Sun Java System Calendar Server 6 2005Q4 Developer's Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–2434
October 2005

050923@13215

Contents

Tables

Figures

Preface

The *Sun Java™ System Calendar Server 6 2005Q4 Developer's Guide* gives detailed instructions on the use of the following Sun Java™ System Calendar Server 6 2005Q4 (Calendar Server) application program interfaces (API's) and a protocol that you can use to customize your server installation:

- Calendar Server Application Program Interface (CSAPI)

  Used to modify server functionality.
- Proxy Authentication SDK (authSDK)

  Plug-in for a portal authentication service.
- Web Calendar Access Protocol (WCAP)

  A protocol with commands used to access calendar services and data.

Topics covered in this preface include:

Who Should Use This Book

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

Before You Read This Book

This book assumes that you are a software engineer 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 RFC's describe in detail the format and definition for times, strings, parameters, and so forth used in WCAP commands.

  The RFC's can 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 an API, an SDK, and a protocol inside Calendar Server. For each interface, there is an overview chapter, followed by one or more reference chapters.

A list of the chapters follows:

- Part 1 Calendar Server API (CSAPI)
  - Chapter 1

    This API allows software engineers to customize server functionality in five areas:

    - Access Control
    - Authentication
    - Calendar Lookup
    - Data-Format Translation
    - User-Attribute Access
  - Chapter 2

    This chapter describes the CSAPI interfaces and their methods. Two types of interfaces exist: client and server.
- Part 2 Proxy Authentication SDK (authSDK)
  - Chapter 3

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

- Chapter 4

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

- Part 3 Web Calendar Access Protocol (WCAP)
  - Chapter 5

    This chapter gives an introduction to the WCAP protocol. WCAP is a command-based system for transmitting calendar data.
  - Chapter 6

    This chapter covers topics of common interest that span multiple commands.
  - Chapter 7

    This chapter details the individual commands.

Related Books

The following Calendar Server documents are available online in PDF and HTML formats:

- *Sun Java System Communications Services 2005Q4 Release Notes*
- *Sun Java System Communications Services 6 2005Q4 Documentation Center*
- *Sun Java System Communications Services 6 2005Q4 Deployment Planning Guide*
- *Sun Java System Calendar Server 6 2005Q4 Administration Guide*
- *Sun Java System Calendar Server 6 2005Q4 Developer's Guide*(this document)
- *Sun Java System Communications Services 6 2005Q4 Event Notification Service Guide*
- *Sun Java System Communications Services 6 2005Q4 Schema Reference*
- *Sun Java System Communications Services 6 2005Q4 Schema Migration Guide*
- *Sun Java System Communications Services 6 2005Q4 Delegated Administrator Guide*
- *Sun Java System Communications Express 6 2005Q4 Administration Guide*
- *Sun Java System Communications Express 6 2005Q4 Customization Guide*
- *Sun Java System Communications Sync 2005Q4 Release Notes*
- *Sun Java Enterprise System Technical Note: Sun Java System Calendar Frequently Asked Questions*
- *Sun Java Enterprise System Glossary*

In addition, the graphical user interface, Communications Express, has online help.

Related Third-Party Web Site References

Third-party URL's are referenced in this document and provide additional, related information.

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

Documentation, Software, Support, and Training

Sun Function	URL	Description
Documentation	http://www.sun.com/documentation (`http://www.sun.com/documentation`)	Download PDF and HTML documents, and order printed documents
Software	http://www.sun.com/software (`http://www.sun.com/software`)	Sun Software Gateway
Support	http://www.sun.com/support (`http://www.sun.com/support/`)	Obtain technical support, download patches
Training	http://www.sun.com/training (`http://www.sun.com/training`)	Learn about Sun courses

Typographic Conventions

The following table describes the typographic changes that are used in this book.

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. Perform a *patch analysis*. Do *not* save the file. [Note that some emphasized items appear bold online.]

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

PART I Calendar Server API (CSAPI)

This part covers the Calendar Server API's and contains the following chapters:

CHAPTER 1

Calendar Server API (CSAPI) Overview

This chapter gives an overview of the Calendar Server API (CSAPI), which is a set of high performance programmatic interfaces that enables you to modify or enhance the feature set of 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

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

A module is a shared object (`.so` file) on UNIX, or dynamic linked library (`.dll` file) on Windows. Each module that you provide must implement one or more of the CSAPI interfaces (or pure virtual base classes) defined in this document. For information about the CSAPI interfaces, see Chapter 2.

Each client-side interface addresses a functional area of Calendar Server. The implementation contained in a module can either augment or override the native Calendar Server functionality in its area.

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

You can use the default code for each of the CSAPI interfaces as a template to create your own plug-ins. The Calendar Server bundle contains all the default code, along with all supporting libraries and headers you need to implement your own plug-ins.

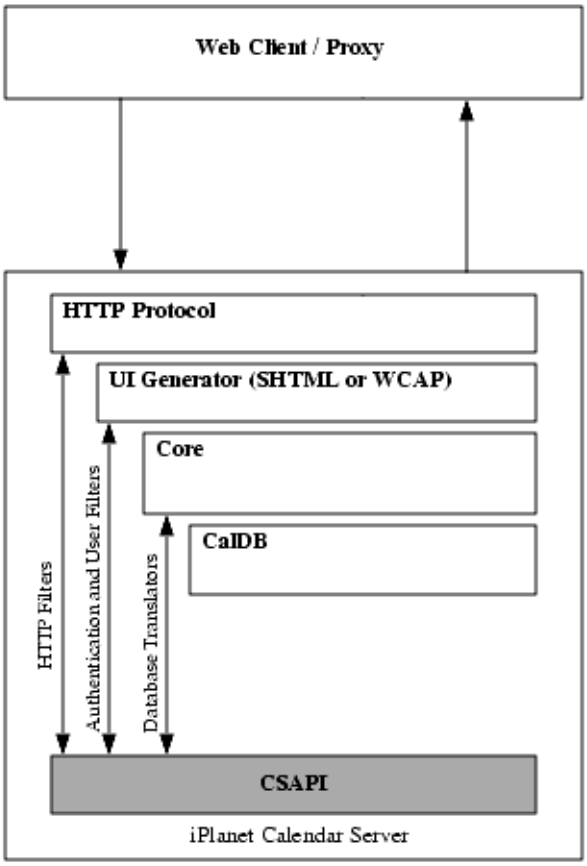

FIGURE 1–1 CSAPI Relationship to Other Subsystems

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 `www.mozilla.org`. For the URL, see the "Dependencies" on page 30 section that follows.

Dependencies

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

For documentation on NSPR see the Mozilla™ technical documentation site:

`http://www.mozilla.org/projects/nspr/reference/html/index.html`

For documentation on XPCOM, see:

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

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

Using CSAPI

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

Loading CSAPI Modules

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

The server checks `ics.conf` for the modules to be dynamically loaded at server startup. If the value of the preference `csapi.plugin.loadall` is `y`, the server loads all shared objects in the `cal/bin/plugins` directory whose names begin with the prefix `cs_`. Otherwise, if the value is `n`, various parameters exist for the various plug-ins. For more information on the preferences in `ics.conf`, see the *Sun Java System Calendar Server 6 2005Q4 Administration Guide*.

To specify the loading of a specific plug-in, `csapi.plugin.loadall` must be set to `n`. In addition, two parameters 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, to load only the calendar-lookup plug-in, the parameters are:

```
csapi.plugin.loadall = "n"
csapi.plugin.calendarlookup = "y"
csapi.plugin.calendarlookup.name = "calendarlookup"
```

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

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

Plug-in Interfaces

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

Plug-in Version Numbers

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

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

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

The following table lists the current version of each plug-in API.

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

Client and Server Plug-in API's

The CSAPI API's fall into two categories: client and server commands.

The interfaces are described in detail in Chapter 2

Client API's

The following table lists the CSAPI client API's, which can be implemented by one or more plug-ins.

Client API's	Description
"API: csIAccessControl" on page 38	Augments or overrides the default access control mechanism.
"API: csIAuthentication" on page 41	Augments or overrides the login authentication mechanism.
"API: csICalendarLookup" on page 46	Augments or overrides the default calendar lookup mechanism.
"API: csIDataTranslator" on page 53	Augments or overrides the format translation of incoming and outgoing data.
"API: csIPlugin" on page 57	Provides version control and descriptive information about the module.
"API: csIQualifiedCalidLookup" on page 61	Retrieves a calendar ID for the specified qualified URL.

Client API's	Description
"API: csIUserAttributes" on page 63	Augments or overrides the mechanism for storing and retrieving user attributes.

Server API's

The following is a list of the server API's, which can be implemented with one or more plug-ins.

Server API's	Description
"API: csICalendarServer" on page 68	Provides general server information, including version number.
"API: csIMalloc" on page 70	Allows access to server's memory allocation mechanism.

Required Initialization Routine

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.

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 Interface 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 1–1 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 Windows: On Solaris, it uses the `pam` library to authenticate against the `local /etc/passwd` file or NIS. On Windows, it uses the WIN32 API `LogonUser`, which authenticates against Microsoft clients. In order for this sample to work properly on Windows, the administrator must enable the privilege for users to log on using 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.

TABLE 1–1 CSAPI Interface Samples *(Continued)*

CSAPI Module Sample	Description
`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 API's are divided between client and server side.

Client and Server API's

Use the API's shown in the following tables to augment or override Calendar Server's default behavior:

TABLE 2–1 Client API's

"API: csIAccessControl" on page 38	Augments or overrides the access control mechanism.
"API: csIAuthentication" on page 41	Augments or overrides the login authentication mechanism.
"API: csICalendarLookup" on page 46	Augments or overrides the default calendar lookup mechanism.
"API: csIDataTranslator" on page 53	Augments or overrides the format translation of incoming and outgoing data.
"API: csIPlugin" on page 57	Provides version control and descriptive information about the module.
"API: csIUserAttributes" on page 63	Augments or overrides the mechanism for storing and retrieving user attributes.
"API: csIQualifiedCalidLookup" on page 61	Retrieves a calendar ID for the specified qualified URL.

TABLE 2–2 Server API's

"API: csICalendarServer" on page 68	Provides general server information, including version number.
"API: csIMalloc" on page 70	Allows access to server's memory allocation mechanism.

API: csIAccessControl

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

Methods

The csIAccessControl interface implements two methods:

"Method: CheckAccess" on page 38	Sets access control criteria for users.
"Method: Init" on page 40	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.

Method: CheckAccess

Purpose

Sets users' calendar access.

Syntax

```
PRUint32 CheckAccess (char* aUser,
                      char* aCalid,
                      PRInt32 *aAccessRequest,
```

```
PRInt32 *aAccessAllowed,
PRInt32 *aReturnCode)=0;
```

Parameters

The method has the following five parameters:

aUser	The authenticated user making the request. For anonymous access, pUserID is “anonymous”.
aCalid	calid for the calendar being accessed.
aAccessRequest	A set of bit flags representing the requested access type.
aAccessAllowed	Output parameter. A set of bit flags representing the allowed accesses. The method checks only the bits specified in aAccessRequest.
aReturnCode	On return, contains a constant that determines whether the server should continue with the default authentication procedure. One of the following constants: NS_CONTINUE_DEFAULT_PROCESSING NS_OVERRIDE_DEFAULT_PROCESSING

Returns

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

Description

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

For anonymous access, the user ID is “anonymous”.

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

Access Types	Bitmaps
ICS_ACCESSTYPE_NONE	0x00000000

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

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

Method: 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.

API: csIAuthentication

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

Methods

The csIAuthentication interface implements five methods:

"Method: ChangePassword" on page 42	Change a user's password.
"Method: Init" on page 43	Confirms that the interface was found and registered.
"Method: Logon" on page 44	Logs in a user.
"Method: Logout" on page 45	Logs out a user.

"Method: VerifyUserExists" on page 45	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 can 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.

Method: ChangePassword

Purpose

Changes the password for the specified user.

Syntax

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

Parameters

The method has the following four parameters:

`aUser`	The user's name.

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

Returns

On success, NS_AUTHENTICATION_CHANGEPASSWORD_SUCCESS.

On failure, NS_AUTHENTICATION_CHANGEPASSWORD_FAILURE.

Description

Changes the password of the specified user.

Method: 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.

Method: Logon

Purpose

Augment or override the authentication procedure for plain text login.

Syntax

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

Parameters

The method has the following three parameters:

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

Returns

On success, `NS_AUTHENTICATION_LOGON_SUCCESS`.

On failure, `NS_AUTHENTICATION_LOGON_FAILURE`.

Description

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

Method: Logout

Purpose

Logout a user.

Syntax

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

Parameters

The method has the following two parameters:

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

Returns

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

Description

None.

Method: VerifyUserExists

Purpose

Verify that the user ID is in the LDAP directory.

Syntax

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

Parameters

The method has the following two parameters:

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

Returns

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

Description

None.

API: csICalendarLookup

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

Methods

The `csICalendarLookup` implements the following methods:

Method	Description
"Method: DeleteHostnameForCalid" on page 47	Removes the host name entry from the calendar lookup database for a specific calendar.
"Method: FreeCalid" on page 48	Frees a previously allocated, fully qualified `calid`.
"Method: FreeType" on page 49	Frees a previously allocated type.

Method	Description
"Method: GetHostnameForCalid" on page 49	Gets the hostname from the calendar lookup database associated with the calendar specified by the `calid`
"Method: Init" on page 50	Confirms that the interface was found and registered.
"Method: QualifyCalid" on page 51	Returns the name of the qualified `calid`.
"Method: QueryType" on page 52	Queries the type of plug-in.
"Method: SetHostnameForCalid" on page 53	Sets the host name for a calendar user associated with a `calid` being created.

Description

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

piReturnCode Values

The four possible return codes for the `piReturnCode` parameter are as follows:

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

Method: DeleteHostnameForCalid

Purpose

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

Syntax

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

Parameters

The following are the parameters and definitions for this method:

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

Returns

Returns zero for success. Returns a non-zero code for failure. See "API: csICalendarLookup" on page 46.

Description

Removes the hostname associated with the specified calendar in the calendar lookup database.

Method: 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 does not continue if unsuccessful.

Returns

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

Description

Used by both implementations of lookup.

Method: 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 does not continue if unsuccessful.

Returns

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

Description

Frees the string allocated in `QueryType` method. Used by both implementations.

Method: GetHostnameForCalid

Purpose

Retrieves the hostname associated with the specified `calid`.

Syntax

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

Parameters

The following are the parameters and definitions for this method:

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

Returns

Returns zero for success. Returns a non-zero code for failure. See "API: csICalendarLookup" on page 46.

Description

Gets the hostname associated with a calendar in the calendar lookup database.

Method: 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.

Method: QualifyCalid

Purpose

Qualifies the relative calid.

Syntax

```
PRUint32 QualifyCalid(char* psCalid,
                      char** ppsQualifiedCalid,
                      PRInt32 *piReturnCode)=0;
```

Parameters

The method has the following parameters:

`psCalid`	The `calid` to be qualified.
`ppsQualifiedCalid`	On return, contains the URL of the qualified `calid`.

piReturnCode	0= successful, non-zero indicates failure.

Returns

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

Description

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

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

Method: 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 does 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 plug-in implements.

Method: SetHostnameForCalid

Purpose

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

Syntax

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

Parameters

This method uses the following parameters:

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

Returns

Returns zero for success. Returns any non-zero for failure.

Description

Sets the hostname for a `calid` that is about to be created.

API: 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:

"Method: GetSupportedContentTypes" on page 54	Informs the server about the content types that this database translator supports.
"Method: Init" on page 55	Confirms that the interface was found and registered.
"Method: Translate" on page 56	Translates calendar data to the specified MIME format.

Description

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

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

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

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

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

Method: 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 converts data to.
`aPreferredInContentType`	The content type the plug-in would prefer to receive. It must be one of the supported content types passed in the first parameter.

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

Returns

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

Description

Get the content type this database translator supports.

Method: 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.

Method: Translate

Purpose

Implement the translation from one content type to another.

Syntax

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

Parameters

The method has the following parameters:

`aInContentType`	The incoming content type.
`aOutContentType`	The outgoing content type.
`aInBuffer`	The input data buffer.
`aOutBuffer`	The output data buffer.
`aInSize`	The input buffer size.
`aOutSize`	The output buffer size.

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

Returns

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

Description

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

API: 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:

"Method: GetDescription" on page 58	Gets a textual description of what the plug-in does.
"Method: GetVendorName" on page 59	Gets a textual description of the vendor supplying this plug-in.
"Method: GetVersion" on page 60	Gets the major and minor version of the plug-in. This value must be greater than or equal to 1.0. For a list of the current version numbers for each plug-in API, see"Plug-in Version Numbers" on page 31.
"Method: Init" on page 60	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.

Method: 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.

Method: 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.

Method: 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.

Method: 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.

API: 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:

"Method: FindCalid" on page 62	Returns a calendar ID for the qualified URL.
"Method: Init" on page 50	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.

Method: FindCalid

Purpose

Finds the calendar ID for the URL specified.

Syntax

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

Parameters

The method has the following parameters:

`pQualifiedURL`	The URL to search on.
`ppCalidOut`	A pointer to the address of the `calid` found.
`piCalidSize`	Size of the `calid` returned.

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

Returns

The `calid` for the qualified URL passed in.

Description

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

Method: 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.

API: csIUserAttributes

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

Methods

The `csIUserAttributes` interface implements four methods:

"Method: FreeAttribute" on page 64	Free the memory used to store a retrieved attribute.
"Method: GetAttribute" on page 65	Retrieve an attribute value for a user.
"Method: Init" on page 66	Confirm that the interface was found and registered.
"Method: SetAttribute" on page 67	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.

Method: FreeAttribute

Purpose

Free the memory associated with your local attribute storage.

Syntax

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

Parameters

The method has the following two parameters:

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

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

Returns

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

Description

When you retrieve the value of an attribute using the `GetAttribute` method, the value is stored at a location that you have allocated, using the memory management technique of your choice. Use the "Method: FreeAttribute" on page 64 method to free that memory when it is no longer needed, using the same memory management technique. (See "API: csIMalloc" on page 70.)

Method: GetAttribute

Purpose

Retrieve an attribute value for a user.

Syntax

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

Parameters

The method has the following four parameters:

`aUser`	The name of the user.

aKey	The attribute key.
aValue	On return, this location contains a pointer to the retrieved attribute value.
aReturnCode	On return, contains a constant that determines whether the server should continue with the default, or with the override processing. One of the following constants: ■ NS_CONTINUE_DEFAULT_PROCESSING ■ NS_OVERRIDE_DEFAULT_PROCESSING

Returns

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

Description

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

Method: 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.

Method: SetAttribute

Purpose

Set an attribute value for a user.

Syntax

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

Parameters

The method has the following parameters:

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

Returns

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

Description

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

API: csICalendarServer

Provides server version information to a plug-in module.

Methods

The `csICalendarServer` interface implements two methods:

"Method: GetVersion" on page 68	Get the calendar server version.
"Method: Init" on page 69	Confirms that the interface was found and registered.

Description

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

Method: 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.

Method: 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.

API: csIMalloc

Allocates and frees memory.

Methods

The `csIMalloc` interface implements six methods:

"Method: Calloc" on page 70	Allocates and initializes memory for a number of objects.
"Method: Free" on page 71	Frees memory that is no longer in use.
"Method: FreeIf" on page 72	Frees memory, allowing a `NULL` pointer.
"Init" on page 72	Confirms that the interface was found and registered.
"Method: Malloc" on page 73	Allocates an amount of memory.
"Method: Realloc" on page 73	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.

Method: 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.

Method: 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 or C++ counterpart to free previously allocated memory.

Method: 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.

Method: 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 or C++ counterpart.

Method: 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 or C++ counterpart to reallocate memory that was previously allocated.

PART II Proxy Authentication SDK

This part covers the Proxy Authentication SDK and contains the following chapters:

- Chapter 3
- Chapter 4

CHAPTER 3

Proxy Authentication SDK Overview

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

Who Will Use the authSDK?

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

What Is the authSDK?

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

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

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

Architecture

The authSDK is pretty simple. It consists of initialization, lookup, and cleanup. Additionally, one other function, "Function: CEXP_SetHttpPort" on page 84, allows the authSDK to use a nonstandard port, and another, "Function: CEXP_GetVersion" on page 82, 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.

Initialization

Call "Function: CEXP_Init" on page 83 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 set up or tear down 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, "Function: CEXP_GenerateLoginURL" on page 81, when you want to generate a new session for a user. Lookup performs no authentication. It generates an entry in the session table for the user name and IP address, and returns a URL associated with that session.

If you pass the hostname of a calendar server, the function contacts that server to generate the session. If not, it queries the LDAP server to determine the host.

To generate a session without actually authenticating the user, you must provide the proxy administrator's ID and password.

Cleanup

If you are in a threaded environment, and you want to clean up resources, such as memory and open LDAP connections, use "CEXP_Shutdown" on page 84.

Functions Overview

The SDK consists of five functions in the SDK, as listed in the following table.

TABLE 3–1 Proxy Authentication SDK Functions

Function	Description
"Function: CEXP_GenerateLoginURL" on page 81	Generates a URL with the valid session ID.
"Function: CEXP_GetVersion" on page 82	Generates the version ID string.
"Function: CEXP_Init" on page 83	Initializes the SDK.
"Function: CEXP_SetHttpPort" on page 84	Specify the port over which you want to contact the calendar server.
"CEXP_Shutdown" on page 84	Performs all shutdown procedures, including freeing memory and shutting down connections.

CHAPTER 4

Proxy Authentication SDK Reference

This chapter describes the Calendar Server Proxy Authentication SDK (authSDK) API. The chapter contains a section for each of the following five authSDK functions and a section on "How to Use the authSDK" on page 85:

Function: 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

`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.

Function: CEXP_GetVersion

Purpose

Gets the version ID string.

Syntax

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

Parameters

None

Returns

A reference to the version ID string.

Function: 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

`pszLdapHost`	A string containing the hostname of the directory server.
`pszLdapMatchAttrib`	A string containing the attribute name. Used to match against the user name.
`pszLdapDN`	A string containing the base DN to search for user records. "DN", for Distinguished Name, is a string representation of an LDAP directory entry's name and location.
`iLdapPort`	An integer specifying the directory server's port number.
`pszLdapBindUser`	A string specifying the DN to bind as.
`pszLdapBindPass`	A string containing the password for the bind DN.
`pszAdminUser`	A string containing the Calendar Server administrator's LDAP user ID.
`pszAdminPassword`	A string containing the Calendar Server administrator's password.

Returns

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

Comment

If the bind DN (`pszLdapBindUser`) and password (`pszLdapBindPass`) are `NULL`, anonymous searching is attempted.

Function: CEXP_SetHttpPort

Purpose

Sets the HTTP port used to contact the calendar server.

Syntax

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

Parameters

`iHttpPort`	An integer specifying the port.

Returns

Nothing.

CEXP_Shutdown

Purpose

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

Syntax

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

Parameters

None

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, include the `expapi.h` header file in the calling code and link with the DLL or shared-object. On some platforms you might also be required to link with other system libraries the authSDK requires.
2. Authenticate your user with your portal authentication program.
3. Call "Function: CEXP_Init" on page 83.

   This function initializes the authSDK configuration information. This is necessary before any other authSDK function is called.
4. Optionally, call "Function: CEXP_SetHttpPort" on page 84.

   By default, the authSDK contacts the standard HTTP port, 80. Use this function to tell the authSDK to contact a nonstandard 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 "Function: CEXP_GenerateLoginURL" on page 81 call.

5. Call "Function: CEXP_GenerateLoginURL" on page 81.

   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 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, inside HTTP headers or JavaScript.™ The client then connects to Calendar Server, presenting the token as proof of identity.

6. Optionally, call "CEXP_Shutdown" on page 84.

   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 login, for example), but plug-ins using the API might want to reclaim resources and continue running.

Other Tips

The following is a list of 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.

PART III WCAP Protocol

This part covers the WCAP Protocol and contains the following chapters:

- Chapter 5
- Chapter 6
- Chapter 7

CHAPTER 5

Web Calendar Access Protocol Overview

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

Introduction

Calendar data is stored in a proprietary format in the various calendar databases. You retrieve calendar data using WCAP commands with the `fmt-out` parameter set to either `text/calendar` or `text/xml`.

Calendar Server communicates with Communications Express using the `text/xml` format.

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

WCAP commands consist of four general categories of usage:

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

Command Overview

The following is a list of commands supported in WCAP. For a detailed description of each command, see Chapter 7

TABLE 5–1 WCAP Command Overview

WCAP Command	Description
"Command: check_id" on page 134	Administrator only: Check if user's session ID is valid.
"Command: createcalendar" on page 136	Create a new calendar.
"Command: deletecalendar" on page 139	Delete an existing calendar.
"Command: deletecomponents_by_range" on page 140	Delete both events and todos in a calendar(s) over a specific time period.
"Command: deleteevents_by_id" on page 142	Delete events given a specific `calid` and `uid` or recurrence-ID pair.
"Command: deleteevents_by_range" on page 145	Delete events in a calendar(s) over a specific time period.
"Command: deletetodos_by_id" on page 148	Delete todos given a specific `calid` and `uid` or recurrence-ID pair.
"Command: deletetodos_by_range" on page 151	Deletes todos in a calendar(s) over a specific time period.
"Command: export" on page 152	Exports a calendar to a file.
"Command: fetchcomponents_by_alarmrange" on page 156	Queries for components that have alarms to trigger over a specific time period.
"Command: fetchcomponents_by_attendee_error" on page 164	Queries for components that had errors while sending group scheduling messages.
"Command: fetchcomponents_by_lastmod" on page 168	Queries for components that have changed, during the specified time range.
"Command: fetch_deletedcomponents" on page 185	Queries the `deletelog` database for deleted components.

TABLE 5–1 WCAP Command Overview *(Continued)*

WCAP Command	Description
"Command: fetchcomponents_by_range" on page 173	Queries for components over a specific time period, with filtering attributes.
"Command: fetchevents_by_id" on page 190	Queries for one or more events by a unique identifier (UID, Recurrence ID, modifier).
"Command: fetchtodos_by_id" on page 195	Queries for one or more todos by a unique identifier (UID, Recurrence ID, modifier).
"Command: get_all_timezones" on page 201	Returns all the time zones the server supports.
"Command: get_calprops" on page 205	Returns calendar properties.
"Command: get_freebusy" on page 208	Returns calendar free-busy time.
"Command: get_guids" on page 213	Returns a set of random UID's.
"Command: gettime" on page 214	Returns the server times for the requested `calid`s.
"Command: get_userprefs" on page 215	Returns user preferences and some server settings.
"Command: import" on page 219	Imports a calendar from a file to a user's calendar.
"Command: list" on page 222	Lists all calendars owned by a user.
"Command: list_subscribed" on page 223	Lists all calendars subscribed to by a user.
"Command: login" on page 224	Authenticates a user and redirects to first HTML view.
"Command: logout" on page 225	Terminates the current user's session and return to login screen.
"Command: ping" on page 226	Administrator only: Pings the calendar server.
"Command: search_calprops" on page 227	Searches for a calendar with the specified parameter values.
"Command: set_calprops" on page 230	Sets calendar properties.
"Command: set_userprefs" on page 235	Sets user preferences.
"Command: storeevents" on page 237	Stores events that are specified in application or URL encoded manner. For storing an even by passing properties in a URL.
"Command: storetodos" on page 246	Stores todos that are specified in the application or URL encoded manner.

TABLE 5–1 WCAP Command Overview *(Continued)*

WCAP Command	Description
"Command: subscribe_calendars" on page 254	Adds calendars to a users subscription list.
"Command: unsubscribe_calendars" on page 255	Removes calendars from a user's subscription list.
"Command: verifyevents_by_ids" on page 256	Fetches events and returns the `uid` or `rid` of events not in the database.
"Command: verifytodos_by_ids" on page 259	Fetches todos and returns the `uid` or `rid` of todos not in the database.
"Command: version" on page 260	Returns the WCAP version that the server supports.

Session Identifiers

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

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

You can customize the authentication mechanism to use a local or external authentication scheme, see "API: csIAccessControl" on page 38.

For more information about how to configure authentication, see chapter 10 in the Calendar Server Administration Guide: *Sun Java System Calendar Server 6 2005Q4 Administration Guide*.

Hosted (Virtual) Domain Mode

If you are using hosted (virtual) domains, all WCAP commands you issue must have fully qualified user ID's (`uid`) and calendar ID's (`calid`), for example `jdoe@example.com`.

In order to be in hosted domain mode, several parameters in the `ics.conf` file must be configured as specified in chapter 12 in the Calendar Server Administration Guide: *Sun Java System Calendar Server 6 2005Q4 Administration Guide*.

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

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

Non-hosted domain mode:

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

In hosted domain mode:

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

Command Formats

Plug-in architecture allows Calendar Server to support multiple command formats. Depending on your needs, you can use a variety of data formats for both clients and server.

WCAP uses HTTP, and follows the standards defined by the WC3 URL specifications.

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

Note – The number of characters that can be passed in for each parameter is limited to 1024 characters.

Client Request Formats

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

Command Format	Description
URI	Requests from client submitted using standard URI syntax.
HTML Form - `urlencoded`	Requests from client submitted as encoded URL's.
HTML Form - `text/xml`	Requests from client submitted using objects formatted as XML.

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

URI Format

Use the following format to submit a URI request:

`http://`*webcalendarserver*`/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 ID's, use the following:

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

See also Chapter 6

HTML Form

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

Note – The maximum length for WCAP parameters is 1024 characters.

Client Side Event Notification

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

When used with HTTP `GET`, commands are for data retrieval.

When used with HTTP `POST`, commands are for data modifications, including creation or deletion.

Server Response Formats

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

CHAPTER 6

WCAP Common Topics

This chapter contains topics of common interest that span multiple commands.

The topics are listed in alphabetical order. The table that follows lists and contains links to these topics:

Access Control Information

Access Control Information (ACI) determines access control for calendars. Access Control Entries (ACE strings) are individual strings of control characters that are stored in LDAP as one of a calendar's properties. There can be multiple ACE strings that apply to a single calendar. Collectively, all the ACE strings that apply to a calendar are called an Access Control List (ACL). When someone has requested access to a calendar, the system searches the calendar's ACL list to check for access rights. The system uses the first ACE encountered that either grants of denies access. 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 WCAP "Command: set_calprops" on page 230 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 can 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.

The "Command: get_userprefs" on page 215 command fails if any node does not have "allow anyone" access rights for reading and searching. For example, the following LDAP modify record for an ACL entry gives the correct privileges to make the command work correctly.

```
dn:  o=usergroup
changetype: modify
add: aci
aci: (targetattr="icscalendar ||cn||givenName||sn||uid||mail")
     (targetfilter=(objectClass=icscalendaruser))
     (version 3.0; acl "Allow calendar administrators to proxy-product=ics,
      class=admin,num=2,version=1" allow (proxy)
      groupdn="ldap:///cn=Calendar Administrators,ou=Groups,o=usergroup";)
```

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. The four types of classes for users are the following:

   - 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 only.

       Applies to calendar components, that is, events and todos. To indicate just the components, pass in the value `c`.

     - The calendar properties only.

       Applies to calendar properties, for example, display name, or 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 can be specified at the same time. To do this, the caller must pass in a string to indicate which bits to check.

   The table that follows lists the Access Control characters used in ACE strings. The third element contains a string with one or more of the Access Control characters.

Access Control Characters	Description
d	Grants the user delete access.
f	Grants the user free-busy access.
r	Grants the user read access.
s	Grants the user schedule access. This means that requests can be made, replies accepted, and other ITIP scheduling interactions honored.
w	Grants the user write access. This includes adding new items, deleting items, and modifying existing items.

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

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

   The ACE can either grant or deny access.

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

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

ACE Summary

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

who ^ flags ^ how ^ grant

Where:

- who = A string, type (str).
- flags = One of the characters `c`, `p`, or `a`.
- how = An access-string composed of one or more of the access control characters described earlier in "Access Control Information" on page 96.
- 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 the `jdoe` 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, free-busy, and read access to a calendar (*@domainname* `a sfr g`), to grant owners write and delete access to components only (`@@o c wd g`), to grant owners self-administration rights, and schedule, free-busy, and read access to both components and calendar properties (`@@o a zsfr g`), to deny `susan` all access to both components and calendar properties (`susan a zsfdwr d`), and to grant read access to all users (`@ c r g`), the entire command is:

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

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

Mapping User Interface Operations to ACL's

TABLE 6–1 Mapping User Interface Operations to ACL's

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

Application ID's (appid parameter)

The following WCAP commands accept the `appid` parameter:

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

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

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

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

If `appid` is present, the Event Notification Service (ENS) returns the value of `appid` as the value of the X-Token `X-NSCP-COMPONENT-SOURCE`. If the `appid` parameter is missing, ENS assigns the standard value to the X-Token (`WCAP`).

"Application ID's (appid parameter)" on page 99 shows the effect of the presence of the `appid` parameter on the value of the X-Token `X-NSCP-COMPONENT-SOURCE`. For more information about ENS, see the*Sun Java System Communications Services 6 2005Q4 Event Notification Service Guide*.

TABLE 6–2 Presence of appid and Value of X-Token `X-NSCP-COMPONENT-SOURCE`

appid Present?	Value of X-Token `X-NSCP-COMPONENT-SOURCE`
no	`WCAP`
yes	`appid`

Changing Language or Character Set

To insert a request for data to be returned in a language other than the system default, set either the `lang` or `charset` parameter. Note that the system default for language is now a server preference that you set in the `ics.conf` file. See the*Sun Java System Calendar Server 6 2005Q4 Administration 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 might 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 RFC's referenced in "Formatting Standards" on page 112.

Note – 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`does 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 or Simplified Chinese
zh_TW	Taiwanese

Note – This does not mean that all of these languages are currently supported by the server. Please check with your Sun Java Enterprise System representative to find out which languages are currently supported by the server.

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

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

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

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

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

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

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

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

Encoded Characters

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

=	%3D
&	%26
””	%22

The %*xx* string is the hexadecimal value of the character. For example, the `&` character is 26 in hexadecimal.

Error Handling

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

Error String

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

Error Codes

“Error Codes” on page 102 lists error codes returned by WCAP commands.

TABLE 6–3 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.

TABLE 6–3 Error Names, Values, and Meanings *(Continued)*

Error Name	Value	Meaning
CREATECALENDAR_ALREADY_EXISTS_FAILED	25	The command `createcalendar.wcap` failed. A calendar with that name already exists in the database.
SET_USERPREFS_FAILED	26	Command failed.
CHANGE_PASSWORD_FAILED	27	Command failed.
ACCESS_DENIED_TO_CALENDAR	28	Command failed. The user is denied access to a calendar.
CALENDAR_DOES_NOT_EXIST	29	Command failed. The requested calendar does not exist in the database.
ILLEGAL_CALID_NAME	30	`createcalendar.wcap` failed. Invalid `calid` passed in.
CANNOT_MODIFY_LINKED_EVENTS	31	`storeevents.wcap` failed. The event to modify was a linked event.
CANNOT_MODIFY_LINKED_TODOS	32	`storetodos.wcap` failed. The todo to modify was a linked todo.
CANNOT_SENT_EMAIL	33	Command failed. 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. The 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 unsupported format calendar data.
CALID_NOT_SPECIFIED	38	Calendar ID was not specified.
GET_FREEBUSY_FAILED	39	Command failed.

TABLE 6–3 Error Names, Values, and Meanings *(Continued)*

Error Name	Value	Meaning
STORE_FAILED_DOUBLE_BOOKED	40	If double booking is not allowed in this calendar, `storeevents` fails with this error when attempting to store an event in a time slot that was already filled. Double booking is not allowed under the following circumstances: ■ If the`ics.conf` parameter is set to:`user.allow.doublebook="yes"`, but the user's LDAP calendar property is set to: `doublebooking=0` (disallowed). The user's LDAP calendar properties override the `ics.conf` setting. ■ If the`ics.conf` parameter is set to: `user.allow.doublebook="no"`, the user's LDAP calendar properties have no effect.
FETCH_BY_ALARM_RANGE_FAILED	41	Command failed.
FETCH_BY_ATTENDEE_ERROR_FAILED	42	Command failed.
ATTENDEE_GROUP_EXPANSION_CLIPPED	43	An LDAP group being expanded was too large and exceeded the maximum number allowed in an expansion. The expansion stopped at the specified maximum limit. The maximum limit defaults to 200. To change the maximum limit, set the server configuration preference `calstore.group.attendee.maxsize`.
USERPREFS_ACCESS_DENIED	44	Either the server does not allow this administrator access to get or modify user preferences, or the requester is not an administrator.

TABLE 6–3 Error Names, Values, and Meanings *(Continued)*

Error Name	Value	Meaning
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 uid's and rid's).
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.)
DWP_CONNECTION_FAILED	48	GSE scheduling engine failed to make connection to DWP.
DWP_MAX_CONNECTION_REACHED	49	Reached the maximum number of connections. When some of the connections are freed, users can successfully connect. Same as error 11001.
DWP_CANNOT_RESOLVE_CALENDAR	50	Front end can't resolve to a particular back end. Same as error11002.
DWP_BAD_DATA	51	Generic response. Check all DWP servers. One might be down. Same as error 11003.
BAD_COMMAND	52	The command sent in was not recognized. This is an internal only error code. It should not appear in the error logs.
NOT_FOUND	53	Returned for all errors from a write to the Berkeley DB. This is an internal only error code. It should not appear in the error logs.
WRITE_IMPORT_CANT_EXPAND_CALID	54	Can't expand calid when importing file.

TABLE 6–3 Error Names, Values, and Meanings *(Continued)*

Error Name	Value	Meaning
GETTIME_FAILED	55	Get server time failed.
FETCH_DELETEDCOMPONENTS_FAILED	56	`fetch_deletedcomponents` failed.
FETCH_DELETEDCOMPONENTS_PARTIAL_RESULT	57	Success but partial result.
WCAP_NO_SUCH_FORMAT	58	Returned in any of the commands when supplied `fmt-out` is not a supported format.
COMPONENT_NOT_FOUND	59	Returned when a fetch or delete is attempted that does not exist.
BAD_ARGUMENTS	60	Currently used when attendee or organizer specified does not have a valid email address.
GET_USERPREFS_FAILED	61	`get_userprefs.wcap` failed. The following error conditions returns error code 61: ■ LDAP access denied ■ no results found ■ LDAP limit exceeded ■ LDAP connection failed
WCAP_MODIFY_NO_EVENT	62	`storeevents.wcap` issued with `storeytype` set to `2` (`WCAP_STORE_TYPE_MODIFY`) and the event doesn't exist.
WCAP_CREATE_EXISTS	63	`storeevents.wcap` issued with `storetype` set to `1` (`WCAP_STORE_TYPE_CREATE`) and the event already exists.
WCAP_MODIFY_CANT_MAKE_COPY	64	`storevents.wcap` issued and copy of event failed during processing.
STORE_FAILED_RECUR_SKIP	65	One instance of a recurring event skips over another
STORE_FAILED_RECUR_SAMEDAY	66	Two instances of a recurring event can't occur on the same day

TABLE 6–3 Error Names, Values, and Meanings *(Continued)*

Error Name	Value	Meaning
BAD_ORG_ARGUMENTS	67	Bad organizer arguments. orgCalid or orgEmail must be passed if any other org" parameter is sent. That is, orgUID can't be sent alone on a storeevents or a storetodos command if it is trying about to "create" the event or task. Note, if no "org" information is passed, the organizer defaults to the calid being passed with the command.
STORE_FAILED_RECUR_PRIVACY	68	Error returned if you try to change the privacy or transparency of a single instance in a recurring series.
LDAP_ERROR	69	For get_calprops, when there is an error is getting LDAP derived token values (X-S1CS-CALPROPS-FB-INCLUDE, X-S1CS-CALPROPS-COMMON-NAME).
GET_INVITE_COUNT_FAILED	70	Error in getting invite count (for get_calprops.wcap and fetchcomponents_by_range.wcap commands)
LIST_FAILED	71	list.wcap failed
LIST_SUBSCRIBED_FAILED	72	list_subscribed.wcap failed
SUBSCRIBE_FAILED	73	subscribe.wcap failed
UNSUBSCRIBE_FAILED	74	unsubscribe.wcap failed
ANONYMOUS_NOT_ALLOWED	75	Command cannot be executed as anonymous. Used only for list.wcap, list_subscribed.wcap, subscribe.wcap, and unsubscribe.wcap commands.

TABLE 6–3 Error Names, Values, and Meanings *(Continued)*

Error Name	Value	Meaning
ACCESS_DENIED	76	Generated if a non-administrator user tries to read or set the calendar-owned list or the calendar-subscribed list of some other user, or if the option is not turned on in the server
BAD_IMPORT_ARGUMENTS	77	Incorrect parameter received by `import.wcap`
READONLY_DATABASE	78	Database is in read-only mode. (returned for all attempts to write to the database)
ATTENDEE_NOT_ALLOWED_TO_REQUEST_ON_MODIFY	79	Attendee is not allowed to modify an event with `method=request`.
TRANSP_RESOURCE_NOT_ALLOWED	80	Resources do not permit the `transparency` parameter.
RECURRING_COMPONENT_NOT_FOUND	81	Recurring component not found. Only happens when `recurring=1` is passed in by fetch commands. This code is returned if part of the recurring series (either the master or an exception) is missing.
CDWP_ERR_MAX_CONNECTION_REACHED	11000	Maximum connections to back-end database reached. As connections are freed up, users can connect to the back-end.
CDWP_ERR_CANNOT_CONNECT	11001	Cannot connect to back-end server. Back-end machine might be down or DWP server is not up and running.
CDWP_ERR_CANNOT_RESOLVE_CALENDAR	11002	Front-end can't resolve calendar to a particular back-end server.
CDWP_ERR_BAD_DATA	11003	Bad data received from DWP connection. This is a generic formatting error. Check all DWP servers. One might be down.

TABLE 6–3 Error Names, Values, and Meanings *(Continued)*

Error Name	Value	Meaning
CDWP_ERR_DWPHOST_CTX_DOES_NOT_EXIST	11004	For the back-end host, context doesn't exist in the context table. Solution is to add back-end host to ics.conf and restart front-end.
CDWP_ERR_HOSTNAME_NOT_RESOLVABLE	11005	DNS or NIS files, or hostname resolver is not set up properly or machine does not exist.
CDWP_ERR_NO_DATA	11006	No data was received from reading the calendar properties from the DWP connection.
CDWP_ERR_AUTH_FAILED	11007	DWP authentication failed.
CDWP_ERR_CHECKVERSION_FAILED	11008	DWP version check failed.

Fetching Component Data

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

This parameter is found in all the fetchcomponents_by_* commands.

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

Fetching Component State Data

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

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

The following table lists component state values. A component state pertains either to the attendee or the organizer.

TABLE 6–4 Component State Values

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

Fetching Deleted Data

If you have deleted component data and need to reconstruct it, you can use the "Command: fetch_deletedcomponents" on page 185 command, which causes the system to process the Delete Log Database (`ics50deletelog.db` in the `csdb` directory). However, it is not possible to recreate the entire component data since not all of it is logged.

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

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

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

For more information on the Delete Log database, see the .

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 `rrules`, `rdates`, `exrules`, and `exdates` properties as the default. To receive all the recurrence data back from the following commands, use `compressed=1`.

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

Note – `compressed` has been deprecated for the current release, and is included for backwards compatibility only. It can be removed from the future releases.

Formatting Standards

Find the exact format and definition for all times, strings, parameters, and so forth, by referring to RFC 2445, RFC 2446, and RFC 2447. Unless otherwise noted, all WCAP commands follow these specifications.

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

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

Note – As a reminder, the maximum parameter value length is 1024 characters.

For more information on time zones, see "Time Zones" on page 127which follows later in this section.

Free-busy Calendars

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

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

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

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

  If `fbinclude=1`, which allows the use of the calendar for free-busy calculations, then the `acl` access rights are used to determine if the user can see the details of the calendar, the free-busy representation only, or neither.

In free-busy calendars, instead of event or todo details, just the word "Busy" appears by the time block. Blocks of time without any scheduled events are listed also, with the word "Free" next to them.

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

10:00-11:00	first meeting
12:00-1:00	lunch
3:00-4:00	second meeting

If user `john` has only free-busy access to the calendar `jdoe`, user `john` gets only a free-busy version of the calendar. The free-busy time for `jdoe` (from 9:00 to 6:00) would appear as the following:

9:00-10:00	Free
10:00-11:00	Busy
11:00-12:00	Free
12:00-1:00	Busy
1:00-3:00	Free
3:00-4:00	Busy
4:00-6:00	Free

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

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

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

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

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

Free-busy Calculation for Private Events

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

Using either `storeevents` or `storetodos`, to keep the event private, set the event or todo parameter to `transparent=1`. Set it to `transparent=0` to allow the event or todo to be included in the free-busy calculation.

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

Group Scheduling

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

Attendee Parameter

Each attendee entry can contain several parameters, such as invitation participation status, whether attendance is required or not, and so forth. 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. Some differences exist, such as, WCAP uses a different delimiter, "^", to set apart these parameters. (However, WCAP uses the standard iCalendar semicolon delimiter for separating attendees.)

For example, where iCalendar would have the following:

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

WCAP would format it this way:

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

Examples of WCAP Attendee Entries

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

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

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

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

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

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

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

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

The following table 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 is stored. For group scheduling, only the PARTSTAT and RSVP parameters are relevant.

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.

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

Method Parameter

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

For group scheduling, specify one of the following ITIP methods:

1	PUBLISH	Used only by the organizer.
2	REQUEST	Used only by the organizer.
4	REPLY	Used only by attendees.

8	CANCEL	Used only by the organizer.

In addition to these ITIP methods, there is another method used by Calendar Server internally (a non-ITIP method):

256	UPDATE	Used by attendee to update only the attendee's copy of a group scheduled event. Does not affect anyone else's calendar data.

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

In an invitation, three types of messages can occur:

- An organizer invites attendees.

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

  - Send a `PUBLISH` message, creating or modifying a meeting. The `method` parameter is set to "1". Note that everything except the attendee information is sent. (Attendees can see the event but not the other attendees.)
  - Send a `REQUEST` message, creating or modifying a meeting, and requesting a response to the invitation from attendees. The `method` parameter is set to "2".

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

  Attendees respond to invitation.

  An attendee sends a `REPLY` message, either accepting or declining the invitation. (The `method` parameter is set to "4".)
- Organizer cancels the meeting.

  When an organizer cancels a meeting, attendees are notified by sending a `CANCEL` using one of the `deleteevents` commands. The `method` parameter is set to "8".

Note – The preferred way to handle a cancellation is to use one of the `deleteevents` commands, rather than `storeevents`.

The following set of examples demonstrates the WCAP commands for an organizer "org" to invite attendees "attA" and "attB" to a meeting. Attendee "attA" accepts the invitation. Attendee "attB" declines the invitation. The `uid` for the meeting is "event_u1". The event is created on both attendees' calendars. Each responds to the event on their own calendar. The response is sent back to the organizer's calendar by the Calendar Server Group Scheduling Engine.

The following is an example of an invitation:

```
storeevents.wcap?id=${SESSIONID of org}&calid=org
                 &dtstart=20020201T200200Z
                 &dtend=20020201T210000Z
                  summary=invite_attA_attB
                 &method=2
                 &attendees=PARTSTAT=ACCEPTED^RSVP=TRUE^org;
                            PARTSTAT=NEEDS-ACTION^RSVP=TRUE^attA;
                            PARTSTAT=NEEDS-ACTION^RSVP=TRUE^attB
                 &fmt-out=text/xml
```

The following is an example of the acceptance:

```
storeevents.wcap?id=${SESSIONID ofattA}&calid=attA
                 &uid=event_u1
                 &method=4
                 &attendees=PARTSTAT=ACCEPTED^RSVP=TRUE^attA
                 &fmt-out=text/xml
```

The following is an example of a declined meeting:

```
storeevents.wcap?id=${SESSIONID ofattB}&calid=attB
                 &uid=event_u1
                 &method=4
                 &attendees=PARTSTAT=DECLINED^RSVP=TRUE^attA
                 &comments=I_cannot_make_it_Sorry
                 &fmt-out=text/xml
```

Output Format

WCAP commands can request the output format in two content types: text/calendar and text/xml.

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

Recurring Components– Overview

Recurrence handling occurs as follows:

- A recurring series of events or todos has a master entry plus entries for exceptions.
- Changing the `rrules` of a single instance returns an error. When `rrules` are modified for a recurring series, the whole series is deleted and recreated.
- Changing `dtstart` of a recurring series entry causes the whole series to be recreated with the new `dtstart`, thereby losing all exceptions.
- Inserting a `rid` that was not part of the original rule is not supported.
- Multiple `rrules` for any component are not supported.

Recurring Components– Creating, Modifying

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

- "rrules" on page 120 - Semicolon-separated list of quoted recurrence-rule strings for recurring events.
- "rdates" on page 121– Semicolon-separated list of ISO 8601 date strings listing recurrence dates.
- "exrules" on page 121– Semicolon-separated list of quoted recurrence-rule strings for dates to exclude.
- "exdates" on page 122– Semicolon-separated list of ISO 8601 date strings listing dates to exclude.
- "rid" on page 122– ISO 8601 Date-Time String giving the recurrence ID of an event.
- "mod" on page 123– Modifier telling which instances of the event to store.
- "rchange" on page 123– A boolean specifying whether or not to replace the `rrules` parameter. If `rchange=1`, the store commands map all `mod` settings (`2-4`) are mapped to `4`. This means that you can not change an `rrules` for only some of the components in a recurring series. To change the `rrules` parameter, all components in the recurring series must be changed.
- "excludedtstart" on page 124– An integer specifying whether or not to include the `dtstart` date in a recurring series if the date does not follow the `rrule`.

Note – The `rrules`, `rdates`, `exrules`, and `exdates` parameters always function in replace mode. That is, no matter what the `replace` parameter value is set to, the values passed in for the recurrence parameters always replace the old parameter values, rather being appended to them.

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

For more information on the `replace` parameter, see "Updating Parameter Values" on page 128

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. Many parameters are possible for recurrence rules. (See RFC 2445 for a complete description of the syntax.)

Three parameters used by Calendar Server for specifying recurrence are `freq`, `count` and `until`:

- 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 repeats. 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 `until` parameter in a rule specifies using an end date as opposed to using the `count` to limit the number of instances created. Instances are created up to the end date or until 60 instances are created, whichever occurs first.

In the event that neither the `count` nor the `until` parameter are specified, the default is 60 instances.

Note – Using the `storeevents.wcap` command to create an event with only `exdates` or `rdates` values, without specifying an `rrules` results in no events being created. The same behavior can be observed with the `storetodos.wcap` command.

The following example shows an `rrules` parameter that specifies the event is to occur daily for 10 instances (`COUNT=10;FREQ=DAILY`):

```
rrules="count%3D10%3Bfreq%3Ddaily"
```

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

```
http://webcalendarserver/storecomponents.wcap
              ?id=b5q2o8ve2rk02nv9t6
              &calid=jdoe
              &uid=333
              &dtstart=20020301T112233Z
              &rrules="count%3D10%3Bfreq%3Ddaily"
              &dtend=20020301T112233
              &summary=uuuu
```

rdates

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

For example, the following `rdates` parameter specifies a recurring event with two recurrence dates (3/31/02 11:22:33 and 5/31/02 11:22:33):

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

The following example URL passes the `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"
```

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 (`COUNT=10;FREQ=DAILY` and `FREQ=WEEEKLY;COUNT=4`).

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%3Ddaily"
              &dtend=20020301T112233
              &summary=uuuu
```

rid

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

For example:

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

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

mod

When modifying recurring components, this parameter specifies whether to apply the changes to one or more instances of the event or todo. The following settings are mapped, but currently only the values `1` and `4` are honored. If `2` or `3` are specified, in certain cases they are mapped to `4`.

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

When creating or modifying a recurring component, if changing `rrules` is allowed (`rchange` is set to `1`), the system assumes a setting of 4, which causes the entire series of events or todos to be deleted and rewritten. If `2` or `3` are specified when trying to change rules or start times, they are mapped to `4`. If `2` or `3` is specified for changing a summary or description, the setting are honored, but no exceptions are created for errors.

rchange

The `rchange` parameter specifies whether recurrences are expanded in "Command: storeevents" on page 237, and "Command: storetodos" on page 246. Normally, events and todos calendar components are not expanded, so the parameter defaults to `0`, which implies the series is recreated.

However, you might not want to expand recurrences when you are modifying multiple events. For example, suppose a meeting recurs every Friday starting Jan. 1, 2002. Use the following URL to change the summary of each event after Feb. 1, 2002 to `changed-event`.

The following example sets the `rchange` parameter to 0, to make the modification without adding additional events:

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

Note that when you are modifying a recurrence series, do not pass in `rrules` unless you are trying to recreate the series with a new rule.

excludedtstart

When creating a recurring series according to the `rrule`, this integer specifies whether to include the `dtstart` date if the date does not follow the `rrules`. For example, if on a Monday, you were creating a recurring series of meetings that were to be held every Wednesday, the `dtstart` would be Monday, but that does not fit the set of dates (all Wednesdays) generated using the `rrules`. Therefore the server must decide whether to include the `dtstart` date or not based on the value of `excludedtstart`.

A value of `0` indicates the `dtstart` date is included in the recurring series and a value of `1` indicates the `dtstart` date is not included in the recurring series. The default is `0`.

Recurring Components–Deleting

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

Use the `mod` parameter to select which occurrences to delete:

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

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

Examples Using deleteevents_by_id

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

```
http://webcalendarserver/deleteevents_by_id.wcap
              ?id=23423423434abc
              &calid=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://webcalendarserver/deleteevents_by_id.wcap
              ?id=23423423434abc
              &calid=jdoe
              &uid=001
              &rid=20020301T112233Z
              &mod=2
```

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

For example, this URL would delete the event that occurs on the date March 1, 200211:22:33 AM GMT and all prior instances of this event (`uid 001`).

```
http://webcalendarserver/deleteevents_by_id.wcap
              ?id=23423423434abc
              &calid=jdoe
              &uid=001
              &rid=20020301T112233Z
              &mod=3
```

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

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

Recurring Components– Fetching

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

- `compressed`– A boolean specifying whether to return all of the recurring entry's data, or to exclude the following parameters:`rrules`, `rdates`, `exrules`, `exdates`.

  Note – This parameter is deprecated in the current release, and is included only for backwards compatibility. It might be removed from future releases.

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

Sorting Order of Returned Events and Todos

Fetch commands that support the `fetchorder` parameter allow you to specify the order in which the events and todos are returned.

Three sorting choices can be specified:

- Ascending order
- Descending order
- Special (or legacy) sorting order–This sorting choice returns things in mostly ascending order.

The sorting logic is as follows:

- For events, put in order according to `dtstart`. If two events have the same `dtstart` date, then compare `dtend` for the two events. Return the one that fits the desired sort order. If they are the same, compare `uid` and return the components from the least value to the highest.
  - For todos, all completed tasks are returned after all sorted not-due tasks. For two not-due tasks, compare `dtdue`. Return in the desired sort order. If they are the same, compare `uid` and return components from the least value to the

highest. For descending sorts, some overdue tasks could be truncated because of the `maxResults` limit.

- For recurring components sorted in ascending order, the master record is returned first and then the exceptions. Depending on the `maxResults` limit, some exceptions could be truncated. In this case, error `81` is given. (See "Error Handling" on page 102.)
- For recurring components sorted in descending order, exceptions are returned first and then the master record. Depending on the `maxResults` limit, it is possible for a master record to be truncated. In this case, error `81` is given. (See "Error Handling" on page 102.)

Time Zones

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

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

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

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

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

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

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

The following commands use both the `tzid` and `tzidout` parameters:

- "Command: fetchcomponents_by_alarmrange" on page 156
- "Command: fetchcomponents_by_lastmod" on page 168
- "Command: fetchcomponents_by_range" on page 173
- "Command: fetchevents_by_id" on page 190
- "Command: fetchtodos_by_id" on page 195
- "Command: storeevents" on page 237
- "Command: storetodos" on page 246

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

- "Command: deleteevents_by_id" on page 142
- "Command: deletetodos_by_id" on page 148
- "Command: get_freebusy" on page 208
- "Command: set_calprops" on page 230

Updating Parameter Values

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

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

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

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

X-Tokens

TABLE 6–5 X-Tokens Returned by WCAP Commands

Token Name	Type	WCAP Command
X-NSCP-ATTENDEE-GSE-STATUS	integer	all fetch commands
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY	string	all fetch commands
X-NSCP-CALPROPS-CALMASTER	string	all fetch commands
X-NSCP-CALPROPS-CATEGORIES	string	all fetch commands
X-NSCP-CALPROPS-CHARSET	string	all fetch commands
X-NSCP-CALPROPS-CHILDREN	string	all fetch commands
X-NSCP-CALPROPS-CREATED	string	all fetch commands
X-NSCP-CALPROPS-DESCRIPTION	string	all fetch commands
X-NSCP-CALPROPS-LANGUAGE	string	all fetch commands
X-NSCP_CALPROPS-LAST-MODIFIED	string	all fetch commands
X-NSCP-CALPROPS-NAME	string	all fetch commands
X-NSCP-CALPROPS-OWNERS	string	all fetch commands
X-NSCP-CALPROPS-PARENT-CALID	string	all fetch commands
X-NSCP-CALPROPS-PRIMARY-OWNER	string	all fetch commands
X-NSCP-CALPROPS-READ	integer	all fetch commands
X-NSCP-CALPROPS-RELATIVE-CALID	string	`get_freebusy`, all fetch commands
X-NSCP-CALPROPS-RESOURCE	string	all fetch commands
X-NSCP-CALPROPS-TZID	string	all fetch commands
X-NSCP-CALPROPS-WRITE	integer	all fetch commands
X-NSCP-CHARSET	string	all fetch commands
X-NSCP-DTEND-TZID	string	all fetch commands
X-NSCP-DTSTART-TZID	string	all fetch commands
X-NSCP-DUE-TZID	string	all fetch commands
X-NSCP-GSE-COMMENT	string	all fetch commands
X-NSCP-GSE-COMPONENT-STATE	string	all fetch commands

TABLE 6–5 X-Tokens Returned by WCAP Commands *(Continued)*

Token Name	Type	WCAP Command
X-NSCP-GUID	GUID integer	get_guids
X-NSCP-LANGUAGE	string	all fetch commands
X-NSCP-LINK-CALID	string	all fetch commands
X-NSCP-ONGOING	integer	all fetch commands
X-NSCP-ORGANIZER-EMAIL	string	all fetch commands
X-NSCP-ORGANIZER-SENT-BY-UID	string	all fetch commands
X-NSCP-ORGANIZER-UID	string	all fetch commands
X-NSCP-ORIGINAL-DTSTART	Date Time Z string	fetch commands
X-NSCP-REQUEST-STATUS	string	deleteevents_by_range, deletetodos_by_range
X-NSCP-TOMBSTONE	integer	all fetch commands
X-NSCP-WCAP-CALENDAR-ID	string	
X-NSCP-WCAP-ERRNO	integer	all
X-NSCP-WCAP-PREF-CN X-NSCP-WCAP-PREF-GIVENNAME X-NSCP-WCAP-PREF-PREFERREDLANGUAGE X-NSCP-WCAP-PREF-SN X-NSCP-WCAP-PREF-NSWCALCALID X-NSCP-WCAP-PREF-CEFONTSIZEDELTA X-NSCP-WCAP-PREF-CETABLESPACING X-NSCP-WCAP-PREF-CECOLORSET X-NSCP-WCAP-PREF-CEBGCOLOR X-NSCP-WCAP-PREF-CECALLIST X-NSCP-WCAP-PREF-CEAGENDALIST X-NSCP-WCAP-PREF-CEAGENDATZMODE_PERSONAL X-NSCP-WCAP-PREF-CECURSORCOLOR	varies by preference	get_userprefs

TABLE 6–5 X-Tokens Returned by WCAP Commands *(Continued)*

Token Name	Type	WCAP Command
X-NSCP-WCAP-SERVER-PREF-ALLOWCHANGEPASSWORD X-NSCP-WCAP-SERVER-PREF-ALLOWCREATECALENDARS X-NSCP-WCAP-SERVER-PREF-ALLOWDELETECALENDARS	varies by preference	get_userprefs
X-NSCP-WCAP-SESSION-ID	string	check_id
X-NSCP-WCAP-USER-ID	string	
X-NSCP-WCAP-VERSION	string	
X-S1CS-ATTENDEE-EXDATELIST	string	all fetch commands
X-S1CS-ATTENDEE-RDATELIST	string	all fetch commands
X-S1CS-CALID	string	all fetch commands
X-S1CS-CALPROPS-ALLOW-DOUBLEBOOKING	string	get_calprops set_calprops
X-S1CS-CALPROPS-COMMON-NAME	string	get_calprops
X-S1CS-CALPROPS-FB-INCLUDE	integer	get_calprops
X-S1CS-CALPROPS-INVITATION-COUNT	string	get_calprops
X-S1CS-CALPROPS-OWNED-CALENDAR	string	list
X-S1CS-CALPROPS-SUBSCRIBED-CALENDAR	string	list_subscribed
X-NSCP-COMPONENT-SOURCE	string	all fetch commands
X-S1CS-EMAIL	string	all fetch commands
X-S1CS-EXPORTVERSION	string	all fetch commands
X-S1CS-RECURRENCE-COUNT	integer	all fetch commands
X-S1CS-RECURRENCE-EXDATELIST	string	all fetch commands
X-S1CS-RECURRENCE-RDATELIST	string	all fetch commands
X-S1CS-RECURRENCE-UNTIL	string	all fetch commands
X-NSCP-TRIGGERED_BY	string	all fetch commands
X-S1CS-TZID-ALIAS	string	all fetch commands

CHAPTER 7

WCAP Command Reference

This chapter contains the WCAP command reference. Each command accepts various parameters, which are defined for each command in this chapter.

Unless otherwise noted, the maximum length value for any parameter accepted by WCAP commands is 1024 characters. While no input length checking is performed by WCAP, any parameter value longer than 1024 can produce unpredictable results.

For all commands that allow the `id` parameter (session ID), it is a required parameter. Two exceptions to this rule are as follows:

- You do not need it 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.

The following is a list of the available WCAP commands:

Command: check_id

Purpose

This administrator only command allows the administrator to verify that a session is still valid.

Parameters

TABLE 7–1 `check_id` Parameters

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

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

Purpose

This command allows the administrator to verify that the session is still valid.

Returns

The server returns the property `X-NSCP-WCAP-CHECK-ID`. If the value of this property is `1` the session is valid. If a zero (`0`) is returned, the session is invalid. It has either timed out or is unrecognized.

Example

The following command returns whether the specified session is valid or not:

```
http://calendarserver/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:-//SunJavaSystem/Calendar Hosting Server//EN
 METHOD:PUBLISH
```

```
VERSION:6.0
X-NSCP-WCAP-CHECK-ID:1
END:VCALENDAR
```

Command: createcalendar

Purpose

Create a new calendar.

Parameters

TABLE 7–2 `createcalendar` Parameter

Parameter	Types	Purposes	Required	Default
`allowdoublebook`	integer (`0`,`1`)	Allows a user to determine whether or not to allow double booking on user's calendars. `0` = Disallow double booking`1` = Allow double booking	N	`1`
`calid`	string	The user's `calid`, used to generate the new calendar's `calid`.	Y	N/A
`id`	unique identifier string	The session identifier.	Y	N/A

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

`set_calprops`	integer (`0`,`1`)	A boolean indicating whether to set the properties of the new calendar. `1` = Set properties.`0` = Do not set properties.	N	`0`

TABLE 7–2 `createcalendar` Parameter *(Continued)*

Parameter	Types	Purposes	Required	Default
`subscribe`	integer (`0` or `1`)	Allows a user to specify if the newly created calendar should be added to the user's subscribed calendar list. `1` = Subscribe to the calendar `0` = Do not subscribe to the calendar	N	`0`
`subscribe`	integer (`0`,`1`)	Allows a user to determine if a newly created calendar should be added to the user's subscribed calendar list. `0` = Do not subscribe `1` = Subscribe	N	`1`

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, the `service.wcap.allowcreatecalendars` parameter in the `ics.conf` file must be set to `"yes"`, which is the default.

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 attempts 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 `FAILED: ILLEGAL_CALID_NAME`.

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 "Command: set_calprops" on page 230 command.

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

Returns

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

Error Codes

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

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

Command: deletecalendar

Purpose

This command deletes a user's calendar.

Parameters

TABLE 7–3 `deletecalendars` Parameter

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

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

`unsubscribe`	integer (`0`,`1`)	Determines if deleted calendars should be removed from the user's subscribed calendar list. `0` = Do not unsubscribe `1` = Unsubscribe deleted calendars	N	`1`

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 can use this command, unless the `ics.conf` parameter `service.wcap.allowdeletecalendars` is set to `"yes"` (which is the default).

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: `FAILED: CALENDAR_DOES_NOT_EXIST (29)`.

Example

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

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

Command: deletecomponents_by_range

Purpose

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

Note – ENS notifications for `appid` are not yet implemented.

Parameters

TABLE 7–4 `deletecomponents_by_range` Parameters

Parameter	Type	Purpose	Required	Default
`appid`	string	A runtime parameter (not stored in the database) that specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the *Sun Java System Communications Services 6 2005Q4 Event Notification Service Guide*.	N	N/A
`calid`	string	Semicolon-separated list of calendar identifiers from which to delete events and todos. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	N	Current user's `calid`
`dtend`	Date-Time string	End time and date of events or todos to be deleted. A value of `0` means delete all events and todos up to the end of time. This value must be in coordinated universal time.	N	`0`
`dtstart`	Date-Time string)	Start time and date of events or todos to be deleted. A value of `0` means delete all events and todos from the beginning of time. This value must be in coordinated universal time.	N	`0`

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

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

TABLE 7–4 `deletecomponents_by_range` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`smtp`	integer (`0`, `1`)	Send email cancellation to user with no calendar. `0` = No `1` = Yes	N	`1`

Description

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

Error Codes

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

Command: deleteevents_by_id

Purpose

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

Parameters

TABLE 7–5 `deleteevents_by_id` Parameters

Parameter	Type	Purpose	Required	Default
`appid`	string	A runtime parameter (not stored in the database) that specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the*Sun Java System Communications Services 6 2005Q4 Event Notification Service Guide*	N	N/A
`calid`	string	Calendar identifier of event to delete. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	Y	N/A

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

`id`	unique identifier string	The session identifier. Required unless the calendar is public.	Y	`NULL`
`mod`	integer `1,2,3,4`	A modifier indicating which recurrences to delete, or semicolon-separated list of modifiers. If a list, it must have same number of elements as `uid` list. One of the following values: `1` = `THISINSTANCE2` = `THISANDFUTURE3` = `THISANDPRIOR 4` = `THISANDALL`	Y	N/A
`notify`	integer `0,1`	A boolean indicating whether or not to notify attendees of this change. `1` = Notify attendees. `0` = Do not notify attendees.	N	`0`

TABLE 7–5 `deleteevents_by_id` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`rid`	string	Recurrence identifier of the event, or semicolon-separated list of recurrence identifiers. If a list, it must have same number of elements as the `uid` list. If there are no recurrences, the value is `0`.	Y	N/A
`smtp`	integer (`0`, `1`)	Send email cancellation to user with no calendar. `0` = No `1` = Yes	N	`1`
`tzid`	time zone ID string	Default time zone to use if the `rid` parameter does not have a time zone specified. For example, "America/Los_Angeles"	N	server's default time zone
`uid`	string	Unique Identifier of an event to be deleted, or semicolon-separated list of unique identifiers.	Y	N/A

Description

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

Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If the `uid` does not exist, the server returns error code `59`. See also, "Error Handling" on page 102

Recurrences

If the `rid` parameter is passed, the command also deletes recurrences, as specified by the `mod` parameter. (See "Recurring Components–Deleting" on page 124.) 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 UID's 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://calendarserver/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:-//SunJavaSystem/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
BEGIN:VEVENT
UID:uid-EVENT1
REQUEST-STATUS:2.0;Success.  Delete successful.
END:VEVENT
BEGIN:VEVENT
UID:uid-EVENT2
REQUEST-STATUS:2.0;Success.  Delete successful.
END:VEVENT
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Command: deleteevents_by_range

Purpose

Delete events from a calendar in a specified range.

Note – ENS notifications for `appid` are not yet implemented.

Parameters

TABLE 7–6 `deleteevents_by_range` Parameters

Parameter	Type	Purpose	Required	Default
`appid`	string	A runtime parameter (not stored in the database) that specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the*Sun Java System Communications Services 6 2005Q4 Event Notification Service Guide*	N	N/A
`calid`	string	Semicolon-separated list of calendar identifiers from which to delete events. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	N	Current user's `calid`
`dtend`	Date Time string	End time and date of events to be deleted. A value of `0` means delete all events until the end of time.	N	`0`
`dtstart`	Date Time string	Start time and date of events to be deleted. A value of `0` means delete all events from the beginning of time.	N	`0`

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

TABLE 7–6 `deleteevents_by_range` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`id`	unique identifier string	The session identifier.	Y	N/A
`smtp`	integer (`0`, `1`)	Send email cancellation to user with no calendar. `0` = No `1` = Yes	N	`1`

Description

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

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

Error Codes

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

See also, "Error Handling" on page 102

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

Command: deletetodos_by_id

Purpose

Delete one or more todos from a calendar.

Parameters

TABLE 7–7 `deletetodos_by_id` Parameters

Parameter	Type	Purpose	Required	Default
`appid`	string	A runtime parameter that is not stored in the database. This parameter specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the*Sun Java System Communications Services 6 2005Q4 Event Notification Service Guide*	N	N/A
`calid`	string	Calendar identifier of the todos to delete. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	Y	N/A

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

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

TABLE 7–7 `deletetodos_by_id` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default

`mod`	integer	A modifier indicating which recurrences to delete, or semicolon-separated list of modifiers. If a list, must have same number of elements as `uid` list.
		One of the following values:
		`1` = `THISINSTANCE`
		`2` = `THISANDFUTURE`
		`3` = `THISANDPRIOR`
		`4` = `THISANDALL`
			Y	N/A

`notify`	integer (`0`,`1`)	A boolean telling whether or not to notify attendees of this change. `1` = Notify attendees. `0` = Do not notify attendees.	N	`0`
`rid`	string	The recurrence identifier of the todo, or a semicolon-separated list of recurrence identifiers. If a list, it must have the same number of elements as the `uid` list. If there are no recurrences, the value is `0`.	Y	N/A
`tzid`	time zone ID string	Default time zone to use if `dtstart`, or `dtend` parameters do not have a time zone specified. For example, "America/Los_Angeles"	N	server's default time zone
`uid`	string	The unique identifier of a todo to be deleted, or a semicolon-separated list of unique identifiers.	Y	N/A

Description

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

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

Error Codes

If the `uid` does not exist, returns error `59`.

See also, "Error Handling" on page 102

Recurrences

If the `rid` parameter is passed, the command also deletes recurrences, as specified by the `mod` parameter. See "Recurring Components–Deleting" on page 124

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 UID's 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://calendarserver/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:-//SunJavaSystem/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
BEGIN:VTODO
UID:uid-TODO1
REQUEST-STATUS:2.0;Success.  Delete successful.
END:VTODO
BEGIN:VTODO
UID:uid-TODO2
REQUEST-STATUS:2.0;Success.  Delete successful.
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Command: deletetodos_by_range

Purpose

Delete todos in a range from a calendar.

Note – ENS notifications for `appid` are not yet implemented.

Parameters

TABLE 7–8 `deletetodos_by_range` Parameters

Parameter	Type	Purpose	Required	Default
`appid`	string	A runtime parameter that is not stored in the database. The parameter specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the*Sun Java System Communications Services 6 2005Q4 Event Notification Service Guide*	N	N/A
`calid`	string	Semicolon-separated list of calendar identifiers from which to delete todos. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	N	Current user's `calid`
`dtstart`	Date Time string	Start time and date of todos to be deleted. A value of `0` means delete all todos from the beginning of time.	N	`0`
`dtend`	Date Time 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`

TABLE 7–8 `deletetodos_by_range` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`id`	unique identifier string	The session identifier.	Y	N/A
`smtp`	integer (`0`, `1`)	Send email cancellation to user with no calendar. `0` = No `1` = Yes	N	`1`

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 todos that occur on the date March 1, 2002.

```
http://calendarserver/deletetodos_by_range.wcap?id=23423423434abc
              &calid=jdoe
              &dtstart=20020301T000000Z
              &dtend=20020301T235959Z
```

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

Error Codes

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

See also, "Error Handling" on page 102

Command: export

Purpose

Export events and todos from a calendar to a file.

Parameters

TABLE 7–9 `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. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's de fault calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	N	Current user's `calid`

`content-out`	string	Content type for output file. One of the following:
		
		`text/calendarx`
		
		`text/xml`
			Y	N/A

`dtend`	Date Time string	End time and date of the events and todos to export. A value of `0` means export all components from the start date to the latest date.	N	`0`
`dtstart`	Date Time string	Start time and date of events and todos to export. A value of `0` means export all components from the earliest date to the end date.	N	`0`
`id`	unique identifier string (`uid`)	The session identifier.	N	`NULL`

Description

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

Range

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

HTTP Post Examples

You must use this command with an HTTP `POST`. This is 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://calendarserver:12345/export.wcap
               ?id=t9u9m0eh8x5pu9b&calid=jdoe;john&dtstart=0&dtend=0
               &content-out=text/calendar"\>
```

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

This is the output generated:

```
HTTP/1.0 200
 Date: Thu, 03 Jun 2002 22:15:52 GMT
 Content-type: text/calendar
 Content-disposition: attachment; filename="export.ics"
 Content-length: 7004
BEGIN:VCALENDAR
METHOD:PUBLISH
 VERSION:6.0
 BEGIN:VEVENT
 UID:tm-001
 RECURRENCE-ID:20020519T010000Z
 DTSTAMP:20020603T221548Z
 SUMMARY:Calendar Staff
 DTSTART:20020518T170000Z
 DTEND:20020518T190000Z
 CREATED:20020603T024254Z
 LAST-MODIFIED:20020603T024254Z
 PRIORITY:1
 SEQ:1
 GEO:37.463581;-121.897606
 DESC:This is the description for event with UID = tm-001
 URL:http://calendarserver/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://calendarserver/susan?uid=tm-001
```

```
LOCATION:Green Conference Room
STATUS:CONFIRMED
TRANSP:OPAQUE
END:VEVENT
END:VCALENDAR
```

Command: fetchcomponents_by_alarmrange

Purpose

Retrieve calendar events and todos with alarm triggers.

Parameters

TABLE 7–10 `fetchcomponents_by_alarmrange` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	A semicolon-separated list of calendar identifiers. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	N	Current user's `calid`
`component-type`	keyword (`event`, `todo`, `all`)	Indicates which components to return: `event` returns only events`todo` returns only todos `all` returns both events and todos If an invalid value is passed in, the system assumes `all`.	N	`all`

TABLE 7–10 `fetchcomponents_by_alarmrange` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`compressed`	integer (`0`,`1`)	This parameter is deprecated in this release and might be deleted in future releases. For `compressed=0`, returns less data. Specifically, it does not return the following parameters:`rrules`, `rdates`, `exrules`, and `exdates`. For `compressed=1`, all recurrence data is returned.	N	`0`
`compstate`	semicolon-separated list of component state keywords	The list of component states to fetch. For `compstate` values, see "Fetching Component State Data" on page 110	N	`ALL`
`dtend`	Date Time string	End time and date of events and todos to be returned. A value of `0` means fetch all events.	N	`0`
`dtstart`	Date Time string	Start time and date of events/todos with alarms ready to go off during the specified time. A value of `0` means fetch all events from the beginning of time.	N	`0`
`emailorcalid`	integer (`0`, `1`)	`0` = The `calid` is returned in the calendar address part of the `ATTENDEE` and `ORGANIZER` properties. The X-Token `X-S1CS-EMAIL` has the RFC 822 email address of the invitee or organization. `1` = The email address is returned in the calendar address part of the `ATTENDEE` and `ORGANIZER` properties. The X-Token `X-S1CS-CALID` contains the `calid` value.	N	`0`
`fetchorder`	integer	Specifies the order in which the events and todos are returned. The values are: ■ `0` - Special legacy order- mostly ascending order. ■ `1` - Ascending order ■ `2` - Descending order See "Sorting Order of Returned Events and Todos" on page 126	N	`0`

TABLE 7–10 `fetchcomponents_by_alarmrange` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`emailorcalid`	integer (`0`, `1`)	`0` = Returns `calid` in calendar address part of the `attendee` or `organizer` property and returns the RFC 822 address of the invitee or organizer in `X-S1CS-EMAIL`. `1` = Returns the RFC 822 compliant email address in the calendar address part of the `attendee` or `organizer` property, and returns the `calid` in `X-S1CS-CALID`.	N	`0`

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

`id`	unique identifier string	The session identifier.	Y	N/A
`maxResults`	integer	The maximum number of events and todos to be returned. When `0`, no maximum is applied and the command returns all events and todos found.	N	`0`
`recurring`	integer (`0`, `1`)	`1`= Return all components in compressed form. Compressed form has master entry plus exceptions. `0` = Do not return components in compressed form.	N	`0` (not compressed)
`relativealarm`	integer (`0`, `4`)	Return the alarm as relative or absolute. `0` = Return alarm values as absolute. `4` = Return alarms as originally created.	N	`0`(absolute)
`tzid`	time zone ID string	If `dtstart` and `dtend` parameters are not already in Zulu time, the time zone to use for translating them to Zulu time. For example, “America/Los_Angeles”	N	server’s default time zone
`tzidout`	time zone ID string	Time zone to report returned data in.	N	Zulu time

Description

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

Output Format

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

maxResults Value

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

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

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

Returns

For each calendar specified in `calid`, the server returns the calendar's events and todos having alarms about to go off within the range specified by `dtstart` and `dtend`.

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

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

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

Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If a calendar cannot be accessed or is missing, `errno` is "Error Codes" on page 102(41).

Example

For example, suppose there are 3 events:

- eventA: alarm on Dec. 25, 2001, 12:30 PM GMT

- eventB: alarm on Feb. 10, 2002, 10:00 AM GMT
- todoA: alarm on Jan. 20, 2002, 1:15 PM GMT

Here are two queries and their return values:

Example 1

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

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

It returns eventA and todoA:

```
BEGIN:VCALENDAR
PRODID:-//SunJavaSystem/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
 X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3c11625900005ffe00000011000010b7
DTSTAMP:20011208T011139Z
SUMMARY:eventA
DTSTART:20011225T133000Z
DTEND:20011225T143000Z
CREATED:20011208T004409Z
LAST-MODIFIED:20011208T010857Z
PRIORITY:0
SEQUENCE:4
ORGANIZER;SENT-BY="jdoe@sesta.com"
;X-NSCP-ORGANIZER-UID=jdoe
```

```
;X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:CONFIRMED
TRANSP:OPAQUE
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL
;PARTSTAT=ACCEPTED;CN="JOHN SMITH"
;RSVP=TRUE
;X-NSCP-ATTENDEE-GSE-STATUS=2
:jdoe
X-NSCP-ORIGINAL-DTSTART:20020210T190000Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20011225T123000Z
ATTENDEE:MAILTO:jsmith@company22.com
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;X-NSCP-GSE-COMMENT="REQUEST-COMPLETED":
31074
END:VEVENT
BEGIN:VTODO
UID:3c1162e200207ff600000015000010b7
DTSTAMP:20011208T011139Z
SUMMARY:todoA
DTSTART:20011208T004626Z
DUE:20020120T141500Z
CREATED:20011208T004626Z
LAST-MODIFIED:20011208T011000Z
PRIORITY:0
SEQUENCE:3
PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="jdoe@sesta.com"
;X-NSCP-ORGANIZER-UID=jdoe
;X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20011208T004626Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020120T131500Z
ATTENDEE:MAILTO:jdoe@sesta.com
END:VALARM
X-NSCP-DUE-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":65538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Example 2

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

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

It returns eventB and todoA:

```
BEGIN:VCALENDAR
PRODID:-//SunJavaSystem/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
 X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3c1162b3000051c300000013000010b7
DTSTAMP:20011208T011645Z
SUMMARY:eventB
DTSTART:20020210T110000Z
DTEND:20020210T120020Z
CREATED:20011208T004539Z
LAST-MODIFIED:20011208T011638Z
PRIORITY:0
SEQUENCE:4
ORGANIZER;SENT-BY="jdoe@sesta.com"
;X-NSCP-ORGANIZER-UID=jdoe
;X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:CONFIRMED
TRANSP:OPAQUE
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL
;PARTSTAT=ACCEPTED;CN="John Smith"
;RSVP=TRUE
;X-NSCP-ATTENDEE-GSE-STATUS=2:jsmith
```

```
X-NSCP-ORIGINAL-DTSTART:20021225T213000Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020210T100000Z
ATTENDEE:MAILTO:jsmith@company22.com
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="REQUEST-COMPLETED":131074
END:VEVENT
BEGIN:VTODO
UID:3c1162e200207ff600000015000010b7
DTSTAMP:20011208T011645Z
SUMMARY:todoA
DTSTART:20011208T004626Z
DUE:20020120T141500Z
CREATED:20011208T004626Z
LAST-MODIFIED:20011208T011000Z
PRIORITY:0
SEQUENCE:3
PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="jdoe@sesta.com"
;X-NSCP-ORGANIZER-UID=jdoe
;X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20011208T004626Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020120T131500Z
ATTENDEE:MAILTO:jdoe@sesta.com
END:VALARM
X-NSCP-DUE-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":
5538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Command: fetchcomponents_by_attendee_error

Purpose.

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

Parameters.

TABLE 7–11 `fetchcomponents_by_attendee_error` Parameters

Parameter	Type	Purpose	Required	Default
`attendee`	string	The attendee's `calid` to search on. The command searches the calendars specified in the `calid` parameter for all errors in events for this attendee. If this parameter is not specified, the command searches the primary owner's calendars for all event errors for any attendee.	N	N/A
`calid`	string	A semicolon-separated list of calendar identifiers. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	N	Current user's `calid`

TABLE 7–11 `fetchcomponents_by_attendee_error` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`component-type`	keyword (`event`, `todo`, `all`)	Indicates which components to return: `event` returns only events `todo` returns only todos `all` returns both events and todos If an invalid value is passed in, the system assumes `all`.	N	`all`
`compressed`	integer (`0`,`1`)	This parameter is deprecated in this release and might be deleted in future releases. For `compressed=0`, returns less data. Specifically, it does not return the following parameters:`rrules`, `rdates`, `exrules`, and `exdates`. For `compressed=1`, all recurrence data is returned.	N	`0`
`emailorcalid`	integer (`0`, `1`)	`0` = The `calid` is returned in the calendar address part of the `ATTENDEE` and `ORGANIZER` properties. The X-Token `X-S1CS-EMAIL` has the RFC 822 email address of the invitee or organization. `1` = The email address is returned in the calendar address part of the `ATTENDEE` and `ORGANIZER` properties. The X-Token `X-S1CS-CALID` contains the `calid` value.	N	`0`
fetchorder	integer	Specifies the order in which the events and todos are returned. The values are: ■ `0` - Special legacy order- mostly ascending order. ■ `1` - Ascending order ■ `2` - Descending order See "Sorting Order of Returned Events and Todos" on page 126	N	`0`

TABLE 7–11 `fetchcomponents_by_attendee_error` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`emailorcalid`	integer (`0`, `1`)	`0` = Returns `calid` in calendar address part of the `attendee` or `organizer` property and returns the RFC 822 address of the invitee or organizer in `X-S1CS-EMAIL`. `1` = Returns the RFC 822 compliant email address in the calendar address part of the `attendee` or `organizer` property, and returns the `calid` in `X-S1CS-CALID`.	N	`0`

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

`id`	unique identifier string	The session identifier.	Y	N/A
`maxResults`	integer	The maximum number of events and todos to be returned. When `0`, no maximum is applied and the command returns all events and todos found.	N	`0`
`recurring`	integer (`0`, `1`)	`1` = Returns all components in compressed form. Compressed form has master entries plus exceptions. `0` = Returns components expanded, but without master record and exceptions.	N	`0` (not compressed)
`relativealarm`	integer (`0`, `4`)	Return the alarm as relative or absolute. `0` = Return alarm values as absolute. `4` = Return alarms as originally created.	N	`0` (absolute)
`tzidout`	standard time zone string	The time zone returned data is translated to.	N	Returns data in Zulu time

Description

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

Output Format

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

maxResults Value

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

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

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

Returns

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

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

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

cal1 Events	cal2 Events
event - 4c1	event - 4c2
attendee: `john`	attendee: `susan`
status: OK	status: error

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

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

Error Codes

If the operation is successful, the error number of `0` is appended to the error string. If the command fails for any reason, `errno` is "Error Codes" on page 102(42).

Command: fetchcomponents_by_lastmod

Purpose.

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

Parameters.

TABLE 7–12 `fetchcomponents_by_lastmod` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	A semicolon-separated list of calendar identifiers. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	N	Current user's `calid`
`component-type`	keyword (`event`, `todo`, `all`)	Indicates which components to return: `event` returns only events`todo` returns only todos `all` returns both events and todos If an invalid value is passed in, the system assumes `all`.	N	`all`
`compressed`	integer (`0`,`1`)	This parameter is deprecated in this release and might be deleted in future releases. For `compressed=0`, returns less data. Specifically, it does not return the following parameters:`rrules`, `rdates`, `exrules`, and `exdates`. For `compressed=1`, all recurrence data is returned.	N	`0`
`compstate`	semicolon-separated list of component state keywords	The list of component states to fetch. For `compstate` values, see "Fetching Component State Data" on page 110	N	`ALL`
`dtend`	Date Time string	End time and date of events to be returned. A value of `0` means fetch all events.	N	`0`
`dtstart`	Date Time string	Start time and date of events to be returned. A value of `0` means fetch all events from the beginning of time.	N	`0`

TABLE 7–12 `fetchcomponents_by_lastmod` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`emailorcalid`	integer (`0`, `1`)	`0` = The `calid` is returned in the calendar address part of the `ATTENDEE` and `ORGANIZER` properties. The X-Token `X-S1CS-EMAI`L has the RFC 822 email address of the invitee or organization. `1` = The email address is returned in the calendar address part of the `ATTENDEE` and `ORGANIZER` properties. The X-Token `X-S1CS-CALID` contains the `calid` value.	N	`0`
`fetchorder`	integer	Specifies the order in which the events and todos are returned. The values are: ■ `0` - Special legacy order- mostly ascending order. ■ `1` - Ascending order ■ `2` - Descending order See "Sorting Order of Returned Events and Todos" on page 126	N	`0`
`emailorcalid`	integer (`0`, `1`)	`0` = Returns `calid` in calendar address part of the `attendee` or `organizer` property and returns the RFC 822 address of the invitee or organizer in `X-S1CS-EMAIL`. `1` = Returns the RFC 822 compliant email address in the calendar address part of the `attendee` or `organizer` property, and returns the `calid` in `X-S1CS-CALID`.	N	`0`

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

`id`	unique identifier string	The session identifier.	Y	N/A
`maxResults`	integer	The maximum number of events and todos to be returned. When `0`, no maximum is applied and the command returns all events and todos found.	N	`0`

TABLE 7–12 `fetchcomponents_by_lastmod` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`recurring`	integer (`0`, `1`)	`1` = Returns all components in compressed form. Compressed form has master entry plus exceptions. `0` = Returns all components expanded with individual instances, not with a master record plus exceptions.	N	`0` (not compressed)
`relativealarm`	integer (`0`, `4`)	Return the alarm as relative or absolute. `0` = Return alarm values as absolute. `4` = Return alarms as originally created.	N	`0`(absolute)
`tzid`	time zone ID string	Time zone to use if `dtstart`, or `dtend` parameters are not in Zulu time. For example, "America/Los_Angeles"	N	server's default time zone
`tzidout`	time zone ID string	Time zone to report returned data in.	N	Zulu time

Description

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

Output Format

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

maxResults Value

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

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

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

Returns

For each calendar specified in `calid`, the server returns the calendar's the events and todos that changed during the range specified by `dtstart` and `dtend`.

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

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

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

Error Codes

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

Example

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

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

Here are some queries and their return values:

```
http://calendarserver/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://calendarserver/fetchcomponents_by_lastmod.wcap
                      ?id=jdoe
                      &dtstart=20011201T112233Z
                      &dtend=20020131T112233Z
```

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

```
http://calendarserver/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.

Command: fetchcomponents_by_range

Purpose

Retrieve calendar events and todos.

Parameters

TABLE 7–13 `fetchcomponents_by_range` Parameters

Parameter	Type	Purpose	Required	Default
`attrset`	integer (`0`,`1`)	Allows the user to indicate to the server if they want to retrieve the complete event or task data from the server, or if they only want a subset of the data for each matching component returned. Intended to minimize the amount of data returned and thereby the amount of processing a client must do. `0` = Returns the following parameters: `uid`, `dtstart`, `dtend`, `summary`. `1` = Returns the following parameters: `uid`, `rrule`, `dtstart`, `dtend`, `summary`, `class`, `location`, `valarm`. `2` = Returns the full event.	N	`2`

TABLE 7–13 `fetchcomponents_by_range` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`calid`	string	A semicolon-separated list of calendar identifiers. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	N	Current user's `calid`
`component-type`	keyword (`event`, `todo`, `all`)	Indicates which components to return: `event` returns only events`todo` returns only todos`all` returns both events and todos If an invalid value is passed in, the system assumes `all`.	N	`all`
`compressed`	integer (`0`,`1`)	This parameter is deprecated in this release and might be deleted in future releases. For `compressed=0`, returns less data. Specifically, it does not return the following parameters:`rrules`, `rdates`, `exrules`, and `exdates`. For `compressed=1`, all recurrence data is returned.	N	`0`
`compstate`	semicolon-separated list of component state keywords	The list of component states to fetch. For `compstate` values, see "Fetching Component State Data" on page 110	N	`ALL`
`dtend`	Date Time string	End time and date of events to be returned. A value of `0` means fetch all events.	N	`0`
`dtstart`	Date Time string	Start time and date of events to be returned. A value of `0` means fetch all events from the beginning of time.	N	`0`

TABLE 7–13 `fetchcomponents_by_range` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`emailorcalid`	integer (`0`, `1`)	`0` = Returns `calid` in calendar address part of the `attendee` or `organizer` property and returns the RFC 822 address of the invitee or organizer in `X-S1CS-EMAIL`. `1` = Returns the RFC 822 compliant email address in the calendar address part of the `attendee` or `organizer` property, and returns the `calid` in `X-S1CS-CALID`.	N	`0`
`filter`	string	A string containing a `name-value` pair representing a filter for an event or todo. The `name` can be any valid event or todo RFC 2445 compliant property. The `value` is what to match in the component. Only single valued filters are supported. For example: `CATEGORY=birthday` Various partial matches are allowed using `*`, for example: ■ For *contains:* `*`*string*`*` ■ For *begins with*: `string*` ■ For *ends with*: `*string` For the *name-value* pair `ALL=`*string*, the following components are checked for a match to *string*: `description`, `summary`, `attendee`, `location`, `organizer`.	N	`NULL`
`emailorcalid`	integer (`0`, `1`)	`0` = The `calid` is returned in the calendar address part of the `ATTENDEE` and `ORGANIZER` properties. The X-Token `X-S1CS-EMAIL` has the RFC 822 email address of the invitee or organization. `1` = The email address is returned in the calendar address part of the `ATTENDEE` and `ORGANIZER` properties. The X-Token `X-S1CS-CALID` contains the `calid` value.	N	`0`

TABLE 7–13 `fetchcomponents_by_range` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`fetchorder`	integer	Specifies the order in which the events and todos are returned. The values are: ■ `0` - Special legacy order- mostly ascending order. ■ `1` - Ascending order ■ `2` - Descending order See "Sorting Order of Returned Events and Todos" on page 126	N	`0`
`filter`	string	Name/value pair that represents a filter for an event. The left side is any valid property of an event or todo from RFC 2445. Only single valued filters are currently supported. For example: `filter=(ATTENDEE=jdoe@sesta.com)` where the left side of the value is a property from RFC 2445 and the right side is the value to match in the events and todos. The following are the only supported properties for filtering: ■ `ATTENDEE` ■ `ORGANIZER` ■ `SUMMARY` ■ `DESCRIPTION` ■ `LOCATION` ■ `CLASS` ■ `CATEGORY`	N	`NULL`

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

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

TABLE 7–13 `fetchcomponents_by_range` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`invitecount`	integer (`0`, `1`)	`1` = Requests the server to return the open invitations count, that is, events where `PARSTAT`=needs-action. The integer count is returned in the X-Token `X-S1CS-CALPROPS-INVITATION-COUNT`. if more than one `calid` is specified in the `calid` parameter, the open invitation count for each calendar is returned in the corresponding iCal or XML block. `0` = Count not requested.	N	`0`
`invitecount`	integer (`0`, `1`)	`0` = Do not return a count of the open invitations. Open invitations are events and todos where `PARTSTAT=needs-action`. `1` = Returns the count of open invitations in `X-S1CS-CALPROPS-INVITATION-COUNT` If more than one calid is specified, the open invitation count for each calendar is returned in the corresponding iCal or XML block.	N	`0`
`maxResults`	integer	Currently not implemented. The maximum number of events and todos to be returned. When `0`, no maximum is applied and the command returns all events and todos found.	N	`0`
`recurring`	integer (`0`, `1`)	`1` = Return all components in compressed form. The compressed form has master entry plus exceptions. `0` = Return components as individual instances, without master record and exceptions.	N	`0` (not compressed)
`relativealarm`	integer (`0`, `4`)	Return the alarm as relative or absolute. `0` = Return alarm values as absolute. `4` = Return alarms as originally created.	N	`0` (absolute)

TABLE 7–13 `fetchcomponents_by_range` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default

`searchOpts`	integer
	(`0`,`1`,`2`,`3`)
		How to perform the search. One of the following:
		`0` = `CONTAINS`
		`1` = `BEGINS_WITH`
		`2` = `ENDS_WITH`
		`3` = `EXACT`
			N	`0`

`tzid`	time zone ID string	Default time zone to use if `dtstart`, or `dtend` parameters are not in Zulu time. For example, "America/Los_Angeles"	N	server's default time zone
`tzidout`	time zone ID string	Time zone to report returned data in.	N	Zulu time

Description

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

Output Format

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

Returns

For each calendar specified in `calid`, the server returns the calendar's properties and the events and todos of that calendar that fall within the range specified by `dtstart` and `dtend`.

Tasks returned by this command:

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

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

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

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

Error Codes

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

Output Format

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

maxResults Value

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

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

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

Error Codes

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

Example

Example 1

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

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

It returns one event and one todo for this period:

```
BEGIN:VCALENDAR
PRODID:-//SunJavaSystem/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
 X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3c11625900005ffe00000011000010b7
DTSTAMP:20011208T015014Z
SUMMARY:eventA
DTSTART:20011225T133000Z
DTEND:20011225T143000Z
CREATED:20011208T004409Z
LAST-MODIFIED:20011208T010857Z
PRIORITY:0
SEQUENCE:4
ORGANIZER;SENT-BY="jdoe@sesta.com"
;X-NSCP-ORGANIZER-UID=jdoe
;X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:CONFIRMED
TRANSP:OPAQUE
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED;
CN="John Smith";RSVP=TRUE;X-NSCP-ATTENDEE-GSE-STATUS=2:jsmith
X-NSCP-ORIGINAL-DTSTART:20020210T190000Z
X-NSCP-LANGUAGE:en
```

```
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20011225T123000Z
ATTENDEE:MAILTO:jsmith@company22.com
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;X-NSCP-GSE-COMMENT="REQUEST-COMPLETED":
31074
END:VEVENT
BEGIN:VTODO
UID:3c1162e200207ff600000015000010b7
DTSTAMP:20011208T015014Z
SUMMARY:todoA
DTSTART:20011208T004626Z
DUE:20020120T141500Z
CREATED:20011208T004626Z
LAST-MODIFIED:20011208T011000Z
PRIORITY:0
SEQUENCE:3
PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="jdoe@sesta.com"
;X-NSCP-ORGANIZER-UID=jdoe
;X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20011208T004626Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020120T131500Z
ATTENDEE:MAILTO:jsmith@company22.com
END:VALARM
X-NSCP-DUE-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":
5538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Example 2

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

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

The following events and todos are returned:

```
BEGIN:VCALENDAR
PRODID:-//SunJavaSystem/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
 X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3c1162b3000051c300000013000010b7
DTSTAMP:20011208T011645Z
SUMMARY:Joe's event
DTSTART:20020110T110000Z
DTEND:20020110T120020Z
CREATED:20011208T004539Z
LAST-MODIFIED:20011208T011638Z
PRIORITY:0
SEQUENCE:4
ORGANIZER;SENT-BY="jdoe@sesta.com";
X-NSCP-ORGANIZER-UID=jdoe;
X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:CONFIRMED
TRANSP:OPAQUE
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL;PARTSTAT=ACCEPTED;
CN="John Smith";RSVP=TRUE;X-NSCP-ATTENDEE-GSE-STATUS=2:jsmith
X-NSCP-ORIGINAL-DTSTART:20021225T213000Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020210T100000Z
ATTENDEE:MAILTO:jsmith@company22.com
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
```

```
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="REQUEST-COMPLETED":31074
END:VEVENT
BEGIN:VTODO
UID:3c1162e200207ff600000015000010b7
DTSTAMP:20011208T011645Z
SUMMARY:Joe's Todo
DTSTART:20011208T004626Z
DUE:20020120T141500Z
CREATED:20011208T004626Z
LAST-MODIFIED:20011208T011000Z
PRIORITY:0
SEQUENCE:3
PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="jdoe@sesta.com";
X-NSCP-ORGANIZER-UID=jdoe;
X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20011208T004626Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020120T131500Z
ATTENDEE:MAILTO:jdoe@sesta.com
END:VALARM
X-NSCP-DUE-TZID:America/Los_AngelesX-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:
X-NSCP-ORGANIZR-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COPONENT-STATE;
X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":6538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
BEGIN:VCALENDAR
PRODID:-//SunJavaSystem/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
 X-NSCP-CALPROPS-LAST-MODIFIED:19700101T000000Z
X-NSCP-CALPROPS-CREATED:19700101T000000Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:susan
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VCALENDAR
PRODID:-//SunJavaSystem/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
 X-NSCP-CALPROPS-LAST-MODIFIED:20011010T001050Z
X-NSCP-CALPROPS-CREATED:20000929T180436Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:susan
X-NSCP-CALPROPS-NAME:default
X-NSCP-CALPROPS-PRIMARY-OWNER:susan
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
```

```
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^frs^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:20020110T110000Z
DTEND:20020110T120020Z
CREATED:20011208T004539Z
LAST-MODIFIED:20011208T011638Z
PRIORITY:0
SEQUENCE:4
ORGANIZER;SENT-BY="susan@sesta.com";
X-NSCP-ORGANIZER-UID=susan;
X-NSCP-ORGANIZER-SENT-BY-UID=susan:susan
STATUS:CONFIRMED
TRANSP:OPAQUE
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL;
PARTSTAT=ACCEPTED;CN="Mary Anderson";RSVP=TRUE;
X-NSCP-ATTENDEE-GSE-STATUS=2:marya
X-NSCP-ORIGINAL-DTSTART:20021225T213000Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020210T100000Z
ATTENDEE:MAILTO:marya@company22.com
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:susan@seata.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="REQUEST-COMPLETED":131074
END:VEVENT
BEGIN:VTODO
UID:3c1162e200207ff600000015000010b7
DTSTAMP:20011208T011645Z
SUMMARY:susan's todo
DTSTART:20011208T004626Z
DUE:20020120T141500Z
CREATED:20011208T004626Z
LAST-MODIFIED:20011208T011000Z
PRIORITY:0
SEQUENCE:3
PERCENT-COMPLETE:0
ORGANIZER;SENT-BY="susan@sesta.com";
X-NSCP-ORGANIZER-UID=crowe;
X-NSCP-ORGANIZER-SENT-BY-UID=susan:susa
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20011208T004626Z
X-NSCP-LANGUAGE:en
```

```
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20020120T131500Z
ATTENDEE:MAILTO:susan@sesta.com
END:VALARM
X-NSCP-DUE-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:mailto:susan@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":65538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Command: fetch_deletedcomponents

Purpose.

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

Parameters.

TABLE 7–14 `fetch_deletedcomponents` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	A semicolon-separated list of calendar identifiers. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	N	Current user's `calid`

TABLE 7–14 `fetch_deletedcomponents` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`component-type`	keyword (`event`, `todo`, `all`)	Indicates which components to return: `event` returns only events`todo` returns only todos `all` returns both events and todos If an invalid value is passed in, the system assumes `all`.	N	`all`
`dtend`	Date Time string	End time and date of events to be returned. A value of `0` means fetch all deleted components in the deletelog database until the end of time. The return value has the server time that is used in the next fetch.	N	`0`
`dtstart`	Date Time string	Start time and date of events to be returned. A value of `0` means fetch all deleted components in the deletelog database from the beginning of time.	N	`0`

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

`id`	unique identifier string	The session identifier.	Y	N/A
`maxResults`	integer	The maximum number of events and todos to be returned. When `0`, no maximum is applied and the command returns all events and todos found.	N	`0`
`recurring`	integer (`0`, `1`)	`1` = Return all components in compressed form. WCAP does not return the expanded instances of the recurring components. Instead, WCAP returns the master entry plus exceptions. If all the instances of the recurring series have been deleted, WCAP returns `dtstart`, `dtend`, `rrules`, `rdates`, `exrules`, `exdates`, and `uid`. `0` = Return components as expanded instances, but no master record.	N	`0` (not compressed)
`tzid`	time zone ID string	Time zone to use if `dtstart`, or `dtend` parameters are not in Zulu time. For example, "America/Los_Angeles"	N	Server's default time zone

TABLE 7–14 `fetch_deletedcomponents` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`tzidout`	time zone ID string	Time zone to report returned data in.	N	Zulu time

Description

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

Output Format

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

maxResults Value

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

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

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

Returns

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

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

Error Codes

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

The following failure codes can be returned:

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

Examples

The first example shows the command defaulting to `recurring=0`, which returns components expanded to individual instances. The second example shows the command using `recurring=1`, which returns the master record plus exceptions.

Fetching Deleted Components

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

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-NSCP-CALPROPS-LAST-MODIFIED:20030110T222754Z
X-NSCP-CALPROPS-CREATED:20030110T221814Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:john doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-OWNERS:""
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^wdeic^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^sf^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
```

```
BEGIN:VEVENT
UID:3e224e5b000041c6000000010000664b
DTSTAMP:20030113T055314Z
DTSTART:20030114T060000Z
DTEND:20030114T070000Z
LAST-MODIFIED:20030113T052800Z
X-NSCP-TRIGGERED_BY:jdoe
END:VEVENT
```

Fetching Deleted Components as Master Record Plus Exceptions

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

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-NSCP-CALPROPS-LAST-MODIFIED:20030110T222754Z
X-NSCP-CALPROPS-CREATED:20030110T221814Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:john doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-OWNERS:""
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^wdeic^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^sf^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3e224e5b000041c6000000010000664b
DTSTAMP:20030113T055314Z
DTSTART:20030114T060000Z
DTEND:20030114T070000Z
LAST-MODIFIED:20030113T052800Z
X-NSCP-TRIGGERED_BY:jdoe
END:VEVENT
BEGIN:VEVENT
UID:3e22553800002781000000003000066eb
DTSTAMP:20030113T055758Z
DTSTART:20030114T060000Z
DTEND:20030114T070000Z
LAST-MODIFIED:20030113T055721Z
RRULE:FREQ=WEEKLY;INTERVAL=1;WKST=SU;COUNT=5
X-NSCP-TRIGGERED_BY:jdoe
```

```
END:VEVENT
BEGIN:VEVENT
UID:3e2255ed00000ff60000000a000066eb
DTSTAMP:20030113T060117Z
DTSTART:20030114T060000Z
DTEND:20030114T070000Z
LAST-MODIFIED:20030113T060107Z
EXDATE:20030116T060000Z
EXDATE:20030116T060000Z
EXDATE:20030116T060000Z
 RRULE:FREQ=DAILY;INTERVAL=1;WKST=SU;COUNT=5
X-NSCP-TRIGGERED_BY:jdoe
END:VEVENT
BEGIN:VTODO
UID:3e2254bd000041c600000001000066eb
DTSTAMP:20030113T055517Z
DTSTART:20030113T055509Z
DUE:20030114T060000Z
LAST-MODIFIED:20030113T055513Z
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Command: fetchevents_by_id

Purpose

Retrieve specific calendar events.

Parameters

TABLE 7–15 `fetchevents_by_id` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	A semicolon-separated list of calendar identifiers. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	N	Current user's `calid`
`compressed`	integer (`0`,`1`)	This parameter is deprecated in this release and might be deleted in future releases. For `compressed=0`, returns less data. Specifically, it does not return the following parameters:`rrules`, `rdates`, `exrules`, and `exdates`. For `compressed=1`, all recurrence data is returned.	N	`0`
`compstate`	semicolon-separated list of component state keywords	The list of component states to fetch. For `compstate` values, see "Fetching Component State Data" on page 110	N	`ALL`
`emailorcalid`	integer (`0`, `1`)	`0` = The `calid` is returned in the calendar address part of the `ATTENDEE` and `ORGANIZER` properties. The X-Token `X-S1CS-EMAIL` has the RFC 822 email address of the invitee or organization. `1` = The email address is returned in the calendar address part of the `ATTENDEE` and `ORGANIZER` properties. The X-Token `X-S1CS-CALID` contains the `calid` value.	N	`0`

TABLE 7–15 `fetchevents_by_id` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`emailorcalid`	integer (`0`, `1`)	`0` = Returns `calid` in calendar address part of the `attendee` or `organizer` property and returns the RFC 822 address of the invitee or organizer in `X-S1CS-EMAIL`. `1` = Returns the RFC 822 compliant email address in the calendar address part of the `attendee` or `organizer` property, and returns the `calid` in `X-S1CS-CALID`.	N	`0`

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

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

`mod`	integer	A modifier indicating which recurrences to retrieve. One of the following values:
		`1` = `THISINSTANCE`
		`2` = `THISANDFUTURE`
		`3` = `THISANDPRIOR`
		`4` = `THISANDALL`
			N	`1` (`THISINSTANCE`)

`recurring`	integer (`0`, `1`)	`1` = Returns all components in compressed form , which contains a master entry plus exceptions `0` = Returns components expanded into individual instances.	N	`0` not compressed
`relativealarm`	integer (`0`, `4`)	Return the alarm as relative or absolute. `0` = Return alarm values as absolute. `4` = Return alarms as originally created.	N	`0` (absolute)
`rid`	ISO 8601 Date Time string	The recurrence identifier for the event. For a nonrecurring event, set to `0`.	N	`0`
`tzid`	time zone ID string	Time zone to use if the `rid` parameter is not in Zulu time. For example, “America/Los_Angeles”	N	server’s default time zone
`tzidout`	time zone ID string	Time zone that returned data should be translated to.	N	Returns data in Zulu time

TABLE 7–15 `fetchevents_by_id` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`uid`	sting	The unique identifier for the event.	Y	N/A

Description

Use this command to retrieve the specified events and recurrences from the specified calendar. You must specify the `id` parameter with the command unless the specified calendar is a public calendar. The command returns recurrences as specified by the `mod` parameter. See "Recurring Components– Overview" on page 119

Output Format

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

Returns

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

Error Codes

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

Example

This query retrieves an event with a specific id.

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

It returns one event:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
 X-NSCP-CALPROPS-LAST-MODIFIED:20011208T005613Z
X-NSCP-CALPROPS-CREATED:20010913T223336Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
X-NSCP-CALPROPS-NAME:John Doe
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-TZID:America/Los_Angeles
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^WDEIC^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^RSF^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^a^frs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^c^dw^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^a^rs^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^c^w^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:lucy^p^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:jjones^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VEVENT
UID:3c11625900005ffe00000011000010b7
DTSTAMP:20011208T015845Z
SUMMARY:eventA
DTSTART:20011225T133000Z
DTEND:20011225T143000Z
CREATED:20011208T004409Z
LAST-MODIFIED:20011208T010857Z
PRIORITY:0
SEQUENCE:4
ORGANIZER;SENT-BY="jdoe@sesta.com";
X-NSCP-ORGANIZER-UID=jdoe;
X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:CONFIRMED
TRANSP:OPAQUE
ATTENDEE;ROLE=REQ-PARTICIPANT;CUTYPE=INDIVIDUAL;
PARTSTAT=ACCEPTED;CN="John Smith";RSVP=TRUE;
X-NSCP-ATTENDEE-GSE-STATUS=2:jsmith
X-NSCP-ORIGINAL-DTSTART:20020210T190000Z
X-NSCP-LANGUAGE:en
BEGIN:VALARM
ACTION:EMAIL
TRIGGER;VALUE=DATE-TIME:20011225T123000Z
ATTENDEE:MAILTO:jdoe@sesta.com
END:VALARM
X-NSCP-DTSTART-TZID:America/Los_Angeles
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
```

```
X-NSCP-GSE-COMMENT="REQUEST-COMPLETED":31074
END:VEVENT
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Command: fetchtodos_by_id

Purpose

Retrieve specific calendar todos.

Parameters

TABLE 7–16 `fetchtodos_by_id` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	A semicolon-separated list of calendar identifiers. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	N	Current user's `calid`
`compressed`	integer (`0`,`1`)	This parameter is deprecated in this release and might be deleted in future releases. For `compressed=0`, returns less data. Specifically, it does not return the following parameters:`rrules`, `rdates`, `exrules`, and `exdates`. For `compressed=1`, all recurrence data is returned.	N	`0`

TABLE 7–16 `fetchtodos_by_id` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`compstate`	semicolon-separated list of component state keywords	The list of component states to fetch. For `compstate` values, see "Fetching Component State Data" on page 110	N	`ALL`
`emailorcalid`	integer (`0`, `1`)	`0` = The `calid` is returned in the calendar address part of the `ATTENDEE` and `ORGANIZER` properties. The X-Token `X-S1CS-EMAIL` has the RFC 822 email address of the invitee or organization. `1` = The email address is returned in the calendar address part of the `ATTENDEE` and `ORGANIZER` properties. The X-Token `X-S1CS-CALID` contains the `calid` value.	N	`0`
`emailorcalid`	integer (`0`, `1`)	`0` = Returns `calid` in calendar address part of the `attendee` or `organizer` property and returns the RFC 822 address of the invitee or organizer in `X-S1CS-EMAIL`. `1` = Returns the RFC 822 compliant email address in the calendar address part of the `attendee` or `organizer` property, and returns the `calid` in `X-S1CS-CALID`.	N	`0`

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

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

`mod`	integer	A modifier indicating which recurrences to retrieve. One of the following values:
		`1` = `THISINSTANCE`
		`2` = `THISANDFUTURE`
		`3` = `THISANDPRIOR`
		`4` = `THISANDALL`
			N	`1` (`THISINSTANCE`)

TABLE 7–16 `fetchtodos_by_id` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`recurring`	integer (`0`, `1`)	`1` = Returns all components in compressed form, with a master entry plus exceptions. `0` = Returns components in expanded form as individual instances.	N	`0` (not compressed)
`relativealarm`	integer (`0`, `4`)	Return the alarm as relative or absolute. `0` = Return alarm values as absolute. `4` = Return alarms as originally created.	N	`0` (absolute)
`rid`	ISO 8601 Date Time string	The recurrence identifier for the todo. For a nonrecurring todo, set to `0`.	N	`0`
`tzid`	time zone ID string	Time zone to use if the `rid` parameter is not in Zulu time. For example, "America/Los_Angeles"	N	server's default time zone
`tzidout`	time zone ID string	Time zone the returned data should be translated to.	N	Returns data in Zulu time
`uid`	sting	The unique identifier for the todo.	Y	N/A

Description

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

Output Format

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

Returns

For each calendar specified in `calid`, the server returns the calendar's todos. If the todo has recurrences, it returns them as specified by the `rid` and `mod` parameters. See "Recurring Components– Overview" on page 119

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

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

Error Codes

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

Example

For example, a todo "weekly todo B" that's due weekly at 5:00 PM 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, `rid=20020201T170000Z`, of the first item is specified, but no modifier is specified. Therefore, it defaults to `1 THISINSTANCE`:

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

The following output is generated:

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

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

Example 2

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

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

The results of the query are as follows:

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

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

```
X-NSCP-ORGANIZER-UID=jdoe;
X-NSCP-ORGANIZER-SENT-BY-UID=jdoe:jdoe
STATUS:NEEDS-ACTION
X-NSCP-ORIGINAL-DTSTART:20020301T170000Z
X-NSCP-LANGUAGE:en
X-NSCP-DUE-TZID:Europe/London
X-NSCP-TOMBSTONE:0
X-NSCP-ONGOING:0
X-NSCP-ORGANIZER-EMAIL:jdoe@sesta.com
X-NSCP-GSE-COMPONENT-STATE;
X-NSCP-GSE-COMMENT="PUBLISH-COMPLETED":65538
END:VTODO
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Command: get_all_timezones

Purpose

Retrieve data about all time zones supported by the server.

Parameters

TABLE 7–17 `get_all_timezones` Parameters

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

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

TABLE 7–17 `get_all_timezones` Parameters *(Continued)*

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

Description

Use this command to retrieve data about all time zones that are supported by the server. The crossover values are defined to be the dates when the time zone 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 time zone does not have daylight-savings, then this value is set to the empty-string.

Returns

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

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

Error Codes

If there was an error in getting the time zones, the server returns the error `FAILED: GET_ALL_TIMEZONES_FAILED` (24).

Example

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

Example 1

This query gets all time zones.

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

This is the result of the query:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
 X-NSCP-CALPROPS-LAST-MODIFIED:19700101T000000Z
X-NSCP-CALPROPS-CREATED:19700101T000000Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID:default
X-NSCP-CALPROPS-LANGUAGE:en
X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^a^r^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@@o^c^wdeic^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^sf^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^c^^g
X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^p^r^g
X-NSCP-CALPROPS-RESOURCE:0
BEGIN:VTIMEZONE
TZID:Africa/Amman
BEGIN:STANDARD
DTSTART:19950920T000000
TZOFFSETFROM:+0300
TZOFFSETTO:+0200
TZNAME:EEST
RRULE:FREQ=YEARLY;BYDAY=-1FR;BYMONTH=9
END:STANDARD
BEGIN:DAYLIGHT
DTSTART:19930420T000000
TZOFFSETFROM:+0200
TZOFFSETTO:+0300
TZNAME:EEDT
RRULE:FREQ=YEARLY;BYDAY=-1FR;BYMONTH=4
END:DAYLIGHT
END:VTIMEZONE
BEGIN:VTIMEZONE
TZID:Africa/Cairo
BEGIN:STANDARD
DTSTART:19950924T000000
TZOFFSETFROM:+0300
TZOFFSETTO:+0200
TZNAME:EEST
COMMENT:this is a comment
RRULE:FREQ=YEARLY;BYDAY=-1FR;BYMONTH=9
END:STANDARD
BEGIN:DAYLIGHT
DTSTART:19950420T000000
TZOFFSETFROM:+0200
TZOFFSETTO:+0300
TZNAME:EEDT
RRULE:FREQ=YEARLY;BYDAY=-1FR;BYMONTH=4
END:DAYLIGHT
END:VTIMEZONE
BEGIN:VTIMEZONE

...
```

other time zones omitted to conserve space

...

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

Example 2

The following is an example of a time zone array element where crossover dates have been limited to the years from mid-1998 to 2006:

```
timezoneList[20] = new TZ(\qAmerica/Los_Angeles\q,
\qPST\q,
\qPDT\q,
\q-0800\q,
\q-0700\q,
new Array
(\q19981025T090000Z\q,\q20020404T100000Z\q,\q20021031T090000Z\q,
\q20020402T100000Z\q,\q20021029T090000Z\q,\q20020401T100000Z\q,
\q20021028T090000Z\q, \q20020407T100000Z\q, \q20021027T090000Z\q,
\q20030406T100000Z\q, \q20031026T090000Z\q, \q20040404T100000Z\q,
\q20041031T090000Z\q, \q20050403T100000Z\q, \q20051030T090000Z\q,
 \q20060402T100000Z\q, \q20061029T090000Z\q))
```

The “America/Phoenix” time zone 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())
```

Command: get_calprops

Purpose

Retrieve calendar properties.

Parameters

TABLE 7–18 `get_calprops` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	semicolon-separated list of strings	Semicolon-separated list of calendar identifiers from which to retrieve properties. The calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	N	Current user's `calid`

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

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

TABLE 7–18 `get_calprops` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`invitecount`	integer (`0`, `1`)	`1` = Requests the server to return the open invitations count, that is, events where `PARSTAT`=needs-action. The integer count is returned in the X-Token `X-S1CS-CALPROPS-INVITATION-COUNT`. if more than one `calid` is specified in the `calid` parameter, the open invitation count for each calendar is returned in the corresponding iCal or iCal XML block. `0` = Count not requested.	N	`0`
`invitecount`	integer (`0`, `1`)	`0` = Do not return a count of the open invitations, meaning events and todos where `PARTSTAT=needs-action`). `1` = Returns the count of open invitations in `X-S1CS-CALPROPS-INVITATION-COUNT` If more than one calid is specified, the open invitation count for each calendar is returned in the corresponding iCal or XML block.	N	`0`

Description

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

Returns

The command returns a page with the following X-Tokens containing property information for the specified calendars:

- `X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY`
- `X-NSCP-CALPROPS-CALMASTER`
- `X-NSCP-CALPROPS-CATEGORIES`
- `X-NSCP-CALPROPS-CHARSET`
- `X-NSCP-CALPROPS-CHILDREN`
- `X-NSCP-CALPROPS-CREATED`
- `X-NSCP-CALPROPS-DESCRIPTION`
- `X-NSCP-CALPROPS-LANGUAGE`
- `X-NSCP-CALPROPS-LAST-MODIFIED`
- `X-NSCP-CALPROPS-NAME`
- `X-NSCP-CALPROPS-OWNERS`

- X-NSCP-CALPROPS-PRIMARY-OWNER
- X-NSCP-CALPROPS-READ
- X-NSCP-CALPROPS-RELATIVE-CALID
- X-NSCP-CALPROPS-RESOURCE
- X-NSCP-CALPROPS-TZID
- X-NSCP-CALPROPS-WRITE
- X-S1CS-CALPROPS-ALLOW-DOUBLEBOOKING
- X-S1CS-CALPROPS-COMMON-NAME
- X-S1CS-CALPROPS-FB-INCLUDE
- X-S1CS-CALPROPS-INVITATION-COUNT

Error Codes

If the calendar exists, but the user does not have `READ` access to it, `errno` is set to `FAILED: ACCESS_DENIED_TO_CALENDAR` (28).

If the fetch fails for any calendar, its error number, `errno`, is set to `FAILED: 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://calendarserver/get_calprops.wcap
                     ?id=2mu95r5so0hq68ts6q3
                     &calid=jdoe;jsmith;susan
                     &fmt-out=text/calendar
```

This is the returned data:

```
BEGIN:VCALENDAR
PRODID:-//SunJavaSystem/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-NSCP-CALPROPS-LAST-MODIFIED:20030415T001028Z
X-NSCP-CALPROPS-CREATED:20030415T001028Z
X-NSCP-CALPROPS-READ:999
X-NSCP-CALPROPS-WRITE:999
X-NSCP-CALPROPS-RELATIVE-CALID;X-S1CS-EMAIL=room1a@netscape.com:Room1A
X-NSCP-CALPROPS-NAME:Galaxy
X-NSCP-CALPROPS-PRIMARY-OWNER:calmaster
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^rsf^g
X-NSCP-CALPROPS-RESOURCE:1
```

```
X-S1CS-CALPROPS-ALLOW-DOUBLEBOOKING:1
X-S1CS-CALPROPS-FB-INCLUDE:1
X-S1CS-CALPROPS-COMMON-NAME: Calendar Master
X-S1CS-CALPROPS-INVITATION-COUNT: 3
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

Command: get_freebusy

Purpose

Get the free-busy calendar information for users.

Parameters

TABLE 7–19 `get_freebusy` Parameters

Parameter	Type	Purpose	Required	Default
`busyonly`	Integer (`0`, `1`)	`0` = return both busy and free periods`1` = return only busy periods	N	`1`
`calid`	string	The calendar identifier. he calid can be supplied in two formats: ■ *string* - calendar identifier ■ `mailto:`*rfc822addr* - An email address appended to "mailto:". The address is mapped to a user with an LDAP lookup, and then the user's default calendar ID is used. Returns: `X-SICS-EMAIL` and `X-NSCP-CALPROPS-RELATIVE-CALID`	Y	Current user's default calendar
`dtstart`	Date Time string	Start time of free-busy search.	Y	N/A
`dtend`	Date Time string	End time of free-busy search.	Y	N/A

TABLE 7–19 `get_freebusy` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`duration`	Integer	Free busy duration time in number of days.	N	`60` or Default taken from `ics.conf`

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

`freebusybegin`	integer	Offset in number of days from the value of `ics.conf` setting `service.wcap.freebusybegin`. Backs off the date range by the value of this parameter. For example, a value of `30` would start the free-busy range 30 days before the current time found in the `ics.conf` parameter.	N	Default `ics.conf` value is `30` (days)
`freebusyend`	integer	Offset in number of days from the value of `ics.conf` setting `service.wcap.freebusyend` to calculate the end of the free-busy range. Extends the date found in the `ics.conf` parameter. For example, a value of `30` would put the end date 30 days beyond the current setting.	N	Default `ics.conf` value is `30` (days)
`id`	unique identifier string	The session identifier.	Y	N/A
`mail`	email address	An email address used to compute free/busy time. The address must be present in the LDAP. When an email address is passed in, all of the user's eligible calendars are used in computing free-busy time. Eligible calendars are those for which this user is the primary owner and have`fbinclude=1`.	Y/N	Either `calid` or `mail` must be specified

TABLE 7–19 `get_freebusy` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`noredirect`	boolean (`0`, `1`)	Parameter passed in with command. When calendar can't be found in the Calendar Server database, if this parameter is set to `0`, the server passes back the redirect URL, if one is present, in the `ics.conf` parameter `service.wcap.freebusy.redirecturl` If this parameter is set to `1`, the server does not check for the redirect URL, instead, it returns an error.	N	`0`
`mail`	RFC 822 address	Can be specified instead of `calid`. But must be a valid user or group email address in the Calendar Server's LDAP. For free-busy lookup, all calendars that have this user as primary owner and have the `fbinclude` calendar property set to `1` are used in computing free-busy time.	Y/N	No default. But must be specified if the calid is not specified
`tzid`	time zone-ID string	Default time zone to use if `dtstart`, or `dtend` parameters do not have a time zone specified. For example, "America/Los_Angeles"	N	Server's default time zone
`tzidout`	time zone ID string	Time zone to report returned data in.	N	Zulu time

Description

This command retrieves the free-busy calendar information for specified users. Free-busy calendar information indicates which times have been scheduled on the user's calendar. Free-busy calendar information does not include any details of the scheduled time.

Free-busy time is calculated for a time period that can be specified in one of three ways:

- `duration` parameter

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

The absolute start and end times to use for this free-busy calculation. These parameters have no default values.

- `freebusybegin` and `freebusyend` parameters.

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

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

For further information about how free-busy calendars are specified, see "Free-busy Calendars" on page 113

Error Codes

If this command fails for any reason, `errno` is set to "Error Codes" on page 102(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 free-busy time for `jdoe` (from 9:00 to 6:00) would be the following:

9:00-10:00	Free
10:00-11:00	Busy
11:00-12:00	Free
12:00-1:00	Busy
1:00-3:00	Free
3:00-4:00	Busy
4:00-6:00	Free

The following URL generates free-busy information found in the calendar `jdoe` between May 1, 2002 and July 1, 2002.

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

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

Here is the output:

```
BEGIN:VCALENDAR
 PRODID:-//SunONE/Calendar Hosting Server//EN
 METHOD:PUBLISH
 VERSION:6.0
  X-NSCP-CALPROPS-LAST-MODIFIED:20010517T012259Z
 X-NSCP-CALPROPS-CREATED:20010517T012259Z
 X-NSCP-CALPROPS-READ:999
 X-NSCP-CALPROPS-WRITE:999
 X-NSCP-CALPROPS-DESCRIPTION:Work Calendar for John Doe
 X-NSCP-CALPROPS-RELATIVE-CALID:jdoe
 X-NSCP-CALPROPS-NAME:John Doe
 X-NSCP-CALPROPS-PRIMARY-OWNER:jdoe
 X-NSCP-CALPROPS-OWNERS:susan
 X-NSCP-CALPROPS-CATEGORIES:business
 X-NSCP-CALPROPS-ACCESS-CONTROL-ENTRY:@^a^S^g
 BEGIN:VFREEBUSY
 DTSTART:20020501T112233Z
 DTEND:20020701T112233Z
 FREEBUSY;FBTYPE=FREE:20020501T112233Z/20020518T170000Z
 FREEBUSY;FBTYPE=BUSY:20020518T170000Z/20020518T190000Z
 FREEBUSY;FBTYPE=FREE:20020518T190000Z/20020525T170000Z
 FREEBUSY;FBTYPE=BUSY:20020525T170000Z/20020525T190000Z
 FREEBUSY;FBTYPE=FREE:20020525T190000Z/20020601T170000Z
 FREEBUSY;FBTYPE=BUSY:20020601T170000Z/20020601T190000Z
 FREEBUSY;FBTYPE=FREE:20020601T190000Z/20020608T170000Z
 FREEBUSY;FBTYPE=BUSY:20020608T170000Z/20020608T190000Z
 FREEBUSY;FBTYPE=FREE:20020608T190000Z/20020615T170000Z
 FREEBUSY;FBTYPE=BUSY:20020615T170000Z/20020615T190000Z
 FREEBUSY;FBTYPE=FREE:20020615T190000Z/20020622T170000Z
 FREEBUSY;FBTYPE=BUSY:20020622T170000Z/20020622T190000Z
 FREEBUSY;FBTYPE=FREE:20020622T190000Z/20020629T170000Z
 FREEBUSY;FBTYPE=BUSY:20020629T170000Z/20020629T190000Z
 FREEBUSY;FBTYPE=FREE:20020629T190000Z/20020701T112233Z
 END:VFREEBUSY
 X-NSCP-WCAP-ERRNO:0
 END:VCALENDAR
```

Command: get_guids

Purpose

Generate a set of globally unique identifiers.

Parameters

TABLE 7–20 `get_guids` Parameters

Parameter	Type	Purpose	Required	Default

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

`guidCount`	integer	Number of `GUID`'s to return.	N	`1`

Description

This command returns the specified number of globally unique identifiers, `GUID`'s. The client need not be authenticated to call this command.

Example

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

```
BEGIN:VCALENDAR
 VERSION:6.0
  PRODID:SunONE Calendar Server 6.0
 X-NSCP-GUID0:e5e4b537465600000b000000c3000000
 X-NSCP-GUID1:e5e4b537d47900000c000000c3000000
 X-NSCP-GUID2:e5e4b537961400000d000000c3000000
 X-NSCP-GUID3:e5e4b5373d3a00000e000000c3000000
 X-NSCP-GUID4:e5e4b537f31400000f000000c3000000
 X-NSCP-GUID5:e5e4b5378259000010000000c3000000
```

```
X-NSCP-GUID6:e5e4b537b0260000110000000c3000000
X-NSCP-GUID7:e5e4b537c2630000120020020c3000000
X-NSCP-GUID8:e5e4b537241f000013000000c3000000
X-NSCP-GUID9:e5e4b537e7330000140000000c3000000
END:VCALENDAR
```

Command: gettime

Purpose

Gets the server time for the requested calendars.

Parameters

TABLE 7–21 `gettime` Parameters

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

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

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

Description

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

Error Codes

- `X-NSCP-WCAP-ERRNO:1` - Session ID timed out or Invalid session ID

- `X-NSCP-WCAP-ERRNO:28` - Command failed. The system denies access to the calendar
- `X-NSCP-WCAP-ERRNO:29` - Command failed. The calendar does not exist in the database
- `X-NSCP-WCAP-ERRNO:55` - Get Server time Failed

Example

Valid session with tzidout

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

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

Command: get_userprefs

Purpose

Retrieve the calendar preferences for the current user.

Parameters

TABLE 7–22 `get_userprefs` Parameters

Parameter	Type	Purpose	Required	Default

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

`id`	unique identifier string	The session identifier.	Y	N/A
`userid`	string	Indicates which user's preferences to get.	N	N/A

Description

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

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

See the*Sun Java System Calendar Server 6 2005Q4 Administration Guide* for more information about server preferences.

Access Control Information (ACI)

The Calendar Server configuration program adds new ACI's. If you are upgrading from an earlier version of Java Enterprise System, you must rerun the configuration program to have the new ACI's added. Or you can use the Directory Server `ldapmodify` command to add them yourself as follows.

In this example, the ACI is added to the root suffix, `o=usergroup`:

```
dn:  o=usergroup
changetype: modify
add: aci
```

```
aci: (targetattr="icscalendar || cn || givenName || sn || uid || mail")
     (targetfilter=(objectClass=icscalendaruser))
     (version 3.0; acl "Allow calendar administrators to proxy-product=ics,
      class=admin,num=2,version=1"; allow (proxy)
      groupdn="ldap:///cn=Calendar Administrators,ou=Groups,o=usergroup";)
```

In the following example, the ACI is added to the `basedn` domain node, `o=sesta.com,o=usergroup`:

Note – All nodes under the `basedn` must be set to allow anyone read and search access rights in order for this command to work. For more information, see the Common Topic "Access Control Information" on page 96

```
dn:  o=sesta.com,o=usergroup
changetype: modify
add: aci
aci:(targetattr="icscalendar || cn || givenName || sn || uid || mail")
    (targetfilter=(objectClass=icscalendaruser))
    (version 3.0; acl "Allow calendar users to read and search other
     users-product=ics,
     class=admin,num=3,version=1"; allow (search,read)
     userdn="ldap:///uid=*,ou=People, o=sesta.com, o=usergroup";)
```

Note – If there is no `basedn` domain node, add the preceding ACI to the root suffix itself by changing the `dn:` value to `o=usergoup`.

Example

The following URL retrieves user preferences for the current user:

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

This is the data returned:

```
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:6.0
 X-NSCP-WCAP-PREF-cn:John Doe
X-NSCP-WCAP-PREF-givenName:John
X-NSCP-WCAP-PREF-mail:jdoe@sesta.com
X-NSCP-WCAP-PREF-preferredlanguage:
X-NSCP-WCAP-PREF-sn:Doe
X-NSCP-WCAP-PREF-icsCalendar:jdoe
```

```
X-NSCP-WCAP-PREF-icsTimezone:Europe/London
X-NSCP-WCAP-PREF-icsDefaultSet:
X-NSCP-WCAP-PREF-icsFirstDay:
X-NSCP-WCAP-PREF-icsSet:name=mygroup$calendar=lucy\\;jjones\\;jdoe
TimeZone$tzmode=specify$tz=America/Denver$mergeInDayView=true
$description=
X-NSCP-WCAP-PREF-icsSubscribed:lucy$,jjones$,jsmith:jdoe
X-NSCP-WCAP-PREF-icsFreeBusy:jdoe
X-NSCP-WCAP-PREF-ceInterval:PT0H30M
X-NSCP-WCAP-PREF-ceDayTail:19
X-NSCP-WCAP-PREF-ceDefaultView:overview
X-NSCP-WCAP-PREF-ceColorSet:pref_group4
X-NSCP-WCAP-PREF-ceToolText:1
X-NSCP-WCAP-PREF-ceToolImage:1
X-NSCP-WCAP-PREF-ceFontFace:PrimSansBT,Verdana,sans-serif
X-NSCP-WCAP-PREF-ceExcludeSatSun:0
X-NSCP-WCAP-PREF-ceGroupInviteAll:1
X-NSCP-WCAP-PREF-ceSingleCalendarTZID:0z
X-NSCP-WCAP-PREF-ceAllCalendarTZIDs:0
X-NSCP-WCAP-PREF-ceNotifyEnable:0
X-NSCP-WCAP-PREF-ceNotifyEmail:jdoe@sesta.com
X-NSCP-WCAP-PREF-ceDefaultAlarmStart:P15M
X-NSCP-WCAP-PREF-ceDefaultAlarmEmail:jdoe@sesta.com
X-NSCP-WCAP-PREF-nswcalCALID:jdoe
X-NSCP-WCAP-PREF-icsDWPHost:DWPserver1
X-NSCP-WCAP-PREF-icsCalendarOwned:jdoe
    $John's Calendar,jdoe:personal$John's Personal Calendar
X-NSCP-WCAP-SERVER-PREF-allowchangepassword:no
X-NSCP-WCAP-SERVER-PREF-allowcreatecalendars:yes
X-NSCP-WCAP-SERVER-PREF-allowdeletecalendars:
X-NSCP-WCAP-SERVER-PREF-allowpublicwritablecalendars:
X-NSCP-WCAP-SERVER-PREF-validateowners:no
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
```

The following string of commands generates the output shown:

```
http://calendarserver/get_userprefs.wcap
        ?id=t95qm0n0es3bo35r
        &fmt-out=text/calendar
        &userid=jdoe
http://calendarserver/get_userprefs.wcap
        ?id=t95qm0n0es3bo35r
        &fmt-out=text/calendar
        &userid=mailto:sue@sesta.com
http://calendarserver/get_userprefs.wcap
        ?id=t95qm0n0es3bo35r
        &fmt-out=text/calendar
        &userid=john123abc
```

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

```
X-NSCP-WCAP-PREF-mail:jdoe@sesta.com
X-NSCP-WCAP-PREF-givenName:John
X-NSCP-WCAP-PREF-sn:Doe
X-NSCP-WCAP-PREF-icsCalendar:jdoe
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
GET /get_userprefs.wcap?id=eo38ue2q2rq6r68u
    &fmt-out=text/calendar&userid=mailto:sue@sesta.com
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-NSCP-WCAP-PREF-cn:Sue Smith
X-NSCP-WCAP-PREF-uid:Sue
X-NSCP-WCAP-PREF-mail:sue@sesta.com
X-NSCP-WCAP-PREF-givenName:Sue
X-NSCP-WCAP-PREF-sn:Smith
X-NSCP-WCAP-PREF-icsCalendar:sue
X-NSCP-WCAP-ERRNO:0
END:VCALENDAR
GET /get_userprefs.wcap?id=eo38ue2q2rq6r68u
    &fmt-out=text/calendar&userid=john123abc
BEGIN:VCALENDAR
PRODID:-//SunONE/Calendar Hosting Server//EN
METHOD:PUBLISHx
VERSION:2.0
X-NSCP-WCAP-ERRNO:61
END:VCALENDAR
```

Command: import

Purpose

Import events and todos from a file to a calendar.

Parameters

TABLE 7–23 `import` Parameters

Parameter	Type	Purpose	Required	Default
`appid`	string	A runtime parameter that is not stored in the database. This parameter specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the *Sun Java System Communications Services 6 2005Q4 Event Notification Service Guide*.	N	N/A
`calid`	string	Identifier of a calendar to which to import event.	Y	N/A
`content-in`	string	Content type of input data. One of the following values:`text/calendar`text/xml`	Y	N/A
`dtend`	Date Time string	End time and date of the events and todos to import. A value of `0` means import all components from the start date to the last date in the file.	N	`0`
`dtstart`	Date Time string	Start time and date of events and todos to import. A value of `0` means import all components from the earliest date in the file to the end date.	N	`0`
`id`	unique identifier string	The session identifier. Required unless the calendar is public.	Y	N/A

Description

Use this command to import to the specified calendar events and todos that have previously been exported to a file using the `export` command. You must specify the file's MIME content type in the `content-in` parameter.

If you do not specify either the starting or ending date, or you pass in `0` as the value for `dtstart` and `dtend`, the command adds all events and todos in the file to the specified calendar. If you specify a starting and ending date, the command imports only events and todos in the file that fall within the time range. Specify starting and ending dates in UTC time, which is indicated by the `Z` at the end of the date-time string.

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

Example

The following `POST` message imports the attached iCalendar file to the calendar `jdoe` using the `import` command. The session ID is required:

```
POST /import.wcap?id=t95qm0n0es3bo35r
  &calid=jdoe&dtstart=0&dtend=0
Content-type: multipart/form-data;
boundary=---------------------------33111928916708
 Content-Length: 679
 -----------------------------------33111928916708
 Content-Disposition: form-data; name="Upload";
filename="C:\\TEMP\\ical1.ics"
 BEGIN:VCALENDAR
 BEGIN:VEVENT
 DTSTART:20020105T100000Z
DTEND:20020105T110000Z
 DTSTAMP:20010104T120020Z
 CREATED:20010105T110000Z
 LAST-MODIFIED:20010104T120020Z
 SUMMARY:Weekly QA Meeting
 UID:random-uid001
 END:VEVENT
 BEGIN:VEVENT
 DTSTART:20020106T100000
 DTEND:20020106T110000
DTSTAMP:20010104T120020
 CREATED:20010105T110000Z
 LAST-MODIFIED:20010104T120020Z
 SUMMARY:Weekly QA Meeting 2
 UID:random-uid002
 END:VEVENT
 END:VCALENDAR
-----------------------------------33111928916708--
```

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

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

Command: list

Purpose

List all calendars owned by current user.

Parameters

TABLE 7–24 `import` Parameters

Parameter	Type	Purpose	Required	Default
`id`	unique identifier string	The session identifier. Required unless the calendar is public.	Y	N/A
`userid`	string	Specifies which user's calendars to display. Can only be used by an administrator, and only if the option is configured on the server.	N	N/A

Description

Returns only those calendars where the user is the primary owner.

Example

```
BEGIN:VCALENDAR
PRODID:-//SunJavaSystem/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-S1CS-CALPROPS-OWNED-CALENDAR:jdoe@example.com
X-S1CS-CALPROPS-OWNED-CALENDAR:jdoe@example.com:MySecondCalendar
X-S1CS-CALPROPS-OWNED-CALENDAR:jdoe@example.com:Vacation
X-S1CS-CALPROPS-OWNED-CALENDAR:jdoe@example.com:ProjectX
END:VCALENDAR
```

Command: list_subscribed

Purpose

List all calendars subscribed to by current user.

Parameters

TABLE 7–25 `list_subscribed` Parameters

Parameter	Type	Purpose	Required	Default
`id`	unique identifier string	The session identifier. Required unless the calendar is public.	Y	N/A
`userid`	string	Specifies which user's calendars to display. Can only be used by an administrator, and only if the option is configured on the server.	N	N/A

Description

Returns calendars the user is subscribed to, including the ones for which the user is the primary owner.

Example

```
BEGIN:VCALENDAR
PRODID:-//SunJavaSystem/Calendar Hosting Server//EN
METHOD:PUBLISH
VERSION:2.0
X-S1CS-CALPROPS-SUBSCRIBED-CALENDAR:jdoe@example.com
X-S1CS-CALPROPS-SUBSCRIBED-CALENDAR:jdoe@example.com:MySecondCalendar
X-S1CS-CALPROPS-SUBSCRIBED-CALENDAR:jdoe@example.com:Vacation
X-S1CS-CALPROPS-SUBSCRIBED-CALENDAR:jdoe@example.com:ProjectX
END:VCALENDAR
```

Command: login

Purpose

Authenticate a specific user.

Parameters

TABLE 7–26 `login` Parameters

Parameter	Type	Purpose	Required	Default
`applyauthfilter`	integer	`1` = Use the `authfilter` for login. **Note –** Communications Express requires this to be turned off (0).	No	`1`

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

`lang`	enum	The user's preferred language.	N	`NULL`
`password`	string	The user's password.	N	N/A
`proxyauth`	string	Used by calendar administrators to perform proxy authorization.	N	N/A
`user`	string	The user's name.	N	`NULL`

Description

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

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

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 "API: csIAccessControl" on page 38. For more information on the Proxy Authentication SDK, see Chapter 3.

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

Returns

The `login` command returns the information shown in this example:

```
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://calendarserver/en/main.html?id=er6en05tv6n3bv9
       &lang=en &host=http://calendarserver/
```

Command: logout

Purpose

Terminate the current user's session.

Parameters

TABLE 7–27 `logout` Parameters

Parameter	Type	Purpose	Required	Default

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

`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://calendarserver/logout.wcap
                      ?id=bu9p3eb8x5p2nm0q3
```

Command: 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
```

Command: search_calprops

Purpose

Search for a calendar's properties.

Parameters

TABLE 7–28 `search_calprops` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	integer (`0`,`1`)	A boolean indicating whether or not to search the `calid` property. `1` = Search the `calid` property `0` = Do not search it.	N	`0`, unless both `primaryOwner` and `name` are `0`
`id`	unique identifier string	The session identifier.	Y	N/A
`maxResults`	integer	The maximum number of results to return.	N	`200`
`name`	integer (`0`,`1`)	A boolean indicating whether or not to search the `name` property. `1` = Search the `name` property`0` = Do not search it.	N	`0`

TABLE 7–28 `search_calprops` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`primaryOwner`	integer (`0,1`)	A boolean indicating whether or not to search the `primaryOwner` property. `1` = Search the `primaryOwner` property.`0` = Do not search it.	N	`0`

`searchOpts`	integer
	`0,1,2,3`
		How to perform the search. One of the following:
		`0 = CONTAINS`
		`1 = BEGINS_WITH`
		`2 = ENDS_WITH`
		`3 = EXACT`
			N	`0`

`search-string`	string	The string to search for in calendars.	Y	N/A

Description

This command requests a search for a calendar using the query type specified by `searchOpts`. WCAP returns the calendar properties for all calendars where a value in the one of the properties, `primaryOwner`, `calid`, `name`, exactly matches the search-string given in the `search-string` parameter. If there are multiple matches, it returns all of them, up to the maximum number of matches specified in `maxResults`.

When searching for the primary owner, the internal search filter is set to find exact matches. If you want to allow the system to perform wildcard searches, such that the search string appears anywhere within the property value, then you must edit the `ics.conf` file by uncommenting the following line:

```
!service.calendarsearch.ldap.primaryownersearchfilter =
"(&(|(uid=*%s*)(cn=*%s*))(objectclass=icsCalendarUser))"
```

Caution – Enabling the wildcard search can negatively impact performance.

Search Properties

This command searches for a match in one of three properties:

- `calid`. The calendar's unique identifier.
- `name`. The calendar's common name.

- 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

The four search options are the following:

- CONTAINS = Returns the calendar properties that contain the search-string.
- BEGINS_WITH = Returns the calendar properties that begin with the search-string.
- ENDS_WITH = Returns the calendar properties that ends with the search-string.
- EXACT = Returns the calendar properties that exactly match the search-string.

Example

The following example URL searches all calendars for the primary owner property, primaryOwner=1 to see if it contains (searchOpts=0) the string jdoe:

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

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

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

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

Command: set_calprops

Purpose

Set the calendar properties of a calendar.

Parameters

TABLE 7–29 `set_calprops` Parameters

Parameter	Type	Purpose	Required	Default
acl	string	A semicolon-separated list of strings specifying the new value of the access control entries.	N	""
cal	encoded string	A list of parameters to decode. There can be multiple instances of this parameter.	N	N/A
calid	string	Identifier of the calendar to modify.	Y	N/A
categories	string	A semicolon-separated list of strings containing the new categories the calendar belongs to.	N	N/A
charset	string	The character set for the calendar.	N	N/A
description	string	The description of the calendar.	N	N/A
doublebooking	integer	Allow or disallow double booking. 1 = Allow double booking. 0 = Disallow double booking.	N	N/A
fbinclude	integer	Specifies whether the calendar can be used in any free/busy lookup. 1 = Include the calendar. 0 = Do not include the calendar. If you want to remove the calendar from the free/busy lookup list, pass in fbinclude=0.	N	N/A
fbinclude	integer (0,1)	Specifies whether the calendar is used for calculating the free-busy time for this user. 0 = Do not include this calendar in the free-busy lookup calculation. 1 = Include this calendar in the free-busy calculation.	N	N/A
fmt-out	string	The format for the returned data. The three format types: text/calendartext/xml	N	text/calendar

TABLE 7–29 `set_calprops` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`id`	unique identifier string	The session identifier.	Y	N/A
`lang`	string	The language of the calendar.	N	N/A
`master`	string	The email contact for the calendar.	N	N/A
`multiple`	integer	The number of calendars for which to set these preferences.	N	`0`
`name`	string	The new text name of the calendar.	N	N/A
`owners`	string	A semicolon-separated list of strings containing he new list non-primary owners.	N	N/A
`read`	integer	This parameter has been deprecated in favor of the `acl` parameter. It remains here only for backwards compatibility. The new read-access value of the calendar. The value can be one of the following: `0 PRIVATE1 PUBLIC 4 PRIMARY_OWNER_ONLY`	N	N/A
`tzid`	string	The new time zone identifier for this calendar.	N	`""`
`write`	integer	This parameter has been deprecated in favor of the `acl` parameter. It remains here only for backwards compatibility. The new write-access value of the calendar. The value can be one of the following: `0 PRIVATE1 PUBLIC 4 PRIMARY_OWNER_ONLY`	N	N/A

Description

This command is an update command, that is, it only changes the values of the parameters you specify. It is not necessary to supply all parameters in the command, only the ones you want 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.

Use `set_calprops` to do the following:

- Change the name of the calendar.

- Change owner of calendar.
- Change category of calendar.
- Change read permission of calendar's event.
- Change write permission of calendar's event.
- Change description of calendar.
- Change character set of calendar.
- Change language of calendar.
- Change email contact of this calendar.
- Change the time zone-identifier of the calendar.
- Allow or disallow double booking for this calendar.

Single Calendar Example

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

```
http://calendarserver?set_calprops.wcap
                      ?id=dfasdfzd3ds
                      &calid=jdoe
                      &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 ID's `xxxx`, `yyyy`, and `zzzz`, setting the descriptions to `X-Calendar`, `Y-Calendar`, and `Z-Calendar`, respectively:

```
http://calendarserver?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 URL's:

```
http://calendarserver?id=fasdfzd3ds&calid=xxxx&desc=X-Calendar
http://calendarserver?id=fasdfzd3ds&calid=yyyy&desc=Y-Calendar
http://calendarserver?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 decodes each `cal` parameter and set the properties appropriately.

Access Control Entries

See "Access Control Information" on page 96, in the Common Topics section at the front of this chapter. Note that due to limitations of the user interface, it is advisable to limit the number of individuals listed in the ACE's to a maximum of 75 per calendar.

Double Booking

If the `ics.conf` parameter `user.allow.doublebook` is set to `"yes"`, then:

- **Allowed** — Double booking is allowed if the user's `doublebooking`calendar property is set to `1`).
- **Disallowed** — Double booking is disallowed if the calendar `doublebooking` property is set to `0`. The command returns error `40 STORE_FAILED_DOUBLE_BOOKED`.

However, if the `user.allow.doublebook` parameter is set to `"no"`, then double booking is disallowed, no matter what the calendar property is set to.

Freebusy Access

See "Free-busy Calendars" on page 113, in the Common Topics section at the front of this chapter.

Choosing a Different Language or Character Set

See "Changing Language or Character Set" on page 100, in the Common Topics section at the front of this chapter.

Command: set_userprefs

Purpose

Modify the preferences or password for a session.

Parameters

TABLE 7–30 `set_userprefs` Parameters

Parameter	Type	Purpose	Required	Default
`add_attrs`	string	Add a new preference.	N	N/A
`convertCalid`	integer (`0`,`1`)	When set to `1` and setting the preferences `icsSet` or `icsSubscribed`, indicates to the server to convert the character `^` to `:` when storing the `calid`. When set to `0`, the parameter is ignored.	N	`0`
`del_attrs`	string	Delete an existing preference.	N	N/A
`fmt-out`	string	The format for the returned data. The two format types: `text/calendartext/xml`	Y	`text/calendar`
`id`	unique identifier string	The session identifier.	Y	N/A
`set_attrs`	string	Modify a preference value.	N	N/A
`userid`	string	Used only by administrators. Indicates which user's preferences to set.	N	N/A

Description

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

Use of this parameter is only necessary when setting the subscribed list of calendars, in `icsSubscribed`, or the subscribed list of groups, in `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 converts the `^` back to a `:`. If the value of the `convertCalid` is `0`, the conversion does not happen.

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

Add a Preference

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

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

Delete a Preference

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

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

Note – If the attribute to be deleted is multi-valued and there are other instances of the preference, only the first instance encountered is deleted. To remove all of the instances of this preference, multiple `set_userprefs` commands must be issued, one for each instance.

For example: After running `get_userprefs`, you see there are two values listed for `icsSubscribed`. To clear both of them, two commands must be issued:

- `/set_userprefs.wcap?id=${SESSIONID}&del_attrs=icsSubscribed`
- `/set_userprefs.wcap?id=${SESSIONID}&del_attrs=icsSubscribed`

Modify a Preference

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

```
http://calendarserver/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://calendarserver/set_userprefs.wcap
                      ?id=b5q2o8ve2rk02nv9t6
                      &userid=jdoe
                      &set_attrs=ceBgcolor=black
```

Command: storeevents

Purpose

Add events to a calendar.

Parameters

TABLE 7–31 `storeevents` Parameters

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

TABLE 7–31 `storeevents` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`alarmStart`	ISO 8601 Date Time string, or ISO 8601 Duration string	The time at which to send the event alarm notification.	N	N/A
`appid`	string	A runtime parameter that is not stored in the database. The parameter specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the *Sun Java System Communications Services 6 2005Q4 Event Notification Service Guide*.	N	N/A
`attachments`	semicolon-separated list of strings	This is for iCalendar interoperability only. The strings are URL's.	N	N/A
`attendees`	semicolon-separated list of strings	An event's iCalendar RFC 2445 attendee properties. For a list of the properties understood by Calendar Server, see "Examples of WCAP Attendee Entries" on page 115. One optional property is specific only to Calendar Server: `SENT-STATUS`, which can have the value of: `NOT-SENT` or `SENT-SUCCEEDED`. The default for this property is `NOT-SENT`. If this property is set to `SENT-SUCCEEDED`, the Group Scheduling Engine (GSE) does not process this attendee.	N	N/A
`calid`	string	Calendar identifier, or email address of the calendar, in which to store the event.	Y	N/A
`categories`	semicolon-separated list of strings	The event categories.	N	N/A
`charset`	string	The character set for the calendar.	N	N/A

TABLE 7–31 `storeevents` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`compressed`	integer (`0`,`1`)	This parameter is deprecated in this release and might be deleted in future releases. For `compressed=0`, returns less data. Specifically, it does not return the following parameters:`rrules`, `rdates`, `exrules`, and `exdates`. For `compressed=1`, all recurrence data is returned.	N	`0`
`contacts`	semicolon-separated list of strings	Contacts for the event.	N	N/A
`desc`	string	Event purpose description. A string of any length. If not passed, `desc` is set to the `summary` value. To include spaces in the string, use the code `%20`.	N	Value of `summary` parameter.
`dtend`	Date Time string	Event end time and date.	N	N/A
`dtstart`	Date Time string	Event start time and date. Required to create or modify events.	Y	N/A
`duration`	ISO 8601 duration string	Event duration. If an event has both a `duration` and a `dtend`, the duration is ignored.	N	N/A
`excludedtstart`	integer (`0`, `1`)	Whether or not to include the `dtstart` date in a recurring series if it does not fall within the `rrules` dates. `0` = include the `dtstart` date `1` = exclude the `dtstart` date	N	`0`
`exdates`	semicolon-separated list of ISO 8601 Date Time Z strings	Event exclusionary recurrence dates. To successfully create events, the `rrules` parameter must be used in conjunction with this parameter.	N	N/A
`exrules`	semicolon-separated list of ISO 8601 Date Time Z strings	Event exclusionary recurrence rules. A semicolon-separated list of recurrence-rule strings. Each rule value must be enclosed in quotes. See "Recurring Components–Overview" on page 119	N	N/A

TABLE 7–31 `storeevents` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`fetch`	integer (`0,1`)	A boolean indicating whether or not to fetch and return newly stored todos. `1` = Fetch and return newly stored todos.`0` = Do not fetch.	N	`0`
`fmt-out`	string	The format for the returned data. The three format types: `text/calendartext/xml`	N	`text/calendar`
`geo`	two semicolon-separated floats	Semicolon-separated string of two float numbers representing the event's geographical location as latitude and longitude. For example, `37.31;-123.2`.	N	`0;0`
`icsClass`	string	Event class. One of the following values: `PUBLICPRIVATECONFIDENTIAL`	N	`PUBLIC`
`icsUrl`	string	Event URL.	N	`""`
`id`	unique identifier string	The session identifier.	Y	N/A
`isAllDay`	integer (`0,1`)	A boolean indicating whether or not the event lasts all day. `1` = Lasts all day. `0` = Does not last all day.	N	`0`
`language`	string	Language of a component record. For example, `en`, `fr`, `de`.	N	N/A
`location`	string	Event location.	N	`""`
`method`	integer (`1,2,4,8`)	ITIP methods for group scheduling. The organizer issues the following ITIP methods: ■ `1` = `PUBLISH` ■ `2` = `REQUEST` ■ `8` = `CANCEL` The attendee issues this ITIP method: `4` = `REPLY`	Y	`1` (`PUBLISH`)

TABLE 7–31 `storeevents` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default

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

`notify`	integer (`0,1`)	This has been deprecated. It remains only for backward compatibility. The Group Scheduling Engine (GSE) module now takes care of all email notifications. A boolean indicating whether or not to notify attendees of a change to an event. `1` = Notify attendees. `0` = Do not notify attendees.	N	`0`
`orgCalid`	string	Calendar identifier of organizer. One of the following parameters must be specified: `orgCalid`, `orgEmail`, or `orgUID`.	N	N/A
`orgCN`	string	Common name of the organizer.	N	N/A
`orgEmail`	email address	Email address of the event contact, who is usually the organizer. One of the following parameters must be specified: `orgCalid`, `orgEmail`, or `orgUID`.	N	N/A
`orgUID`	`userid`	The `userid` of the organizer. One of the following parameters must be specified: `orgCalid`, `orgEmail`, or `orgUID`.	N	N/A
`priority`	integer (`0-9`)	Event priority. Follows RFC 2445. `0` = undefined `1`= highest `9`= lowest	N	`5`

TABLE 7–31 `storeevents` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`rchange`	integer (`0,1`)	A boolean indicating whether or not to expand a recurring event. `1` = expand `0` = do not expand	N	`0`
`rdates`	semicolon-separated list of ISO 8601 Date Time Z strings	Event recurrence dates. To successfully create events, the `rrules` parameter must also be specified.	N	N/A
`relatedTos`	semicolon-separated list of quoted strings	Other events to which this event is related.	N	N/A
`replace`	Integer (0,1)	A boolean. For parameters with semicolon-separated values: `1` = `update`Replace the old values with the new passed-in values. `0` = `append` Adds the new passed-in values to the old ones.	N	`0`
`resources`	semicolon-separated list of strings	The resources associated with the event.	N	N/A
`rid`	ISO 8601 Date Time string	Event recurrence identifier. Not required to create events. If this parameter is not set when trying to modify events, the whole series of events is modified.	N N	N/A
`rrules`	semicolon-separated list of strings	Event recurrence rules. A semicolon-separated list of recurrence-rule strings. Each rule value must be enclosed in quotes. See "Recurring Components–Overview" on page 119	N	N/A
`seq`	integer	(Not implemented) Event sequence number.	N	`0`
`smtp`	integer (`0, 1`)	Send email cancellation to user with no calendar. `0` = No `1` = Yes	N	`1`

TABLE 7–31 `storeevents` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default

`status`	integer	The event status code. One of the following values:
		`0` `CONFIRMED`
		`1` `CANCELLED`
		`2` `TENTATIVE`
		`3` `NEEDS_ACTION`
		`4` `COMPLETED`
		`5` `IN_PROCESS`
		`6` `DRAFT`
		`7` `FINAL`
			N	N/A

`storetype`	integer	Designates whether an explicit "create" or "modify" is attempted on an event. An error results if an attempt is made to create an event that already exists, or to modify an event that does not exist. The error returned is `STOREEVENTS_FAILED (14)`
		The following values are valid:
		`0` `WCAP_STORE_TYPE_NONE`
		`1` `WCAP_STORE_TYPE_CREATE`
		`2` `WCAP_STORE_TYPPE_MODIFY`
		If the attribute is not passed or has a value of `0`, no error conditions are reported.
			N	0

`summary`	string	Event summary. A string of any length. Required for new events. Not required for modifying events. To include spaces in the string, use the code `%20`.	Y/N	Default `summary` available for new events
`transparent`	integer (`0`, `1`)	`1`= transparent. Exclude this event from free-busy calculations. `0` = opaque. Include it in free-busy calculations. If the `isAllDay` parameter is set to `1`, the default is transparent instead of opaque.	N	`0` (opaque) `1` (transparent, if `isAllDay=1`)

TABLE 7–31 `storeevents` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`tzid`	time zone ID string	The time zone used to translate dates to Zulu time for storage. If this parameter is missing, and the time string has no Z after it, the calendar server time zone ID is used.	N	Calendar server time zone ID
tzidout	time zone ID string	Time zone returned data should be translated to.	N	Returns data in Zulu time
`uid`	string	Unique identifier of the event to be stored. System generated for new events. Required to modify events.	N Y	Default `uid` available for new events
`X-`*property name*	string	One or more X-Token properties, in iCalendar RFC 2445 format. For more information on X-Tokens, see "X-Tokens" on page 129.	N	N/A

Description

Use this command to creates or modify events with the specified attributes and stores them in the specified calendar in the database.

The command creates and stores recurrences as specified by the `rrules`, `exrules`, `rid`, `mod`, and `rchange` parameters. See "Recurring Components– Overview" on page 119

Use the `language` parameter to specify the language of the event. See "Changing Language or Character Set" on page 100

For an explanation of how to use the `attendee` and `method` parameters to do group scheduling, see the Common Topics section "Group Scheduling" on page 115.

For an explanation of how to replace, append or delete a parameter, see the explanation in the Common Topics section "Updating Parameter Values" on page 128

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

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

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

Required Parameters

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

Each case requires a different set of parameters :

- To create new events requires only the `dtstart` parameter.

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

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

Double Booking

If the `ics.conf` parameter `user.allow.doublebook` is set to `"yes"`, then:

- **Allowed** — Double booking is allowed if the user's `doublebooking`calendar property is set to `1`).
- **Disallowed** — Double booking is disallowed if the calendar `doublebooking` property is set to `0`. WCAP returns error `40 STORE_FAILED_DOUBLE_BOOKED`.

However, if the `user.allow.doublebook` parameter is set to `"no"`, then double booking is disallowed, no matter what the calendar property is set to.

Duration and dtend

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

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

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

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

Notice that the `T` in the string separates the date information from the time information.

Returns

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

Error Codes

This command cannot modify a linked event. If you attempt to issue the command, it fails and returns: `FAILED: CANNOT_MODIFY_LINKED_EVENTS` (31) in the `errno` array.

If double booking is disallowed, when you try to store an event in a time slot that is already scheduled, the command fails, and returns: `FAILED: STORE_FAILED_DOUBLE_BOOKED`(40).

Example

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

```
http://calendarserver/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
```

Command: storetodos

Purpose

Add one or more todos to a calendar.

Parameters

"Command: storetodos" on page 246 lists `storetodos` parameters:

TABLE 7–32 `storetodos` Parameters

Parameter	Type	Purpose	Required	Default
`alarmAudio`	ISO 8601 Date Time string	The time at which to sound an audio alarm.	N	N/A
`alarmDescription`	string	The message send out with the alarm	N	"This is the alarm description"
`alarmEmails`	semicolon-separated list of email addresses	Recipients of alarm notifications for the todo.	N	N/A
`alarmFlashing`	ISO 8601 Date Time string	The time at which to run a flashing alarm.	N	N/A
`alarmPopup`	ISO 8601 Date Time string ISO 8601 Duration string	The time at which to pop up a dialog alarm.	N	N/A
`alarmStart`	ISO 8601 Date Time string ISO 8601 Duration string	The time at which to send an alarm notification of the todo.	N	N/A
`appid`	string	A runtime parameter that is not stored in the database. The parameter specifies which application is making the request. ENS uses this parameter to determine which X-Tokens to return. Does not affect WCAP command output. For more information on the ENS X-Tokens, see the *Sun Java System Communications Services 6 2005Q4 Event Notification Service Guide*.	N	N/A
`attachments`	semicolon-separated list of strings	This parameter exists to support iCalendar interoperability only. The strings are URL's.	N	N/A

TABLE 7–32 `storetodos` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`attendees`	semicolon-separated list of strings	A todo's iCalendar RFC 2445 attendee properties. For a list of the properties understood by Calendar Server, see "Examples of WCAP Attendee Entries" on page 115	N	N/A
`calid`	string	Calendar identifier (or email address of `calid`) in which to store the todo.	Y	N/A
`categories`	semicolon-separated list of strings	The todo categories.	N	N/A
`charset`	string	The character set for the calendar.	N	N/A
`completed`	ISO 8601 Date Time string	Completion date of the todo. A value of `0` means the todo is not yet completed.	N	`0`
`compressed`	integer (`0`,`1`)	This parameter has been deprecated in this release and might be removed in future releases. For `compressed=0`, returns less data. Specifically, it does not return the following parameters:`rrules`, `rdates`, `xrule`, and `exdates`. For `compressed=1`, all recurrence data is returned.	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. If not passed, `desc` is set to the `summary` value. To include spaces in the string, use the code `%20`.	N	Value in `summary` parameter.
`dtstart`	ISO 8601 Date Time string	Start time and date of the todo. Not required to modify todos. Required to create todos.	N Y	N/A
`due`	ISO 8601 Date Time string	End time and date of the todo.	N	N/A
`duration`	ISO 8601 duration string	Todo duration.	N	N/A

TABLE 7–32 `storetodos` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`excludedtstart`	integer (`0`, `1`)	Whether or not to include the `dtstart` date in a recurring series if it does not fall within the `rrules` dates. `0` = include the `dtstart` date `1` = exclude the `dtstart` date	N	`0`
`exdates`	semicolon-separated list of ISO 8601 Date Time strings	Exclusionary recurrence dates of the todo. To successfully create todos, the `rrules` parameter must also be specified.	N	N/A
`exrules`	semicolon-separated list strings	Todo exclusionary recurrence rules. A semicolon-separated list of recurrence-rule strings. Each rule value must be enclosed in quotes. See "Recurring Components–Overview" on page 119	N	N/A
`fetch`	integer (`0`,`1`)	A boolean indicating whether or not to fetch and return newly stored todos. `1` = Fetch and return newly stored todos. `0` = Do not fetch todos.	N	`0`
`fmt-out`	string	The format for the returned data. The three format types: `text/calendartext/xml`	N	`text/calendar`
`geo`	two semicolon-separated floats	Semicolon-separated string of two float numbers representing the todo's geographical location (latitude and longitude). For example, `37.31;-123.2`.	N	`0;0`
`icsClass`	string	Todo class. One of the following values: `PUBLICPRIVATECONFIDENTIAL`	N	`PUBLIC`
`icsUrl`	string	Todo URL.	N	`""`
`id`	unique identifier string	The session identifier.	Y	N/A
`isAllDay`	integer (`0`,`1`)	A boolean indicating whether or not it is an all day todo. `1` = An all day todo. `0` = Not an all day todo.	N	`0`

TABLE 7–32 `storetodos` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`language`	string	Language of event. (For example, `en`, `fr`, `de`)	N	N/A
`location`	string	Event location.	N	`""`
`method`	integer (`1`,`2`,`4`,`8`)	ITIP method for group scheduling. `1` = `PUBLISH` (organizer only uses this)`2` = `REQUEST` (organizer only uses this)`4` = `REPLY` (attendees only use this)`8` = `CANCEL` (organizer only uses this)	Y	`1` (`PUBLISH`)

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

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

TABLE 7–32 `storetodos` Parameters *(Continued)*

Parameter	Type	Purpose	Required	Default
`orgUID`	`userid`	The `userid` of the organizer. One of the following parameters must be specified: `orgCalid`, `orgEmail`, or `orgUID`.	N	N/A
`percent`	integer (`0-100`)	Percentage completion of the todo.	N	`0`
`priority`	integer (`0-9`)	Event priority. Follows RFC 2445. `0` = undefined `1`= highest `9`= lowest	N	`5`
`rchange`	integer (`0,1`)	A boolean indicating whether or not to replace the `rrule`: `1` = Replace the rule.`0` = Do not replace it.	N	`0`
`rdates`	semicolon-separated list of ISO 8601 Date Time Z strings	Recurrence dates of the todo. To successfully create todos, the `rrules` parameter must also be specified.	N	N/A
`relatedTos`	semicolon-separated list of quoted strings	Other todos to which this todo is related.	N	N/A
`replace`	Integer (0,1)	A boolean. For parameters with semicolon-separated values: `1` = update (replace the old values with the new passed-in values) `0` = append (add the new passed-in values to the old ones)	N	`0`
`resources`	semicolon-separated list of strings	The resources associated with the todo.	N	N/A
`rid`	ISO 8601 Date Time string	Recurrence identifier of the todo. Not required for new todos. If this parameter is not specified the whole series is modified.	N	N/A
`rrules`	semicolon-separated list of strings	Todo recurrence rules. A semicolon-separated list of recurrence-rule strings. Each rule value must be enclosed in quotes. See "Recurring Components–Overview" on page 119	N	N/A

TABLE 7–32 `storetodos` Parameters *(Continued)*

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

`storetype`	integer	Designates whether an explicit "create" or "modify" is attempted on a todo. An error results if an attempt is made to create a todo that already exists, or to modify a todo that does not exist. The error returned is `STORETODOS_FAILED (15)`
		The following values are valid:
		`0 WCAP_STORE_TYPE_NONE`
		`1 WCAP_STORE_TYPE_CREATE`
		`2 WCAP_STORE_TYPE_MODIFY`
		If the attribute is not passed or has a value of `0`, no error conditions are reported.
			N	`0`

`summary`	string	Todo summary. A string of any length. Required for new todos. Not required for modifying todos. To include spaces in the string, use the code `%20`.	Y/N	Default `summary` available for new todos
`transparent`	integer (`0`, `1`)	`1` = private todos transparent. Exclude private todos from free-busy calculations. `0` = opaque. Include private todos in free-busy calculations.	N	`0` (opaque)
`tzid`	time zone ID string,	The time zone used to convert time parameter values to Zulu time for storage. If this parameter is not present, and the string does not end in Z, then the calendar server time-zone ID is used.	N	`Calendar server time zone ID`

TABLE 7–32 `storetodos` Parameters *(Continued)*

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

Description

Use this command to create and modify todos with the specified attributes and stores them in the specified calendar in the database.

The command creates and stores recurrences as specified by `rrules`, `exrules`, `rid`, `mod`, and `rchange` parameters. See "Recurring Components– Overview" on page 119.

For group scheduling, use the `attendee` and `method` parameters as explained in "Group Scheduling" on page 115.

For an explanation of how to replace, append or delete a parameter, see "Updating Parameter Values" on page 128.

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

Required Parameters

This command creates new todos and modifies existing todos. Each case requires a different set of parameters:

- To create new todos requires the `dtstart` parameter.

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

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

Duration and Due

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

Specify the duration in the ISO 8601 format. For example:

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

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

Returns

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

Error Codes

This command cannot modify a linked todo. The command fails and returns: `FAILED: CANNOT_MODIFY_LINKED_TODOS` (32).

Command: subscribe_calendars

Purpose

Add the specified calendars to the user's calendar subscription list.

Parameters

TABLE 7–33 `subscribe_calendars` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	A semicolon separated list of calendar identifiers.	Y	N/A
`id`	unique identifier string	The session identifier. Required unless the calendar is public.	Y	N/A
`userid`	string	Specifies which user is subscribing. Can only be used by an administrator, and only if the option is configured on the server.	N	N/A

Description

Adds the calendars specified in the `calid` parameter to the user's subscription list. A check is made to see if the calendar exists. If not, an error code is returned.

Example

```
http://calendar.sesta.com/subscribe_calendars.wcap
                    ?id=br6p3t6bh5po35r
                    &calid=john@sesta.com;william@sesta.com:baseball
```

Command: unsubscribe_calendars

Purpose

Remove the specified calendars to the user's calendar subscription list.

Parameters

TABLE 7–34 `subscribe_calendars` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	A semicolon separated list of calendar identifiers.	Y	N/A
`id`	unique identifier string	The session identifier. Required unless the calendar is public.	Y	N/A
`userid`	string	Specifies which user is subscribing. Can only be used by an administrator, and only if the option is configured on the server.	N	N/A

Description

Removes the calendars specified in the `calid` parameter to the user's subscription list. No check is made to see if the calendar exists.

Example

```
http://calendar.sesta.com/unsubscribe_calendars.wcap
                    ?id=br6p3t6bh5po35r
                    &calid=john@sesta.com;william@sesta.com:baseball
```

Command: verifyevents_by_ids

Purpose

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

Parameters

TABLE 7–35 `verifyevents_by_ids` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	Calendar from which to fetch events.	N	User's default calendar
`fmt-out`	string	The format for the returned data. The three format types: `text/calendartext/xml`	N	`text/calendar`
`id`	string	Uniquely identifies the session (session ID). The `id` must be specified with the command unless the calendar to fetch from is a public calendar.	N	null
`rid`	ISO 8601 Date Time string	Corresponding set of comma separated event `rid`s. 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

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

```
http://calendarserver/verifyevents_by_ids.wcap
              ?id=$n3o2m05sx9v6t98t8u2p
              &calid=jdoe
              &uid=3bd9e72f000027cf0000000600002113;
                   3bd9e71700004503000000100002113;
                   3bd9e71700004503000000100002113;
                   3bd9cd4700002206000000010000202d
              &rid=0;20021027T230000Z;20021026T230000Z;0
              &fmt-out=text/calendar
```

```
BEGIN:VCALENDAR
  PRODID:-//SunONE/Calendar Hosting Server//EN
  VERSION:6.0
    BEGIN:VEVENT
  UID:3bd9e71700004503000000100002113
  RECURRENCE-ID:20021027T230000Z
  END:VEVENT
BEGIN:VEVENT
  UID:3bd9cd4700002206000000010000202d
  END:VEVENT
 END:VCALENDAR
```

Example 2

```
http://calendarserver/verifyevents_by_ids.wcap
                     ?id=$n3o2m05sx9v6t98t8u2p
                     &calid=savri
                     &uid=3bd9e72f000027cf0000000600002113;
                          3bd9e71700004503000000100002113;
                          3bd9e71700004503000000100002113;
                          3bd9cd4700002206000000010000202d
                          &rid=0;20021027T230000Z;20021026T230000Z;0
                          &fmt-out=text/xml
```

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

Command: 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 7–36 `verifytodos_by_ids` Parameters

Parameter	Type	Purpose	Required	Default
`calid`	string	Calendar from which to fetch todos.	N	User's default calendar

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

`id`	string	Uniquely identifies the session (session ID). The `id` must be specified with the command unless the calendar to fetch from is public calendar.	N	`""`
`rid`	ISO 8601 Date Time string	Corresponding set of comma separated todo `rids`. If this is a non-recurring todo, `rid=0`.	Y	N/A
`tzid`	time zone ID string	Time zone, such as "America/Los_Angeles"	N	Server's default time zone
`uid`	string	Comma separated set of todo `uids`.	Y	N/A

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, and if the `rid`s were not zero, 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

```
http://calendarserver/verifytodos_by_ids.wcap
                      ?id=bo35r2pr3e5po35r
                      &calid=jdoe
                      &uid=3bde188f0000472d0000000b00000399
                      &rid=20021029T200200Z
                      &fmt-out=text/xml
```

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

Command: version

Purpose

To get the current WCAP version.

Parameters

TABLE 7–37 `version` Parameters

Parameter	Type	Purpose	Required	Default

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

Description

This command gets the current WCAP version. (Note: this is different from the server version as well as the HTTP version.)

Returns

The commands supports output types iCal and XML. The variable `X-NSCP-WCAPVERSION` contains the WCAP version number.

Example

The following examples are for each of output data types.

- iCalendar:

```
http://calendarserver/version.wcap
                      ?fmt-out=text/calendar

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

- XML:

```
http://calendarserver/version.wcap
                      ?fmt-out=text/xml

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

Index

Numbers and Symbols

A

C

D

E

F

G

I

L

M

O

P

Q

R

S

T

U

V

W

X

Z