Oracle® Calendar

Application Developer's Guide

Release 2 (9.0.4)

Part No. B10893-01

May 2003

This guide contains considerations and reference material for the use of the Oracle Calendar SDK and the Oracle Calendar web services toolkit.

Oracle Calendar Application Developer’s Guide Release 2 (9.0.4)

Part No. B10893-01

Primary Author: David Wood

Contributors: Graham Gilmore, Keith MacDonald, Eric Plamondon, Jean-Marc Robillard

Contents

Send Us Your Comments ix

Preface xi

Intended Audience xi
Documentation Accessibility xi
Structure xii
Related Documents xiii
Conventions xiii

Part I Oracle Calendar SDK

1 Calendar SDK Overview

SDK Contents 1-1
What's new in this release 1-2
Best Practises 1-2

2 Calendar SDK Implementation Considerations

Character Sets 2-1
iCalendar Support 2-1
iCalendar input 2-2
ATTACH 2-2
ATTENDEE 2-2
CATEGORIES 2-2
CLASS 2-2

DESCRIPTION 2-3
DTSTART, DTEND and DURATION 2-3
LOCATION 2-3
PRIORITY 2-3
RRULE 2-3
STATUS 2-3
SUMMARY 2-3
UID 2-3
VALARM 2-4
iCalendar output 2-4
ATTENDEE 2-4
CATEGORIES 2-4
CLASS 2-4
DESCRIPTION 2-4
DURATION or DTEND 2-4
DTSTART 2-5
LOCATION 2-5
ORGANIZER 2-5
PRIORITY 2-5
RESOURCES 2-5
RRULE 2-5
STATUS 2-5
SUMMARY 2-5
UID 2-5
VALARM 2-5
vCard Support 2-6
Security Model 2-6
Alarms 2-6
User identification 2-7
Data Streams 2-9
Access Control 2-10

3 Calendar SDK Function Reference

SDK Functions 3-1
CSDK_SetConfigFile 3-3

CSDK_CreateSession 3-4
CSDK_DestroySession 3-4
CSDK_GetCapabilities 3-5
CSDK_Authenticate 3-5
CSDK_ConfigureACE........ 3-8
CSDK_Connect 3-9
CSDK_ConnectAsSysop 3-10
CSDK_Deauthenticate 3-11
CSDK_Disconnect........ 3-12
CSDK_SetIdentity........ 3-12
CSDK_DestroyHandle........ 3-13
CSDK_DestroyMultipleHandles........ 3-14
CSDK_GetHandle........ 3-15
CSDK_GetHandleInfo 3-17
CSDK_CreateCallbackStream........ 3-19
CSDK_CreateFileStreamFromFilenames 3-21
CSDK_CreateMemoryStream........ 3-23
CSDK_DestroyMultipleStreams........ 3-25
CSDK_DestroyStream........ 3-26
CSDK_DeleteContacts 3-26
CSDK_FetchContactsByQuery 3-27
CSDK_FetchContactsByUID........ 3-28
CSDK_StoreContacts........ 3-29
CSDK_DeleteEvents........ 3-33
CSDK_FetchEventsByAlarmRange 3-34
CSDK_FetchEventsByRange........ 3-36
CSDK_FetchEventsByUID 3-38
CSDK_StoreEvents........ 3-40
CSDK_DeleteTasks........ 3-46
CSDK_FetchTasksByAlarmRange 3-47
CSDK_FetchTasksByRange........ 3-48
CSDK_FetchTasksByUID 3-49
CSDK_StoreTasks........ 3-51
CSDK_AddConditionToQuery 3-52
CSDK_CreateQuery 3-53

CSDK_DestroyQuery 3-54
CSDK_DestroyResult 3-54
CSDK_GetFirstFailure 3-55
CSDK_GetFirstParseError 3-56
CSDK_GetFirstResult 3-58
CSDK_GetNextFailure 3-59
CSDK_GetNextParseError 3-61
CSDK_GetNextResult 3-63
CSDK_GetStatusLevels 3-64
CSDK_GetStatusString 3-65
Superceded Functions 3-65
Legacy Functions 3-66
CAPI_CreateFileStream 3-67
CAPI_AuthenticateAsSysop 3-69

4 Calendar SDK Configuration Settings

blocking 4-2
client_name 4-2
client_version 4-3
cncachesize 4-3
direntrycachesize 4-3
emailcachesize 4-4
itemcachesize 4-4
log_activity 4-4
log_modulesinclude 4-5
max_caldomain 4-5
max_masternode 4-6
max_sysop 4-6
max_user 4-6
securitycachesize 4-7
tzcachesize 4-7

5 Calendar SDK Flags and Capabilities

CSDK Flags 5-1
Capabilities 5-4

6 Calendar SDK Status Codes

7 Calendar SDK FAQ and Troubleshooting

Frequently Asked Questions 7-1
Troubleshooting 7-2

Part II Oracle Calendar Web Services Toolkit

8 Calendar Web Services Toolkit Overview

About Web Services 8-1
The Web Services Toolkit 8-2
Toolkit Contents 8-2
Security 8-2

9 Calendar Web Services SOAP

HTTP Headers 9-1
Security and Authentication 9-2
Design 9-2
Basic Authentication 9-3
Building Queries 9-5
SOAP Envelope 9-6
SOAP Body 9-7
SOAP Faults 9-7
Calendar Web Service Language (CWSL) 9-9
NoOp 9-10
Search 9-10
Using Search 9-11
vQuery 9-12
Create 9-15
Delete 9-17
Modify 9-18
Property Modify 9-19
Property Add 9-20
Property Delete 9-20

Modifiable Properties 9-21
Ping 9-22
xCal 9-22
Simple Events 9-23
Repeating Events 9-23
Recurring Events 9-26
Expanded Filtered Recurring / Repeating Events 9-28
Daily Notes 9-29
Day Events 9-29
Holidays 9-30
Tasks 9-31
X-ORACLE-EVENTTYPE 9-31
Priority Mapping 9-31

10 Calendar Web Services Client-Side Java Implementation

Java Classes 10-1
Fetching Data 10-2
Soap Faults and Exceptions 10-4
Local Time 10-6

11 Calendar Web Services Status Codes

Module Codes 11-2
Error Codes 11-3

Index

Send Us Your Comments

Oracle Calendar Application Developer's Guide Release 2 (9.0.4)

Part No. B10893-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: infodev_us@oracle.com
- FAX: (650) 633-3836 Attn: Oracle Collaboration Suite Documentation Manager
- Postal service:
  Oracle Corporation
  Oracle Collaboration Suite Documentation Manager
  500 Oracle Parkway, Mailstop 2op5
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

This manual contains considerations and reference material for the use of the Oracle Calendar SDK and the Oracle Calendar web services toolkit.

Intended Audience

This manual is intended for any programmers and developers who intend to use the Oracle Calendar SDK or the Oracle Calendar web services toolkit to create custom applications for calendar access.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle Corporation is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

This manual contains 11 chapters, divided into two parts:

Part I, "Oracle Calendar SDK"

Chapter 1

This chapter introduces the Oracle Calendar SDK.

Chapter 2

This chapter contains an overview of various elements of the Oracle Calendar SDK, as well as items to consider before implementation.

Chapter 3

This chapter contains detailed documentation on the functions provided with this development kit.

Chapter 4

This chapter contains information on the configuration parameters that can be supplied to the Oracle Calendar SDK.

Chapter 5

This chapter contains a variety of information on flags and capabilities.

Chapter 6

This chapter contains an alphabetical list and brief explanation of every status code the SDK provides as feedback.

Chapter 7

This chapter contains frequently asked questions and troubleshooting information on the Oracle Calendar SDK.

Part II, "Oracle Calendar Web Services Toolkit"

Chapter 8

This chapter introduces the Oracle Calendar web services toolkit.

Chapter 9

This chapter describes how the Oracle Calendar web services toolkit uses Extended Markup Language (XML) and Simple Object Access Protocol (SOAP) to retrieve and store iCalendar objects.

Chapter 10

This chapter describes the design of the set of Java classes used to provide contextual collaboration through the access of calendaring data via Oracle Calendar web services.

Chapter 11

This chapter contains a list of web services status codes and brief explanations of what they mean.

Related Documents

For more information, see the following manuals in the Oracle Collaboration Suite documentation set:

- *Oracle Calendar SDK Javadoc* (on the product CD)
- *Oracle Calendar web services Javadoc* (on the product CD)
- *Oracle Calendar Administrator's Guide*
- *Oracle Calendar Reference Manual*

Conventions

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

Part I

Oracle Calendar SDK

This part of the Oracle Calendar Application Developer's Guide describes the Oracle Calendar SDK.

This part contains the following chapters:

- Chapter 1, "Calendar SDK Overview"
- Chapter 2, "Calendar SDK Implementation Considerations"
- Chapter 3, "Calendar SDK Function Reference"
- Chapter 4, "Calendar SDK Configuration Settings"
- Chapter 5, "Calendar SDK Flags and Capabilities"
- Chapter 6, "Calendar SDK Status Codes"
- Chapter 7, "Calendar SDK FAQ and Troubleshooting"

1 Calendar SDK Overview

The Oracle Calendar SDK is a set of functions, written in C / C++ with corresponding Java functions, that a developer can use to create applications that interface with Oracle Calendar. Using a native C interface, the SDK allows for implementation using any language that can call C functions natively.

Using standard iCalendar objects to represent meetings and events, a developer can use CSDK functions to create programs that read / write calendar data, storing the information on the Oracle Calendar server.

Examples of programs that can be created include automated calendar subscriptions, custom interfaces to the Calendar server, or even migration utilities that allow for data extraction from any other system capable of producing iCalendar output.

SDK Contents

The Oracle Calendar SDK includes:

- A shared library implementing the APIs
- A C header file
- Javadoc HTML documentation for the SDK.
- Oracle Calendar client libraries
- Oracle Calendar ACE (authentication, compression, encryption) modules
- Sample / Demo programs.

What’s new in this release

This release of the Oracle Calendar SDK includes the following new features:

- Task operations (VTODO objects)
- Contact operations (VCARD objects)
- Corresponding Java functions with JNI (Java native interface)
- Remote designate operations
- Connection pooling

The inclusion of Java functions facilitates Java implementations that were previously implemented by third parties.

Connection pooling adds configuration options for the connection model used by the SDK. This will greatly enhance resource usage and efficiency when implementing various applications (especially web-based and multi-threaded environments), and promote reuse of existing connections.

Best Practises

The Oracle Calendar SDK is a subset of the core functions used when designing Oracle Calendar desktop clients and server applications. It should be used for rapid development of utilities and applications that extend the existing Oracle Calendar applications, and not as a tool to replace the existing interfaces or logic.

The SDK has been used to integrate with portals, FCGI-based applications, and even simple command-line scripts which might display a day's events. The SDK is best used to achieve a specific goal.

2 Calendar SDK Implementation Considerations

This chapter discusses a variety of factors to be taken into consideration in your Oracle Calendar SDK implementations:

- Character Sets
- iCalendar Support
- Security Model
- Alarms
- User identification
- Data Streams
- Access Control

Character Sets

All SDK functions operate only on UTF-8 encoded text. All strings given to the SDK functions must be in UTF-8 and all strings returned by the SDK will be in UTF-8. For more information on UTF-8, refer to RFC 2279.

iCalendar Support

The Oracle Calendar SDK uses the iCalendar format (as specified in RFC 2445) for dealing with calendar data. iCalendar information saved via the SDK can be retrieved later. However, not all iCalendar data is actively supported by this revision of the SDK. In particular, VFREEBUSY and VJOURNAL components are not supported.

iCalendar input

The iCalendar data is mapped to native data structures. Data for these properties will not always be completely preserved. Some properties are stored only per event, rather than per instance, so only one value is preserved. This section describes the mappings used for the most common properties.

ATTACH

This property is currently ignored.

ATTENDEE

When storing "ATTENDEE" properties, an attempt will be made to correlate attendee properties with calendar users. The legacy function CAPI_StoreEvents will compare the attendee address with the addresses of each supplied handle to identify the calendar user. CSDK_StoreEvents performs a look-up on the calendar server to find the corresponding calendar user. Non-calendar users will still be invited (as "external attendees") when using CSDK_StoreEvents.

CATEGORIES

When using CAPI_StoreEvents, the CATEGORIES property is used to specify the event type. Recognized values are "APPOINTMENT", "DAILY NOTE", "DAY EVENT" and "HOLIDAY". When using CSDK_StoreEvents, the CATEGORIES value is stored on the server and will be returned by the various CSK_FetchEvents functions. (The property X-ORACLE-EVENTTYPE is used with CSDK_StoreEvents to specify the event type and the same values are recognized).

CLASS

This property is mapped to access level. The mapping between iCalendar and the calendar server's access levels is as follows:

From iCalendar to calendar server:

- PUBLIC->PUBLIC
- PRIVATE->PERSONAL
- CONFIDENTIAL->CONFIDENTIAL
- X-ORACLE-NORMAL -> NORMAL (using CSDK_StoreEvents)
- X-CST-NORMAL -> NORMAL (using CAPI_StoreEvents)

This property is stored per event.

DESCRIPTION

This is set to the Event's details. It will be truncated if it is longer than 32 Kb. This property is stored per event when calling CAPI_StoreEvents, and per instance when calling CSDK_StoreEvents.

DTSTART, DTEND and DURATION

If DTEND is present, it will be used to calculate the event duration; the actual end time is not stored. As event times are measured in minutes, the start time and duration will have their 'seconds' component set to zero.

LOCATION

This is stored in the location field. When using CorporateTime Server 5.0 or earlier, this value will be truncated to 32 bytes.

PRIORITY

This property is mapped to one of the calendar server's 5 priority values. This property is stored per event.

RRULE

When using CSDK_StoreEvents, recurrence-rules and exceptions to the rules are stored on the calendar server. When using the legacy CAPI_StoreEvents function, recurrence rules are expanded upon event creation and the recurrence-rule itself is not stored on the calendar server.

STATUS

A value of TENTATIVE indicates a tentative event. Any other value (even CANCELLED) will indicate a non-tentative event.

SUMMARY

This property is mapped to the event title. When using CorporateTime Server 5.0 or earlier, this value will be truncated to 64 bytes.

UID

If a UID is not specified in stored data the calendar server will assign a UID. When using CAPI_StoreEvents, the server-assigned UIDs can be read by calling CAPI_GetLastStoredUIDs. When using CSDK_StoreEvents, the UIDs are returned via the CSDKRequestResult.

VALARM

The current user is able to fetch and store her own reminders. It is not possible to access another user's alarms.

iCalendar output

When data created with the SDK or other Oracle Calendar clients is fetched, the following properties are available. Other properties including Oracle-specific extensions (X-ORACLE-...) may also be returned. It is possible to limit the list of properties returned when fetching data. In some cases, there are performance gains to be made by doing so.

ATTENDEE

A property is generated for each ATTENDEE. The parameters PARTSTAT, ROLE, CUTYPE, and CN are obtained from the attendee and user information.

CATEGORIES

When calling CAPI_FetchEventsBy..., the event type is returned in this property. CSDK_FetchEventsBy... will return a user-specified value (which may have been stored using the SDK or another client).

CLASS

The event access level. The access levels are mapped as:

- PUBLIC -> PUBLIC
- PRIVATE -> PERSONAL
- CONFIDENTIAL -> CONFIDENTIAL
- NORMAL -> PUBLIC

DESCRIPTION

When calling CAPI_FetchEventsBy..., the event details (as seen by the native clients) are returned. CSDK_FetchEventsBy... will return instance-specific details if they exist, or the event details.

DURATION or DTEND

The duration of the event, or the end-time of the event.

DTSTART

The start time of the event.

LOCATION

The event location.

ORGANIZER

The event owner.

PRIORITY

The event priority.

RESOURCES

When calling CAPI_FetchEventsBy..., this property will contain the names of all attendees which are resources. CSDK_FetchEventsBy... will return a string which may have been stored using CSDK_StoreEvents or by another Oracle Calendar client.

RRULE

The recurrence-rule stored on the server. This is not available when calling CAPI_FetchEventsBy...

STATUS

A tentative event will have a TENTATIVE status. Non-tentative events will be marked as CONFIRMED. No other STATUS values are generated.

SUMMARY

The event or instance title.

UID

The user-specified UID (as set on event creation) if available, otherwise a server-generated UID.

VALARM

Converted from native reminders (for the current user only).

vCard Support

The Oracle Calendar SDK uses the vCard format for dealing with contact data. vCard information saved via the SDK can be retrieved later. Versions 2.1 and 3.0 are both supported.

Security Model

There are two parts to the security model: storing and fetching events. These are handled by different security paradigms.

The owner of an event can add or delete properties of that event. When an "ATTENDEE" property is created for a calendar user, and a handle has been supplied for that user, the property is created with default values for its parameters. The owner of the event cannot modify the parameters of that property, only the user to whom it corresponds can do that.

When a user is updating their "ATTENDEE" property no error will be returned if there is an attempt to modify other event data, but the modifications will not occur.

It is possible for a user to refuse invitations from another user. In that case an "ATTENDEE" property will not be created for that user and the status for that user's handle will indicate that the invitation was refused. This may also occur when attempting to double book resources.

When fetching events the security model is based on the iCalendar classification of the event. Users grant other users different access levels to different classes of events. The three access levels are: no read access, read the start and end times of the event only, and read all details of the event. When fetching events with the SDK this results in some events for which only the "UID," "DTSTART," "DURATION" and "DTEND" properties will be returned. All other events will be invisible or all of their properties will be returned.

The Oracle Calendar SDK does not allow users to modify the security records which govern this behaviour.

Alarms

Alarms are considered private to each user, so users cannot read or write alarms for each other. Since users cannot read each other’s alarms it is not possible for users to do fetches by alarm range on each other's calendars. Any user may set an alarm for an event which they are attending, so the same events can have a different alarm when fetched by a different user.

User identification

Users are identified to the Oracle Calendar SDK by a userID string, or by using a search string specifying, for example, the user's name. The string format is flexible and allows the caller to specify a number of optional parameters. Depending on the server configuration, some of these options (such as the Node ID) may be required in the identification string. The same user identification string format is used both at logon and when obtaining a handle, however not all options will be applicable in both cases.

Logging into the server as a resource is not supported, but it is possible to work as a designate for a resource.

All options are specified using key-value pairs. The entire string is a collection of such pairs. The userID is separated from the extended data by an ASCII '?' character. The character immediately following this one is the delimiter of each subsequent field value pair. The delimiter may be any ASCII character except a digit, a letter, NUL, '*' or '='. The rest of the string consists of field-value pairs separated by the chosen delimiter. The string is terminated by a delimiter followed by a NUL character. Field value pairs consist of a field name, followed by an equal sign ('='), followed by the value. The value is a string which does not contain the delimiter character, the NUL character, and for user identification strings, the slash ('/') (aka solidus) character.

For example, the field name G denotes the given name, and S denotes the surname. The following is a sample legal string for identifying a user. No userID is specified, so the optional parameters would be used to search for the user. (Note that if a search results in multiple matches, the SDK will return an error to the caller; a userID is the best method of specifying a user, if it is available.) Even with no userID, we still have the question mark '?' character separating the userID from the extended string. The character immediately following, in this case a slash '/', is used as the delimiter. Note that the string ends with the delimiter character, and is NUL terminated.

```
?/S=Bunny/G=Bugs/
```

Any field used for identifying a user may be terminated with a '*', which is used as a wildcard. This is not available for specifying nodes. The following will also match the preceding user:

```
?/S=Bu*/G=Bugs/
```

Remember that if multiple users match a given search string, the the SDK call will return an error.

Resources have a different name structure; they are identified with the single field RS which indicates resource name.

The following grammar (in ABNF form, as described in RFC 2234) describes legal logon strings. The description diverges from ABNF in that values in double quotes are case-sensitive, ie. field names must be in uppercase. Also, the delimiter character must be the same in all cases in a single string.

```
logon-string = userid "?" [ DELIMITER ext-string ] %x00

userid = *( ALPHA / DIGIT / "-")

ext-string = 1*( field )

field = ( node / company-domain / surname / given-name / initials / generation /
org-unit / organization / country / admin / private / resource-name ) DELIMITER
```

Specifying a particular field more than once is, while redundant, still legal, although only the last field will be used.

```
node = "ND=" node-number
node-number = 1*DIGIT
company-domain = "CD=" 1*VALUE-CHAR
surname = "S=" 1*VALUE-CHAR ["*"]
given-name = "G=" 1*VALUE-CHAR ["*"]
initials = "I=" 1*VALUE-CHAR ["*"]
generation = "X=" 1*VALUE-CHAR ["*"]
org-unit = ( "OU1" / "OU2" / "OU3" / "OU4" ) "=" 1*VALUE-CHAR ["*"]
organization = "O=" 1*VALUE-CHAR ["*"]
country = "C=" 1*VALUE-CHAR ["*"] DELIMITER
admin = "A=" 1*VALUE-CHAR ["*"] DELIMITER
private = "P=" 1*VALUE-CHAR ["*"] DELIMITER
resource-name = "RS=" 1*VALUE-CHAR ["*"] DELIMITER

DELIMITER = %x01-%x29 / %x2B-%x2F / %x3A-%x3C / %x3E-%x40 / %x5B-%x60 /
%x7B-%x7F
```

Note also that the DELIMITER cannot be used as a VALUE-CHAR

```
VALUE-CHAR = %x01-29 / %x2B-2E / %x30-7F
```

Data Streams

By default, the SDK deals with MIME (see RFC 2045) encapsulated iCalendar and vCard objects for both input and output. A single request may fetch data from a list of calendars. A reply to such a request will consist of a separate iCalendar object for each calendar in the list, inside separate MIME parts. That is, a request for events from calendarA and calendarB results in a MIME stream of this form:

```
... MIME envelope
--MIMEBOUNDARYasdfasdf
Content-type: text/calendar
Content-Transfer-Encoding: quoted-printable
BEGIN:VCALENDAR
... events from calendarA
END:VCALENDAR

--MIMEBOUNDARYasdfasdf
Content-type: text/calendar
Content-Transfer-Encoding: quoted-printable
BEGIN:VCALENDAR
... events from calendarB
END:VCALENDAR

--MIMEBOUNDARYasdfasdf--
```

The order of the iCalendar objects corresponds to the order of the calendars in the request list. If a request results in an empty solution set, the return stream will be an empty iCalendar object. If there is any sort of error with a calendar the iCalendar reply object corresponding to that calendar will be empty.

On a successful fetch the "VCALENDAR" may contain many "VEVENT" components, each containing the requested properties, if available. iCalendar allows these different components to contain information about different instances of the same event. The returned data may use any of the following methods to give instance specific information:

- Data for each instance can be placed in a different "VEVENT" component, with a different "DTSTART".
- Data for multiple instances can be placed in a single "VEVENT" by identifying instances with the properties "RRULE", "RDATE", "EXRULE" and "EXDATE"
- Hybrids of the preceding two methods allow grouping of multiple instances which share all properties except their start time in a single "VEVENT" component, and returning many such components.

Please note that the "DTSTART" property returned indicates the start time of the first instance identified in the "VEVENT" component in which it resides and not the start time of the first instance of the event in the Calendar Store. Furthermore the number of "VEVENT" components returned in the calendar has no relation to the number of instances of the event. Consequently, when fetching events, if the recurrence identifying properties are not requested, there will be no way to determine how many instances exist, and to which instances each returned property applies.

When storing, data supplied to the SDK must consist of a single "VCALENDAR" component inside a single MIME part. The calendar may contain many "VEVENT" objects, but these must all be information about a single event. For example, this is a valid input:

```
Content-type: text/calendar
Content-Transfer-Encoding: 7bit
BEGIN:VCALENDAR
VERSION:2.0
BEGIN:VEVENT
event properties
END:VEVENT
BEGIN:VEVENT
event properties
END:VEVENT
END:VCALENDAR
```

Access Control

Access to data through the SDK is controlled by the calendar server. It is based on the requester's identity and the data / operation being requested. the SDK provides an interface to request reading any combination of properties. Properties that the requesting user is not authorized to read will not be returned.

Users will only have privileges to modify the events to which they are invited, or which they own. If the user is the owner of the event they will have full privileges to modify the event (except for modifying other users' attendance information), otherwise if they are invited to the event they will have restricted privileges to modify information relating to their own attendance, such as acceptance and alarms.

The SDK will silently ignore attempts to modify properties that the user is not permitted to modify.

Errors may occur for specific agendas when attempting to modify events or when creating events. These errors will be returned using a supplied array of status values, allowing the rest of the operation to proceed.

3

Calendar SDK Function Reference

This chapter contains detailed information on useful functions included with the Oracle Calendar SDK.

SDK Functions

This section provides details on the following functions:

- CSDK_SetConfigFile
- CSDK_CreateSession
- CSDK_DestroySession
- CSDK_GetCapabilities
- CSDK_Authenticate
- CSDK_ConfigureACE
- CSDK_Connect
- CSDK_ConnectAsSysop
- CSDK_Deauthenticate
- CSDK_Disconnect
- CSDK_SetIdentity
- CSDK_DestroyHandle
- CSDK_DestroyMultipleHandles
- CSDK_GetHandle
- CSDK_GetHandleInfo

- CSDK_CreateCallbackStream
- CSDK_CreateFileStreamFromFilenames
- CSDK_CreateMemoryStream
- CSDK_DestroyMultipleStreams
- CSDK_DestroyStream
- CSDK_DeleteContacts
- CSDK_FetchContactsByQuery
- CSDK_FetchContactsByUID
- CSDK_StoreContacts
- CSDK_DeleteEvents
- CSDK_FetchEventsByAlarmRange
- CSDK_FetchEventsByRange
- CSDK_FetchEventsByUID
- CSDK_StoreEvents
- CSDK_DeleteTasks
- CSDK_FetchTasksByAlarmRange
- CSDK_FetchTasksByRange
- CSDK_FetchTasksByUID
- CSDK_StoreTasks
- CSDK_AddConditionToQuery
- CSDK_CreateQuery
- CSDK_DestroyQuery
- CSDK_DestroyResult
- CSDK_GetFirstFailure
- CSDK_GetFirstParseError
- CSDK_GetFirstResult
- CSDK_GetNextFailure
- CSDK_GetNextParseError

- CSDK_GetNextResult
- CSDK_GetStatusLevels
- CSDK_GetStatusString

CSDK_SetConfigFile

```
CAPIStatus CSDK_SetConfigFile  (  const char *    in_configFileName,
                                  const char *    in_logFileName
                               )
```

Calling this function will allow the SDK to read configuration settings which control error logging, and the other configuration parameters listed in the "Configuration" section of this manual.

If called, this function should be the first SDK function called by your process and should not be called by each thread.

Parameters:

- in_configFileName: A null-terminated string containing the filename of the config file.
- in_logFileName: the name of a file to write log messages to. If this file cannot be created or written to, output will be sent to a file named `Console.log` in the "current" directory.

Returns:

CAPIStatus

Return values:

CAPI_STAT_API_NULL: one of the input parameters was NULL

CAPI_STAT_CONFIG_CANNOT_OPEN : Failed to open in_configFileName

Equivalent Java Method

oracle.calendar.sdk.Api.init()

Example 3–1 Set Configuration File

Create a file "capi.ini" with the contents:

```
[LOG]
log_activity = true
log_modulesinclude = { CAPI }
```

and call CSDK_SetConfigFile:

```
     CAPIStatus stat = CSDK_SetConfigFile("capi.ini", "capi.log");
```

This will turn on "activity" level logging in the SDK and the output will go into capi.log.

CSDK_CreateSession

```
CAPIStatus CSDK_CreateSession  (  CAPIFlag    in_flags,
                                  CAPISession *    out_session
                               )
```

Create a new session.

Parameters:

- in_flags: CSDK_FLAG_NONE
- out_session: new session

Cleanup Required:

The session must be destroyed using CSDK_DestroySession().

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session constructor

CSDK_DestroySession

```
CAPIStatus CSDK_DestroySession  (  CAPISession *    io_session  )
```

Destroy a session.

Parameters:

io_session: pointer to session to destroy. Will point to CAPI_SESSION_INITIALIZER on output.

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session finalizer

CSDK_GetCapabilities

```
CAPIStatus CSDK_GetCapabilities  (  CAPISession    in_session,
                                    CAPICapabilityID    in_capabilityID,
                                    CAPIFlag    in_flags,
                                    const char **    out_value
                                  )
```

Returns information on this SDK release and / or the calendar server.

Parameters:

- in_session: session. If NULL, then no server capabilities can be requested
- in_capabilityID: ID for a capability (see CAPI_CAPAB_* in ctapi.h)
- in_flags: CSDK_FLAG_NONE at this time
- out_value: information is returned in this parameter. The values are returned as read-only strings and are only valid until the next CAPI call which uses the same session.

Changes:

CAPI 2.5: type of in_capabilityID was changed from "long" to "CAPICapabilityID"

Equivalent Java Method:

oracle.calendar.sdk.Session.getCapabilities

CSDK_Authenticate

```
CAPIStatus CSDK_Authenticate  (  CAPISession    in_session,
                                 CAPIFlag    in_flags,
                                 const char *    in_user,
                                 const char *    in_password
                               )
```

Authenticates a calendar user.

This must be done prior to making any calls to store or fetch data.

Parameters:

- in_session: session
- in_flags: Bit flags modifying behavior. This must be CSDK_FLAG_NONE currently.
- in_user: Must be a null-terminated string
- in_password: User's password. May be NULL.

Returns:

CAPIStatus

Example 3–2 Authenticate Calendar User - No Node

Connect to a server running on the default port of calserver.acme.com, to authenticate as userID keithm using default ACE settings:

(When no node is specified, either a master node or default node must be configured on the specified host.)

```
{
    CAPISession mySession = CSDK_SESSION_INITIALIZER;
    CAPIStatus myStatus = CSDK_CreateSession(&mySession);
    if (myStatus == CAPI_STAT_OK)
    {
        myStatus = CSDK_Connect(mySession, CAPI_FLAG_NONE, "calserver.acme.com");
    }
    if (myStatus == CAPI_STAT_OK)
    {
        myStatus = CSDK_Authenticate(mySession,
                                     CAPI_FLAG_NONE,
                                     "keithm",
                                     "abcdefg");
    }
}
```

Example 3–3 Authenticate Calendar User

Connect to a server running on the default port of calserver.acme.com, to authenticate as user "Keith MacDonald" using default ACE settings:

```
{
    CAPISession mySession = CSDK_SESSION_INITIALIZER;
    CAPIStatus myStatus = CSDK_CreateSession(&mySession);
    if (myStatus == CAPI_STAT_OK)
    {
        myStatus = CSDK_Connect(mySession, CAPI_FLAG_NONE, "calserver.acme.com");
    }
    if (myStatus == CAPI_STAT_OK)
    {
        myStatus = CSDK_Authenticate(mySession,
                                     CAPI_FLAG_NONE,
                                     "?/S=MacDonald/G=Keith/ND=200/",
                                     "abcdefg");
    }
}
```

Example 3–4   Authenticate Calendar User - Default Port

Connect to a server running on the default port of calserver.acme.com, to authenticate as userID ernesth on node 200 using default ACE settings:

```
{
    CAPISession mySession = CSDK_SESSION_INITIALIZER;
    CAPIStatus myStatus = CSDK_CreateSession(&mySession);
    if (myStatus == CAPI_STAT_OK)
    {
        myStatus = CSDK_Connect(mySession, CAPI_FLAG_NONE, "calserver.acme.com");
    }
    if (myStatus == CAPI_STAT_OK)
    {
        myStatus = CSDK_Authenticate(mySession,
                                     CAPI_FLAG_NONE,
                                     "ernesth?/ND=200/",
                                     "abcdefg");
    }
}
```

Example 3–5   Authenticate Calendar User - Kerberos

Connect to a server running on port 12345 of calserver.acme.com, use gssapi:kerberos5 authentication:

```
{
    CAPISession mySession = CSDK_SESSION_INITIALIZER;
    CAPIStatus myStatus = CSDK_CreateSession(&mySession);
    if (myStatus == CAPI_STAT_OK)
    {
        myStatus = CSDK_Connect(mySession, CAPI_FLAG_NONE, "calserver.acme.com:12345");
    }
    if (myStatus == CAPI_STAT_OK)
    {
        myStatus = CSDK_ConfigureACE(mySession,
                                     CAPI_FLAG_NONE,
                                     "gssapi:kerberos5",
                                     NULL,
                                     NULL);
    }
    if (myStatus == CAPI_STAT_OK)
    {
        myStatus = CSDK_Authenticate(mySession,
                                     CAPI_FLAG_NONE,
                                     "",     // don't pass in user string
                                     "");    // don't pass in password
    }
}
```

Cleanup Required:

A call to CSDK_Deauthenticate must be made between calls to CSDK_Authenticate.

Equivalent Java Method:

oracle.calendar.sdk.Session.authenticate()

CSDK_ConfigureACE

```
CAPIStatus CSDK_ConfigureACE  (  CAPISession    in_session,
                                 CAPIFlag    in_flags,
                                 const char *    in_authMechanism,
                                 const char *    in_compMechanism,
                                 const char *    in_encrMechanism
                                )
```

This function will configure the given session to use specific ACE (Authentication, Compression, Encryption) mechanisms between the SDK client and the calendar server.

If this function is not called, the default mechanisms as set on the calendar server will be used.

NULL values can be specified to select the server's default mechanism for any of the three types of mechanisms.

Parameters:

- in_session: session
- in_flags: CSDK_FLAG_NONE

- in_authMechanism: name of authentication mechanism (e.g. "cs-standard", "gssapi:kerberos5", NULL)
- in_compMechanism: name of compression mechanism (e.g. "cs-simple", "NONE", NULL)
- in_encrMechanism: name of encryption mechanism (e.g. "cs-acipher1", "NONE", NULL)

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session.configureACE()

CSDK_Connect

```
CAPIStatus CSDK_Connect  (  CAPISession    in_session,
                            CAPIFlag    in_flags,
                            const char *    in_host
                          )
```

Establish a connection with a calendar service.

Parameters:

- in_session: session
- in_flags: bit flags
- in_host: calendar server host (with optional port number; e.g. "calserver.acme.com" or "calserver.acme.com:12345") The host[:port] may optionally be followed by ?/CD=<calendar domain>/

Returns:

CAPIStatus

Example 3–6   Connect to Calendar Server

Normal usage: Connect to the calendar server calserver.acme.com. This connection can be used to authenticate as any user known to the masternode.

```
 {
     CAPIStatus  myStatus  = CAPI_STAT_OK;
```

```
    CAPISession mySession = CAPI_SESSION_INITIALIZER;
    myStatus = CSDK_CreateSession(CAPI_FLAG_NONE, &mySession);
    if (myStatus == CAPI_STAT_OK)
    {
        myStatus = CSDK_connect(mySession, CAPI_FLAG_NONE, "calserver.acme.com");
    }
}
```

Cleanup Required:

The server connection should be released by calling CSDK_Disconnect

Equivalent Java Method:

oracle.calendar.sdk.Session.connect()

CSDK_ConnectAsSysop

```
CAPIStatus CSDK_ConnectAsSysop  (  CAPISession     in_session,
                                   CAPIFlag        in_flags,
                                   const char *    in_host,
                                   const char *    in_nodeId,
                                   const char *    in_password
                                  )
```

Log on as Calendar SYSOP.

Once logged on, the Calendar SYSOP can assume the identity of any user on the same node by calling CSDK_SetIdentity().

A node must always be specified since masternode and calendar-domain functionality is not available during logon as Calendar SYSOP.

If ACE mechanisms have been configured on the session, these will be ignored. The admin default ACE settings from the calendar server will be used for all Calendar SYSOP connections.

Calendar SYSOP authentication is only available with version 5.3 and newer servers. An error will be returned if the provided host does not support this feature. A calendar server may be configured to refuse Calendar SYSOP logon via the SDK in which case a security error will be returned.

The operations available to Calendar SYSOPs are limited to:

- disconnecting by calling CSDK_Disconnect()
- switching identity to a user by calling CSDK_SetIdentity()

Once the identity has been set to a user, all operations will be performed as if that user had logged in.

See CSDK_Connect for the format of the in_host parameter.

Parameters:

- in_session: session
- in_flags: Bit flags modifying behaviour. This must be CSDK_FLAG_NONE currently.
- in_host: Calendar server host name (optional port no.)
- in_nodeId: node ID to connect to as Calendar SYSOP. Node aliases are not currently supported.
- in_password: Calendar SYSOP's password.

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session.connectAsSysop()

See also:

CSDK_SetIdentity

CSDK_Deauthenticate

```
CAPIStatus CSDK_Deauthenticate  (  CAPISession    in_session,
                                   CAPIFlag    in_flags
                                 )
```

Deauthenticate the current user.

An unauthenticated server connection is kept open and can be used to re-authenticate again. The server connection is kept open until either a call to CSDK_Disconnect() or the session is destroyed.

Parameters:

- in_session: session

- in_flags: Bit flags modifying behaviour. This must be CSDK_FLAG_NONE currently.

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session.deauthenticate()

CSDK_Disconnect

```
CAPIStatus CSDK_Disconnect  (  CAPISession    in_session,
                               CAPIFlag    in_flags
                            )
```

Disconnects from the calendar server.

Parameters:

- in_session: session
- in_flags: Bit flags modifying behaviour. This must be CSDK_FLAG_NONE currently.

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session.disconnect()

CSDK_SetIdentity

```
CAPIStatus CSDK_SetIdentity  (  CAPISession    in_session,
                                const char *    in_user,
                                CAPIFlag    in_flags
                              )
```

Allow an authenticated user to work on behalf of another calendar user or resource.

For subsequent calls to work, designate rights must have been granted to the authenticated user.

The format of the in_user parameter is the same as in the CSDK_Authenticate function. The authenticated user may revert to her original identity by using NULL as username.

If you've logged in as Calendar SYSOP (CSDK_ConnectAsSysop), then designate rights are ignored and you will be able to work as any calendar user or resource. All calendar operations will appear to have been done by the user, rather than on behalf of the user by a designate.

Parameters:

in_session: session

in_user: person (or resource) to work as - an X400 or uid

in_flags: CSDK_FLAG_NONE at this time

Returns:

CAPIStatus

Example 3–7   Work on Behalf of Another user

```
myStatus = CSDK_SetIdentity(mySession, "dantea", CAPI_FLAG_NONE);
myStatus = CSDK_SetIdentity(mySession, "?/S=Alighieri/G=Dante/", CAPI_FLAG_NONE);
myStatus = CSDK_SetIdentity(mySession, "?/RS=Conference Room/ND=1234/", CAPI_FLAG_NONE);
```

Changes:

CAPI 2.5: Resource names must be an exact match. (There used to be an implicit wildcard at the end of the string.)

SDK 9.0.4: SetIdentity can be used to work on behalf of a user on another node using designate rights. This does NOT apply to connections opened via CSDK_ConnectAsSysop().

Equivalent Java Method:

oracle.calendar.sdk.Session.setIdentity()

CSDK_DestroyHandle

```
CAPIStatus CSDK_DestroyHandle  (  CAPISession    in_session,
                                  CAPIHandle *    io_handle
                               )
```

Destroys one handle returned by CSDK_GetHandle().

Parameters:

- in_session: session handle
- io_handle: handle (returned by CSDK_GetHandle) to destroy

Returns:

CAPIStatus

Example 3–8 Destroy Handle

```
 {
       CAPIHandle h1 = CSDK_HANDLE_INITIALIZER;
       CSDK_GetHandle(mySession, "arthur", CSDK_FLAG_NONE, &h1);
       ...
       CSDK_DestroyHandle(mySession, &h1);
 }
```

Equivalent Java Method:

None. oracle.calendar.sdk.Handle finalizer will destroy handles.

CSDK_DestroyMultipleHandles

```
CAPIStatus CSDK_DestroyMultipleHandles  (  CAPISession    in_session,
                                           CAPIHandle *    io_handles,
                                           int    in_numHandles,
                                           CAPIFlag    in_flags
                                          )
```

Destroy multiple handles returned by calls to CSDK_GetHandle().

Parameters:

- in_session: login session handle
- io_handles: Array of handles (returned by CSDK_GetHandle) to destroy
- in_numHandles: The size of the handle array
- in_flags: bit flags (none at this time; set to CSDK_FLAG_NONE)

Returns:

CAPIStatus

Example 3–9 Destroy Multiple Handles

```
{
    CAPIHandle h1 = CSDK_HANDLE_INITIALIZER;
    CAPIHandle h2 = CSDK_HANDLE_INITIALIZER;
    CSDK_GetHandle(mySession, "arthur", CSDK_FLAG_NONE, &h1);
    CSDK_GetHandle(mySession, "tim...", CSDK_FLAG_NONE, &h2);
    ...
    CAPIHandle handles[] = {h1, h2};
    CSDK_DestroyMultipleHandles(mySession, handles, 2, CSDK_FLAG_NONE);
}
```

Equivalent Java Method:

None. oracle.calendar.sdk.Handle finalizer will destroy handles.

CSDK_GetHandle

```
CAPIStatus CSDK_GetHandle  (  CAPISession    in_session,
                              const char *    in_user,
                              CAPIFlag    in_flags,
                              CAPIHandle *    out_handle
                             )
```

This function returns a handle to a particular user's calendar store.

With this handle subsequent calls can access items in this agenda. If an error is returned no CAPIHandle will be allocated and no cleanup is required.

The in_user string follows the same format as that of the string used by CSDK_ Authenticate.

A handle to the current user is returned if in_user is NULL.

This function is blocked for a Calendar SYSOP that has not assumed the identity of a user.

Parameters:

- in_session: login session handle
- in_user: user as defined for CSDK_Authenticate. May be NULL in which case a handle to the current user is returned.
- in_flags: bit flags (none at this time; set to CSDK_FLAG_NONE)
- out_handle: handle for in_user. Must point to NULL on entry.

Returns:

CAPIStatus

Return values:

- CAPI_STAT_OK
- CAPI_STAT_DATA_USERID
- CAPI_STAT_SERVICE_MEM
- CAPI_STAT_SERVICE_FILE
- CAPI_STAT_SERVICE_NET
- CAPI_STAT_API_FLAGS
- CAPI_STAT_API_NULL
- CAPI_STAT_API_HANDLE
- CAPI_STAT_API_SESSION
- CAPI_STAT_LIBRARY

Cleanup Required:

This function allocates a handle which must be cleaned up with a call to CSDK_DestroyHandle. If an error is returned no handle is allocated and no clean up is required.

Example 3–10   Get Handle Using UserID

Get a handle for a user whose userID is "ivoa":

```
{
    CAPIHandle shrubber = CSDK_HANDLE_INITIALIZER;
    stat = CSDK_GetHandle(mySession, "ivoa", CAPI_FLAG_NONE, &shrubber);
}
```

Example 3–11   Get Handle with First and Last Name

Get a handle for a user named "Arnold Layne" (Surname Layne, Given name Arnold):

```
{
    CAPIHandle arnold = CSDK_HANDLE_INITIALIZER;
    stat = CSDK_GetHandle(mySession, "?/S=Layne/G=Arnold/, CAPI_FLAG_NONE, &arnold);
}
```

Example 3–12 Get Handle for Resource

Get a handle for a resource named "keg" on node "1234":

```
{
    CAPIHandle keg = CSDK_HANDLE_INITIALIZER;
    stat = CSDK_GetHandle(mySession, "?/RS=keg/ND=1234/", CAPI_FLAG_NONE, &keg);
}
```

Example 3–13 Get Handle for Current User

Get a handle for the current user:

```
{
    CAPIHandle currUser = CSDK_HANDLE_INITIALIZER;
    stat = CSDK_GetHandle(mySession, NULL, CAPI_FLAG_NONE, &currUser);
}
```

Changes:

CAPI 2.5: Resource names must be an exact match. (There used to be an implicit wildcard at the end of the string.)

Equivalent Java Method:

oracle.calendar.sdk.Session.getHandle()

CSDK_GetHandleInfo

```
CAPIStatus CSDK_GetHandleInfo  (  CAPISession    in_session,
                                  CAPIHandle    in_handle,
                                  CAPIFlag    in_flags,
                                  const char **    out_info
                                 )
```

This function returns information about the agenda of the supplied handle.

Three pieces of information can be returned, chosen by the value of in_flags. The information is returned as a pointer to a read-only string.

CAPI_HANDLE_TYPE indicates the type of the handle, this can be "user" or "resource" and indicates what type of agenda this is. CAPI_HANDLE_NAME returns the name of the agenda owner, or resource, in the form of a sequence of field-value pairs, separated by "/". This string, when prepended with a '?' is of an appropriate format to be passed to CSDK_GetHandle. A description of this format is given in "User identification" section of this manual. CAPI_HANDLE_MAILTO returns the e-mail address of who the agenda belongs to. Since not all users will have e-mail addresses set on the calendar server, an error (CAPI_STAT_DATA_EMAIL_NOTSET) will be returned when no e-mail address is set.

Parameters:

- in_session: login session handle
- in_handle: handle to get info for
- in_flags: CAPI_HANDLE_TYPE, CAPI_HANDLE_NAME or CAPI_HANDLE_MAILTO
- out_info: read-only handle information

Returns:

CAPIStatus

Changes:

CAPI 2.5: now returns CAPI_STAT_DATA_EMAIL_NOTSET if no e-mail address is set on the server.

Example 3–14 Print the Name of the Logged-in User

```
{
    CAPIHandle   loginUser = CSDK_HANDLE_INITIALIZER;
    const char * fullName = NULL;
    stat = CSDK_GetHandle(mySession, NULL, CSDK_FLAG_NONE, &loginUser);
    stat = CSDK_HandleInfo(mySession, loginUser, CAPI_HANDLE_NAME, &fullName);
    cout << "Currently logged in as " << fullName << endl;
    CSDK_DestroyHandle(mySession,
                       &loginUser);
}
```

Example 3–15 Print E-mail Address

Print out Doctor Winston's e-mail address:

```
{
    CAPIHandle   doctor = CSDK_HANDLE_INITIALIZER;
    const char * email  = NULL;
    stat = CAPI_GetHandle(mySession, "drwinston", CSDK_FLAG_NONE, &doctor);
    stat = CAPI_HandleInfo(mySession, doctor, CAPI_HANDLE_MAILTO, &email);
    cout << "drwinston's email address is " << email << endl;
    CSDK_DestroyHandle(mySession,
                       &doctor);
}
```

Equivalent Java Methods:

- oracle.calendar.sdk.Session.Handle.getEmail()
- oracle.calendar.sdk.Session.Handle.getName()
- oracle.calendar.sdk.Session.Handle.getType()

CSDK_CreateCallbackStream

```
CAPIStatus CSDK_CreateCallbackStream  (  CAPISession     in_session,
                                         CAPIStream *    out_stream,
                                         CAPICallback    in_sendCallback,
                                         void *          in_sendUserData,
                                         CAPICallback    in_recvCallback,
                                         void *          in_recvUserData,
                                         CAPIFlag        in_flags
                                        )
```

A callback stream can be used to either supply data to, or receive data from the SDK.

C function pointers are given to the SDK for each action (send, receive) that the SDK will call to either read or send data.

During a CSDK_Store...() call, the SDK will call the function in_sendCallback passing in the value in_sendUserData (which is typically used to store some context to be used by the callback function).

During a CSDK_Fetch...() call, the SDK will call the function in_recvCallback passing in the value in_recvUserData (which is typically used to store some context to be used by the callback function).

Both types of callback functions use the same function signature:

```
typedef int (*CAPICallback)(
     void *      in_userData,    // user-defined data (the value supplied in CAPI_CreateCallbackStream)
     char *      io_data,        // buffer to read or write
     size_t      in_dataSize,    // the number of characters to read or write
     size_t *    out_datSize);   // the number of characters read or written
```

The return values from the callbacks must be:

- Send callback:
    - CAPI_CALLBACK_CONTINUE: there is more data to be read from the stream

- CAPI_CALLBACK_DONE: there is no more data to be read from the stream
- a positive integer: an error has occurred. This positive integer will be returned as part of the CAPIStatus returned in bit 5 with the value CAPI_STAT_API_CALLBACK_ERROR

- Receive callback:
    - CAPI_CALLBACK_CONTINUE: no error
    - a positive integer: an error has occurred (e.g. the stream cannot receive any more data). This positive integer will be returned as part of the CAPIStatus returned in bit 5 with the value CAPI_STAT_API_CALLBACK_ERROR

When the SDK has finished writing data to the receive callback, the callback will be called with in_dataSize == 0.

In many applications, it is easier to use either a memory stream, or file stream than a callback stream.

Parameters:

- in_session: login session handle
- out_stream: on output, will point to new stream.
- in_sendCallback: send data callback
- in_sendUserData: a value which will be passed to in_sendCallback
- in_recvCallback: receive data callback
- in_recvUserData: a value which will be passed to in_recvCallback
- in_flags: bit flags (must be CSDK_FLAG_NONE at this time)

Cleanup Required:

The stream returned by this function must be destroyed by calling CSDK_DestroyStreams()

Returns:

CAPIStatus

Return values:

CAPI_STAT_API_NULL: both supplied callbacks were NULL

See also:

CSDK_CreateMemoryStream, CSDK_CreateFileStreamFromFilenames

Equivalent Java Method:

None. The Java APIs only use String and StringBuffer objects to send and receive data.

CSDK_CreateFileStreamFromFilenames

```
CAPIStatus CSDK_CreateFileStreamFromFilenames  (  CAPISession     in_session,
                                                  CAPIStream *    out_stream,
                                                  const char *    in_readFileName,
                                                  const char *    in_readMode,
                                                  const char *    in_writeFileName,
                                                  const char *    in_writeMode,
                                                  CAPIFlag        in_flags
                                                 )
```

Creates file stream to allow the SDK to read from or write to files.

Parameters:

- in_session: login session handle
- out_stream: in output, will point to new stream
- in_readFileName: name of file to read from
- in_readMode: mode to pass to fopen() while opening in_readFileName
- in_writeFileName: name of file to write to
- in_writeMode: mode to pass to fopen() while opening in_writeFileName
- in_flags: bit flags (must be CSDK_FLAG_NONE at this time)

Cleanup Required:

The stream returned by this function must be destroyed by calling CSDK_DestroyStreams.

Returns:

CAPIStatus

Return values:

- CAPI_STAT_SERVICE_FILE_MODE: an invalid mode was passed in

- CAPI_STAT_SERVICE_FILE_OPEN: failed to open a file

Example 3–16 Store Events from the File "events.ics"

```
CAPIStream myInputStream = NULL;
CAPIStatus status = CSDK_CreateFileStreamFromFilenames(mySession,
                                                       &myInputStream,
                                                       "events.ics",
                                                       "rb",
                                                       NULL,    // no output file
                                                       NULL,    // no output file mode
                                                       CSDK_FLAG_NONE);
if (status == CAPI_STAT_OK)
{
    status = CAPI_StoreEvent(mySession,
                             myHandles,
                             numHandles,
                             handleStatus,
                             CAPI_STORE_REPLACE,
                             myInputStream);
    CSDK_DestroyStreams(mySession,
                        &myInputStream,
                        1,
                        CSDK_FLAG_NONE);
}
```

Example 3–17 Fetch Events and Write Them to the File "myAgenda.ics"

```
CAPIStream myOutputStream = NULL;
CAPIStatus status = CSDK_CreateFileStreamFromFilenames(mySession,
                                                       &myOutputStream,
                                                       NULL,  // no input file
                                                       NULL,   // no input file mode
                                                       "myAgenda.ics",
                                                       "wb",
                                                       CSDK_FLAG_NONE);
if (status == CAPI_STAT_OK)
{
    status = CAPI_FetchEventsByRange(mySession,
                                     myHandles,
                                     numHandles,
                                     handleStatus,
                                     CAPI_FLAG_NONE,
                                     "20020722T000000",
                                     "20020722T235900",
                                     NULL,
                                     0,
                                     myOutputStream);
    CSDK_DestroyStreams(mySession,
                                       &myOutputStream,
                                       1,
                                       CSDK_FLAG_NONE);
   }
```

Equivalent Java Method:

None. The Java APIs only use String and StringBuffer objects to send and receive data.

CSDK_CreateMemoryStream

```
CAPIStatus CSDK_CreateMemoryStream  (  CAPISession    in_session,
                                       CAPIStream *    out_stream,
                                       const char *    in_readBuffer,
                                       const char **    out_writeBufferPtr,
                                       CAPIFlag    in_flags
                                      )
```

A memory stream uses data buffers to pass data between your application and the SDK.

This is often the simplest type of stream to use.

Read buffers are read by the SDK during CSDK_Store... calls and write buffers are written to by the SDK during CSDK_Fetch... calls. The read buffers are managed by your application, whereas the SDK will allocate and free the write buffers. The write buffer is freed by the SDK when the memory stream is destroyed.

Parameters:

- in_session: session
- out_stream: on output, will point to new stream.
- in_readBuffer: buffer for the SDK to read from
- out_writeBufferPtr: This address will point to the buffer the SDK is writing into.
- in_flags: bit flags (must be CSDK_FLAG_NONE at this time)

Cleanup Required:

The stream returned by this function must be destroyed by calling CSDK_DestroyStream.

Returns:

CAPIStatus

Return values:

CAPI_STAT_API_NULL: both supplied buffers were NULL

Example 3–18 Store Events from the Buffer "events"

```
const char events[] = "MIME-Version: 1.0\r\n"\
                      "Content-Type: text/calendar\r\n"\
                      "Content-Transfer-Encoding: quoted-printable\r\n\r\n"\
                      "BEGIN:VCALENDAR\r\n"\
                      "VERSION:2.0\r\n"\
                      ...etc
                      "END:VCALENDAR\r\n";
CAPIStream myInputStream = NULL;
CAPIStatus status = CSDK_CreateMemoryStream(mySession,
                                            &myInputStream,
                                            events,
                                            NULL, // no write buffer
                                            CSDK_FLAG_NONE);
if (status == CAPI_STAT_OK)
{
    status = CAPI_StoreEvent(mySession,
                             myHandles,
                             numHandles,
                             handleStatus,
                             CAPI_STORE_REPLACE,
                             myInputStream);
    CSDK_DestroyStreams(mySession,
                        &myInputStream,
                        1,
                        CSDK_FLAG_NONE);
}
```

Example 3–19 Fetch Events and Write Them to a Buffer

```
const char * todaysEvents = NULL;
CAPIStream myOutputStream = NULL;
CAPIStatus status = CSDK_CreateMemoryStream(mySession,
                                            &myOutputStream,
                                            NULL, // no read buffer
                                            &todaysEvents,
                                            CSDK_FLAG_NONE);
if (status == CAPI_STAT_OK)
{
    status = CAPI_FetchEventsByRange(mySession,
                                     myHandles,
                                          numHandles,
                                          handleStatus,
                                          CAPI_FLAG_NONE,
                                          "20020722T000000",
                                          "20020722T235900",
                                          NULL,
                                          0,
                                          myOutputStream);
    if (status == CAPI_STAT_OK)
    {
        printf("Today's events:%s", todaysEvents);
    }
    CSDK_DestroyStreams(mySession,
                        &myOutputStream,
                        1,
                        CAPI_FLAG_NONE);
}
```

CSDK_DestroyMultipleStreams

```
CAPIStatus CSDK_DestroyMultipleStreams  (  CAPISession    in_session,
                                           CAPIStream *    in_streams,
                                           int    in_numStreams,
                                           CAPIFlag    in_flags
                                          )
```

This function destroys streams created by the various CSDK_Create*Stream functions.

Parameters:

- in_session: the session to which streams are associated.
- in_streams: array of streams to destroy
- in_numStreams: the number of streams in in_streams to destroy
- in_flags: bit flags modifying behavior. Must be CSDK_FLAG_NONE at this time.

Returns:

CAPIStatus

Equivalent Java Method:

None. The Java APIs only use String and StringBuffer objects to send and receive data.

CSDK_DestroyStream

```
CAPIStatus CSDK_DestroyStream  (  CAPISession    in_session,
                                  CAPIStream *    io_stream
                               )
```

This function destroys a stream created by any of the various CSDK_Create*Stream functions.

Parameters:

- in_session: the session to which streams are associated.
- io_stream: stream to destroy.

Returns:

CAPIStatus

Equivalent Java Method:

None. The Java APIs only use String and StringBuffer objects to send and receive data.

CSDK_DeleteContacts

```
CAPIStatus CSDK_DeleteContacts  (  CAPISession           in_session,
                                   CAPIFlag              in_flags,
                                   CAPIUIDSet            in_UIDs,
                                   CSDKRequestResult *   out_requestResult
                                )
```

Deletes vCards specified by a set of UIDs.

This function is blocked for a Calendar SYSOP that has not assumed the identity of a user.

Parameters:

- in_session: login session handle
- in_flags: CSDK_FLAG_NONE
- in_UIDs: NULL terminated array of strings containing UIDs of vCards to delete
- out_requestResult: If non-NULL, will be filled in with detailed results of the transaction.

Returns:

CAPIStatus

Return values:

- CAPI_STAT_API_HANDLE_NULL: the session was NULL
- CAPI_STAT_API_NULL: in_UIDs was NULL

Equivalent Java Method:

oracle.calendar.sdk.Session.deleteContacts()

CSDK_FetchContactsByQuery

```
CAPIStatus CSDK_FetchContactsByQuery  (  CAPISession           in_session,
                                         CAPIFlag              in_flags,
                                         CSDKQuery             in_query,
                                         const char **         in_requestProperties,
                                         CAPIStream            in_stream,
                                         CSDKRequestResult *   out_requestResult
                                        )
```

This function fetches contacts which satisfy the conditions specified in the query.

The returned vCards are returned via in_sendStream, and by default will be in MIME format. Each fetched vCard will be in a separate MIME part. The character set will be UTF-8.

To avoid having the stream MIME-encapsulated, pass in the flag CSDK_FLAG_STREAM_NOT_MIME.

Parameters:

- in_session: session handle
- in_flags: bit flags:
  - CSDK_FLAG_NONE
  - CSDK_FLAG_STREAM_NOT_MIME: do NOT wrap the output in a MIME container
- in_query: a query object containing the search criteria
- in_requestProperties: To fetch only specific vCard properties of the contacts, pass in an array of property names. The array MUST be terminated with either

a NULL (zero) pointer, or an empty (length zero) string. An empty array (i.e. NULL or an empty string), will cause all available properties to be returned.

- in_stream: stream for the SDK to write into
- out_requestResult: If non-NULL, will be filled in with detailed results of the transaction.

Cleanup Required:

The request result must be destroyed using CSDK_DestroyRequestResult

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session.fetchContactsByQuery()

CSDK_FetchContactsByUID

```
CAPIStatus CSDK_FetchContactsByUID  (  CAPISession         in_session,
                                       CAPIFlag            in_flags,
                                       CAPIUIDSet          in_UIDs,
                                      const char **       in_requestProperties,
                                       CAPIStream          in_stream,
                                       CSDKRequestResult * out_requestResult
                                      )
```

Fetches vCards from an authenticated connection through in_session.

The fetched vCards are returned via in_sendStream, and will be in MIME format. Each vCard fetched vCard will be in a separate MIME part. The character set will be UTF-8.

To avoid having the stream MIME-encapsulated, pass in the flag CSDK_FLAG_STREAM_NOT_MIME.

This function is blocked for a Calendar SYSOP that has not assumed the identity of a user.

Parameters:

- in_session: login session handle
- in_flags: bit flags:

    - CSDK_FLAG_NONE
    - CSDK_FLAG_STREAM_NOT_MIME: do NOT wrap the output in a MIME container
- in_UIDs: NULL terminated array of strings containing UIDs of vCards to fetch
- in_requestProperties: To fetch only specific vCard properties of the contacts, pass in an array of property names. The array MUST be terminated with either a NULL (zero) pointer, or an empty (length zero) string. An empty array (i.e. NULL or an empty string), will cause all available properties to be returned.
- in_stream: stream for the SDK to write into
- out_requestResult: If non-NULL, will be filled in with detailed results of the transaction.

Cleanup Required:
The result must be destroyed using CSDK_DestroyRequestResult

Returns:
CAPIStatus

Return values:

- CAPI_STAT_API_HANDLE_NULL: the session was NULL
- CAPI_STAT_API_STREAM_NULL: the stream was NULL
- CAPI_STAT_API_NULL: in_UIDSet was NULL
- CAPI_STAT_API_BADPARAM: in_UIDCount was 0

Equivalent Java Method:
oracle.calendar.sdk.Session.fetchContactsByUID()

CSDK_StoreContacts

```
CAPIStatus CSDK_StoreContacts  (  CAPISession           in_session,
                                  CAPIFlag              in_flags,
                                  CAPIStream            in_stream,
                                  CSDKRequestResult *   out_requestResult
                                )
```

Stores vCards on a server through an authenticated connection by in_session.

The vCards must be passed in via a CAPIStream.

By default, the incoming stream is assumed to be MIME-encapsulated vCard. When storing a stream that is not MIME-encapsulated, specify the flag CSDK_FLAG_STREAM_NOT_MIME.

Versions 2.1 and 3.0 of vCard are supported.

When storing multiple vCards, every vCard must be in a separate MIME part; any MIME part containing a vCard to be stored must contain the 'Content-Type: text/x-vcard' header. The only supported character sets for the MIME parts are UTF-8 and US-ASCII.

There are several modes that can be used:

- CSDK_FLAG_STORE_REPLACE: This completely replaces a vCard that already exists on the server. It reads the UID contained within the UID property of the given vCard, removes that contact from the server, and stores the new one. An error is returned if no contact with the given UID exists on the server.
- CSDK_FLAG_STORE_MODIFY: Updates a contact already on the server with the new vCard. The vCard with that UID is then updated: all properties contained in the vCard on the server that are present in the passed-in vCard are modified to contain the property values of the passed-in vCard. Also, all properties that exist in the passed-in vCard that don't exist on the server vCard are added to the server vCard. All other properties not present in the passed-in vCard that exist on the server are ignored.
- CSDK_FLAG_STORE_REMOVE: The contact on the server is fetched, the properties contained within the passed-in vCard are deleted from the fetched vCard, and then the fetched vCard is stored on the server.
- CSDK_FLAG_STORE_IMPORT: This mode checks if the contact already exists on the server via the UID. If it does, then it acts exactly as if CSDK_FLAG_STORE_REPLACE was passed in. Otherwise, it acts exactly like as if CSDK_FLAG_STORE_CREATE was passed in.

Notes:

The flags supplied are used for each vCard supplied. Results are written to the CSDKRequestResult.

This version of the SDK cannot preserve the supplied UIDs when adding contacts to the server. The CSDKRequestResult contains the server-assigned UIDs which can be used to refer to the stored vCards.

The CSDKRequestResult may contain information about errors parsing the vCard streams.

This function is blocked for a Calendar SYSOP that has not assumed the identity of a user.

Parameters:

- in_session: the session
- in_flags: flags modifying behavior:
  - CSDK_FLAG_STORE_CREATE: create if no contact with the given UID exists, otherwise return an error
  - CSDK_FLAG_STORE_REPLACE: completely replace contact on server with this copy - error if object doesn't exist
  - CSDK_FLAG_STORE_IMPORT: if contact exists, CSDK_FLAG_STORE_ REPLACE else CSDK_FLAG_STORE_CREATE
  - CSDK_FLAG_STORE_MODIFY: modify only the supplied properties of an existing contact
  - CSDK_FLAG_STORE_REMOVE: remove specified properties
  - CSDK_FLAG_STREAM_NOT_MIME: incoming stream is not inside a MIME wrapper
- in_stream: the stream contained vCards
- out_requestResult: pointer to a RequestResult that will get filled (pass NULL if you don't want this information returned).

Returns:

CAPIStatus

Return values:

- CAPI_STAT_API_HANDLE_NULL: the session was NULL
- CAPI_STAT_API_STREAM_NULL: the stream was NULL
- CAPI_STAT_DATA_VCARD_DUPERROR: tried to store vCard in CSDK_ FLAG_STORE_CREATE mode that already exists on the server.
- CAPI_STAT_DATA_UID_NOTFOUND: tried to update, replace, or delete properties but no UID was found in the passed-in vCard.

- CAPI_STAT_API_BADPARAM: invalid flag, or multiple flags were set

Example 3–20 Store a Contact

```
stat = CSDK_StoreContacts(mySession,
                          myStream,
                          CSDK_FLAG_STORE_IMPORT,
                          &myRequestResult);
```

Example 3–21 Store a Contact with vCard

```
strcpy(outVCard,
"MIME-Version: 1.0\015\012\
Content-Type: multipart/mixed;\015\012\
boundary=\"------------CA94974D4D8713DE5B12E6CD\"\015\012\
\015\012\
This is a multi-part message in MIME format.\015\012\
--------------CA94974D4D8713DE5B12E6CD\015\012\
Content-Type: text/x-vcard; charset=UTF-8;\015\012\
name=\"example.vcf\"\015\012\
Content-Disposition: attachment;\015\012\
filename=\"example.vcf\"\015\012\
Content-Transfer-Encoding: quoted-printable\015\012\
\015\012\
BEGIN:VCARD\015\012\
URL:http://www.somewebsite.com\015\012\
ORG:oracle;windows;\015\012\
TITLE:worker\015\012\
EMAIL;TYPE=INTERNET:someone@somewhere.com\015\012\
ADR;TYPE=WORK:;;who knows;snodown;qc;h1l 2H1;Canada\015\012\
NOTE;ENCODING=QUOTED-PRINTABLE;:This is a note\015\012\
N;ENCODING=QUOTED-PRINTABLE:Last;First;Middle\015\012\
FN;ENCODING=QUOTED-PRINTABLE:First Middle Last\015\012\
REV:20011105T145136Z\015\012\
VERSION:2.1\015\012\
END:VCARD\015\012\
\015\012\
--------------CA94974D4D8713DE5B12E6CD--\015\012\015\012");
//
CAPIStatus stat;
//
stat = CSDK_CreateMemoryStream(mySession,
                               &myStream,
                               outVCard,
                               NULL,
                               CAPI_FLAG_NONE);
//
stat = CSDK_StoreContacts(mySession,
                          myStream,
                                              CSDK_FLAG_STORE_IMPORT,
                                              &myRequestResult);
```

Cleanup Required:

The request result must be destroyed using CSDK_DestroyRequestResult

Equivalent Java Method:

oracle.calendar.sdk.Session.storeContacts()

CSDK_DeleteEvents

```
CAPIStatus CSDK_DeleteEvents  (  CAPISession           in_session,
                                 CAPIFlag              in_flags,
                                 CAPIUIDSet            in_UIDs,
                                 const char *          in_RECURRENCEID,
                                 int                   in_modifier,
                                 CSDKRequestResult *   out_requestResult
                                )
```

This function deletes specified events.

Must be acting as the event owner for this to succeed. Unlike CAPI_DeleteEvent(), this does not "uninvite" attendees - it deletes the event. Individual (or a range) of instances can be deleted using {in_RECURRENCEID, in_modifier} - but only a single UID can be used in this case.

Parameters:

- in_session: login session handle
- in_flags: bit flags
- in_UIDs: An array of strings containing the UID(s) of the event(s) to delete. The array MUST be terminated with either a NULL (zero) pointer, or an empty (length zero) string.
- in_RECURRENCEID: To delete ALL occurrences of an event, pass in NULL or an empty string. To delete individual (or a range of) occurrences, specify an iCalendar recurrence-id in either DATE or DATE-TIME format which identifies one occurrence of the event.

- in_modifier: When a recurrence-id is specified using in_RECURRENCEID, this modifier determines whether the specified occurrences, or a range of occurrences will be deleted. Values are:
  - CAPI_THISINSTANCE
  - CAPI_THISANDPRIOR
  - CAPI_THISANDFUTURE
- out_requestResult: If non-NULL, will be filled in with detailed results of the transaction

Cleanup Required:

The request result must be destroyed using CSDK_DestroyRequestResult

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session.deleteEvents()

CSDK_FetchEventsByAlarmRange

```
CAPIStatus CSDK_FetchEventsByAlarmRange  (  CAPISession    in_session,
                                            CAPIFlag    in_flags,
                                            CAPIHandle *    in_agendas,
                                            const char *    in_start,
                                            const char *    in_end,
                                            const char **    in_requestProperties,
                                            CAPIStream    in_stream,
                                            CSDKRequestResult *    out_requestResult
                                           )
```

This function fetches events which have alarms (reminders) which will trigger within the time range specified.

The end of the time range is exclusive.

Parameters:

- in_session: login session handle
- in_flags: bit flags:
  - CSDK_FLAG_NONE: default behaviour

- CSDK_FLAG_FETCH_AGENDA_ATTENDEE_ONLY: only get/return ATTENDEE for the agenda being viewed
- CSDK_FLAG_FETCH_COMBINED: return all events in one VCALENDAR rather than one VCALENDAR per agenda. This is faster.
- CSDK_FLAG_FETCH_EXCLUDE_HOLIDAYS: do not fetch holidays
- CSDK_FLAG_FETCH_EXCLUDE_DAILYNOTES: do not fetch daily notes
- CSDK_FLAG_FETCH_EXCLUDE_DAYEVENTS: do not fetch day events
- CSDK_FLAG_FETCH_EXCLUDE_APPOINTMENTS: do not fetch appointments
- CSDK_FLAG_FETCH_EXCLUDE_ACCEPTED: do not fetch accepted events
- CSDK_FLAG_FETCH_EXCLUDE_DECLINED: do not fetch declined events
- CSDK_FLAG_FETCH_EXCLUDE_UNCONFIRMED: do not fetch unconfirmed events
- CSDK_FLAG_FETCH_LOCALTIMES: return all dates & times in the user's time zone (the user's time zone preference stored on the calendar server)
- CSDK_FLAG_FETCH_DO_NOT_EXPAND_RRULE: do not expand recurrence rules. This will cause the entire event to be returned instead of only the instances which have alarms scheduled to trigger during the range.
- CSDK_FLAG_STREAM_NOT_MIME: do not wrap the iCalendar in a MIME container

- in_agendas: The agenda(s) on which to search for events. This parameter is an array of CAPIHandles, and MUST be terminated with a zero value (CAPI_HANDLE_INITIALIZER). A NULL value, or an empty array will search on the current user's agenda.
- in_start: Beginning of date/time range. May be of any of the following forms:
  - DATE. e.g. 20020928
  - DATE-TIME. Must be in either floating (e.g. 20020929T120000) or UTC time (e.g. 20020929T170000Z). Floating time uses the user's time zone (the user's time zone preference stored on the calendar server).

- DURATION. A relative date or date-time expressed using the duration notation (e.g. +PT5DT3H 5 days 3 hours in the future, -PT1W 1 week in the past)

- in_end: End of date/time range. May be in any of the formats shown for in_start
- in_requestProperties: To fetch only specific iCalendar properties of the events, pass in an array of property names. The array MUST be terminated with either a NULL (zero) pointer, or an empty (length zero) string. An empty array (i.e. NULL or an empty string), will cause all available properties to be returned.
- in_stream: stream for the SDK to write into
- out_requestResult: If non-NULL, will be filled in with detailed results of the transaction.

Cleanup Required:

The request result must be destroyed using CSDK_DestroyRequestResult

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session.fetchEventsByAlarmRange()

CSDK_FetchEventsByRange

```
CAPIStatus CSDK_FetchEventsByRange  (  CAPISession    in_session,
                                       CAPIFlag    in_flags,
                                       CAPIHandle *    in_agendas,
                                       const char *    in_start,
                                       const char *    in_end,
                                       const char **    in_requestProperties,
                                       CAPIStream    in_stream,
                                       CSDKRequestResult *    out_requestResult
                                      )
```

This function fetches events which occur within the time range specified.

Parameters:

- in_session: login session handle

- in_flags: bit flags:
    - CSDK_FLAG_NONE: default behaviour
    - CSDK_FLAG_FETCH_AGENDA_ATTENDEE_ONLY: only get/return ATTENDEE for the agenda being viewed
    - CSDK_FLAG_FETCH_COMBINED: return all events in one VCALENDAR rather than one VCALENDAR per agenda. This is faster.
    - CSDK_FLAG_FETCH_EXCLUDE_HOLIDAYS: do not fetch holidays
    - CSDK_FLAG_FETCH_EXCLUDE_DAILYNOTES: do not fetch daily notes
    - CSDK_FLAG_FETCH_EXCLUDE_DAYEVENTS: do not fetch day events
    - CSDK_FLAG_FETCH_EXCLUDE_APPOINTMENTS: do not fetch appointments
    - CSDK_FLAG_FETCH_EXCLUDE_ACCEPTED: do not fetch accepted events
    - CSDK_FLAG_FETCH_EXCLUDE_DECLINED: do not fetch declined events
    - CSDK_FLAG_FETCH_EXCLUDE_UNCONFIRMED: do not fetch unconfirmed events
    - CSDK_FLAG_FETCH_LOCALTIMES: return all dates & times in the user's time zone (the user's time zone preference stored on the calendar server)
    - CSDK_FLAG_FETCH_DO_NOT_EXPAND_RRULE: do not expand recurrence rules. This will cause the entire event to be returned instead of only the instances which fall during the range.
    - CSDK_FLAG_STREAM_NOT_MIME: do NOT wrap the iCalendar in a MIME container
- in_agendas: The agenda(s) on which to search for events. This parameter is an array of CAPIHandles, and MUST be terminated with a zero value (CAPI_HANDLE_INITIALIZER). A NULL value, or an empty array will search on the current user's agenda.
- in_start: Beginning of date/time range. May be of any of the following forms:
    - DATE. e.g. 20020928
    - DATE-TIME. Must be in either floating (e.g. 20020929T120000) or UTC time (e.g. 20020929T170000Z). Floating time uses the user's time zone (the user's time zone preference stored on the calendar server).

  - DURATION. A relative date or date-time expressed using the duration notation (e.g. +PT5DT3H 5 days 3 hours in the future, -PT1W 1 week in the past)
- in_end: End of date/time range. May be in any of the formats shown for in_start
- in_requestProperties: To fetch only specific iCalendar properties of the events, pass in an array of property names. The array MUST be terminated with either a NULL (zero) pointer, or an empty (length zero) string. An empty array (i.e. NULL or an empty string), will cause all available properties to be returned.
- in_stream: stream for the SDK to write into
- out_requestResult: If non-NULL, will be filled in with detailed results of the transaction.

Cleanup Required:

The result must be destroyed using CSDK_DestroyRequestResult

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session.fetchEventsByRange()

CSDK_FetchEventsByUID

```
CAPIStatus CSDK_FetchEventsByUID  (  CAPISession    in_session,
                                     CAPIFlag    in_flags,
                                     CAPIHandle    in_agenda,
                                     CAPIUIDSet    in_UIDs,
                                     const char *    in_RECURRENCEID,
                                     int    in_modifier,
                                     const char **    in_requestProperties,
                                     CAPIStream    in_stream,
                                     CSDKRequestResult *    out_requestResult
                                    )
```

This function fetches events by their UIDs.

Specific instances of one event may be fetched using the in_RECURRENCEID and in_modifier parameters.

Specific properties can be requested using the in_requestProperties parameter. This parameter is a NULL(zero)-terminated or "empty string"-terminated array of C strings containing the property names to be returned. For maximum performance, limit the properties you request (particularly the ATTENDEE property) to only what you need.

Parameters:

- in_session: login session handle
- in_flags: bit flags:
  - CSDK_FLAG_NONE: default behaviour
  - CSDK_FLAG_FETCH_AGENDA_ATTENDEE_ONLY: only get/return ATTENDEE for the agenda being viewed
  - CSDK_FLAG_FETCH_LOCALTIMES: return all dates & times in the user's time zone (the user's time zone preference stored on the calendar server)
  - CSDK_FLAG_FETCH_EXPAND_RRULE: expand recurrence rules and return a set of VEVENTs — one per instance generated by the recurrence rule.
  - CSDK_FLAG_STREAM_NOT_MIME: do NOT wrap the iCalendar in a MIME container
- in_agenda: The agenda on which to search for event(s) with the given UID(s). A NULL value will search on the current user's agenda.
- in_UIDs: An array of strings containing the UID(s) of the events to fetch. The array MUST be terminated with either a NULL (zero) pointer, or an empty (length zero) string.
- in_RECURRENCEID: To fetch ALL occurrences of an event, pass in NULL or an empty string. To fetch individual (or a range of) occurrences, specify an iCalendar recurrence-id in either DATE or DATE-TIME format which identifies one occurrence of the event.
- in_modifier: When a recurrence-id is specified using in_RECURRENCEID, this modifier determines whether the specified occurrences, or a range of occurrences will be fetched. Values are:
  - CAPI_THISINSTANCE
  - CAPI_THISANDPRIOR
  - CAPI_THISANDFUTURE

- in_requestProperties: To fetch only specific iCalendar properties of the events, pass in an array of property names. The array MUST be terminated with either a NULL (zero) pointer, or an empty (length zero) string. An empty array (i.e. NULL or an empty string), will cause all available properties to be returned.
- in_stream: stream for the SDK to write into
- out_requestResult: If non-NULL, will be filled in with detailed results of the transaction.

Cleanup Required:

The result must be destroyed using CSDK_DestroyRequestResult

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session.fetchEventsByUID()

CSDK_StoreEvents

```
CAPIStatus CSDK_StoreEvents  (  CAPISession    in_session,
                                CAPIFlag    in_flags,
                                CAPIStream    in_stream,
                                CSDKRequestResult *    out_requestResult
                             )
```

This function reads one VCALENDAR object from in_stream and stores each contained VEVENT on the server.

Attendees

Unlike CAPI_StoreEvent, the only attendees of the event will be those specified using ATTENDEE properties in the iCalendar (with the exception of the case where CSDK_FLAG_STORE_INVITE_SELF is used, in which case the logged-in user will always be invited regardless of whether an ATTENDEE property is supplied for that user).

The address specified in the ATTENDEE properties is used to identify calendar users. If no calendar user exists with the address specified in the ATTENDEE property value, then the attendee is considered "external" and will be invited using an Internet standard protocol such as iMIP when the calendar server is capable of doing so.

The ATTENDEE.PARTSTAT parameter is ignored except:

- for the logged in user
- for external (non-calendar) attendees

Resources

The Oracle Calendar Server stores a PARTSTAT value for each resource, but resources do not have e-mail addresses. To permit the usage of the ATTENDEE property for inviting resources, the following syntax is supported:

```
ATTENDEE;CUTYPE=RESOURCE;CN=projecter:MAILTO:ignored@foobar.com
```

Groups

Oracle Calendar Groups can be invited using a non-standard (but legal) ATTENDEE property of the form:

```
ATTENDEE;CUTYPE=GROUP;CN=developers:MAILTO:ignored@foobar.com
```

As suggested by the example, the property value (MAILTO:ignored@foobar.com) is NOT used. In this example, the group "developers" will be expanded and the members will be invited as calendar users. When fetching this event, the members of the group (at the time of the call to CSDK_StoreEvents) will be returned as individual ATTENDEE properties.

The server does not enforce uniqueness of group names — if multiple matches are found, an error will be returned.

Errors:

Detailed error information is returned through the out_requestResult parameter. Unless a parse error result is returned, there will be at least one result per VEVENT stored, containing a CAPIStatus value for storing that VEVENT. Passing in a NULL (zero) value for out_requestResult will prevent the request results from being returned, but is not considered an error.

Recurrence rules

Recurrence rules (RRULE) are supported by this function and require that the event's DTSTART be specified in local (using a TZID=... and a VTIMEZONE component) or floating time (as per RFC 2445). A limitation of the calendar server requires that no more that one RRULE can be specified for a given VEVENT, nor can the RRULE be changed when modifying an event (the only way to change the occurrences is to use RDATEs and / or EXDATEs).

UIDs

The calendar server prevents any user/resource from owning more than one event with a given UID. However, UIDs are not neccessarly unique on the calendar server, so a user/resource may be invited to more than one event with a given UID. Users of the SDK should attempt to provide globally unique UIDs when adding events to the calendar server.

Storing an event without a UID will result in a new UID being generated by the calendar server and there will be a small performance penalty. The generated UIDs are returned as part of the results in out_requestResult.

How do I...

- **Add instances to an event?** For recurring events (which use RRULEs), simply store a VEVENT with the event's UID and one or more RDATE properties using the flag CSDK_FLAG_STORE_MODIFY. For repeating events (not using RRULEs), store a VEVENT with the event's UID.
- **Remove instances from an event?** Store VEVENT(s) specifying the event's UID and EXDATE(s) for instances to remove. The EXDATE values must match exactly with the instance's RECURRENCE-ID.
- **Add attendees to an event?** Store VEVENT(s) with ATTENDEE properties for new attendees using flag CSDK_FLAG_STORE_MODIFY. RECURRENCE-ID property can be specified in the VEVENT to invite the attendee to only the specified instance.
- **Remove attendees from an event?** Fetch event, remove the ATTENDEE property for the user to uninvite, then store using the mode CSDK_FLAG_STORE_REPLACE.

Parameters:

- in_session: login session handle
- in_flags: Flags modifying behavior. One of:
  - CSDK_FLAG_STORE_CREATE: create if no event with the given UID exists, otherwise return an error
  - CSDK_FLAG_STORE_REPLACE: completely replace event on server with this copy — error if object doesn't exist
  - CSDK_FLAG_STORE_IMPORT: if event exists, CSDK_FLAG_STORE_REPLACE else CSDK_FLAG_STORE_CREATE

  - CSDK_FLAG_STORE_MODIFY: modify only the supplied properties of an existing event
  - CSDK_FLAG_STORE_REPLY: any attendee of an event can use this mode to update their own attendance status and alarms.

  And optionally, a combination of:

  - CSDK_FLAG_STORE_INVITE_SELF: add current user as an attendee, even if no ATTENDEE is in the iCal
  - CSDK_FLAG_STREAM_NOT_MIME: incoming stream is not inside a MIME wrapper
  - CAPI_NOTIFY_EMAIL: send e-mail notification (default is to NOT send)
  - CAPI_NOTIFY_SMS: send SMS notification (default is to NOT send)
- in_stream: stream for the SDK to read data from
- out_requestResult: If non-NULL, will be filled in with detailed results of the transaction. This may include error messages from reading the iCalendar data or any other errors encountered while processing the request.

Returns:

CAPIStatus

Cleanup Required:

The request result must be destroyed using CSDK_DestroyRequestResult

Example 3–22 Add an Event to the Current User's Calendar

```
{
    static const char * ical = {"BEGIN:VCALENDAR\r\n"
                                "VERSION:2.0\r\n"
                                "BEGIN:VEVENT\r\n"
                                "DTSTART:20021225T100000Z\r\n"
                                "DTEND:20021225T233000Z\r\n"
                                "SUMMARY:work\r\n"
                                "LOCATION:office\r\n"
                                "END:VEVENT\r\n"
                                "END:VCALENDAR\r\n"};
    //
    CAPIStream memoryStream = CSDK_STREAM_INITIALIZER;
    status = CSDK_CreateMemoryStream(mySession,
                                     &memoryStream,
                                     ical,
                                       NULL,
                                       CSDK_FLAG_NONE);
      if (!status)
      {
          status = CSDK_StoreEvents(mySession,
                                    CSDK_FLAG_STORE_CREATE | CSDK_FLAG_STORE_INVITE_SELF | CSDK_FLAG_STREAM_
NOT_MIME,
                                    memoryStream,
                                    NULL);
          //
          CSDK_DestroyStream(mySession,
                             &memoryStream);
      }
  }
```

Example 3–23 Invite People to a Meeting

```
{
      // invite these people to a meeting:
      const char * attendees[] = {"?/S=Who/G=Cindy Lou/",
                                  "?/S=Who/G=Lou Lou/",
                                  "?/S=Who/G=Betty Lou/"
                                  "grinch");
      //
      const int     numAttendees   = (sizeof(attendees) / sizeof(attendees[0]));
      const char ** emailAddresses = (const char **)malloc((numAttendees + 1) * sizeof(const char *));
      CAPHandle  *  handles        = (CAPHandle *)malloc(numAttendees * sizeof(CAPHandle *));
      //
      // get handles:
      for (int i = 0; !status && (i < numAttendees); i++)
      {
          status = CSDK_GetHandle(mySession, attendees[i], CSDK_FLAG_NONE, &handles[i]);
      }
      //
      // terminate the array:
      handles[numAttendees] = CSDK_HANDLE_INITIALIZER;
      //
      if (!status)
      {
          // get e-mail addresses for each handle using CSDK_GetHandleInfo()
          ...
      }
      //
      if (!status)
      {
          static const char * iCalEvent1 = {"BEGIN:VCALENDAR\r\n"
                                            "VERSION:2.0\r\n"
                                            "BEGIN:VEVENT\r\n"
                                            "DTSTART:20021225T100000Z\r\n"
                                  "DTEND:20021225T103000Z\r\n"
                                  "SUMMARY:SING\r\n"
                                  "LOCATION:Whoville town square\r\n"};
    //
    static const char * iCalEvent2 = {"END:VEVENT\r\n"
                                  "END:VCALENDAR\r\n"};
    //
    string iCalEvent = iCalEvent1;
    for (int attendee = 0; attendee < numAttendees; attendee++)
    {
        if (emailAddresses[attendee])
        {
            iCalEvent += "ATTENDEE:mailto:";
            iCalEvent += emailAddresses[attendee];
            iCalEvent += "\r\n";
        }
    }
    iCalEvent += iCalEvent2;
}
//
if (emailAddresses)
{
    free(emailAddresses);
}
//
CSDK_DestroyMultipleHandles(mySession,
                            handles,
                            numAttendees,
                            CSDK_FLAG_NONE);
if (handles)
{
    free(handles);
}
//
if (!status)
{
    CAPIStream memoryStream = CSDK_STREAM_INITIALIZER;
    status = CSDK_CreateMemoryStream(mySession,
                                     &memoryStream,
                                     iCalEvent.c_str(),
                                     NULL,
                                     CSDK_FLAG_NONE);
    if (!status)
    {
        status = CSDK_StoreEvents(mySession,
                                  CSDK_FLAG_STORE_CREATE | CSDK_FLAG_STREAM_NOT_MIME,
                                  memoryStream,
                                  NULL);
    }
```

```
        //
        CSDK_DestroyStream(mySession,
                           &memoryStream);
    }
}
```

Equivalent Java Method:

oracle.calendar.sdk.Session.storeEvents

CSDK_DeleteTasks

```
CAPIStatus CSDK_DeleteTasks  (  CAPISession          in_session,
                                CAPIFlag             in_flags,
                                CAPIUIDSet           in_UIDs,
                                CSDKRequestResult *  out_requestResult
                              )
```

This function deletes tasks on the current user's agenda.

This function is blocked for a Calendar SYSOP that has not assumed the identity of a user.

Parameters:

- in_session: session
- in_flags: CSDK_FLAG_NONE
- in_UIDs: NULL terminated array of task UIDs
- out_requestResult: returned request result object

Returns:

- CAPIStatus

Return values:

- CAPI_STAT_OK
- CAPI_STAT_API_SESSION_NULL: in_session is NULL
- CAPI_STAT_API_NULL: in_UIDSet is NULL

Equivalent Java Method:

oracle.calendar.sdk.Session.deleteTasks()

CSDK_FetchTasksByAlarmRange

```
CAPIStatus CSDK_FetchTasksByAlarmRange  (  CAPISession    in_session,
                                           CAPIFlag    in_flags,
                                           CAPIHandle *    in_handles,
                                           const char *    in_start,
                                           const char *    in_end,
                                           const char **    in_requestProperties,
                                           CAPIStream    in_stream,
                                           CSDKRequestResult *    out_requestResult
                                          )
```

This function fetches tasks which have alarms (reminders) which will trigger within the time range specified.

The end of the time range is exclusive.

Parameters:

- in_session: login session handle
- in_flags: bit flags:
  - CSDK_FLAG_NONE
  - CSDK_FLAG_STREAM_NOT_MIME : do NOT wrap the iCalendar in a MIME container
- in_handles: The agenda(s) on which to search for tasks. This parameter is an array of CAPIHandles, and MUST be terminated with a zero value (CSDK_HANDLE_INITIALIZER). A NULL value, or an empty array will search on the current user's agenda.
- in_start: Beginning of date/time range. May be of any of the following forms:
  - DATE. e.g. 20020928
  - DATE-TIME. Must be in either floating (e.g. 20020929T120000) or UTC time (e.g. 20020929T170000Z)
  - DURATION. A relative date or date-time expressed using the duration notation (e.g. +PT5DT3H 5 days 3 hours in the future, -PT1W 1 week in the past)
- in_end: End of date/time range. May be in any of the formats shown for in_start
- in_requestProperties: To fetch only specific iCalendar properties of the tasks, pass in an array of property names. The array MUST be terminated with either

a NULL (zero) pointer, or an empty (length zero) string. An empty array (i.e. NULL or an empty string), will cause all available properties to be returned.

- in_stream: stream for the SDK to write into
- out_requestResult: If non-NULL, will be filled in with detailed results of the transaction.

Cleanup Required:

The request result must be destroyed using CSDK_DestroyRequestResult

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session.fetchTasksByAlarmRange()

CSDK_FetchTasksByRange

```
CAPIStatus CSDK_FetchTasksByRange  (  CAPISession          in_session,
                                      CAPIFlag             in_flags,
                                      CAPIHandle *         in_handles,
                                      const char *         in_start,
                                      const char *         in_end,
                                     const char **        in_requestProperties,
                                      CAPIStream           in_stream,
                                      CSDKRequestResult *  out_requestResult
                                     )
```

This function fetches tasks which are active within the time range specified.

The end of the time range is exclusive.

Parameters:

- in_session: login session handle
- in_flags: bit flags:
  - CSDK_FLAG_NONE
  - CSDK_FLAG_FETCH_NO_MIME: do NOT wrap the iCalendar in a MIME container

- in_handles: The agenda(s) on which to search for tasks. This parameter is an array of CAPIHandles, and MUST be terminated with a zero value (CSDK_HANDLE_INITIALIZER). A NULL value, or an empty array will search on the current user's agenda.
- in_start: Beginning of date/time range. May be of any of the following forms:
  - DATE. e.g. 20020928
  - DATE-TIME. Must be in either floating (e.g. 20020929T120000) or UTC time (e.g. 20020929T170000Z)
  - DURATION. A relative date or date-time expressed using the duration notation (e.g. +PT5DT3H 5 days 3 hours in the future, -PT1W 1 week in the past)
- in_end: End of date/time range. May be in any of the formats shown for in_start
- in_requestProperties: To fetch only specific iCalendar properties of the tasks, pass in an array of property names. The array MUST be terminated with either a NULL (zero) pointer, or an empty (length zero) string. An empty array (i.e. NULL or an empty string), will cause all available properties to be returned.
- in_stream: stream for the SDK to write into
- out_requestResult: If non-NULL, will be filled in with detailed results of the transaction.

Cleanup Required:

The request result must be destroyed using CSDK_DestroyRequestResult

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session.fetchTasksByRange()

CSDK_FetchTasksByUID

```
CAPIStatus CSDK_FetchTasksByUID  (  CAPISession    in_session,
                                    CAPIHandle    in_handle,
                                    CAPIFlag    in_flags,
                                    CAPIUIDSet    in_UIDs,
                                         const char **    in_requestProperties,
                                         CAPIStream    in_stream,
                                         CSDKRequestResult *    out_requestResult
                                        )
```

This function retrieves tasks with given UIDs on the given agenda.

Parameters:

- in_session: login session handle
- in_flags: bit flags:
  - CSDK_FLAG_NONE
  - CSDK_FLAG_STREAM_NOT_MIME: do NOT wrap the iCalendar in a MIME container
- in_handle: The agenda on which to search for tasks with the given UIDs. A NULL value will search on the current user's agenda.
- in_UIDs: An array of strings containing the UID(s) of the tasks to fetch. The array MUST be terminated with either a NULL (zero) pointer, or an empty (length zero) string.
- in_requestProperties: To fetch only specific iCalendar properties of the tasks, pass in an array of property names. The array MUST be terminated with either a NULL (zero) pointer, or an empty (length zero) string. An empty array (i.e. NULL or an empty string), will cause all available properties to be returned.
- in_stream: stream for the SDK to write into
- out_requestResult: If non-NULL, will be filled in with detailed results of the transaction.

Cleanup Required:

The result must be destroyed using CSDK_DestroyRequestResult

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Session.fetchTasksByUID()

CSDK_StoreTasks

```
CAPIStatus CSDK_StoreTasks  (  CAPISession    in_session,
                               CAPIFlag    in_flags,
                               CAPIStream    in_stream,
                               CSDKRequestResult *    out_requestResult
                              )
```

This function creates / modifies tasks on the current user's agenda depending on the SDK store flag passed.

Only one flag should be used. If multiple flags are passed the error CAPI_STAT_API_BADPARAM will be returned. There are five possible flags that can be used:

- CSDK_FLAG_STORE_IMPORT: stores the task if it does not exist and modifies the task if it exists.
- CSDK_FLAG_STORE_CREATE: stores the task. If the task exists an error will be returned.
- CSDK_FLAG_STORE_REPLACE: modifes an existing task. If the task does not exist the error is returned.
- CSDK_FLAG_STORE_MODIFY - modifies specified properties
- CSDK_FLAG_STORE_REMOVE - deletes specified properties

Other flags may be specifed along with one of the above store flags:

- CSDK_FLAG_STREAM_NOT_MIME: incoming stream is not inside a MIME wrapper

This function is blocked for a Calendar SYSOP that has not assumed the identity of a user.

Parameters:

- in_session: session
- in_flags: store flag
- in_stream: input stream
- out_requestResult: returned request result object (may be NULL)

Returns:

CAPIStatus

Return values:

- CAPI_STAT_OK
- CAPI_STAT_API_SESSION_NULL: in_session is NULL
- CAPI_STAT_API_STREAM_NULL: in_inputStream is NULL
- CAPI_STAT_API_FLAGS: in_flags is invalid

Cleanup Required:

The request result must be destroyed using CSDK_DestroyRequestResult

Equivalent Java Method:

oracle.calendar.sdk.Session.storeTasks

CSDK_AddConditionToQuery

```
CAPIStatus CSDK_AddConditionToQuery  (  CSDKQuery    in_query,
                                        CSDKCondition *    in_condition,
                                        CSDKOperator    in_operator
                                       )
```

Adds a condition to a query object. Each query may have multiple conditions, each AND'ed or OR'ed with the previous condition(s). There is no way to group conditions, and the OR operator (CSDK_LOP_OR) has a higher priority than the AND operator (CSDK_LOP_AND). Thus, C1 OR C2 AND C3 evaluates as (C1 OR C2) AND C3

Parameters:

- in_query: a query object created by CSDK_CreateQuery
- in_condition: condition to add to query
- in_operator: specifies the operator to use between existing conditions and this one (e.g. "OR", "AND")

Returns:

CAPIStatus

Equivalent Java Method:

oracle.calendar.sdk.Query.addCondition()

CSDK_CreateQuery

```
CAPIStatus CSDK_CreateQuery  (  CSDKCondition *    in_condition,
                                CSDKQuery *    out_query
                              )
```

Creates a query object to be used with CSDK_FetchContactsByQuery. An initial condition is specified (e.g. "Name begins with A") and more conditions may be added using CSDK_AddConditionToQuery.

Parameters:

- in_condition: initial condition for query
- out_query: on output, will contain new query object

Cleanup Required:

The query object MUST be destroyed by calling CSDK_DestroyQuery

Returns:

CAPIStatus

Example 3–24   Create a Query

```
//
CSDKCondition cond;
//
cond.prop  = "N";
cond.op    = CSDK_OP_STARTSWITH;
cond.value = "A";
//
CSDKQuery myQuery = CSDK_QUERY_INITIALIZER;
stat = CSDK_CreateQuery(&cond,
                        &myQuery);
//
stat = CSDK_FetchContactsByQuery(mySession,
                               CSDK_FLAG_STREAM_NOT_MIME,
                               myQuery,
                               NULL,    // get all properties
                               myStream,
                               &requestResult);
//
CSDK_DestroyQuery(&myQuery);
```

Equivalent Java Method:

oracle.calendar.sdk.Query constructor.

CSDK_DestroyQuery

```
CAPIStatus CSDK_DestroyQuery  (  CSDKQuery *    io_query  )
```

Destroys a query object created by CSDK_CreateQuery

Parameters:

io_query: a pointer to a query object to destroy. Will point to CSDK_QUERY_ INITIALIZER on exit.

Returns:

CAPIStatus

Equivalent Java Method:

None. oracle.calendar.sdk.Query finalizer destroys object.

CSDK_DestroyResult

```
CAPIStatus CSDK_DestroyResult  (  CSDKRequestResult *    io_requestResult  )
```

This function disposes of all the results in in_requestResult.

Parameters:

io_requestResult : the RequestResult to destroy

Returns:

CAPIStatus

Return values:

- CAPI_STAT_API_NULL: io_requestResult has a NULL value

Equivalent Java Method:

oracle.calendar.sdk.Result finalizer

CSDK_GetFirstFailure

```
CAPIStatus CSDK_GetFirstFailure  (  CSDKRequestResult    in_requestResult,
                                    CAPIHandle *         out_user,
                                    const char **        out_uid,
                                    CAPIStatus *         out_status
                                   )
```

This function returns the first failure obtained from the API function from which in_requestResult was returned.

A failure is a result which has a status other than CAPI_STAT_OK.

Note: A request result contains the reference to the "current" failure, so only one thread should extract failures from a given request result at a time.

Parameters:

- in_requestResult: the RequestResult to extra information from
- out_user: the user who caused this particular result
- out_uid: the uid returned by this result
- out_status: the status returned by this result

Returns:

CAPIStatus

Return values:

- CAPI_STAT_API_NULL: one of the required parameters has a NULL value
- CAPI_STAT_DATA_RRESULT_EOR: no failures in the RequestResult

Example 3–25 Get First Failure Returned from API

```
const char * vcardUID = 0;
CAPIStatus   vcardStatus = CAPI_STAT_OK;
CSDKRequestResult * myRequestResult = 0;
//
stat = CSDK_StoreContacts(mySession,
                          myStream,
                          CSDK_FLAG_STORE_IMPORT,
                          &myRequestResult);
//
```

```
  CAPIStatus failStat = CSDK_GetFirstFalure(myRequestResult,
                                            NULL,
                                            &vcardUID,
                                            &vcardStatus);
  //
  if (failStat == CAPI_STAT_DATA_RRESULT_EOR)
  {
      cout << "Store of VCARD with UID " << vcardUID << " suceeded." << endl;
  }
  else
  {
      const char * statusName = 0;
      CSDK_GetStatusString(vcardStatus, &statusName);
      cout << "Store of VCARD with UID " << vcardUID << " failed with CAPIStatus " <<
statusName << "." << endl;
  }
  //
  CSDK_DestroyResult(&myRequestResult);
```

Equivalent Java Method:

oracle.calendar.sdk.Result.getFirstFailure()

CSDK_GetFirstParseError

```
CAPIStatus CSDK_GetFirstParseError  (  CSDKRequestResult    in_requestResult,
                                        CAPIStatus *    out_status,
                                        const char **    out_errorBuffer,
                                        const char **    out_errorLocation,
                                        const char **    out_message
                                      )
```

This function returns the first parsing error obtained from a request result.

A parse error can be generated by any of the CSDK_Store* functions as they attempt to interpret incoming iCalendar or vCard.

> **Note:** A request result contains the reference to the "current" parse error, so only one thread should extract parse errors from a given request result at a time.

A pointer to a copy of the data stream is returned through out_errorBuffer, and a pointer to the parse error location in the buffer is returned via out_errorLocation. Both pointers are valid only until the request result is destroyed.

Parameters:

- in_requestResult: the RequestResult to extract information from
- out_status: the result's status
- out_errorBuffer: the beginning of the buffer with the error
- out_errorLocation: the location in *out_errorBuffer where the error occurred
- out_message: may contain additional information (NULL may be returned if no message is available)

Returns:

CAPIStatus

Return values:

- CAPI_STAT_API_NULL: one of the required parameters has a NULL value
- CAPI_STAT_DATA_RRESULT_EOR: no parse errors in the RequestResult

Example 3–26 Get First Parsing Error from Request Result

```
  CAPIStatus          stat = CAPI_STAT_OK;
  CSDKRequestResult * myRequestResult = 0;
  //
  stat = CSDK_StoreContacts(mySession,
                            myStream,
                            CSDK_FLAG_STORE_IMPORT,
                            &myRequestResult);
  //
  const char * buffer = 0;
  const char * errorLocation = 0;
  const char * message = 0;
  //
  CAPIStatus parseStat = CSDK_GetFirstParseError(myRequestResult,
                                                 NULL,
                                                 &buffer,
                                                 &errorLocation,
                                                 &message);
  //
  if (parseStat != CAPI_STAT_DATA_RRESULT_EOR)
  {
      cout << "Error (" << message << ") parsing vCard.  Buffer:'" << vcardUID
<< "'  Error starting at:'" << errorLocation << "'" << endl;
  }
```

```
//
CSDK_DestroyResult(&myRequestResult);
```

Equivalent Java Method:

oracle.calendar.sdk.Result.getFirstParseError()

CSDK_GetFirstResult

```
CAPIStatus CSDK_GetFirstResult  (  CSDKRequestResult    in_requestResult,
                                   CAPIHandle *         out_user,
                                   const char **        out_uid,
                                   CAPIStatus *         out_status
                                )
```

This function returns the first result obtained from the API function from which in_requestResult was returned.

A result is either a failure or a success. A failure is a result which has a status other than CAPI_STAT_OK.

Note: A request result contains the reference to the "current" result, so only one thread should extract the result from a given request result at a time.

Parameters:

- in_requestResult: the RequestResult to extract information from
- out_user: the user who caused this particular result
- out_uid: the uid returned by this result
- out_status: the status returned by this result

Returns:

CAPIStatus

Return values:

- CAPI_STAT_API_NULL: one of the required parameters has a NULL value
- CAPI_STAT_DATA_RRESULT_EOR: no results in the RequestResult

Example 3–27 Get First Result from API

```
  const char * vcardUID = 0;
  CAPIStatus   vcardStatus = CAPI_STAT_OK;
  CSDKRequestResult * myRequestResult = 0;
  //
  stat = CSDK_StoreContacts(mySession,
                            myStream,
                            CSDK_FLAG_STORE_IMPORT,
                            &myRequestResult);
  //
  CSDK_GetFirstResult(myRequestResult,
                      NULL,
                      &vcardUID,
                      &vcardStatus);
  //
  if (vcardStatus == CAPI_STAT_OK)
  {
      cout << "Store of VCARD with UID " << vcardUID << " succeeded." << endl;
  }
  else
  {
      const char * statusName = 0;
      CSDK_GetStatusString(vcardStatus, &statusName);
      cout << "Store of VCARD with UID " << vcardUID << " failed with CAPIStatus " <<
statusName << "." << endl;
  }
  //
  CSDK_DestroyResult(&myRequestResult);
```

Equivalent Java Method:

oracle.calendar.sdk.Result.getFirstResult()

CSDK_GetNextFailure

```
CAPIStatus CSDK_GetNextFailure  (  CSDKRequestResult    in_requestResult,
                                   CAPIHandle *    out_user,
                                   const char **    out_uid,
                                   CAPIStatus *    out_status
                                 )
```

This function returns the next failure contained in a CSDKRequestResult.

A call to CSDK_GetFirstFailure must precede this call.

Note: A request result contains the reference to the "current" failure, so only one thread should extract failures from a given request result at a time.

Parameters:

- in_requestResult: the RequestResult to extra information from
- out_user: the user who caused this particular result
- out_uid: the uid returned by this result
- out_status: the status returned by this result

Returns:

CAPIStatus:

Return values:

- CAPI_STAT_API_NULL: one of the required parameters has a NULL value
- CAPI_STAT_DATA_RRESULT_EOR: no more failures in the RequestResult

Example 3–28 Get Next Failure Contained in a CSDKRequestResult

```
const char * vcardUID = 0;
CAPIStatus   stat = CAPI_STAT_OK;
CSDKRequestResult * myRequestResult = 0;
//
stat = CSDK_StoreContacts(mySession,
                          myStream,
                          CSDK_FLAG_STORE_IMPORT,
                          &myRequestResult);
//
stat = CSDK_GetFirstFailure(myRequestResult,
                            NULL,
                            &vcardUID,
                            &vcardStatus);
//
while (stat != CAPI_STAT_DATA_RRESULT_EOR)
{
    const char * statusName = 0;
    CSDK_GetStatusString(vcardStatus, &statusName);
    cout << "Store of VCARD with UID " << vcardUID << " failed with status " << statusName << endl;
    //
    stat = CSDK_GetNextFailure(myRequestResult,
                                NULL,
                                &vcardUID,
                                &vcardStatus);
}
//
CSDK_DestroyResult(&myRequestResult);
```

Equivalent Java Method:

oracle.calendar.sdk.Result.getNextFailure()

CSDK_GetNextParseError

```
CAPIStatus CSDK_GetNextParseError  (  CSDKRequestResult    in_requestResult,
                                      CAPIStatus *    out_status,
                                      const char **    out_errorBuffer,
                                      const char **    out_errorLocation,
                                      const char **    out_message
                                     )
```

This function returns the next parsing error obtained from a request result.

A call to CSDK_GetFirstParseError must precede this call.

Note: A request result contains the reference to the "current" parse error, so only one thread should extract parse errors from a given request result at a time.

A pointer to a copy of the data stream is returned through out_errorBuffer, and a pointer to the parse error location in the buffer is returned via out_errorLocation. Both pointers are valid only until the request result is destroyed.

Parameters:

- in_requestResult: the RequestResult to extract information from
- out_status: the result's status
- out_errorBuffer: the beginning of the buffer with the error
- out_errorLocation: the location in *out_errorBuffer where the error occurred
- out_message: may contain additional information

Returns:

CAPIStatus

Return values:

- CAPI_STAT_API_NULL: one of the required parameters has a NULL value
- CAPI_STAT_DATA_RRESULT_EOR: no parse errors in the RequestResult

Example 3–29 Get Next Parsing Error Obtained from a Request Result

```
  CAPIStatus          stat = CAPI_STAT_OK;
  CSDKRequestResult * myRequestResult = 0;
  //
  stat = CSDK_StoreContacts(mySession,
                            myStream,
                            CSDK_FLAG_STORE_IMPORT,
                            &myRequestResult);
  //
  const char * buffer = 0;
  const char * errorLocation = 0;
  const char * message = 0;
  //
  CAPIStatus parseStat = CSDK_GetFirstParseError(myRequestResult,
                                                 NULL,
                                                 &buffer,
                                                 &errorLocation,
                                                 &message);
  //
  while (parseStat != CAPI_STAT_DATA_RRESULT_EOR)
  {
      cout << "Error (" << message << ") parsing vCard.  Buffer:'" << vcardUID
<< "'  Error starting at:'" << errorLocation << "'" << endl;
      parseStat = CSDK_GetNextParseError(myRequestResult,
                                         NULL,
                                         &buffer,
                                         &errorLocation,
                                         &message);
  }
  //
  CSDK_DestroyResult(&myRequestResult);
```

Equivalent Java Method:

oracle.calendar.sdk.Result.getNextParseError()

CSDK_GetNextResult

```
CAPIStatus CSDK_GetNextResult  (  CSDKRequestResult    in_requestResult,
                                  CAPIHandle *         out_user,
                                  const char **        out_uid,
                                  CAPIStatus *         out_status
                                 )
```

This function returns the next result contained in a CSDKRequestResult.

A call to CSDK_GetFirstResult must precede this call.

> **Note:** A request result contains the reference to the "current" result, so only one thread should extract the result from a given request result at a time.

Parameters:

- in_requestResult: the RequestResult to extra information from
- out_user: the user who caused this particular result
- out_uid: the uid returned by this result
- out_status: the status returned by this result

Returns:

CAPIStatus

Return values:

- CAPI_STAT_API_NULL: one of the required parameters has a NULL value
- CAPI_STAT_DATA_RRESULT_EOR: no more results in the RequestResult

Example 3–30 Get Next Result Contained in a CSDKRequestResult

```
const char * vcardUID = 0;
CAPIStatus   stat = CAPI_STAT_OK;
CSDKRequestResult * myRequestResult = 0;
//
stat = CSDK_StoreContacts(mySession,
                          myStream,
                          CSDK_FLAG_STORE_IMPORT,
                          &myRequestResult);
//
stat = CSDK_GetFirstResult(myRequestResult,
                              NULL,
                              &vcardUID,
                              &vcardStatus);
  //
  while (stat != CAPI_STAT_DATA_RRESULT_EOR)
  {
      const char * statusName = 0;
      CSDK_GetStatusString(vcardStatus, &statusName);
      cout << "Store of VCARD with UID " << vcardUID << " returned status " << statusName
<< endl;
      //
      stat = CSDK_GetNextResult(myRequestResult,
                                 NULL,
                                 &vcardUID,
                                 &vcardStatus);
  }
  //
  CSDK_DestroyResult(&myRequestResult);
```

Equivalent Java Method:

oracle.calendar.sdk.Result.getNextResult()

CSDK_GetStatusLevels

```
void CSDK_GetStatusLevels  (  CAPIStatus    in_status,
                              unsigned long *    out_level1,
                              unsigned long *    out_level2,
                              unsigned long *    out_level3,
                              unsigned long *    out_level4,
                              unsigned long *    out_level5
                             )
```

This function decomposes a CAPIStatus into its sub-parts.

Each part of the status code specifies more precisely the actual error.

Parameters:

- in_status: SDK status
- out_level1: will contain the int result for level1
- out_level2: will contain the int result for level2
- out_level3: will contain the int result for level3
- out_level4: will contain the int result for level4

- out_level5: will contain the int result for level5

Changes:

CAPI 2.5: types of "out_level[12345]" changed from "int *" to "unsigned long *"

Equivalent Java Method:

oracle.calendar.sdk.Api.getStatusLevels()

CSDK_GetStatusString

```
void CSDK_GetStatusString  (  CAPIStatus    in_status,
                              const char **    out_string
                             )
```

This function returns a read-only string representation of a CAPIStatus.

This is generally more useful than the numeric representation.

Parameters:

- in_status: SDK status
- out_string: will contain const pointer to the result string

Cleanup Required:

None. The string returned is a const string cannot be freed

Equivalent Java Method:

oracle.calendar.sdk.Api.getStatusString()

Superceded Functions

The following functions have been superceded by their newer, renamed equivalents. While it is best to use the new functions, you can use the older functions provided you define CSDK_USE_OLD_NAMES.

Superceded function	New version
CAPI_CreateCallbackStream	CSDK_CreateCallbackStream
CAPI_CreateFileStreamFromFilenames	CSDK_CreateFileStreamFromFilenames

Superceded function	New version
CAPI_CreateMemoryStream	CSDK_CreateMemoryStream
CAPI_DestroyHandles	CSDK_DestroyHandles
CAPI_DestroyStreams	CSDK_DestroyStreams
CAPI_GetCapabilities	CSDK_GetCapabilities
CAPI_GetHandle	CSDK_GetHandle
CAPI_GetStatusLevels	CSDK_GetStatusLevels
CAPI_GetStatusString	CSDK_GetStatusString
CAPI_HandleInfo	CSDK_GetHandleInfo
CAPI_SetConfigFile	CSDK_SetConfigFile
CAPI_SetIdentity	CSDK_SetIdentity

Legacy Functions

The following functions were included with previous versions of the SDK and are still supported for this release:

- CAPI_CreateFileStream
- CAPI_AuthenticateAsSysop
- CAPI_Connect
- CAPI_DeleteEvent
- CAPI_FetchEventByID
- CAPI_FetchEventsByAlarmRange
- CAPI_FetchEventsByRange
- CAPI_GetLastStoredUIDs
- CAPI_Logoff
- CAPI_Logon
- CAPI_StoreEvent

Included here is documentation for the functions CAPI_CreateFileStream and CAPI_AuthenticateAsSysop. For more information about these and other legacy functions, see the SDK source files or the documentation included with a previous release.

CAPI_CreateFileStream

```
CSDKStatus CAPI_CreateFileStream  (  CAPISession    in_session,
                                     CAPIStream *    out_stream,
                                     FILE *    in_readFile,
                                     FILE *    in_writeFile,
                                     CAPIFlag    in_flags
                                   )
```

Creates file stream to allow the SDK to read from or write to open files.

For compatibility reasons, it is safer to use CSDK_CreateFileStreamFromFilenames which will prevent the need for passing FILE * variables between your application and CAPI.

Files must have been opened using fopen() with an appropriate mode.

Parameters:

- in_session: login session handle
- out_stream: in output, will point to new stream
- in_readFile: FILE * to read from
- in_writeFile: FILE * to write to
- in_flags: bit flags (must be CSDK_FLAG_NONE at this time)

Returns:

CAPIStatus

Cleanup Required:

The stream returned by this function must be destroyed by calling CSDK_ DestroyStreams()

Example 3–31 Store Events from the File "events.ics"

```
FILE * myFileFullOfMIMEEncapsulatediCal = fopen("events.ics", "rb");
if (myFileFullOfMIMEEncapsulatediCal != NULL)
{
    CAPIStream myInputStream = NULL;
    CAPIStatus status = CAPI_CreateFileStream(mySession,
                                              &myInputStream,
                                              myFileFullOfMIMEEncapsulatediCal,
                                              NULL,  // no output file
                                              CSDK_FLAG_NONE);
    if (status == CAPI_STAT_OK)
    {
        status = CAPI_StoreEvent(mySession,
                                 myHandles,
                                 numHandles,
                                 handleStatus,
                                 CAPI_STORE_REPLACE,
                                 myInputStream);
    }
    fclose(myFileFullOfMIMEEncapsulatediCal);
    CSDK_DestroyStreams(mySession,
                        &myInputStream,
                        1,
                        CSDK_FLAG_NONE);
}
```

Example 3–32   Fetch Events and Write Them to the File "myAgenda.ics"

```
FILE * outputFile = fopen("myAgenda.ics", "wb");
if (outputFile != NULL)
{
    CAPIStream myOutputStream = NULL;
    CAPIStatus status = CAPI_CreateFileStream(mySession,
                                              &myOutputStream,
                                              NULL,  // no input file
                                              outputFile,
                                              CSDK_FLAG_NONE);
    if (status == CAPI_STAT_OK)
    {
        status = CAPI_FetchEventsByRange(mySession,
                                         myHandles,
                                         numHandles,
                                         handleStatus,
                                         CSDK_FLAG_NONE,
                                         "20020722T000000",
                                         "20020722T235900",
                                         NULL,
                                         0,
                                         myOutputStream);
    }
    fclose(outputFile);
    CSDK_DestroyStreams(mySession,
                        &myOutputStream,
                        1,
                              CSDK_FLAG_NONE);
    }
```

Equivalent Java Method:

None. The Java APIs only use String and StringBuffer objects to send and receive data.

CAPI_AuthenticateAsSysop

```
CAPIStatus CAPI_AuthenticateAsSysop  (  const char *    in_password,
                                        const char *    in_host,
                                        const char *    in_node,
                                        CAPIFlag    in_flags,
                                        CAPISession *    io_session
                                       )
```

Log on as Calendar SYSOP.

Once logged on, the Calendar SYSOP can assume the identity of any user by calling CAPI_SetIdentity().

A node should always be specified since masternode and calendar-domain functionality is not available during logon as Calendar SYSOP.

If the host parameter specifies ACE mechanisms, these will be ignored. The admin default ACE settings from the calendar server will be used.

Calendar SYSOP authentication is only available with version 5.3 and newer servers. An error will be returned if the provided host does not support this feature. A calendar server may be configured to refuse Calendar SYSOP logon via CAPI in which case a security error will be returned.

The operations available to Calendar SYSOPs are limited to:

- logging off by calling CAPI_Logoff()
- switching identity to a user by calling CAPI_SetIdentity()

Once the identity has been set to a user, all operations will be performed as if that user had logged in.

See CAPI_Connect() for the format of the in_host parameter.

Parameters:

in_password: Calendar SYSOP's password. May be NULL.

in_host: Calendar server host name (optional port no.)

in_node: node ID to connect to as Calendar SYSOP.

in_flags: Bit flags modifying behaviour. This must be CAPI_FLAG_NONE currently.

io_session: Session opened by CAPI_Connect

Returns:

CAPIStatus

Example 3–33   Logon as Calendar SYSOP

```
Example:
// Connect to myNode on the server running on the default port of
calserver.acme.com, to authenticate as Calendar SYSOP:
 {
     CAPIStatus  myStatus  = CAPI_AuthenticateAsSysop("theSysopPassword",
                                                      "calserver.acme.com",
                                                      "1",
                                                      CAPI_FLAG_NONE,
                                                      &mySession);
 }
```

Cleanup Required:

The sessions created by calling this routine must be destroyed by calling CAPI_Logoff.

Changes:

SDK 9.0.4: in_node must now be specified as a number (e.g. "100"). Node aliases are no longer supported, and NULL can no longer be passed in.

4 Calendar SDK Configuration Settings

This chapter contains detailed information on Oracle Calendar SDK configuration settings. These settings may be placed in a text file, the name of which must be passed to the function CSDK_SetConfigFile. The structure of the file is:

[<section>]

<keyword>=<value>

…

Example 4–1 Configuration Settings Text File

```
[CAPI]
client_name = mycool.fcgi
client_version = 1.0
[LOG]
log_activity = true
log_modulesinclude = {CAPI}
```

The supported configuration settings are:

- blocking
- client_name
- client_version
- cncachesize
- direntrycachesize
- emailcachesize
- itemcachesize
- log_activity

- log_modulesinclude
- max_caldomain
- max_masternode
- max_sysop
- max_user
- securitycachesize
- tzcachesize

blocking

Used to configure whether a connection request will block or return an error if there are no available connections.

Section

CONNPOOL

Values

true / false

Default Value

true

client_name

Used to set the application name which be visible in the server stats.

Section

CAPI

Values

any string

Default Value

""

client_version

Used to set the application version which be visible in the server stats.

Section

CAPI

Values

any string

Default Value

""

cncachesize

Used to set the maximum number of entries to hold in common name cache.

Section

CACHE

Values

[0..U32MAX]

Default Value

512

direntrycachesize

Used to set the maximum number of entries to hold in the directory entry cache.

Section

CACHE

Values

[0..U32MAX]

Default Value

512

emailcachesize

Used to set the maximum number of entries to hold in the e-mail address cache.

Section

CACHE

Values

[0..U32MAX]

Default Value

512

itemcachesize

Used to set the maximum number of entries to hold in the item cache.

Section

CACHE

Values

[0..U32MAX]

Default Value

512

log_activity

Used to enable "activity" (high-level) logging.

Section

LOG

Values
true / false

Default Value
false

See also:

log_modulesinclude

log_modulesinclude

Used to control which modules have logging enabled.

This should be set to "{CAPI}", otherwise no logging will be performed even if it is enabled (e.g. via log_activity = true)

Section
LOG

Values
"" or "{ CAPI }"

Default Value
""

max_caldomain

Used to set the maximum # of caldomainserver connections for the given server name / node ID in the connection pool.

Section
CONNPOOL

Values
[0..S32MAX]

Default Value
0

max_masternode

Used to set the maximum number of masternode connections for the given server name/node ID in the connection pool.

Section

CONNPOOL

Values

[0..S32MAX]

Default Value

0

max_sysop

Used to set the maximum number of Calendar SYSOP connections for the given server name/node ID in the connection pool.

Section

CONNPOOL

Values

[0..S32MAX]

Default Value

0

max_user

Used to set the maximum number of user connections for the given server name/node ID in the connection pool.

Section

CONNPOOL

Values
[0..S32MAX]

Default Value
none, value required to use connection pooling

securitycachesize
Used to set the maximum number of entries to hold in the security record cache.

Section
CACHE

Values
[0..U32MAX]

Default Value
512

tzcachesize
Used to set the maximum number of entries to hold in the time zone record cache.

Section
CACHE

Values
[0..U32MAX]

Default Value
256

5

Calendar SDK Flags and Capabilities

CSDK Flags

CSDK_FLAG_NONE

Used to select the default behavior.

CAPI_FLAG_STORE_DELPROPS

Used with CAPI_Store* functions to specify that the supplied properties are to be cleared or deleted on the server.

CAPI_FLAG_STORE_MODPROPS

Used with CAPI_Store* functions to specify that the supplied properties are to be modified on the server without changing other properties (where possible).

CSDK_FLAG_FETCH_AGENDA_ATTENDEE_ONLY

Used with CSDK_FetchEvent* calls to filter out ATTENDEE properties for all attendees other than the agenda being viewed.

CSDK_FLAG_FETCH_COMBINED

Used with CSDK_FetchEvent* calls to return all events in a single VCALENDAR rather than one VCALENDAR per agenda.

This is faster and produces smaller output.

CSDK_FLAG_FETCH_DO_NOT_EXPAND_RRULE

Can be passed to CSDK_FetchEvent* calls to request that recurrence rules not be expanded.

This flag is set by default with CSDK_FetchEventsByUID and can be overridden by using CSDK_FLAG_FETCH_EXPAND_RRULE.

CSDK_FLAG_FETCH_EXCLUDE_ACCEPTED

Used with CSDK_FetchEvent* calls to exclude events the caller has accepted.

CSDK_FLAG_FETCH_EXCLUDE_APPOINTMENTS

Used with CAPI_FetchEvent* and CSDK_FetchEvent* calls to exclude regular meetings (appointments).

CSDK_FLAG_FETCH_EXCLUDE_DAILYNOTES

Used with CAPI_FetchEvent* and CSDK_FetchEvent* calls to exclude daily notes.

CSDK_FLAG_FETCH_EXCLUDE_DAYEVENTS

Used with CAPI_FetchEvent* and CSDK_FetchEvent* calls to exclude day events.

CSDK_FLAG_FETCH_EXCLUDE_DECLINED

Used with CSDK_FetchEvent* calls to exclude events the caller has declined.

CSDK_FLAG_FETCH_EXCLUDE_HOLIDAYS

Used with CAPI_FetchEvent* and CSDK_FetchEvent* calls to exclude holidays.

CSDK_FLAG_FETCH_EXCLUDE_NOTOWNER

Used with CSDK_FetchEvent* calls to exclude events which are not owned by the caller.

CSDK_FLAG_FETCH_EXCLUDE_UNCONFIRMED

Used with CSDK_FetchEvent* calls to exclude events the caller has not confirmed.

CSDK_FLAG_FETCH_EXPAND_RRULE

Can be passed to CSDK_FetchEvent* calls to request that recurrence rules be expanded.

This flag is set by default with CSDK_FetchEventsByRange and CSDK_FetchEventsByAlarmRange and can be overridden by using CSDK_FLAG_FETCH_DO_NOT_EXPAND_RRULE.

CSDK_FLAG_FETCH_LOCALTIMES

Used with CSDK_FetchEvent* and CSDK_FetchTask* calls to request that times be returned in the local time zone.

The current user's preferred time zone as set on the calendar server will be used as the local time zone

CAPI_FLAG_FETCH_NO_FIELDHOLDERS

Used with CAPI_FetchEventsBy* to avoid reading some event properties from the server.

There are significant performance gains when using this flag, but X- properties and user-assigned UIDs will not be returned.

CSDK_FLAG_FETCH_VCARD_VERSION_2_1

Used with CSDK_FetchContacts* calls to request version 2.1 vCards.

CSDK_FLAG_FETCH_VCARD_VERSION_3_0

Used with CSDK_FetchContacts* calls to request version 3.0 vCards (default).

CSDK_FLAG_STORE_CREATE

Used with CSDK_Store* calls to create a new object on the server.

If an object already exists with the same UID, an error will be returned

CSDK_FLAG_STORE_IMPORT

Used with CSDK_Store* calls to create (CSDK_FLAG_STORE_CREATE) a new object (task/contact) on the server if none exists with the given UID, or to completely replace (CSDK_FLAG_STORE_REPLACE) an existing object.

CSDK_FLAG_STORE_INVITE_SELF

Used with CSDK_StoreEvents to invite the current user without requiring an ATTENDEE property.

CSDK_FLAG_STORE_MODIFY

Used with CSDK_Store* calls to add, or modify the given properties to an existing object on the server.

CSDK_FLAG_STORE_REMOVE

Used with CSDK_StoreContacts and CSDK_StoreTasks* calls to remove the given properties from an existing object on the server.

The UID property must be specified in the input, but will not itself be removed from the server. This mode will not completely delete an object on the server - use CSDK_Delete* instead.

CSDK_FLAG_STORE_REPLACE

Used with CSDK_Store* functions to specify that the object (event/task/contact) on the server should be completely replaced by the given object.

CSDK_FLAG_STORE_REPLY

Used with CSDK_StoreEvents to reply (set ATTENDEE and VALARM) to events organized by other people.

CSDK_FLAG_STREAM_NOT_MIME

Used with CSDK_Store* and CSDK_Fetch* calls to specify that the stream should not be MIME encapsulated.

Capabilities

Typedefs

typedef long CAPICapabilityID

CAPI_CAPAB_AUTH

Returns the authentication mechanisms supported by the server

E.g.: `cs-standard,gssapi:kerberos5,sasl:KERBEROS_V4`. A server connection must exist to read this capability.

CAPI_CAPAB_CAPI_VERSION

Returns the SDK version as a string.

E.g.: `9.0.4`

CAPI_CAPAB_COMP

Returns the compression mechanisms supported by the server

E.g.: `cs-simple,none`. A server connection must exist to read this capability.

CAPI_CAPAB_ENCR

Returns the encryption mechanisms supported by the server

E.g.: `cs-light,none`. A server connection must exist to read this capability.

CAPI_CAPAB_MAXDATE

Returns the largest date that the SDK can handle (`20371129`).

CAPI_CAPAB_SERVER_VERSION

Returns the server version as a string.

E.g.: `6.0`. A server connection must exist to read this capability.

CAPI_CAPAB_UNSUPPORTED_ICAL_COMP

Returns a comma delimited list of iCal components that the SDK does not process.

E.g.: `VJOURNAL,VFREEBUSY`

CAPI_CAPAB_UNSUPPORTED_ICAL_PROP

Returns a comma delimited list of iCal properties that the SDK does not process.

E.g.: `GEO,COMMENT`. A server connection must exist to read this capability.

CAPI_CAPAB_VERSION

Same as CAPI_CAPAB_CAPI_VERSION.

6

Calendar SDK Status Codes

This chapter documents all CAPIStatus values that may be returned by the SDK functions, in alphabetical order. The functions CSDK_GetStatusString and CSDK_GetStatusLevels may be useful when interpreting CAPIStatus values.

CAPI_STAT_API

API class status.

CAPI_STAT_API_BADPARAM

A bad parameter was passed.

CAPI_STAT_API_CALLBACK

There was a problem with a callback.

CAPI_STAT_API_CALLBACK_ERROR

The callback returned an error, which is returned in bit field 5.

CAPI_STAT_API_FLAGS

Bad flags were passed.

CAPI_STAT_API_HANDLE

There was a problem with a handle.

CAPI_STAT_API_HANDLE_BAD

The passed handle was corrupt.

CAPI_STAT_API_HANDLE_NOTNULL
The passed handle was not null.

CAPI_STAT_API_HANDLE_NULL
The passed handle was null.

CAPI_STAT_API_NULL
A null pointer was passed.

CAPI_STAT_API_SESSION
There was a problem with a session.

CAPI_STAT_API_SESSION_BAD
The passed session was corrupt.

CAPI_STAT_API_SESSION_NOTNULL
The passed session was not null.

CAPI_STAT_API_SESSION_NULL
The passed session was null.

CAPI_STAT_API_STREAM
There was a problem with a stream.

CAPI_STAT_API_STREAM_BAD
The passed stream was corrupt.

CAPI_STAT_API_STREAM_NOTNULL
The passed stream was not null.

CAPI_STAT_API_STREAM_NULL
The passed stream was null.

CAPI_STAT_DATA
Data class status.

CAPI_STAT_DATA_COOKIE
Information about the supplied cookie.

CAPI_STAT_DATA_DATE
Information about a date.

CAPI_STAT_DATA_DATE_FORMAT
The format of the date data is incorrect.

CAPI_STAT_DATA_DATE_INVALID
A specified date is invalid (e.g. Feb 30th)

CAPI_STAT_DATA_DATE_OUTOFRANGE
A specified date is out of the range supported by this implementation.

CAPI_STAT_DATA_DATE_RANGE
The date range is incorrect.

CAPI_STAT_DATA_EMAIL
Information about email.

CAPI_STAT_DATA_EMAIL_NOTSET
No email address is set on the server for 1 or more users/resources.

CAPI_STAT_DATA_ENCODING
Information about the encoding of supplied data.

CAPI_STAT_DATA_HOSTNAME
Information about a hostname.

CAPI_STAT_DATA_HOSTNAME_FORMAT
The format of the hostname string was wrong.

CAPI_STAT_DATA_HOSTNAME_HOST
The hostname string could not be resolved to a host.

CAPI_STAT_DATA_HOSTNAME_SERVER
No server could be found on the specified host and port.

CAPI_STAT_DATA_ICAL
Information about iCalendar data.

CAPI_STAT_DATA_ICAL_COMPEXTRA
An extra component was encountered. Either multiple specifications of a component which should only appear once, or a component which should not appear.

CAPI_STAT_DATA_ICAL_COMPMISSING
An expected or required component was missing.

CAPI_STAT_DATA_ICAL_COMPNAME
There was a problem with a component name.

CAPI_STAT_DATA_ICAL_COMPVALUE
There was a problem with what a component contained.

CAPI_STAT_DATA_ICAL_FOLDING
There was a problem in the line folding.

CAPI_STAT_DATA_ICAL_IMPLEMENT
A problem with this particular iCalendar implementation.

CAPI_STAT_DATA_ICAL_LINEOVERFLOW
One of the iCal data lines was too long, breaching the iCalendar spec (RFC 2445).

CAPI_STAT_DATA_ICAL_NONE
The provided data was not iCalendar data.

CAPI_STAT_DATA_ICAL_OVERFLOW
There was an overflow when parsing the iCalendar data. This is caused by an internal limitation of the iCalendar library, and not by a breach of the spec.

CAPI_STAT_DATA_ICAL_PARAMEXTRA

An extra parameter was encountered. Either multiple specifications of a parameter which should only appear once, or a parameter which should not appear.

CAPI_STAT_DATA_ICAL_PARAMMISSING

An expected or required parameter was missing.

CAPI_STAT_DATA_ICAL_PARAMNAME

There was a problem with a parameter name.

CAPI_STAT_DATA_ICAL_PARAMVALUE

There was a problem with a parameter value.

CAPI_STAT_DATA_ICAL_PROPEXTRA

An extra property was encountered. Either multiple specifications of a property which should only appear once, or a property which should not appear.

CAPI_STAT_DATA_ICAL_PROPMISSING

An expected or required property was missing.

CAPI_STAT_DATA_ICAL_PROPNAME

There was a problem with a property name.

CAPI_STAT_DATA_ICAL_PROPVALUE

There was a problem with a property value.

CAPI_STAT_DATA_ICAL_RECURMODE

There was a problem with the recurrence specification. The rules laid out in the description of CAPI_StoreEvent were breached.

CAPI_STAT_DATA_MIME

Information about MIME data.

CAPI_STAT_DATA_MIME_CHARSET

An unsupported character set was specified in a MIME header.

CAPI_STAT_DATA_MIME_COMMENT
A comment could not be parsed.

CAPI_STAT_DATA_MIME_ENCODING
The encoding specified in the MIME object is not supported.

CAPI_STAT_DATA_MIME_HEADER
A header could not be parsed.

CAPI_STAT_DATA_MIME_IMPLEMENT
A restriction specific to this MIME implementation was breached.

CAPI_STAT_DATA_MIME_IMPLEMENT_NESTING
The MIME object was nested too deeply.

CAPI_STAT_DATA_MIME_LENGTH
One of the header lines was too long.

CAPI_STAT_DATA_MIME_NOICAL
No MIME parts were found whose headers indicated that they contain iCalendar data.

CAPI_STAT_DATA_MIME_NONE
No MIME data was found.

CAPI_STAT_DATA_MIME_OVERFLOW
An overflow occurred while reading MIME data.

CAPI_STAT_DATA_UID
Information about a UID.

CAPI_STAT_DATA_UID_FORMAT
The format of the UID string was wrong.

CAPI_STAT_DATA_UID_NOTFOUND
An object with the supplied UID could not be found.

CAPI_STAT_DATA_UID_RECURRENCE

The specified recurrence could not be found.

CAPI_STAT_DATA_USERID

Information about a userID.

CAPI_STAT_DATA_USERID_EXT

There was a problem with the Extended part of the UserId string.

CAPI_STAT_DATA_USERID_EXT_CONFLICT

Either userID AND X.400 were specified, or both a node and a calendar domain were specified.

CAPI_STAT_DATA_USERID_EXT_FORMAT

The format of the extended string was bad.

CAPI_STAT_DATA_USERID_EXT_INIFILE

There was a problem with the configuration file.

CAPI_STAT_DATA_USERID_EXT_MANY

Multiple users were identified by the string.

CAPI_STAT_DATA_USERID_EXT_NODE

The specified node could not be found.

CAPI_STAT_DATA_USERID_EXT_NONE

No users were identified by the string.

CAPI_STAT_DATA_USERID_FORMAT

The format of the UserId string was wrong.

CAPI_STAT_DATA_USERID_ID

There was a problem with the Id part of the UserId string.

CAPI_STAT_LIBRARY

Library class status.

CAPI_STAT_LIBRARY_IMPLEMENTATION
The feature is not fully implemented.

CAPI_STAT_LIBRARY_INTERNAL
An internal error occured in the library.

CAPI_STAT_LIBRARY_INTERNAL_COSMICRAY
Something completely unexpected happened internally.

CAPI_STAT_LIBRARY_INTERNAL_DATA
There was a corruption of data in the library.

CAPI_STAT_LIBRARY_INTERNAL_EXPIRY
The library has expired.

CAPI_STAT_LIBRARY_INTERNAL_FUNCTION
The library miscalled a function.

CAPI_STAT_LIBRARY_INTERNAL_OVERFLOW
Some internal maximum was exceeded.

CAPI_STAT_LIBRARY_INTERNAL_PROTOCOL
The library abused a protocol.

CAPI_STAT_LIBRARY_INTERNAL_UNKNOWN_EXCEPTION
The SDK received an unknown C++ exception.

CAPI_STAT_LIBRARY_INTERNAL_UNKNOWN_LIBRARY_ERRCODE
Failed to map an error code from a dependant library.

CAPI_STAT_LIBRARY_SERVER
A limitation of or occurence on the server.

CAPI_STAT_LIBRARY_SERVER_BUSY
The server cannot service the request right now because it is busy.

CAPI_STAT_LIBRARY_SERVER_SUPPORT
The server does not provide some necessary support.

CAPI_STAT_LIBRARY_SERVER_SUPPORT_CHARSET
There is no support for the required character set.

CAPI_STAT_LIBRARY_SERVER_SUPPORT_STANDARDS
There is no support for the SDK on this server.

CAPI_STAT_LIBRARY_SERVER_SUPPORT_UID
There is no support for storing UIDs.

CAPI_STAT_LIBRARY_SERVER_USERDATA
There is some problem with user data on the server.

CAPI_STAT_OK
Operation completed successfully. Value 0.

CAPI_STAT_SECUR
Security class status.

CAPI_STAT_SECUR_LOGON
There was a security error on logon.

CAPI_STAT_SECUR_LOGON_AUTH
Logon authentication failed.

CAPI_STAT_SECUR_LOGON_LOCKED
The specified account is locked.

CAPI_STAT_SECUR_LOGON_LOCKED_RESOURCE
Logon is locked for resources.

CAPI_STAT_SECUR_LOGON_LOCKED_SYSOP
Logon is locked for Calendar SYSOPs.

CAPI_STAT_SECUR_READ
There was a security error on read.

CAPI_STAT_SECUR_READ_ALARM
There was a security error reading alarm data.

CAPI_STAT_SECUR_READ_PROPS
There was a security error reading properties.

CAPI_STAT_SECUR_SERVER
There was a security error in the server.

CAPI_STAT_SECUR_SERVER_LICENSE
There was a licensing error on the server.

CAPI_STAT_SECUR_SERVER_SET_IDENTITY_SYSOP
The server requires a SetIdentity call after a Calendar SYSOP logon to perform the operation.

CAPI_STAT_SECUR_WRITE
There was a security error on write.

CAPI_STAT_SECUR_WRITE_AGENDA
There was a security error writing to an agenda.

CAPI_STAT_SECUR_WRITE_EVENT
There was a security error writing to an event.

CAPI_STAT_SERVICE
Service class status.

CAPI_STAT_SERVICE_ACE
There was a problem caused by one of the ACE plugins.

CAPI_STAT_SERVICE_ACE_LOAD
Required ACE plugin could not be loaded.

CAPI_STAT_SERVICE_ACE_SUPPORT
Requested ACE option not supported.

CAPI_STAT_SERVICE_FILE
There was a problem with system file services.

CAPI_STAT_SERVICE_FILE_CLOSE
There was a problem closing a file.

CAPI_STAT_SERVICE_FILE_DELETE
There was a problem deleting a file.

CAPI_STAT_SERVICE_FILE_MODE
There was a problem with the read or write mode for a file.

CAPI_STAT_SERVICE_FILE_OPEN
There was a problem opening a file.

CAPI_STAT_SERVICE_FILE_READ
There was a problem reading from a file.

CAPI_STAT_SERVICE_FILE_TEMP
There was a problem allocating a temporary file.

CAPI_STAT_SERVICE_FILE_WRITE
There was a problem writing to a file.

CAPI_STAT_SERVICE_LIBRARY
There was a problem with the standard library services.

CAPI_STAT_SERVICE_MEM
There was a problem with system memory services.

CAPI_STAT_SERVICE_MEM_ALLOC
Could not allocate memory.

CAPI_STAT_SERVICE_NET
There was a problem with network services.

CAPI_STAT_SERVICE_NET_TIMEOUT
Timeout while waiting for network services.

CAPI_STAT_SERVICE_THREAD
There was a problem with system thread services.

CAPI_STAT_SERVICE_TIME
There was a problem with the standard time services.

CAPI_STAT_SERVICE_TIME_GMTIME
GMTime could not be obtained.

7

Calendar SDK FAQ and Troubleshooting

This chapter contains frequently asked questions and troubleshooting information for the Oracle Calendar SDK.

Frequently Asked Questions

Can I use previous versions of the SDK ("CAPI") with the new Oracle Calendar server? What about my existing applications?

The newest release of the Calendar SDK will include a migration utility for pre-9.0.4 versions. This will allow you to port your old applications to the new Calendar server.

What's the difference between the Steltor packages and the newest Oracle Calendar SDK?

With the 9.0.4 release of the Oracle Calendar SDK, many new features have been added. In addition to the new server, which provides better performance and more efficient processing, the following has been added to the SDK package:

- Connection pooling design
- JNI functions
- Remote Designate support
- Task and Contact support
- Calendar SYSOP log-in

Can I write an Oracle Calendar SDK program using Visual Basic or other programming languages?

The Oracle Calendar SDK is a package of C/C++ function calls, so any language that can natively support C can be used to create a wrapper and access these functions.

Options include using Visual Basic, Perl, Java, and Python. There may be other independent efforts in existence; the Oracle Calendar SDK forum on Oracle OTN is a good place to look for such information.

How do I uninvite someone from an event?

Either replace the event by calling CSDK_StoreEvents and supplying an iCalendar without an ATTENDEE property for the user, or use CAPI_GetHandle for the user you want to remove from the list of attendees, followed by a call to CAPI_ DeleteEvent.

How do I invite someone to an event?

Provide an ATTENDEE property for the user when calling CSDK_StoreEvents.

Can I log in as the calendar server administrator (Calendar SYSOP) using Oracle Calendar SDK?

Yes. Calendar SYSOP login is supported as of release 9.0.4 of the Oracle Calendar SDK.

How do I delete an event completely?

Use the function CSDK_DeleteEvents, supplying the event UID.

Troubleshooting

When I run the Oracle Calendar SDK demos, I get the error "libcapi.so not found, no such file or directory", or "Cannot load library libcapi". How do I fix this?

See the latest Oracle Calendar SDK Readme file for this information.

I'm getting error 2148073984 from calling an Oracle Calendar SDK function. What does that mean? Where can I find more information on Oracle Calendar SDK errors?

Chapter 6, "Calendar SDK Status Codes", contains a list of Oracle Calendar SDK status codes. Each code can be divided into five fields, each describing a different level of the problem.

The two helper functions, CSDK_GetStatusLevels() and CSDK_GetStatusString(), can help decode the error easily. You can also look at the demo applications that ship with Oracle Calendar SDK for an example of how to decode error codes.

Why don't I get ATTENDEE properties when fetching an event using Oracle Calendar SDK?

Since an e-mail address is required to generate the iCalendar ATTENDEE property, Oracle Calendar SDK will skip any attendee who does not have an e-mail address when exporting an iCalendar event. The suggested workaround is to make sure all attendees have an e-mail address stored on the calendar server.

Why do my accented e characters appear as =C3=A9 in events I retrieve using Oracle Calendar SDK?

Oracle Calendar SDK encodes its output in UTF-8, in which the character is represented as 2 bytes: 0xC3 and 0xA9. When returning MIME-encapsulated data, Oracle Calendar SDK further encodes those bytes in quoted-printable strings, which results in =C3=A9.

There seem to be a lot of extra =3D characters in the ATTENDEE property. Did I just discover a bug?

No, this is not a bug. `=3D` is really an equal sign (`=`) encoded in quoted-printable. So, where you might expect to see `partstat=confirmed`, you would actually see `partstat=3Dconfirmed`.

I just stored an iCalendar event with 5 ATTENDEE properties. Why are none of them showing in my native client?

When using CAPI_StoreEvents to invite people to an event, you need to supply a CAPIHandle for each attendee.

Why do I get a LNK1106 error when compiling Oracle Calendar SDK with Microsoft Visual C++?

Oracle Calendar SDK was compiled with Visual C++ 6; older versions need a patch to work. Search the Microsoft Web site for available patches.

Ensuring that you are using Visual C++ will mitigate linking errors you may encounter with other C compilers. Refer to the demo applications project for recommended compiler and linker settings.

Why is my program aborting whenever I use CSDK_CreateFileStream()? I tried the other type of streams and they worked fine.

You need to link your application using Microsoft's C runtime library (the /MD switch if you're using the command line) as opposed to the static C libraries. This is required because the FILE pointer used by the file stream has different definitions depending on which version of the C library is being used.

CSDK_CreateFileStreamFromFilenames can be used, in which case the FILE pointer will not need to be passed between the SDK and your application.

Why does CSDK_GetHandle() tell me it can't find my resource?

There are a couple of possible answers:

1. Your resource name may contain accented or special characters such as "‡". Oracle Calendar SDK expects user and resource identification strings to be in UTF-8. Be sure to use the proper UTF-8 values to describe accented or special characters.

2. Your resource name contains a forward-slash / character, such as ""Training / Meeting Room"". The use of this character inside resource names conflicts with the / used by default as the field delimiter. To fix this, simply use another character as your delimiter. For example, RS=Work/.

Why does Oracle Calendar SDK seem to be ignoring my vAlarm information when I use CAPI_StoreEvent?

You can only set a vAlarm (reminder) on events in your own agenda.

Why won't CAPI_FetchEventsByRange() show any of my event reminders?

This is the normal behavior of the function. To retrieve reminders (vAlarms), it is recommended that you use the functions CSDK_FetchEventsBy..., CAPI_FetchEventByID() or CAPI_FetchEventsByAlarmRange().

Part II

Oracle Calendar Web Services Toolkit

This part of the Oracle Calendar Application Developer's Guide describes the Oracle Calendar web services toolkit.

This part contains the following chapters:

- Chapter 8, "Calendar Web Services Toolkit Overview"
- Chapter 9, "Calendar Web Services SOAP"
- Chapter 10, "Calendar Web Services Client-Side Java Implementation"
- Chapter 11, "Calendar Web Services Status Codes"

8 Calendar Web Services Toolkit Overview

This chapter provides an overview of Oracle Calendar web services and the web services toolkit.

Related documents:

- **IETF iCal RFC** http://www.ietf.org/rfc/rfc2445.txt
- **Web Services Activity** http://www.w3.org/2002/ws/
- **Web Services Description Language (WSDL)** http://www.w3.org/TR/wsdl
- **Simple Object Access Protocol (SOAP) 1.1** http://www.w3.org/TR/SOAP/

About Web Services

Oracle Calendar web services is a component of the Oracle Calendar application system, which handles application level services. Web services allows applications to retrieve, through common XML queries, calendaring data for display in any portal, client application, or backend server. iCal data is coded in XML, wherein iCal becomes xCal. SOAP is used to encapsulate the messages for delivery. The calendaring data web services SOAP is stored directly on the Calendar Server store. This is in effect the CWSL, or Calendar web services Language.

Oracle Calendar web services is comprised of three main areas. The first is SOAP, providing the communication protocol by which a client and server exchange "objects" over HTTP. The second is the Web service Description Language (WSDL), with which third-party integrators can query the Web Service for a description of its functionality (similar to IDL). The final element of web services is the registry where a Web Service can be published and located by a client. This element is known as Universal Description, Discovery and Integration (UDDI).

This current implementation does not provide any WSDL or UDDI support. However, future versions may provide the ability to publish WSDL to a UDDI registry.

The Web Services Toolkit

Developers can use the Oracle Calendar web services toolkit to build web services applications and create SOAP 1.1 queries. The toolkit contains the functionality to search, create, modify, and delete calendar events, as well as search tasks. It gives SOAP access to the calendar server database through a series of Java classes, known as the Calendarlet. This allows developers to use a Java IDE, abstracting the XML structure required to build applications.

Use the Calendarlet to create your own clients and integrate calendaring data into your own applications. To integrate calendaring data within any portal, client application, or backend server, you need to be able to make an HTTP connection to the web server hosting web services, generate SOAP messages and parse the SOAP responses (using any technology that can send and receive HTTP strings), and make use of an existing XML toolkit to generate outgoing and parse incoming HTTP strings with a SOAP client toolkit. The toolkit supports the use of HTTP proxies.

Toolkit Contents

The Oracle Calendar web services toolkit includes:

- `Calendarlet.tar`: The calendarlet JAR file.
- `Javadoc.tar`: The Javadoc HTML documentation for the calendarlet.
- `Ws_testtool.tar`: The Java source for the calendar web services toolkit testing tool, including sample source code.

Security

- BasicAuth
  - Uses plain text userIDs and passwords
  - Should configure HTTP Server for SSL, if the Calendar is to be available outside a "trusted" environment
  - Requires a master-node Calendar Server configuration

All authentication is performed by the calendar server. The Java-side classes use web services to pass the information without performing any action on them. However, security information must be set within the Java web services classes in order to authenticate.

9 Calendar Web Services SOAP

This chapter describes how the Oracle Calendar web services toolkit uses Extended Markup Language (XML) and Simple Object Access Protocol (SOAP) to retrieve and store iCalendar objects.

HTTP Headers

The HTTP header for a proper SOAP v1.1 transaction must contain the following elements:

```
POST <uri> HTTP/1.1
Content-Type: text/xml; charset="UTF-8"
Content-Length: <char length>
SOAPAction: <urn>

...soap envelope...
```

The <uri> is typically the URI for the Oracle Calendar application system (typically, `/ocas-bin/ocas.fcgi`). This is used mainly by the Web Server (Oracle HTTP Server or Apache) to identify the application system, invoke its internal fcgi protocol module, and pass the request to web services.

Within Oracle Collaboration Suite, to bypass Oracle Single Sign-On mechanism (SSO), a separate URL may be required (typically, `/ocws-bin/ocas.fcgi`).

The Content-Type charset identifier is optional. If it is not provided, UTF-8 is assumed. However, UTF-8 is the only charset encoding supported. All other charsets will result in an error.

The HTTP response for an error is a 500 status code (for Internal Server Error). This is returned if the actual SOAP envelope is corrupt (in other words, we cannot determine what the data coming in is) or if a SOAP level error occurs. Keep in mind that all application level errors are returned within a SOAP Fault, along with the 500 HTTP status code.

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml
Content-Length: <char length>

...Optional soap envelope...
```

If the SOAP envelope can be properly executed, the SOAP information is correct, and the application level function succeeds, the 200 status code (success) will be returned.

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: <char length>

... Soap envelope...
```

Security and Authentication

This section describes:

- Security and Authentication Design
- Basic Authentication

Although data encryption is a very important security element, at the present time there are no plans to encrypt data within SOAP requests.

Design

Within the SOAP domain, there are many efforts underway to define and standardize the authentication, security, and encryption of SOAP messages. Groups such as W3C, IETF, OASIS, and WS-I are all working toward the same end. Unfortunately, at the time of development of Oracle Calendar web services, no definitive specification had been approved. However, some general trends were respected when defining the features that web services supports, including:

- HTTP SSL and Web-based certificates
- Simple authentication
- An application-specific authentication mechanism (for Oracle Collaboration Suite).

Note: The proposed specification "WS-Security" outlines some authentication mechanisms, however this is a working draft that does not carry industry approval.

The adopted practice with all these mechanisms is to include the required information within a series of SOAP headers, with the exception of HTTP level functionality (that is, SSL and certificates).

```
<SOAP-ENV:Envelope>
     <SOAP-ENV:Header>
          ... some encryption, signature, and
              authentication info goes here ...
     </SOAP-ENV:Header>
     <SOAP-ENV:Body>
          ... a soap method goes here ...
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope >
```

At the application layer, only plain text authentication is supported. The user's password must be provided in plaintext only (NOT a base-64 encoded string) at the beginning of each transaction.

Security is provided at the transport, protocol and application levels. At the HTTP layer, there are two options: Normal or SSL. This layer is handled completely at the Web server level (that is, Apache and Oracle HTTP Server), providing encrypted data between the HTTP client and HTTP server. The Calendar Application Server has no dependencies on this layer.

The SOAP client must support SSL; not all toolkits do.

Basic Authentication

The web services Basic Authentication is implemented using the SOAP header.

The initial version requires a BasicAuth element in the header for each request. If the element is not present, a SOAP Fault is generated.

```
HTTP/1.1 200 OK
Content-Type: text/xml:charset="UTF-8"
Content-Length: <char length>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
```

```
          soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <soap:Header>
          <auth:BasicChallenge
                    xmlns:auth=
               "http://www.soap-authentication.org/2002/01/">
               <Realm>Oracle Web Services</Realm>
          </auth:BasicChallenge>
     </soap:Header>
     <soap:Body>
          <Reply xmlns:cwsl=
               "http://www.oracle.com/WebServices/Calendaring/1.0/">
          </Reply>
     </soap:Body>
</soap:Envelope>
```

`<Realm>` is used to provide a hint to the client. This is a configurable parameter in the ocws.conf file.

```
[basicauth]
Realm=Oracle Web Services           # default
```

A typical SOAP session with Basic Authentication contains the user's credentials within the soap header of the first message.

```
POST <uri> HTTP/1.1
Content-Type: text/xml; charset="UTF-8"
Content-Length: <char length>
SOAPAction: <urn>


<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
          soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <soap:Header>
          <auth:BasicAuth
               xmlns:auth="http://www.soap-authentication.org/2002/01/">
               <Name>myname</Name>
               <Password>mypassword</Password>
          </auth:BasicAuth>
     </soap:Header>
     <soap:Body>
          ...
     </soap:Body>
</soap:Envelope>
```

The user name must be the Calendar Server's User ID. X.400 login is not permitted. Also, the User ID and Password must be properly XML encoded.

If the Basic Authentication fails, a SOAP fault is returned, indicating the source of the problem.

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml
Content-Length: <char length>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
          soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <soap:Body>
          <soap:Fault>
               <faultcode>soap:Server::Data::CalConnection</faultcode>
               <faultstring>A security error occurred</faultstring>
               <detail>
                    <cwsl:Error xmlns:cwsl=
               "http://www.oracle.com/WebServices/Calendaring/1.0/">
                         <Code>0020-00-00-00000017</Code>
                         ...
                    </cwsl:Error>
               </detail>
          </soap:Fault>
     </soap:Body>
</soap:Envelope>
```

The BasicAuth mechanism is to be used mainly for development and testing purposes. Alone, the mechanism provides little security, due to the use of plain text passwords. If this mechanism is to be used in a production environment, an SLL web configuration is highly recommended.

Building Queries

SOAP queries make use of Universal Identifiers (UIDs) and Global Unique Identifiers (GUIDS). The web services API is based around the ability to uniquely identify a Calendar Store object, retrieve it, and store a reference for last use. In web services, the data-independent property to use is:

```
x-oracle-data-guid
```

This Data GUID maps to various data type specific properties stored on the calendar server. For Events, the following properties are available:

```
uid                                # a UID settable upon creation
```

```
x-oracle-event-guid            # identifier of the main event
x-oracle-eventinstance-guid    # identifier of the instance within
                               # the event
x-oracle-data-guid             # mapped to x-oracle-eventinstance-guid
```

For tasks, the following properties are available:

```
uid                         # a UID settable upon creation
x-oracle-data-guid          # generated internally by the Application
                            # Server (OCAS) and cannot be used against
                            # any other product. This will be changed
                            # once the Calendar Store supports the
                            # x-oracle-task-guid attribute.
```

SOAP Envelope

The SOAP Envelope is a predefined XML packet used to identify the SOAP message:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
     soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/1999/XMLSchema">
     ...soap header...
     ...soap body...
</soap:Envelope>
```

`Xsi` and `xsd` are options defining a namespace used within the message; these will appear if required (i.e. if there is an element in the soap header or soap body requiring it).

`Xsd` is used to provide basic predefined type definitions, such as string, integer, etc. `Xsi` is used to define the "type" attribute for an entry.

```
<location xsi:type="xsd:string">Soleil</location>
```

There are 3 main ways of providing type information within SOAP:

- The data content types are agreed to by both parties ahead of time. This is not useful for general SOAP interaction, only one-to-one site integrations.
- Using XML Schemas, where the schema and namespace is used to relate all typing information.

- Using XML Schemas and explicit type attributes, where each element in the SOAP XML tree requires an `xsi:type` attribute.

Since xCal and CWSL have their own XML Schemas, they do not use explicit type attributes.

There are important issues to be outlined at this point; most current SOAP implementations add an XML document header line before the SOAP envelope. However this is not part of the current SOAP v1.1 specification, but an improvement included in SOAP v1.2.

```
<?xml version="1.0" encoding="UTF-8" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/1999/XMLSchema">
     ...soap header...
     ...soap body...
</soap:Envelope>
```

In order to maintain consistency between SOAP implementations, the default behavior is to provide the XML document header if the original request has one.

```
POST <uri> HTTP/1.1
Content-Type: text/xml; charset="UTF-8"
Content-Length: <char length>
User-Agent: <user agent>
SOAPAction: <urn>

...soap envelope...
```

SOAP Body

The SOAP body contains the actual methods used to perform actions on Calendar Server and web services errors.

SOAP Faults

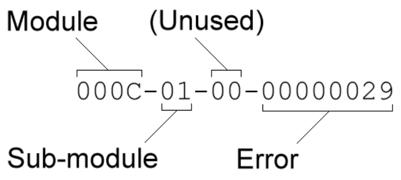
When any kind of error is returned, a SOAP Fault element appears in the Body of the SOAP response. Within a SOAP fault, there are specific elements to be provided:

- A faultcode, which can be one of the following values:
  - VersionMismatch indicating the SOAP namespace is incorrect.
  - Client indicating a problem originating from the incoming message.

  - Server indicating a problem occurred during the processing of the request.
- A faultstring, which is the textual message of the error that has occurred. This is the application system error string. The default string language is English.
- A detail element, used as the container to provide extended information. In our case, the complete application system error log entry is returned to the SOAP client. If server side event logging is set to debug in `ocas.conf`, then Line, FileName, Version, LastMod, and Author are returned.

```
<soap:Body>
     <soap:Fault>
          <faultcode>soap:Server</faultcode>
     <faultstring>Unable to locate the entry in
                    the preferences
          </faultstring>
          <detail>
               <cwsl:Error xmlns:cwsl=
               "http://www.oracle.com/WebServices/Calendaring/1.0/">
                    <Class>Error::Data::CalConnection</Class>
                    <Code>000C-01-00-00000029</Code>
                    <Line>1450</Line>
                    <FileName>UniapiConnection.cpp,v</FileName>
                    <Version>1.43</Version>
                    <LastMod>2002/05/23 20:54:48</LastMod>
                    <Author>frederic</Author>
                    <Date>Web May 29 14:05:42 2002</Date>
                    <PID>19458</PID>
                    <TID>5</TID>
               </cwsl:Error>
          </detail>
     </soap:Fault>
</soap:Body>
```

As an example, the preceding `Code` tag indicates the type of error as follows:



Generally you need only concern yourself with the first and last segments, which in this case are:

- **Module `000C`** = SYS_MODULE_DATAACCESS
- **Error `00000029`** = e_soapSOAPRequestCode_MissingModifyCmd

For a list of Module and Error values, see Chapter 11, "Calendar Web Services Status Codes".

A fault can occur at any point in the access of interaction with various components within the application system and the Calendar Server.

Calendar Web Service Language (CWSL)

CWSL defines the grammar to be used to exchange data between a calendaring SOAP client and calendaring SOAP server. The following methods, taken directly from the CAP draft dated March 2002, provide the main functionality for the Calendaring web service Language. It should be noted that some of the CAP method names are reused here in the CWSL, but the semantics and meaning are changed to reflect a Web-based protocol environment.

The Calendaring language uses "http://www.oracle.com/WebServices/Calendaring/1.0/" as the namespace.

The following Session command is supported:

- **NoOp** performs no operation on the data store, but is used to preauthenticate.

The following Calendaring commands are supported:

- **Create** performs a create of a new meeting.
- **Delete** performs a delete of an event or instance.
- **Modify** performs an update of an event or instance for specific properties.
- **Search** performs a request to retrieve data through the service.

The following Calendaring command is also defined:

- **Ping** performs a simple check to ensure that web services is active.

It is important to note that some of these methods are greatly restricted in this release.

NoOp

This command is used to verify the validity of an authentication SOAP header, without executing anything on the server.

```
POST <uri> HTTP/1.1
Content-Type: text/xml; charset="UTF-8"
Content-Length: <char length>
SOAPAction: http://www.oracle.com/WebServices/Calendaring/ 1.0/NoOp


<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
          soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <soap:Header>
          ...
     </soap:Header>
     <soap:Body>
          <cwsl:NoOp xmlns:cwsl=
               "http://www.oracle.com/WebServices/Calendaring/1.0/">
               <CmdId>a command id</CmdId>
          </cwsl:NoOp>
     </soap:Body>
</soap:Envelope>
```

The web service response for a success command is:

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: <char length>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
          soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     ...
     <soap:Body>
          <cwsl:NoOpReply xmlns:cwsl=
               "http://www.oracle.com/WebServices/Calendaring/1.0/">
               <CmdId>a command id</CmdId>
          </cwsl:NoOpReply>
     </soap:Body>
</soap:Envelope>
```

The NoOp command can only fail if there is a SOAP header problem.

Search

This section describes how to retrieve data using the Search and vQuery commands.

Using Search

This command is used to retrieve Events, Tasks, Contacts, and User information from the Calendar Server.

```
POST <uri> HTTP/1.1
Content-Type: text/xml; charset="UTF-8"
Content-Length: <char length>
SOAPAction: http://www.oracle.com/WebServices/Calendaring/1.0/Search


<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
          soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <soap:Header>
          <h:BasicAuth xmlns:h="http://soap-authentication.org/2002/10/">
               <Name>myname</Name>
               <Password>mypassword</Password>
          </h:BasicAuth>
     </soap:Header>
     <soap:Body>
          <cwsl:Search xmlns:cwsl=
               "http://www.oracle.com/WebServices/Calendaring/1.0/">
               <CmdId>a client id</CmdId>
               <vQuery>
                    ...
               </vQuery>
          </cwsl:Search>
     </soap:Body>
</soap:Envelope>
```

If there was no prior authentication, the `<soap:Header>` can include a `<BasicAuth>` block to ensure all the functionality is performed in one round trip.

`<CmdId>` is a SOAP client-provided string and appears in the response to identify the originating Search entry.

`<vQuery>` is the search query criteria and can only appear once. (See the following section, "vQuery".)

The data returned is contained within a `<cwsl:Reply>`. There is one `<cwsl:Reply>` element for each `<cwsl:Search>` element.

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: <char length>
SOAPAction: http://www.oracle.com/WebServices/Calendaring/1.0/Search
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
          soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <soap:Body>
          <cwsl:Reply xmlns:cwsl=
               "http://www.oracle.com/WebServices/Calendaring/1.0/">
               <CmdId>a client id</CmdId>
               ... data ...
          </cwsl:Reply>
     </soap:Body>
</soap:Envelope>
```

`<CmdId>` appears in the response, identifying the originating Search command.

If an error occurs during the processing of the request, a SOAP fault is returned. If there are multiple `<cwsl:Search>` elements, a fault is also returned.

vQuery

The vQuery forms the basis for requesting data from web service. There are very specific limitations on the vQueries in the initial release.

To limit the potential impact on the Calendar server, the `ocws.conf` file has a few settings to override any query that is received.

```
[webservices]
maxattendee=200          # limit the total number of attendees that can
                         # be returned per instance. The default is 200.
maxresults=200           # limit the total number of meetings or tasks
                         # that can be returned in one query
```

Events can be queried by a single unique identifier or by a series of identifiers:

```
<!-- Ability to fetch a single event -->
<!-- Remove the Where clause to return all events -->
<vQuery>
     <Select>*</Select>
     <From>VEVENT</From>
     <Where>x-oracle-data-guid='event guid'</Where>
</vQuery>

<!-- Ability to fetch a multiple events -->
<vQuery>
     <Select>*</Select>
     <From>VEVENT</From>
     <Where>
          x-oracle-data-guid ='event id 1' OR
          x-oracle-data-guid ='event id 2' OR
          x-oracle-data-guid ='event id 3'
     </Where>
</vQuery>
```

`<Select>` may be provided, however it is not supported in the current version of web services. It may not appear if all attributes are required. The value may be "*" to request all event attributes. The property may appear multiple times in order to list the individual attributes required. If only individual attributes are required, they must be explicitly listed.

Events can be queried by date range:

```
<!-- Ability to fetch events within a time range -->
<vQuery>
     <Select>*</Select>
     <From>VEVENT</From>
     <Where>dtend &gt;= 'starttime' AND dtstart &lt;= 'endtime'</Where>
</vQuery>
```

In this example, `starttime` and `endtime` provide the time range, in UTC, to be returned. Please note the proper XML encoding of the string within the `<Where>` clause.

All other event query forms will generate an error.

The event query result set is returned using the xCal draft specification, embedded within the `<cwsl:Reply>` tag.

```
<soap:Body>
     <cwsl:Reply xmlns:cwsl=
          "http://www.oracle.com/WebServices/Calendaring/1.0/">
          <CmdId>a client id</CmdId>
<xCal:iCalendar xmlns:xCal=
          "http://www.oracle.com/WebServices/Calendaring/1.0/">
     <vcalendar version="2.0" prodid=...>
          <vevent>
               <x-oracle-data-guid>fjldjfdslkjfdksj
               </x-oracle-data-guid>
               <dtstamp>19980309T231000Z</dtstamp>
               <uid>ffdtasfdtasfdta</uid>
               <summary>My event</summary>
               <location>Soleil</location>
               <x-oracle-eventtype>PUBLIC</x-oracle-eventtype>
          </vevent>
     </vcalendar>
```

```
</xCal:iCalendar>
     </cwsl:Reply>
</soap:Body>
```

There will be no sorting of returned data.

Tasks can be queried by a single unique identifier or by a series of identifiers:

```
<!-- Ability to fetch a single task -->
<!-- Remove the Where clause to return all tasks -->
<vQuery>
     <Select>*</Select>
     <From>VTODO</From>
     <Where>x-oracle-data-guid = 'task guid'</Where>
</vQuery>

<!-- Ability to fetch multiple tasks -->
<vQuery>
     <Select>*</Select>
     <From>VTODO</From>
     <Where>
          x-oracle-data-guid = 'task id 1' OR
          x-oracle-data-guid = 'task id 2' OR
          x-oracle-data-guid = 'task id 3'
     </Where>
</vQuery>
```

The `<Where>` clause contains the `x-oracle-data-guid =` string where the right-hand side is an iCal task GUID.

Active Tasks can be queried by date range:

```
<!-- Ability to fetch active tasks by time range -->
<vQuery>
     <Select>*</Select>
     <From>VTODO</From>
     <vCall>
          <ActiveTasks>
               <StartTime>20020701T000000Z</StartTime>
               <EndTime>20020801T000000Z</EndTime>
          </ActiveTasks>
     </vCall>
</vQuery>
```

`<Select>` may be provided, however it is not supported in the current version of web services. It may not appear if all attributes are required. The value may be "*" to request all task attributes. If only individual attributes are required, they must be explicitly listed.

Note: There is no sort order for the returned data.

All other task query forms will generate an error.

The `<vCall>` element indicates the use of an internal procedure (like a database stored procedure). The child element provides the name of the stored procedure to be invoked, as well as the arguments required by the call.

The task query result set is returned using the xCal draft specification, embedded within the `<cwsl:Reply>` tag.

```
<soap:Body>
     <cwsl:Reply xmlns:cwsl=
          "http://www.oracle.com/WebServices/Calendaring/1.0/">
          <CmdId>a client id</CmdId>
<xCal:iCalendar xmlns:xCal=
     "http://www.oracle.com/WebServices/Calendaring/1.0/">
     <vcalendar version="2.0" prodid=...>
          <vtodo>
               <x-oracle-data-guid>JKLFJLJK</x-oracle-data-guid>
               <uid>ffdtasfdtasfdta</uid>
               <dtstamp>19980309T231000Z</dtstamp>
               <summary>My task</summary>
               <priority>2</priority>
          </vevent>
     </vcalendar>
</xCal:iCalendar>
     </cwsl:Reply>
</soap:Body>
```

Create

The ability to create a meeting on the calendar server is provided through the Create SOAP method. Like other methods, the implementation is limited to creating simple meetings or daily notes. Attributes are limited to class, description, dstart, duration, location, priority, summary, start time, UID and event-type.

```
<!-- Sent to Web Services -->
<Body>
```

```
    <Create>
        <CmdId>a command id</CmdId>
        <!-- here is the event select query -->
        <!-- The original event property/values -->
        <iCalendar>
            <vcalendar version="2.0" prodid="any string">
                <vevent>
                    <class>PUBLIC|PRIVATE|CONFIDENTIAL</class>
                    <description>a long description</description>
                    <dtstart>mmmm</dtstart>
                    <duration>nnnnM/ duration >
                    <location>my location</location>
                    <priority>0|2|5|7|9</priority>
                    <summary>a title</summary>
                    <uid>a client uid</uid>
                    <x-oracle-eventtype>APPOINTMENT|DAY EVENT|
                                        DAILY NOTE|HOLIDAY
                    </x-oracle-eventtype>
                </vevent>
            </vcalendar>
        </iCalendar>
    </Create>
</Body>
```

The following attributes are to be specified:

Attribute	Required	Supported
class	yes	yes
description	no	yes
dtstart	yes	yes
dtend	no	no
duration	yes	yes
location	no	yes
organizer	no	no
priority	no	yes
summary	no	yes
uid	no	yes
attendee	no	no

Attribute	Required	Supported
x-oracle-eventtype	yes	yes
x-oracle-event-guid	no	no
x-oracle-eventinstance-guid	no	no
x-oracle-data-guid	no	no

```
<!-- Received from Web Services -->
<!-- failure will result in a SOAP fault -->
<Body>
     <CreateReply>
          <CmdId>a command id</CmdId>
          <x-oracle-data-guid>cccc</x-oracle-data-guid>
     </CreateReply>
</Body>
```

The response will be either a SOAP fault or a CreateReply containing the GUID of the event just created.

The following Create issues are to be expected with this release:

- The user's attendee status will be "ACCEPTED."
- The user's default reminders are not applied (vAlarms).
- The CLASS property is required. There is no way to create a meeting with the user's default class (normal, public, confidential, private) access level applied.
- There is no current mechanism for creating a recurring meeting.
- There is no current mechanism for creating a meeting with additional attendees.

Delete

The Delete SOAP method provides the ability to delete a meeting, daily note or day event. This includes the ability to delete an instance of a repeating/recurring meeting.

```
<!-- Sent to Web Services -->
<Body>
     <Delete>
          <CmdId>a command id</CmdId>
          <!-- here is the event select query -->
          <vQuery>
```

```
                <select>*</select>
                <from>VEVENT</from>
                <where>x-oracle-data-guid = 'some guid'</where>
            </vQuery>
        </Delete>
</Body>
```

The vQuery uniquely identifies a previously returned data GUID. The GUID contains all the information to uniquely identify the instance of a repeating meeting or the recurrence of a recurrence rule.

The vQuery is used to identify the item to be deleted; only one item can be deleted at a time.

```
<!-Received from Web Services -->
<!-- failure will result in a SOAP fault -->
<Body>
    <DeleteReply>
        <CmdId>a command id</CmdId>
        <x-oracle-data-guid>cccc</x-oracle-data-guid>
    </DeleteReply>
</Body>
```

`DeleteReply` contains the GUID of the deleted item.

Modify

The SOAP Modify method permits the modification of xCal properties and attributes. Only a few attributes can be modified. A modify method is made up of 3 sections: the query, the original iCalendar object properties, and the new iCalendar object properties.

```
<!-- Sent to Web Services -->
<Body>
    <Modify>
        <CmdId>a command id</CmdId>
        <!-- here is the event select query -->
        <vQuery>
            ...
        </vQuery>
        <!-- The original event property/values -->
        <iCalendar>
            ...
        </iCalendar>
        <!-- The modified event property/values -->
```

```
          <iCalendar>
               ...
          </iCalendar>
     </Modify>
     </Body>
```

The vQuery uniquely identifies a previously returned data GUID. The GUID contains all the information to uniquely identify the instance of a repeating meeting or the recurrence of a recurrence rule. The vQuery is used to identify the item to be modified; only one item can be modified at a time, including one simple event (meeting, daily note, or day event) or one instance of a repeating meeting. In a modify operation, properties can be changed, added, or removed (see the following table).

The modify reply returns a GUID of the event. It is very important to note that the GUID can change depending on the type of change applied to the calendar server. Some updates require a delete / recreate type of interaction.

```
<!-Received from Web Services -->
<!-- failure will result in a SOAP fault -->
<Body>
     <ModifyReply>
          <CmdId>a command id</CmdId>
          <x-oracle-data-guid>cccc</x-oracle-data-guid>
     </ModifyReply>
</Body>
```

Property Modify

The first iCalendar object contains the properties to be modified, along with the original values. If there are attributes associated with them, those must be present as well. The second iCalendar object contains the new properties values to be applied.

```
          <!-- The original event property/values -->
          <iCalendar>
               <vcalendar>
                    <vevent>
                         <summary>my old title</summary>
                         <location>my old location</location>
                    </vevent>
               </vcalendar>
          </iCalendar>
          <!-- The modified event property/values -->
```

```
<iCalendar>
     <vcalendar>
          <vevent>
               <summary>my new title</summary>
               <location>my new location</location>
          </vevent>
     </vcalendar>
</iCalendar>
```

Property Add

The first iCalendar object does not contain any reference to the property to be added. The second iCalendar object contains the new property and value.

```
<!-- The original event property/values -->
<iCalendar>
     <vcalendar>
          <vevent>
          </vevent>
     </vcalendar>
</iCalendar>
<!-- The modified event property/values -->
<iCalendar>
     <vcalendar>
          <vevent>
               <summary>my new title</summary>
          </vevent>
     </vCalendar>
</iCalendar>
```

Property Delete

The first iCalendar object contains the original property and value. The second iCalendar object does not contain the property.

```
<!-- The original event property/values -->
<iCalendar>
     <vcalendar>
          <vevent>
               <summary>my old title</summary>
          </vevent>
     </vcalendar>
</iCalendar>
<!-- The modified event property/values -->
```

```
<iCalendar>
     <vcalendar>
          <vevent>
          </vevent>
     </vcalendar>
</iCalendar>
```

Modifiable Properties

Property	Modify	Add	Delete
class	yes	no	no
description	yes	yes	yes
dtstart	yes	no	no
dtend	yes	no	no
duration	no	no	no
location	yes	yes	yes
organizer	no	no	no
priority	yes	no	no
summary	yes	yes	yes
uid	no	no	no
attendee	no	no	no
x-oracle-eventtype	no	no	no
x-oracle-event-guid	no	no	no
x-oracle-eventinstance-guid	no	no	no
x-oracle-data-guid	no	no	no

If there are other properties within the modify SOAP method, a SOAP fault will be generated (i.e. class, uid, x-oracle-eventtype, Web Conferencing attributes).

Ping

This command is used to test to see if the Oracle Calendar web service application server is active. The command has no other effect on web services.

```
POST <uri> HTTP/1.1
Content-Type: text/xml; charset="UTF-8"
Content-Length: <char length>
SOAPAction: http://www.oracle.com/WebServices/Calendaring/1.0/Ping


<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
          soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <soap:Body>
          <cwsl:Ping xmlns:cwsl=
               "http://www.oracle.com/WebServices/Calendaring/1.0/">
          </cwsl:Ping>
     </soap:Body>
</soap:Envelope>
```

The web service response for a success command is:

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: <char length>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
          soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <soap:PingReply xmlns:cwsl=
               "http://www.oracle.com/WebServices/Calendaring/1.0/">
     </soap: PingReply>
</soap:Envelope>
```

xCal

Once the protocol aspects have been taken care of, we are left with the data. For events and tasks, the xCal Draft is used as the official reference for the data format output in the SOAP response.

The current implementation of web services does not support the retrieval of repeating and recurring meetings as a whole. When a Search is performed, any meeting with instances or recurrence rules stored on the server is expanded to separate each instance into an individual meeting. This helps processing and UI generation. The interpretation of recurrence rules is very complicated and error prone.

In order to understand how web services supports repeating and recurring meetings stored on the calendar server, a few examples are provided.

NOTE: only vEvents and vTodos are supported in the xCal specification; vJournal and vFreebusy are not.

It is important to note that not all xCal elements and properties nor all calendar server attributes are supported in this release.

Simple Events

```
<vcalendar>
     <vevent>
          <class>CONFIDENTIAL</class>
          <description>a description</desription>
          <dtend>20021101T120000Z</dtend>
          <dtstart>20021101T110000Z</dtstart>
          <location></location>
          <organizer cn="James Baldwin">
               mailto:james.baldwin@oracle.com
          </organizer>
          <priority>1</priority>
          <status>CONFIRMED</status>
          <summary>a meeting</summary>
          <uid>ORACLE:CALSERV:EVENT:48390483290843290</uid>
          <attendee cn="James Baldwin" partstat="ACCEPTED">
               mailto:james.baldwin@oracle.com
          </attendee>
          <x-oracle-eventtype>APPOINTMENT</x-oracle-eventtype>
          <x-oracle-event-guid>fdjskljfdlkj</x-oracle-event-guid>
          <x-oracle-eventinstance-guid>fdjskljfdlkj
          </x-oracle-eventinstance-guid>
          <x-oracle-data-guid>fdjskljfdlkj</x-oracle-data-guid>
     </vevent>
</vcalendar>
```

Repeating Events

If a repeating meeting were pulled directly from the calendar server, it would have one vCalendar element with multiple, related vEvents. The main vEvent would contain an `rrule` element outlining the rule for the meeting followed by instance, exception, and time zone information. All event-guids would be the same through the vCalendar, but the instance-guids would be different.

This is an example of a repeating meeting:

```
<vcalendar>
     <vevent>
          <uid>ORACLE:CALSERV:EVENT:6205/391</uid>
          <summary>SDK Development Review</summary>
          <status>CONFIRMED</status>
          <priority>5</priority>
          <organizer cn="Ray Bradbury">mailto:</organizer>
          <location>Salle Saturne (Client/Server)</location>
          <description>
               Review of SDK 9.0.4 and Calendar Web Services
               9.0.4 Development Plans with PM
          </description>
          <dtstart tzid="EST5EDT">20021002T103000</dtstart>
          <dtend tzid="EST5EDT">20021002T120000</dtend>
          <class>PUBLIC</class>
          <rrule>FREQ=DAILY;INTERVAL=1;COUNT=1</rrule>
          <rdate>20021030T153000Z</rdate>
          <attendee cn="James Baldwin"
                    partstat="NEEDS-ACTION">mailto:</attendee>
          <attendee partstat="NEEDS-ACTION">
               mailto:Salle Saturne (Client/Server)</attendee>
          <attendee cn="Albert Camus"
                    partstat="NEEDS-ACTION">mailto:</attendee>
          <attendee cn="Lewis Carroll"
                    partstat="NEEDS-ACTION">mailto:</attendee>
          <attendee cn="Frederic Chopin"
                    partstat="ACCEPTED">mailto:</attendee>
          <attendee cn="Emily Dickinson"
                    partstat="NEEDS-ACTION">mailto:</attendee>
          <attendee cn="Ray Bradbury"
                    partstat="ACCEPTED">mailto:</attendee>
          <attendee cn="Alfred Hitchcock"
                    partstat="NEEDS-ACTION">mailto:</attendee>
          <attendee cn="Yasunari Kawabata"
                    partstat="ACCEPTED">mailto:</attendee>
          <x-oracle-eventtype>APPOINTMENT</x-oracle-eventtype>
          <x-oracle-event-guid>fdjskljfdlkj</x-oracle-event-guid>
     </vevent>
     <vevent>
          <uid>ORACLE:CALSERV:EVENT:6205/391</uid>
          <summary>SDK Development Review</summary>
     <status>CONFIRMED</status>
     <recurrance-id>20021030T153000Z</recurrance-id>
     <priority>5</priority>
     <organizer cn="Ray Bradbury">mailto:</organizer>
     <location>Salle Saturne (Client/Server)</location>
     <description></description>
     <dtstart>20021030T153000Z</dtstart>
     <dtend>20021030T170000Z</dtend>
     <class>PUBLIC</class>
     <attendee cn="James Baldwin"
          partstat="ACCEPTED">mailto:</attendee>
     <attendee partstat="NEEDS-ACTION">
          mailto:Salle Saturne (Client/Server)</attendee>
     <attendee cn="Albert Camus"
          partstat="ACCEPTED">mailto:</attendee>
     <attendee cn="Lewis Carroll"
          partstat="ACCEPTED">mailto:</attendee>
     <attendee cn="Frederic Chopin"
          partstat="ACCEPTED">mailto:</attendee>
     <attendee cn="Emily Dickinson"
          partstat="ACCEPTED">mailto:</attendee>
     <attendee cn="Ray Bradbury"
          partstat="ACCEPTED">mailto:</attendee>
     <attendee cn="Alfred Hitchcock"
          partstat="ACCEPTED">mailto:</attendee>
     <attendee cn="Yasunari Kawabata"
          partstat="DECLINED">mailto:</attendee>
     <x-oracle-eventtype>APPOINTMENT</x-oracle-eventtype>
     <x-oracle-event-guid>fdjskljfdlkj</x-oracle-event-guid>
     <x-oracle-eventinstance-guid>fdjskljfdlkj
     </x-oracle-eventinstance-guid>
     <x-oracle-data-guid>fdjskljfdlkj</x-oracle-data-guid>
</vevent>
<timezone>
     <tzid>EST5EDT</tzid>
     <daylight>
          <tzoffsetto>-400</tzoffsetto>
          <tzoffsetfrom>-500</tzoffsetfrom>
          <dtstart>19910401T020000</dtstart>
          <rrule>
          FREQ=YEARLY;UNTIL=UNTIL=20750407T020000Z;BYMONTH=4;BYDAY=1SU
          </rrule>
          <tzname>
               Eastern Standard Time, Eastern Daylight Time
          </tzname>
          </daylight>
          <standard>
               <tzoffsetto>-0500</tzoffsetto>
               <tzoffsettfrom>-0400</tzoffsettfrom>
               <dtstart>19911025T020000</dtstart>
               <rrule>
          FREQ=YEARLY;UNTIL=20751031T020000Z;BYMONTH=10;BYDAY=-1SU
               </rrule>
               <tzname>
                    Eastern Standard Time, Eastern Daylight Time
               </tzname>
          </standard>
     </timezone>
</vcalendar>
```

Recurring Events

Like a repeating meeting, an event with a recurrence rule requires an interpretation of the data with the vCalendar. Again the recurrence rule provides the information required to interpret or generate all the specific meeting information. This requires complicated processing on the client side.

The Oracle Connector for Outlook can create events with pure recurrence rules.

```
<vcalendar>
     <vevent>
          <uid>ORACLE:CALSERV:EVENT:10750/391</uid>
          <summary>test recur weekly 3 times</summary>
          <status>TENTATIVE</status>
          <priority>5</priority>
          <organizer cn="Stephen King">mailto:</organizer>
          <location></location>
          <description></description>
          <dtstart tzid="EST5EDT">20021025T123000</dtstart>
          <dtend tzid="EST5EDT">20021025T133000</dtend>
          <created>20021101T191154Z</created>
          <class>PUBLIC</class>
          <rrule>
               FREQ=WEEKLY;BYDAY=FR;COUNT=3;INTERVAL=1;WKST=SU
          </rrule>
          <rdate>20021101T173000Z</rdate>
          <attendee cn="Stephen King"
               partstat="ACCEPTED">mailto:</attendee>
          <x-oracle-event-guid>fdjskljfdlkj</x-oracle-event-guid>
          <x-oracle-eventtype>APPOINTMENT</x-oracle-eventtype>
```

```
</vevent>
<vevent>
     <uid>ORACLE:CALSERV:EVENT:10750/391</uid>
     <summary>test recur weekly 3 times</summary>
     <status>CONFIRMED</status>
     <recurrance-id>20021101T173000Z</recurrance-id>
     <priority>5</priority>
     <organizer cn="Stephen King">mailto:</organizer>
     <location></location>
     <description></description>
     <dtstart>20021101T173000Z</dtstart>
     <dtend>20021101T183000Z</dtend>
     <created>20021101T191154Z</created>
     <class>PUBLIC</class>
     <attendee cn="Stephen King"
     partstat="ACCEPTED">mailto:</attendee>
     <x-oracle-event-guid>fdjskljfdlkj</x-oracle-event-guid>
     <x-oracle-eventtype>APPOINTMENT</x-oracle-eventtype>
     <x-oracle-eventinstance-guid>fdjskljfdlkj
     </x-oracle-eventinstance-guid>
     <x-oracle-data-guid>fdjskljfdlkj</x-oracle-data-guid>
</vevent>
<vtimezone>
     <tzid>EST5EDT</tzid>
     <daylight>
          <tzoffsetto>-0400</tzoffsetto>
          <tzoffsettfrom>-0500</tzoffsettfrom>
          <dtstart>19910401T020000</dtstart>
          <rrule>
     FREQ=YEARLY;UNTIL=20750407T020000Z;BYMONTH=4;BYDAY=1SU
          </rrule>
          <tzname>
          Eastern Standard Time, Eastern Daylight Time
          </tzname>
     </daylight>
     <standard>
          <tzoffsetto>-0500</tzoffsetto>
          <tzoffsettfrom>-0400</tzoffsettfrom>
          <dtstart>19911025T020000</dtstart>
          <rrule>
FREQ=YEARLY;UNTIL=20751031T020000Z;BYMONTH=10;BYDAY=-1SU
          </rrule>
          <tzname>
          Eastern Standard Time, Eastern Daylight Time
          </tzname>
          </standard>
     </vtimezone>
</vcalendar>
```

Expanded Filtered Recurring/Repeating Events

web services will provide repeating and recurrence rules as broken up individual elements ("expanded").

Expanded repeating and recurring meetings will have the following form:

```
<vcalendar>
     <vevent>
          <uid>ORACLE:CALSERV:EVENT:6205/391@20021030T153000Z</uid>
          <summary>SDK Development Review</summary>
          <status>CONFIRMED</status>
          <priority>5</priority>
          <organizer cn="Ray Bradbury">mailto:</organizer>
          <location>Salle Saturne (Client/Server)</location>
          <description></description>
          <dtstart>20021030T153000Z</dtstart>
          <dtend>20021030T170000Z</dtend>
          <class>PUBLIC</class>
          <attendee cn="James Baldwin"
               partstat="ACCEPTED">mailto:</attendee>
          <attendee partstat="NEEDS-ACTION">
               mailto:Salle Saturne (Client/Server)</attendee>
          <attendee cn="Albert Camus"
               partstat="ACCEPTED">mailto:</attendee>
          <attendee cn="Lewis Carroll"
               partstat="ACCEPTED">mailto:</attendee>
          <attendee cn="Frederic Chopin"
               partstat="ACCEPTED">mailto:</attendee>
          <attendee cn="Emily Dickinson"
               partstat="ACCEPTED">mailto:</attendee>
          <attendee cn="Ray Bradbury"
               partstat="ACCEPTED">mailto:</attendee>
          <attendee cn="Alfred Hitchcock"
               partstat="ACCEPTED">mailto:</attendee>
          <attendee cn="Yasunari Kawabata"
               partstat="DECLINED">mailto:</attendee>
          <x-oracle-event-guid>fdjkfdsjfljkds</x-oracle-event-guid>
          <x-oracle-eventtype>APPOINTMENT</x-oracle-eventtype>
          <x-oracle-eventinstance-guid>fdjskljfdlkj
          </x-oracle-eventinstance-guid>
          <x-oracle-data-guid>fdjskljfdlkj</x-oracle-data-guid>
     </vevent>
</vcalendar>
```

Daily Notes

```
<vcalendar>
     <vevent>
          <class>CONFIDENTIAL</class>
          <description>a description</description>
          <dtend value="DATE">20021101</dtend>
          <dtstart value="DATE">20021101</dtstart>
          <organizer cn="Par Lagerkvist">
               mailto:par.lagerkvist@oracle.com
          </organizer>
          <priority>3</priority>
          <status>CONFIRMED</status>
          <summary>a daily note</summary>
          <uid>ORACLE:CALSERV:EVENT:49304932-04932-09</uid>
          <attendee cn="Par Lagerkvist" partstat="ACCEPTED">
               mailto:par.lagerkvist@oracle.com
          </attendee>
          <x-oracle-event-guid>fdjskljfdlkj</x-oracle-event-guid>
          <x-oracle-eventtype>DAILY NOTE</x-oracle-eventtype>
          <x-oracle-eventinstance-guid>fdjskljfdlkj
          </x-oracle-eventinstance-guid>
          <x-oracle-data-guid>fdjskljfdlkj</x-oracle-data-guid>
     </vevent>
</vcalendar>
```

Day Events

```
<vcalendar>
     <vevent>
          <class>CONFIDENTIAL</class>
          <description>a description</description>
          <dtend value="DATE">20021101</dtend>
          <dtstart value="DATE">20021101</dtstart>
          <organizer cn="Par Lagerkvist">
               mailto:par.lagerkvist@oracle.com
          </organizer>
          <priority>3</priority>
          <status>CONFIRMED</status>
          <summary>a day event</summary>
```

```
            <uid>ORACLE:CALSERV:EVENT:49304932-04932-09</uid>
            <attendee cn="Par Lagerkvist" partstat="ACCEPTED">
                 mailto:par.lagerkvist@oracle.com
            </attendee>
            <x-oracle-event-guid>fdjskljfdlkj</x-oracle-event-guid>
            <x-oracle-eventtype>DAY EVENT</x-oracle-eventtype>
            <x-oracle-eventinstance-guid>fdjskljfdlkj
            </x-oracle-eventinstance-guid>
            <x-oracle-data-guid>fdjskljfdlkj</x-oracle-data-guid>
       </vevent>
</vcalendar>
```

Holidays

Holidays are represented as:

```
<vcalendar>
     <vevent>
          <categories>
               <item>Holiday</item>
          </categories>
          <class>PUBLIC</class>
          <description>a description</description>
          <dtend value="DATE">20021031</dtend>
          <dtstart value="DATE">20021031</dtstart>
          <organizer cn="Par Lagerkvist">
               mailto:par.lagerkvist@oracle.com
          </organizer>
          <priority>3</priority>
          <status>CONFIRMED</status>
          <summary>a holiday</summary>
          <uid>ORACLE:CALSERV:EVENT:49304932-04932-09</uid>
          <attendee cn="Par Lagerkvist" partstat="ACCEPTED">
               mailto:par.lagerkvist@oracle.com
          </attendee>
          <x-oracle-event-guid>fdjskljfdlkj</x-oracle-event-guid>
          <x-oracle-eventtype>HOLIDAY</x-oracle-eventtype>
          <x-oracle-eventinstance-guid>fdjskljfdlkj
          </x-oracle-eventinstance-guid>
          <x-oracle-data-guid>fdjskljfdlkj</x-oracle-data-guid>
     </vevent>
</vcalendar>
```

Tasks

```
<vcalendar>
     <vtodo>
          <class>PRIVATE</class>
          <completed>20021002T210000Z</completed>
          <created>20021002T210000Z</created>
          <description>the task description</description>
          <percent>0</percent>
          <priority>9</priority>
          <summary>The task title</summary>
          <uid>fdjskljfdlkj</uid>
          <due>20021102T210000Z</due>
          <x-oracle-data-guid>
          ORACLE:CALSERV:TASK:328321890328/489043209
          </x-oracle-data-guid>
     </vtodo>
</vcalendar>
```

X-ORACLE-EVENTTYPE

This new property is used to identify the event type stored on the calendar server. This attribute is selectable in the <select> clause and is returned as an element in a vEvent. The possible values are:

- APPOINTMENT - to identify a regular blocking meeting.
- DAILY NOTE - to identify a non-blocking note associated with a calendar day.
- DAY EVENT - to identify a non-blocking all day calendaring event.
- HOLIDAY - to identify a non-blocking holiday specialization of a day event.

Priority Mapping

In xCal, priorities are mapping to specific Calendar Server levels for maximum compatibility with the suite of calendar clients.

- 0 maps to HIGHEST
- 1 maps to HIGH
- 5 maps to NORMAL
- 7 maps to LOW
- 9 maps to LOWEST

10 Calendar Web Services Client-Side Java Implementation

This chapter describes the design of the set of Java classes used to provide contextual collaboration through the access of calendaring data via Oracle Calendar web services. These "Calendarlet" classes attempt to hide the many details of using web services technology in a Java environment.

The class implementation does not attempt to provide all the iCalendar properties and attributes, but is focused on the elements supported in the V2 implementation of the Oracle Calendar web services toolkit.

Note: You can find JavaDoc information and TestTool samples in the Oracle Calendar web services toolkit.

Java Classes

There are a few general steps to follow when using the Calendarlet classes:

- Initialize your authentication mechanism.
- Initialize your Query, including data type.
- Bind the authentication and query object to a Calendarlet instance.
- Set the target URL in the Calendarlet instance.
- Make the SOAP call.
- Parse the results.

The Calendarlet class implementation relies heavily on Apache SOAP classes to perform most of the protocol level handling. For incoming and outgoing messages, these same Apache SOAP classes are used, along with W3C DOM classes. To generate outgoing messages, Calendarlet and iCalendar classes are instantiated and set on parent classes. To generate the final XML stream, all classes implement a getElement() method. This is intended to build an XML DOM representation of the SOAP message to be transmitted. The lower level Apache SOAP calls require this DOM structure to obtain the final stream.

For incoming messages, the Calendarlet and iCalendar classes are reconstructed through the unmarshall() static method on each class, again using the XML DOM received from the lower level Apache SOAP classes. This unmarshalling of the DOM consists of the parent class recognizing a child tag and invoking that child's class unmarshall() method.

If for any reason there is an XML parsing error, a low level Apache SOAP exception is thrown; the Calendarlet classes will never get a chance to parse the data. If there is a contextual error, meaning the XML is valid but elements are in the wrong place or not recognized, a Calendarlet exception will be thrown.

Ideally, all incoming xCal (the XML binding of iCalendar) can have extended elements within the data. However, for this implementation, extended elements will only be handled at the vEvent level.

The Calendarlet class provides some debugging support. There are two main features:

- The ability to capture the input and output buffers; the method setWantIOBuffers() must be called before invoking a SOAP method (not recommended for a final deployment). Both the input and output buffers are captured and stored in the CalendaringResponse class.
- The ability to get the total processing time (in milliseconds) of the SOAP request, also stored within the CalendarletResponse.

Fetching Data

Using the Java Calendarlet classes, the following code could be used to generate the SOAP request:

```
// initialize the authentication information
// and set the user id
BasicAuth auth = new BasicAuth();

auth.setName("*** userid ***");
auth.setPassword("*** password ***");
```

```
// initialize the event search command and query
SearchCommand search = new SearchCommand();

search.setCmdId("my cmd id 1");

// create a query to retrieve unconfirmed events
vQuery            query       = new vQuery();
query.setFrom(vQuery.k_queryFromEvent);

// determine the datestamps for a weeks worth of events

// use the CalendarUtils to get a proper timestamp with
// time zone information set properly
Calendar today = CalendarUtils.getToday();

int dayOfWeek = today.get(Calendar.DAY_OF_WEEK);

Calendar beginWeek = (Calendar)today.clone();
Calendar endWeek   = (Calendar)today.clone();

beginWeek.add(Calendar.DATE, 1 - dayOfWeek);
endWeek.add(Calendar.DATE, 8 - dayOfWeek);
endWeek.add(Calendar.MINUTE, -1);

// use the CalendarUtils to help generate a date range query
query.setWhere(CalendarUtils.getDateRangeQuery(beginWeek, endWeek));

search.setQuery(query);

// create the calendar client SOAP stub
// and set the basic authentication header
Calendarlet cws = new Calendarlet();

// set the Web Services host URL
cws.setEndPointURL("http://www.thehost.com");

cws.setAuthenticationHeader(auth.getElement());

// make the SOAP call
CalendaringResponse response = cws.Search(search.getElement());
```

The resulting response contains iCalendar objects, with the requested properties returned. Using the Calendarlet Java classes, the iCalendars can be traversed to find the vEvents.

```
// get the SOAP reply
Reply reply = (Reply)response.getCalendarReply();

// traverse all the iCalendar objects
Vector someiCalendars =  iCalendar.unmarshallVector(
                                               reply.getEntries());
int    numiCalendars  = someiCalendars.size();

for (int i = 0;  i < numiCalendars; i++)
{
     iCalendar iCalObj = (iCalendar)someiCalendars.get(i);

     // traverse all the vCalendar objects
     Vector somevCalendars = iCalObj.getvCalendars();
     int    numvCalendars  = somevCalendars.size();

     for (int j = 0;  j < numvCalendars;  j++)
     {
          vCalendar vCalObj = (vCalendar)somevCalendars.get(j);

          // traverse all the vEvent objects
          Vector somevEvents = vCalObj.getComponents();
          int    numvEvents  = somevEvents.size();

          for (int k = 0;  k < numvEvents;  k++)
          {
               vEvent vEventObj = (vEvent)somevEvents.get(i);

               // get the specific properties
               String title     = vEventObj.getSummary();
               String dtstart   = vEventObj.getDtStart();
               String dtend     = vEventObj.getDtEnd();
               String eventType = vEventObj.getXEventType();

               // do something interesting with the meeting info
          }
     }
}
```

Soap Faults and Exceptions

There are two types of errors that can occur with the web services toolkit: A Java exception or a SOAP fault.

The Java Exception originates from the Calendarlet class, the underlying Apache SOAP or W3C DOM classes, or the Java Runtime. For each SOAP method that is invoked on the Calendarlet class, an exception may be thrown as a result of some internal processing error or an XML parsing problem. These are typically client-side unexpected errors that must be properly handled.

The SOAP Fault is the result of an error from the Oracle Calendar web service (that is, a remote server-side error). Whenever a server-side error occurs, the web service returns a SOAP Fault as the response to the HTTP transaction. There is no Java-based exception thrown. Within a SOAP Fault, the details field may contain an Oracle Calendar web services Error object, with an important error code. For a list of error codes, see Chapter 11, "Calendar Web Services Status Codes".

A CalendarUtils method can help determine whether a SOAP fault has occurred and retrieve the web services error.

```
// Calendar response
CalendaringResponse response = cws.Search(...);

// get the vector of entries embedded
// in the SOAP body
Vector bodyEntries = response.getBodyEntries();

// determine if there was a SOAP Fault
if (!CalendarUtils.isSOAPFault(bodyEntries))
{
     // do regular processing
}
else
{
     // get the SOAP fault object
     org.apache.soap.Fault soapFault =
               CalendarUtils.getSOAPFault(bodyEntries);

     // get the calendar Web Services error
     Error calendaringError = Error.unmarshall(soapFault);

     // get the Web Services error code
     String errorCode = calendaringError.getCode();
}
```

Local Time

There are two important date formats to be aware of: `Date` and `DateTime`. The `DateTime` format contains both date and time information within the string, while `Date` contains only time information. `DateTime` is generally used for regular meetings, while `Date` is used for Day Events, Daily Notes, and Holidays.

To help generate UTC datetime strings for a vQuery, the CalendarUtils will have a class to take a standard Java Calendar class object and generate a proper string of the form "yyyyMMddThhmmssZ". Java's Calendar class can have a Java time zone associated with it. It is up to the user of the Calendarlet classes to determine the proper time zone and set it in the Java Calendar class.

```
// set the date through some mechanism
// ensure the proper time zone is set
TimeZone localTimezone = TimeZone.getDefault();
Calendar theDate       = Calendar.getInstance(localTimezone);

String utcString = CalendarUtils.getUTCDateTime(theDate);
```

To help generate UTC date strings for a vQuery, the CalendarUtils will have a class to take a standard Java Calendar class object and generate a proper string of the form "yyyyMMdd". Java's Calendar class can have a Java time zone associated with it. It is up to the user of the Calendarlet classes to determine the proper time zone and set it in the Java Calendar class.

```
// set the date through some mechanism
// ensure the proper time zone is set
TimeZone localTimezone = TimeZone.getDefault();
Calendar theDate       = Calendar.getInstance(localTimezone);

String utcString = CalendarUtils.getUTCDate(theDate);
```

Since many calendaring query operations are relative to today's date, an additional CalendarUtils method is provided to help base vQuery datetime stamps. The method will return a datetime stamp of midnight today, local time and will be of the form "yyyyMMddTxxxx00Z", where xxxx is the hour and minute offset from UTC (note some time zones are half-hour offsets).

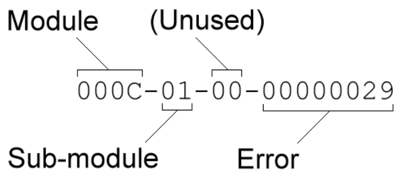
```
String utcToday = CalendarUtils.getToday();
```

Typical calendar server query time ranges are from local midnight of a specific date to one minute before midnight of the day before the last date. For example, if today is June 01, 2003 in EST time, the getToday() method will return 20030602T040000Z. For a day date range, the end date would be 20030603T035900Z.

11 Calendar Web Services Status Codes

Each status code in a SOAP fault is made up of four segments; the first describes the source module, the last describes the error type. (The second and third segments are not generally used at this time.) This chapter lists the Module and Error codes that can be displayed in a SOAP fault.

A sample `Code` tag might look as follows:



Generally you need only concern yourself with the first and last segments, which, in the preceding example, we can determine from the tables in this chapter to be:

- **Module `000C`** = SYS_MODULE_DATAACCESS
- **Error `00000029`** = e_soapSOAPRequestCode_MissingModifyCmd

For more details on working with SOAP faults, see Chapter 9, "Calendar Web Services SOAP".

Module Codes

Each of these 64-bit codes corresponds to the source Module of an error in Oracle Calendar. Each Module name is preceded by "SYS_MODULE_".

Code	Module	Description
0x0000	NONE	N/A
0x0001	UNIAPI	Calendar Server
0x0002	APPLICATION	Calendar Applications
0x0003	MEMORYMGR	Memory Manager
0x0004	CONNECTION	Connection Service
0x0005	DISPATCH	Dispatch Service
0x0006	LINKDB	Link Database Service
0x0007	MESSAGECAT	Message Catalogue Service
0x0008	PREFERENCE	Preference Service
0x0009	REGISTRY	Registry Services
0x000A	SESSIONDB	Session Database Service
0x000B	SYSTEM	System Service
0x000C	DATAACCESS	Data Access Service
0x000D	DATAMANAGER	Data Manager Service
0x000E	SYNC	Synchronization Service
0x000F	SYNCML	SyncML Module
0x0010	MALCLIENT	MAL Module
0x0011	NLSSERVICE	NLS Service
0x0012	MALSYSTEM	MAL System Service
0x0013	PLUGINCONFIG	Plug-in (component) Service
0x0014	MOBILE	Mobile Module
0x0015	XMLSERVICE	SOAP Module
0x0016	WINDOWS	Windows Error
0x0017	FCGI	FCGI Toolkit Error

Error Codes

The following table lists the error codes that can be generated within the 0015 SOAP module.

Code	Error
00000001	Ok = 1
00000002	NoInputData
00000003	NoSoapAction
00000004	NoPost
00000005	NoUTF8
00000006	MethodNotSuppported
00000007	NoSoapContent
00000008	UnexpectedBasicAuthFailure
00000009	WrongSoapMethod
0000000A	SOAPActionMismatch
0000000B	SoapActionNamespace
0000000C	VersionMismatch
0000000D	BasicAuthNamespace
0000000E	Unused001
0000000F	SearchGenerationError
00000010	UnexpectedParserError
00000011	SAX2FatalError
00000012	SAX2Error
00000013	SAX2Warning
00000014	SAXException
00000015	XMLException
00000016	PingGenerationError
00000017	SecurityError
00000018	Unused002
00000019	Unused003

Code	Error
0000001A	Unused004
0000001B	Unused005
0000001C	Unused006
0000001D	SearchNamespace
0000001E	MissingSearchCmd
0000001F	BadSearchFrom
00000020	BasicAuthMissingName
00000021	BasicAuthMissingPassword
00000022	MissingQuery
00000023	Unexpected
00000024	CreateGenerationError
00000025	CreateNamespace
00000026	MissingCreateCmd
00000027	ModifyGenerationError
00000028	InvalidModifyNamespace
00000029	MissingModifyCmd
0000002A	MissingModifyQuery
0000002B	InvalidModifyQueryFrom
0000002C	MissingModifyOriginalElement
0000002D	MissingModifyModifiedElement
0000002E	DeleteGenerationError
0000002F	DeleteNamespace
00000030	MissingDeleteCmd
00000031	MissingDeleteQuery
00000032	InvalidDeleteQueryFrom
00000033	MissingCreateElement
00000034	InvalidCreateElement
00000035	NoopGenerationError

Code	Error
00000036	InvalidAuthentication
00000037	InvalidTrustedAuthNamespace
00000038	TrustedAuthMissingName
00000039	TrustedAuthMissingToken
0000003A	UnexpectedTrustedAuthFailure
0000003B	InvalidProxyAuthNamespace
0000003C	ProxyAuthMissingName
0000003D	ProxyAuthMissingAppName
0000003E	ProxyAuthMissingAppPassword
0000003F	UnexpectedProxyAuthFailure
00000040	UnableToLocateEvent
00000041	MoreThanOneEvent
00000042	MissingEventType
00000043	MissingClass
00000044	MissingDtStart
00000045	MissingDuration
00000046	UnableToConvertxCal
00000047	InvalidClass
00000048	MissingUID
00000049	UnsupportedCreateEventType
0000004A	InvalidModifyComponent
0000004B	ModifyFoundDataGuidProperty
0000004C	BadPriority
0000004D	QuietLoginGenerationError
0000004E	QuietLoginNamespace
0000004F	ModifyClassNotSupported
00000050	ModifyDtStartNotSupported
00000051	ModifyDtEndNotSupported

Code	Error
00000052	ModifyDurationNotSupported
00000053	ModifyEventTypeNotSupported
00000054	ModifyPriorityNotSupported
00000055	ModifyDataGuidNotSupported
00000056	ModifyUidNotSupported
00000057	UnsupportedDtEnd
00000058	UnsupportedDataGuid
00000059	UnsupportedOrganizer
0000005A	UnsupportedAttendee
0000005B	ModifyOrganizerNotSupported
0000005C	ModifyAttendeeNotSupported
0000005D	SearchScore
0000005E	ModifySearchScoreNotSupported

Index

A

accented characters (SDK), 7-3
accessibility, document, xi
alarms, 2-6
applications, types of
  SDK, 1-1, 1-2
  web services, 8-2
audience, xi

B

blocking, 4-2

C

CAPI_AuthenticateAsSysop, 3-69
CAPI_FLAG_STORE_DELPROPS, 5-1
CAPI_FLAG_STORE_MODPROPS, 5-1
CAPI_STAT_API, 6-1
CAPI_STAT_API_BADPARAM, 6-1
CAPI_STAT_API_CALLBACK, 6-1
CAPI_STAT_API_CALLBACK_ERROR, 6-1
CAPI_STAT_API_FLAGS, 6-1
CAPI_STAT_API_HANDLE, 6-1
CAPI_STAT_API_HANDLE_BAD, 6-1
CAPI_STAT_API_HANDLE_NOTNULL, 6-2
CAPI_STAT_API_HANDLE_NULL, 6-2
CAPI_STAT_API_NULL, 6-2
CAPI_STAT_API_SESSION, 6-2
CAPI_STAT_API_SESSION_BAD, 6-2
CAPI_STAT_API_SESSION_NOTNULL, 6-2
CAPI_STAT_API_SESSION_NULL, 6-2
CAPI_STAT_API_STREAM, 6-2
CAPI_STAT_API_STREAM_BAD, 6-2
CAPI_STAT_API_STREAM_NOTNULL, 6-2
CAPI_STAT_API_STREAM_NULL, 6-2
CAPI_STAT_DATA, 6-2
CAPI_STAT_DATA_COOKIE, 6-3
CAPI_STAT_DATA_DATE, 6-3
CAPI_STAT_DATA_DATE_FORMAT, 6-3
CAPI_STAT_DATA_DATE_INVALID, 6-3
CAPI_STAT_DATA_DATE_OUTOFRANGE, 6-3
CAPI_STAT_DATA_DATE_RANGE, 6-3
CAPI_STAT_DATA_EMAIL, 6-3
CAPI_STAT_DATA_EMAIL_NOTSET, 6-3
CAPI_STAT_DATA_ENCODING, 6-3
CAPI_STAT_DATA_HOSTNAME, 6-3
CAPI_STAT_DATA_HOSTNAME_FORMAT, 6-3
CAPI_STAT_DATA_HOSTNAME_HOST, 6-3
CAPI_STAT_DATA_HOSTNAME_SERVER, 6-4
CAPI_STAT_DATA_ICAL, 6-4
CAPI_STAT_DATA_ICAL_COMPEXTRA, 6-4
CAPI_STAT_DATA_ICAL_COMPMISSING, 6-4
CAPI_STAT_DATA_ICAL_COMPNAME, 6-4
CAPI_STAT_DATA_ICAL_COMPVALUE, 6-4
CAPI_STAT_DATA_ICAL_FOLDING, 6-4
CAPI_STAT_DATA_ICAL_IMPLEMENT, 6-4
CAPI_STAT_DATA_ICAL_LINEOVERFLOW, 6-4
CAPI_STAT_DATA_ICAL_NONE, 6-4
CAPI_STAT_DATA_ICAL_OVERFLOW, 6-4
CAPI_STAT_DATA_ICAL_PARAMEXTRA, 6-5
CAPI_STAT_DATA_ICAL_PARAMMISSING, 6-5
CAPI_STAT_DATA_ICAL_PARAMNAME, 6-5
CAPI_STAT_DATA_ICAL_PARAMVALUE, 6-5
CAPI_STAT_DATA_ICAL_PROPEXTRA, 6-5
CAPI_STAT_DATA_ICAL_PROPMISSING, 6-5
CAPI_STAT_DATA_ICAL_PROPNAME, 6-5

CAPI_STAT_DATA_ICAL_PROPVALUE, 6-5
CAPI_STAT_DATA_ICAL_RECURMODE, 6-5
CAPI_STAT_DATA_MIME, 6-5
CAPI_STAT_DATA_MIME_CHARSET, 6-5
CAPI_STAT_DATA_MIME_COMMENT, 6-6
CAPI_STAT_DATA_MIME_ENCODING, 6-6
CAPI_STAT_DATA_MIME_HEADER, 6-6
CAPI_STAT_DATA_MIME_IMPLEMENT, 6-6
CAPI_STAT_DATA_MIME_IMPLEMENT_
NESTING, 6-6
CAPI_STAT_DATA_MIME_LENGTH, 6-6
CAPI_STAT_DATA_MIME_NOICAL, 6-6
CAPI_STAT_DATA_MIME_NONE, 6-6
CAPI_STAT_DATA_MIME_OVERFLOW, 6-6
CAPI_STAT_DATA_UID, 6-6
CAPI_STAT_DATA_UID_FORMAT, 6-6
CAPI_STAT_DATA_UID_NOTFOUND, 6-6
CAPI_STAT_DATA_UID_RECURRENCE, 6-7
CAPI_STAT_DATA_USERID, 6-7
CAPI_STAT_DATA_USERID_EXT, 6-7
CAPI_STAT_DATA_USERID_EXT_
CONFLICT, 6-7
CAPI_STAT_DATA_USERID_EXT_FORMAT, 6-7
CAPI_STAT_DATA_USERID_EXT_INIFILE, 6-7
CAPI_STAT_DATA_USERID_EXT_MANY, 6-7
CAPI_STAT_DATA_USERID_EXT_NODE, 6-7
CAPI_STAT_DATA_USERID_EXT_NONE, 6-7
CAPI_STAT_DATA_USERID_FORMAT, 6-7
CAPI_STAT_DATA_USERID_ID, 6-7
CAPI_STAT_LIBRARY, 6-7
CAPI_STAT_LIBRARY_IMPLEMENTATION, 6-8
CAPI_STAT_LIBRARY_INTERNAL, 6-8
CAPI_STAT_LIBRARY_INTERNAL_
COSMICRAY, 6-8
CAPI_STAT_LIBRARY_INTERNAL_DATA, 6-8
CAPI_STAT_LIBRARY_INTERNAL_EXPIRY, 6-8
CAPI_STAT_LIBRARY_INTERNAL_
FUNCTION, 6-8
CAPI_STAT_LIBRARY_INTERNAL_
OVERFLOW, 6-8
CAPI_STAT_LIBRARY_INTERNAL_
PROTOCOL, 6-8
CAPI_STAT_LIBRARY_INTERNAL_UNKNOWN_
EXCEPTION, 6-8
CAPI_STAT_LIBRARY_INTERNAL_UNKNOWN_
LIBRARY_ERRCODE, 6-8
CAPI_STAT_LIBRARY_SERVER, 6-8
CAPI_STAT_LIBRARY_SERVER_BUSY, 6-8
CAPI_STAT_LIBRARY_SERVER_SUPPORT, 6-9
CAPI_STAT_LIBRARY_SERVER_SUPPORT_
CHARSET, 6-9
CAPI_STAT_LIBRARY_SERVER_SUPPORT_
STANDARDS, 6-9
CAPI_STAT_LIBRARY_SERVER_USERDATA, 6-9
CAPI_STAT_OK, 6-9
CAPI_STAT_SECUR, 6-9
CAPI_STAT_SECUR_LOGON, 6-9
CAPI_STAT_SECUR_LOGON_AUTH, 6-9
CAPI_STAT_SECUR_LOGON_LOCKED, 6-9
CAPI_STAT_SECUR_LOGON_LOCKED_
RESOURCE, 6-9
CAPI_STAT_SECUR_LOGON_LOCKED_
SYSOP, 6-9
CAPI_STAT_SECUR_READ, 6-10
CAPI_STAT_SECUR_READ_ALARM, 6-10
CAPI_STAT_SECUR_READ_PROPS, 6-10
CAPI_STAT_SECUR_SERVER, 6-10
CAPI_STAT_SECUR_SERVER_LICENSE, 6-10
CAPI_STAT_SECUR_SERVER_SET_IDENTITY_
SYSOP, 6-10
CAPI_STAT_SECUR_WRITE, 6-10
CAPI_STAT_SECUR_WRITE_AGENDA, 6-10
CAPI_STAT_SECUR_WRITE_EVENT, 6-10
CAPI_STAT_SERVICE, 6-10
CAPI_STAT_SERVICE_ACE, 6-10
CAPI_STAT_SERVICE_ACE_LOAD, 6-10
CAPI_STAT_SERVICE_ACE_SUPPORT, 6-11
CAPI_STAT_SERVICE_FILE, 6-11
CAPI_STAT_SERVICE_FILE_CLOSE, 6-11
CAPI_STAT_SERVICE_FILE_DELETE, 6-11
CAPI_STAT_SERVICE_FILE_MODE, 6-11
CAPI_STAT_SERVICE_FILE_OPEN, 6-11
CAPI_STAT_SERVICE_FILE_READ, 6-11
CAPI_STAT_SERVICE_FILE_TEMP, 6-11
CAPI_STAT_SERVICE_FILE_WRITE, 6-11
CAPI_STAT_SERVICE_LIBRARY, 6-11
CAPI_STAT_SERVICE_MEM, 6-11
CAPI_STAT_SERVICE_MEM_ALLOC, 6-11
CAPI_STAT_SERVICE_NET, 6-12
CAPI_STAT_SERVICE_NET_TIMEOUT, 6-12

CAPI_STAT_SERVICE_THREAD, 6-12
CAPI_STAT_SERVICE_TIME, 6-12
CAPI_STAT_SERVICE_TIME_GMTIME, 6-12
character sets, 2-1
client_name, 4-2
client_version, 4-3
cncachesize, 4-3
Create method, 9-15
CSDK, 5-2
CSDK_AddConditionToQuery, 3-52
CSDK_Authenticate, 3-5
CSDK_ConfigureACE, 3-8
CSDK_Connect, 3-9
CSDK_ConnectAsSysop, 3-10
CSDK_CreateCallbackStream, 3-19
CSDK_CreateFileStream, 3-67
CSDK_CreateFileStreamFromFilenames, 3-21
CSDK_CreateMemoryStream, 3-23
CSDK_CreateQuery, 3-53
CSDK_CreateSession, 3-4
CSDK_Deauthenticate, 3-11
CSDK_DeleteContacts, 3-26
CSDK_DeleteEvents, 3-33
CSDK_DeleteTasks, 3-46
CSDK_DestroyHandle, 3-13
CSDK_DestroyMultipleHandles, 3-14
CSDK_DestroyMultipleStreams, 3-25
CSDK_DestroyQuery, 3-54
CSDK_DestroyResult, 3-54
CSDK_DestroySession, 3-4
CSDK_DestroyStream, 3-26
CSDK_Disconnect, 3-12
CSDK_FetchContactsByQuery, 3-27
CSDK_FetchContactsByUID, 3-28
CSDK_FetchEventsByAlarmRange, 3-34
CSDK_FetchEventsByRange, 3-36
CSDK_FetchEventsByUID, 3-38
CSDK_FetchTasksByAlarmRange, 3-47
CSDK_FetchTasksByRange, 3-48
CSDK_FetchTasksByUID, 3-49
CSDK_FLAG_FETCH_AGENDA_ATTENDEE_
ONLY, 5-1
CSDK_FLAG_FETCH_COMBINED, 5-1
CSDK_FLAG_FETCH_EXCLUDE_
ACCEPTED, 5-2
CSDK_FLAG_FETCH_EXCLUDE_
APPOINTMENTS, 5-2
CSDK_FLAG_FETCH_EXCLUDE_
DAILYNOTES, 5-2
CSDK_FLAG_FETCH_EXCLUDE_
DAYEVENTS, 5-2
CSDK_FLAG_FETCH_EXCLUDE_DECLINED, 5-2
CSDK_FLAG_FETCH_EXCLUDE_
HOLIDAYS, 5-2
CSDK_FLAG_NONE, 5-1
CSDK_GetCapabilities, 3-5
CSDK_GetFirstFailure, 3-55
CSDK_GetFirstParseError, 3-56
CSDK_GetFirstResult, 3-58
CSDK_GetHandle, 3-15
CSDK_GetHandleInfo, 3-17
CSDK_GetNextFailure, 3-59
CSDK_GetNextParseError, 3-61
CSDK_GetNextResult, 3-63
CSDK_GetStatusLevels, 3-64
CSDK_GetStatusString, 3-65
CSDK_SetConfigFile, 3-3
CSDK_SetIdentity, 3-12
CSDK_StoreContacts, 3-29
CSDK_StoreEvents, 3-40
CSDK_StoreTasks, 3-51

D

data streams (SDK), 2-9
Delete method, 9-17
direntrycachesize, 4-3

E

emailcachesize, 4-4
events
  day, 9-29
  recurring, 9-26
  repeating, 9-23
  simple, 9-23

F

functions

legacy (SDK), 3-66
SDK, 3-3 to 3-65
superceded (SDK), 3-65

H

holidays, 9-30
HTTP headers, 9-1

I

iCalendar, 2-1
itemcachesize, 4-4

J

Java classes, 10-1

L

log_activity, 4-4
log_modulesinclude, 4-5

M

max_caldomain, 4-5
max_masternode, 4-6
max_user, 4-6
Modify method, 9-18

N

NoOp, 9-10

P

Ping, 9-22

S

SDK
accented characters, 7-3
alarms, 2-6
best practises, 1-2
character sets, 2-1
configuration settings, 4-1
contents, 1-1
data streams, 2-9
FAQ, 7-1
functions, 3-3 to 3-65
iCalendar, 2-1
implementation considerations, 2-1
introduction, 1-1
languages, using with, 7-2
previous versions, 7-1
security model, 2-6
troubleshooting, 7-1
user identification, 2-7
search (web services), 9-11
security
SDK, 2-6
web services, 8-2, 9-2
securitycachesize, 4-7
SOAP, 9-1
SOAP faults, 10-4, 11-1

T

tasks, 9-31
time, local (WS), 10-6
tzcachesize, 4-7

U

user identification (SDK), 2-7

V

Visual C (SDK), 7-4
vQuery, 9-12

W

web services
about, 8-1
Create method, 9-15
CWSL, 9-9
daily notes, 9-29
day events, 9-29
delete method, 9-17
faults, 10-4, 11-1

fetching data, 10-2
holidays, 9-30
HTTP headers, 9-1
introduction (toolkit), 8-2
Java classes, 10-1
Modify method, 9-18
NoOp, 9-10
Ping, 9-22
priority mapping, 9-31
recurring events, 9-26
repeating events, 9-23
search, 9-11
security, 9-2
simple events, 9-23
SOAP, 9-1
tasks, 9-31
time, local, 10-6
vQuery, 9-12
xCal, 9-22
X-ORACLE-EVENTTYPE, 9-31

X

xCal, 9-22
X-ORACLE-EVENTTYPE, 9-31