order for this command to work. For more information, see the Common Topic "Access Control Entries" on page 72.

Example

The following URL retrieves user preferences for the current user:

```
http://webcalendarserver/get_userprefs.wcap?id=b5q2o8ve2rk02nv9t6&
calid=jdoe&fmt-out=text/calendar
```

This is the data returned:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-WCAP-PREF-cn:John Doe
X-NSCP-WCAP-PREF-givenName:John
X-NSCP-WCAP-PREF-mail:jdoe@sesta.com
X-NSCP-WCAP-PREF-preferredlanguage:
X-NSCP-WCAP-PREF-sn:Doe
X-NSCP-WCAP-PREF-icsCalendar:jdoe
X-NSCP-WCAP-PREF-icsTimezone:Europe/London
X-NSCP-WCAP-PREF-icsDefaultSet:
X-NSCP-WCAP-PREF-icsFirstDay:
X-NSCP-WCAP-PREF-icsSet:name=mygroup$calendar=lucy\;jjones\;jdoe

TimeZone$tzmode=specify$tz=America/Denver$mergeInDayView=true
$description=
X-NSCP-WCAP-PREF-icsSubscribed:lucy$,jjones$,jsmith:jdoe
X-NSCP-WCAP-PREF-icsFreeBusy:jdoe
X-NSCP-WCAP-PREF-ceInterval:PT0H30M
X-NSCP-WCAP-PREF-ceDayTail:19
X-NSCP-WCAP-PREF-ceDefaultView:overview
X-NSCP-WCAP-PREF-ceColorSet:pref_group4
X-NSCP-WCAP-PREF-ceToolText:1
X-NSCP-WCAP-PREF-ceToolImage:1
X-NSCP-WCAP-PREF-ceFontFace:PrimSansBT,Verdana,sans-serif
X-NSCP-WCAP-PREF-ceExcludeSatSun:0
X-NSCP-WCAP-PREF-ceGroupInviteAll:1
X-NSCP-WCAP-PREF-ceSingleCalendarTZID:0z
X-NSCP-WCAP-PREF-ceAllCalendarTZIDs:0
X-NSCP-WCAP-PREF-ceNotifyEnable:0
X-NSCP-WCAP-PREF-ceNotifyEmail:jdoe@sesta.com
X-NSCP-WCAP-PREF-ceDefaultAlarmStart:P15M
X-NSCP-WCAP-PREF-ceDefaultAlarmEmail:jdoe@sesta.com
X-NSCP-WCAP-PREF-nswcalCALID:jdoe
X-NSCP-WCAP-PREF-icsDWPHost:DWPserver1
X-NSCP-WCAP-PREF-icsCalendarOwned:jdoe$John's
Calendar,jdoe:personal$John's Personal Calendar
X-NSCP-WCAP-SERVER-PREF-allowchangepassword:no
```

```
X-NSCP-WCAP-SERVER-PREF-allowcreatecalendars:yes
X-NSCP-WCAP-SERVER-PREF-allowdeletecalendars:
X-NSCP-WCAP-SERVER-PREF-allowpublicwritablecalendars:
X-NSCP-WCAP-SERVER-PREF-validateowners:no
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Other user's preferences can be retrieved. If LDAP proxy authorization is turned on, displaying preferences of other users are subject to LDAP access control. If LDAP proxy authorization is turned off, anonymous searches are performed on LDAP.

The following ACI binds to the LDAP server as `calmaster` and `allowproxy` for other users. The ACI is set for domain `sesta.com`.

```
dn: o=sesta.com
changetype: modify
add: aci
aci:
(target="ldap:///uid=*,o=sesta.com")(targetattr=*(version3.0;acl"allowAll-
calmaster";allow
(all)(userdn="ldap:///uid=calmaster,o=sesta.com");)
aci:
(target="ldap///uid=*,o=siroe.com")(targetattr="*")(version3.0;acl"allowpr
oxy-calmaster";allow
(proxy)(userdn="ldap:///uid=*,o=sesta.com");)
```

The following is a sample output of the following commands:

```
wc
/get_userprefs.wcap?id=t95qm0n0es3bo35r&fmt-out=text/calendar&userid=jdoe
wc
/get_userprefs.wcap?id=t95qm0n0es3bo35r&fmt-out=text/calendar&userid=mailt
o:sue@sesta.com
wc
/get_userprefs.wcap?id=t95qm0n0es3bo35r&fmt-out=text/calendar&userid=john1
23abc
```

```
GET
/get_userprefs.wcap?id=eo38ue2q2rq6r68u&fmt-out=text/calendar&userid=jdoe
```

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-NSCP-WCAP-PREF-cn:JohnDoe,TEST TEST-2
X-NSCP-WCAP-PREF-uid:jdoe
X-NSCP-WCAP-PREF-mail:jdoe@sesta.com
```

```
X-NSCP-WCAP-PREF-givenName:John
X-NSCP-WCAP-PREF-sn:Doe
X-NSCP-WCAP-PREF-icsCalendar:jdoe
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

```
GET
/get_userprefs.wcap?id=eo38ue2q2rq6r68u&fmt-out=text/calendar&userid=mailt
o:sue@sesta.com
```

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-NSCP-WCAP-PREF-cn:Sue Smith
X-NSCP-WCAP-PREF-uid:Sue
X-NSCP-WCAP-PREF-mail:sue@sesta.com
X-NSCP-WCAP-PREF-givenName:Sue
X-NSCP-WCAP-PREF-sn:Smith
X-NSCP-WCAP-PREF-icsCalendar:sue
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

```
GET
/get_userprefs.wcap?id=eo38ue2q2rq6r68u&fmt-out=text/calendar&userid=john1
23abc
```

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISHx
VERSION:2.0
X-NSCP-WCAP-ERRNO:61
END:VCALENDAR
```

import

Purpose

Import events and todos from a file to a calendar.

Parameters

Table 6-33 lists `import` parameters:

Table 6-33 `import` Parameters

Parameter	Type	Purpose	Required	Default
`appid`	string	A runtime parameter (not stored in the database) that specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the *SunONE Messaging and Collaboration Event Notification Service Manual*.	N	N/A
`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:
		`text/calendar`
		`text/xml`
			Y	N/A

`dtend`	ISO 8601 DateTime string	End time and date of the events and todos to import. A value of `0` means import all components from the start date to the last date in the file.	N	`0`
`dtstart`	ISO 8601 DateTime string	Start time and date of events and todos to import. 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:20020105T100000Z
DTEND:20020105T110000Z
DTSTAMP:20010104T120020Z
CREATED:20010105T110000Z
LAST-MODIFIED:20010104T120020Z
SUMMARY:Weekly QA Meeting
UID:random-uid001
END:VEVENT
BEGIN:VEVENT
DTSTART:20020106T100000
DTEND:20020106T110000
DTSTAMP:20010104T120020
CREATED:20010105T110000Z
LAST-MODIFIED:20010104T120020Z
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=t95qm0n0es3bo35r&cal
id=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 6-34 lists `login` parameters:

Table 6-34 `login` Parameters

Parameter	Type	Purpose	Required	Default

`fmt-out`	string	Specifies the desired output format.
		Specify `text/html` to log in to the Calendar Express user interface. This format type is invalid in all other commands.
		If you are not logging into the Calendar Express user interface, choose one of the three output format types that follows:
		`text/calendar`
		`text/xml`
		When `text/calendar` or `text/xml` is specified, the `refresh` parameter is automatically set to `1`.
			N	`text/calendar`

`lang`	enum	The user's preferred language.	N	`NULL`
`password`	string	The user's password.	N	N/A
`proxyauth`	string	Used by calendar administrators to perform proxy authorization.	N	N/A
`refresh`	integer (0, 1)	A boolean indicating whether to return just the new session ID or everything. 1 = Return only the session identifier. 0 = Return everything.	N	`1`
`user`	string	The user's name.	N	`NULL`

Description.

This command logs a specific user into 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.

fmt-out=text/html

This data type is allowed in only one command in WCAP, `login`. It is for the express purpose of logging into the SHTML user interface. When `fmt-out=text/html` occurs in `login`, the command is redirected to `command.shtml`, which connects the user to the Calendar Server user interface. Since this data type is not the default for the parameter, you must specify it explicitly when logging in to the Calendar Server interface.

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 "csIAccessControl" on page 23). For more information on the Proxy Authentication SDK, see Chapter 3, "Proxy Authentication SDK Overview" for the overview and Chapter 4, "Proxy Authentication SDK Reference" 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, the default value is `1`.

With the `refresh` parameter set to `1`, 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 output containing the location of the entry page to the 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 2002 22:38:33 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache
```

```
Content-Length: 0
Last-modified: Tue, 11 May 2002 22:38:33 GMT
Location:
http://webcalendarserver/en/main.html?id=er6en05tv6n3bv9&lang=en
 &host=http://webcalendarserver/
```

logout

Purpose

Terminate the current user's session.

Parameters

Table 6-35 lists logout parameters:

Table 6-35 logout Parameters

Parameter	Type	Purpose	Required	Default
fmt-out	string	The format for the returned data. The three format types: text/calendar text/xml	N	text calendar
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 2002 21:31:42 GMT
Content-type: text/html; charset=iso-8859-1
Content-length: 190
Last-modified: Thu, 03 Jun 2002 21:31:42 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache
```

search_calprops

Purpose

Search for a calendar's properties.

Parameters

Table 6-36 lists `search_calprops` parameters:

Table 6-36 `search_calprops` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	integer (0,1)	A boolean indicating whether or not to search the `calid` property. `1` = Search the `calid` property `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 6-36 search_calprops Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
name	integer (0,1)	A boolean indicating whether or not to search the name property. 1 = Search the name property 0 = Do not search it.	N	0
primaryOwner	integer (0,1)	A boolean indicating whether or not to search the primaryOwner property. 1 = Search the primaryOwner property. 0 = Do not search it.	N	0
searchOpts	integer 0,1,2,3	How to perform the search. One of the following: 0 = CONTAINS 1 = BEGINS_WITH 2 = ENDS_WITH 3 = EXACT	N	0
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 "jdoe"

http://*webcalendarserver*/search_calprops.wcap?id=n3o3m05sx9v6t98t8u2p&search-string=jdoe&primaryOwner=1&searchOpts=0&maxResults=50&fmt-out=text/calendar

The following data is a result of the example URL above:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
BEGIN:VCALENDAR
```

```
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
X-NSCP-CALPROPS-LAST-MODIFIED:20010917T213724Z
X-NSCP-CALPROPS-CREATED:20010917T213724Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe:sports
X-NSCP-CALPROPS-NAME:Sports Calendar
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-RESOURCE:0
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

set_calprops

Purpose

Set the calendar properties of a calendar.

Parameters

Table 6-37 lists set_calprops parameters:

Table 6-37 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. 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

Table 6-37 `set_calprops` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`description`	string	The description of the calendar.	N	N/A

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`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

`read`	integer	This parameter has been deprecated in favor of the `acl` parameter. It remains here only for backwards compatibility.
		The new read-access value of the calendar. The value can be one of the following:
		`0 PRIVATE`
		`1 PUBLIC`
		`4 PRIMARY_OWNER_ONLY`
			N	N/A

`tzid`	string	The new timezone identifier for this calendar.	N	,,,,

`write`	integer	This parameter has been deprecated in favor of the `acl` parameter. It remains here only for backwards compatibility.
		The new write-access value of the calendar. The value can be one of the following:
		`0 PRIVATE`
		`1 PUBLIC`
		`4 PRIMARY_OWNER_ONLY`
			N	N/A

Description.

This command is an update command, that is, it only changes the values of the parameters you specify. 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. The functionality has been replaced with the `acl` parameter. They are included for backwards compatibility 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 email 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=jdoe&categ
ories=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 72, in the Common Topics section at the front of this chapter. Note that due to limitations of the user interface, it is advisable to limit the number of individuals listed in the ACEs to a maximum of 75 per calendar.

Freebusy Access

See "Freebusy Calendars" on page 86, in the Common Topics section at the front of this chapter.

Choosing a Different Language or Character Set

See "Changing Language or Character Set" on page 77, in the Common Topics section at the front of this chapter.

set_userprefs

Purpose

Modify the preferences or password for a session.

Parameters

Table 6-38 lists `set_userprefs` parameters:

Table 6-38 `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`. When set to `0`, the parameter is ignored.	N	`0`
`del_attrs`	string	Delete an existing preference.	N	N/A

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			Y	`text/calendar`

`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&userid=j
doe&set_attrs=ceBgcolor=black
```

storeevents

Purpose

Add events to a calendar.

Parameters

Table 6-39 lists `storeevents` parameters:

Table 6-39 `storeevents` Parameters

Parameter	Type	Purpose	Required	Default
`alarmAudio`	ISO 8601 Date Time string	The time at which to sound an audio alarm.	N	N/A
`alarmDescription`	string	The message sent out with the alarm	N	“This is the alarm description”
`alarmEmails`	semicolon-separated list of email addresses	Recipients of alarm notifications for the event.	N	N/A
`alarmFlashing`	ISO 8601 Date Time string	The time at which to run flashing alarm.	N	N/A
`alarmPopup`	ISO 8601 Date Time string, or ISO 8601 Duration string	The time at which to pop up a dialog alarm.	N	N/A
`alarmStart`	ISO 8601 Date Time string, or ISO 8601 Duration string	The time at which to send the event alarm notification.	N	N/A
`appid`	string	A runtime parameter (not stored in the database) that specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the *SunONE Messaging and Collaboration Event Notification Service Manual*.	N	N/A
`attachments`	semicolon-separated list of strings	This is for iCalendar interoperability only. The strings are URLs.	N	N/A

Table 6-39 `storeevents` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`attendees`	semicolon-separated list of strings	An event's iCalendar RFC 2445 attendee properties. For a list of the properties understood by Calendar Server, see Table 6-7. There is one optional property that is specific only to Sun ONE Calendar Server: `SENT-STATUS`, which can have the value of: `NOT-SENT` or `SENT-SUCCEEDED`. The default for this property is `NOT-SENT`. If this property is set to `SENT-SUCCEEDED`, the the Group Scheduling Engine (GSE) will not process this attendee.	N	N/A
`calid`	string	Calendar identifier (or email address of `calid`) in which to store the event.	Y	N/A
`categories`	semicolon-separated list of strings	The event categories.	N	N/A
`charset`	string	The character set for the calendar.	N	N/A

`compressed`	integer (`0`,`1`)	This parameter has been deprecated for this release and may be removed from future releases.
		For `compressed=0`, returns less data. Specifically, it does not return the following parameters:
		`rrules`
		`rdate`
		`exrule`
		`exdate`
		For `compressed=1`, all recurrence data is returned.
			N	`0`

`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. To include spaces in the string, use the code `%20`.	N	Value of `summary` parameter.
`dtend`	ISO 8601 Date Time string	Event end time and date.	N	N/A

Table 6-39 `storeevents` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`dtstart`	ISO 8601 Date Time string	Event start time and date. Required to create or modify events.	 Y	N/A
`duration`	ISO 8601 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 ISO 8601 Date Time strings	Event exclusionary recurrence dates. To successfully create events, the `rrules` parameter must be used in conjunction with this parameter.	N	N/A
`exrules`	semicolon-separated list of strings	Event exclusionary recurrence rules. A semicolon-separated list of recurrence-rule strings. Each rule value must be enclosed in quotes. See "Recurring Components – Overview."	N	N/A
`fetch`	integer (0,1)	A boolean indicating whether or not to fetch and return newly stored todos. `1` = Fetch and return newly stored todos. `0` = Do not fetch.	N	`0`

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`geo`	two semicolon-separated floats	Semicolon-separated string of two float numbers representing the event's geographical location (latitude and longitude). For example, `37.31;-123.2`.	N	`0;0`

`icsClass`	string	Event class.
		One of the following values:
		`PUBLIC`
		`PRIVATE`
		`CONFIDENTIAL`
			N	`PUBLIC`

`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. `1` = Lasts all day. `0` = Does not last all day.	N	`0`

Table 6-39 `storeevents` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`language`	string	Language of event. (For example, “en”, “fr”, “de”)	N	N/A
`location`	string	Event location.	N	“”
`method`	integer (1,2,4,8,16,32)	ITIP method for group scheduling. 1 = PUBLISH (organizer only uses this) 2 = REQUEST (organizer only uses this) 4 = REPLY (attendees only use this) 8 = CANCEL (organizer only uses this) 16 = MOVE 32 = COUNTER (only attendees use this)	Y	1 (PUBLISH)
`mod`	integer	Specifies the recurrences to modify. Not required for creating events. Required to modify events. One of the following values: 1 = THISINSTANCE 2 = THISANDFUTURE 3 = THISANDPRIOR 4 = THISANDALL	 N Y	N/A
`notify`	integer (0,1)	This has been deprecated. It remains only for backward compatibility. The Group Scheduling Engine (GSE) module now takes care of all email notifications. A boolean indicating whether or not to notify attendees of a change to an event. 1 = Notify attendees. 0 = Do not notify attendees.	N	0
`orgCalid`	string	Calendar identifier of organizer. One of the following parameters must be specified: orgCalid, orgEmail, or orgUID.	N	N/A
`orgCN`	string	Common name of the organizer.	N	N/A
`orgEmail`	email address	Email address of the event contact (usually the organizer). One of the following parameters must be specified: orgCalid, orgEmail, or orgUID.	N	N/A
`orgUID`	`userid`	The `userid` of the organizer. One of the following parameters must be specified: orgCalid, orgEmail, or orgUID.	N	N/A

Table 6-39 `storeevents` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`priority`	integer (0-9)	Event priority. `0` = lowest 9= highest	N	`0`
`rchange`	integer (`0`,1)	A boolean indicating whether or not to expand a recurring event. `1` = expand `0` = do not expand	N	`0`
`rdates`	semicolon-separated list of ISO 8601 Date Time strings	Event recurrence dates. To successfully create events, the `rrules` parameter must also be specified.	N	N/A
`relatedTos`	semicolon-separated list of quoted strings	Other events to which this event is related.	N	N/A
`replace`	Integer (0,1)	A boolean. For parameters with semicolon-separated values: `1` = update (replace the old values with the new passed-in values) `0` = append (add the new passed-in values to the old ones)	N	`0`
`resources`	semicolon-separated list of strings	The resources associated with the event.	N	N/A
`rid`	ISO 8601 DateTime string	Event recurrence identifier. Not required to create events. If this parameter is not set when trying to modify events, the whole series of events is modified.	 N N	N/A
`rrules`	semicolon-separated list of strings	Event recurrence rules. A semicolon-separated list of recurrence-rule strings. Each rule value must be enclosed in quotes. See “Recurring Components – Overview.”	N	N/A
`seq`	integer	Event sequence number.	N	`0`
`smtp`	integer (0, 1)	Send email invitation to user without calendar. 0—No 1—Yes	N	`1`

Table 6-39 `storeevents` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`status`	integer	The event status code. One of the following values: `0` CONFIRMED `1` CANCELLED `2` TENTATIVE `3` NEEDS_ACTION `4` COMPLETED `5` IN_PROCESS `6` DRAFT `7` FINAL	N	N/A
`storetype`	integer	Designates whether an explicit “create” or “modify” is attempted on an event. An error results if an attempt is made to create an event that already exists, or to modify an event that does not exist. The error returned is `STOREEVENTS_FAILED (14)` The following values are valid: 0 WCAP_STORE_TYPE_NONE 1 WCAP_STORE_TYPE_CREATE 2 WCAP_STORE_TYPPE_MODIFY If the attribute is not passed or has a value of `0`, no error conditions are reported.	N	0
`summary`	string	Event summary. A string of any length. Required for new events; not required for modifying events. To include spaces in the string, use the code `%20`.	Y/N	Default `summary` available for new events
`transparent`	integer (`0`, `1`)	Is the private event transparent (`1`), or opaque (`0`)? If it is transparent, exclude this private event from freebusy calculations. If opaque, then include it in freebusy calculations. If the isAllDay parameter is set to `1`, the default is transparent instead of opaque.	N	`0` (opaque) `1` (transparent, if `isAllDay=1`)
tzid	time zone ID string	The time zone used to translate dates to Zulu time for storage. If this parameter is missing, and the time string has no Z after it, the calendar server time zone ID is used.	N	Calendar server time zone ID.
tzidout	time zone ID string	Time zone returned data should be translated to.	N	Returns data in Zulu time

Table 6-39 `storeevents` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`uid`	string	Unique identifier of the event to be stored. System generated for new events. Required to modify events.	 N Y	`Default uid` available for new events.
`X-`*property name*	string	One or more X properties, in iCalendar 2445 format (note that WCAP uses ^ as the separator)	N	N/A

Description.

Use this command to creates or modify 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 "Recurring Components – Overview."

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 "Changing Language or Character Set" on page 77 for a list of possible language codes.

For an explanation of how to use the `attendee` and `method` parameters to do group scheduling, see the Common Topics section "Group Scheduling" on page 88.

For an explanation of how to replace, append or delete a parameter, see the explanation in the Common Topics section "Updating Parameter Values" on page 100.

The server does not support attachments. The `attachments` parameter exists to support iCalendar interoperability only, and is not functional.

It is possible to delete an attendee in an existing meeting by assigning the value `X-NSCP-WCAP-ATTENDEE-DELETE` to the `attendee` parameter `PARTSTAT`. For example, to delete attendee jdoe, the attendee parameter would contain the following:

```
PARTSTAT=X-NSCP-WCAP-ATTENDEE-DELETE^jdoe
```

Required Parameters

This command creates new events and modifies existing events. You can not add and modify events in the same command. You must do one or the other.

There is a different set of parameter requirements for both cases:

- To create new events requires only the dtstart parameter.

  Every other parameter is optional. The server generates the uid.

- To modify existing events requires two parameters:

  - uid
  - mod

  All other parameters are optional. If a parameter is not specified, the event will retain the previous value of the property.

Duration and Dtend

The ending date (dtend) overrides duration. If you specify both duration and dtend, the command ignores duration.

Duration strings can be used in three parameters: duration, alarmPopup and alarmStart.

Specify the duration in iCal format. For example:

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

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

Returns

The command returns the error value. To have the command return the stored todo data, specify the fetch parameter (fetch=1). In addition, use the tzidout parameter to specify the time zone the returned data should be translated to. If the tzidout parameter is missing, the data will be returned in Zulu time.

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).

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=20020101T103000&dtend=20020101T113000&uid=001
&summary=new%20year%20event
```

The above example results in the following entry in an iCalendar database:

```
BEGIN:VEVENT
DTSTART:20020101T183000Z
DTEND:20020101T193000Z
UID:001
SUMMARY:new year event
END:VEVENT
```

storetodos

Purpose

Add one or more todos to a calendar.

Parameters

Table 6-40 lists `storetodos` parameters:

Table 6-40 `storetodos` Parameters

Parameter	Type	Purpose	Required	Default
`alarmAudio`	ISO 8601 Date Time string	The time at which to sound an audio alarm.	N	N/A
`alarmDescription`	string	The message send out with the alarm	N	“This is the alarm description”
`alarmEmails`	semicolon-separated list of email addresses	Recipients of alarm notifications for the todo.	N	N/A
`alarmFlashing`	ISO 8601 Date Time string	The time at which to run a flashing alarm.	N	N/A
`alarmPopup`	ISO 8601 Date Time string ISO 8601 Duration string	The time at which to pop up a dialog alarm.	N	N/A

Table 6-40 `storetodos` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`alarmStart`	ISO 8601 Date Time string ISO 8601 Duration string	The time at which to send an alarm notification of the todo.	N	N/A
`appid`	string	A runtime parameter (not stored in the database) that specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the *SunONE Messaging and Collaboration Event Notification Service Manual*.	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	A todo's iCalendar RFC 2445 attendee properties. For a list of the properties understood by Calendar Server, see Table 6-7	N	N/A
`calid`	string	Calendar identifier (or email address of `calid`) in which to store the todo.	Y	N/A
`categories`	semicolon-separated list of strings	The todo categories.	N	N/A
`charset`	string	The character set for the calendar.	N	N/A
`completed`	ISO 8601 Date Time string	Completion date of the todo. A value of `0` means the todo is not yet completed.	N	`0`

`compressed`	integer (`0`,`1`)	This parameter has been deprecated in this release and may be removed in future releases.
		For `compressed=0`, returns less data. Specifically, it does not return the following parameters:
		`rrules`
		`rdate`
		`exrule`
		`exdate`
		For `compressed=1`, all recurrence data is returned.
			N	`0`

`contacts`	semicolon-separated list of strings	Contacts for the todo.	N	N/A

Table 6-40 `storetodos` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`desc`	string	Purpose of the todo. A string of any length. If not passed, `desc` is set to the `summary` value. To include spaces in the string, use the code `%20`.	N	Value in `summary` parameter.
`dtstart`	ISO 8601 Date Time string	Start time and date of the todo. Not required to modify todos. Required to create todos.	 N Y	N/A
`due`	ISO 8601 Date Time string	End time and date of the todo.	N	N/A
`duration`	ISO 8601 duration string	Todo duration.	N	N/A
`exdates`	semicolon-separated list of ISO 8601 TimeDate strings	Exclusionary recurrence dates of the todo. To successfully create todos, the `rrules` parameter must also be specified.	N	N/A
`exrules`	semicolon-separated list strings	Todo exclusionary recurrence rules. A semicolon-separated list of recurrence-rule strings. Each rule value must be enclosed in quotes. See "Recurring Components – Overview."	N	N/A
`fetch`	integer (`0`,`1`)	A boolean indicating whether or not to fetch and return newly stored todos. `1` = Fetch and return newly stored todos. `0` = Do not fetch todos.	N	`0`

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

`geo`	two semicolon-separated floats	Semicolon-separated string of two float numbers representing the todo's geographical location (latitude and longitude). For example, 37.31;-123.2.	N	`0;0`

`icsClass`	string	Todo class. One of the following values:
		`PUBLIC`
		`PRIVATE`
		`CONFIDENTIAL`
			N	`PUBLIC`

`icsUrl`	string	Todo URL.	N	""

Table 6-40 `storetodos` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`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. `1` = An all day todo. `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.
		One of the following:
		`1` = `PUBLISH`
		`2` = `REQUEST`
		`4` = `REPLY`
		`8` = `CANCEL`
		`16` = `MOVE`
		`32` = `COUNTER` (only attendees use this)
			Y	`1` (`PUBLISH`)

`mod`	integer	Specifies the recurrences to store/modify.
		Not required for new todos.
		Required to modify todos.
		One of the following values:
		`1 = THISINSTANCE`
		`2 = THISANDFUTURE`
		`3 = THISANDPRIOR`
		`4 = THISANDALL`
			
			N
			Y
				N/A

`notify`	integer 0,1	This parameter has been deprecated. It remains to provide backward compatibility. The Group Scheduling Engine (GSE) handles sending of email notifications. A boolean indicating whether or not to notify attendees of a changed todo. `1` = Notify attendees of the change. `0` = Do not notify attendees.	N	`0`
`orgCalid`	string	Calendar identifier of organizer. One of the following parameters must be specified: `orgCalid`, `orgEmail`, or `orgUID`.	N	N/A
`orgCN`	string	Common name of the organizer.	N	N/A

Table 6-40 `storetodos` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`orgEmail`	email address	The email address contact for the todo. (Usually the organizer's email.) One of the following parameters must be specified: `orgCalid`, `orgEmail`, or `orgUID`.	N	N/A
`orgUID`	`userid`	The `userid` of the organizer. One of the following parameters must be specified: `orgCalid`, `orgEmail`, or `orgUID`.	N	N/A
`percent`	integer (`0`-`100`)	Percentage completion of the todo.	N	`0`
`priority`	integer (`0`-`9`)	The priority of the todo. `0` = Lowest priority. `9`= Highest priority.	N	`0`
`rchange`	integer (0,1)	A boolean indicating whether or not to replace the rrule: `1` = Replace the rule. `0` = Do not replace it.	N	`0`
`rdates`	semicolon-separated list of ISO 8601 TimeDate Z strings	Recurrence dates of the todo. To successfully create todos, the `rrules` parameter must also be specified.	N	N/A
`relatedTos`	semicolon-separated list of quoted strings	Other todos to which this todo is related.	N	N/A
`replace`	Integer (0,1)	A boolean. For parameters with semicolon-separated values: `1` = update (replace the old values with the new passed-in values) `0` = append (add the new passed-in values to the old ones)	N	`0`
`resources`	semicolon-separated list of strings	The resources associated with the todo.	N	N/A
`rid`	ISO 8601 TimeDate string	Recurrence identifier of the todo. Not required for new todos. If this parameter is not specified the whole series is modified.	N	N/A
`rrules`	semicolon-separated list of strings	Todo recurrence rules. A semicolon-separated list of recurrence-rule strings. Each rule value must be enclosed in quotes. See "Recurring Components – Overview."	N	N/A
`seq`	integer	Sequence number of the todo.	N	`0`

Table 6-40 `storetodos` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default

`status`	integer	A code for the status of the todo. One of the following values:
		`0 CONFIRMED`
		`1 CANCELLED`
		`2 TENTATIVE`
		`3 NEEDS_ACTION`
		`4 COMPLETED`
		`5 IN_PROCESS`
		`6 DRAFT`
		`7 FINAL`
		Note: Setting `status=4` sets the `status` as `COMPLETED`, but does not set the `COMPLETED_TIME_STAMP` and `PERCENT` properties. All three parameters must be set if a task is completed.
			N	N/A

`storetype`	integer	Designates whether an explicit “create” or “modify” is attempted on a todo. An error results if an attempt is made to create a todo that already exists, or to modify a todo that does not exist. The error returned is `STORETODOS_FAILED (15)` The following values are valid: 0 WCAP_STORE_TYPE_NONE 1 WCAP_STORE_TYPE_CREATE 2 WCAP_STORE_TYPPE_MODIFY If the attribute is not passed or has a value of `0`, no error conditions are reported.	N	0
`summary`	string	Todo summary. A string of any length. Required for new todos; not required for modifying todos. To include spaces in the string, use the code `%20`.	Y/N	Default `summary` available for new todos
`transparent`	integer (`0`, `1`)	Is the private todo transparent (`1`), or opaque (`0`)? If it is transparent, exclude this private todo from freebusy calculations. If opaque, then include it in freebusy calculations.	N	0 (opaque)
`tzid`	time zone ID string,	The timezone used to convert time parameter values to Zulu time for storage. If this parameter is not present, and the string does not end in Z, then the calendar server tim zone ID is used.	N	Calendar server time zone ID

Table 6-40 `storetodos` Parameters *(Continued) (Continued)*

Parameter	Type	Purpose	Required	Default
`tzidout`	time zone ID string	Time zone returned data should be translated to. Only valid when the `fetch` parameter is set to `1` (`fetch=1`).	N	Returns data in Zulu time
`uid`	string	Unique identifier of the todo to be stored. System generated for new todos; required to modify todos.	N/Y	Default `uid` for new todos

Description.

Use this command to create and modify 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 "Recurring Components – Overview."

When the `notify value` is `1`, it sends an IMIP `PUBLISH` message to the email attendees of the todo.

For group scheduling, used the attendee and method parameters as explained in the Common Topics section "Group Scheduling" on page 88.

For an explanation of how to replace, append or delete a parameter, see the explanation in the Common Topics section "Updating Parameter Values" on page 100.

The server does not support attachments. The `attachments` parameter exists to support iCalendar interoperability only, and is not functional.

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 the dtstart parameter.

  Every other parameter is optional. The server generates the `uid`.

- To modify existing todos requires two parameters:
  - `uid`
  - `mod`

All other parameters are optional. If a parameter is not specified, the todo will retain the previous value of the property.

Duration and Due

The due date (`due`) overrides `duration`. If you specify both `duration` and `due`, the command ignores `duration`.

Specify the duration in the ISO 8601 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).

Returns

The command returns the error value. To have the command return the stored todo data, specify the `fetch` parameter (`fetch=1`). In addition, use the `tzidout` parameter to specify the time zone the returned data should be translated to. If the `tzidout` parameter is missing, the data will be returned in Zulu time.

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).

verifyevents_by_ids

Purpose

This command is used to verify if the event specified with the uid and rid pair exists in the database.

Parameters

Table 6-43 lists verifyevents_by_ids parameters:

Table 6-41 verifyevents_by_ids Parameters

Parameter	Type	Purpose	Required	Default
calid	string	Calendar from which to fetch events.	N	User's default calendar
fmt-out	string	The format for the returned data. The three format types: text/calendar text/xml	N	text/calendar
id	string	Uniquely identifies the session (session ID).	N*	null
rid	ISO 8601 DateTime string	Corresponding set of comma separated event rids. If this is a non-recurring event, rid=0.	Y	N/A
tzid	time zone ID string	Time zone, such as "America/Los_Angeles"	N	Server's default time zone
uid	string	Comma separated set of event uids.	Y	N/A

* The id must be specified with the command unless the calendar to fetch from is public calendar.

Description.

This command tries to fetch events matching the passed in uid-rid pairs.

Returns

If the events are found, the data is returned in the format specified by the fmt-out parameter. If no format was specified, the output defaults to text/calendar.

If the events are not found, if the rids were not zero (0), then the uids and rids are returned.

If the events are not found and the rids were zero (0), then only the uids are returned.

Example

Example 1 is in text/calendar format, and example 2 is in text/xml output.

Example 1

```
verifyevents_by_ids.wcap?id=$n3o2m05sx9v6t98t8u2p&calid=jdoe&uid=3bd9e72f0
00027cf0000000600002113;3bd9e71700004503000000100002113;3bd9e717000045030
000000100002113;3bd9cd4700002206000000010000202d&rid=0;20021027T230000Z;20
021026T230000Z;0&fmt-out=text/calendar
```

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
VERSION:6.0
BEGIN:VEVENT
UID:3bd9e71700004503000000100002113
RECURRENCE-ID:20021027T230000Z
END:VEVENT

BEGIN:VEVENT
UID:3bd9cd4700002206000000010000202d
END:VEVENT
END:VCALENDAR
```

Example 2

```
verifyevents_by_ids.wcap?id=$n3o2m05sx9v6t98t8u2p&calid=savri&uid=3bd9e72f
000027cf0000000600002113;3bd9e71700004503000000100002113;3bd9e71700004503
0000000100002113;3bd9cd4700002206000000010000202d&rid=0;20021027T230000Z;2
0021026T230000Z;0&fmt-out=text/xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<iCalendar>
<iCal version="6.0" prodid="-//SunONE/Calendar Hosting Server//EN">
<EVENT>
<UID>3bd9e71700004503000000100002113</UID>
<RECURID>20021027T230000Z</RECURID>
</EVENT>
<EVENT>
<UID>3bd9cd4700002206000000010000202d</UID>
</EVENT>
</iCal>
</iCalendar>
```

verifytodos_by_ids

Purpose

This command is used to verify if the todo specified with the uid and rid pair exists in the database.

Parameters

Table 6-42 lists verifytodos_by_ids parameters:

Table 6-42 verifytodos_by_ids Parameters

Parameter	Type	Purpose	Required	Default
calid	string	Calendar from which to fetch todos.	N	User's default calendar
fmt-out	string	The format for the returned data. The three format types: text/calendar text/xml	N	text/calendar
id	string	Uniquely identifies the session (session ID).	N*	null
rid	ISO 8601 DateTime string	Corresponding set of comma separated todo rids. If this is a non-recurring todo, rid=0.	Y	N/A
tzid	time zone ID string	Time zone, such as "America/Los_Angeles"	N	Server's default time zone
uid	string	Comma separated set of todo uids.	Y	N/A

* The id must be specified with the command unless the calendar to fetch from is public calendar.

Description.

This command tries to fetch todos matching the passed in uid-rid pairs.

Returns

If the todos are found, the data is returned in the format specified by the fmt-out parameter. If no format was specified, the output defaults to text/calendar.

If the todos are not found, if the rids were not zero (0), then the uids and rids are returned.

If the todos are not found and the rids were zero (0), then only the uids are returned.

Example

```
verifytodos_by_ids.wcap?id=bo35r2pr3e5po35r&calid=jdoe&uid=3bde188f0000472
d0000000b00000399&rid=20021029T200200Z&fmt-out=text/xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<iCalendar>
<iCal version="6.0" prodid="-//SunONE/Calendar Hosting Server//EN">
<TODO>
<UID>3bde188f0000472d0000000b00000399</UID>
<RECURID>20021029T200200Z</RECURID>
</TODO>
</iCal>
</iCalendar>
```

version

Purpose

To get the current WCAP version.

Parameters

Table 6-43 lists the `version` parameter:

Table 6-43 `version` Parameters

Parameter	Type	Purpose	Required	Default

`fmt-out`	string	The format for the returned data.
		The three format types:
		`text/calendar`
		`text/xml`
			N	`text/calendar`

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 iCal and XML, the variable *X-NSCP-WCAPVERSION* contains the WCAP version number.

Example

The following examples are for each of output data types.

- iCalendar: (URL: `/version.wcap?fmt-out=text/calendar`)

```
BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-NSCP-WCAPVERSION:2.0.0
END:VCALENDAR
```

- XML: (URL: /version.wcap?fmt-out=text/xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<iCalendar>
<iCal version="2.0" prodid="-//iPlanet/Calendar Hosting Server//EN">
<X-NSCP-WCAPVERSION>2.0.0</X-NSCP-WCAPVERSION>
</iCal>
</iCalendar>
```

Glossary

ACE (access control entry) A string that provides access control for calendars, calendar properties, and calendar components such as events and todos (tasks). An example of an ACE is `jsmith^c^wd^g`.

ACL (access control list) A set of access control entry (ACE) strings that collectively provide access control for calendars, calendar properties, and calendar components such as events and todos (tasks). An example of an ACL with three ACEs, with each ACE separated by a semi-colon is `@@o^a^r^g;@@o^c^wdeic^g;@^a^sf^g`.

alarm event An event generated and sent by the Sun™ ONE Calendar Server Event Notification Service (ENS). When an alarm event occurs, a message reminder is sent to specific recipients.

authentication The verification of a user ordinarily done using a user ID and a corresponding password. Knowledge of the password is assumed to guarantee that the user is authentic. The Calendar Server requires a directory service such as an LDAP server for user authentication.

base DN The distinguished name (DN) that identifies the starting point of a search in an LDAP directory. Also known as a search base. For example, `ou=people,o=sesta.com`.

Berkeley DB A transactional database intended for high-concurrency read-write workloads and for applications that require transactions and recoverability. The Calendar Server uses the Berkeley DB from Sleepycat Software Inc. for storing calendar data.

Calendar Express A Web-based calendar client program that provides access to the Calendar Server for end users.

calendar group A collection of several calendars that can help a user manage more than one calendar.

calendar ID (`calid`**)** A unique identifier associated with a calendar in the Calendar Server database. The format for a calendar ID is *`userid`*`[:`*`calendar`*`]` where *`userid`* is the user ID and *`calendar`* is the calendar name.

Calendar Lookup Database (CLD) (CLD) A mechanism (distributed as a plug-in) that determines the physical location of a calendar when the calendar database is distributed over several servers. The Database Wire Protocol (DWP) uses the CLD mechanism to fully qualify a calendar ID (`calid`). DWP can use the returned URL to determine the location of the calendar, along with its access protocol.

Calendar Server Application Programming Interface CSAPI A programmatic interface that provides the capability to modify or enhance the feature set of the Calendar Server. CSAPI modules are plug-ins that are loaded from the `cal/bin/plugins` directory when the Calendar Server is started.

Calendar Access Protocol (CAP) A standard Internet protocol for calendaring based on requirements identified by the Internet Engineering Task Force (IETF).

cal_svr_base A directory location relative to other files on a server. For example, the default Calendar Server installation on Solaris systems uses the path `/opt/SUNWics5/` as the server root.

common name (cn) An attribute that identifies the person or object defined by the entry in an LDAP directory.

component state A set of attributes that describe a calendar event such as a meeting. In WCAP, the `compstate` parameter allows fetch commands to return events by component state. For example, `compstate` might be `REPLY-DECLINED` (attendee has declined a meeting) or `REQUEST_NEEDS-ACTION` (attendee has not taken action on a meeting yet).

Calendar User Agent (CUA) An application that a calendar client uses to access the Calendar Server.

default calendar The calendar a user first sees after logging into Calendar Express. Usually, the calendar ID of a default calendar is the same as the user's user ID. For example, `wchang@sesta.com` would have a default calendar named `wchang`.

directory service A centralized repository of directory information for use by other servers. The Calendar Server requires that a calendar user be stored in a directory server such as an LDAP server. The Calendar Server then uses the directory server for user authentication and for the storage and retrieval of user preferences. See also LDAP (Lightweight Directory Access Protocol).

distinguished name (DN) A string representation that uniquely identifies a user, system, or organization. A DN identifies an entry in an LDAP directory from which searches will occur. Also known as a search base. For example, `ou=people,o=sesta.com`.

Database Wire Protocol (DWP) A Calendar Server proprietary protocol that allows multiple servers to be linked together within the same Calendar Server system to form a distributed calendar store. The Calendar Servers uses DWP to retrieve remote data stored in the calendar database.

event A entry with an associated date and time in a calendar. For example, an event might be a new meeting or appointment on a calendar.

Event Notification Service (ENS) A generic service that accepts reports of server-level events that can be categorized and then notifies other servers that have registered interest in certain categories of events.

Extensible Markup Language (XML) A flexible programming language developed by the World Wide Web Consortium (W3C) to create common information formats and share both the format and the data on the Web, intranets, and elsewhere. XML is extensible because, unlike HTML, the markup symbols are unlimited and self-defining. The Calendar Server uses XML and XSL to generate the Calendar Express user interface.

Extensible Style Language (XSL) A language used to create style sheets for XML. XSL describes how data sent over the Web using the XML is to be presented to the user. The Calendar Server uses XSL and XML to generate the Calendar Express user interface.

Group ID (GID) On UNIX systems, the group for Calendar Server files such as counters and logs. The GID is stored in the `ics.conf` file in the `local.servergid` parameter.

GMT (Greenwich Mean Time) The mean solar time of the meridian of Greenwich, England, and the time standard against which all other time zones in the world are referred. GMT is not affected by Daylight Savings Time or Summer Time.

Group Scheduling Engine (GSE) The Calendar Server process that handles group scheduling. The GSE enables a user to schedule events with other calendar users on the same server or on a different server. The other user can then modify, cancel, or reply to the event.

High Availability (HA) A configuration that enables two Solaris servers to run a single instance of Calendar Server that remains continuously available after any single point of failure in hardware (disk, server, or network) or software has occurred in either of the servers.

horizontal scalability The Calendar Server's capability to run on a single server or as a group of processes that are spread across multiple server with a wide variety of possible configuration options.

Hypertext Transfer Protocol (HTTP) A standard protocol that allows the transfer of hypertext documents over the Web. The Calendar Server uses HTTP as its primary transport.

instance A Calendar Server configuration of one or more server processes. Multiple Calendar Server instances can be configured per server.

ISO 8601 An ISO (International Organization for Standardization) standard that specifies the numeric representation of date and time. The Calendar Server uses ISO 8601 standard notations to represent date, time, and duration strings.

LDAP (Lightweight Directory Access Protocol) A directory service protocol defined by the Internet Engineering Task Force (IETF) used for the storage, retrieval, and distribution of information, including user profiles, distribution lists, and configuration data.

LDAP server A software server that maintains an LDAP directory and services queries to the directory. The Calendar Server uses the Sun ONE Directory Server, which is an implementation of an LDAP server.

notification A message describing an event occurrence. An example of a notification in Calendar Server is a reminder for an upcoming meeting.

notification service A service that receives subscriptions and notifications from other servers and then relays notifications to specific subscribers. The Calendar Server `csnotifyd` service sends notifications of events and todos (tasks) using the Event Notification Service (ENS) as the broker for the events.

permissions The settings that control the access to a calendar. For example, in Calendar Express, permissions include Availability, Invite, Read, Delete, and Modify. Calendar Server administrators set permissions as access control entry (ACE) strings using command-line utilities. See also ACE (access control entry) and ACL (access control list).

plug-in An accessory program that can be loaded and then used as part of the overall system. For example, the Calendar Server can use a plug-in to access a non-LDAP directory service.

resource calendar A calendar associated with a resource such as a meeting room or equipment such as a notebook computer or overhead projector.

RFC (Request For Comments) A series of numbered international documents (such as RFC 2445, RFC 2446, and RFC 2447) that set standards that are voluntarily followed by Internet software developers. RFC standards are written informally by experts based on their technical experience and not by formal committees.

server root A directory location relative to other files on a server. For example, the default Calendar Server installation on Solaris systems uses the `/opt/SUNWics5/`path as the server root. In syntax and examples, *`cal_svr_base`* is used to represent the server root for Calendar Server.

service A component of an overall system. The Calendar Server has the following services: Administration Service (csadmind). HTTP Service (cshttpd), Notification Service (csnotifyd), Event Notification Service (enpd), and Distributed Database Service (csdwpd).

SHTML (Server-side Include Hypertext Markup Language) An HTML file that includes embedded server-side includes (SSIs).

Single Sign-on (SSO) An authentication mechanism that enables a user to log in once and then access 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 login to each application separately.

task In Calendar Express on the client side, a component of a calendar that specifies something to be done. On the server side, a task is called a todo.

time zone A geographical region that uses the same time. There are 25 hourly time zones from -12 through +12 (GMT is 0). Each time zone is measured relative to GMT. Most time zones have localized designations in three-letter abbreviations. The Calendar Server also identifies time zones using a time zone ID (TZID) such as America/Los_Angeles or Asia/Calcutta.

todo On the server side, a a component of a calendar that specifies something to be done. In Calendar Express on the client side, a todo is called a task.

user ID (uid) A unique string identifying a user to a system. The Calendar Server identifies each user by a user ID.

Universal Principle Name (UPN) The value for a logged-in user that includes the login name combined with the domain to which the user belongs. For example, user `bill` in domain `sesta.com` has the UPN `bill@sesta.com`.

WCAP (Web Calendar Access Protocol) A high-level, command-based protocol used by clients to communicate with the Calendar Server.

Zulu time A military designation for GMT and UTC (Coordinated Universal Time).

Index

SYMBOLS

% encoded characters 67
% symbol for encoded characters 79

A

Access Control Entries (ACE) (WCAP) 72, 191
acl parameter (WCAP) 72, 191
active server test (WCAP) 184
adding
  events (WCAP) 193
adding todos (WCAP) 202
addlink command (WCAP) 105
administrator commands (WCAP)
  change_password 106
  createcalendar 108
  deletecalendar 110
  ping 184
APIs
  authSDK
    architecture 54
    CEXP_GenerateLoginURL 58
    CEXP_GetVersion 58
    CEXP_Init 59
    CEXP_SetHttpPort 60
    CEXP_Shutdown 60
    initialization 54
    introduction 53
  CSAPI
    csIAccessControl 23
    csIAuthentication 26
    csICalendarLookup 30
    csICalendarServer 47
    csIDataTranslator 36
    csIMalloc 49
    csIPlugin 39
    csIQualifiedCalidLookup 42
    csIUserAttributes 44
    interfaces 23
    introduction 15
WCAP
  addlink 105
  change_password 106
  check_id 107
  createcalendar 108
  deletecalendar 110
  deletecomponents_by_range 112
  deleteevents_by_id 113
  deleteevents_by_range 116
  deletetodos_by_id 117
  deletetodos_by_range 120
  export 121
  fetchcomponents_by_alarmrange 124
  fetchcomponents_by_attendee_error 132
  fetchcomponents_by_lastmod 135, 148
  fetchcomponents_by_range 138
  fetchevents_by_id 154
  fetchtodos_by_id 157
  get_all_timezones 163
  get_calprops 167
  get_freebusy 170, 174
  get_guids 173
  get_userprefs 176
  import 179

introduction 63
login 182
logout 184
ping 184
search_calprops 185
set_calprops 188
set_userprefs 192
storeevents 193
storetodos 202
version 213
architecture
authSDK 54
authentication
session identifiers (WCAP) 66
user (WCAP) 182
authSDK
architecture 54
cleanup 54
definition 53
functions
CEXP_GenerateLoginURL 54, 58
CEXP_GetVersion 54, 58
CEXP_Init 54, 59
CEXP_SetHttpPort 54, 60
CEXP_Shutdown 54, 60
initialization 54
integrating and using 60
introduction 53
lookup 54

C

calendar properties
retrieving (WCAP) 167
calendars
creating new (WCAP) 108
deleting (WCAP) 110
deleting components (WCAP) 112
deleting events (WCAP) 113, 116
deleting todos (WCAP) 117, 120
freebusy (WCAP) 86, 170, 174, 192
MIME types (CSAPI) 37
preferences (WCAP) 176
properties (WCAP) 188
restricting viewing of details (WCAP) 86, 192
scheduling (WCAP) 86, 192
Calloc method (CSAPI) 49
change_password command (WCAP) 106
ChangePassword method (CSAPI) 27
check_id command (WCAP) 107
CheckAccess method (CSAPI) 24
client APIs
list 19
client APIs (CSAPI)
csIAccessControl 23
csIAuthentication 26
csICalendarLookup 30
csIDataTranslator 36
csIPlugin 39
csIQualifiedCalidLookup 42
csIUserAttributes 44
client request formats (WCAP) 67
command formats (WCAP) 67
command overview (WCAP) 64
component state values table (WCAP) 84
components (WCAP)
importing 179
recurrence handling 92
retrieving 138
retrieving changes 135, 148
retrieving errors 132
createcalendar command (WCAP) 108
CSAPI
architecture 16
client APIs
csIAccessControl 23
csIAuthentication 26
csICalendarLookup 30
csIDataTranslator 36
csIPlugin 39
csIQualifiedCalidLookup 42
csIUserAttributes 44
list 19
csICalendarLookup methods 31
dependencies
NSPR 18
XPCOM 18
introduction 15
list of interfaces 23

method return codes 27
module structure 19
requirements
threadsafe plug-ins 17
server APIs
csICalendarServer 47
csIMalloc 49
list 19
csIAccessControl (CSAPI)
CheckAccess method 24
Init method 25
csIAuthentication (CSAPI)
ChangePassword method 27
Init method 28
Logon method 28
Logout method 29
VerifyUserExists method 30
csICalendarLookup (CSAPI)
FindCalid method 43
FreeCalid method 32
FreeType method 33
GetHostnameForCalid method 32, 33
Init method 34, 43
QualifyCalid method 34
QueryType method 35
SetHostnameForCalid method 36
csICalendarLookup methods 31
csICalendarServer (CSAPI)
GetVersion method 48
Init method 48
csIDataTranslator (CSAPI)
GetSupportedContentType method 37
Init method 38
Translate method 39
csIMalloc (CSAPI)
Calloc method 49
Free method 50
FreeIf method 50
Init method 51
Malloc method 51
csIPlugin (CSAPI)
GetDescription method 40
GetVendorName method) 41
GetVersion method 41
Init method 42
csIRealloc (CSAPI)
Calloc method 52
csIUserAttributes (CSAPI)
FreeAttribute method 44
GetAttribute method 45
Init method 46
SetAttribute method 47

D

default format, WCAP commands 68
deletecalendar command (WCAP) 110
deletecomponents_by_range command (WCAP) 112
deleteevents_by_id command (WCAP) 113
deleteevents_by_range command (WCAP) 116
deletetodos_by_id command (WCAP) 117
deletetodos_by_range command (WCAP) 120
deleting
components (WCAP) 112

E

encoded characters example (WCAP) 79
errors
return codes (WCAP) 79
event notification
client side (WCAP) 68
events
adding (WCAP) 193
alarm triggers (WCAP) 124
deleting (WCAP) 113, 116
exporting (WCAP) 121
importing (WCAP) 179
recurrence handling (WCAP) 92
retrieving (WCAP) 138, 154
retrieving changes (WCAP) 135, 148
retrieving errors (WCAP) 132
export command (WCAP) 121

F

fetchcomponents_by_alarmrange command (WCAP) 124
fetchcomponents_by_attendee_error command (WCAP) 132
fetchcomponents_by_lastmod command (WCAP) 135, 148
fetchcomponents_by_range command (WCAP) 138
fetchevents_by_id command (WCAP) 154
fetchtodos_by_id command (WCAP) 157
FindCalid method (CSAPI) 43
finding a calendar (WCAP) 185
formatting
  client requests (WCAP) 67
  output formats (WCAP) 92
  server request formats (WCAP) 68
Free method (CSAPI) 50
FreeAttribute method (CSAPI) 44
freebusy
  definition (WCAP) 86, 192
  retrieving (WCAP) 170, 174
FreeCalid method (CSAPI) 32
FreeIf method (CSAPI) 50
FreeType method (CSAPI) 33

G

get_all_timezones command (WCAP) 163
get_calprops command (WCAP) 167
get_freebusy command (WCAP) 170, 174
get_guids command (WCAP) 173
get_userprefs command (WCAP) 176
GetAttribute method (CSAPI) 45
GetDescription method (CSAPI) 40
GetHostnameForCalid method (CSAPI) 32, 33
GetSupportedContentType method (CSAPI) 37
GetVendorName method (CSAPI) 41
GetVersion method (CSAPI) 41, 48
globally unique identifiers (GUIDs) (WCAP) 173

I

import command (WCAP) 179
Init method (CSAPI) 25, 28, 34, 38, 42, 43, 46, 48, 51
interfaces
  csIAuthentication 26
  csIDataTranslator 36
  csiMalloc 49
  csIPlugin 40
  csIUserAttributes 44

L

local.servergid 217
login command (WCAP) 182
Logon method (CSAPI) 28
logout command (WCAP) 184
Logout method (CSAPI) 29

M

Malloc method (CSAPI) 51
MIME types (CSAPI) 37
modifying
  password (WCAP) 192
  preferences (WCAP) 192

N

new parameters
  compstate values 84

O

output formats (WCAP) 68, 92

P

passwords, modifying (WCAP) 106, 192
ping command (WCAP) 184
plug-in interfaces (CSAPI) 19
preferences
  modifying (WCAP) 192
  retrieving (WCAP) 176
properties
  retrieving calendar (WCAP) 167
  setting calendar (WCAP) 188
Proxy Authentication SDK, see authSDK 53

Q

QualifyCalid method (CSAPI) 34
QueryType method (CSAPI) 35

R

Realloc method (CSAPI) 52
recurrence handling (WCAP) 92
  delete options 97
  deleting recurring components 97
  exdates parameter 96
  exrules parameter 95
  mod parameter 96
  rchange parameter 97
  rdates parameter 95
  rid parameter 96
  rrules parameter 94
retrieving a calendar (WCAP) 185
return codes
  CSAPI methods 27
  error string (WCAP) 79

S

search_calprops command (WCAP) 185
server APIs (CSAPI)
  csICalendarServer 47
  csIMalloc 49
server interfaces (CSAPI) 21
session identifiers (WCAP) 66
sessions
  password modification (WCAP) 192
  preferences modification (WCAP) 192
sessions, validating (WCAP) 107
set_calprops command (WCAP) 188
set_userprefs command (WCAP) 192
SetAttribute method (CSAPI) 47
SetHostnameForCalid method (CSAPI) 36
storeevents command (WCAP) 193
storetodos command (WCAP) 202

T

terminate user session (WCAP) 184
timezones, retrieving (WCAP) 163
todos (WCAP)
  adding 202
  alarm triggers 124
  deleting 117, 120
  exporting 121
  importing 179
  recurrence handling 92
  retrieving 138, 157
  retrieving changes 135, 148
  retrieving errors 132
Translate method (CSAPI) 39

U

UI generator
  SHTML 63
  WCAP 63
URI / URL
  format (WCAP) 67
user access (WCAP) 72, 191

user interface
SHTML 63

V

VerifyUserExists method (CSAPI) 30
version command (WCAP) 213

W

WCAP
Access Control Entries (ACE) 72, 191
administrator commands
change_password 106
createcalendar 108
deletecalendar 110
ping 184
client request format 67
client side event notification 68
command
formats 67
list 70
overview 64
commands
addlink 105
change_password 106
check_id 107
createcalendar 108
deletecalendar 110
deletecomponents_by_range 112
deleteevents_by_id 113
deleteevents_by_range 116
deletetodos_by_id 117
deletetodos_by_range 120
export 121
fetchcomponents_by_alarmrange 124
fetchcomponents_by_attendee_error 132
fetchcomponents_by_lastmod 135, 148
fetchcomponents_by_range 138
fetchevents_by_id 154
fetchtodos_by_id 157
get_all_timezones 163
get_calprops 167
get_freebusy 170, 174
get_guids 173
get_userprefs 176
import 179
login 182
logout 184
ping 184
search_calprops 185
set_calprops 188
set_userprefs 192
storeevents 193
storetodos 202
version 213
encoded characters 67
error handling 79
freebusy access 86, 192
HTML form submission 68
introduction 63
new parameters
compstate values 84
output formats 68, 92
recurrence handling 92
return codes 79
session identifiers 66
UI generator 63
URI format 67