# Programmer's Manual

*iPlanet™ Calendar Server*

**Release 5.1**

January 2002

# Contents

# List of Tables

# About This Reference

This document describes the architecture of iPlanet™ Calendar Server 5.1 and gives detailed instructions on the use of the following two APIs and one protocol that you may use to customize your server installation:

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

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

Topics covered in this chapter include:

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

## Who Should Read This Book

This guide is for programmers who want to customize applications in order to implement iPlanet Calendar Server 5.1.

# 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 iPlanet Calendar Server, as well as containing an overall architecture discussion of the product. For each interface there is an overview chapter, followed by a reference chapter, where available.

A list of the chapters follows:

- About This Reference (this chapter)
- Chapter 1, "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 iPlanet Calendar Server.

- Chapter 4, “Proxy Authentication SDK Reference”

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

- Chapter 5, “Single Sign-on Authentication”

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

- Chapter 6, “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 7, “WCAP Commands”

- Glossary

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

# Document Conventions

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

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

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

# Where to Find Related Information

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

- *iPlanet Calendar Server Administrator’s Guide*

- *iPlanet Calendar Server Installation Guide*
- *iPlanet Messaging and Collaboration Event Notification Service Manual*
- *iPlanet Messaging and Collaboration Schema Reference Manual*

# Where to Find This Book Online

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

`http://docs.iplanet.com/docs/manuals/calendar/ics51/pr/contents.htm`

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

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

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

This chapter has the following sections:

- CSAPI Architecture
  - 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.

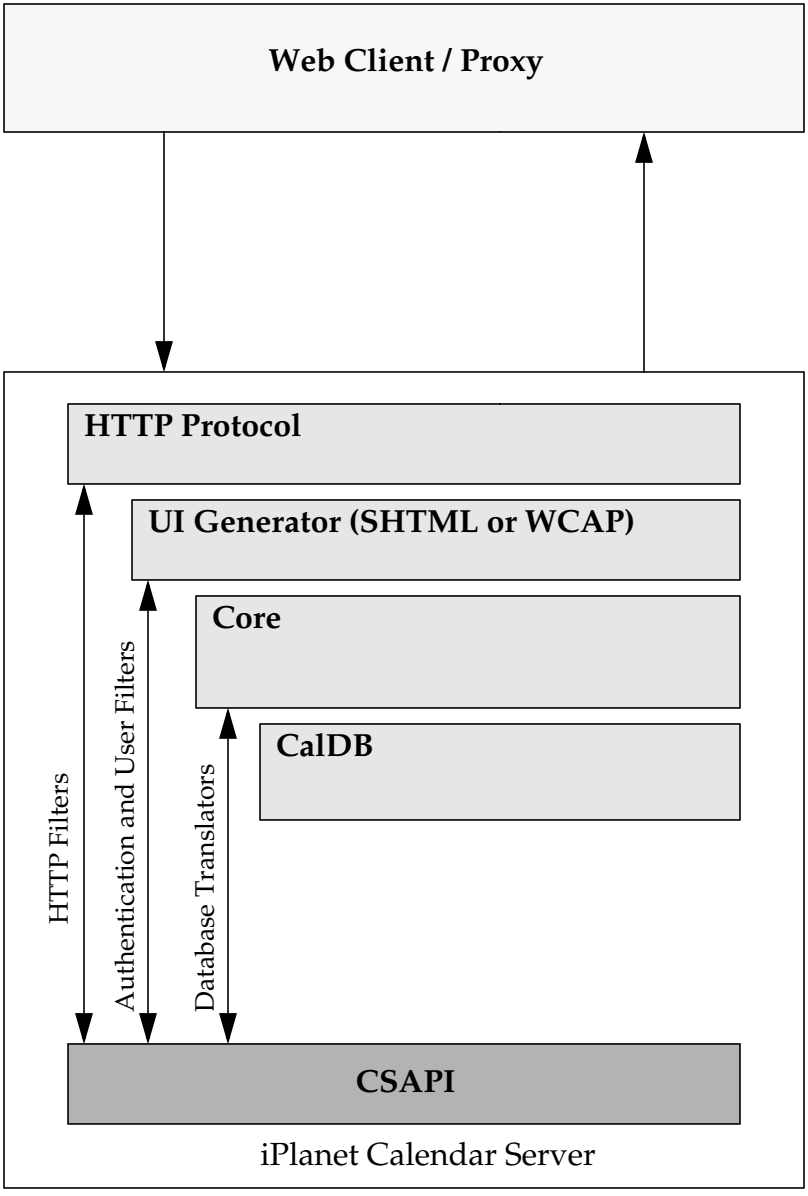
Figure 1-1, which follows, shows the relationship of CSAPI modules to other subsystems within iPlanet Calendar Server. Depending on which functional group or groups a CSAPI module supports, it can interact with one or more areas of iPlanet Calendar Server functionality, such as data formatting, authentication,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

A module is a shared object (`.so` file) on Unix, or dynamic linked library (`.dll` file) on Windows NT. Each module that you provide implements one or more of the CSAPI interfaces (or pure virtual base classes) defined in this document; see Chapter 2, “CSAPI Reference”. Each client-side interface addresses a functional area of iPlanet Calendar Server. The implementation contained in a module can either augment or override the native iPlanet Calendar Server functionality in its area.

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

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

**Figure 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/docs/refList/refNSPR`

For documentation on XPCOM, see:

http://www.mozilla.org/projects/xpcom

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

# Using CSAPI

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

## Loading CSAPI Modules

iPlanet Calendar Server loads CSAPI modules from the `cal/bin/plugins` directory at startup and unloads them at server shutdown. All plug-in modules must reside in this directory and have filenames that are prefaced with `cs_`.

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

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

# Plug-in Interfaces

All CSAPI plug-ins support one and only one exported symbol, *NSGetFactory*, as required by the XPCOM specification. From this entry point, Calendar Server calls the XPCOM method `QueryInterface` to find an object implementing the `csIPlugin` interface. This allows the server to query the plug-in for version, description, and vendor information. This interface is optional; however, it is highly recommended that you implement it so the server can ensure version control.

# Client and Server APIs

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

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

**Table 1-1** CSAPI Client APIs

| CSAPI Module Interface | Description |
|---|---|
| `csIAccessControl` | Augments or overrides the default access control mechanism. |
| `csIAuthentication` | Augments or overrides the login authentication mechanism. |
| `csICalendarLookup` | Augments or overrides the default calendar lookup mechanism. |
| `csIDataTranslator` | Augments or overrides the format translation of incoming and outgoing data. |
| `csIPlugin` | Provides version control and descriptive information about the module. |
| `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 1-2:

**Table 1-2** CSAPI Server APIs

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

## Server Query Example

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

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

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

# CSAPI Samples

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

The following sample modules are provided:

**Table 1-3** CSAPI Interface Samples

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

Chapter 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 iPlanet 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 iPlanet Calendar Server.

### Methods

The csIAccessControl interface implements two methods:

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

### Description

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

## CheckAccess

### Purpose

Sets users' calendar access.

### Syntax

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

### Parameters

The method has the following five parameters:

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

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

### Returns

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

### Description

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

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

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

**Table 2-3** ICS_ACCESSTYPE Constants

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

**Table 2-3** ICS_ACCESSTYPE Constants

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

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

## Init

### Purpose

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

### Syntax

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

### Parameters

The method has the following parameter:

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

### Returns

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

### Description

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

# csIAuthentication

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

### Methods

The csIAuthentication interface implements five methods:

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

### Description

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

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

## ChangePassword

### Purpose

Changes the password for the specified user.

### Syntax

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

### Parameters

The method has the following four parameters:

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

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

#### Returns

On success, `NS_AUTHENTICATION_CHANGEPASSWORD_SUCCESS`. On failure, `NS_AUTHENTICATION_CHANGEPASSWORD_FAILURE`.

#### Description

Changes the password of the specified user.

## Init

#### Purpose

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

#### Syntax

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

#### Parameters

The method has the following parameter:

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

#### Returns

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

#### Description

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

# Logon

### Purpose

Augment or override the authentication procedure for plain text login.

### Syntax

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

### Parameters

The method has the following three parameters:

| | |
|---|---|
| `aUser` | The user's login name. |
| `aPassword` | The plain text password. |
| `aReturnCode` | On return, contains a constant that determines whether the server should continue with the default authentication procedure.<br>One of the following constants:<br>• `NS_CONTINUE_DEFAULT_PROCESSING`<br>• `NS_OVERRIDE_DEFAULT_PROCESSING` |

### Returns

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

### Description

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

# Logout

### Purpose

Logout a user.

### Syntax

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

#### Parameters

The method has the following two parameters:

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

#### Returns

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

#### Description

## VerifyUserExists

#### Purpose

Verify that the user ID is in the LDAP directory.

#### Syntax

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

#### Parameters

The method has the following two parameters:

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

#### Returns

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

### Description

# csICalendarLookup

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

### Methods

The `csICalendarLookup` implements five methods:

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

### Description

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

## Init

### Purpose

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

### Syntax

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

### Parameters

The method has the following parameter:

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

### Returns

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

### Description

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

## QualifyCalid

### Purpose

Qualifies the relative calid.

### Syntax

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

### Parameters

The method has the following parameters:

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

### Returns

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

### Description

## FreeCalid

### Purpose

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

### Syntax

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

### Parameters

The method has the following parameters:

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

### Returns

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

### Description

## QueryType

### Purpose

Query type of database plug-in.

### Syntax

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

### Parameters

The method has the following parameters:

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

### Returns

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

### Description

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

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

## FreeType

### Purpose

Frees a previously allocated database plug-in type.

### Syntax

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

### Parameters

The method has the following parameters:

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

### Returns

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

### Description

Frees the string allocated in `QueryType` method.

# csIDataTranslator

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

### Methods

The `csIDataTranslator` interface implements three methods:

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

### Description

This interface allows you to manipulate or change the HTML Body content of calendar data flowing to, or from, the database, or between various data translators. The data translator manipulates the output format (`fmt-out`) component of a WCAP response.

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

**Table 2-4** MIME Types

| MIME Type | Description |
|---|---|
| `text/calendar` | iCalendar |
| `text/xml` | iCalendar in XML |
| `text/js` | Native JavaScript |

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

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

## GetSupportedContentTypes

### Purpose

Gets the content type this database translator supports.

### Syntax

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

### Parameters

The method has the following parameters:

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

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

#### Returns

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

#### Description

Get the content type this database translator supports.

## Init

#### Purpose

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

#### Syntax

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

#### Parameters

The method has the following parameter:

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

#### Returns

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

### Description

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

## Translate

### Purpose

Implement the translation from one content type to another.

### Syntax

```
PRUint32 Translate (char * aInContentType,
                    char * aOutContentType,
                    char ** aInBuffer,
                    char ** aOutBuffer,
                    PRInt32 * aInSize,
                    PRInt32 * aOutsize,
                    PRInt32 * aReturncode) = 0;
```

### Parameters

The method has the following parameters:

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

### Returns

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

### Description

This method retrieves content of the specified input type from the specified buffer, translates the content from its original format to the output format, stores the translated content in the specified output buffer, and stores the size of the output buffer at the specified location.

# csIPlugin

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

### Methods

The `csIPlugin` interface implements four methods:

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

### Description

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

## GetDescription

### Purpose

Retrieve a text description of the module.

### Syntax

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

### Parameters

The method has the following parameter:

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

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

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

## GetVendorName

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

### Syntax
```
PRInt32 GetVendorName (nsString& aVendorName) = 0;
```

### Parameters
The method has the following parameter:

| | |
|---|---|
| `aVendorName` | On return, contains the text description of the vendor. |

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

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

## GetVersion

### Purpose
Provide server with version information on startup.

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

### Parameters
The method has the following two parameters:

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

#### Returns

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

#### Description

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

### Init

#### Purpose

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

#### Syntax

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

#### Parameters

The method has the following parameter:

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

#### Returns

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

#### Description

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

## csIQualifiedCalidLookup

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

#### Methods

The csICalendarLookup implements two methods:

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

### Description

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

## FindCalid

### Purpose

Finds the calendar ID for the URL specified.

### Syntax

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

### Parameters

The method has the following parameters:

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

### Returns

The `calid` for the qualified URL passed in.

### Description

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

## Init

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

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

#### Parameters
The method has the following parameter:

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

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

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

# csIUserAttributes

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

#### Methods
The `csIUserAttributes` interface implements four methods:

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

### Description

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

## FreeAttribute

### Purpose

Free the memory associated with your local attribute storage.

### Syntax

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

### Parameters

The method has the following two parameters:

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

### Returns

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

### Description

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

## GetAttribute

### Purpose

Retrieve an attribute value for a user.

### Syntax

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

### Parameters

The method has the following four parameters:

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

### Returns

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

### Description

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

## Init

### Purpose

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

### Syntax

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

### Parameters

The method has the following parameter:

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

### Returns

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

### Description

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

## SetAttribute

### Purpose

Set an attribute value for a user.

### Syntax

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

### Parameters

The method has the following parameters:

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

#### Returns

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

#### Description

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

# csICalendarServer

Provides server version information to a plug-in module.

#### Methods

The csICalendarServer interface implements two methods:

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

#### Description

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

## GetVersion

#### Purpose

Provide plug-in module with server version information.

#### Syntax

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

#### Parameters

The method has the following two parameters:

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

#### Returns

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

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

### Init

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

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

#### Parameters
The method has no parameters.

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

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

## csIMalloc

Allocates and frees memory.

#### Methods
The `csIMalloc` interface implements six methods:

| | |
|---|---|
| `Calloc` | Allocates and initializes memory for a number of objects. |
| `Free` | Frees memory that is no longer in use. |
| `FreeIf` | Frees memory, allowing a `NULL` pointer. |
| `Init` | Confirms that the interface was found and registered. |
| `Malloc` | Allocates an amount of memory. |
| `Realloc` | Reallocates previously allocated memory. |

### Description

Plug-in modules can use this object to take advantage of the server's efficient memory allocation technique.The object is valid for the full lifetime of the client, so `Init` does not return a reference.

## Calloc

### Purpose

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

### Syntax

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

### Parameters

The method has the following two parameters:

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

### Returns

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

### Description

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

## Free

### Purpose

Free memory previously allocated by the `Malloc` method.

### Syntax

```
PRUint32 Free (void * aPtr) = 0;
```

### Parameters

The method has the following parameter:

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

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

### Description
Use this method in the same way as its C/C++ counterpart to free previously allocated memory.

## FreeIf

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

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

### Parameters
The method has the following parameter:

| | |
|---|---|
| aPtr | A pointer to the memory to be freed or NULL. |

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

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

## Init

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

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

### Parameters
The method has no parameters.

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

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

## Malloc

### Purpose
Allocate a specified amount of memory.

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

### Parameters
The method has the following parameter:

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

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

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

## Realloc

### Purpose
Reallocates memory that was previously allocated.

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

### Parameters
The method has the following parameter:

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

### Returns

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

### Description

Use this method in the same way as its C/C++ counterpart to reallocate memory that was previously allocated.

Chapter 3

# Proxy Authentication SDK Overview

This chapter describes the iPlanet 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 iPlanet Calendar Server. When a portal system authenticates a user, authSDK functions notify iPlanet Calendar Server, which then allows the user access to various services without reauthentication.

## What Is the authSDK?

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

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

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

# Architecture

The authSDK is pretty simple. It consists of initialization, lookup, and cleanup. Additionally, one other function,`CEXP_SetHttpPort`, allows the authSDK to use a non-standard port, and another,`CEXP_GetVersion`, gets the authSDK version number should you need to contact customer or technical support. For a complete description of the API functions, see Chapter 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 admin's ID and password.

## Cleanup

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

# Functions Overview

There are five functions in the SDK, as listed in Table 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 iPlanet 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 List

There are five authSDK functions:

- `CEXP_GenerateLoginURL`

  Generates a URL with the valid session ID.

- `CEXP_GetVersion`

  Generates the version ID string.

- `CEXP_Init`

  Initializes the SDK.

- `CEXP_SetHttpPort`

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

- `CEXP_Shutdown`

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

# Proxy Authentication SDK Functions

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

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

## CEXP_GenerateLoginURL

### Purpose

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

### Syntax

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

### Parameters

There are four parameters:

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

### Returns

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

## CEXP_GetVersion

### Purpose
Gets the version ID string.

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

### Parameters
There are no parameters:

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

## CEXP_Init

### Purpose
Initializes the SDK.

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

### Parameters
There are eight parameters:

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

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

# Single Sign-on Authentication

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

This chapter has the following topics:

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

## What is Single Sign-on?

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

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

# Limitations of Single Sign-on

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

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

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

# Process Flow

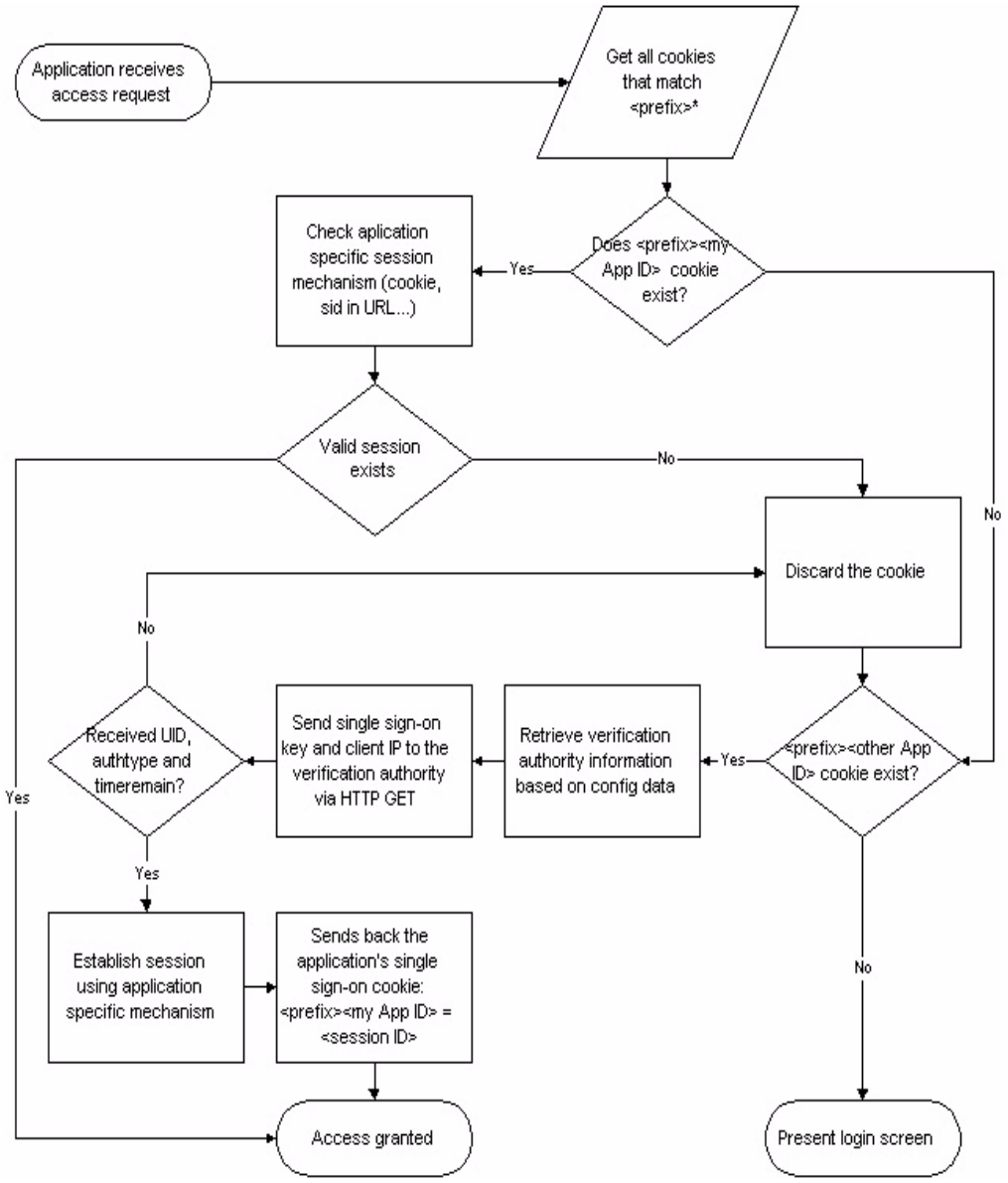
As diagrammed in the figure that follows, the flow starts with an application receiving access requests. Since a circle of trusted applications share the same prefix, the program fetches all cookies with the matching prefix.

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

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

Figure 5-1 illustrates Single Sign-on process flow.

**Figure 5-1** Single Sign-on Process Flow



# Implementation Requirements

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

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

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

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

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

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

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

## Cookie Information

The cookie format is as follows:

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

### Other recommended settings:

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

## Trusted Applications Record

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

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

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

## Single Sign-off Parameter

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

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

## Prefix String

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

# Single Sign-on Example

This example illustrates the steps involved in completing a Single Sign-on cycle.

# The Example

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

1. John checks his mail.

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

   The cookie returned is: `ssogrp13fr7d=l3khj5l3k9ldh`.

2. John then checks his calendar.

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

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

   - WebCal issues HTTP `GET`:

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

     `Cookie: ssogrp13fr7d=l3khj5l3k9ldh`

   - WebMail verifies this is a valid key and returns:

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

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

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

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

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

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

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

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

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

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

   - HRapp does not check for the `ssogrp1*` cookies.

   - HRapp generates a session `B53P997KD` for John and sends back a cookie: `ssogrp1adf38=lka79jy5d3l3r`.

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

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

   - WebMail receives the logoff request from `jsmith` running on `208.12.60.3`.

   - In addition to invalidating John's WebMail session, it also removes all `ssogrp1*` cookies.

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

# Configuration Parameters for the Example

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

  - WebMail:

    ```
    appid=3fr7d
    3fr7d.ip=198.93.96.111
    3fr7d.verificationurl=http://webmail host/VerifySSO?
    ```

- WebCal:

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

- HRapp:

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

# Issues

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

- Security
- Management
- Scalability
- Performance

## Security

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

## Management

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

## Scalability

Cookie number limitations:

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

## Performance

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

  This should not happen very often for two reasons:

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

Chapter 6

# 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 iPlanet Calendar Server. This chapter has the following sections:

- Introduction
- Command Overview
- Command Formats

## Introduction

The default client UI protocol for iPlanet Calendar Server 5.x is an SHTML protocol, and is private. However, you can use WCAP for compatibility to support the iPlanet Calendar Server 2.x client UI protocol. WCAP is also the only way to retrieve calendar data in `text/calendar` and `text/xml` formats.

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, iPlanet 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 (iCalendar).
- `text/xml` format.

  This XML format follows the iCalendar DTD.

- `text/js` with embedded JavaScript objects.

  This was the default for the iPlanet Calendar Server 2.x user interface. It has been deprecated for iPlanet Calendar Server 5.x in favor of the .shtml approach.

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 6-1 describes the high level list of commands supported in WCAP . For a detailed description of each command, see Chapter 7, "WCAP Commands".

**Table 6-1** WCAP Command Overview

| WCAP Command | Description |
|---|---|
| `addlink` | Add event links from one calendar to another. |
| `change_password` | Change the user's password. |
| `createcalendar` | Create a new calendar. |
| `deletecalendar` | Delete an existing calendar. |
| `deletecomponents_by_range` | Delete both events and todos in a calendar(s) over a specific time period. |
| `deleteevents_by_id` | Delete events given a specific calid and uid/recurrence-ID pair. |
| `deleteevents_by_range` | Delete events in a calendar(s) over a specific time period. |
| `deletetodos_by_id` | Delete todos given a specified calid and userid/recurrence-ID pair. |
| `deletetodos_by_range` | Deletes todos in a calendar(s) over a specific time period. |
| `export` | Exports a calendar to a file. |
| fetchcomponents_by_alarmrange | Queries for components that have alarms to trigger over a specific time period. |

**Table 6-1** WCAP Command Overview *(Continued)*

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

## Session Identifiers

For many WCAP commands, you must specify the session identifier (`id`) that is returned by the `login` command. The session identifier ensures that data is accessible only to authenticated users with the required level of privilege or ownership.

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

You can customize the authentication mechanism to use a local or external authentication scheme. See Chapter 2, “CSAPI Reference”, for the section “csIAccessControl,” on page 24.

# Command Formats

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

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

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

## Client Request Formats

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

| Command Format | Description |
|---|---|
| URI | Requests from client submitted using standard URI syntax. |
| HTML Form - `urlencoded` | Requests from client submitted using native JavaScript objects. |

| | |
|---|---|
| HTML Form - `text/xml` | Requests from client submitted using JavaScript objects formatted as XML. |
| HTML Form - `text/calendar` | Requests from client submitted using JavaScript objects formatted as iCalendar. |

## URI Format

Use the following format to submit a URI request:

`http://`*`webcalenderserver`*`/COMMAND?PARAM=VAL&PARAM=VAL...`

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

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

**NOTE** This is a change from WCAP for iPlanet Calendar Server 2.x. Previously, the quoted-printable equivalent for a semicolon was `=3B`.

See also “Common topics,” on page 79, in Chapter 7, “WCAP Commands”.

## HTML Form

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

## Client Side Event Notification

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

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

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

# Server Response Formats

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

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

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

Chapter 7

# WCAP Commands

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

## Common topics

# Commands

Table 7-1 lists the WCAP commands in alphabetical order.

**Table 7-1** WCAP Commands

| | |
|---|---|
| `addlink` | Add event links from one calendar to another. |
| `change_password` | Change the user's password. |
| `check_id` | Check to see if the session is still valid. For Admin's use only. |
| `createcalendar` | Create a new calendar. |
| `deletecalendar` | Delete an existing calendar. |
| `deletecomponents_by_range` | Delete both events and todos in a calendar(s) over a specific time period. |
| `deleteevents_by_id` | Delete events given a specific calid and uid/recurrence-ID pair. |
| `deleteevents_by_range` | Delete events in a calendar(s) over a specific time period. |
| `deletetodos_by_id` | Delete todos given a specified calid and userid/recurrence-ID pair. |
| `deletetodos_by_range` | Deletes todos in a calendar(s) over a specific time period. |
| `export` | Exports a calendar to a file. |
| `fetchcomponents_by_alarmrange` | Queries for components over a specific time period that have alarms to trigger. |
| `fetchcomponents_by_attendee_error` | Queries for components that had errors while sending group scheduling messages. |
| `fetchcomponents_by_lastmod` | Queries for components that have changed, during the specified time range. |
| `fetchcomponents_by_range` | Queries for components over a specific time period, with filtering attributes. |
| `fetchevents_by_id` | Queries for one or more events by a unique identifier (UID, Recurrence ID, modifier). |
| `fetchtodos_by_id` | Queries for one or more todos by a unique identifier (UID, Recurrence ID, modifier). |
| `get_all_timezones` | Returns all the timezones the server supports. |
| `get_calprops` | Returns calendar properties. |
| `get_freebusy` | Returns calendar freebusy time. |
| `get_guids` | Returns a set of random UIDs. |

**Table 7-1** WCAP Commands *(Continued)*

| | |
|---|---|
| `get_userprefs` | Returns user preferences and some server settings. |
| `import` | Imports a calendar from a file to a user's calendar. |
| `login` | Authenticates a user and redirects to first HTML view. |
| `logout` | Terminates the current user's session and return to login screen. |
| `ping` | Administrator only: Pings the calendar server. |
| `search_calprops` | Searches for a calendar with the specified parameter values. |
| `set_calprops` | Sets calendar properties. |
| `set_userprefs` | Sets user preferences. |
| `storeevents` | Stores events that are specified in application/urlencoded manner. For storing an even by passing properties in a URL. |
| `storetodos` | Stores todos that are specified in the application/urlencoded manner. |
| `upload_file` | Uploads a file to the server. |
| `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. |
| `write_file` | Writes a file to the database. |

# Common Topics

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

## Access Control Entries

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

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

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

Here is an example of an ACE string:

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

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

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

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

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

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

The ACE can be applied to:

- The entire calendar.

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

- The components of the calendar only.

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

- The calendar properties of the calendar only.

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

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

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

   Table 7-2 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 7-2** Access Control Characters

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

**Table 7-2** Access Control Characters

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

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

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

   The ACE can either grant or deny access.

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

### *ACE Summary*

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

who ^ flags ^ how ^ grant

Where:

- who = A string, type (str).
- flags = One of the characters `c`, `p`, or `a`.
- how = An access-string composed of one or more of the access control characters described earlier in Table 7-2.
- grant = One of the characters `g`, or `d`

### *Extended Examples*

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

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

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

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

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

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

# Choosing a Different Language or Character Set

To insert a request for data to be returned in a language other than the system default, set either the `lang` or `charset` parameter. Note that the system default for language is now a server preference that you set in the ics.conf file. See the *iPlanet 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 of Time, Strings, Parameters, Etc.," on page 93 in this chapter.

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

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

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

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

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

# Deleting Recurring Components

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

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

**Table 7-3** `mod` Parameter Delete Options

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

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

To delete just the single instance of the event, the `mod` parameter should be set to `1`. For example, this URL would delete just the event that occurs on the date March 1, 2002 11:22:33 AM GMT.

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

# Encoded Characters

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

%3D = '='

%26 = '&'

%22 = '"'

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

# Error Handling

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

## Error String

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

## Layer Error Number Array.

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

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

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

There are three layer error arrays:

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

## Layer Count Array.

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

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

## Error Codes

Table 7-4 list some of the error codes returned in the error number array.

**Table 7-4** Error Names, Values, and Meanings

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

**Table 7-4** Error Names, Values, and Meanings *(Continued)*

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

**Table 7-4** Error Names, Values, and Meanings *(Continued)*

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

# 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.

It is found in the following four commands:

- `fetchcomponents_by_alarmrange`
- `fetchcomponents_by_attendee_error`
- `fetchcomponents_by_lastmod`
- `fetchcomponents_by_range`

# 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 new parameter was added to the following commands:

- `fetchcomponents_by_alarmrange`
- `fetchcomponents_by_attendee_error`
- `fetchcomponents_by_lastmod`
- `fetchcomponents_by_range`
- `fetchevents_by_id`
- `fetchtodos_by_id`
- `storeevents`
- `storetodos`

**NOTE** This parameter is only valid when `fmt-out` is `text/calendar` or `text/xml`.

# Fetching Specific 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 7-5 lists component state values. A component state pertains either to the attendee or the organizer.

**Table 7-5** Component State Values for `compstate` Parameter

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

# Formatting of Time, Strings, Parameters, Etc.

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

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

# Freebusy Access

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

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

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

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

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

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

# Output Format

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

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

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

**NOTE** One exception to this is the `login` command. You must specify `fmt-out=text/html` in the `login` command to start the iPlanet Calendar Server 5.x user interface.

## Condensed Output

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

**Table 7-6** Output Parameters Included in `brief` Output

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

# Recurrence Handling

There are six parameters used to specify recurrence:

1. rrules. Semicolon-separated list of quoted recurrence-rule strings for recurring events.
2. rdates. Semicolon-separated list of ISO 8601 date strings listing recurrence dates.
3. exrules. Semicolon-separated list of quoted recurrence-rule strings for dates to exclude.
4. exdates. Semicolon-separated list of ISO 8601 date strings listing dates to exclude.
5. rid. ISO 8601 DateTimeString giving the recurrence ID of an event.
6. mod. Numeric values 1-4. Modifier telling which instances of the event to store.
7. rchange. A boolean specifying whether or not to expand recurrences in `storecomponents`.

## rrules

The `rrules` parameter takes a semicolon-separated list of quoted recurrence rule strings. Each string represents a recurrence rule of the event. Each string must be enclosed in quotes. There are many parameters possible for recurrence rules. See 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`.

The following example shows an `rrules` parameter that specifies two recurrence rules:

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

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

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

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

```
http://webcalendarserver/storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=20020301T112233Z
&rrules="count%3D10%3Bfreq%3Ddaily";"freq%3Dweekly%3Bcount%3D4"
&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=uu
uu
```

## rid

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

For example:

```
http://webcalendarserver/storecomponents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=333&dtstart=20020301T112233Z
&rid=20020331T112233;dtend=20020301T112233&summary=uuuu&mod=1
```

## mod

This parameter specifies whether to apply the changes to one or more instances of the event or todo. The `mod` values result in the following behavior:

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

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

### rchange

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

However, you might not want to expand recurrences when you are modifying multiple events. For example, suppose a meeting recurs every Friday starting Jan. 1, 2002. Use the following URL to change the summary of each event after Feb. 1, 2002 to `changed-event`. This example sets the `rchange` parameter to 0, to make the modification without adding additional events:

```
http://webcalendarserver/storeevents.wcap?id=b5q2o8ve2rk02nv9t6
&calid=jdoe&uid=abcxyz&dtstart=20020201T112233Z
&rrules="byday%3Dfr%3Bfreq%3Dweekly"&summary=changed-event
&rid=20020201T112233Z&mod=2&rchange=0
```

## Time Zones

To support a universal standard, iPlanet 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` parameter.

The `tzid` parameter is used only if the date and time strings, passed in with the dtstart, dtend, and rid parameters, are not in Zulu time. If the date and time strings are not specified 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 `tzid` parameter is passed in. In this case the Zulu time is translated into the time zone specified in the `tzid` parameter.

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.

The following commands use the `tzid` parameter to store or retrieve data:

- `deleteevents_by_id`
- `deletetodos_by_id`
- `fetchcomponents_by_alarmrange`
- `fetchcomponents_by_lastmod`
- `fetchcomponents_by_range`
- `fetchevents_by_id`
- `fetchtodos_by_id`
- `get_freebusy`
- `set_calprops`
- `storeevents`
- `storetodos`

# Commands

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

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

## addlink

### Purpose

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

### Parameters

Table 7-7 lists this command's five parameters:

**Table 7-7** `addlink` Parameters

| Parameter | Types | Purposes | Required | Default |
|---|---|---|---|---|
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `destCal` | string | The calendar containing the events or todos. You must have read access to this calendar. | Y | N/A |
| `rid` | semicolon separated list of strings. | A list of event and/or todo recurrence identifiers to which to create links. | Y | N/A |
| `srcCal` | string | The calendar receiving the links. You must have write access to this calendar. | Y | N/A |
| `uid` | semicolon separated list of strings. | A list of event and/or todo identifiers to which to create links. | Y | N/A |

### Description

Use this command to add links in the destination calendar to the specified events and/or todos in the source calendar.

You must specify the `id` parameter with the command unless the specified calendar is a public calendar.

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

Unless the following three requirements are met, the transaction will fail:

- The source calendar and destination calendar parameters must be valid calendar ID's.
- The user must have write access to the destination calendar and read access to the source calendar.
- The uid and rid lists must have the same number of elements.

#### Returns

If the transaction fails, an error of ADDLINK_FAILED(12) returns; otherwise, the return code is 0.

#### Example

This example shows how to add two event links to the calendar jdoe. The links should point to events in the pub calendar. The events are 1111 and 2222, (their uids). Neither event is recurring, so their recurrence-id is 0.

This URL executes the above transaction.

```
http://webcalendarserver/addlink.wcap?id=b5q2o8ve2rk02nv9t6&destCal
=jdoe&srcCal=pub&uid=1111;2222&rid=0;0
```

## change_password

#### Purpose

Change the password of the current user. This command is deprecated in iPlanet Calendar Server 5.x. It is here only for backward compatibility with iPlanet Calendar Server 2.x. See the "Administrator's Guide" for details on changing a password.

#### Parameters

Table 7-8 lists this command's four parameters:

**Table 7-8** change_password Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| id | string | The session identifier. | Y | N/A |
| newPassword | string | The new user password. | Y | N/A |
| oldPassword | string | The previous user password. | Y | N/A |

**Table 7-8** `change_password` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |

### 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 7-8 lists this command's four parameters:

**Table 7-9** `change_password` Parameters

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

**Table 7-9** `change_password` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |

#### 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:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
X-NSCP-WCAP-CHECK-ID:1
END:VCALENDAR lendar
```

# createcalendar

### Purpose

Create a new calendar.

### Parameters

Table 7-10 lists this command's four parameters:

**Table 7-10** `createcalendars` Parameters

| Parameter | Types | Purposes | Required | Default |
|---|---|---|---|---|
| `calid` | string | The user's `calid`, used to generate the new calendar's `calid`. | Y | N/A |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | Y | `text/js` |
| `set_calprops` | integer (`0`,`1`) | A boolean indicating whether to set the properties of the new calendar.<br>`1` = Set properties.<br>`0` = Do not set properties. | N | `0` |

### Description

Use this command to create a new calendar for the current user. To enable users who do not have administrative privileges to use this command, set the `service.wcap.allowcreatecalendars` preference in the `ics.conf` file.

#### Creating a Valid Calid

The new `calid` of the created calendar is a combination of the user's `userid` and the `calid` parameter passed in. The system retrieves the `userid` by doing a lookup on the session specified with the `id` parameter. The format for the new calendar's `calid` is *`userid`*:*`calid`*. For example, if the user is `jdoe`, and the `calid` parameter is `tv`, the new calendar's calid is `jdoe:tv`.

The server will attempt to truncate `calid` parameters that are too long or contain any illegal characters. If the server is unable to truncate the `calid` parameter, the error returned is `ILLEGAL_CALID_NAME(30)`.

Valid characters for the `calid` parameter are:

- Alphabet characters (A-Z, a-z)
- Numeric characters (0-9)
- Three special characters
  - Dash (-)
  - Underscore (_)
  - Period (.)

For example, these are legal values for the `calid` parameter: `calendar1`, `calendar-1`, `calendar_1`, `calendar.1`

#### Setting Calendar Properties

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.

#### Returns

The returned output shows the calendar properties (retrieved with a call to the fetchcomponents_by_range command) formatted according to the `fmt-out` value.

#### Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If the newly created `calid` already exists in the database, an error code returns: `CREATECALENDAR_ALREADY_EXISTS_FAILED(25)`

#### Example

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

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

# deletecalendar

### Purpose

This command deletes a user's calendar.

### Parameters

Table 7-11 lists this command's three parameters:

**Table 7-11** `deletecalendars` Parameters

| Parameter | Types | Purposes | Required | Default |
|---|---|---|---|---|
| `calid` | string | The name of the calendar to delete. | Y | N/A |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `fmt-out` | string | The format for the returned data. | Y | `text/js` |
| | | The three format types: | | |
| | | `text/calendar`<br>`text/xml`<br>`text/js` | | |

### Description

Use this command to delete a user's calendar. You must pass in the `calid`, which is the name of the calendar to delete.

Only users with administrative privilege may use this command unless the `service.wcap.allowdeletecalendars` preference is set.

### Returns

The returned output is the formatted output from a call to `fetchcomponents_by_range`.

### Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If the `calid` doesn't exist in the database, the `delete_layer_errno[x]` value is set to `1`, where `x` is the calendar's index in the passed `calid` list. In addition, the `errno` variable contains the error: `CALENDAR_DOES_NOT_EXIST(29)`.

### Example

For example, sending this URL deletes the calendar named `newcal`.

`http://webcalendarserver/deletecalendar.wcap?id=b5q2o8ve2rk02nv9t6&calid=newcal`

# deletecomponents_by_range

### Purpose

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

### Parameters

Table 7-12 lists this command's six parameters:

**Table 7-12** `deletecomponents_by_range` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief output.<br>`0` = Complete output. | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to delete events and todos. | N | Current user's `calid`. |
| `dtend` | ISO 8601 DateTime Z string (UTC) | End time and date of events or todos to be deleted.<br><br>A value of `0` means delete all events and todos up to the end of time. This value must be in coordinated universal time. | N | `0` |
| `dtstart` | ISO 8601 DateTime Z string (UTC) | Start time and date of events or todos to be deleted.<br><br>A value of `0` means delete all events and todos from the beginning of time. This value must be in coordinated universal time. | N | `0` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

### Description.

Use this command to delete the events and todos that fall completely within the specified range from the specified calendars. If a range is not specified, it deletes all events and todos. The range parameters, `dtstart` and `dtend`, should be specified in UTC time (the 'Z' must be on the end). Otherwise, results are unpredictable.

### Error Codes

If the operation is successful, the error number of `0` is appended to the error string, If an error occurs while deleting from the calendar, the `delete_layer_errno[x]` value is set to `1`, where `x` is the calendar's index in the passed `calid` list. In addition, `errno` contains the error: `DELETECOMPONENTS_BY_RANGE_FAILED(21)`.

### Example

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

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

## deleteevents_by_id

### Purpose

Deletes one or more events from a calendar by event identifier.

### Parameters

Table 7-13 lists this command's eight parameters:

**Table 7-13** `deleteevents_by_id` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | Calendar ID of event to delete. | Y | N/A |

**Table 7-13** `deleteevents_by_id` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. Required unless the calendar is public. | Y | `NULL` |
| `mod` | integer<br>`1,2,3,4` | A modifier indicating which recurrences to delete, or semicolon-separated list of modifiers. If a list, it must have same number of elements as `uid` list.<br>One of the following values:<br>`1` = `THISINSTANCE`<br>`2` = `THISANDFUTURE`<br>`3` = `THISANDPRIOR`<br>`4` = `THISANDALL` | Y | N/A |
| `notify` | integer<br>`0,1` | A boolean indicating whether or not to notify attendees of this change.<br>`1` = Notify attendees.<br>`0` = Do not notify attendees. | N | `0` |
| `rid` | string | Recurrence identifier of the event, or semicolon-separated list of recurrence identifiers.<br>If a list, it must have same number of elements as the `uid` list.<br>If there are no recurrences, the value is `0`. | Y | N/A |
| `tzid` | time zone ID string | Default time zone to use if the `rid` parameter does not have a time zone specified.<br>For example, "America/Los_Angeles" | N | server's default time zone |
| `uid` | string | Unique Identifier of an event to be deleted, or semicolon-separated list of unique identifiers. | Y | N/A |

**Description.**

Use this command to delete the specified event or events from the specified calendar.

#### Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If there was an error, the server returns two error arrays, `delete_layer_errno` and `delete_event_count`, with each element in the arrays representing the corresponding event in the `uid` list. Each element in `delete_layer_errno` indicates the error number for the corresponding event. Each element in `delete_event_count` indicated the number of occurrences successfully deleted for the corresponding event.

See also, "Error Handling" on page 88.

#### Notify

The `notify` parameter specifies whether or not to send an IMIP `CANCEL` message to the email attendees of the event. To send the cancellation message, set the `notify` value to `1`.

For example, here's a URL that sends the IMIP `CANCEL` message to all attendees of the event with `uid=001`.

```
http://webcalendarserver/deleteevents_by_id.wcap?id=3423423asdfasf&
calid=jdoe&uid=001&notify=1
```

#### Recurrences

If the `rid` parameter is passed, the command also deletes recurrences, as specified by the `mod` parameter. (See "Deleting Recurring Components" on page 86.) 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:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
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.

### Parameters

Table 7-14 lists this command's six parameters:

**Table 7-14** `deleteevents_by_range` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to delete events. | N | Current user's `calid`. |
| `dtend` | ISO 8601 DateTime string | End time and date of events to be deleted.<br><br>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.<br><br>A value of `0` means delete all events from the beginning of time. | N | `0` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |

**Table 7-14** `deleteevents_by_range` Parameters *(Continued)*

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

**Description.**
Use this command to delete the events that fall completely within the specified range from the specified calendars. If a range is not specified (`dtstart` and `dtend`), it deletes all events from the specified calendars.

You must specify the `id` parameter with the command unless the specified calendar is a public calendar. The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data returns in the default JavaScript format.

**Error Codes**
If the operation is successful, the error number of `0` is appended to the error string, `errno`. If an error occurs while deleting from the calendar, the `delete_layer_errno[x]` value is set to `1`, where `x` is the calendar's index in the passed `calid` list. In addition, the `errno` variable contains the error: `DELETEEVENTS_BY_RANGE_FAILED(22)`.

See also, "Error Handling" on page 88.

**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 7-15 lists this command's eight parameters:

**Table 7-15** `deletetodos_by_id` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | Calendar ID of the todo/todos to delete. | Y | N/A |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `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.<br><br>One of the following values:<br><br>`1` = THISINSTANCE<br>`2` = THISANDFUTURE<br>`3` = THISANDPRIOR<br>`4` = THISANDALL | Y | N/A |
| `notify` | integer (0,1) | A boolean telling whether or not to notify attendees of this change.<br><br>`1` = Notify attendees.<br>`0` = Do not notify attendees. | N | `0` |
| `rid` | string | The recurrence identifier of the todo, or a semicolon-separated list of recurrence identifiers.<br><br>If a list, it must have the same number of elements as the `uid` list.<br><br>If there are no recurrences, the value is `0`. | Y | N/A |

**Table 7-15** `deletetodos_by_id` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `tzid` | time zone ID string | Default time zone to use if `dtstart`, or `dtend` parameters do not have a time zone specified.<br>For example, "America/Los_Angeles" | N | server's default time zone |
| `uid` | string | The unique identifier of a todo to be deleted, or a semicolon-separated list of unique identifiers. | Y | N/A |

**Description.**
Use this command to delete the specified todo from the specified calendar.

You must specify the `id` parameter with the command unless the specified calendar is a public calendar. The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data returns in the default JavaScript format.

**Error Codes**
If there was an error, the server returns two error arrays, `delete_layer_errno` and `delete_todo_count`, with each element in the arrays representing the corresponding todo in the `uid` list. Each element in `delete_layer_errno` indicates the error number for the corresponding todo. Each element in `delete_todo_count` indicates the number of occurrences successfully deleted for the corresponding todo.

See also, "Error Handling" on page 88.

**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=jdoe&uid=001&notify=1
```

**Recurrences**
If the `rid` parameter is passed, the command also deletes recurrences, as specified by the `mod` parameter. See "Deleting Recurring Components" on page 86.

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:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
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.

### Parameters

Table 7-16 lists this command's six parameters:

**Table 7-16** `deletetodos_by_range` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to delete todos. | N | Current user's `calid` |
| `dtstart` | ISO 8601 DateTime string | Start time and date of todos to be deleted. A value of `0` means delete all todos up to a specified start-time. | N | `0` |
| `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.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |

### Description.

Use this command to delete the todos that fall completely within the specified range from the specified calendars. If a range is not specified, it deletes all todos. For example, the following URL would delete just the 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 JavaScript format.

### Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If an error occurs while deleting from the calendar, the `delete_layer_errno[x]` value is set to `1`, where `x` is the calendar's index in the passed `calid` list. In addition, the `errno` variable contains the error: `DELETETODOS_BY_RANGE_FAILED(23)`.

See also, "Error Handling" on page 88.

## export

### Purpose

Export events and todos from a calendar to a file.

### Parameters

Table 7-17 lists this command's five parameters:

**Table 7-17** `export` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `calid` | string | A semicolon-separated list of calendar identifiers from which to export events and todos. The user must have read access to all calendars in the list. | N | Current user's `calid`. |
| `content-out` | string | Content type for output file. One of the following:<br>`text/calendar`<br>`text/xml` | Y | N/A |
| `dtend` | ISO 8601 DateTime Z string. (UTC) | End time and date of the events and todos to export. A value of `0` means export all components from the start date to the latest date. | N | `0` |
| `dtstart` | ISO 8601 DateTime Z string. (UTC) | Start time and date of events and todos to export. A value of `0` means export all components from the earliest date to the end date. | N | `0` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

**Description.**

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

**Range**

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

**HTTP Post Examples**

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

### *Example 1*

The following HTTP `POST` message exports all components of the calendars `jdoe` and `john` to an iCalendar file named `export.ica`:

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

### *Example 2*

The following HTML generates a `POST` message using the `export` command, producing files in both iCalendar and XML formats:

```
<form METHOD=POST ENCTYPE="multipart/form-data" NAME="john.ics"
ACTION="http://webcalendarserver:12345/export.wcap?id=t9u9m0eh8x5pu
9b
  &calid=jdoe;john&dtstart=0&dtend=0&content-out=text/calendar">
```

```
<ul>
<li>Press Export ICAL Now:<input type="submit" value="Export ICAL
now">
</li> </ul> </form>
<form METHOD=POST ENCTYPE="multipart/form-data" NAME="john.xml"
ACTION="http://webcalendarserver:12345/export.wcap?id=t9u9m0eh8x5pu
9b
  &calid=jdoe;john&dtstart=0&dtend=0&content-out=text/xml">
<ul>
<li>Press Export XML Now:<input type="submit" value="Export XML
now”>
</ul> </form>
```

This is the output generated:

```
HTTP/1.0 200
Date: Thu, 03 Jun 2002 22:15:52 GMT
Content-type: text/calendar
Content-disposition: attachment; filename="export.ics"
Content-length: 7004

BEGIN:VCALENDAR
METHOD:PUBLISH
VERSION:5.1
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 7-18 lists this command's seven parameters:

**Table 7-18** `fetchcomponents_by_alarmrange` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (`0`,`1`) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to retrieve components. | N | Current user's `calid`. |

**Table 7-18** `fetchcomponents_by_alarmrange` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `compressed` | integer (`0`,`1`) | For `compressed=0`, returns less data. Specifically, it does not return the following parameters:<br>`rrules`<br>`rdate`<br>`exrule`<br>`exdate`<br>For `compressed=1`, all recurrence data is returned.<br>This parameter can only be used when `fmt-out` is `text/xml`, or `text/calendar` | N | `0` |
| `compstate` | semicolon-separated list of component state keywords | The list of component states to fetch.<br>For `compstate` values, see Table 7-5 on page 93 | N | `ALL` |
| `dtend` | ISO 8601 DateTime Z string | End time and date of events and todos to be returned.<br>A value of `0` means fetch all events. | N | `0` |
| `dtstart` | ISO 8601 DateTime Z string | Start time and date of events/todos with alarms ready to go off during the specified time.<br>A value of `0` means fetch all events from the beginning of time. | N | `0` |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `maxResults` | integer | The maximum number of events and todos to be returned. When `0`, no maximum is applied and the command returns all events and todos found. | N | `0` |

**Table 7-18** `fetchcomponents_by_alarmrange` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `tzid` | time zone ID string | Default time zone to use if the `rid` parameter does not have a time zone specified.<br><br>For example, "America/Los_Angeles" | N | server's default time zone |

#### Description.

This command returns a list of events and todos having alarms that are about to go off during the specified time.

#### Output Format

The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data is returned in the default JavaScript format.

For JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all-day events, print out in the `eventD` array. All day events are signified by having the `isAllDay` flag turned on.

#### maxResults Value

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

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

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

#### Returns

For each calendar specified in `calid`, the server returns the calendar's events and todos having alarms about to go off within the range specified by `dtstart` and `dtend`. Specify these in the ISO 8601 DateTime Z string format.

If neither the starting nor ending date-time is specified, the server returns all events and todos with alarms, up to the specified maximum.

### Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If a calendar cannot be accessed or is missing, `errno` is `FETCH_BY_ALARM_RANGE_FAILED(41)`.

### Example

For example, suppose there are 3 events:

- eventA: alarm on 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, 2002 and Jan. 31, 2002.

```
http://webcalendarserver/fetchcomponents_by_alarmrange.wcap?id=abcd
efg&dtstart=20011201T112233Z&dtend=20020131T112233Z&fmt-out=text/ca
lendar
```

It returns eventA and todoA:

```
BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
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:20030210T190000Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20031225T123000Z
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:20030120T141500Z
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 6, 2002.

```
http://webcalendarserver/fetchcomponents_by_alarmrange.wcap?id=abcd
efg&dtstart=20020101T000000Z&dtend=20020601T000000Z&fmt-out=
text/calendar
```

It returns eventB and todoA:

```
BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
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:20011225T213000Z
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 7-19 lists this command's five parameters

**Table 7-19** `fetchcomponents_by_attendee_error` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `attendee` | string | The attendee's `calid` to search on. The command searches the calendars specified in the `calid` parameter for all errors in events for this attendee.<br><br>If this parameter is not specified, the command searches the primaryOwner's calendars for all event errors for any attendee. | N | N/A |
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = brief<br>`0` = complete | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to retrieve components. | N | Current user's `calid`. |
| `compressed` | integer (`0`,`1`) | For `compressed=0`, returns less data. Specifically, it does not return the following parameters:<br>`rrules`<br>`rdate`<br>`exrule`<br>`exdate`<br><br>For `compressed=1`, all recurrence data is returned.<br><br>This parameter can only be used when `fmt-out` is `text/xml`, or `text/calendar` | N | `0` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

**Table 7-19** `fetchcomponents_by_attendee_error` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `maxResults` | integer | The maximum number of events and todos to be returned. When `0`, no maximum is applied and the command returns all events and todos found. | N | `0` |

#### Description.

Use this command to retrieve a list of events and todos that had errors when sending group scheduling messages. This command works almost like `fetchcomponents_by_range`.

#### Output Format

The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data is returned in the default JavaScript format.

For JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all-day events, print out in the `eventD` array. All-day events are signified by having the `isAllDay` flag turned on.

#### maxResults Value

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

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

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

#### Returns

For each calendar specified in `calid`, the server returns the events and todos that had errors for the specified attendee while sending group scheduling messages.

For example, if the `calid` parameter specifies calendars `cal1` and `cal2`, and the `attendee` parameter specifies `jdoe`, then both `cal1` and `cal2` would be searched for events with errors that had `jdoe` as an attendee. In the table that follows, cal1 and cal2 each have four events with associated attendees:

| cal1 Events | cal2 Events |
|---|---|
| event - 1c1 | event - 1c2 |
| attendee: `jdoe` | attendee: `john` |
| status: error | status: OK |
| event - 2c1 | event - 2c2 |
| attendee: `susan` | attendee: `jdoe` |
| status: error | status: error |
| event - 3c1 | event - 3c2 |
| attendee: `jdoe` | attendee: `susan` |
| status: OK | status: OK |
| event - 4c1 | event - 4c2 |
| attendee: `john` | attendee: `susan` |
| status: OK | status: error |

For attendee `jdoe`, the command returns: events 1c1 and 2c2.

#### Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If the command fails for any reason, `errno` is `FETCH_BY_ATTENDEE_ERROR_FAILED(42)`.

## fetchcomponents_by_lastmod

#### Purpose.

Fetch a list of components that have changed during a specified time period.

**Parameters.**

Table 7-20 lists this command's seven parameters

**Table 7-20** `fetchcomponents_by_lastmod` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = brief<br>`0` = complete | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to retrieve components. | N | Current user's `calid`. |
| `compressed` | integer (`0`,`1`) | For `compressed=0`, returns less data. Specifically, it does not return the following parameters:<br>`rrules`<br>`rdate`<br>`exrule`<br>`exdate`<br><br>For `compressed=1`, all recurrence data is returned.<br><br>This parameter can only be used when `fmt-out` is `text/xml`, or `text/calendar` | N | `0` |
| `compstate` | semicolon-separated list of component state keywords | The list of component states to fetch.<br><br>For `compstate` values, see Table 7-5 on page 93 | N | `ALL` |
| `dtend` | ISO 8601 DateTime Z string | End time and date of events to be returned.<br><br>A value of `0` means fetch all events. | N | `0` |
| `dtstart` | ISO 8601 DateTime Z string | Start time and date of events to be returned.<br><br>A value of `0` means fetch all events from the beginning of time. | N | `0` |

**Table 7-20** `fetchcomponents_by_lastmod` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `maxResults` | integer | The maximum number of events and todos to be returned. When `0`, no maximum is applied and the command returns all events and todos found. | N | `0` |
| `tzid` | time zone ID string | Default time zone to use if `dtstart`, or `dtend` parameters do not have a time zone specified.<br>For example, "America/Los_Angeles" | N | server's default time zone |

**Description.**
Use this command to retrieve a list of events and todos that have changed during a specific time period. This command works almost like `fetchcomponents_by_range`.

**Output Format**
The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data is returned in the default JavaScript format.

For JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all-day events, print out in the `eventD` array. All-day events are signified by having the `isAllDay` flag turned on.

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

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

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

#### Returns

For each calendar specified in `calid`, the server returns the calendar's the events and todos that changed during the range specified by `dtstart` and `dtend`. Specify these in the ISO 8601 DateTime Z string format.

If neither the starting nor ending date-time is specified, the server returns all events and todos that have changed, up to the specified maximum.

#### Error Codes

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

#### Example

For example, the calendar `jdoe` has these three events:

- eventA: last-modified on Feb. 10, 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 12/1/2002 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 7-21 lists this command's seven parameters:

**Table 7-21** `fetchcomponents_by_range` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | Semicolon-separated list of calendar identifiers from which to retrieve components. | N | Current user's `calid`. |
| `compressed` | integer (`0`,`1`) | For `compressed=0`, returns less data. Specifically, it does not return the following parameters:<br>`rrules`<br>`rdate`<br>`exrule`<br>`exdate`<br><br>For `compressed=1`, all recurrence data is returned.<br><br>This parameter can only be used when `fmt-out` is `text/xml`, or `text/calendar` | N | `0` |
| `compstate` | semicolon-separated list of component state keywords | The list of component states to fetch.<br><br>For `compstate` values, see Table 7-5 on page 93 | N | `ALL` |
| `dtend` | ISO 8601 DateTime Z string | End time and date of events to be returned.<br><br>A value of `0` means fetch all events. | N | `0` |

**Table 7-21** `fetchcomponents_by_range` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `dtstart` | ISO 8601 DateTime Z string | Start time and date of events to be returned.<br><br>A value of `0` means fetch all events from the beginning of time. | N | `0` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `maxResults` | integer | The maximum number of events and todos to be returned. When `0`, no maximum is applied and the command returns all events and todos found. | N | `0` |
| `tzid` | time zone ID string | Default time zone to use if `dtstart`, or `dtend` parameters do not have a time zone specified.<br><br>For example, "America/Los_Angeles" | N | server's default time zone |

**Description.**
Use this command to retrieve properties, events, and todos from one or more specified calendars.

**Output Format**
The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data is returned in the default JavaScript format.

For JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all day events, print out in the `eventD` array. All day events are signified by having the `isAllDay` flag turned on.

**Returns**
For each calendar specified in `calid`, the server returns the calendar's properties and the events and todos of that calendar that fall within the range specified by `dtstart` and `dtend`. Specify these in the ISO 8601 DateTime Z string format.

If neither the starting nor ending date-time is specified, the server returns all events and todos, up to the specified maximum.e

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

Here is the returned text:

timezoneList[0]=new TZ('GMT',
'GMT',
'GMT',
'+0000',
'+0000',
new Array())

#### 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 JavaScript format.

For JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all-day events, print out in the `eventD` array. All-day events are signified by having the `isAllDay` flag turned on.

#### maxResults Value

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

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

If the `maxResults` parameter is set to `0` or is not passed, then the command does not cap the number of returned components, and the returned JavaScript does not contain the `var maxResults` statement.

### Error Codes

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

### Example

Two examples follow. The first fetches events for the current user from Dec. 1, 2002 to Jan. 31, 2003, use the following URL. The second example fetches all events for calendars jdoe and susan between the same dates.

*Example 1*

```
http://webcalendarserver/fetchcomponents_by_range.wcap?id=bes6bbe2m
u98uw9&dtstart=20021201T000000Z&dtend=20030131T000000Z&fmt-out=text
/calendar
```

It returns one event and one todo for this period:

```
BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
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:20021225T133000Z
```

```
DTEND:20021225T143000Z
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:jsmit@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:20021208T015014Z
SUMMARY:todoA
DTSTART:20021208T004626Z
DUE:20030120T141500Z
CREATED:20021208T004626Z
LAST-MODIFIED:20021208T011000Z
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:20030120T131500Z
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*

```
http://webcalendarserver/fetchcomponents_by_range.wcap?id=bes6bbe2m
u98uw9&calid=jdoe;jsmith&dtstart=20020101T000000Z&dtend=20020202T00
0000Z&fmt-out=text/calendar
```

The following events and todos are returned:

```
BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20020913T223336Z
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: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:20011225T213000Z
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:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
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:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
X-NSCP-CALPROPS-LAST-MODIFIED:20021010T001050Z
X-NSCP-CALPROPS-CREATED:20020929T180436Z
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:@^a^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^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:20020210T110000Z
DTEND:20020210T120020Z
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:20011225T213000Z
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:susan@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":65538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

## fetchevents_by_id

### Purpose

Retrieve specific calendar events.

### Parameters

Table 7-22 lists this command's seven parameters:

**Table 7-22** `fetchevents_by_id` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | The unique identifier for the calendar from which to retrieve events. | N | Current user's `calid`. |
| `compressed` | integer (`0`,`1`) | For `compressed=0`, returns less data. Specifically, it does not return the following parameters:<br>`rrules`<br>`rdate`<br>`exrule`<br>`exdate`<br><br>For `compressed=1`, all recurrence data is returned.<br><br>This parameter can only be used when `fmt-out` is `text/xml`, or `text/calendar` | N | `0` |

**Table 7-22** `fetchevents_by_id` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `compstate` | semicolon-separated list of component state keywords | The list of component states to fetch.<br>For `compstate` values, see Table 7-5 on page 93 | N | `ALL` |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `mod` | integer | A modifier indicating which recurrences to retrieve. One of the following values:<br>1 = `THISINSTANCE`<br>2 = `THISANDFUTURE`<br>3 = `THISANDPRIOR`<br>4 = `THISANDALL` | N | `1`<br>`(THISINSTANCE)` |
| `rid` | ISO 8601 DateTime Z string | The recurrence identifier for the event. For a nonrecurring event, set to `0`. | N | `0` |
| `tzid` | time zone ID string | Default time zone to use if the `rid` parameter does not have a time zone specified.<br>For example, “America/Los_Angeles” | N | server’s default time zone |
| `uid` | sting | The unique identifier for the event. | Y | N/A |

**Description.**
Use this command to retrieve the specified events and recurrences from the specified calendar. You must specify the `id` parameter with the command unless the specified calendar is a public calendar. The command returns recurrences as specified by the `mod` parameter. See “Recurrence Handling” on page 96

**Output Format**
The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data is returned in the default JavaScript format.

For JavaScript output, events are divided into two arrays. Events that span a smaller range of time, most normal events, print out in the `event` array. All events that last longer than 24 hours, or are all-day events, print out in the `eventD` array. All-day events are signified by having the `isAllDay` flag turned on.

#### Returns

The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data is returned in the default JavaScript format.

#### Error Codes

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

#### Example

This query retrieves an event with a specific id.

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

It returns one event:

```
BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20020913T223336Z
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@iplanet.com
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@iplanet.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 7-23 lists this command's seven parameters:

**Table 7-23** `fetchtodos_by_id` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief ouput.<br>`0` = Complete ouput. | N | `0` |
| `calid` | string | The unique identifier for the calendar from which to retrieve todos. | N | Current user's `calid`. |
| `compressed` | integer (`0`,`1`) | For `compressed=0`, returns less data. Specifically, it does not return the following parameters:<br>`rrules`<br>`rdate`<br>`exrule`<br>`exdate`<br><br>For `compressed=1`, all recurrence data is returned.<br><br>This parameter can only be used when `fmt-out` is `text/xml`, or `text/calendar` | N | `0` |
| `compstate` | semicolon-separated list of component state keywords | The list of component states to fetch.<br><br>For `compstate` values, see Table 7-5 on page 93 | N | `ALL` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

**Table 7-23** `fetchtodos_by_id` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `mod` | integer | A modifier indicating which recurrences to retrieve. One of the following values:<br>1 = THISINSTANCE<br>2 = THISANDFUTURE<br>3 = THISANDPRIOR<br>4 = THISANDALL | N | `1` (`THISINSTANCE`) |
| `rid` | ISO 8601 DateTime Z string | The recurrence identifier for the todo. For a nonrecurring todo, set to `0`. | N | `0` |
| `tzid` | time zone ID string | Default time zone to use if the `rid` parameter does not have a time zone specified.<br>For example, "America/Los_Angeles" | N | server's default time zone |
| `uid` | sting | The unique identifier for the todo. | Y | N/A |

#### Description

Use this command to retrieve the specified todo and its recurrences from the specified calendar. You must specify the `id` parameter with the command unless the specified calendar is a public calendar.

#### Output Format

The server returns data in the format specified by the `fmt-out` parameter. If this parameter is not passed, the data is returned in the default JavaScript format.

For JavaScript output, todos are divided into two arrays. Todos that span a smaller range of time, most normal todos, print out in the `todo` array. All todos that last longer than 24 hours, or are all-day todos, print out in the `todoD` array. All-day todos are signified by having the `isAllDay` flag turned on.

#### Returns

For each calendar specified in `calid`, the server returns the calendar's todos of that calendar. If the todo has recurrences, it returns them as specified by the `rid` and `mod` parameters. See "Recurrence Handling" on page 96.

#### 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:-//iPlanet/Calendar Hosting Server//EN

METHOD:PUBLIS
VERSION:5.1
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20020913T223336Z
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=3c15309d00003702002002140000 3189&
rid=20020222T170000Z&mod=2&fmt-out=text/calendar
```

The results of the query are as follows:

```
BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20020913T223336Z
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:3c15309d0000370200200021400003189
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:3c15309d0000370200200021400003189
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 7-24 lists this command's four parameters:

**Table 7-24** `get_all_timezones` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `dtend` | ISO 8601 DateTime Z string | End date of the crossover values to retrieve.<br>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.<br>A value of `0` means get all crossover dates from the first known year (1987). | N | `0` |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

**Description.**
Use this command to retrieve data about all timezones that are supported by the server. The crossover values are defined to be the dates when the timezone enters/exits daylight savings time. The odd index dates are the beginning of daylight-savings. The even index dates are the end of daylight-savings. If the timezone does not have daylight-savings, then this value will be set to the empty-string.

**Returns**
If you specify a range of years with the `dtstart` and `dtend` parameters, the command returns only the crossover dates for the years within the range. Otherwise, it returns all crossover dates from the first to the last known year (1987-2087).

The server returns data in the format specified by the `fmt-out` parameter. If you do not pass in the `fmt-out` parameter, the server uses the default JavaScript format.

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

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

*Example 1*

This query gets all time zones.

```
http://birdie.red.iplanet.com/get_all_timezones.wcap?
id=2m2ns6w9x9h2mr6p3b&fmt-out=text/calendar
```

This is the result of the query:

```
BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
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:probably messed up also
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 7-25 lists this command's three parameters:

**Table 7-25** `get_calprops` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `calid` | semicolon-separated list of strings | A list of calendar identifiers for the calendars from which to retrieve properties. | N | Current user's `calid`. |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

### Description.

Use this command to retrieve the calendar properties for the specified calendars.

### Returns

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

- relative ID
- display name
- parent calendar ID
- timezone ID
- read-access value (`0` = cannot read, `1` = can read)
- write-access value (`0` = cannot 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, `layer_errno[x]` is set to the value `1` for that calendar's index.

If the calendar is not found, the server sets `layer_errno[x]` to the value `2` for that calendar's index.

If the fetch fails for any calendar, its error number, `errno`, is set to `GET_CALPROPS_FAILED(20)`.

#### Example

In the following example, you want to retrieve the calendar properties for the calendars `jdoe`, `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:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20020913T223336Z
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:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
X-NSCP-CALPROPS-LAST-MODIFIED:20021115T234638Z
X-NSCP-CALPROPS-CREATED:20021115T234638Z
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:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
X-NSCP-CALPROPS-LAST-MODIFIED:20021130T002458Z
X-NSCP-CALPROPS-CREATED:20020914T015956Z
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 7-26 lists this command's five parameters:

**Table 7-26** `get_freebusy` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `busyonly` | Integer (`0`, `1`) | `0` = return both busy and free periods<br>`1` = return only busy periods | N | `0` |
| `calid` | semicolon-separated list of strings | A list of calendar identifiers for the calendars from which to retrieve properties. | N | 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 |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `tzid` | timezone-ID string | Default time zone to use if `dtstart`, or `dtend` parameters do not have a time zone specified.<br>For example, "America/Los_Angeles" | N | Server's default timezone. |

### Description.

This command to retrieve the freebusy information for specified users. Freebusy information tells whether or not a user's time has been scheduled. It does not indicate why the user is busy.

### Error Codes

If this command fails for any reason, `errno` is set to `GET_FREEBUSY_FAILED(39)`.

**Example**

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

| | |
|---|---|
| 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&c
alid=jsun&dtstart=20020501T112233Z&dtend=20020701T112233Z&fmt-out=t
ext/calendar
```

Here is the output:

```
BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
X-NSCP-CALPROPS-LAST-MODIFIED:20020517T012259Z
X-NSCP-CALPROPS-CREATED:20020517T012259Z
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 7-27 lists this command's two parameters:

**Table 7-27** `get_guids` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `guidCount` | integer | Number of `GUID`s to return. | N | `1` |
| `fmt-out` | string | The format for the returned data. | N | `text/js` |
| | | The three format types: | | |
| | | `text/calendar`<br>`text/xml`<br>`text/js` | | |

### 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:5.1
PRODID:iPlanet Calendar Server 5.1
X-NSCP-GUID0:e5e4b537465600000b000000c3000000
X-NSCP-GUID1:e5e4b537d47900000c000000c3000000
X-NSCP-GUID2:e5e4b537961400000d000000c3000000
X-NSCP-GUID3:e5e4b5373d3a00000e000000c3000000
X-NSCP-GUID4:e5e4b537f31400000f000000c3000000
X-NSCP-GUID5:e5e4b5378259000010000000c3000000
X-NSCP-GUID6:e5e4b537b026000011000000c3000000
X-NSCP-GUID7:e5e4b537c263000012002002c3000000
X-NSCP-GUID8:e5e4b537241f000013000000c3000000
X-NSCP-GUID9:e5e4b537e733000014000000c3000000
END:VCALENDAR
```

## get_userprefs

### Purpose

Retrieve the calendar preferences for the current user.

### Parameters

Table 7-28 lists this command's two parameters:

**Table 7-28** `get_userprefs` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `userid` | string | Used only by administrators. Indicates which user's preferences to get. | N | N/A |

**Description.**

This command retrieves all the calendar preferences for the current user, and the following server preferences relating to this user:

- `allowchangepassword`. Users can change the password.
- `allowcreatecalendars`. Users can create calendars.
- `allowdeletecalendars`. Users can delete calendars.
- `allowpublicwritablecalendars`. Users can have publicly writable calendars.
- `validateowners`. If set to 1, the server must validate that each owner of a calendar exists in the directory (whether the directory is LDAP or a CSAPI compatible user mechanism).
- allowsetprefs. If set to 1, allow set_userprefs.wcap to modify the user preferences.

To use the parameter `userid`, two conditions must be met. The server configuration preference `service.admin.calmaster.wcap.allowgetmodifyuserprefs` must be set to "yes" in the `ics.conf` file, and the requestor must be logged in as an administrator, using the `login.wcap` command.

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

**Example**

The following URL retrieves user preferences for the current user:

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

This is the data returned:

```
BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
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-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:
X-NSCP-WCAP-PREF-ceDefaultAlarmEmail:jdoe@sesta.com
X-NSCP-WCAP-SERVER-PREF-allowchangepassword:no
X-NSCP-WCAP-SERVER-PREF-allowcreatecalendars:yes
X-NSCP-WCAP-SERVER-PREF-allowdeletecalendars:yes
X-NSCP-WCAP-SERVER-PREF-allowpublicwritablecalendars:yes
X-NSCP-WCAP-SERVER-PREF-validateowners:no
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

# import

### Purpose

Import events and todos from a file to a calendar.

### Parameters

Table 7-29 lists this command's five parameters:

**Table 7-29** `import` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `calid` | string | Identifier of a calendar to which to import event. | Y | N/A |
| `content-in` | string | Content type of input data. One of the following values:<br>`text/calendar`<br>`text/xml` | Y | N/A |
| `dtend` | ISO 8601 DateTime Z string | End time and date of the events and todos to import.<br>A value of `0` means import all components from the start date to the last date in the file. | N | `0` |
| `dtstart` | ISO 8601 DateTime Z string | Start time and date of events and todos to import.<br>A value of `0` means import all components from the earliest date in the file to the end date. | N | `0` |

**Table 7-29** `import` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `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:20020105T100000
DTEND:20020105T110000
DTSTAMP:20020104T120020
CREATED:20020105T110000Z
LAST-MODIFIED:20020104T120020Z
SUMMARY:Weekly QA Meeting
UID:random-uid001
END:VEVENT
BEGIN:VEVENT
DTSTART:20020106T100000
```

```
DTEND:20020106T110000
DTSTAMP:20020104T120020
CREATED:20020105T110000Z
LAST-MODIFIED:20020104T120020Z
SUMMARY:Weekly QA Meeting 2
UID:random-uid002
END:VEVENT
END:VCALENDAR
-----------------------------33111928916708--
```

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

```
<FORM METHOD=POST ENCTYPE="multipart/form-data"
ACTION="http://webcalendarserver:12345/import.wcap?id=t95qm0n0es3bo
35r&calid=jdoe&dtstart=0&dtend=0&content-in=text/calendar">
<ol>
<li>file to import:<input type="file" accept="text" name="Upload">
</li>
<li>Press Import Now:<input type="submit" value="Import Now"></li>
</ol>
</FORM>
```

# login

### Purpose

Authenticate a specific user.

### Parameters

Table 7-30 lists this command's five parameters:

**Table 7-30** `login` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js`<br>`text/html (see NOTE below)`<br><br>If `text/calendar` or `text/xml`, the refresh parameter is automatically set to 1.<br><br>NOTE: You must specify `text/html` to log in to the user interface for iPlanet Calendar Server 5.x. That is the only use for this format type. The format type is invalid in all other commands. | N | `text/js` |
| `lang` | enum | The user's preferred language. | N | `NULL` |
| `password` | string | The user's password. | N | N/A |
| `refresh` | integer (0, 1) | A boolean indicating whether to return just the new session ID or everything.<br><br>1 = Return only the session identifier.<br>0 = Return everything. | N | `0` |
| `user` | string | The user's name. | N | `NULL` |

### Description.

This command logs a specific user into iPlanet Calendar Server, authenticating the user to the server with a user name and password convention.

The user name is a plain text string that uniquely identifies the user to the server. This user name could, for example, be the same as a user's email address. The password is also plain text.

#### 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 iPlanet 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 iPlanet 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 "csIAuthentication" on page 26). For more information on the Proxy Authentication SDK, see Chapter 3 for the overview and Chapter 4 for the API Reference.

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

#### Example

For example, the following URL attempts to login user `jdoe`:

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

#### Returns

If you do not pass in the `refresh` parameter, and `fmt-out` specifies `text/js`, the default value is 0. If `fmt-out` specifies one of the two other styles, `text/calendar` or `text/xml`, the default value is `1`.

If you specify `1` in the `refresh` parameter, the returned page contains only the session identifier in a line such as the following:

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

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

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

With the parameter set to `0`, the `login` command returns everything, including the default html file:

```
HTTP/1.0 302 OK
Date: Tue, 11 May 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/

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

# logout

### Purpose

Terminate the current user's session.

### Parameters

Table 7-31 lists this command's two parameters:

**Table 7-31** `logout` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |

### Description.

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

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

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

## ping

### Purpose

Determine whether the calendar server is active.

### Parameters

This command takes no parameters.

### Description.

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

Only users with administrative privilege can use this command.

### Returns

For this example, the administrator's `userid` and `calid` are both `adminX`.

```
HTTP/1.0 200
Date: Thu, 03 Jun 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

<html><head><script>
function color(s) { if (s) document.bgColor=s }

var id='bb5rt6eb5pu9v9w9'
var userid='adminX'
var calid='adminX'
var errno=new Array()
parent.ceCB(window.name)
</script></head></html>
```

## search_calprops

### Purpose

Search for a calendar's properties.

### Parameters

Table 7-32 lists this command's seven parameters:

**Table 7-32** `search_calprops` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `calid` | integer (0,1) | A boolean indicating whether or not to search the `calid` property.<br>`1` = Search the `calid` property<br>`0` = Do not search it. | N | `0`, unless both `primaryOwner` and `name` are `0` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `maxResults` | integer | The maximum number of results to return. | N | `200` |
| `name` | integer (0,1) | A boolean indicating whether or not to search the `name` property.<br>`1` = Search the `calid` property<br>`0` = Do not search it. | N | `0` |
| `primaryOwner` | integer (0,1) | A boolean indicating whether or not to search the `primaryOwner` property.<br>`1` = Search the `primaryOwner` property.<br>`0` = Do not search it. | N | `0` |
| `searchOpts` | integer<br>0,1,2,3 | How to perform the search. One of the following:<br>`0 = CONTAINS`<br>`1 = BEGINS_WITH`<br>`2 = ENDS_WITH`<br>`3 = EXACT` | N | `0` |
| `search-string` | string | The string to search for in calendars. | Y | N/A |

**Description.**
This command searches for a calendar using the query type specified by `searchOpts`. It returns the calendar properties for all calendars where a string in the specified properties (`primaryOwner`, `calid`, `name`), matches the `search-string`, using the specified `seachOpts`, up to the specified maximum number of matches (`maxResults`).

**Search Properties**
This command searches for a matching string in one of three properties:

- `calid`. The calendar's unique identifier.
- `name`. The calendar's common name (text).
- `primaryOwner`. The calendar's primary owner.

To search for the value of a specific property, set that parameter to `1`. If both `primaryOwner` and `name` are set to `0`, `calid` defaults to `1` and the server assumes the `search-string` is a `calid`, regardless of the `calid` parameter setting.

**Search Options**
There are four search options:

- Return the calendar properties that contain the `search-string` (`CONTAINS`).
- Return the calendar properties that begin with the `search-string` (`BEGINS_WITH`).
- Return the calendar properties that ends with the `search-string` (`ENDS_WITH`).
- Return the calendar properties that exactly match the `search-string` (`EXACT`).

**Example**
The following example URL searches the primary owner property (primaryOwner=1) in all calendars to see if it contains (searchOpts=0) the string "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:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
```

```
X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20020913T223336Z
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:-//iPlanet/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:5.1
X-NSCP-CALPROPS-LAST-MODIFIED:20020917T213724Z
X-NSCP-CALPROPS-CREATED:20020917T213724Z
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 7-33 lists this command's fifteen parameters:

**Table 7-33** `set_calprops` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| acl | string | A semicolon-separated list of strings specifying the new value of the access control entries. | N | ,,,, |
| `cal` | encoded string | A list of parameters to decode.<br>There can be multiple instances of this parameter. | N | N/A |
| `calid` | string | Identifier of the calendar to modify. | Y | N/A |
| `categories` | string | A semicolon-separated list of strings containing the new categories the calendar belongs to. | N | N/A |
| `charset` | string | The character set for the calendar. | N | N/A |
| `description` | string | The description of the calendar. | N | N/A |
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `lang` | string | The language of the calendar. | N | N/A |
| `master` | string | The email contact for the calendar. | N | N/A |
| `multiple` | integer | The number of calendars for which to set these preferences. | N | `0` |
| `name` | string | The new text name of the calendar. | N | N/A |
| `owners` | string | A semicolon-separated list of strings containing he new list non-primary owners. | N | N/A |

**Table 7-33** `set_calprops` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `read` | integer | This parameter no longer has functionality in iPlanet Calendar Server 5.x. It remains here only for backwards compatibility with iPlanet Calendar Server 2.x. The `acl` parameter replaces it.<br><br>The new read-access value of the calendar. The value can be one of the following:<br><br>`0 PRIVATE`<br>`1 PUBLIC`<br>`4 PRIMARY_OWNER_ONLY` | N | N/A |
| `tzid` | string | The new timezone identifier for this calendar. | N | ,,,, |
| `write` | integer | This parameter no longer has functionality in version iPlanet Calendar Server 5.x. It remains here only for backwards compatibility with iPlanet Calendar Server 2.x. The `acl` parameter replaces it.<br><br>The new write-access value of the calendar. The value can be one of the following:<br><br>`0 PRIVATE`<br>`1 PUBLIC`<br>`4 PRIMARY_OWNER_ONLY` | N | N/A |

**Description.**

This command only changes the values of the parameters you specify. In this way, then, it is an *update* command, rather than a *replace* command as it was in 2.x. It is not necessary to supply all parameters in the command, only the ones you wish to change. Calendar properties are special states of a calendar, which includes the calendar's name, read and write permission values (`acl` parameter), the list of owners, and the list of categories.

The `read` and `write` parameters have been deprecated in iPlanet Calendar Server 5.x. The functionality has been replaced with the `acl` parameter. They are included for backwards compatibility with iPlanet Calendar Server 2.x only.

Use `set_calprops` to do the following:

- Change the name of the calendar.
- Change owner of calendar.

- Change category of calendar.
- Change read permission of calendar's event.
- Change write permission of calendar's event.
- Change description of calendar.
- Change character set of calendar.
- Change language of calendar.
- Change e-mail contact of this calendar.
- Change the timezone-identifier of the calendar.

#### Single Calendar Example

Here is a sample URL that sets calendar properties: (The `calid` parameter is required.)

```
http://webcalendarserver?set_calprops.wcap?id=dfasdfzd3ds&calid=jdo
e&categories=business;meeting&name=John%39s%32Calendar
```

#### Multiple Calendars Example

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

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

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

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

This is the equivalent of the following three URLs:

`http://webcalendarserver?id=fasdfzd3ds&calid=xxxx&desc=X-Calendar`

`http://webcalendarserver?id=fasdfzd3ds&calid=yyyy&desc=Y-Calendar`

`http://webcalendarserver?id=fasdfzd3ds&calid=zzzz&desc=Z-Calendar`

In the example, notice that since the `multiple` parameter is set to `3`, there are three instances of the `cal` parameter. The value of each `cal` parameter is an encoded list of parameters and their values. The server will decode each `cal` parameter and set the properties appropriately.

#### Access Control Entries

See "Access Control Entries," on page 82, in the Common Topics section at the front of this chapter.

#### Freebusy Access

See "Freebusy Access," on page 94, in the Common Topics section at the front of this chapter.

#### Choosing a Different Language or Character Set

See "Choosing a Different Language or Character Set," on page 85, in the Common Topics section at the front of this chapter.

## set_userprefs

#### Purpose

Modify the preferences or password for a session.

#### Parameters

Table 7-34 lists this command's five parameters:

**Table 7-34** `set_userprefs` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `add_attrs` | string | Add a new preference. | N | N/A |
| `convertCalid` | integer (`0`,`1`) | When set to `1` and setting the preferences `icsSet` or `icsSubscribed`, indicates to the server to convert the character "^" to ":" when storing the `calid`.<br>When set to `0`, the parameter is ignored. | N | `0` |
| `del_attrs` | string | Delete an existing preference. | N | N/A |

**Table 7-34** `set_userprefs` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | Y | `text/js` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `set_attrs` | string | Modify a preference value. | N | N/A |
| `userid` | string | Used only by administrators. Indicates which user's preferences to set. | N | N/A |

**Description.**

This command modifies the preferences for the current user. You may also modify the user's password through LDAP.

Use of this parameter is only necessary when setting the subscribed list of calendars (`icsSubscribed`), or the subscribed list of groups (`icsSet`). The `calid` on incoming commands must have the colon, ":", replaced with a caret, "^". For example, if the `calid` is `jdoe:personal`, then WCAP must receive it as `jdoe^personal` in order for the command to work properly.

If the value of `convertCalid` is `1`, then WCAP will convert the "^" back to a ":". If the value of the `convertCalid` is `0`, the conversion will not be done

When the administrator is logged in, and the ics.conf file preference `service.admin.calmaster.wcap.allowgetmodifyuserprefs` is set to "yes", the `userid` parameter specifies which user's preferences to set.

**Returns**

The function returns the text of `get_userprefs`.

**Examples**

For example, the following URL adds a new preference, `ceBgcolor`, to the calendar and sets it to `black`:

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

This URL deletes the calendar preference `ceBgcolor` from the user's preferences.

```
http://webcalendarserver/set_userprefs.wcap?id=b5q2o8ve2rk02nv9t6
&del_attrs=ceBgcolor
```

This URL would modify the calendar preference `ceBgcolor` to have the value `white`:

```
http://webcalendarserver/set_useprefs.wcap?id=b5q2o8ve2rk02nv9t6
&set_attrs=ceBgcolor=white
```

This URL would allow the logged-in administrator to modify the calendar preference `ceBgcolor` to have the value `black` for user `jdoe`:

```
http://webcalendarserver/set_userprefs.wcap?id=b5q2o8ve2rk02nv9t6&u
serid=jdoe&set_attrs=ceBgcolor=black
```

## storeevents

### Purpose

Add events to a calendar.

### Parameters

Table 7-35 lists this command's forty-three parameters:

**Table 7-35** `storeevents` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `alarmAudio` | ISO 8601 Date Time Z 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 event. | N | N/A |
| `alarmFlashing` | ISO 8601 Date Time Z string | The time at which to run flashing alarm. | N | N/A |
| `alarmPopup` | ISO 8601 Date Time Z string | The time at which to pop up a dialog alarm. | N | N/A |
| `alarmStart` | ISO 8601 DateTime Z string | The time at which to send the event alarm notification. | N | N/A |
| `attachments` | semicolon-separated list of strings | This is for iCalendar interoperability only. The strings are URLs. | N | N/A |

**Table 7-35** `storeevents` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `attendees` | semicolon-separated list of strings | The attendees of the event. | N | N/A |
| brief | integer (`0`,`1`) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = brief<br>`0` = complete | N | `0` |
| `calid` | string | Calendar identifier in which to store the event. | Y | N/A |
| `categories` | semicolon-separated list of strings | The event categories. | N | N/A |
| `compressed` | integer (`0`,`1`) | For `compressed=0`, returns less data. Specifically, it does not return the following parameters:<br>`rrules`<br>`rdate`<br>`exrule`<br>`exdate`<br><br>For `compressed=1`, all recurrence data is returned.<br><br>This parameter can only be used when `fmt-out` is `text/xml`, or `text/calendar` | 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.<br><br>To include spaces in the string, use the code `%20`. | N | Value of `summary` parameter. |
| `dtend` | ISO 8601 DateTime Z string | Event end time and date. | N | N/A |

**Table 7-35** `storeevents` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `dtstart` | ISO 8601 DateTime Z string | Event start time and date. | | N/A |
| | | Not required to modify events. | N | |
| | | Required to create events. | Y | |
| `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 DateTime Z strings | Event exclusionary recurrence dates. | N | N/A |
| `exrules` | semicolon-separated list of strings | Event exclusionary recurrence rules. A semicolon-separated list of recurrence-rule strings. | N | N/A |
| | | Each rule value must be enclosed in quotes. See "Recurrence Handling" on page 96. | | |
| fetch | integer (`0`,`1`) | A boolean indicating whether or not to fetch and return newly stored todos. | N | `0` |
| | | `1` = Fetch and return newly stored todos.<br>`0` = Do not fetch. | | |
| fmt-out | string | The format for the returned data. | N | `text/js` |
| | | The three format types: | | |
| | | `text/calendar`<br>`text/xml`<br>`text/js` | | |
| `geo` | two semicolon-separated floats | Semicolon-separated string of two float numbers representing the event's geographical location (latitude and longitude). | N | `0;0` |
| | | For example, `37.31;-123.2`. | | |
| `icsClass` | string | Event class. | N | `PUBLIC` |
| | | One of the following values: | | |
| | | `PUBLIC`<br>`PRIVATE`<br>`CONFIDENTIAL` | | |

**Table 7-35** `storeevents` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `icsUrl` | string | Event URL. | N | `""` |
| `id` | unique identifier string | The session identifier. | Y | N/A |
| `isAllDay` | integer (0,1) | A boolean indicating whether or not the event lasts all day.<br>`1` = Lasts all day.<br>`0` = Does not last all day. | N | `0` |
| `language` | string | Language of event. (For example, “en”, “fr”, “de”) | N | N/A |
| `location` | string | Event location. | N | `""` |
| method | integer (1,2,4,8,16,32) | ITIP method for group scheduling.<br>`1` = `PUBLISH` (organizer only uses this)<br>`2` = `REQUEST` (organizer only uses this)<br>`4` = `REPLY` (attendees only use this)<br>`8` = `CANCEL` (organizer only uses this)<br>`16` = `MOVE`<br>`32` = `COUNTER` (attendees only use this) | N | `1(PUBLISH)` |
| `mod` | integer | Specifies the recurrences to store/modify.<br>Not required for creating events.<br>Required to modify events.<br>One of the following values:<br>`1 = THISINSTANCE`<br>`2 = THISANDFUTURE`<br>`3 = THISANDPRIOR`<br>`4 = THISANDALL` | <br>N<br>Y | N/A |
| notify | integer (0,1) | This has been deprecated in iPlanet Calendar Server 5.x and remains only for iPlanet Calendar Server 2.x compatibility. The Group Scheduling Engine (GSE) module takes care of all email notifications.<br>A boolean indicating whether or not to notify attendees of a change to an event.<br>`1` = Notify attendees.<br>`0` = Do not notify attendees. | N | `0` |

**Table 7-35** `storeevents` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `orgEmail` | email address | Email address of the event contact (usually the organizer). | N | N/A |
| `orgUID` | `userid` | The `userid` of the organizer. | N | N/A |
| `priority` | integer (0-9) | Event priority. | N | `0` |
| | | `0` = lowest<br>`9`= highest | | |
| `rchange` | integer (`0`,`1`) | A boolean indicating whether or not to expand a recurring event. | N | `1` |
| | | `1` = expand<br>`0` = do not expand | | |
| `rdates` | semicolon-separated list of ISO 8601 DateTime Z strings | Event recurrence dates. | N | N/A |
| `relatedTos` | semicolon-separated list of quoted strings | Other events to which this event is related. | N | N/A |
| `resources` | semicolon-separated list of strings | The resources associated with the event. | N | N/A |
| `rid` | ISO 8601 DateTime Z string | Event recurrence identifier. | | N/A |
| | | Not required to create events. | N | |
| | | Required to modify events. | Y | |
| `rrules` | semicolon-separated list of strings | Event recurrence rules. A semicolon-separated list of recurrence-rule strings. | N | N/A |
| | | Each rule value must be enclosed in quotes. See "Recurrence Handling" on page 96. | | |
| `seq` | integer | Event sequence number. | N | `0` |

**Table 7-35** `storeevents` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `status` | integer | The event status code. One of the following values:<br><br>`0` `CONFIRMED`<br>`1` `CANCELLED`<br>`2` `TENTATIVE`<br>`3` `NEEDS_ACTION`<br>`4` `COMPLETED`<br>`5` `IN_PROCESS`<br>`6` `DRAFT`<br>`7` `FINAL` | N | N/A |
| `summary` | string | Event summary. A string of any length. | | |
| | | Required for new events. | Y | N/A |
| | | Not required for modifying events. | N | `default summary` |
| | | To include spaces in the string, use the code `%20`. | | |
| tzid | time zone ID string, such as "America/ Los_Angeles" | The timezone with which to interpret all dates passed in to, or returned by, the server. | N | `"GMT"` |
| | | The default is coordinated universal time (UTC or Z format). | | |
| `uid` | string | Unique identifier of the event to be stored. | | |
| | | System generated for new events. | N | N/A |
| | | Required to modify events. | Y | `default uid` |

**Description.**

This command creates or modifies events with the specified attributes and stores them in the specified calendar in the database.

The command creates and stores recurrences as specified by the `rrules`,`exrules`, `rid`, `mod`, and `rchange` parameters. See "Recurrence Handling" on page 96.

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

Use the `language` parameter to specify the language of the event. See "Choosing a Different Language or Character Set" on page 85 for a list of possible language codes.

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

#### Returns

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

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

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

#### Error Codes

This command cannot modify a linked event. The command will fail and return `CANNOT_MODIFY_LINKED_EVENTS(31)` in the `errno` array.

The command fails, and returns the error `STORE_FAILED_DOUBLE_BOOKED(40)`, when it tries to store an event in a time slot that is already scheduled (double booking).

#### Required Parameters

This command creates new events and modifies existing events. There is a different set of parameter requirements for both cases:

- To create new events requires only two parameters:
  - `dtstart`
  - `summary`

  Every other parameter is optional. The server generates the `uid`.
- To modify existing events requires three parameters:
  - `uid`
  - `rid`
  - `mod`

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

#### Duration

Specify the `duration` in 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 of 1 day
- `PT15M` represents a duration of 15 minutes

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

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

#### Example

For example, this URL would call storeevents.wcap and would result in storing an event in the calendar `john`,

```
http://webcalendarserver/storeevents.wcap?id=3423423asdfasf
&calid=john&dtstart=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
```

#### Group Scheduling - Attendee Parameter

The two most important parameters for creating a group-scheduled event are `attendee` and `method`. The `attendee` parameter is a semicolon separated list of attendee entries. The attendee entry is explained in the section below.

Each attendee entry may contain several parameters, such as invitation participation status, whether attendance is required or not, etc. All such parameters are encapsulated in a syntax very similar to the ATTENDEE property defined in the iCalendar Specification (RFC 2445). Reading the entire document is recommended in order to have the necessary background information to understand the WCAP attendee syntax. There are some differences, such as, WCAP uses a different delimiter, "^", to set apart these parameters. (WCAP uses the standard iCalendar semicolon delimiter for separating attendees.)

For example, where iCalendar would have the following:

```
PARSTAT=ACCEPTED;RSVP=TRUE:mailto:abc@xyz.com
```

WCAP would format it this way:

```
PARSTAT=ACCEPTED^RSVP^=TRUE^mailto:abc@xyz.com
```

### *Examples of WCAP Attendee Entries*

If attendee A (attA) accepts an invitation, the WCAP command would contain:

```
PARTSTAT=ACCEPTED^RSVP=TRUE^attA
```

If attendee B (attB) declines an invitation, the WCAP command would contain:

```
PARTSTAT=DECLINED^RSVP=TRUE^attB
```

If the email attendee jdoe@xyz.com has not yet decided to attend and is not required to respond, the WCAP command would contain:

```
PARTSTAT=NEEDS-ACTION^RSVP=FALSE:mailto:jdoe@xyz.com
```

Table 7-36 lists the parameters in the iCalendar ATTENDEE property understood by WCAP. Most of the parameters are optional. Not all are fully supported by Calendar Server, although the information will be stored. For group scheduling, only the PARTSTAT and RSVP parameters are relevant.

**Table 7-36** iCalendar ATTENDEE parameters understood by WCAP

| Parameters | Purpose |
|---|---|
| PARTSTAT | The only required parameter. This shows the attendees participation status. |
| CUTYPE | Calendar user type. |
| MEMBER | List of groups the attendee is part of. WCAP has no understanding of these groups. |
| ROLE | Role of the attendee in this meeting. |
| RSVP | Attendee response required or not. |
| DELEGATED-TO | To whom the attendee delegates attendance. |
| DELEGATED-FROM | Attendee is a delegate for this person. |
| SENT-BY | The calendar user acting on behalf of the specified user. |
| CN | Display name of attendee. |
| DIR | Directory entry reference. |
| LANG | Language of the entry. |

### Group Scheduling - Method Parameter

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

In an invitation, three types of messages may occur:

- An organizer invites attendees.

  When an organizer creates a meeting, there are two ways to invite people:

  - Send a `PUBLISH` message, creating or modifying a meeting, and notify the attendees. The `method` parameter is set to "1".
  - Send a `REQUEST` message, creating or modifying a meeting, and requesting a response to the invitation from attendees. The `method` parameter is set to "2".

  Only the organizer of the meeting can send a `PUBLISH` or `REQUEST` message.

- Attendees respond to invitation.

  An attendee sends a `REPLY` message, either accepting or declining the invitation. (The `method` parameter is set to "4".)

- Organizer cancels the meeting.

  When an organizer cancels a meeting, attendees are notified by sending a `CANCEL` using one of the `deleteevents` commands. The `method` parameter is set to "8".

**NOTE** The preferred way to handle a cancellation is to use one of the `deleteevents` commands, rather than `storeevents`.

The following set of examples demonstrates the WCAP commands for an organizer "org" to invite attendees "attA" and "attB" to a meeting. Attendee "attA" accepts the invitation; attendee "attB" declines it. The `uid` for the meeting is "event_u1". The event will be created on both attendees' calendars. Each will respond to the event on their own calendar. The response will be sent back to the organizer's calendar by the iPlanet Calendar Server Group Scheduling Engine.

The 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 acceptance:

```
storeevents.wcap?id=${SESSIONID ofattA}&calid=attA&uid=event_u1
   &method=4&attendees=PARTSTAT=ACCEPTED^RSVP=TRUE^attA
   &fmt-out=text/xml
```

The declined meeting:

```
storeevents.wcap?id=${SESSIONID ofattB}&calid=attB&uid=event_u1
   &method=4&attendees=PARTSTAT=DECLINED^RSVP=TRUE^attA
   &comments=I_cannot_make_it_Sorry&fmt-out=text/xml
```

## storetodos

### Purpose

Add one or more todos to a calendar.

### Parameters

Table 7-37 lists this command's the forty-four parameters:

**Table 7-37** `storetodos` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `alarmAudio` | ISO 8601Date Time Z 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 Z string | The time at which to run a flashing alarm. | N | N/A |
| `alarmPopup` | ISO 8601 Date Time Z string | The time at which to pop up a dialog alarm. | N | N/A |
| `alarmStart` | ISO 8601 DateTime Z string | The time at which to send an alarm notification of the todo. | N | N/A |
| `attachments` | semicolon-separated list of strings | This parameter exists to support iCalendar interoperability only. The strings are URLs. | N | N/A |
| `attendees` | semicolon-separated list of strings | The todo attendees. | N | N/A |

**Table 7-37** `storetodos` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `brief` | integer (0,1) | A boolean indicating whether or not to print out a brief version of the JavaScript output.<br><br>Applies only when output type (`fmt-out`) is JavaScript (`text/js`).<br><br>`1` = Brief output.<br>`0` = Complete output. | N | `0` |
| `calid` | string | Calendar identifier in which to store the todo. | Y | N/A |
| `categories` | semicolon-separated list of strings | The todo categories. | N | N/A |
| `completed` | ISO 8601 DateTime Z string | Completion date of the todo. A value of `0` means the todo is not yet completed. | N | `0` |
| `compressed` | integer (`0`,`1`) | For `compressed=0`, returns less data. Specifically, it does not return the following parameters:<br>`rrules`<br>`rdate`<br>`exrule`<br>`exdate`<br><br>For `compressed=1`, all recurrence data is returned.<br><br>This parameter can only be used when `fmt-out` is `text/xml`, or `text/calendar` | N | `0` |
| `contacts` | semicolon-separated list of strings | Contacts for the todo. | N | N/A |
| `desc` | string | Purpose of the todo. A string of any length.<br><br>If not passed, `desc` is set to the `summary` value.<br><br>To include spaces in the string, use the code `%20`. | N | Value in `summary` parameter. |
| `dtstart` | ISO 8601 DateTime string | Start time and date of the todo.<br><br>Not required to modify todos.<br><br>Required to create todos. | <br>N<br>Y | N/A |

**Table 7-37** `storetodos` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `due` | ISO 8601 DateTime Z 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 Z strings | Exclusionary recurrence dates of the todo. | N | N/A |
| `exrules` | semicolon-separated list strings | Todo exclusionary recurrence rules. A semicolon-separated list of recurrence-rule strings.<br><br>Each rule value must be enclosed in quotes. See "Recurrence Handling" on page 96. | N | N/A |
| `fetch` | integer (`0`,`1`) | A boolean indicating whether or not to fetch and return newly stored todos.<br><br>`1` = Fetch and return newly stored todos.<br>`0` = Do not fetch todos. | N | `0` |
| `fmt-out` | string | The format for the returned data.<br><br>The three format types:<br><br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |
| `geo` | two semicolon-separated floats | Semicolon-separated string of two float numbers representing the todo's geographical location (latitude and longitude).<br><br>For example, 37.31;-123.2. | N | `0;0` |
| `icsClass` | string | Todo class. One of the following values:<br><br>`PUBLIC`<br>`PRIVATE`<br>`CONFIDENTIAL` | N | `PUBLIC` |
| `icsUrl` | string | Todo URL. | N | "" |
| `id` | unique identifier string | The session identifier. | Y | N/A |

**Table 7-37** `storetodos` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `isAllDay` | integer (`0`,`1`) | A boolean indicating whether or not it is an all day todo.<br><br>`1` = An all day todo.<br>`0` = Not an all day todo. | N | `0` |
| `language` | string | The language of the todo. (For example, “en”, “fr”, “de”.) | N | N/A |
| `location` | string | Todo location. | N | “” |
| `method` | integer (1,2,4,8,16,32) | ITIP method for group scheduling.<br><br>One of the following:<br><br>`1` = `PUBLISH`<br>`2` = `REQUEST`<br>`4` = `REPLY`<br>`8` = `CANCEL`<br>`16` = `MOVE`<br>`32` = `COUNTER` | N | `1 (PUBLISH)` |
| `mod` | integer | Specifies the recurrences to store/modify.<br><br>Not required for new todos.<br><br>Required to modify todos.<br><br>One of the following values:<br><br>`1 = THISINSTANCE`<br>`2 = THISANDFUTURE`<br>`3 = THISANDPRIOR`<br>`4 = THISANDALL` | <br><br>N<br><br>Y | N/A |
| `notify` | integer<br><br>0,1 | This parameter has been deprecated in iPlanet Calendar Server 5.x. It remains to provide iPlanet Calendar Server 2.x compatibility. The Group Scheduling Engine (GSE) handles sending of email notifications.<br><br>A boolean indicating whether or not to notify attendees of a changed todo.<br><br>`1` = Notify attendees of the change.<br>`0` = Do not notify attendees. | N | `0` |
| `orgEmail` | email address | The email address contact for the todo. (Usually the organizer’s email.) | N | N/A |

**Table 7-37** `storetodos` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `orgUID` | `userid` | The `userid` of the organizer. | N | N/A |
| `percent` | integer (`0`-`100`) | Percentage completion of the todo. | N | `0` |
| `priority` | integer (`0`-`9`) | The priority of the todo. | N | `0` |
| | | `0` = Lowest priority.<br>`9`= Highest priority. | | |
| `rchange` | integer (0,1) | A boolean indicating whether or not to expand a recurring todo. | N | `1` |
| | | `1` = Expand the recurring todo.<br>`0` = Do not expand it. | | |
| `rdates` | semicolon-separated list of ISO 8601 TimeDate Z strings | Recurrence dates of the todo. | N | N/A |
| `relatedTos` | semicolon-separated list of quoted strings | Other todos to which this todo is related. | N | N/A |
| `resources` | semicolon-separated list of strings | The resources associated with the todo. | N | N/A |
| `rid` | ISO 8601 TimeDate Z string | Recurrence identifier of the todo. | | N/A |
| | | Not required for new todos. | N | |
| | | Required to modify todos. | Y | |
| `rrules` | semicolon-separated list of strings | Todo recurrence rules. A semicolon-separated list of recurrence-rule strings. | N | N/A |
| | | Each rule value must be enclosed in quotes. See "Recurrence Handling" on page 96. | | |
| `seq` | integer | Sequence number of the todo. | N | `0` |
| `status` | integer | A code for the status of the todo. One of the following values:<br>`0 CONFIRMED`<br>`1 CANCELLED`<br>`2 TENTATIVE`<br>`3 NEEDS_ACTION`<br>`4 COMPLETED`<br>`5 IN_PROCESS`<br>`6 DRAFT`<br>`7 FINAL` | N | N/A |

**Table 7-37** `storetodos` Parameters *(Continued)*

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `summary` | string | Todo summary. A string of any length. | | |
| | | Required for new todos. | Y | `default summary` |
| | | Not required for modifying todos. | N | N/A |
| | | To include spaces in the string, use the code `%20`. | | |
| `tzid` | time zone ID string, such as "America/Los_Angeles" | A timezone. All dates are interpreted with reference to this timezone. | N | "GMT" |
| | | If not passed, all dates are interpreted as being in coordinated universal time (UTC or Z format). | | |
| `uid` | string | Unique identifier of the todo to be stored. | | |
| | | System generated for new todos. | N | N/A |
| | | Required to modify todos. | Y | `default uid` |

**Description.**
Use this command to create and modifies todos with the specified attributes and stores them in the specified calendar in the database.

The command creates and stores recurrences as specified by `rrules`, `exrules`, `rid`, `mod`, and `rchange` parameters. See "Recurrence Handling" on page 96.

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

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

**`method` Parameter**
For group scheduling, specify one of the following ITIP methods:

| | | |
|---|---|---|
| `1` | `PUBLISH` | Used only by the organizer. |
| `2` | `REQUEST` | Used only by the organizer. |
| `4` | `REPLY` | Used only by attendees. |
| `8` | `CANCEL` | Used only by the organizer. |

| | | |
|---|---|---|
| 16 | MOVE | N/A |
| 32 | COUNTER | Used only by attendees. |

#### Returns

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

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

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

#### Error Codes

This command cannot modify a linked todo. The command will fail and return an error of `CANNOT_MODIFY_LINKED_TODOS(32)` in `errno`.

The command fails, and returns the error `STORE_FAILED_DOUBLE_BOOKED(40)`, when it tries to store a todo in a time slot that is already scheduled (double booking).

#### Required Parameters

This command creates new todos and modifies existing todos. There is a different set of parameter requirements for both cases:

- To create new todos requires only two parameters:
  - `dtstart`
  - `summary`

  Every other parameter is optional. The server generates the `uid`.
- To modify existing todos requires three parameters:
  - `uid`
  - `rid`
  - `mod`

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

#### Duration

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

Specify the duration in the 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).

## upload_file

### Purpose

Uploads a file to the server. Used in conjunction with `write_file` for file import.

### Parameters

Table 7-38 lists this command's one parameter:

**Table 7-38** `upload_file` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `idt` | unique identifier string | Session ID. | Y | N/A |

### Description.

This command used in conjunction with `write_file`, handles file importing for iPlanet Calendar Server 2.x compatibility. It only works through a `POST` command since a file is attached with it.

File import is handled in the following way through the UI:

- Upload the user's files from the user's computer to the server using the `upload_file` command. The `upload_file` command must be a `POST`.
- Take the uploaded file(s) and write them to the database using the `write_file` command.

The reason for this separation is that the UI cannot combine both parameters' values and file attachment through the JavaScript UI. Also, this separation allows for uploading multiple files to the server before actually writing them to the database.

### upload_file Example

The data sent to `upload_file` should look exactly like `import.wcap` data. The only files that can be supported are iCalendar and XML files. Although the user can upload any file, only iCalendar and XML files can be successfully added to the database.

The returned JavaScript from `upload.wcap` looks like the following:

```
Completed sending of import HTTP/1.1 200
Date: Tue, 05 Oct 2002 23:37:51 GMT
Content-type: text/html; charset=iso-8859-1
Content-length: 301
Last-modified: Tue, 05 Oct 2002 23:37:51 GMT
Pragma: no-cache
Expires: 0
Cache-Control: no-cache
Connection: Keep-Alive

<html><head><script>
function color(s) { if (s) document.bgColor=s
}
var
filename='s://ns//server//msg//calendar//core//parser//test//ical1.
ics'

var uniquefilename='cf8bfa37b3470000010000000ef000000'

var size=640
var errno=0
var errstr=''
</script></head>
<body onLoad=parent.uploadCB()>
</body></html>
```

### write_file Example

Notice the bold line containing the *uniquefilename* JavaScript variable in the previous example. This variable should be used as the input to the `uniquefilename` parameter in the `write_file` command.

After the UI has uploaded the appropriate files using `upload_file`, the UI should call `write_file` to actually write the file to the database.

For example, use this URL to write the previous example's upload call to the calendar `jdoe`. Notice that the `uniquefilename` parameter matches the *uniquefilename* JavaScript variable returned in `upload_file.wcap`.

```
http://webcalendarserver/write_file.wcap?id=abc&calid=jdoe&dtstart=
0&dtend=0&content-in=text/calendar&uniquefilename=cf8bfa37b34700000
1000000ef000000
```

To write two files to the server, put semicolons between the unique filenames. The files must be of the same content-type.

```
http://webcalendarserver/write_file.wcap?id=abc&calid=jdoe&dtstart=
0&dtend=0&content-in=text/calendar&uniquefilename=cf8bfa37b34700000
1000000ef000000;dg9cgb48c4580000010000000ab000000
```

## 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 7-41 lists this command's five parameters:

**Table 7-39** `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.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/calendar` |
| `id` | string | Uniquely identifies the session (session ID). | N* | `null` |
| `rid` | ISO 8601 DateTime string | Corresponding set of comma separated event `rid`s.<br>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 `uid`s. | 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 `rid`s were not zero (0), then the `uid`s and `rid`s are returned.

If the events are not found and the `rid`s were zero (0), then only the `uid`s 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=3b
d9e72f000027cf0000000600002113;3bd9e71700004503000000100002113;3bd
9e71700004503000000100002113;3bd9cd4700002206000000010000202d&rid=
0;20021027T230000Z;20021026T230000Z;0&fmt-out=text/calendar
```

```
BEGIN:VCALENDAR
PRODID:-//iPlanet/Calendar Hosting Server//EN
VERSION:5.1
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=3
bd9e72f000027cf0000000600002113;3bd9e71700004503000000100002113;3b
d9e71700004503000000100002113;3bd9cd4700002206000000010000202d&rid
=0;20021027T230000Z;20021026T230000Z;0&fmt-out=text/xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<iCalendar>
<iCal version="5.1" prodid="-//iPlanet/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 7-40 lists this command's five parameters:

**Table 7-40** `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.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/calendar` |
| `id` | string | Uniquely identifies the session (session ID). | N* | `null` |
| `rid` | ISO 8601 DateTime string | Corresponding set of comma separated todo `rid`s.<br>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 `uid`s. | 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 `rid`s were not zero (0), then the `uid`s and `rid`s are returned.

If the todos are not found and the `rid`s were zero (0), then only the `uid`s are returned.

### Example

```
verifytodos_by_ids.wcap?id=bo35r2pr3e5po35r&calid=jdoe&uid=3bde188f
0000472d0000000b00000399&rid=20021029T200200Z&fmt-out=text/xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<iCalendar>
<iCal version="5.1" prodid="-//iPlanet/Calendar Hosting Server//EN">
<TODO>
<UID>3bde188f0000472d0000000b00000399</UID>
<RECURID>20021029T200200Z</RECURID>
</TODO>
</iCal>
</iCalendar>
```

## version

### Purpose

To get the current WCAP version.

### Parameters

Table 7-41 lists this command's one parameter:

**Table 7-41** `version` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `fmt-out` | string | The format for the returned data.<br>The three format types:<br>`text/calendar`<br>`text/xml`<br>`text/js` | N | `text/js` |

**Description.**
This command gets the current WCAP version. (Note: this is different from the server version as well as the HTTP version.)

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

**Example**
The following examples are for each of output data types.

- JavaScript: (URL:`/version.wcap`)

```
<html><head><script>
function color(s) { if (s) document.bgColor=s
}
var id='0'
var userid=''
var calid=''
var errno=new Array()
var wcapversion=1.0.0
parent.ceCB(window.name)
</script></head></html>
```

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

```
BEGIN:VCALENDAR
VERSION:5.1
PRODID:-//iPlanet/Calendar Hosting Server//EN
X-NSCP-WCAPVERSION:WCAP
END:VCALENDAR
```

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

```
<?xml version="1.0" encoding="UTF-8">
<iCalendar>
<iCal>
<X-NSCP-WCAPVERSION>1.0.0</X-NSCP-WCAPVERSION>
</iCal>
</iCalendar>
```

# write_file

**Purpose**
Writes a file to the database.

### Parameters

Table 7-42 lists this command's six parameters:

**Table 7-42** `write_file` Parameters

| Parameter | Type | Purpose | Required | Default |
|---|---|---|---|---|
| `calid` | string | Command imports file to this calendar. | Y | N/A |
| `content-in` | string | The input-content data type. | Y | N/A |
| `dtend` | ISO 8601 DateTime "Z" string | The end range of events to import.<br>A value of `0` causes importing of all events up to the end of time. | N | `0` |
| `dtstart` | ISO 8601 DateTime "Z" string | The start range of events to import.<br>A value of `0` causes importing of all events from the beginning of time. | N | `0` |
| `id` | unique identifier string | Session ID. | Y | N/A |
| `uniquefilename` | semicolon-separated list of strings | The filenames to import from the server. | Y | N/A |

### Description.

The `write_file` command, used in conjunction with the `upload_file` command, handles importing files. The parameters correspond exactly to those of the import command. See `upload_file` for an example and a detailed description of the file import process.

# 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 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).

**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 5.1 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 iPlanet Directory Server or Netscape 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.

**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).

**server root** 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.

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

**Single Sign-on (SSO)** A 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



## A

## C

## D



## E

## F



## G



## I



## L

## T



## U



## V



## W