

Oracle® Communications Calendar Server

System Administrator's Guide

Release 7.0.5

E54935-01

February 2015

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	xi
Audience	xi
Related Documents	xi
Documentation Accessibility	xi
Part I Monitoring and Managing Calendar Server	
1 Calendar Server System Administration Overview	
About Calendar Server	1-1
Overview of Calendar Server Administration Tasks	1-1
About Calendar Server Administration Tools	1-2
Directory Placeholders Used in This Guide	1-2
2 Stopping and Starting Calendar Server Services	
Stopping and Starting Calendar Server	2-1
Starting and Stopping the Remote Document Store	2-1
3 Monitoring and Managing Calendar Server	
Administering GlassFish Server	3-1
Administering the Document Store	3-1
Changing the Password Used for Remote Document Store Authentication	3-1
Using Calendar Server Administration Utilities	3-2
Managing Logging	3-2
Overview of Calendar Server Logging	3-2
Logging Calendar Server Information to the GlassFish Server Log File	3-3
Configuring Logging	3-3
Viewing the Document Store Logs	3-4
Using the scheduling Log	3-4
Using the commands Log	3-5
Administering Calendar Server Access	3-7
Overview of ACLs	3-7
Calendar Access Controls	3-8
Scheduling Access Controls	3-8
Setting Access Control for LDAP Groups	3-9

Retrieving Access Control Information	3-9
Access Control Configuration Parameters	3-9
Command-Line Utilities for Access Control	3-10
WCAP Commands for Access Control	3-10
Managing Domain ACLs	3-10
Managing Dynamic Group ACLs.....	3-10
Administering Scheduling Options	3-10
Configuring Scheduling Options.....	3-11
Overview of Calendar Booking Window	3-11
Configuring a Booking Window	3-12
Modifying Calendar Double Booking.....	3-13
Controlling Double Booking When Creating Accounts Automatically.....	3-13
Modifying Configuration Parameters That Control Double Booking	3-14
Overriding the Account Autocreation Through LDAP.....	3-14
Manually Creating Accounts.....	3-15
Modifying Double Booking on Existing Accounts.....	3-15
Inviting LDAP Groups	3-15
Administering Resource Calendars	3-15
About Resource Calendars	3-16
Provisioning Resource Calendars (commadmin).....	3-16
Provisioning Resource Calendars (Delegated Administrator Console).....	3-17
Managing a Resource Calendar's Mailbox	3-17
Administering Time Zones Support	3-18
Adding New WCAP Time Zones	3-18
Adding an Alias to an Existing Time Zone	3-18
Adding a New Time Zone	3-19
Customizing Calendar Notifications.....	3-19
Administering the Calendar Server Back End Databases.....	3-19
Administering the MySQL Database	3-19
Administering the Oracle Database	3-20
Backing Up and Restoring Calendar Server Data.....	3-20
Removing Unwanted Calendar Data to Reclaim Space	3-20
Purging Deleted Calendar Entries.....	3-20
Purging Messages From the Scheduling Inbox and Outbox	3-20

4 Setting Up and Managing Calendar Server Users

Provisioning Calendar Server Users.....	4-1
Overview of Provisioning Calendar Server	4-1
Provisioning Calendar Users by Using Delegated Administrator	4-2
Provisioning Calendar Users Across Virtual Domains	4-3
Managing Calendar Users and Accounts.....	4-3
Defining Valid Calendar Users	4-4
Enabling and Disabling Automatic Account Creation	4-4
Creating Calendar Account with Default Calendar Automatically Upon Login.....	4-4
Preventing a User or Resource From Accessing Calendar Server	4-5
Checking for Active Calendar Users.....	4-5
Removing Calendar Users	4-5

Removing a Calendar User (Example).....	4-6
Moving Calendar Users to a New Back-End Database	4-7
Changing a User's Email Address in the Calendar Server Database	4-8
Subscribing and Unsubscribing Calendars	4-8
About Configuring External Authentication	4-9
Configuring Calendar Server for External Authentication.....	4-9
Example: External Authentication by Using cn.....	4-10
Configuring Proxy Authentication	4-11

5 Enabling Advanced Features

Enabling Attachments	5-1
Enabling Apple iCal Private/Confidential Support	5-1
Enabling SMS Calendar Notifications in Convergence	5-2
Enabling the iSchedule Channel to Handle iMIP Messages.....	5-2
Enabling CalDAV and CardDAV Autodiscovery	5-2

6 Configuring CalDAV Clients

Prerequisites	6-1
Configuring CalDAV Clients	6-1
Configuring Apple iCal for Calendar Server	6-2
Configuring Apple iPhone for Calendar Server.....	6-2
Configuring Lightning 1.0 beta2 for Calendar Server	6-2
Configuring Lightning 1.0 beta for Calendar Server	6-3
Configuring Lightning 0.9 for Calendar Server.....	6-4
Accessing a Shared Calendar	6-5
Configuring a CalDAV Account by Using Non-standard or Demo Settings	6-6
iOS 3.x and 4.x Non-standard Configuration	6-6
Apple iCal Non-standard Configuration	6-7
Configuring Android for Calendar Server	6-7
Using the iPhone Configuration Utility	6-7
Exporting and Importing Calendars in Thunderbird Lightning	6-7
Exporting a Calendar.....	6-7
Importing a Calendar	6-8
Client Issues	6-8
Troubleshooting CalDAV Clients.....	6-8
Connector for Microsoft Outlook and Event Time Modifications	6-8

7 Configuring and Managing Virus Scanning

About Calendar Server and Virus Scanning.....	7-1
Calendar Server Virus Scanning Architecture	7-1
Configuring Calendar Server Virus Scanning	7-3
Configuring the MTA	7-3
Installing a Standalone Message Transfer Agent.....	7-4
Configuring the Messaging Server MTA for the Virus Spam Filter.....	7-4
Creating an Incoming SMTP Channel That Uses the Filter	7-4
Configuring the Rewrite Rule to Detect Calendar Data and Discard it After Scanning ..	7-4

Configuring Calendar Server for Virus Scanning	7-5
Example MTA Configuration for Calendar Server Virus Scanning	7-5
Calendar Server Configuration Examples	7-8
Calendar Server Virus Scan Command-line Utility	7-9
Virus Scan Logging	7-9
MTA Logging	7-9

8 Using Calendar Server Notifications

Overview of Notification Architecture	8-1
About Reminders (Alarms)	8-3
About Server Email Notifications	8-3
Enabling Calendar Server Notifications	8-3
Enabling Notifications on an Account	8-4
Modifying Notifications on an Account	8-5
Managing Notification Templates	8-5
Notification Types	8-5
Templates, Resource Bundle, and Other Configuration Files	8-7
Notification Configuration	8-7
Resource Bundles	8-7
Template Files	8-7
Customizing Templates	8-10
Preserving Customized Template Files During Calendar Server Upgrade	8-13
Writing a Java Messaging Service Consumer	8-13
Notification Message Format	8-13
Code Sample	8-14
Managing Calendar Server Java Messaging Server Destinations	8-15
Overview of Calendar Server JMS Destinations	8-15
Using the imqcmd Command with Calendar Server	8-15
Listing a JMS Destination's Metrics	8-16
Purging All Messages	8-16
Monitoring Disk Utilization	8-16
Accessing Remote Brokers Tip	8-16
Presence Notifications	8-16
Configuring Presence Notifications	8-16

9 Troubleshooting Calendar Server

Troubleshooting Calendar Server Initial Configuration	9-1
Troubleshooting GlassFish Server and Java	9-1
Troubleshooting Tips	9-1
Using the asadmin Command to Specify GlassFish Server Port	9-2
Using the GlassFish Server Administration Console to Check Calendar Server Status	9-2
Using the asadmin Command-line Utility to Check Calendar Server Status	9-2
Troubleshooting the Calendar Server davserver Process	9-2
Troubleshooting a Failing davadmin Command	9-2
Troubleshooting MySQL Server Errors	9-3
Importing a Convergence ics File	9-4
Refreshing Domain Information	9-4

Troubleshooting the iSchedule Back End on MySQL Server	9-4
Enabling Telemetry Logging	9-5
Common Errors in Log Files.....	9-5
Using the Same Start and End Date for an Event.....	9-5
Same UID Already in Use	9-5
No Specification of Content-type Header	9-5
Deleting a Non-existing File	9-6
Posting to Calendar Collection Without a File Name.....	9-6
Using a Non-implemented HTTP Method.....	9-6
Using the Browser Servlet	9-6
Troubleshooting CalDAV Clients	9-6
Lightning	9-7
Apple iCal.....	9-7
iPod touch	9-7
Known Issues.....	9-8
Troubleshooting Clients Running iOS 5 and Mac OS 10.7	9-8
Mac OS 10.9 iCal Client Not Able to Delete Events	9-8
Checking Active Calendar Users	9-8
Troubleshooting Calendar Server Agent Alerts in Instant Messaging Server	9-9

10 Improving Calendar Server Performance

Tuning Calendar Server Logging.....	10-1
Tuning Oracle GlassFish Server.....	10-1
Tuning JVM Options.....	10-1
Tuning JDBC Pool	10-2
Tuning HTTP Service and Listener	10-2
Tuning MySQL Server	10-2
Tuning Oracle Solaris CMT Server	10-3
Tuning Reference	10-4

11 Backing Up and Restoring Calendar Server Files and Data

Overview of Calendar Server Backup and Restore	11-1
Calendar Server Backup and Restore Techniques.....	11-2
Using the davadmin db backup Command.....	11-2
ZFS Snapshots.....	11-2
MySQL Backup and Restore Techniques.....	11-2
MySQL Asynchronous Replication	11-2
MySQL Database Dump	11-3
Point-In-Time Binlog Backup and Recovery	11-3
Oracle Database Backup and Restore Techniques	11-3

Part II Administering a High-Availability System

12 Configuring a High-Availability Database

Overview of MySQL Server Asynchronous Replication	12-1
MySQL Server Asynchronous Replication Example	12-2

MySQL Server Two-Way Replication Example	12-2
Replication Synchronization Issues	12-3
Using the Multi-Host Failover Feature of JDBC Connector/J	12-3
Test for MySQL Server Asynchronous Replication (Manual)	12-4
Test for MySQL Server Two-Way Replication with Connector/J Failover.....	12-7

13 Configuring Calendar Server for Highly Availability

Front End High Availability: Configuring Calendar Server in GlassFish HA Cluster	13-1
About This Deployment Example	13-1
Prerequisites and Deployment Example Architecture	13-1
Configuring a Calendar Server Cluster Deployment	13-2
Configuring Machine A Where DAS, Node Agent, and HADB Reside	13-2
Configuring Machine B Where Node Agent Resides	13-4
Configuring Machine C Where Web Server 7 Is Used as the Load Balancer.....	13-4
GlassFish Server Load Balancer Limitations	13-5
Back End High Availability: MySQL Async Replication	13-5
Back End High Availability: Oracle Data Guard	13-6
Document Store High Availability	13-6

14 Configuring Calendar Server on a GlassFish Server Cluster

Prerequisites for Deploying Calendar Server on a GlassFish Cluster.....	14-1
Example GlassFish Server Cluster Deployment Architecture	14-2
Configuring a Calendar Server and GlassFish Server Cluster Deployment.....	14-2
To Create a Two-Instance GlassFish Server Cluster	14-3
Deploying Calendar Server on a Two-Instance GlassFish Server Cluster.....	14-3
Limitations of This Deployment.....	14-5

Part III Calendar Server Reference

15 Calendar Server Configuration Reference

davserver.properties File.....	15-1
davservercreds.properties File.....	15-1
Document Store Server Configuration File.....	15-2
certmap.conf File	15-2
davadmin.properties File	15-2
Notification Templates.....	15-3

16 Calendar Server Command-Line Utilities

Overview of the Command-Line Utilities	16-1
davadmin Security	16-2
Environment Variables.....	16-2
davadmin Utility	16-2
Location	16-2
General Syntax.....	16-2
Ways to Provide Options	16-3
Clifile Properties.....	16-3

Common Options	16-4
davadmin Operations.....	16-5
Tool-Only Options	16-6
Exit Code	16-6
davadmin account	16-6
Syntax.....	16-6
account Operation.....	16-6
Options for account Operation.....	16-7
davadmin account Examples	16-9
davadmin backend	16-10
Syntax.....	16-10
backend Operation.....	16-11
Options for backend Operation	16-11
davadmin backend Examples	16-11
davadmin cache	16-12
Syntax.....	16-12
cache Operation.....	16-12
Options for the cache Operation.....	16-12
davadmin calendar	16-13
Syntax.....	16-13
calendar Operation	16-13
Options for calendar Operation	16-13
davadmin calendar Examples	16-14
davadmin calcomponent	16-15
Syntax.....	16-15
calcomponent Operation.....	16-15
Options for calcomponent Operation	16-15
davadmin calcomponent Examples	16-16
davadmin config	16-16
Syntax.....	16-16
config Operation.....	16-17
Options for config Operation	16-17
davadmin config Examples	16-17
davadmin db	16-18
Syntax.....	16-18
db Operation.....	16-18
Options for db Operation.....	16-19
davadmin db Examples.....	16-21
davadmin ldappool	16-22
Syntax.....	16-22
ldappool Operations	16-23
Options for ldappool Operation	16-23
davadmin ldappool Examples	16-23
davadmin migration	16-24
Syntax.....	16-24
migration Operation	16-24
Options for migration Operation.....	16-24

davadmin migration Examples.....	16-26
davadmin passfile	16-26
Syntax.....	16-26
passfile Operation	16-27
Options for passfile Operation.....	16-27
davadmin passfile Examples	16-27
davadmin vscan	16-28
Syntax.....	16-28
vscan Operation.....	16-28
Options for vscan Operation	16-29
davadmin vscan Examples	16-29
JConsole	16-30
AdminAccountMXBean Operation	16-30
AdminBackendMXBean Operation.....	16-30
AdminCalComponentMXBean Operation.....	16-30
AdminCalendarMXBean Operation.....	16-31
AdminConfigMBean Operation.....	16-31
AdminMigrationMXBean Operation	16-31
AdminMiscMXBean Operation.....	16-31
AdminUtilMXBean	16-31
Starting GlassFish Server in Secure Mode.....	16-31
Summary of davadmin Changes by Release	16-31
Changes in Calendar Server 7 Update 1	16-31
Changes in Calendar Server 7 Update 2	16-32
Changes in Calendar Server 7 Update 2 Patch 5	16-32
Changes in Calendar Server 7 Update 3	16-32
Changes in Calendar Server 7.0.4.14.0	16-32
Changes in Calendar Server 7.0.4.16.0	16-33
Changes in Calendar Server 7.0.5.17.0	16-33
Deprecated Options	16-33

17 Calendar Server Configuration Parameters

18 Time Zone Database

Africa	18-1
America	18-2
Antarctica	18-5
Arctic	18-5
Asia	18-5
Atlantic	18-6
Australia	18-6
Europe	18-7
Indian	18-8
Pacific	18-8

Preface

This guide explains how to administer Oracle Communications Calendar Server (also referred to as Calendar Server 7) and its accompanying software components.

Audience

This document is intended for system administrators whose responsibility includes Calendar Server. This guide assumes you are familiar with the following topics:

- Oracle Communications Calendar Server and Oracle Communications Messaging Server protocols
- Oracle GlassFish Server
- Oracle Directory Server Enterprise Edition and LDAP
- System administration and networking
- General deployment architectures

Related Documents

For more information, see the following documents in the Calendar Server documentation set:

- *Calendar Server Concepts*: Provides an overview of Calendar Server.
- *Calendar Server Installation and Configuration Guide*: Provides instructions for installing and configuring Calendar Server.
- *Calendar Server Release Notes*: Describes the new features, fixes, known issues, troubleshooting tips, and required third-party products and licensing.
- *Calendar Server Security Guide*: Provides guidelines and recommendations for setting up Calendar Server in a secure configuration.
- *Calendar Server WCAP Developer's Guide*: Describes how to use the Web Calendar Access Protocol 7.0 (WCAPbis) with Calendar Server.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Part I

Monitoring and Managing Calendar Server

Part I contains the following chapters:

- [Calendar Server System Administration Overview](#)
- [Stopping and Starting Calendar Server Services](#)
- [Monitoring and Managing Calendar Server](#)
- [Setting Up and Managing Calendar Server Users](#)
- [Enabling Advanced Features](#)
- [Configuring CalDAV Clients](#)
- [Configuring and Managing Virus Scanning](#)
- [Using Calendar Server Notifications](#)
- [Troubleshooting Calendar Server](#)
- [Improving Calendar Server Performance](#)
- [Backing Up and Restoring Calendar Server Files and Data](#)

Calendar Server System Administration Overview

This chapter provides an overview of Oracle Communications Calendar Server, and describes the basic administration tasks and tools used to perform those tasks.

About Calendar Server

Oracle Communications Calendar Server (also referred to as Calendar Server 7 and formerly known as Oracle Communications Calendar Server for CALDAV Clients and Sun Java System Calendar Server) is a carrier-grade, highly scalable, secure, and reliable calendaring and scheduling platform. Calendar Server is compliant with the latest calendaring and scheduling standards, including the CalDAV access protocol, which makes it usable with Apple iCal, iPhone, Thunderbird Lightning, and any other CalDAV client.

Calendar Server provides a number of calendaring and scheduling capabilities, including:

- Personal appointments (one-time/recurring) and reminders
- Multiple calendars per user (Work calendar, Home calendar)
- Document store for storing event/task attachments
- Multiple access points, including desktop clients (Apple iCal, Outlook, Lightning), Convergence web client, and mobile clients (iPhone, Android)
- Availability checks
- Invitation notifications
- Special handling of resource scheduling
- Comprehensive access control settings including settings for groups
- Sharing and subscription of calendars

Overview of Calendar Server Administration Tasks

A Calendar Server administrator is responsible for the day-to-day tasks of maintaining and managing Calendar Server and its users. The tasks also include managing Calendar Server components, GlassFish Server, and potentially other Unified Communications Suite components.

You perform the following tasks as a Calendar Server administrator:

- Stopping and starting Calendar Server

- Managing calendar accounts
- Monitoring Calendar Server
- Tuning Calendar Server performance
- Migrating data to Calendar Server
- Managing the Calendar Server back-end database
- Backing up and restoring files
- Troubleshooting Calendar Server

About Calendar Server Administration Tools

Calendar Server is deployed on an Oracle GlassFish Server domain. You use the GlassFish Server Administration Console and **asadmin** command to manage the Calendar Server web container. See the GlassFish Server documentation for more information.

Calendar Server provides a number of command-line utilities for administering the server. These utilities run under the umbrella command, **davadmin**. For more information, see "[Calendar Server Command-Line Utilities](#)."

Directory Placeholders Used in This Guide

[Table 1–1](#) lists the placeholders that are used in this guide:

Table 1–1 *Calendar Server Directory Placeholders*

Placeholder	Directory
<i>CalendarServer_home</i>	Specifies the installation location for the Calendar Server software. The default is /opt/sun/comms/davserver .
<i>DelegatedAdmin_home</i>	Specifies the installation location for the Delegated Administrator software. The default is /opt/sun/comms/da .
<i>GlassFish_home</i>	Specifies the installation location for the Oracle GlassFish Server software. The default is /opt/glassfish3/glassfish .

Stopping and Starting Calendar Server Services

This chapter describes how to stop and start Oracle Communications Calendar Server services.

Stopping and Starting Calendar Server

To stop and start the Calendar Server process, you use the **asadmin stop-domain** and **asadmin start-domain** commands to stop and start the GlassFish Server container in which Calendar Server is deployed. The following examples assume a default GlassFish Server installation with Calendar Server deployed in **domain1**.

- To stop Calendar Server:

```
GlassFish_home/bin/asadmin stop-domain domain1
```

- To start Calendar Server:

```
GlassFish_home/bin/asadmin start-domain domain1
```

Starting and Stopping the Remote Document Store

The Calendar Server document store is used to store and retrieve *large data*, such as todo attachments.

To stop and start the Calendar Server remote document store server, use the **stop-as** and **start-as** commands.

- To start the remote document store server:

```
CalendarServer_home/sbin/start-as
```

- To stop the remote document store server:

```
CalendarServer_home/sbin/stop-as
```

Monitoring and Managing Calendar Server

This chapter provides details on monitoring and managing Oracle Communications Calendar Server.

Administering GlassFish Server

Calendar Server depends on Oracle GlassFish Server as a web container. See the Oracle GlassFish Server 3 documentation for details on administering GlassFish Server.

The following Glassfish Server documentation will help you get started:

- "Certificates and SSL" in *Glassfish Server Security Guide*
- "asadmin Utility" in *Glassfish Server Administration Guide*
- "High Availability in GlassFish Server" in *Glassfish Server High Availability Administration Guide*

Administering the Document Store

The Calendar Server document store is used to store and retrieve "large data," such as calendar events with many invitees, and todos with large attachments. Normally, you configure the document store as part of the Calendar Server installation process. You set up one document store per configured Calendar Server back end. You do so by specifying the location of the document store directory for Calendar Server to use in the `store.dav.defaultbackend.dbdir` configuration parameter, if the store is local, and the `store.dav.backend_name.attachstorehost` configuration parameter, if the store is remote. For more information, see the topic on configuring the document store in *Calendar Server Installation and Configuration Guide*.

Administering the document store involves:

- [Starting and Stopping the Remote Document Store](#)
- [Changing the Password Used for Remote Document Store Authentication](#)

Changing the Password Used for Remote Document Store Authentication

When changing passwords used for remote document store authentication, you must change them on both the local Calendar Server host and on each remote document store host to keep them synchronized.

To change the remote document store password:

1. Use the following **davadmin** command to change the password on each remote document store host.

```
cd CalendarServer_home/sbin
./davadmin passfile modify -0
```

Respond to the prompts.

2. Stop then restart the document store server for the password change to take effect.

```
cd CalendarServer_home/sbin
./stop-as
./start-as
```

3. Use the following **davadmin** command to change the password on the local Calendar Server host.

```
cd CalendarServer_home/sbin
./davadmin passfile modify
```

Respond to the prompts.

Note: When you run the **davadmin db backup** command, you are prompted for the document store password. To avoid having to enter a password every time when running this command, create a password file by running the **davadmin passfile** command.

Using Calendar Server Administration Utilities

Calendar Server provides a number of command-line utilities for server administration. These utilities run under the umbrella command, **davadmin**, which is a simple shell script. By default, the **davadmin** utility is installed in the *CalendarServer_home/sbin* directory with user or group **bin/bin** permissions. See "[Calendar Server Command-Line Utilities](#)" for more information.

Managing Logging

Managing logging includes:

- [Logging Calendar Server Information to the GlassFish Server Log File](#)
- [Configuring Logging](#)
- [Viewing the Document Store Logs](#)
- [Using the scheduling Log](#)
- [Using the commands Log](#)

Overview of Calendar Server Logging

Calendar Server 7 maintains the following types of log files:

- **commands:** Stores information about requests that are sent to the server and information related to each operation performed to satisfy those requests. The **commands** log contains servlet and core operation classes entries that are designed to help you monitor requests to the server and help diagnose problems. See "[Using the commands Log](#)" for more information on the **commands** log.

- **errors**: Stores error and debug-level information that is supplied by the server for use in diagnosing problems.
- **scheduling**: Stores information on scheduling actions, showing when invitations are enqueued and dequeued.
- **telemetry**: Stores entire Calendar Server servlet request and response transcripts.
- **scan**: Stores information on virus scanning actions.

Each log file has its own configuration parameters that control the log file location, maximum size, log level, and number of files allowed.

Log files are created with a suffix of *.number*, for example, **commands.0**, **commands.1**, and so on. The log file numbered **.0** is the newest, the log file numbered **.1** is next newest, and so on. When a log file is filled to its maximum configured size, the logging system increments each of the existing log file suffixes to the next higher number, starting with the highest. If the number of log files reaches the configured maximum, the highest numbered log file is deleted and the next higher takes its place.

For example, Calendar Server is started for the first time and you have configured the maximum number of log files at 10. The logging system begins writing messages to the log file with the **.0** suffix. When the **.0** log file is filled to capacity, the logging system increments its suffix to the next higher number and the file becomes **.1**. The logging system then creates a new **.0** log file and begins writing messages to it. When the **.0** file become full, the logging system increments the **.1** file to **.2**, increments the **.0** file to **.1**, and creates a new **.0** file. This process continues until the maximum number of configured log files is reached. When that happens, the logging system deletes the highest numbered (oldest) log file, **.9**, increments each of the lower numbered files' suffixes, and creates a new **.0** log file.

The Calendar Server log files are kept separate from the GlassFish Server log files. The GlassFish Server log files are maintained in the *GlassFish_home/domains/domain_name/logs* directory, for example, */opt/glassfish3/glassfish/domains/domain1/logs*. Even though the container's log file is the root log file, by default, information that is logged to the Calendar Server's log files is not logged to the container's log file.

Logging Calendar Server Information to the GlassFish Server Log File

The Calendar Server **logToParent** flag is to **false** by default, preventing logging of information to the GlassFish Server log file.

To log calendar information to both the GlassFish Server log file (**server.log**) and the Calendar Server log file (**commands.0**), set the **log.dav.commands.logtoparent** parameter to **true**:

```
./davadmin config modify -u admin -o log.dav.commands.logtoparent -v true
```

Configuring Logging

Use the **davadmin** command to configure Calendar Server logging configuration parameters as shown in [Table 3-1](#).

name can be **commands**, **errors**, **scheduling**, **telemetry**, or **scan**, depending on the type of logging you want to configure; use **error** to configure Calendar Server error logging. **SEVERE** and **WARNING** messages need immediate attention. **FINE**, **FINER**, and **FINEST** messages are usually informational only, but can provide more context for troubleshooting when accompanying **SEVERE** and **WARNING** messages.

For more information about the logging configuration parameters and their default values, see "[Calendar Server Configuration Parameters](#)."

Table 3–1 Calendar Server Log File Parameters

Parameter	Description
<code>log.dav.name.logdir</code>	Specifies the log file directory path
<code>log.dav.name.loglevel</code>	Specifies the log level: <ul style="list-style-type: none"> ▪ OFF: No information is logged. ▪ SEVERE: Logs catastrophic errors. ▪ WARNING: Logs major errors or exceptions with the system. ▪ INFO: Logs general informational messages. This is the default level. ▪ FINE: Logs general debugging and tracing information to show the higher level flow through the code or more detailed information about a problem. ▪ FINER: Logs more details than FINE. ▪ FINEST or ALL: Logs the finest grain details about code flow or problem information. Enabling this level can result in massive amounts of data in the log file, making it hard to parse.
<code>log.dav.name.logtoparent</code>	Enables or disables logging to the GlassFish Server log file. When set to true , messages are logged to both the GlassFish Server log file and the Calendar Server log file. Set to false to disable logging to the GlassFish Server log file.
<code>log.dav.name.maxlogfiles</code>	Specifies the maximum number of log files
<code>log.dav.name.maxlogfilesize</code>	Specifies the log file's maximum size

Viewing the Document Store Logs

The document store logs are named `astore.number` and are located in the `CalendarServer_home/logs` directory. Change to this directory to view the log files.

Using the scheduling Log

The `scheduling` log file stores information on scheduling actions, showing when invitations are enqueued and dequeued. [Table 3–2](#) describes the scheduling codes in the `scheduling` log file.

Table 3–2 Codes Used in Scheduling Log Files

Code	Log Level Needed	Description
E	INFO	Enqueuing of an inbound scheduling message
J	INFO	Rejection of attempted enqueue
DL	FINE	Successful dequeue for a local recipient
DE	FINE	Successful dequeue for an external (iSchedule) recipient
DM	FINE	Successful dequeue for an iMIP recipient
QE	INFO	Temporary failure to dequeue for an external (iSchedule) recipient
QM	INFO	Temporary failure to dequeue for an iMIP recipient
R	INFO	Permanent failure to dequeue

By default, enqueues are logged, as well as unsuccessful dequeues, such as wrong user, temporary errors, and so on. To see successful dequeues in the log, you must set the **scheduling** log level to at least **FINE**.

The following log sample shows sample dequeues and enqueues.

```
[2012-06-01T16:26:56.018+0200] E
"a11bdb82-422a11dd-8002d2ed-c263972e@example.com/calendar-outbox/REQUEST-133856081
6008-3-.ics" 6954475.scen REQUEST mailto:james.smith@example.com
mailto:ron.white@example.com "1.2;Delivered"
[2012-06-01T16:26:56.019+0200] E
"a11bdb82-422a11dd-8002d2ed-c263972e@example.com/calendar-outbox/REQUEST-133856081
6008-3-.ics" 6954475.scen REQUEST mailto:james.smith@example.com
mailto:mary.jones@example.com "1.2;Delivered"
[2012-06-01T16:26:56.083+0200] DL
"a11bdb82-422a11dd-8002d2ed-c263972e@example.com/calendar-outbox/REQUEST-133856081
6008-3-.ics" 6954475.scen REQUEST mailto:james.smith@example.com
mailto:ron.white@example.com "Success"
[2012-06-01T16:26:56.239+0200] DL
"a11bdb82-422a11dd-8002d2ed-c263972e@example.com/calendar-outbox/REQUEST-133856081
6008-3-.ics" 6954475.scen REQUEST mailto:james.smith@example.com
mailto:mary.jones@example.com "Success"
```

invitation from james to ron and mary with UID "6954475.scen" was submitted (E) at "2012-06-01T16:26:56.018+0200" and delivered at 2012-06-01T16:26:56.083 to ron and at 2012-06-01T16:26:56.239 to mary

This example shows the following information:

1. Timestamp
2. Scheduling codes (E,DL)
3. Relative URI of scheduling message being processed
4. iCalendar UID of the event/tasks
5. Type of message (iTIP REQUEST, REPLY)
6. Originator
7. Recipient
8. iTIP detailed status code

Using the commands Log

The Calendar Server **commands** log file contains per servlet entries that are designed to help monitor requests to the server and help diagnose problems. The **commands** log file includes the principal account that logged in and what operations were done from that account.

[Table 3–3](#) describes the **command** log fields. The **commands** log records contain three set fields and one variable field.

Table 3–3 *commands Log Fields*

Field	Description
Time stamp	Identifies when the log entry is created.
Sequence	Unique number for each request.
Servlet	Name of the Calendar Server servlet that handles the request.

Table 3-3 (Cont.) commands Log Fields

Field	Description
Variable	<p>Logs information about the start and end of specific internal server operations.</p> <p>For HTTP commands that are logged from the servlet layers, this field also logs the HTTP request coming in with a [REQ], the HTTP method, URI information, IP address, host name, and port, as well as the user principal information for that request. The corresponding response is marked as [RES], followed by an HTTP status.</p>

The following log entries are for a simple CalDAV query of a calendar event performed by **caluser8@example.com**:

```
[2011-11-16T11:50:21.512-0700] <22> DavServlet[REQ] GET
/davserver/dav/home/caluser8@example.com/calendar/test.ics 127.0.0.1
localhost:8080{principal: caluser8@example.com}
[2011-11-16T11:50:21.512-0700] <22> DavServlet----- {authenticated principal:
caluser8@example.com}
[2011-11-16T11:50:21.512-0700] <22> DavServlet----- Authentication:
caluser8@example.com login_time=0.0 secs,start_service_time=0.0 secs.
[2011-11-16T11:50:21.513-0700] <22> DavServlet----- Get
/davserver/dav/home/caluser8@example.com/calendar/test.ics start.
[2011-11-16T11:50:21.517-0700] <22> DavServlet----- Get end. Processing
time=0.0040 secs.
[2011-11-16T11:50:21.517-0700] <22> DavServlet----- Get
/davserver/dav/home/caluser8@example.com/calendar/test.ics start.
[2011-11-16T11:50:21.521-0700] <22> DavServlet----- Get end. Processing
time=0.0040 secs.
[2011-11-16T11:50:21.526-0700] <22> DavServlet[RES] [200] Command execution time:
0.014 secs
```

The following log entries are from a **list_subscribed.wcap** command executed by user **arnaud@example.com**.

```
[2011-11-14T13:48:36.504-0700] <2056> WCAPServlet [REQ] GET
/davserver/wcap/login.wcap?user=arnaud&password=*****&fmt-out=text/xml 127.0.0.1
localhost:8080{principal: arnaud@example.com}
[2011-11-14T13:48:36.504-0700] <2056> WCAPServlet----- {authenticated principal:
arnaud@example.com}
[2011-11-14T13:48:36.504-0700] <2056> WCAPServlet----- Authentication:
arnaud@example.com login_time=0.0 secs,start_service_time=0.0 secs.
[2011-11-14T13:48:36.504-0700] <2056> WCAPServlet----- Search
/home/arnaud@example.com/ start. scope=SCOPE_ONE
filter={DAV:}resourcetype=DEFAULT_CALENDAR
[2011-11-14T13:48:36.507-0700] <2056> WCAPServlet----- Search end. Processing
time=0.0030 secs. NbEvaluatedNodes=2,NbMatchingNodes=1
[2011-11-14T13:48:36.509-0700] <2056> WCAPServlet[RES] [200] Command execution
time: 0.0060 secs
[2011-11-14T13:48:36.565-0700] <2057> WCAPServlet [REQ] GET /davserver/wcap/list_
subscribed.wcap?id=W6a505a75-cf21-4d68-90b6-35095ad51ccb&fmt-out=text/xml
127.0.0.1 localhost:8080{authenticated principal: arnaud@example.com}
[2011-11-14T13:48:36.596-0700] <2056> WCAPServlet----- ListSubscribedCalendars
/home/arnaud@example.com/calendar_subscribed_set start.
[2011-11-14T13:48:36.596-0700] <2056> WCAPServlet----- Get
/home/arnaud@example.com/calendar_subscribed_set start.
[2011-11-14T13:48:36.598-0700] <2056> WCAPServlet----- Get end. Processing
time=0.0020 secs.
[2011-11-14T13:48:36.600-0700] <2056> WCAPServlet----- ListSubscribedCalendars
```



```

end. Processing time=0.0040 secs.
[2011-11-14T13:48:36.600-0700] <2056> WCAPServlet----- Search
/home/arnaud@example.com/ start. scope=SCOPE_ONE
filter=|({DAV:}resourcetype=CALENDAR)({DAV:}resourcetype=DEFAULT_CALENDAR)
[2011-11-14T13:48:36.612-0700] <2056> WCAPServlet----- Search end. Processing
time=0.012 secs. NbEvaluatedNodes=10,NbMatchingNodes=5
...
[2011-11-14T13:48:36.613-0700] <2057> WCAPServlet[RES] [200] Command execution
time: 0.049 secs
[2011-11-14T13:48:36.668-0700] <2058> WCAPServlet [REQ] GET /davserver/wcap/list_
subscribed.wcap?id=W6a505a75-cf21-4d68-90b6-35095ad51ccb&fmt-out=text/xml
127.0.0.1 localhost:8080{authenticated principal: arnaud@example.com}
[2011-11-14T13:48:36.668-0700] <2056> WCAPServlet----- ListSubscribedCalendars
/home/arnaud@example.com/calendar_subscribed_set start.
[2011-11-14T13:48:36.668-0700] <2056> WCAPServlet----- Get
/home/arnaud@example.com/calendar_subscribed_set start.
[2011-11-14T13:48:36.670-0700] <2056> WCAPServlet----- Get end. Processing
time=0.0020 secs.
[2011-11-14T13:48:36.672-0700] <2056> WCAPServlet----- ListSubscribedCalendars
end. Processing time=0.0040 secs.
[2011-11-14T13:48:36.672-0700] <2056> WCAPServlet----- Search
/home/arnaud@example.com/ start. scope=SCOPE_ONE
filter=|({DAV:}resourcetype=CALENDAR)({DAV:}resourcetype=DEFAULT_CALENDAR)
[2011-11-14T13:48:36.691-0700] <2056> WCAPServlet----- Search end. Processing
time=0.019 secs. NbEvaluatedNodes=9,NbMatchingNodes=4
[2011-11-14T13:48:36.692-0700] <2058> WCAPServlet[RES] [200] Command execution
time: 0.025 secs

```

In this example, following the initial **login.wcap** command, the test issued multiple **list_subscribed.wcap** commands to the Calendar Server WCAP servlet by using the same session ID obtained from the **login** command. The email address of the user principal who issues the request is also included as part of the fourth field, between curly braces.

Administering Calendar Server Access

Calendar Server uses Access Control Lists (ACLs) to determine access control for calendars and scheduling.

Overview of ACLs

An Access Control List (ACL) consists of one or more Access Control Entries (ACEs), which are strings that grant a particular level of access. ACEs collectively apply to the same calendar or component, or for scheduling, to an account. Each ACE in an ACL must be separated by a semicolon. Multiple ACE strings can apply to a single calendar or a single account.

ACLs are denied unless explicitly granted. Some control access is "built-in" to Calendar Server. For example, calendar owners have full access to their calendars. Granting of a particular ACE means implicitly granting anything considered a "lower" ACE.

ACEs are in the form, *ace_principal:right*, where *ace_principal* can be "@" for all, "@domain" for a domain, "user@domain" for a user and "group@domain" for a group. See ["Calendar Access Controls"](#) for ACE rights for calendars and scheduling.

ACEs function in the following way:

- More specific access rights override other access rights.

- Access rights granted to a specific user are more specific than rights granted to a user as member of a group.
- Rights granted as part of the "all" users setting are considered least specific.
- If a user is a member of multiple groups, the highest level of access granted individually by any one of the groups is the access level of the user.
- Calendar Server access control does not take into consideration nesting levels within each group.

You set Calendar Server access controls by using either the **davadmin** command or WCAP commands. Calendar Server uses the **acl** parameter to facilitate storing of the ACE strings. The **acl** parameter is a semicolon-separated list of ACE strings.

Calendar Access Controls

You can set the following four levels of calendar access controls on each calendar collection:

- none (level n)
- read (level r)
- read+write+delete (level w)
- read+write+delete+manage (level a)

An ACE is granted to all (@), domain, user or group. Definition of "all" is made server-wide through the **davcore.acl.calendaranonymousall** configuration parameter. If set to **false**, "all" does not include unauthenticated users. Users and groups are represented by their **mail** address. If you change the **davcore.acl.calendaranonymousall** parameter, the change does not affect ACLs that were previously configured. Changing **davcore.acl.calendaranonymousall** only affects new ACLs.

The following example shows an ACE in which all users get read access, **userA@example.com** gets read, write, delete, and manage access, and all members of **groupA@example.com** get read, write, and delete access.

```
@:r;userA@example.com:a;groupA@example.com:w
```

The **davcore.acl.defaultcalendaracl** configuration parameter defines a default ACL for all calendar collections. You can change this value by using the **davadmin config** command. Calendar Server uses default ACLs for all calendars for which ACLs are not explicitly set.

Scheduling Access Controls

You can set scheduling permissions for an account, which are used for checking a user's freebusy information, inviting a user, and inviting on behalf of a user. The four levels of scheduling access levels are:

- none (level n)
- freebusy (level f)
- freebusy+schedule_invite (level s)
- freebusy+schedule_invite+manage (level m)

An ACE is granted to all (@), domain, user or group. Definition of "all" is made server-wide through the **davcore.acl.schedulinganonymousall** configuration parameter. If set to **false**, "all" does not include unauthenticated users.

You define a default ACL for scheduling by using the **davcore.acl.defaultschedulingacl** configuration parameter.

To invite someone else, you must have a scheduling right of at least **s** for that user.

Setting Access Control for LDAP Groups

In addition to granting calendar and scheduling ACEs to users, you can grant them to LDAP groups. The group is represented by its mail address just like a user. An ACE granted to a group is effective for all members of the group. Any user-specific ACEs granted to a group member override the ACEs granted through group membership.

When evaluating group members for ACL evaluation, only internal group members are considered. That is, only members defined in LDAP by using their DN, directly using the **uniquemember** attribute, or indirectly as an LDAP URL that resolves to member DNs belonging to the group by using the **memberurl** attribute, are considered for ACL evaluation.

Retrieving Access Control Information

You use the **davadmin** command or WCAP commands **get_calprops.wcap**, **search_calprops.wcap**, and **get_accountprops.wcap** to retrieve the access control rights of a logged-in user to a particular calendar, or user. The ACL itself is viewable by owners, administrators, and those users with manage rights only. All other users can get their access rights through the **X-S1CS-MYRIGHTS** property that is returned by the **get** and **search** commands. The value of this property is either calendar-level rights (**n**, **r**, **w**, or **a**); or scheduling rights (**n**, **f**, **s** or **m**), depending on the WCAP call.

See *Calendar Server WCAP Developer's Guide* for information on the **get_calprops.wcap**, **search_calprops.wcap**, and **get_accountprops.wcap** commands.

Access Control Configuration Parameters

Table 3–4 describes the configuration parameters that Calendar Server uses for access control.

Table 3–4 Access Control Configuration Parameters

Parameter	Description
davcore.acl.defaultcalendaracl	Specifies the default access control settings used when creating a new user calendar. The default is: ""
davcore.acl.defaultschedulingacl	Specifies the default access control used for scheduling that is set on a scheduling inbox creation (from the server configuration parameter). The default is: @:s
davcore.acl.calendaranonymousall	Determines if all (@) includes anonymous principals for user calendar access. The default is: true
davcore.acl.schedulinganonymousall	Determines if all (@) includes anonymous principals for scheduling access. The default is: true

Table 3–4 (Cont.) Access Control Configuration Parameters

Parameter	Description
<code>davcore.acl.defaultresourcecalendaracl</code>	Specifies the default access control settings used when creating a new resource calendar. The default is: <code>@:r</code>
<code>davcore.acl.defaultresourceschedulingacl</code>	Specifies the default access control settings set on scheduling inboxes of resource calendars. The default is: <code>@:s</code>

See "[Calendar Server Configuration Parameters](#)" for more information on these access control configuration parameters.

Command-Line Utilities for Access Control

Use the `davadmin calendar` command to get or set calendar ACLs for calendars and the `davadmin account` command to get or set scheduling ACLs for access control. See "[Calendar Server Command-Line Utilities](#)" for more information.

WCAP Commands for Access Control

Use `get_accountprops.wcap` and `set_accountprops.wcap` to access and set an account's scheduling rights. Use `get_calprops.wcap` and `set_calprops.wcap` to access and set the access rights to a calendar. Use `search_calprops.wcap` to view a user's "MYRIGHTS" (privilege level of access to other users' calendars).

For information on these commands, see the topic on Web Calendar Access Protocol overview in *Calendar Server WCAP Developer's Guide*.

Managing Domain ACLs

Domain ACLs control calendar operations that span multiple domains. Calendar Server combines domain ACLs with the calendar and scheduling ACLs to grant or deny levels of access to any calendaring or scheduling operation. All operations within a single domain rely strictly on the calendar and scheduling ACLs.

For more information, see the topic on managing domain access controls in *Calendar Server Security Guide*.

Managing Dynamic Group ACLs

The group ACL feature supports the use of dynamic groups. A dynamic group in LDAP uses the member URL attribute to specify an LDAP filter for the membership of the group. For example, the following URL uses a "department=marketing" filter for group membership:

```
[ldap:///o=mcom.com??sub?(department=marketing)]
```

Users that are determined to be members through the search filter are granted whatever access is given to the group in the ACL.

Administering Scheduling Options

This section describes how manage Calendar Server scheduling rules, booking window, and LDAP group invitation.

Administering scheduling options involves:

- [Configuring Scheduling Options](#)
- [Configuring a Booking Window](#)
- [Modifying Calendar Double Booking](#)
- [Controlling Double Booking When Creating Accounts Automatically](#)
- [Modifying Configuration Parameters That Control Double Booking](#)
- [Overriding the Account Autocreation Through LDAP](#)
- [Manually Creating Accounts](#)
- [Modifying Double Booking on Existing Accounts](#)
- [Inviting LDAP Groups](#)

Configuring Scheduling Options

Calendar Server processes incoming invitations and delivers them to recipients, including delivery to default calendars for internal recipients, without any extra client interaction. If you need Calendar Server to perform additional checks and processing during scheduling, configure the **attendantflag** of the recipient's inbox by using either the **davadmin account** command or the **set_accountprops.wcap** command.

The **attendantflag** properties are:

- Auto Decline of Recurring Meetings. You can disallow recurring meetings for some resource calendars. Any invitation for a recurring meeting received on such a calendar is declined, regardless of its availability.
- Auto Decline on Scheduling Conflict. Calendar Server performs an upfront freebusy check on internal recipients and rejects the invitation if the scheduling results in a conflict and the recipient is set up to auto decline on conflict.
- Auto Accept of invitation. Calendar Server can automatically accept incoming invitations if the recipient is set up for it.

Default settings of the flag are determined by the following configuration parameters:

- **davcore.autocreate.calattendantuserflags**: default value for users (0 = no auto accept, no auto decline booking conflict, no recurrence check on invitations)
- **davcore.autocreate.calattendantresourceflags**: default value for resource calendars (3 = auto accept invitation and auto decline on booking conflict)

Overview of Calendar Booking Window

The booking window is the scheduling time frame that determines how far into the future a calendar or resource can be booked. The optional **minbookingwindow** setting calculates the earliest date and time when a reservation can be made on a calendar for an event starting on a specific date and time. The **maxbookingwindow** setting defines the latest date and time when a resource can be reserved for an event starting on a specific date and time.

If the **minbookingwindow** value is defined, scheduling for an event at a certain time can occur only if the current time is equal to or greater than the date and time calculated by subtracting this value from the event's proposed start time. If the **minbookingwindow** setting is not defined, then bookings can be made at any time before the end of the booking window. The **minbookingwindow** takes a value in the range of 0 to 2 Gbytes. A negative integer value indicates that the **minbookingwindow** is not honored during a freebusy check. The default value is -1.

The **maxbookingwindow** setting (the default value is 365 days) defines the latest date and time when a calendar or resource can be reserved for an event starting on a specific date and time. If the current time is equal to or before the value obtained by subtracting the **maxbookingwindow** value from the start date and time of the event, then the invitation is successful. If this setting is absent, then the scheduling can occur any time from **minbookingwindow**. The **maxbookingwindow** takes a value in the range of 0 to 2 Gbytes.

Taken together, the **minbookingwindow** and **maxbookingwindow** settings provide the window of time events can be scheduled on the calendar, relative to the scheduling time. If a single event's timing is outside that window or a recurring event's instances go beyond the window (either before the minimum bound or after the maximum bound), all instances of the event are declined. Otherwise, only instances that are in conflict with other events are declined, if double booking is disallowed. In the case when no minimum bound is set, the event is autodeclined only when any instance is beyond the upper bound specified by **maxbookingwindow** settings.

In the case when no minimum bound is set, the event is autodeclined only when any instance is beyond the upper bound specified by **maxbookingwindow** settings.

You set the global booking window settings by using the **davcore.scheduling.minbookingwindow** parameter and the **davcore.scheduling.maxbookingwindow** parameter. You can override the global minimum and maximum values by using account-specific settings. These account-level minimum and maximum booking window properties are stored as scheduling inbox collection properties.

In general, only use the **davcore.scheduling.minbookingwindow** parameter for specialized resources or ones that require upfront time to be readied. For example, you might have a conference room that needs to be configured for Internet connectivity and it normally takes a week to do so. In this case, you would set the **davcore.scheduling.minbookingwindow** parameter to 7 (days). The conference room resource calendar would then only be available for booking 7 days in advance.

Note: Calendar Server performs a booking window check only if the account is set up to decline on doublebooking or when outside of booking window, that is, if the attendant flag for the **davcore.autocreate.calattendantuserflags** or **davcore.autocreate.calattendantresourceflags** configuration parameters is set only to 2, 3, 6, or 7. For information on double booking, see "[Modifying Calendar Double Booking](#)."

Configuring a Booking Window

To configure both the minimum and maximum booking windows for accounts, you can use either the **davadmin** command or the **set_accountprops.wcap** interface. In absence of an account property, Calendar Server defaults to using the corresponding system-wide booking window configuration. For example:

- **davadmin** command:

```
./davadmin account modify -a resource1@example.com -y  
minbookingwindow=10,maxbookingwindow=365
```

- **set_accountprops.wcap** command:

```
$(wcapbase)/set_  
accountprops.wcap?account=$(resourceEmail) &minbookingwindow=10&maxbookingwindow
```

```
=365&fmt-out=text/json
```

The minimum and maximum booking window settings are used only if the attendant flag is also set appropriately, that is, set to 2, 3, 6, or 7.

Modifying Calendar Double Booking

Double booking is the ability to schedule and display two events on a calendar at the same time. Calendar Server keeps track of double booking based on a per-account property. You can use the following ways to control double booking:

1. Use account autcreation to automatically assign the double-booking property flag. Additionally, you can control the value assigned during autcreation on a per-account basis by using specific LDAP values in the account's LDAP entry.
2. Manually create accounts with the desired property flag setting.
3. Modify the value of an existing account by using the **davadmin account** command or a client that uses the **wcap_setaccountprops** command.

Note: This feature concerns double booking by invitation only. It does not prevent users with write permission from double booking the calendar by directly creating events in it.

Controlling Double Booking When Creating Accounts Automatically

Because automatic creation of calendar accounts happens when users log in to Calendar Server, you create users by provisioning the users in LDAP then providing instructions for logging in. For more information, see "[Creating Calendar Account with Default Calendar Automatically Upon Login.](#)"

The two Calendar Server "autocreate" configuration parameters that control double booking are:

- For users: **davcore.autocreate.calattendantuserflags** (default is **0**, no auto decline, no auto accept)
- For resources: **davcore.autocreate.calattendantresourceflags** (default is **3**, auto decline and auto accept)

[Table 3–5](#) describes the flag options. Both the **davcore.autocreate.calattendantuserflags** and **davcore.autocreate.calattendantresourceflags** configuration parameters take the options described in the table. Double booking is allowed on calendars when the value of these attendant flag options is **0, 1, 4, or 5**.

Table 3–5 *Flag Options*

Option Value	Description
0	Does not perform autoaccept, does not check booking conflict, does not check recurrence on invitations
1	Automatically accepts invitations
2	Automatically declines if invitation results in booking conflict
3	Automatically accepts invitation and automatically declines on booking conflict
4	Automatically declines recurring meeting invitations

Table 3–5 (Cont.) Flag Options

Option Value	Description
5	Automatically accepts invitations and automatically declines recurring meeting invitations
6	Automatically declines recurring invitations and invitations that cause a booking conflict
7	Automatically accepts invitations, automatically declines recurring invitations and invitations that cause a booking conflict

Note: At the system-wide level, if the **davcore.scheduling.allowownerdoublebooking** parameter is set to **true** (the default value is **false**), then resource calendar owners can double book the resource even if an attendant flag is set that prevents double booking.

Modifying Configuration Parameters That Control Double Booking

Use the **davadmin config modify** command to change double booking behavior.

For example, the following command causes invitations for resources to be automatically accepted on invitation and declined on booking conflict or if outside the allowed booking window.

```
./davadmin config modify -u admin -o davcore.autocreate.calattendantresourceflags -v 3
```

The following command configures the system to not perform autoaccept, not check booking conflict, and not check recurrence on invitations for users:

```
./davadmin config modify -u admin -o davcore.autocreate.calattendantuserflags -v 0
```

Overriding the Account Autocreation Through LDAP

You can use LDAP to override the double booking value that is set on individual accounts during autocreation.

1. Check the value of the **davcore.ldapattr.icsdoublebooking** configuration parameter and change it if necessary.

The value is an LDAP attribute that controls the double booking setting used during autocreation. By default, this attribute is **icsDoublebooking**.

```
./davadmin config list -o davcore.ldapattr.icsdoublebooking
Enter Admin password: password
davcore.ldapattr.icsdoublebooking: icsDoubleBooking
```

2. Update the account's entry in LDAP.

For example, if you use the **icsDoublebooking** attribute, a value of **1** enables double booking, and a value of **0** prohibits double-booking. The autoaccept behavior is also controlled similarly. The default attribute that controls autoaccept is **icsAutoaccept** and it is defined by the **davcore.ldapattr.icsautoaccept** configuration parameter.

Manually Creating Accounts

Instead of relying on account autocreation (when a user logs in for the first time or a user or resource is invited to an event for the first time), you can use the **davadmin account create** command to explicitly create the account with the desired double booking flag setting.

For example, the following command creates a resource calendar that allows double booking and no auto-accept:

```
./davadmin account create -a "resource@example.com" -y "attendanceflag=0"
```

Note: For accounts created through the **davadmin** command, the same defaults for autocreation are used if you do not specify a value.

Modifying Double Booking on Existing Accounts

You can use the **davadmin account modify** command to change the double booking behavior of any account at any time. For example, the following command modifies a resource calendar so that double booking is no longer allowed:

```
./davadmin account modify -a "resource@example.com" -y "attendanceflag=2"
```

Inviting LDAP Groups

You can invite entire groups in the LDAP directory as one attendee. The group is maintained as one attendee in the organizer's calendar, but the Scheduling Service expands the group and adds each member as a recipient of the invitation. When storing the invitation in each recipient's calendar, that recipient is added as an ATTENDEE, which is referenced as a member of the group. When a recipient replies, that recipient is added as an individual ATTENDEE, also referenced as member of the initial group in the organizer's calendar. This feature can be used to invite both static and dynamic groups in LDAP.

The following configuration parameters control this feature:

- **davcore.serverlimits.maxgroupexpansion** - Limits the level of nested group expansion. By default, it is 3 (three levels deep)
- **davcore.serverlimits.maxattendeesperinstance** - For scheduling, limits the number of members as a result of group expansion. The default is 1000.
- **davcore.ldapattr.dngroupmember=uniquemember**,
davcore.ldapattr.urlgroupmember=memberurl,
anddavcore.ldapattr.mailgroupmember=mgrprfc822mailmember - Specify the various type of LDAP group memberships. **davcore.ldapattr.dngroupmember** is used for group members specified as a DN, which denotes static membership. **davcore.ldapattr.urlgroupmember** is used for group members specified through an LDAP filter, which denotes dynamic membership. **davcore.ldapattr.mailgroupmember** is used for group members specified through an email address.

If you have your own schema elements that follow the semantics of the preceding default settings, you could add those attributes to the corresponding list by using a space delimited fashion.

Administering Resource Calendars

Administering resource calendars involves:

- [Provisioning Resource Calendars \(commadmin\)](#)
- [Provisioning Resource Calendars \(Delegated Administrator Console\)](#)
- [Managing a Resource Calendar's Mailbox](#)

About Resource Calendars

Entities that can be scheduled but that do not control their own attendance status are called *resources*. You provision resources in LDAP, either by using Delegated Administrator or LDAP tools. See "Messaging Server and Calendar Server LDAP Object Classes and Attributes" in *Communications Suite Schema Reference* for object classes and attributes required or allowed by Calendar Server 7. Once provisioned, the actual calendars are automatically created on first invite, if auto-creation is enabled. You can also create the calendar account with the default calendar for the provisioned resource by using the **davadmin account create** command. See "[davadmin account](#)" for details.

You can manage resource accounts and calendars just like user accounts and calendars. In addition, you can set the resource account owner by using either the **davadmin account** or **set_accountprops.wcap** commands.

An LDAP "mail" attribute (most often the **mail** attribute) is required to be present for resource entries. Though resources do not check email, Calendar Server uses this address value to identify and schedule the resource, and thus it must be unique to the resource. You do not need to specify other values, such as owner's email address. Depending on your site's requirements, you may choose to discard or manage the email that is sent to resource email addresses. See "[Managing a Resource Calendar's Mailbox](#)" for more information.

Note: When sharing a resource calendar and you do not receive the email notification advising of the share in your local language then set the **preferredLanguage** attribute to be your local language in the resource LDAP.

Provisioning Resource Calendars (commadmin)

When you have multiple back-end hosts, use **-A davstore:backend -E email** to provision the Calendar Server back-end host for the resource, where *backend* is the JBDC resource name without the JBDC prefix.

This is mandatory for:

- Multiple Calendar Server 7 back-end hosts
- Calendar Server 6 and 7 coexistent deployments

Example for one back-end host:

```
DelegatedAdmin_home/bin/commadmin resource create -D admin -c bigdipper -N "Big Dipper Conference Room" -E bigdipper@us.example.com -o calmaster
```

The LDAP entry for this example resembles the following.

```
dn: uid=bigdipper,ou=People,o=us.example.com,o=isp
cn: Big Dipper Conference Room
davuniqueid: d256e98e-fb1c-470e-9f78-eb80bc5e5ee8
icscalendar: bigdipper@us.example.com
icsstatus: active
inetresourcestatus: active
```

```
mail: bigdipper@us.example.com
objectclass: daventity
objectclass: inetresource
objectclass: icscalendarresource
objectclass: top
owner: uid=calmaster, ou=People, o=us.example.com, o=isp
uid: bigdipper
```

Example for multiple back-end host deployment:

```
DelegatedAdmin_home/bin/commadmin resource create -D calmaster -d demo.example.com
-w password -u room1 -c room1 -N Room1 -A davstore:defaultbackend -E
room1@demo.example.com
```

Notes:

- For **commadmin resource create**, use the **-o owner** option, if you want a Convergence user to be able to subscribe to the calendar.
- The **-o owner** option can only be used on a uid in the same domain as the resource.

Provisioning Resource Calendars (Delegated Administrator Console)

To provision resource calendars by using the Delegated Administrator console:

1. Log in to Delegated Administrator Console.
2. Select the organization in which to create the resource calendar.
3. Click the **Calendar Resources** tab.
4. Click **New**.

The New Calendar Resource page is displayed.

5. Type the required resource information, including Resource ID, Calendar Resource Name, and Resource Owner.

You cannot create a resource without a resource owner.

6. In a multiple back-end deployment, make sure that you type the correct calendar store.
7. Click **Next**.
The summary page is displayed.
8. Click **Finish** to create the resource.

Managing a Resource Calendar's Mailbox

Use one of the following options for a resource calendar's mailbox:

- In the resource's LDAP entry, set the mail delivery option to forward and set the forwarding address to the bitbucket channel.

Here is sample LDIF:

```
dn: uid=calresbitbucket,ou=People, o=example.com, o=dav
uid: calresbitbucket
cn: CalResBitBucket
description: Conference Room
mail: calresbitbucket@example.com
icsStatus: active
objectClass: top
```

```
objectClass: inetresource
objectClass: icscalendarresource
objectClass: daventity
objectClass: inetMailUser
objectClass: inetlocalmailrecipient
inetResourceStatus: active
owner: uid=john, ou=People, o=example.com, o=dav
mailDeliveryOption: forward
mailForwardingAddress: calresbitbucket@[channel:bitbucket]
mailhost: icsmail.example.com
```

- Assign the resource a valid email address and manage its mailbox. Either assign the password and management of that mailbox to the owner of the resource, or expire and expunge the email account more aggressively, so that email does not build up.

Administering Time Zones Support

Time zones are an important part of any time and date based application like calendaring and scheduling. Calendar Server uses the standard Time Zone Database, which is maintained by IANA, for time zone information. The timez one information is compiled and shipped along with Calendar Server. Each Calendar Server patch is updated to the latest available Time Zone Database. For more information on IANA, see the website at:

<http://www.iana.org/time-zones>

Administering time zones includes:

- [Adding New WCAP Time Zones](#)
- [Adding an Alias to an Existing Time Zone](#)
- [Adding an Alias to an Existing Time Zone](#)

Adding New WCAP Time Zones

Calendar Server makes a subset of supported time zones available to WCAP clients. Calendar Server derives the supported WCAP time zones to match those that the Convergence client supports. If you modify the Convergence client to support a new time zone, you must also add the new time zone to Calendar Server's WCAP time zone list.

The list of WCAP time zones is derived from the list provided in the *CalendarServer_home/config/timezoneids.txt* file. The file consists of the supported Time Zone Database **timezoneid** strings followed by their aliases, if any. The alias names are separated by a colon character. The file has one line per supported time zone.

For more information on how this works with Convergence, see the topic on adding and modifying Calendar Server time zones in *Convergence Customization Guide*.

Adding an Alias to an Existing Time Zone

The following task applies to WCAP clients that use time zone aliases. Currently, the Convergence client does not support time zone aliases.

To add an alias to an existing time zone:

1. Edit the *CalendarServer_home/config/timezoneids.txt* file.

2. Find the corresponding time zone line and add a colon followed by the new alias name at the end of the line. For example, to add the alias **US West Coast** to the **America/Los_Angeles** time zone entry, change:

```
America/Los_Angeles:Pacific Standard Time:US/Pacific
```

to

```
America/Los_Angeles:Pacific Standard Time:US/Pacific:US West Coast
```

3. Restart Calendar Server.

See "[Stopping and Starting Calendar Server](#)" for details.

Adding a New Time Zone

To add a new time zone:

1. Find the time zone or its equivalent in the "[Time Zone Database](#)" list supported by the server.
2. Edit the `CalendarServer_home/config/timezoneids.txt` file.
3. Add an entry for that time zone ID to the end of the file.
4. If you prefer a different name, add that name as an alias too, by adding a colon and the name following the time zone ID entry.
5. Restart Calendar Server.

See "[Stopping and Starting Calendar Server](#)" for details.

Customizing Calendar Notifications

Calendar Server provides preformatted notification messages to be sent to calendar owners when changes occur in calendar resources and properties. You can customize these files for your own deployment. See "[Using Calendar Server Notifications](#)" for details.

Administering the Calendar Server Back End Databases

This section describes how to administer the Calendar Server back end databases.

Administering the MySQL Database

The following links provide information about administering MySQL. For more information, consult the MySQL documentation directly.

- "[Starting and Stopping MySQL Automatically](#)" in *MySQL 5.5 Reference Manual*
- "[MySQL Server and Server-Startup Programs](#)" in *MySQL 5.5 Reference Manual*
- "[MySQL Administrative and Utility Programs](#)" in *MySQL 5.5 Reference Manual*
- [Backing Up and Restoring Calendar Server Files and Data](#)

Caution: You can view contents of the back-end store by using standard MySQL tools. Do not use MySQL tools to modify your data.

Administering the Oracle Database

The following links provide information about administering Oracle Database. For more information, consult the Oracle Database documentation directly.

- "Stopping and Starting Oracle Software" in *Oracle Database Administrator's Reference 11g Release 1 (11.1) for Linux and UNIX-Based Operating Systems*
- "Administering Oracle Database" in *Oracle Database Administrator's Reference 11g Release 1 (11.1) for Linux and UNIX-Based Operating Systems*
- [Backing Up and Restoring Calendar Server Files and Data](#)

Caution: You can view contents of the back-end store by using standard Oracle Database tools. Do not use Oracle Database tools to modify your data.

Backing Up and Restoring Calendar Server Data

See "[Backing Up and Restoring Calendar Server Files and Data](#)" for information.

Removing Unwanted Calendar Data to Reclaim Space

To reclaim space in the Calendar Server database, you can purge deleted calendar entries and purge messages from the scheduling inbox and outbox.

Purging Deleted Calendar Entries

When calendar data is deleted, either by users deleting events and tasks, or by using the **davadmin account delete**, the data is only marked for deletion. The data is actually purged from the calendar database when the expiry time is reached. The default expiry time is 30 days and is controlled by the **store.dav.defaultbackend.purgedelay** configuration parameter. See "[Calendar Server Configuration Parameters](#)" for more information on this parameter.

Purging Messages From the Scheduling Inbox and Outbox

Oracle Communications Calendar Server 7 supports implicit scheduling. The actual scheduling process involves writing of the iTIP request to the sender's calendar outbox, then posting it to the recipients' inboxes, and eventually writing to the recipients' default calendars. The interim iTIP messages are stored as resources in the Calendar Server users' scheduling outbox and inbox. You can automatically purge these resources in the outbox and inbox collections.

To set the interval to purge messages from the scheduling outbox and inbox, use the **davcore.scheduling.calendaroutboxexpirytime** and **davcore.scheduling.calendarinboxexpirytime** parameters. See "[Calendar Server Configuration Parameters](#)" for more information on these options. These parameters enable you to set the expiration time for scheduling messages in all the users' outbox and inbox.

For each parameter, specify the number of seconds after which the resources in the outbox or inbox should be deleted. The default for **davcore.scheduling.calendaroutboxexpirytime** is 604800 seconds (7 days), and the default for **davcore.scheduling.calendarinboxexpirytime** is 2592000 seconds (30 days).

Setting Up and Managing Calendar Server Users

This chapter describes how to provision Calendar Server users and manage calendar accounts.

- [Provisioning Calendar Server Users](#)
- [Managing Calendar Users and Accounts](#)
- [Subscribing and Unsubscribing Calendars](#)
- [About Configuring External Authentication](#)
- [Configuring Proxy Authentication](#)

Provisioning Calendar Server Users

This section describes how to provision Calendar Server users and contains the following topics:

- [Overview of Provisioning Calendar Server](#)
- [Provisioning Calendar Users by Using Delegated Administrator](#)
- [Provisioning Calendar Users Across Virtual Domains](#)

Overview of Provisioning Calendar Server

Calendar Server uses Directory Server to store and retrieve user and resource information and to perform authentication. Calendar Server does not add or modify LDAP data. Calendar Server data (such as todos and events) is stored in an SQL database, which can be either MySQL Server or Oracle Database.

Note: Calendar Server 7 supports the same LDAP schema used by Calendar Server 6. Some additional object classes and attributes were added for Calendar Server 7.

By default, Calendar Server automatically creates the necessary entries in the SQL database for users upon their initial Calendar Server login. However, you must also perform some basic LDAP user provisioning for users and resources to be able to access Calendar Server services, and for Calendar Server automatic account creation to work. You can provision Calendar Server users and resources in the Directory Server LDAP by using either Delegated Administrator or LDAP tools.

You must provision Calendar Server users and resources so that Calendar Server can automatically create accounts and users and resources can access Calendar Server services. You must provision users and resources with the following attributes:

- An email attribute, such as **mail**.
- A unique ID attribute corresponding to the value for the server **davcore.uriinfo.permanentuniqueid** configuration parameter. The default value is **davUniqueId**. Be sure to also index the attribute used for **davcore.uriinfo.permanentuniqueid**, as Calendar Server performs searches on it.

To define these attributes, the corresponding object classes must be present in LDAP. The **davEntity** object class defines the **davUniqueId** attribute. In addition to the preceding two LDAP requirements, if your deployment consists of multiple back-end databases, you must define the Store ID attribute. The **davEntity** object class also defines the default Store ID attribute. The default value for Store ID is **davStore**.

By default, if you provision Calendar Server users for email and unique ID attributes (and the Store ID attribute when multiple back-end databases are deployed), users have a status of **active**. The **active** status enables users to access Calendar Server services. To deny Calendar Server services to users, you specify a value of either **inactive** or **deleted** for the user's **icsStatus** attribute.

If you have a co-existent deployment of both Calendar Server 6 and Calendar Server 7, and are migrating users to Calendar Server 7, you must update the user's LDAP once the user is marked for migration and taken offline for migration. You can only migrate the user at that point. Calendar Server uses an LDAP attribute to determine if a user has been migrated. By default, the **davStore** attribute is used, but you can choose another attribute if desired. In a single back-end deployment, this attribute must be added with the value of **defaultbackend**. In a multiple back-end deployment, the value must be the logical back-end ID for the database where the user's data resides after migration. Again, the object class that defines the **davStore** attribute is **davEntity**.

For more information on Calendar Server LDAP schema, object classes, and attributes, see *Calendar Server Concepts and Communications Suite Schema Reference*.

Provisioning Calendar Users by Using Delegated Administrator

Delegated Administrator 7 supports Calendar Server 7 provisioning. Calendar Server 7 makes use of the **davStore** attribute and not **icsdwpghost** (which is used in Calendar Server 6) to assign a specific back-end in a multiple back-end scenario. You can add the **davStore** LDAP attribute to users' and resources' subject entries to associate those users and resources with a particular back-end Calendar Server store. The value of the **davStore** attribute is equal to one of the **davStore** IDs defined in the server configuration. **davStore** is single valued. When not present, a server configurable default **davStore** ID is used.

When a user or group is assigned a calendar service, the **davEntity** object class is added along with **icscalendaruser** or **icscalendargroup** object class, enabling you to provision the user or the group with a **davStore** attribute.

The new user wizard with the Calendar Service Details consists of the following fields:

- Calendar Host
- Calendar Store
- Timezone

where the Calendar Host and Timezone are applicable to Calendar Server 6 and Calendar Store is applicable to Calendar Server 7.

The user properties section in the Calendar Service Details consists of the following fields:

- Calendar Host: (**icsdwphost**)
- Calendar Store: (**davstore**)
- Email Address: (**mail**)
- Default Calendar: (**icscalendar**)
- Owned Calendar: (**icscalendarowned**)
- Subscribed Calendar: (**icssubscribed**)
- Timezone: (**icstimezone**)
- Calendar Service Status: (**icsstatus**)

where the Calendar Host, Default Calendar, Timezone, Owned Calendar, Subscribed Calendar are applicable to Calendar Server 6 and Calendar Store is applicable to Calendar Server 7. Email Address and Calendar Service Status are applicable to both servers.

A new field called Calendar Store is added for Calendar Server 7 users. When configuring a user in Delegated Administrator with Calendar services, a valid davstore ID has to be entered in the Calendar Store field instead of a host name.

Provisioning Calendar Users Across Virtual Domains

Calendar Server supports hosted (or virtual) domains. In a hosted domain installation, each domain shares the same instance of Calendar Server, which enables multiple domains to exist on a single server. Each domain defines a name space within which all users, groups, and resources are unique. Each domain also has a set of attributes and preferences that you specifically set.

Installing and configuring hosted domains on a server involves these high-level steps:

1. Installing and configuring Calendar Server and Delegated Administrator.

For more information, see *Calendar Server Installation and Configuration Guide* and *Delegated Administrator Installation and Configuration Guide*.

2. Using Delegated Administrator to create the hosted domain, and users, resources, and groups in that hosted domain.

For more information, see *Delegated Administrator System Administrator's Guide*.

3. Setting domain ACLs.

For more information, see the topic on managing domain access controls in *Calendar Server Security Guide*.

Note: Always perform your provisioning for Schema 2 with Delegated Administrator. Schema 1 provisioning tools do not support hosted domains.

Managing Calendar Users and Accounts

Managing calendar users and accounts includes:

- [Defining Valid Calendar Users](#)
- [Enabling and Disabling Automatic Account Creation](#)

- [Creating Calendar Account with Default Calendar Automatically Upon Login](#)
- [Preventing a User or Resource From Accessing Calendar Server](#)
- [Checking for Active Calendar Users](#)
- [Removing Calendar Users](#)
- [Removing a Calendar User \(Example\)](#)
- [Moving Calendar Users to a New Back-End Database](#)
- [Changing a User's Email Address in the Calendar Server Database](#)

Defining Valid Calendar Users

The `davcore.ldapattr.userobject` configuration parameter defines what LDAP object class is required for a calendar user entry to be considered valid. By default, `davcore.ldapattr.userobject` is empty. A value that would make sense is `icsCalendarUser`, however, if you use custom provisioning, you can define your own object class instead.

To define a valid calendar user:

Set the `davcore.ldapattr.userobject` configuration parameter to a valid LDAP object class. For example:

```
./davadmin config modify -o davcore.ldapattr.userobject -v icsCalendarUser
```

Setting `davcore.ldapattr.userobject` to the LDAP attribute `icsCalendarUser` would consider users without that object class in their LDAP entries as invalid calendar users, therefore no auto-provisioning would take place for them in the calendar store.

Enabling and Disabling Automatic Account Creation

You can enable or disable, on a system-wide basis, automatic account creation. When automatic account creation is enabled, users' accounts are automatically created for them when they first log in to Calendar Server.

- To enable automatic calendar accounts creation:

```
./davadmin config modify -o davcore.autocreate.enableautocreate -v true
```
- To disable automatic calendar accounts creation:

```
./davadmin config modify -o davcore.autocreate.enableautocreate -v false
```

See the `davcore.autocreate.enableautocreate` parameter in "[Calendar Server Configuration Parameters](#)" for more information.

Creating Calendar Account with Default Calendar Automatically Upon Login

To create accounts with default calendars automatically when users log in:

1. Create users by provisioning them in LDAP.
2. Use the `davadmin` command to set any of the `davcore.autocreate.*` parameters to customize your deployment.

See "[Calendar Server Configuration Parameters](#)" for more information.

3. Enable automatic account creation.

See "[Enabling and Disabling Automatic Account Creation](#)."

4. Provide users instructions for logging in to Calendar Server.

Preventing a User or Resource From Accessing Calendar Server

To prevent a particular user from accessing Calendar Server, set the **icsStatus** attribute to **inactive**. You can set the **icsStatus** attribute on both a per user or resource, or domain basis. For more information on **icsStatus**, see *Communications Suite Schema Reference*.

Checking for Active Calendar Users

Use the **commands** log to check for which users have been querying their calendars. In this way you can determine which users are active. For more information, see "[Using the commands Log](#)."

Tip: Use a cron job script to automatically scan the **commands** log file and generate this kind of report.

Removing Calendar Users

Completely removing a calendar user involves deleting it from the Calendar Server database and the LDAP directory.

To remove a Calendar Server user:

1. Run the **davadmin account delete** command to delete the user account and calendars from the Calendar Server back-end database.
2. Set the **icsStatus** attribute for users being deleted to "removed."

You must do this so that the last step of deleting the account from LDAP by using the **commadmin** command is successful. Even when you delete the user account in Step 1, the LDAP entry for the user remains with the **icsStatus** attribute unchanged. Thus, you must manually set the **icsStatus** attribute to "removed" so that running the Delegated Administrator **commadmin domain purge** command removes the user's LDAP entry.

3. Run the Delegated Administrator **commadmin domain purge** command to remove the user's LDAP entry.

Note: The **commadmin domain purge** command does not, however, remove the user as a member from any groups of which the user is a member. To completely remove a user's entry from the directory you must enable the Referential Integrity plug-in. See the topic on maintaining referential integrity in *Oracle Fusion Middleware Administration Guide for Oracle Directory Server Enterprise Edition 11*.

When you remove a calendar user by running the **davadmin account delete** command, the back-end resources for the user are deleted, and then purged according to the value of the **store.dav.defaultbackend.purgedelay** configuration parameter. See "[Removing Unwanted Calendar Data to Reclaim Space](#)" for more information.

Note: The **search_calprops.wcap** command does not return calendars belonging to accounts which have a status of 'inactive,' 'removed,' or 'deleted.'

Removing a Calendar User (Example)

These steps show how to use the command-line to remove a calendar user. In this example, the user is:

```
dn: uid=jsmith,ou=People,o=us.example.com,o=isp
```

These steps use the **ldapmodify** command to make changes to the Directory Server LDAP. Other LDAP tools are also available.

1. In LDAP, search for the user(s) to be removed. For example:

```
ldapsearch -h ds.us.example.com -b "o=isp" "(uid=jsmith)"
version: 1
dn: uid=jsmith,ou=People,o=us.example.com,o=isp
...
```

2. Run the **davadmin** command to remove the calendar account and its calendars from the Calendar Server back-end database:

```
CalendarServer_home/sbin/davadmin account delete -a jsmith@us.example.com
Enter Admin password:
Are you sure you want to delete the account of jsmith@us.example.com and all of
its calendars (y/n)? [n] y
```

3. Verify that the calendar account has been removed from the Calendar Server back-end database:

```
CalendarServer_home/sbin/davadmin account list -a jsmith@us.example.com
Enter Admin password:
Unknown user: jsmith@us.example.com
```

4. Run the **ldapmodify** command to change the user's status to **removed** in LDAP:

```
ldapmodify -h ds.us.example.com -D "cn=Directory Manager" -w password
dn: uid=jsmith,ou=People,o=us.example.com,o=isp
changetype: modify
replace: icsStatus
icsStatus: removed
^D
modifying entry uid=jsmith,ou=People,o=us.example.com,o=isp
```

Alternately, you can run the following **commadmin** command:

```
DelegatedAdmin_home/bin/commadmin user modify -D admin -d us.example.com -l
jsmith -A icsstatus:removed
```

5. (Optional) If user account is configured for mail service, remove the service for the user and run the **msuserpurge** command to purge the account from the Messaging Server message store. For example:

```
DelegatedAdmin_home/bin/commadmin user delete -l jsmith -S mail
MessagingServer_home/lib/msuserpurge -d us.example.com -g 0
```

The **-g 0** option performs an immediate purge.

6. Run the **commadmin** command to purge the account:

```
DelegatedAdmin_home/bin/commadmin domain purge -D admin -n us.example.com -d
us.example.com -g 0
Enter login password:
OK
```

The **-g 0** option performs an immediate purge.

7. Verify that the user has been removed from LDAP:

```
ldapsearch -h ds.us.example.com -b "o=isp" "(uid=jsmith) "
```

Nothing is returned from this command, indicating that the user is no longer present in LDAP.

Moving Calendar Users to a New Back-End Database

To move an existing calendar user from one back-end database to another:

1. Take the user offline by using the **ldapmodify** command to set the LDAP attribute **icsstatus** to **inactive**.
2. Back up the user's data by using the **davadmin db backup** command.
3. Delete the user's data from the old back-end database by using the **davadmin account delete** command.
4. Update the user's **davStore** attribute to the new back-end database by using the **ldapmodify** command.
5. Restore the user's data to the new back-end database by using the **davadmin db restore** command.
6. Re-activate the user by setting the LDAP attribute **icsstatus** to **active**.
7. Restart GlassFish Server.
8. Verify that the data has been successfully moved by having the user log in and view calendar data.

The following example shows how to move user **caltest1@backend1.com** from **backend1** to the default back-end **host1**. In this example:

- The default back end is on host **host1** and the database name on the default back end is **caldav**.
 - The other back end is on host **backend1** and the database name on **backend1** is **caldav_backend1**.
 - The user **caltest1@backend1.com** has data located on **backend1**.
1. Take the user offline by using the **ldapmodify** command to set the LDAP attribute **icsstatus** to **inactive**.

```
ldapmodify -D "cn=Directory Manager" -w password
dn: uid=caltest1, ou=People, o=backend1.com, o=dav
changetype: modify
replace: icsStatus
icsStatus:inactive
```

2. Back up the user's data by using the **davadmin db backup** command.

```
./davadmin db backup -u database_user -a caltest1@backend1.com -H backend1 -d
caldav_backend1 -k caltest1.bak
```

3. Delete the user's data from the old back-end database by using the **davadmin account delete** command.

```
./davadmin account delete -u admin_user -a caltest1@backend1.com
```

4. Update the user's **davStore** attribute to the new back-end database by using the **ldapmodify** command.

```
ldapmodify -D "cn=Directory Manager" -w password
dn: uid=caltest1, ou=People, o=backend1.com, o=dav
changetype: modify
replace: davStore:
davStore: defaultbackend
```

5. Restore the user's data to the new back-end database by using the **davadmin db restore** command.

```
./davadmin db restore -H host1 -u database_user -d caldav -k caltest1.bak
```

6. Re-activate the user by setting the LDAP attribute **icsstatus** to **active**.

```
ldapmodify -D "cn=Directory Manager" -w password
dn: uid=caltest1, ou=People, o=backend1.com, o=dav
changetype: modify
replace: icsStatus
icsStatus: active
```

7. Restart GlassFish Server.

Changing a User's Email Address in the Calendar Server Database

When a user undergoes an email address change, use this procedure to fix notification addresses, organizer addresses, attendee addresses, and alarm addresses in the Calendar Server database.

To change a user's email address in the Calendar Server database:

1. Add the previous email address value to the LDAP attribute defined as **mailAlternateAddress**, that is, the attribute defined by the **davcore.ldapattr.mailalternateaddress** configuration parameter.
2. Replace the value for the user's mail attribute in LDAP with the new email address.
3. Run the **davadmin account repair -a** command on the old email address. See **davadmin account** in "[Calendar Server Command-Line Utilities](#)" for more information on this command.
4. (Optional) Remove the previous email value from the **mailAlternateAddress** values.

Subscribing and Unsubscribing Calendars

Calendar Server administrators can subscribe or unsubscribe calendars for a user by using the **davadmin account** command with **subscribe** or **unsubscribe** actions.

To subscribe a user to another user's calendar:

1. User A gives User B read, write, or all rights to User A's calendar.
2. User B is subscribed to User A's calendar:

```
davadmin account subscribe -a User_B -c User_A's_calendar
```

For example:

```
./davadmin account subscribe -a userb@example.com -c
/home/usera@example.com/calendar/
```

To unsubscribe a user from another user's calendar:

```
davadmin account unsubscribe -a User_B -c User_A's_calendar
```

For example:

```
./davadmin account unsubscribe -a userb@example.com -c
/home/usera@example.com/calendar/
```

To subscribe using a collection file:

1. Create a file such as **/tmp/list_of_collections** with the following content:

```
/home/usera@example.com/calendar/
/home/userc@example.com/call/
/home/userd@example.com/personal/
```

2. Run the following command:

```
./davadmin account subscribe -a userb@example.com -C /tmp/list_of_collections
```

See the **account Operation** commands in "[Calendar Server Command-Line Utilities](#)" for more information.

About Configuring External Authentication

Calendar Server enables user authentication against a separate, LDAP directory external to the Calendar server environment. Such a configuration is useful in hosted environments for delegating one administrative aspect to a provider (managing the Calendar Server front- and back-end hosts and LDAP directory with non-sensitive data), while maintaining control over the LDAP user passwords in the internal, corporate network. In this setup, Calendar Server uses the external directory for user authentication.

Configuring Calendar Server for external authentication consists of the following high-level steps:

1. Creating the LDAP pool for the external authentication directory
2. Specifying how to search for users in that directory
3. Specifying how to map the external user entry back into a Unified Communications Suite directory user entry

Configuring Calendar Server for External Authentication

1. Use the **davadmin ldappool** command to create then verify an LDAP pool for the external authentication directory. For example:

```
cd CalendarServer_home/sbin
./davadmin ldappool create -n myldap -y
'ldaphost=ldap.example.com,ldapport=389,ldapusessl=true,binddn="cn=Directory
Manager",bindpassword=password'
Enter Admin password: password

./davadmin ldappool list
Enter Admin password:
Pool Name: myldap
ldaphost: ldap.example.com
ldapport: 389
ldapusessl: true
ldappoolsize: 10
binddn: cn=Directory Manager
bindpassword: *****
```

```
ldaptimeout: 60
ldappoolrefreshinterval: 1
```

You can also set other parameters, such as **ldappoolsize** and **ldaptimeout**. See the **dadadmin ldappool** command in "[Calendar Server Command-Line Utilities](#)" for more information.

2. Define how to search for users in that directory by adding the **externalAuthPreUrlTemplate** LDAP attribute to each of the domain entries associated with that external directory. The attribute value is an LDAP URL (<http://tools.ietf.org/html/rfc4516>) of the following form:

```
ldap://server_name/search_base_DN?attributes?scope?search_filter
```

where:

server_name must correspond to a defined LDAP pool.

search_base_DN and *search_filter* can contain the following patterns:

- **%o** (original login ID, as provided by the user over protocol)
- **%U** (user part of login ID)
- **%V** (domain part of login ID)

The **%** character in **%o**, **%U**, and **%V** needs to be encoded as per the general URI definition. That is, the **%** character becomes **%25**.

3. Define how to map the external user entry back into a UCS directory user entry.

Do the external directory user entries have a **mail** attribute value corresponding to their internal Unified Communications Suite directory attribute value?

- If yes, no further configuration is required. (In Step 2, the list of attributes to retrieve should simply include the **mail** attribute.)
- If no, define a search to be issued against the Unified Communications Suite directory by using the **externalAuthPostUrlTemplate** domain entry attribute. As with the **externalAuthPreUrlTemplate** attribute, the **externalAuthPostUrlTemplate** value is an LDAP URL of the following form:

```
ldap://server_name/search_base_DN?attributes?scope?search_filter
```

For more information, see the **externalAuthPostUrlTemplate** attribute description in *Communication Suite Schema Reference*.

Note the following:

- The *server_name* is ignored and should be empty because the lookup is performed against the internal Communications Suite directory.
- The *attributes* to be retrieved must include the **mail** attribute.

Example: External Authentication by Using cn

In this external authentication scenario, **example.com** is the default domain and uses the **cn** attribute as the login ID. Each user entry in the external authentication directory contains a **ucsUid** attribute value that corresponds to the internal Unified Communications Suite directory **uid** attribute value (in the Unified Communications Suite user entry). The LDAP pool is **myldap** and the following two attributes have been added to the **example.com** domain entry:

```
externalAuthPreUrlTemplate: ldap://myldap/dc=example,dc=com?ucsUid?sub?(cn=%25U)
```



```
externalAuthPostUrlTemplate:
ldap:///uid=%25A[ucsUid],ou=people,o=example.com?mail?base?(objectclass=*)
```

The LDAP entry in the external directory for a sample user, **John Doe**, looks like the following:

```
dn:cn=John Doe,ou=people,o=marketing,dc=example,dc=com
cn:John Doe
ucsUid:jdoe
...
```

The LDAP entry in the internal Unified Communications Suite directory for **John Doe** looks like the following:

```
dn:uid=jdoe,ou=people,o=example.com
cn:Doe, John
uid:jdoe
mail:john.doe@example.com
...
```

The user authenticates by using the login ID **John Doe**. Given that this login ID has no domain part, the default domain (**example.com**) is assumed and the **externalAuthPreUrlTemplate** attribute of that domain entry is used to construct the following search against the server defined by **authPool**:

- base DN: **dc=example,dc=com**
- scope: subtree search
- filter: **(cn=John Doe)**
- attributes to retrieve: **ucsUid**

The entry **dn:cn=John Doe,ou=people,o=marketing,dc=example,dc=com** is returned and that dn is used to issue an LDAP bind request to verify the user's password.

That same entry (containing the **ucsUid** attribute) is used to construct a second search, against the Unified Communications Suite user/group directory:

- base DN: **uid=jdoe,ou=people,o=example.com**
- scope: base search
- filter: **(objectClass=*)**
- attributes to retrieve: **mail**

The correct entry is found and the authentication is considered successful.

Configuring Proxy Authentication

You can configure Calendar Server for proxy authentication to enable a calendar administrator to log in to Calendar Server on behalf of a calendar user. For more information, see *Calendar Server Security Guide*.

Enabling Advanced Features

This chapter provides details on enabling advanced features in Oracle Communications Calendar Server.

Topics in this section:

- [Enabling Attachments](#)
- [Enabling Apple iCal Private/Confidential Support](#)
- [Enabling SMS Calendar Notifications in Convergence](#)
- [Enabling the iSchedule Channel to Handle iMIP Messages](#)
- [Enabling CalDAV and CardDAV Autodiscovery](#)

Enabling Attachments

To enable or disable attachments:

1. List the **davcore.attachment.enable** parameter value:

```
./davadmin config list -u admin -o davcore.attachment.enable
```

2. Modify the parameter value:

- a. To enable attachments:

```
./davadmin config modify -u admin -o davcore.attachment.enable -v true
```

- b. To disable attachments:

```
./davadmin config modify -u admin -o davcore.attachment.enable -v false
```

You do not need to restart the Calendar Server process for a change in the **davcore.attachment.enable** parameter to take effect.

Enabling Apple iCal Private/Confidential Support

To enable Apple iCal private/confidential support, set the **davcore.acl.appleprivateevent** to **true**.

```
./davadmin config modify -u admin -o davcore.acl.appleprivateevent -v true
```

This triggers the sending of the DAV header with the value **calendarserver-private-events** to the iCal client. As a result, the iCal client includes a check box labeled "private" in the event creation and modification UI. When the private check box is checked, the iCal property **X-CALENDARSERVER-ACCESS** is set to **CONFIDENTIAL**. If the check box is unchecked, the

X-CALENDARSERVER-ACCESS is set to **PUBLIC**. Apple iCal currently supports only these two values.

Enabling SMS Calendar Notifications in Convergence

You can enable SMS notifications for calendar event reminders (but not for calendar invitations) in Convergence. See "How Do I Turn on SMS Notifications for Calendar Event Reminders in Convergence?" at:

<https://wikis.oracle.com/pages/viewpage.action?pageId=15467240>

Enabling the iSchedule Channel to Handle iMIP Messages

The iSchedule database is used to manage external calendar invitations. The established standard for scheduling between two separate calendar servers is still only through iCalendar Message-Based Interoperability Protocol (iMIP), which sends calendar data over email. Previously, end users had to manually import such invitations and responses that arrived in email into their calendars. Starting with Messaging Server 7 Update 5, you can use an iSchedule channel that interprets such mail messages and posts them to the Calendar server directly. Thus, external invitations and responses get into users' calendars without any user intervention. See the topic on using the iSchedule channel to handle iMIP messages in *Messaging Server Unified Configuration Administration Guide* for instructions on how to set up Messaging Server. No special setup is required for Calendar server.

You can also use the **service.dav.ischedulewhitelist** configuration parameter to prevent denial of service attacks on the iSchedule port. The **service.dav.ischedulewhitelist** parameter lists the hosts from which iSchedule **POST** requests are allowed. The parameter takes a space separated list of single host IP addresses and/or Classless Inter-Domain Routing (CIDR) entries. A CIDR entry is a base IP address followed by a number indicating how many upper bits to mask. For example, specifying a CIDR of **10.20.30.0/24** matches all addresses from the IP address **10.20.30.0** to the IP address **10.20.30.255**. An entry of **0.0.0.0/0** allows all requests. The default setting for the parameter is an empty list, which denies all requests except for those from **localhost**.

You can also secure the iSchedule port. For more information, see *Calendar Server Security Guide*.

Enabling CalDAV and CardDAV Autodiscovery

Calendar Server supports the **.well-known** URI concept to access the Principal URI without having to specify the entire URI. That is, access to **/** (root) or **/.well-known/caldav/** is redirected to the **/dav/principals/** URI.

To take full advantage of this functionality, see the IETF Tools website for information about how to configure a corresponding DNS record at:

<http://tools.ietf.org/html/rfc6764>

Configuring CalDAV Clients

This chapter describes how to configure Apple and Lightning clients to communicate with Oracle Communications Calendar Server.

Topics:

- [Prerequisites](#)
- [Configuring CalDAV Clients](#)
- [Using the iPhone Configuration Utility](#)
- [Exporting and Importing Calendars in Thunderbird Lightning](#)
- [Client Issues](#)

Prerequisites

1. Users need to be provisioned in Directory Server. For information on provisioning users, see "[Overview of Provisioning Calendar Server](#)." For information on how Calendar Server works with Directory Server, see *Calendar Server Concepts*.
2. Obtain the following information on your Calendar Server 7 deployment:
 - GlassFish Server host name and port where Calendar Server 7 is installed
 - user identifier (email address or uid@domain)

Configuring CalDAV Clients

This section contains the following tasks:

- [Configuring Apple iCal for Calendar Server](#)
- [Configuring Apple iPhone for Calendar Server](#)
- [Configuring Lightning 1.0 beta2 for Calendar Server](#)
- [Configuring Lightning 1.0 beta for Calendar Server](#)
- [Configuring Lightning 0.9 for Calendar Server](#)
- [Accessing a Shared Calendar](#)
- [Configuring a CalDAV Account by Using Non-standard or Demo Settings](#)
- [Configuring Android for Calendar Server](#)

Configuring Apple iCal for Calendar Server

To configure Apple iCal:

1. Launch iCal.
2. Choose **Preferences** from the iCal menu and click **Accounts**.
3. To add a new account, click the **Add (+)** button.
4. Choose **CalDAV** from the Account type menu.
5. Enter your user name, password, and server name, then click **Create**.
The client indicates "Verifying CalDAV account", then "Account verified."
6. You can now use the Calendar application.

Configuring Apple iPhone for Calendar Server

To configure Apple iPhone:

1. Navigate to the Mail, Contacts, and Calendar settings menu.
2. Select **Add Account**.
3. Select **Other**.
4. Select **Add CalDAV Account**.
5. Enter your Server address, User Name, and Password.
6. Tap **Next**.
The client indicates "Verifying CalDAV account", then "Account verified."
7. You can now use the Calendar application.

Configuring Lightning 1.0 beta2 for Calendar Server

These instructions assume that you have already installed at least Thunderbird 3.1.x on your client machine.

To configure Lightning 1.0 beta 2:

1. Download Lightning 1.0 beta2 to your client machine from the Lightning Calendar web site at:
<http://www.mozilla.org/projects/calendar/releases/lightning1.0b2.html>
2. Download, but do not execute, the appropriate binary for your client platform. If the downloaded file is a zip file, unzip it.
3. Create a Thunderbird profile as follows:
 - a. In Mozilla Thunderbird, choose **Add Ons** or **Extensions** from the Tools menu, depending on the version of Thunderbird.
 - b. Click the **Install** button.
A file chooser is displayed.
 - c. Navigate to the previously downloaded (and perhaps unzipped) **.XPI** file, select it, and click **OK**.
 - d. In the Software Installation dialog box, click **Install Now**.
 - e. Click **Restart Thunderbird**.

- f. Click **Calendar** in the Events and Tasks menu item at the top of the Thunderbird UI.
- g. From the File menu, choose **New**, then **Calendar**.
If this selection is grayed out, you might need first to open the default calendar in Thunderbird.
- h. Choose **On the Network**.
- i. Choose **CalDAV**.
- j. Enter the URL of the calendar, for example:
`https://example.com/dav/home/jsmith@example.com/calendar/`
- k. Enter your name, choose a color scheme, choose to set alarms or not, and select your email address.
- l. Enter your user name and password for the CalDAV server.
A confirmation dialog box informs you that your calendar has been created.
- m. Click **Finish**.
The new calendar appears in the listing of calendars on the left side of the Thunderbird UI.

Configuring Lightning 1.0 beta for Calendar Server

These instructions assume that you have already installed at least Mozilla Thunderbird 2.0.0.x on your client machine.

To configure Lightning 1.0 beta:

1. Download Lightning 1.0 beta 1 to your client machine from the Lightning Calendar web site at:
<http://www.mozilla.org/projects/calendar/lightning/download.html>
2. Download, but do not execute, the appropriate binary for your client platform.
If the downloaded file is a zip file, unzip it.
3. Create a Thunderbird profile as follows:
 - a. In Mozilla Thunderbird, choose **Add Ons** or **Extensions** from the Tools menu, depending on the version of Thunderbird.
 - b. Click the **Install** button.
A file chooser is displayed.
 - c. Navigate to the previously downloaded (and perhaps unzipped) **.XPI** file, select it, and click **OK**.
 - d. In the Software Installation dialog box, click **Install Now**.
 - e. Click **Restart Thunderbird**.
 - f. Click the Calendar icon in the lower left corner of the Thunderbird UI.
 - g. From the File menu, choose **New** then **Calendar**.
If this selection is grayed out, you might need first to open the default calendar in Thunderbird.
 - h. Choose **On the Network**.

- i. Choose **CalDAV**.
 - j. Enter the URL of the calendar, for example:
`http://example.com:3080/dav/home/jsmith@example.com/calendar/`
 In this example, the default URI of / was used during initial configuration.
 The general format is:
`http://glassfish_host:glassfish_port/baseuri/dav/home/email_address/calendar/`
 - k. Enter your name, choose a color scheme, choose to set alarms or not, and select your email address.
 - l. Enter your user name and password for the CalDAV server.
 A confirmation dialog box informs you that your calendar has been created.
 - m. Click **Finish**.
 The new calendar appears in the listing of calendars on the left side of the Thunderbird UI.
4. Lightning 1.0 has CalDAV scheduling capability but it is turned off by default. Turn on the following configuration preferences for CalDAV scheduling to work by using the Config Editor.
- **calendar.itip.notify**
 - **calendar.caldav.sched.enabled**
- Windows: From the Tools menu, selection **Options**, then **Advanced**, then **Config Editor**.
- UNIX: From the Edit menu, select **Preferences**, **Advanced**, then **General**.

Configuring Lightning 0.9 for Calendar Server

These instructions assume that you have already installed at least Mozilla Thunderbird 2.0.0.x on your client machine.

To configure Lightning 0.9:

1. Download Lightning 0.9 to your client machine from the Lightning Calendar web site at:
<http://www.mozilla.org/projects/calendar/releases/lightning0.9.html>
2. Download, but do not execute, the appropriate binary for your client platform.
 If the downloaded file is a zip file, unzip it.
3. Create a Thunderbird profile as follows:
 - a. In Mozilla Thunderbird, choose **Add Ons** or **Extensions** from the Tools menu, depending on the version of Thunderbird.
 - b. Click the **Install** button.
 A file chooser is displayed.
 - c. Navigate to the previously downloaded (and perhaps unzipped) **.XPI** file, select it, and click OK.
 - d. In the Software Installation dialog box, click **Install Now**.
 - e. Click **Restart Thunderbird**.

- f. Click the Calendar icon in the lower left corner of the Thunderbird UI.
 - g. From the File menu, choose **New** then **Calendar**.
If this selection is grayed out, you might need first to open the default calendar in Thunderbird.
 - h. Choose **On the Network**.
 - i. Choose **CalDAV**.
 - j. Enter the URL of the calendar, for example:
`http://example.com:3080/dav/home/jsmith@example.com/calendar/`
In this example, the default URI of / was used during initial configuration.
The general format is:
`http://glassfish_host:glassfish_port/baseuri/dav/home/email_address/calendar/`
 - k. Enter your name, choose a color scheme, choose to set alarms or not, and select your email address.
 - l. Enter your user name and password for the CalDAV server.
A confirmation dialog box informs you that your calendar has been created.
 - m. Click **Finish**.
The new calendar appears in the listing of calendars on the left side of the Thunderbird UI.
4. Lightning 0.9 has CalDAV scheduling capability but it is turned off by default. Turn on the following configuration preferences for CalDAV scheduling to work by using the Config Editor.
 - **calendar.itip.notify**
 - **calendar.caldav.sched.enabled**

Windows: From the Tools menu, selection **Options**, then **Advanced**, then **Config Editor**.

UNIX: From the Edit menu, select **Preferences**, **Advanced**, then **General**.

Accessing a Shared Calendar

The following steps describe how user A can access user B's calendar:

1. In Convergence, user B grants user A read, read/write, or owner privilege through the Share panel.
2. Alternately, an administrator can use the **davadmin calendar** command to set the calendar ACLs.
3. To view the newly shared calendar of user B, user A creates a new calendar or account on the calendar client.
 - Lightning: User A enters user B's calendar URL
 - Apple iCal: User A enters user B's principal URL (in the Server Option of the Apple iCal Account Creation panel)

Configuring a CalDAV Account by Using Non-standard or Demo Settings

The previous information assumes settings for valid for a production system but not for a demo server, for example:

- Use of standard ports (443 or 80)
- SSL is the default
- Account URL follows a fixed pattern: **http(s)://server_name/principals/users/username/**

Demo servers usually run on non-standard port numbers and they do not always own the full namespace, leading to account URLs (actually principal URL) that look more like the following one for iCal:

```
http://caldav.example.com:3080/demo/dav/principals/username/
```

Similarly, a demo Lightning URL might resemble the following:

```
http://caldav.example.com:3080/demo/dav/home/username/calendar/
```

For information on configuring the default context URI for a Calendar Server deployment, see *Calendar Server Installation and Configuration Guide*.

iOS 3.x and 4.x Non-standard Configuration

Typing the previous kind of URL can be very tedious and error prone, especially given that the iPhone advanced configuration panel offers just a tiny text box. The following procedure simplify the configuration process, assuming that you have a mail account already configured.

1. From your usual desktop client, email the principal URL to yourself.
Check that the URL is valid (by using a regular browser) before sending it.
The principal URL varies across servers. It is the same that you might have configured if you are using the Apple iCal client.
2. Copy the URL from the iPhone Mail App.
 - a. From the iPhone Mail App, open the email.
 - b. Press and hold on the URL in the message. You should be asked whether you want to open or copy the link.
 - c. Select **copy**.
3. Navigate to the CalDAV account creation panel.
4. Enter the server information.
 - a. Tap on the **Server** field.
A Paste button should appear on top of the text field.
 - b. Press **Paste**.
The full URL is shown. The client accepts a full URL in the server name field.
5. Enter the user name and password.
 - a. Go to the User Name field. The full principal URL is replaced by the server name only, which is to be expected.
 - b. Enter your password and tap **Next**.
The client indicates "Verifying CalDAV account", then "Account verified."

6. You can now use the Calendar application.

Apple iCal Non-standard Configuration

1. Launch iCal.
2. Choose **Preferences** from the iCal menu and click **Accounts**.
3. To add a new account, click the **Add (+)** button.
4. Choose **CalDAV** from the Account type menu.
5. Enter your user name and password.
6. In Server Address, enter the principal URL, for example:
`http://caldav.example.com:3080/demo/dav/principals/username/`
7. Click **Create**.

You can now use the Calendar application.

For the regular server configuration, you would click the server options and enter the principal URI, for example:

`http://caldav.example.com/dav/principals/username/`

Configuring Android for Calendar Server

Download and install the Android CalDAV-Sync client to synchronize events and tasks, and the Android task app to synchronize all tasks, from the Android Apps web site at:

<https://play.google.com/store/apps/details?id=org.dmfs.caldav.lib&hl=en>

Note the following limitations:

- When you create an event with an attachment, the event is created without the server storing the attachment.
- You cannot see attachments added to events by other clients.

Using the iPhone Configuration Utility

Apple provides the Apple Configurator to install and manage installation profiles. Enterprises might find this utility helpful to manage their end user accounts. For more information, see the Apple Configurator web page at:

<https://itunes.apple.com/us/app/apple-configurator/id434433123?mt=12>

Exporting and Importing Calendars in Thunderbird Lightning

This section contains the following tasks:

- [Exporting a Calendar](#)
- [Importing a Calendar](#)

Exporting a Calendar

To export a calendar:

1. Open any Calendar view.

2. Choose **Export Calendar** from the File menu.
3. Select the calendar.
4. When prompted to save the file, use the iCalendar format (the default is HTML).

Importing a Calendar

To import a calendar:

1. Open any Calendar view.
2. Choose **Import Calendar** from the File menu.
3. Select the exported file.

Client Issues

Topics in this section:

- [Troubleshooting CalDAV Clients](#)
- [Connector for Microsoft Outlook and Event Time Modifications](#)

Troubleshooting CalDAV Clients

For information on troubleshooting issues with Lightning and Apple clients, see ["Troubleshooting CalDAV Clients."](#)

Connector for Microsoft Outlook and Event Time Modifications

If you use Connector for Microsoft Outlook to create or modify the time of an event, and later make a change to the event time by using Convergence, the event "jumps" to a new time. In some cases, the event appears to "vanish" but in reality it "jumps" to the following day. Currently, there is no workaround.

To reproduce:

1. Log into Convergence and create an event.
2. Log into Connector for Outlook and move the event.
3. Log into Convergence and make sure event is moved by refreshing.
4. Move the event again, but this time on Convergence.

The event "jumps" to a new time.

Configuring and Managing Virus Scanning

This chapter describes how to configure and manage virus scanning for Oracle Communications Calendar Server.

Topics:

- [About Calendar Server and Virus Scanning](#)
- [Calendar Server Virus Scanning Architecture](#)
- [Configuring Calendar Server Virus Scanning](#)
- [Example MTA Configuration for Calendar Server Virus Scanning](#)
- [Calendar Server Virus Scan Command-line Utility](#)
- [Virus Scan Logging](#)
- [MTA Logging](#)

About Calendar Server and Virus Scanning

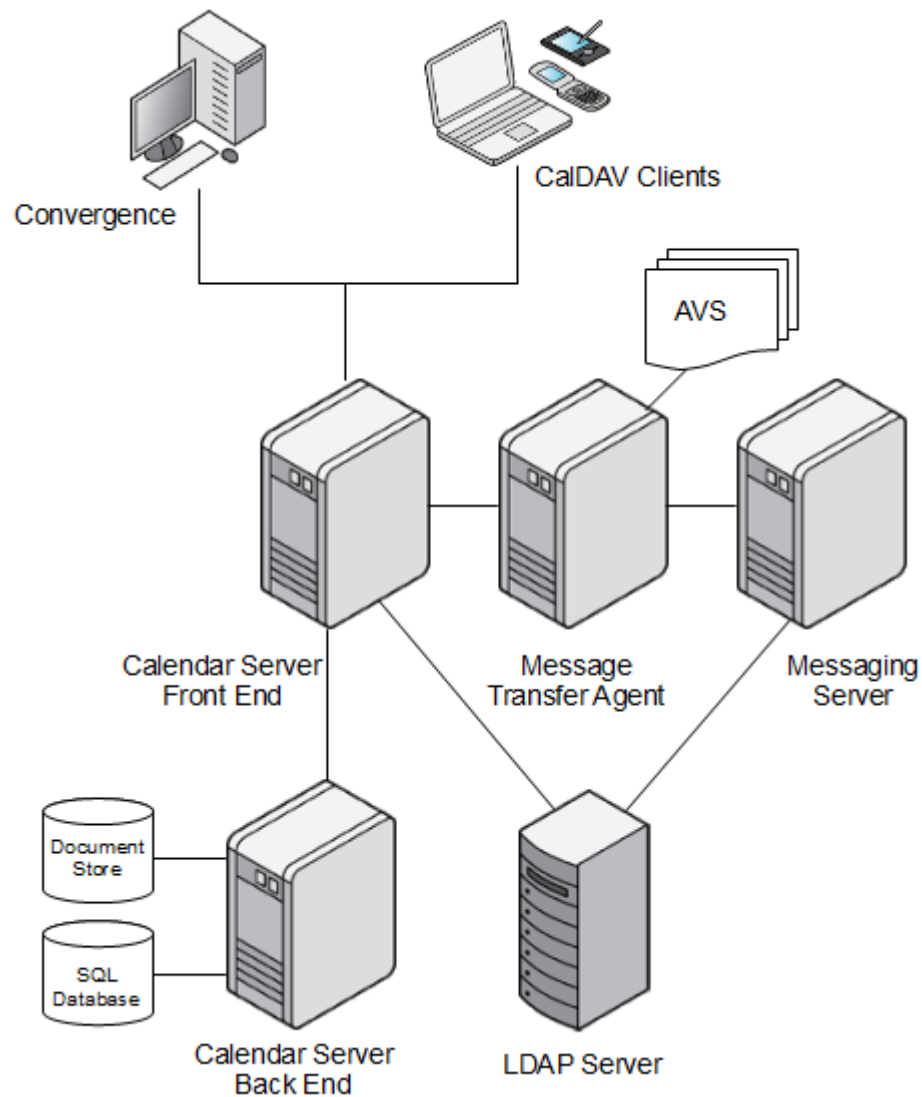
To enhance security within your deployment, you can use Calendar Server attachments virus scanning. Calendar Server virus scanning can examine calendar attachments in a "real-time" mode to test and optionally reject incoming infected data. You can also choose to scan and optionally delete infected existing data "on-demand."

Virus scanning is not performed by Calendar Server itself. Instead, you configure an Oracle Communications Messaging Server's Message Transfer Agent (MTA) to filter the calendar data. You can configure Calendar Server to share an existing MTA that has already been configured for Messaging Server virus scanning. Or, if you prefer, you can configure a standalone MTA that functions only for Calendar Server virus scanning.

Calendar Server reports all virus scanning activities, as well as detected viruses, in its log file, for both real-time and on-demand scanning.

Calendar Server Virus Scanning Architecture

[Figure 7-1](#) depicts the Calendar Server virus scanning logical architecture.

Figure 7-1 Calendar Server Virus Scanning Architecture

The following information describes how a calendar attachment is scanned for viruses.

1. A calendar client submits a calendar event and attachment to Calendar Server.
2. Calendar Server receives the event and attachment then packages the attachment as an email message for the MTA that has been configured to scan calendar attachments.
3. Calendar Server sends the email message containing the attachment by using the SMTP protocol to the configured MTA.
4. Calendar Server keeps the connection to the MTA open as it awaits the response from the MTA. During this time, the calendar client is also waiting for the MTA to reply back to Calendar Server with its verdict.
5. The Calendar Server function responsible for connecting to the MTA keeps the attachment.

Later, after the scan has completed, the function either stores the attachment (and possibly the event) in the Calendar Server document store or aborts if the MTA finds a virus. See step 7 for details.

6. The MTA receives the package on a specific channel that is configured for a **sourcespamfilter**, which in turn is linked to an Anti-Virus Scanner (AVS).
You actually define a **sourcespamfilter n** , where n is an integer in a given range, to define one of the possible **sourcespamfilters** on the system.
7. The AVS scans the package.
 - a. If the AVS detects a virus, the MTA refuses the message and replies with a virus positive message to Calendar Server over the open connection.
 - b. If the AVS does not detect a virus, the MTA uses a Messaging Server rewrite rule to send the package to the bitbucket channel and discard it. Calendar Server logs an error either when it detect a virus or the AVS is not working.
8. Once it is notified by the MTA, Calendar Server decides if it can continue processing the calendaring request normally or abort. If the **davcore.viruscan.onlinevirusaction** parameter remains unset (is using the default value) or is set to **reject**, the submission is rejected. The client receives the reply **HttpStatus.FORBIDDEN** (Virus Detected in Attachment). Otherwise if **davcore.viruscan.onlinevirusaction** is set to "keep," the attachment is accepted. The **davcore.viruscan.onlinefailureaction** parameter works similarly, except that the default action is "keep."

Configuring Calendar Server Virus Scanning

The high-level steps to prepare your deployment to perform virus scanning for Calendar Server include:

1. (Optional) Installing the Messaging Server MTA
2. Configuring the Messaging Server MTA
3. Configuring the MTA for the virus scan filter
4. Creating the incoming SMTP port and channel for Calendar Server virus scan
5. Configuring the rewrite rule to discard Calendar Server data after scanning
6. Configuring Calendar Server virus scanning parameters

The following sections describe configuring Messaging Server and Calendar Server in more detail.

Topics in this section:

- [Configuring the MTA](#)
- [Configuring the Messaging Server MTA for the Virus Spam Filter](#)
- [Configuring Calendar Server for Virus Scanning](#)

Configuring the MTA

It is possible that your deployment has already deployed Messaging Server and an MTA to perform email virus scanning. If so, you can reuse this existing MTA to also scan calendar attachments for viruses. If this is not the case, you can install and configure a stand alone MTA.

Prerequisite: Calendar Server virus scanning requires at least Messaging Server 7 Update 4 patch 23.

Installing a Standalone Message Transfer Agent

When installing a standalone MTA for Calendar Server virus scanning, be sure to use meaningful values for administrator postmaster, mail domain, and other configuration settings. If you use values that are not meaningful to your deployment, errors can result.

The general steps to install an MTA include:

1. Installing the Messaging Server software
2. Running the Messaging Server **configure** script
3. Disabling the Message Store and Webmail Server

For details, see the topic on installing a Message Transfer Agent in *Unified Communications Suite Installation Guide*.

When configuring Messaging Server, the "configure" step requires a valid Directory Server host that is used to include configuration data such as the default mail domain and messaging administrator account. The Directory Server host that you specify needs to be available during virus scanning operations. However, due to MTA caching of LDAP data, this host is not heavily utilized.

Configuring the Messaging Server MTA for the Virus Spam Filter

The MTA itself does not check for viruses. You configure the MTA to communicate with the desired virus scanning software, also referred to as the AVS. For instructions, refer to the topic on integrating spam and virus filtering programs Into Messaging Server in **Messaging Server Administration Guide**.

The filter should use a Sieve rule to "refuse" the message from Calendar Server if a virus is found by the virus scanning software. The Sieve rule returns **FilterVerdictPositive**. Calendar Server checks SMTP return values for this exact string, which is defined in the **option.dat** file. See "[Example MTA Configuration for Calendar Server Virus Scanning](#)" for more information.

Note: You configure the MTA to perform a Sieve refuse action if there is a virus, which returns an SMTP code 5xy plus the MTA-configured string **FilterVerdictPositive**. Calendar Server responds to the target string, where other errors are considered failures in service.

Creating an Incoming SMTP Channel That Uses the Filter

You create a new incoming SMTP port in Messaging Server's **dispatcher.cnf** file, strictly for Calendar Server virus scanning use. In this way, Calendar Server traffic is tracked. In addition, a separate SMTP port makes it easier to destroy all data being scanned. You associate this incoming SMTP port with a new MTA channel in the **imta.cnf** file. Finally, you configure the receiving channel to use the **sourcespamfiltern** that is configured with the desired virus scan software, so that incoming calendar data is tested. For instructions, refer to *Messaging Server Administration Guide*.

Configuring the Rewrite Rule to Detect Calendar Data and Discard it After Scanning

Calendar Server sends the attachment data as an email with a user recipient email address. You configure the MTA to detect the chosen email address. The email address is set up to use the MTA's host name and domain, so that the MTA does not need to perform a lookup for the domain. The user email address itself is not significant since

incoming data is not actually delivered. See the topic on rewrite rules and channels in *Messaging Server Administration Guide* for more information.

Configuring Calendar Server for Virus Scanning

You use the **davadmin** command to configure Calendar Server parameters for virus scanning. Some parameters are required. Others are optional.

1. Configure the following required parameters:

- **davcore.virusscan.emailaddress**
- **davcore.virusscan.host**
- **davcore.virusscan.port**
- **davcore.virusscan.onlineenable**
- **davcore.virusscan.onlinevirusaction**

The syntax for the **davadmin** command in this instance is as follows:

```
davadmin config modify -u adminID -o parameter -v value
```

For example:

```
./davadmin config modify -u admin -o davcore.virusscan.emailaddress -v myvirususer@mymachine.example.com
```

The email address' domain must match the MTA's domain. The user name itself is not significant.

2. Configure optional parameters.

See "[Calendar Server Configuration Parameters](#)" for more information.

Example MTA Configuration for Calendar Server Virus Scanning

This example describes how to configure a Messaging Server MTA for Calendar Server virus scanning.

1. Install Messaging Server software and configure an MTA.

If necessary, use the "Installation Scenario for Message Transfer Agent" instructions in *Unified Communications Suite Installation Guide*.

In this example, the following values are used:

The Fully Qualified Host Name is required: mymachine.example.com

The LDAP directory server the MTA will use: myldap.example.com

The LDAP port: 389

The LDAP Bind user: cn=Directory Manager

The LDAP password: mypassword

The system user name and group: mailsrv mailsrv

The default mail domain: example.com

The postmaster email address admin@example.com

The password for messaging admin: mypassword

2. Disable the Message Store and Webmail server.

```
./configutil -o local.store.enable -v 0
```

```
./configutil -o service.http.enable -v 0
```

3. Start the MTA.

```
./start-msg
```

4. Configure the MTA for the virus scan filter.

This example uses ClamAV for the virus scanning software package to work with the MTA.

- a. Create the ClamAV configuration file, **clamav.mtaconf**, in the **/opt/sun/comms/messaging64/lib/** directory.
- b. Make sure that **clamav.mtaconf** file contains the following information:

```
HOST=localhost  
PORT=3310
```

5. Edit the **clamd.conf** file to contain the follow information:

On Solaris: **/opt/ClamAV/etc/clamd.conf**

On Linux: **/etc/clamd.conf**

```
LogFile /tmp/clamd.log  
LogTime yes  
LogVerbose yes  
FixStaleSocket yes  
TCPSocket 3310  
TCPAddr 127.0.0.1
```

6. (Solaris only) Set the path to the ClamAV **bin** directory.

```
setenv PATH /opt/ClamAV/bin:$PATH
```

7. Become **root** and start ClamAV.

```
su -  
cd /opt/ClamAV/sbin/  
clamd session
```

8. Create a "filter" on the MTA that serves as the connection to the ClamAV server.

Add the following information to the **option.dat** file in the **config** directory:

```
SPAMFILTER1_LIBRARY=/opt/sun/comms/messaging64/lib/libclamav.so  
SPAMFILTER1_CONFIG_FILE=/opt/sun/comms/messaging64/lib/clamav.mtaconf  
SPAMFILTER1_NULL_ACTION=data:,require ["reject","ereject","refuse"]; refuse  
"FilterVerdictPositive";
```

This example uses filter 1, hence many of the keywords have "1" in them. For example, **SPAMFILTER1_LIBRARY** is a registered MTA keyword. The MTA needs to know where to locate the ClamAV configuration file. It also needs to know the location for the already existing ClamAV client library that the MTA provides. This is the MTA's specific code that knows how to communicate to ClamAV servers. Finally, an "action" is needed to tell the MTA what to do depending on what is returned by clamAV.

By using this information for spam filter 1, the MTA knows where to find the existing MTA library, where to find the configuration file for communicating to ClamAV, and how to handle the response back from ClamAV. The "action" is a sieve string that explains that if there is a virus detected, then "refuse" the SMTP submission with the **FilterVerdictPositive** string. This is the string that is sent back to the Calendar server, and must be exact. So far, this configuration does not attach the filter to any incoming data. But now that this spam filter is configured, it can be used in the channel definitions.

9. Create the incoming SMTP channel which uses the filter.
 - a. Create an SMTP port the Calendar Server uses.
 - b. A virus scan port and channel called **tcp_vscan** can be created by adding code to the **dispatcher.cnf** file:

```
!
! Virus Scan Port
!
[SERVICE=SMTP_VSCAN]
PORT=3025
IMAGE=IMTA_BIN:tcp_smtp_server
LOGFILE=IMTA_LOG:tcp_vscan_server.log
PARAMETER=CHANNEL=tcp_vscan
STACKSIZE=2048000
! Uncomment the following line and set INTERFACE_ADDRESS to an appropriate
! host IP (dotted quad) if the dispatcher needs to listen on a specific
! interface (e.g. in a HA environment).
!INTERFACE_ADDRESS=
```

10. Create a matching channel definition in the channel definition configuration. Rewrite rules and channel definitions are located in the **imta.cnf** file. Add this channel in the channel area, paying strict attention to syntax rules described in the MTA documentation in *Messaging Server Administration Guide*.

```
!
! tcp_vscan
tcp_vscan smtp sourcespamfilter1 missingrecipientpolicy 6 pool SMTP_POOL
tcp_vscan-daemon
```

Note: This example uses **sourcespamfilter1**, which is the spam filter already configured. All incoming SMTP submissions on this port and channel are submitted to ClamAV, and if a virus is found, Calendar Server receives the correct message.

11. Configure the rewrite rule to send calendar data to be discarded after scanning.

With the virus scan channel and virus spam filter configured, Calendar Server receives the proper return values from ClamAV. However, the incoming message needs to be handled by the MTA. A rewrite rule is required to send it to be destroyed in the bitbucket channel. Add the following rewrite rule to the **imta.cnf** file just before the rule "Rules to select local users."

```
! Avoid all lookups and just force to bitbucket channel for messages
! coming in the tcp_vscan channel:
$* $U%$H@bitbucket-daemon$Mtcp_vscan
```

This rewrite rule checks for email coming in on the **tcp_vscan port** and sends it to the bitbucket (where it is destroyed).

12. Configure Calendar Server's virus scan email address to be in the MTA's domain. The user name is not significant.

Set **davcore.virusscan.emailaddress** to **joe@mymachine.example.com**.

13. Recompile the MTA configuration.

```
./imsimta cnbuild
./imsimta restart
```

Summary:

1. Calendar Server sends data to be scanned to the Messaging Server MTA by using an email address of **joe@mymachine.example.com**.
2. This is done on the specified port configured in the **dispatcher.cnf** file.
3. This email arrives at the MTA on the **tcp_vscan** channel, and is subjected to **sourcespamfilter1**, which is tied to ClamAV through the configuration in the **option.dat** file.
4. If virus scanning software detects a virus, a refuse action is sent back through SMTP to Calendar Server with the string **FilterVerdictPositive**.
5. If the virus scanning software does not detect a virus, the incoming message is subjected to rewrite rules that send it to the bitbucket for deletion.

The MTA communicates to LDAP to look up **example.com**, but caches LDAP's response so it does not make this call often.

For more information on the virus scanning configuring parameters, see ["Calendar Server Configuration Parameters."](#)

Calendar Server Configuration Examples

- To set the MTA host:

```
davadmin config modify -u admin -o "davcore.virusscan.host" -v  
"myhost.example.com"
```

- To set the SMTP port:

```
davadmin config modify -u admin -o "davcore.virusscan.port" -v "3025"
```

- To set the email address:

```
davadmin config modify -u admin -o "davcore.virusscan.emailaddress" -v  
"myvirususer@mymachine.example.com"
```

- To set the timeout value:

```
davadmin config modify -u admin -o "davcore.virusscan.timeout" -v "1000"
```

- To enable scanning on incoming data:

```
davadmin config modify -u admin -o "davcore.virusscan.onlineenable" -v "true"
```

- To reject viruses discovered in attachments by the MTA:

```
davadmin config modify -u admin -o "davcore.virusscan.onlinevirusaction" -v  
"reject"
```

- To reject viruses if the AVS is not functioning or is not responding:

```
davadmin config modify -u admin -o "davcore.virusscan.onlinefailureaction" -v  
"reject"
```

- To automatically delete a virus when scanning.

```
davadmin config modify -u admin -o "davcore.virusscan.clivirusaction" -v  
"delete"
```

Calendar Server Virus Scan Command-line Utility

Use the **davadmin vscan** command to perform virus scanning operations. The **davadmin vscan** command must be followed by the **scan** action. For more information, see "[Calendar Server Command-Line Utilities](#)."

Virus Scan Logging

Virus scan activity for both online and CLI is logged in the calendar server's "scan" log. Found virus are reported in the log. Actions taken against viruses are reported if any actions are configured. Owing components that are found to reference data that is found to be a virus are reported. The time just before a **davadmin** scan is started is printed at the end of a scan, in case this time may be useful with the **-T** option in future scans.

Because the **davadmin scan** command runs on the GlassFish Server (and not the **davadmin** client), most useful information is printed in the Calendar Server's "scan" log, not always in the standard output of the **davadmin command**. This also provides a central repository for all historical virus scan related information and tracking.

MTA Logging

See the MTA documentation in *Messaging Server Administration Guide* for logging information.

To view and test channel traffic, add the keyword **logging** to the **defaults** channel in the **imta.cnf** file. Add **LOG_CONNECTION=255** and **LOG_FILTER=1** to the **option.dat** file. Use the MTA documentation to interpret channel operations such as "E" enqueue and "D" dequeue, "O" open connection, "C" close connection. View messages coming in on the **tcp_vscan**, and dequeue onto the **bitbucket** channel.

Using Calendar Server Notifications

This chapter describes the Oracle Communications Calendar Server notification architecture, how to enable notifications, the different types of notifications, and how to customize notifications.

Topics:

- [Overview of Notification Architecture](#)
- [About Reminders \(Alarms\)](#)
- [About Server Email Notifications](#)
- [Managing Notification Templates](#)
- [Writing a Java Messaging Service Consumer](#)
- [Managing Calendar Server Java Messaging Server Destinations](#)
- [Presence Notifications](#)

Overview of Notification Architecture

Calendar Server is capable of generating notifications for any change to the calendar data in the database, or for some preset trigger. Notifications are published as Java Message Service (JMS) messages. Calendar Server also includes a JMS consumer program that consumes the JMS notifications and sends email messages to end users. One type of such end user email notification, reminders, (sometimes called alarms), are set by end-users for themselves, so that they are notified about their upcoming events and todos. Another type of notification is sent by the server when a user, different than the one being notified, makes a change to the calendar database, for example, by modifying an event invitation, granting a calendar permission, and so on.

Calendar Server notification services use a publish/subscribe paradigm. Calendar Server publishes messages, in this case, notifications. Receiving clients (the subscribers) receive only those messages that they are interested in.

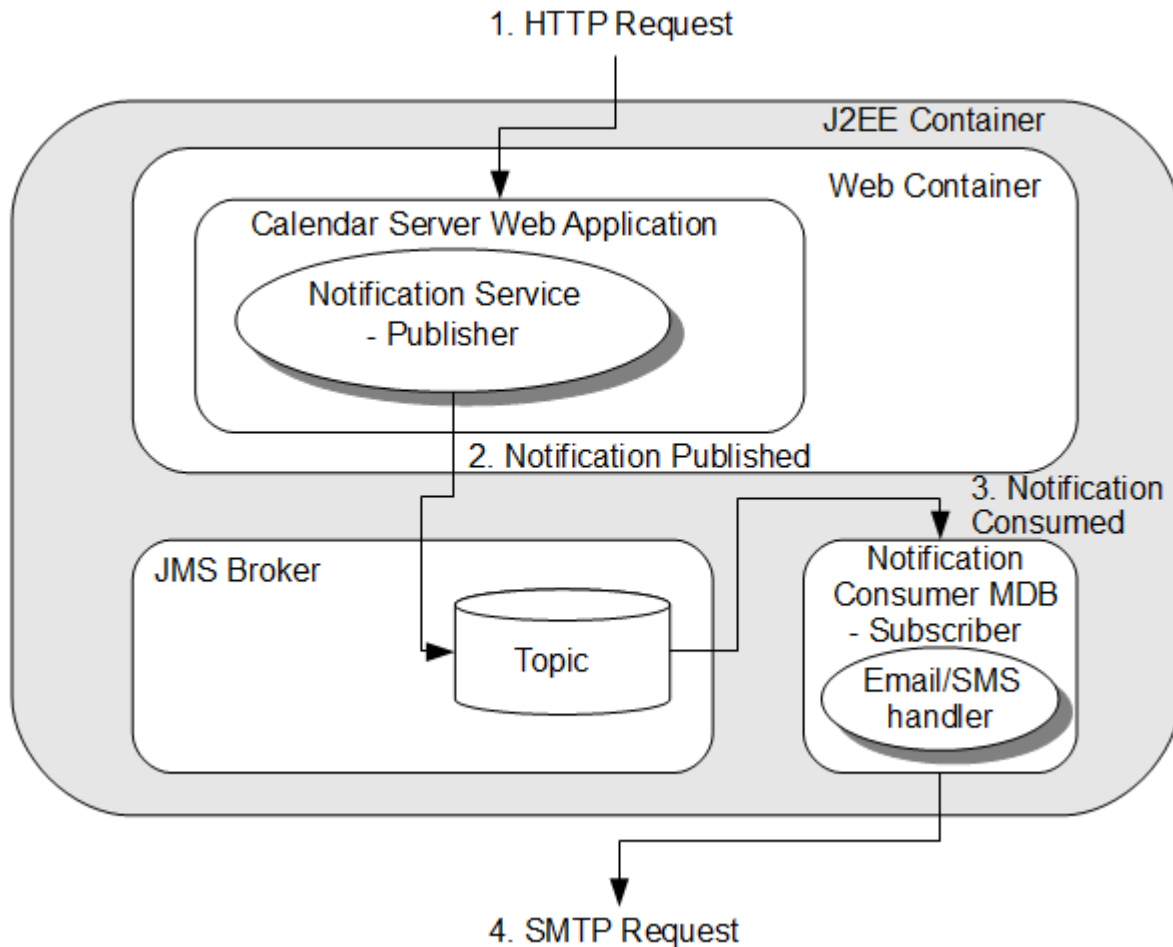
Calendar Server utilizes the built-in Java Messaging Service (JMS) in Oracle GlassFish Server to communicate calendar data changes and calendar alarm triggers. Calendar Server bundles a consumer program that "consumes" this information and sends email for certain subset of the notifications as detailed in "[Notification Types](#)." For more information, see the JMS website at:

<http://java.sun.com/products/jms/>

Figure 8–1 shows that the Calendar Server notification service consists of two major components, the *Notification Service* and *Notification Consumer*. The Notification Service component is part of the Calendar Server itself, and is the publisher that posts

messages of a pre-configured JMS topic managed by the JMS provider. The Notification Consumer component is the subscriber or the message consumer of that JMS topic.

Figure 8-1 Calendar Server 7 Notifications Services Architecture



The Notification Service component provides interfaces for Calendar Server to publish JMS messages to a specific JMS topic (**DavNotificationTopic**) of the JMS broker. The Notification Service component is part of the main Calendar Server servlet that is deployed in the GlassFish Server web container. The Notification Consumer component listens on the JMS bus for the specific topic (**DavNotificationTopic**) notification messages, consumes the messages, and sends notification email to recipients, if applicable. The consumer checks the notification type and other instructions provided in the JMS message to determine what action is to be taken. The Notification Consumer component message-driven bean (MDB) runs in the GlassFish Server J2EE container. The consumer MDB is deployed in **EMBEDDED** mode, and thus is running in the same JVM of the J2EE container.

You can choose to write your own customized Notification Consumer programs. See ["Writing a Java Messaging Service Consumer."](#)

About Reminders (Alarms)

Calendar Server sends out email for upcoming events and tasks if the owners of the events and tasks have set an email or SMS reminder. (Convergence users can enable default reminders.) The information is stored along with the event or task in the standard **VALARM** format as specified in RFC 5545 with action set to **EMAIL**. The server maintains a queue of these alarms and when the right time arrives, it posts the relevant information to the JMS bus with the notification type set to **ALARM**. The notification consumer fills in the right alarm template file based on the instructions in the JMS payload and the email is sent. For reminders to work, you only need to set the **notification.dav.enablejmsnotif** parameter to **true**, as well as the correct SMTP configuration settings.

Calendar Server supports the Alarm-Agent Property. This property specifies whether a client, server, both client and server, or none, is responsible for processing an alarm when it is triggered. This is in accordance with the Extended VALARM draft. To view the draft, see the IETF website at:

<http://tools.ietf.org/html/draft-daboo-valarm-extensions-04>

For details on how to set alarms by using the store commands in the WCAP protocol, see the *Calendar Server WCAP Developer's Guide*.

About Server Email Notifications

Server notifications are used to notify users mostly about changes to their calendars due to actions by other users, including event or task invitations, granting permission to a calendar, and so on. To enable email notifications at a server level, both the **notification.dav.enablejmsnotif** and **notification.dav.enableemailnotif** configuration parameters must be set to **true**. In addition, notification should be enabled on a per account basis. In case of event and task invitations or responses, to include the actual event or task in standard ics format, the **notification.dav.enableimipmailnotif** needs to be enabled as well (set to **true**).

Calendar Server supports RFC 6047 and sends iMIP invitations and responses to external users as a consequence. External users reside either on a different Calendar Server deployment administered by a separate group, or on an outside calendaring system, such as Exchange, Google Calendar, and so on. However, this is a separate feature from the notifications that are explained in this chapter. Calendar Server uses the **notification.dav.enableimip** configuration parameter to control iMIP notifications. Both iMIP and server email notifications use the **notification.dav.smtp*** configuration parameters to configure the SMTP server to use.

In addition to external users, internal users that have their status set to **inactive** can also be configured to receive iMIP invitations. The **davcore.scheduling.rejectinactiverecipients** parameter enables and disables this capability. If this value is set to **false**, internal users whose status attribute (**icsStatus** by default) is set to **inactive** in the LDAP directory receive iMIP invitations just like external users. For users whose status is set to **deleted** or **inactive**, no iMIP invitations are sent under any circumstances.

Enabling Calendar Server Notifications

[Table 8–1](#) describes the Calendar Server notifications that are controlled by the configuration parameters.

Table 8–1 Notification Configuration Parameters

Parameter	Description
notification.dav.enableemailnotif	Controls server-wide email notification. When this parameter is set to true , Calendar Server sends email notifications for new event, task, calendar creation, and access changes, if end users choose to receive them. End users can choose to receive notifications either by enabling their own account through Convergence or by requesting that an administrator do so by using the davadmin command. These notifications are text emails sent to users for actions that have already been recorded in their calendars. If set to false , server-wide email notification is disabled.
notification.dav.enablejmsnotif	Controls server-wide JMS notification. When set to true , Calendar Server publishes notifications to the JMS bus. This parameter must be set to true for any notification to work.
notification.dav.enableimipmailnotif	Controls server-wide inclusion of actual event/task ical content in email notification. When this parameter is set to true , iCal content is included in the server-wide JMS notification email sent to users on the internal deployment. By default, iCal content is not included in notifications. If this parameter is enabled, email notifications with ics content can be interpreted by iCal aware clients and even used for responding from the email client itself. For this feature to work correctly, notification.dav.enableemailnotif , notification.dav.enablejmsnotif , and notification.dav.enableimipmailnotif must all be enabled.

You can enable or disable these parameters by using Jconsole or the **davadmin** utility. You do not need to restart the server for a change to these parameters to take effect.

The settings are not cumulative. That is, to receive email notification, not only should **notification.dav.enableemailnotif** be set to **true**, so should **notification.dav.enablejmsnotif**. Similarly, to get ics information in notifications, all three configuration options must be set to **true**.

Other **notification.dav.*** configuration parameters control items such as the SMTP server to use and its settings, maximum notification payload, location of notification templates, and so on. The **davcore.autocreate.enableemailnotification** parameter determines if notification is enabled by default on a newly created account and the **davcore.autocreate.emailnotificationaddressattr** parameter specifies which LDAP attribute to set as the default notification address when autocreating an account. (The default value is **mail**.) For more details, see "[Calendar Server Configuration Parameters](#)."

Enabling Notifications on an Account

To enable notifications for all accounts:

1. Use the **davadmin** command to set the **davcore.autocreate.enableemailnotification** to **true**.

```
./davadmin config modify -o davcore.autocreate.enableemailnotification -v true
```

Enter Admin password: *password*

2. If necessary, change the value of the LDAP attribute corresponding to **davcore.autocreate.emailnotificationaddressattr**, which is used to set the email notification address during account autocreation. The default value is **mail**.

Modifying Notifications on an Account

Calendar Server stores the values for the **davcore.autocreate.enableemailnotification** and **davcore.autocreate.emailnotificationaddressattr** parameters in the database as properties for each account. These parameters can be modified in two ways:

- User: Use a WCAP client that is capable of running the **set_accountprops.wcap** command, specifying a new value for **notifemail** and **notifrecipients**.
- Administrator: Run the **davadmin account** command.

For more information on **davadmin account** see "[Calendar Server Command-Line Utilities](#)."

For information on the **get_accountprops.wcap** command, see *Calendar Server WCAP Developer's Guide*.

Managing Notification Templates

This section describes the Calendar Server notification service in more detail and how to customize notification templates for your deployment.

Topics in this section:

- [Notification Types](#)
- [Templates, Resource Bundle, and Other Configuration Files](#)
- [Customizing Templates](#)
- [Preserving Customized Template Files During Calendar Server Upgrade](#)

Notification Types

The notification message contains a type field that indicates what action triggered the notification and thus helps the consumer decide how to process it.

[Table 8–2](#) describes the notification types. It also lists the payload data, which is the resource content (for example, iCal data) in byte array format. Attachments are not included.

Table 8–2 Notification Types

Notification Type	Description	Payload	CS7 Consumer Action
ALARM	Alarm	iCal data	Email is sent if ACTION type is EMAIL.
AUTOCREATE	Initial creation of a user's home collection (and its default sub-collections)	None	Email sent if creation happened as a result of a scheduling invitation. Creation due to user login or explicit account creation by using the davadmin command does not trigger an email.
CREATE_CAL_COLLECTION	Creation of a calendar collection	None	None.

Table 8–2 (Cont.) Notification Types

Notification Type	Description	Payload	CS7 Consumer Action
CREATE_CAL_RESOURCE	Creation of an entry (event or task) in a calendar collection	iCal data	None.
CREATE_COLLECTION	Creation of a non-calendar collection	None	None.
CREATE_RESOURCE	Creation of an entry in a non-calendar collection	iCal data	None.
DELETE_CAL_COLLECTION	Deletion of a calendar collection	None	None.
DELETE_CAL_RESOURCE	Deletion of an entry (event or task) in a calendar collection	iCal data	None.
DELETE_COLLECTION	Deletion of a non-calendar collection	None	None.
DELETE_RESOURCE	Deletion of an entry in a non-calendar collection	iCal data	None.
EVENT_START	Event start for presence integration	UID, DTSTART, DTEND	Notification email is triggered if presence notification is enabled (davcore.presence.enable=true).
EVENT_END	Event end for presence integration	UID, DTSTART, DTEND	Notification email is triggered if presence notification is enabled (davcore.presence.enable=true).
MODIFY_CAL_RESOURCE	Modification of an entry (event or task) in a calendar collection	iCal data	None.
MODIFY_RESOURCE	Modification of an entry in a non-calendar collection	iCal data	None.
MOVE_CAL_COLLECTION	A calendar collection was moved	None	None.
MOVE_CAL_RESOURCE	An entry in a calendar collection was moved	iCal data	None.
MOVE_COLLECTION	A non-calendar collection was moved	None	None.
MOVE_RESOURCE	An entry in a non-calendar collection was moved	None	None.
SHARE_ACCOUNT	An account was shared	None	An email is sent if additional permission was granted.
SHARE_CAL_COLLECTION	A calendar collection was shared	None	An email is sent if additional permission was granted.
SCHEDULE_ITIP*	Scheduling iTIP message	iCal data	iTIP scheduling: Announces an iTIP scheduling event, task, or a significant change to an event or task to an external attendee.
SCHEDULE_RECEIVE	Scheduling message is received	iCal data	Sends an email notification of the invitation or the response as long as it refers to an event or task in the future. Notifies attendee of new event, task, or a significant change to the event/task.
SCHEDULE_SEND	Scheduling message is sent	iCal data	None.
NONE	Undefined type	iCal data	Not applicable.

SCHEDULE_ITIP* notification type is used by the notification service to directly send iMIP messages to external invitees by using the same template substitution mechanism. No posting is done to the JMS bus.

Templates, Resource Bundle, and Other Configuration Files

This section contains the following topics:

- [Notification Configuration](#)
- [Resource Bundles](#)
- [Template Files](#)

Notification Configuration

You enable or disable notifications and set the values of the SMTP server used by the notification consumer by using the **davadmin** command or Jconsole. See "[Calendar Server Configuration Parameters](#)" for details on each of the configuration properties that you can set for notifications.

Resource Bundles

The value of the user's locale/preferred language attribute (defined by the **davcore.ldapattr.preferredlang** configuration parameter) in the user's directory entry is used to localize notification email. The attribute is retrieved from LDAP every time a notification is triggered and is then passed along as part of the notification object being published. If the user does not have any preferred locale/language, it defaults to the consumer module's system's default.

Template Files

Notification templates are files that contain pre-formatted notification messages. For example, **request.fmt** is used for scheduling request notification email message, while **sms.fmt** contains a short template for alarm SMS messages.

[Table 8–3](#) describes the available notification email templates. In a deployed production environment, by default the templates should be located in the **/config/templates** sub-directory, for example, **/opt/sun/comms/davserver/config/templates/**. The location of the templates directory is defined by the **notification.dav.configdir** configuration parameter.

Table 8–3 Scenarios That Trigger Notifications and Templates Files Used

Message Type	Notification Type	Template Files	From	To	Description
Alarm	ALARM	alarm.fmt?alarm_todo.fmt	User's scheduling address	Recipients listed in alarm	Email reminder for an upcoming event or todo.
Alarm	ALARM	sms.fmt	User's scheduling address	Recipients listed in alarm	SMS reminder for an upcoming event or todo. The SMS message is a more concise message but is still sent by email.
Auto creation	AUTOCREATE	autocreate.fmt	User's scheduling address.	User's scheduling address	Notifies of auto creation of user's home collection due to the arrival of the very first invitation.
Event Request Notification	SCHEDULE_RECEIVE	request.fmt, request_recur.fmt	Organizer	Notification recipients	Notifies attendee of a new event invitation or significant change to an invitation.

Table 8–3 (Cont.) Scenarios That Trigger Notifications and Templates Files Used

Message Type	Notification Type	Template Files	From	To	Description
Todo Request Notification	SCHEDUL E_ RECEIVE	request_ todo.fmt, request_recur_ todo.fmt	Organizer	Notification recipients	Notifies attendee of a new todo or significant change to a todo.
Event reply Notification	SCHEDUL E_ RECEIVE	reply_ requeststatus.f mt	Attendee	Organizer	Notifies the organizer of the status of an invitation, when the status is of value 3.x and 4.x, which indicates some issues with the scheduling.
Todo Reply Notification	SCHEDUL E_ RECEIVE	reply_ requeststatus_ todo.fmt	Attendee	Organizer	Notifies the organizer of the status of a todo, when the status is of value 3.x and 4.x, which indicates some issues with the scheduling.
Event Cancel Notification	SCHEDUL E_ RECEIVE	cancel.fmt, cancel_ recur.fmt, cancel_imip_ todo.fmt, cancel_recur_ imip_todo.fmt	Organizer	Notification recipients	Notifies of a canceled event (to attendee).
Todo Cancel Notification	SCHEDUL E_ RECEIVE	cancel_ todo.fmt, cancel_recur_ todo.fmt	Organizer	Notification recipients	Notifies of a canceled todo (to attendee).
Event Reply Notification	SCHEDUL E_ RECEIVE	reply.fmt, reply_recur.fmt, reply_imip_ todo.fmt, reply_ recur_imip_ todo.fmt	Attendee	Organizer	Notifies of the following reply scenarios: <ol style="list-style-type: none"> 1. Notifies the event organizer that an attendee accepted the invitation; 2. Notifies the event organizer that an attendee declined the invitation; 3. Notifies the event organizer that an attendee tentatively accepted the invitation.
Todo Reply Notification	SCHEDUL E_ RECEIVE	reply_todo.fmt, reply_recur_ todo.fmt	Attendee	Organizer	Notifies of the following reply scenarios: <ol style="list-style-type: none"> 1. Notifies the todo organizer that an attendee accepted the todo; 2. Notifies the todo organizer that an attendee declined the todo; 3. Notifies the todo organizer that an attendee tentatively accepted the todo.
Event Request Notification with iMIP Data	SCHEDUL E_ RECEIVE	request_ imip.fmt, request_recur_ imip.fmt, request_imip_ todo.fmt, request_recur_ imip_todo.fmt	Organizer	Notification recipients	Notifies attendee of a new event or significant change to the event. The notification contains iCal information because the notification.dav.enableimipemailnotif configuration parameter has been set to true.

Table 8–3 (Cont.) Scenarios That Trigger Notifications and Templates Files Used

Message Type	Notification Type	Template Files	From	To	Description
Event Cancel Notification with iMIP Data	SCHEDULE_RECEIVE	cancel_imip.fmt, cancel_recur_imip.fmt, cancel_imip.fmt, cancel_recur_imip.fmt	Organizer	Notification recipients	Notifies of a canceled event (to attendee). The notification contains iCal information because the notification.dav.enableimipemailnotif configuration parameter has been set to true .
Event Reply Notification with iMIP Data	SCHEDULE_RECEIVE	reply_imip.fmt, reply_recur_imip.fmt, reply_imip.fmt, reply_recur_imip.fmt	Attendee	Organizer	Notifies of the following reply scenarios: <ol style="list-style-type: none"> 1. Notifies the event organizer that an attendee accepted the invitation; 2. Notifies the event organizer that an attendee declined the invitation; 3. Notifies the event organizer that an attendee tentatively accepted the invitation. The notification contains iCal information because the notification.dav.enableimipemailnotif configuration parameter has been set to true.
iTIP Event Request	SCHEDULE_ITIP	itip_eventrequest.fmt	Organizer	External attendee	iTIP scheduling: Announces an iTIP scheduling event or a significant change to an event to an external attendee.
iTIP Todo Request	SCHEDULE_ITIP	itip_todorequest.fmt	Organizer	External attendee	iTIP scheduling: Announces an iTIP todo or a significant change to a todo to an external attendee.
iTIP Event Cancel	SCHEDULE_ITIP	itip_eventcancel.fmt	Organizer	External attendee	iTIP scheduling: Notifies of a cancellation of an iTIP scheduling event to an external attendee.
iTIP Todo Cancel	SCHEDULE_ITIP	itip_todocancel.fmt	Organizer	External attendee	iTIP scheduling: Notifies of a cancellation of an iTIP todo to an external attendee.
iTIP Event Reply	SCHEDULE_ITIP	itip_eventreply.fmt	External attendee	Organizer	iTIP scheduling: Replies to an iTIP scheduling event. <ol style="list-style-type: none"> 1. attendee accepted the invitation; 2. attendee declined the invitation; 3. attendee tentatively accepted the invitation.

Table 8–3 (Cont.) Scenarios That Trigger Notifications and Templates Files Used

Message Type	Notification Type	Template Files	From	To	Description
iTIP Todo Reply	SCHEDULE_ITIP	itip_todoreply.fmt	External attendee	Organizer	iTIP scheduling: Replies to an iTIP todo. <ol style="list-style-type: none"> attendee accepted the todo; attendee declined the todo; attendee tentatively accepted the todo.
Share calendar account	SHARE_ACCOUNT	share_account.fmt	Sharer's email address	Sharee's email address	Notifies of a calendar account being shared.
Share calendar collection	SHARE_CAL_COLLECTION	share_cal.fmt	Sharer's email address	Sharee's email address	Notifies of a calendar collection being shared.

Notes about notifications and templates:

- Notification recipients: A recipient list stored in the property, **SUN_NOTIFRECIPIENT**. By default, it's the scheduling address of the LDAP user on behalf of whom the operation is executed. It can be modified through interfaces provided by WCAP or by using the **davadmin** command.
- _recur** files: Templates of file names containing "_recur" are used for notifications regarding recurring resources.
- _imip** templates: These templates are used by the iSchedule gateway, and contain x-headers added by the gateway for special processing instructions.

Customizing Templates

Because JavaMail has interfaces to parse an entire string into a MIME message, a notification template file is designed to be a well-formatted email MIME message that contains character sequences denoted by a starting "%{" , and an ending "}" .

A template contains two types of trinkets:

- Resource bundle key: A place holder for locale-specific resource, in the format of `#{key}`;
 For example, trinket `#{summary}` contains a key "summary" that uniquely identifies a locale-specific object in the resource bundle.
- Value trinket: A place holder for notification field value, in the format of `%{trinket}`;

For a complete list of keys, refer to the **email.properties** file.

[Table 8–4](#) describes all notification values and trinkets.

Table 8–4 Notification Value Trinkets

Name	Description or Note	Example
summary	Summary	Not applicable
from	Email from value for this notification	Not applicable
to	Email to value for this notification	Not applicable

Table 8–4 (Cont.) Notification Value Trinkets

Name	Description or Note	Example
organizer	Organizer in ical	Not applicable
attendees	Attendee list in ical	Not applicable
sender	On behalf of sender	Not applicable
recipient	Original recipient	Not applicable
start	Start date/time for this notification	Not applicable
end	End date/time for this notification	Not applicable
location	Location in ical	Not applicable
description	Description in ical	Not applicable
note_recurring	Used in template in recurring resources	Not applicable
partstat	Used in reply	[ACCEPTED, TENTATIVE, DECLINED]
requeststatus	Used in <code>reply_requeststatus</code> templates	As defined in RFC5545
due	Used in todo templates	Not applicable
alarm_summary	Used in alarm templates	Not applicable
cal_owner	Owner of the shared calender. Used in share templates.	Not applicable
displayname	The displayname of a calendar	Not applicable
ical	Used for iMIP messages	Entire ical raw data

The following example shows an event request template, `request.fmt`, and the resulting notification message constructed from the template.

Event Request Template

```
Subject: ${event_request_notification} ${summary}
From: ${from}
To: ${to}
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 7BIT
${summary}: ${summary}
${organizer}: ${organizer}
${attendees}: ${attendees}
${start}: ${start}
${end}: ${end}
${location}: ${location}
${description}:
${description}
```

Resulting Notification Message

```
Subject: Event Request Notification: test
From: caluser39@example.com
To: caluser8@example.com
MIME-version: 1.0
Content-type: text/plain; charset=utf-8
Content-transfer-encoding: 7BIT
```

```

Summary: test
Organizer: caluser7@example.com
Attendees: caluser8@example.com
Start: Tue December 01, 2009 10:30:00 AM PST
End: Tue December 01, 2009 11:30:00 AM PST
Location: Loveland conf room, BRM05
Description: test notes.

```

The following example shows a customized **request.fmt** template, and the resulting notification message constructed from the template. In the resource bundle, a new entry should be included as shown in the example (**summary_cap=SUMMARY**).

Customized request.fmt

```

Subject: ${event_request_notification} ${summary}
From: ${from}
To: ${to}
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 7BIT
${summary_cap}: ${summary}
${organizer}: ${organizer}
${attendees}: ${attendees}
${start}: ${start}
${end}: ${end}
${location}: ${location}
${description}:
${description}

```

Resulting Notification Message

```

Subject: Event Request Notification: test
From: caluser39@example.com
To: caluser8@example.com
MIME-version: 1.0
Content-type: text/plain; charset=utf-8
Content-transfer-encoding: 7BIT
SUMMARY: test
Organizer: caluser7@example.com
Attendees: caluser8@example.com
Start: Tue December 01, 2009 10:30:00 AM PST
End: Tue December 01, 2009 11:30:00 AM PST
Location: Loveland conf room, BRM05
Description: test notes.

```

The Calendar Notifications module constructs the notification message from the corresponding template. Based on the user's preferred language/locale, the Notification module retrieves the locale-specific template from a runtime templates cache. It then performs more customization with the notification values. If the locale-specific template is not found, the original template is loaded and localized. The localized template is then stored in the cache, and thus should be constructed only once.

You can customize a template as long as it is in valid MIME format. Each resource bundle key is defined in the resource bundles and can be adjusted and added as long as it has a matching entry in the bundle files. All notification value trinkets are predefined in the Java source code, and should not be changed.

Preserving Customized Template Files During Calendar Server Upgrade

Customized notification template files are preserved during a Calendar Server upgrade. Normally, there should be no problem merging customized notification template files during the upgrade. If the upgrade encounters a problem with merging these files, the following message is displayed:

```
log_msg "There are conflicts in merging $file customization"
log_msg "Please finish the merge by manually resolving the conflicts in
$cfgFileNew"
```

The `$file` and `$cfgFileNew` are substituted with actual file names.

Writing a Java Messaging Service Consumer

Calendar Server Notification Services use a publish/subscribe paradigm. The information in this section describes how to create your own consumer program.

All Calendar Server notification messages are posted to a pre-defined JMQ Topic called `DavNotificationTopic`. Each message consists of the associated iCal data as the message body and some additional information passed in as properties.

Topics in this section:

- [Notification Message Format](#)
- [Code Sample](#)

Notification Message Format

The Notification data posted to the JMS bus is a JMS Message object. The iCal data is sent as a JMS message body, while all other information is sent as message properties.

The properties are as follows:

- **type** (notification type): Indicates the type of change that occurred. See "[Notification Types](#)" for the types that are currently defined. The following notification types trigger a notification message to be sent from the MDB consumer:
 - ALARM
 - AUTOCREATE
 - SCHEDULE_RECEIVE
 - SCHEDULE_ITIP (email directly from Calendar Server)
 - SHARE_ACCOUNT
 - SHARE_CAL_COLLECTION
- **resourceURI**: A string property indicating the URI of the changed resource.
- **fromAddress**: A string Property indicating the originator, the scheduling address of the principal URI who made the change.
- **toAddresses**: A string Property consisting of a comma separated list of recipient address(es) where the notification is to be delivered. The final list of recipients are further calculated based on the notification type and other conditions at the consumer.
- **locale**: A string Property that specifies the locale/preferred language of the owner of the resource or collection on which a notification was triggered.

- **notificationDate:** A Long Property with the timestamp when the notification was posted to the JMS bus.

The Notification Message payload is the resource content (that is, iCal data) in byte array format. Attachments are not included.

An instruction field in a notification carries a special instruction on processing of a notification. For example, an instruction of **EXCEED_PAYLOAD_LIMIT** indicates the iCal data involved in this change exceeds the pre-configured maximum JMS payload size, and thus the consumer needs to fetch the data separately.

Code Sample

The following sample code for a consumer assumes that you know how to implement Message Driven Beans (MDB), and that you are familiar with Calendar Server's notification model and types.

```
@MessageDriven(mappedName = "jms/DavNotificationTopic",
activationConfig = {
    @ActivationConfigProperty(propertyName = "subscriptionDurability", propertyValue =
"Durable"),
    @ActivationConfigProperty(propertyName = "clientID", propertyValue =
"CalDAVNotifConsumerID"),
    @ActivationConfigProperty(propertyName = "subscriptionName", propertyValue =
"CalDAVNotifConsumer")
})
public class NotificationConsumer implements MessageListener {
    ...
    public void onMessage(Message contents) {
        ...
        public void onMessage(Message contents) {
            if (contents instanceof StreamMessage) {
                Notification notif = null;
                // read the JMS message,
                StreamMessage msg = (StreamMessage) contents;
                try {
                    msg.reset();
                    byte[] theData = (byte[]) msg.readObject();
                    //first, filter out those types of notification that we want to process.
                    NotificationType type = NotificationType.valueOf(msg.getStringProperty("type"));
                    switch (type) {
                        case ACLCHANGE:
                        case CREATE_COLLECTION:
                        case CREATE_CAL_COLLECTION:
                        case DELETE_COLLECTION:
                        case DELETE_CAL_COLLECTION:
                        case SCHEDULE_SEND:
                        case CREATE_RESOURCE:
                        case CREATE_CAL_RESOURCE:
                        case DELETE_RESOURCE:
                        case DELETE_CAL_RESOURCE:
                        case MODIFY_RESOURCE:
                        case MODIFY_CAL_RESOURCE:
                            return;
                        case ALARM:
                        case AUTOCREATE:
                        case SCHEDULE_RECEIVE:
                            default:
                                break;
                    }
                }
            }
        }
    }
}
```

```

notif = new Notification(type,
msg.getStringProperty("resourceURI"),
msg.getStringProperty("fromAddress"),
msg.getStringProperty("toAddresses").split(", "),
msg.getStringProperty("locale"),
new Date(msg.getLongProperty("notificationDate")),
theData);
} catch (MessageEOFException meofex) {
LOGGER.log(Level.WARNING, "Error reading message data object: "
+ "unexpected end of message stream has been reached. \n",
meofex);
} catch (MessageFormatException mfex) {
LOGGER.warning("Invalid type conversion.\n" + mfex);
} catch (JMSEException jmse) {
LOGGER.log(Level.WARNING, "Error reading JMS message: "
+ "JMS provider fails to read the message due to some internal error.\n",
jmse);
}
if(notif != null) {
try {
// Get iCal data
byte[] data = notif.getData();
}
} .....

```

Managing Calendar Server Java Messaging Server Destinations

This section describes how to manage Java Messaging Server (JMS) destinations in Calendar Server by using the **imqcmd** command. For a complete list of **imqcmd** options, see the "Command Utility" chapter in *Sun Java System Message Queue 4.1 Administration Guide*.

Topics in this section:

- [Overview of Calendar Server JMS Destinations](#)
- [Using the imqcmd Command with Calendar Server](#)

Overview of Calendar Server JMS Destinations

The JMS API enables messages to be specified as either PERSISTENT or NON_PERSISTENT. By default, Calendar Server JMS notification messages are delivered in PERSISTENT mode. Thus, you should monitor and purge JMS messages for cases when the destination's accumulated messages are taking up too much of the system's resources. Calendar Server uses the **DavNotificationTopic** JMS topic.

Using the imqcmd Command with Calendar Server

Use the following tasks to use the JMS **imqcmd** command to work with JMS destinations:

- [Listing a JMS Destination's Metrics](#)
- [Purging All Messages](#)
- [Monitoring Disk Utilization](#)
- [Accessing Remote Brokers Tip](#)

Listing a JMS Destination's Metrics

To list a JMS destination's metrics:

1. Change directories to the *GlassFish_home/imq/bin* directory.
2. List and display the metrics of the JMS topic used by the Calendar Server, **DavNotificationTopic**.

```
./imqcmd list dst -t t -n DavNotificationTopic
...
./imqcmd metrics dst -t t -n DavNotificationTopic
...
```

Purging All Messages

Occasionally, you might need to purge all messages queued at the **DavNotificationTopic** physical destination, if the destination's accumulated messages are taking up too much of the system's resources. Purging a physical destination deletes all messages queued at the destination. Consider pausing the destination to temporarily suspend the delivery of messages from producers to the destination previous to the purge operation. Also, take a snapshot of the metrics before and after you run the **purge** command.

To purge all messages:

```
./imqcmd pause dst -t t -n DavNotificationTopic PRODUCERS
...
./imqcmd purge dst -t t -n DavNotificationTopic
...
./imqcmd resume dst -t t -n DavNotificationTopic
...
```

Monitoring Disk Utilization

To monitor a physical destination's disk utilization, use the **imqcmd metrics** command with the **dsk** option:

```
imqcmd metrics dst -t t -n DavNotificationTopic -m dsk -u admin
```

Accessing Remote Brokers Tip

You can also use the **-b host:port** option to specify a remote broker host name and port, for example, **-b host1.example.com:7676**.

Presence Notifications

Calendar Server publishes a JMS message on an event start and end that presence clients, including Oracle Communications Instant Messaging Server, can use to automatically set presence status. The client then displays a status message based on how that client consumes and posts the status. Calendar Server publishes this JMS message through its existing JMS infrastructure, which is also used to publish alarms for database changes.

Configuring Presence Notifications

1. If you upgraded to Calendar Server 7.0.4.14.0, run the **davadmin account upgrade** command.
2. Enable server-wide presence by setting the **davcore.presence.enable** parameter to **true**.

See "[Enabling Notifications on an Account](#)" for more information.

3. Configure the **davcore.presence.advancepresencetriggerinterval** parameter to set time difference in seconds, between actual event timings and trigger time.

Troubleshooting Calendar Server

This chapter describes troubleshooting strategies for Oracle Communications Calendar Server.

Troubleshooting Calendar Server Initial Configuration

If you experience trouble configuring Calendar Server while running the initial configurator script, **init-config**, and you are receiving an error from Oracle GlassFish Server, make sure that you are running at least Java 7 and that your environment is correctly configured. For more information, see *Glassfish Server Installation Guide*.

Troubleshooting GlassFish Server and Java

If you upgrade your Java SE to Java SE Development Kit 7, Update 7 (JDK 7u7) or later, you must also upgrade GlassFish Server to the recommended patch level. Otherwise, you may encounter problems running the **davadmin** command.

Troubleshooting Tips

Begin troubleshooting by making sure that the GlassFish Server web container is running and that Calendar Server is deployed. You can use either the GlassFish Server Administration Console or the **asadmin** command-line utility.

Topics in this section:

- [Using the asadmin Command to Specify GlassFish Server Port](#)
- [Using the GlassFish Server Administration Console to Check Calendar Server Status](#)
- [Using the asadmin Command-line Utility to Check Calendar Server Status](#)
- [Troubleshooting the Calendar Server davserver Process](#)
- [Troubleshooting a Failing davadmin Command](#)
- [Troubleshooting MySQL Server Errors](#)
- [Importing a Convergence ics File](#)
- [Refreshing Domain Information](#)
- [Troubleshooting the iSchedule Back End on MySQL Server](#)

Using the `asadmin` Command to Specify GlassFish Server Port

If you have more than one GlassFish Server instance installed, use the `asadmin -p` to specify the instance's administrative port number.

Using the GlassFish Server Administration Console to Check Calendar Server Status

1. Start the console.
2. Navigate to Web Applications under the Applications tab.
3. Make sure that `davserver` is deployed and enabled.

Using the `asadmin` Command-line Utility to Check Calendar Server Status

Run the following commands:

```
./asadmin list-components -p admin-port --type=web
davserver web-module
Command list-components executed successfully.
```

```
./asadmin show-component-status -p admin-port davserver
Status of davserver is enabled.
Command show-component-status executed successfully.
```

Troubleshooting the Calendar Server `davserver` Process

Follow these steps to troubleshoot the `davserver` process.

1. If `davserver` is not enabled, check the GlassFish Server log, `server.log`, in the `GlassFish_home/domains/domain1/logs` directory.
2. If `davserver` is deployed and enabled but clients have trouble connecting, check the `davserver` log, `calendar.*`, in the `/var/opt/sun/comms/davserver/logs` or equivalent directory. To increase the log level, use the `davadmin` command, for example:

```
./davadmin config modify -o log.dav.errors.loglevel -v FINE
```

See "[Calendar Server Command-Line Utilities](#)" for more information on the `davadmin` command.

Troubleshooting a Failing `davadmin` Command

If a `davadmin` command fails to run, use the `-e` option to get more details about the failure. For example:

```
./davadmin version
Enter Admin password:*****
DAV server connection failed. Is the server running?
```

```
./davadmin version -e
Enter Admin password:*****
JMXconnection exception for url
service:jmx:rmi:///jndi/rmi://commsuite.example.com:46633/jmxrmi - Exception
creating connection to: 1.1.1.1; nested exception is:
java.net.SocketException: java.security.NoSuchAlgorithmException: Error
constructing implementation (algorithm: Default, provider: SunJSSE, class:
com.sun.net.ssl.internal.ssl.DefaultSSLContextImpl)
```

This example shows SSL errors. In this case, you would make sure that the truststore file pointed to by the command through the **-s** option, or the **commandfile** option, or the default one, if none were specified explicitly, exists and is valid. The default truststore file, **.asadmintruststore**, is located in the **config** directory.

To verify:

1. As **root**, run an **asadmin** command on the GlassFish Server host on which Calendar Server is deployed. An **.asadmintruststore** file is created under the root (*/*) directory.
2. Make sure that this file is the same as the one in the Calendar Server **config** directory.

Also, see "[Troubleshooting GlassFish Server and Java.](#)"

Troubleshooting MySQL Server Errors

If you find a MySQL Server back-end error, make sure MySQL Server is running by pinging the JDBC **connectionpool**.

1. Start the GlassFish Server Administration Console.
2. Select **JDBC Resources** from Resources, then select **Connection Pools**.
3. Choose the **caldavPool** and perform a ping.
4. If the ping fails, check the Pool properties to make sure they are all correct.
5. You can also perform a command-line ping as follows:

```
./asadmin list-jdbc-connection-pools -p admin-port
__CallFlowPool
__TimerPool
DerbyPool
caldavPool
Command list-jdbc-connection-pools executed successfully.
```

```
./asadmin ping-connection-pool -p admin-port caldavPool
Command ping-connection-pool executed successfully.
```

6. Even if you ping the pool, sometimes Calendar Server is not able to load the back end. In this case, you see errors similar to the following:

```
SEVERE [2009-09-03T22:00:53.310-0700] <...JdbcBackend.getDataSource> Cannot
lookup DataSource: javax.naming.NameNotFoundException: defaultbackend1 not
found
```

```
SEVERE [2009-09-03T22:00:53.313-0700] <...DavServer.loadBackend> failed to
instantiate or create backend
com.sun.comms.davserver.backends.BackendException: Cannot get DataSource:
javax.naming.NameNotFoundException: defaultbackend1 not found(OPERATION_NOT_
SUPPORTED)
```

7. To see the pool and resource data clearly, view the GlassFish Server configuration file, for example:

```
GlassFish_home/domains/domain1/config/domain.xml
```

8. If cause of error is not clear, delete and recreate the Connection Pool and JDBC resource by using the **asadmin** command, for example:

```
./asadmin delete-jdbc-connection-pool -p admin-port caldavPool
./asadmin create-jdbc-connection-pool -p admin-port --user admin
```

```
--datasourceclassname com.mysql.jdbc.jdbc2.optional.MysqlDataSource --restype
javax.sql.DataSource --property
"DatabaseName=caldav:serverName=mysqlhost:user=caldav:password=mysqlpass:portNu
mber=3306:networkProtocol=jdbc" caldavPool

./asadmin create-jdbc-resource -p admin-port --user admin --connectionpoolid
caldavPool jdbc/defaultbackend
```

If you recreate the JDBC resource, be sure to use the same user name and password that you initially used to create the resource. Restart GlassFish Server after recreating the **connectionpool** and resource.

Importing a Convergence ics File

You might see the following error when importing an ics file that was created in Convergence.

```
./davadmin calresource import -u admin -a caluser6@example.com -m convergence_
caluser6_2.2.ics
```

```
Unable to import the resource into 'calendar'. DUE: 20090905T000000Z is before or
equal to DTSTART: 20090905T000000Z
```

This is likely a Convergence problem, because it creates the todo by setting **DTSTART** and **DUE** to the same time. This is due to the restrictions described in RFC5545. The description section states that **DUE** must be later than **DTSTART**.

The workaround is to manually fix the iCal data to have **DTSTART** before **DUE**.

Refreshing Domain Information

Calendar Server fetches and caches some domain information that is stored in LDAP, such as domain status. The system does not periodically refresh domain information, unlike user and group information.

If you need to refresh domain information, you can use one of the following methods:

- Restart the GlassFish Server.
- Using the **davadmin** command, make a change to any of the LDAP-related configuration options (**base.ldapinfo.***), which causes the server to refresh all cached LDAP data.

Troubleshooting the iSchedule Back End on MySQL Server

If you are unable to do a **POST** command to **/davserver/dav/ischedule/**, check the following:

1. Verify that the **davcore.scheduling.ischedulebackendid=ischedulebackendid** parameter has been set in the **davserver.properties** file.
2. Verify that you can ping the **ischedulepool** in GlassFish Server. If you get the error "Access denied for user 'mysql'@'localhost' to database 'ischedule'," then run the following MySQL Server command:

```
GRANT ALL ON ischedule.* TO 'mysql'@'localhost'
```

3. Verify that you can now ping **ischedulepool**.
4. Restart GlassFish Server and do a "clean and build" in NetBeans.

Enabling Telemetry Logging

To troubleshoot issues with a particular calendar user or client, it is useful to log all protocol interactions. You can force all telemetry logs by setting the **service.dav.telemetry.forcetelemetry** parameter to **true**. Do not use this setting unless required as it generates lots of data.

To enable telemetry logging at a reduced level, set the **service.dav.telemetry.filter** parameter. This parameter takes a space separated list of request URI prefixes that should be logged. For example:

- `/wcap/` logs all WCAP access.
- `/dav/principals/caluser1/ /dav/home/caluser1/` logs all Calendar Server access to **caluser1**'s account (both principals and home collections, and all the resources underneath).

Common Errors in Log Files

This section presents common errors that you might see in the Calendar Server log files. For more information about using log files, see "[Managing Logging](#)."

Topics in this section:

- [Using the Same Start and End Date for an Event](#)
- [Same UID Already in Use](#)
- [No Specification of Content-type Header](#)
- [Deleting a Non-existing File](#)
- [Posting to Calendar Collection Without a File Name](#)
- [Using a Non-implemented HTTP Method](#)

Using the Same Start and End Date for an Event

```
FINE [2009-08-24T19:28:57.020-0700] <...DavServlet.service> Got a non standard
condition: DTEND: 20090829T000000 is before or equal to DTSTART: 20090829T000000
```

```
INFO [2009-08-24T19:28:57.021-0700] <...DavServerServlet.service> [RES] [403]
Command execution time: 0.041 secs
```

Same UID Already in Use

```
FINE [2009-08-24T19:30:50.044-0700] <...DavServlet.service> Got a non standard
condition: uid q3EfPB0C4EHHj918X2GVU1 already in use in
/dav/home/modendahl/calendar/765345.ics
```

```
INFO [2009-08-24T19:30:50.046-0700] <...DavServerServlet.service> [RES] [403]
Command execution time: 0.063 secs
```

No Specification of Content-type Header

```
FINE [2009-08-24T19:32:07.803-0700] <...DavServlet.service> Got a non standard
condition: unsupported content-type: application/octet-stream
```

```
INFO [2009-08-24T19:32:07.805-0700] <...DavServerServlet.service> [RES] [403]
Command execution time: 0.019 secs
```

Deleting a Non-existing File

```
FINE    [2009-08-24T19:32:58.098-0700] <...DavServlet.service> Got a non standard
condition: getNode returned null for uri /dav/home/modendahl/calendar/teeeest.ics

INFO    [2009-08-24T19:32:58.099-0700] <...DavServerServlet.service> [RES] [404]
Command execution time: 0.012 secs
```

Posting to Calendar Collection Without a File Name

```
FINE    [2009-08-24T19:33:39.239-0700] <...DavServlet.service> Got a non standard
condition: Invalid Resource Type in POST:CALENDAR_RESOURCE

INFO    [2009-08-24T19:33:39.241-0700] <...DavServerServlet.service> [RES] [403]
Command execution time: 0.02 secs
```

Using a Non-implemented HTTP Method

```
INFO    [2009-08-24T19:35:10.416-0700] <...DavServerServlet.service> [REQ] CONNECT
/dav/home/modendahl/calendar/ 192.18.127.57 ics-s6.sfbay.example.com:8080

INFO    [2009-08-24T19:35:10.418-0700] <...DavServerServlet.service> [RES] [501]
Command execution time: 0.0020 secs
```

Using the Browser Servlet

You can use a browser servlet to view an account's properties stored in collections and resources. You might find this helpful when troubleshooting calendar problems.

To access this browser servlet, take any valid **dav** URI and replace the **dav** prefix following **davserver** with **browse**. For example, in a browser, change the following:

```
http://example.com:3080/davserver/dav/home/smithj/calendar/
```

to:

```
http://example.com:3080/davserver/browse/home/smithj/calendar/
```

The servlet returns a view of the account's properties stored in collections and resources. You can navigate among properties and delete them as well. The servlet also has some import function if you want to use a server-side import instead of a client-side import.

The delete and file import features are enabled only when the logging level is set at **FINE** or lower. To specify the logging level, use the **log.dav.errors.loglevel** configuration parameter.

Tip: You can log in with Calendar Server administrator (the default is **calmaster**) credentials to view multiple accounts with one login. Also, when viewing multiple accounts, clear your browser cache before viewing the next account.

Troubleshooting CalDAV Clients

This section describes client issues.

Topics in this section:

- [Lightning](#)

- [Apple iCal](#)
- [iPod touch](#)
- [Known Issues](#)
- [Troubleshooting Clients Running iOS 5 and Mac OS 10.7](#)
- [Mac OS 10.9 iCal Client Not Able to Delete Events](#)
- [Checking Active Calendar Users](#)

Lightning

Lightning does not support more than one calendar account. It allows only one account per server at a time.

Lightning 0.9 does not support multiple reminders for single events. Lightning 1.0 beta 1 does support multiple reminders.

Lightning is not able to create an account if the user name contains special characters.

If Lightning 1.0 beta1 is installed with Thunderbird 3, and you try to go back to Thunderbird 2 and Lightning 0.9, when starting Thunderbird the following error occurs:

The Calendar data in your profile was updated by a newer version of Lightning, and continuing will probably cause the information to be lost or corrupted. Lightning will now be disabled and Thunderbird restarted.

To fix this error:

1. Export all your calendar data in iCalendar format.
2. Remove the calendar database **storage.sdb** file from your profile.
3. Restart the Thunderbird and import the iCalendar file.

Thunderbird on Solaris needs to be reloaded twice to get the newly created Todo.

Inviting users on Thunderbird for Solaris and checking their availability does not show free/busy check properly. Instead, the invitee is always shown as free even when the invitee is busy.

Calendar import is failing with Thunderbird on Windows and Solaris. The failed import displays the "Modification_failed" error message. Logging back in to the profile loads the imported data to the calendar.

Apple iCal

Apple iCal adds its own default reminder to an event if you select the "Add a default alarm to all new events and invitations" option. Thus, if calendar1 exports the event (with no reminder) and calendar2 imports it, the imported reminder has a default alarm set.

Apple iCal is not able to create an account if the user name contains special characters.

If the event is created with "Repeat on Weekdays only" option from Lightning or Convergence, the Apple iCal will convert it to "Every day" and display it.

iPod touch

The following information was found with iPod touch 3.1.3 firmware.

Supported Features:

- Event is supported.
- Reminders are supported with iPod touch, up to a maximum of two reminders for a single event.
- Recurrence is supported.
- iPod touch client enables you to create duplicate calendar.

Unsupported Features:

- Todos.
- [AppleiPhone]STATUS not being taken into account when invitation event canceled by organizer.
- Invitations can be viewed by not accepted, declined or rejected.
- When the organizer of the event cancels the event, invitees do not have any information that event is canceled.
- Attachments.
- Free/busy.
- Availability check.
- Import-Export functionality.
- Share/Subscribe of calendar.

Known Issues

Apple iPhone STATUS not being taken into account when invitation event canceled by organizer

When an iPhone 3 user gets an invitation from a Lightning user, there is no option to accept, reject, or decline the event. Additionally, when the inviting user deletes the event, the iPhone user does not receive an event notification, nor is the event deleted from the user's calendar. The event is in read-only mode. This issue is fixed starting with the iPhone 4 release.

Connector for Microsoft Outlook

See "[Connector for Microsoft Outlook and Event Time Modifications.](#)"

Troubleshooting Clients Running iOS 5 and Mac OS 10.7

For correct setup and data synchronization to occur on devices running iOS 5 and Mac OS 10.7, make sure that you have installed at least Calendar Server 7 Update 2 Patch 5.

Mac OS 10.9 iCal Client Not Able to Delete Events

Currently, the Mac OS 10.9 iCal Client enables you to create or move events, but not delete events.

Checking Active Calendar Users

See "[Checking for Active Calendar Users.](#)"

Troubleshooting Calendar Server Agent Alerts in Instant Messaging Server

You can configure Instant Messaging Server for Java Message Service (JMS) to support Calendar Server 7 Agent alerts. If you find that you are not receiving event reminders (alarms) in an XMPP-enabled instant messaging client, verify that the password configuration has been properly configured.

Improving Calendar Server Performance

This chapter describes how to tune your Oracle Communications Calendar Server deployment.

Topics:

- [Tuning Calendar Server Logging](#)
- [Tuning Oracle GlassFish Server](#)
- [Tuning MySQL Server](#)
- [Tuning Oracle Solaris CMT Server](#)
- [Tuning Reference](#)

Tuning Calendar Server Logging

The Calendar Server logging function is I/O intensive. For optimal performance, decrease the log level to **WARNING**. Another option is to store the log directory on a fast storage system, such as a solid-state (SSD) system.

To change the log level:

```
davadmin config modify -o log.dav.errors.loglevel -v WARNING
davadmin config modify -o log.dav.commands.loglevel -v WARNING
```

Tuning Oracle GlassFish Server

The following GlassFish Server configuration is for a medium-sized deployment. Adjust the values accordingly for your deployment.

- [Tuning JVM Options](#)
- [Tuning JDBC Pool](#)
- [Tuning HTTP Service and Listener](#)

Tuning JVM Options

```
-XX:+UseParallelOldGC
-XX:ParallelGCThreads=6
-Xms3200m
-XX:MaxPermSize=192m
-server
-Dsun.rmi.dgc.server.gcInterval=1800000
-Dsun.rmi.dgc.client.gcInterval=1800000
-Xmx3200m
```

```
-XX:NewRatio=2
```

Tuning JDBC Pool

```
max-pool-size=200
cachePrepStmts=true
prepStmtCacheSize=512
```

Tuning HTTP Service and Listener

Table 10-1 shows the HTTP service tuning settings.

Table 10-1 HTTP Service Tuning

HTTP Setting	Attribute	Value
keep-alive	max-connections	250
Not applicable.	thread-count	25
Not applicable.	timeout-in-seconds	30
request-processing	header-buffer-length-in-bytes	16384
Not applicable.	initial-thread-count	10
Not applicable.	request-timeout-in-seconds	20
Not applicable.	thread-count	50
Not applicable.	thread-increment	10
connection-pool	max-pending-count	4096
Not applicable.	queue-size-in-bytes	4096
Not applicable.	receive-buffer-size-in-bytes	4096
Not applicable.	send-buffer-size-in-bytes	8192

Table 10-2 shows the HTTP listener tuning settings.

Table 10-2 HTTP Listener Tuning

HTTP Listener Setting	Value
acceptor-threads	1
accessLoggingEnabled	false
xpowered-by	false

Tuning MySQL Server

Configure the cache size and max connection size. For example:

```
back_log = 50
max_connections = 200
binlog_cache_size = 1M
max_heap_table_size = 64M
sort_buffer_size = 8M
join_buffer_size = 8M
thread_cache_size = 8
thread_concurrency = 8
query_cache_size = 64M
query_cache_limit = 2M
```

```

ft_min_word_len = 4
memlock
thread_stack = 192K
transaction_isolation = REPEATABLE-READ
tmp_table_size = 64M
log-bin=mysql-bin
expire_logs_days=1
binlog_format=mixed
slow-query-log = 1
long_query_time = 2
log_long_format
tmpdir = /tmp
innodb_additional_mem_pool_size = 16M
innodb_buffer_pool_size = 2G
innodb_data_file_path = ibdata1:10M:autoextend
innodb_file_io_threads = 4
innodb_thread_concurrency = 16
innodb_flush_log_at_trx_commit = 1
innodb_log_buffer_size = 8M
innodb_log_file_size = 256M
innodb_log_files_in_group = 3
innodb_max_dirty_pages_pct = 90
innodb_lock_wait_timeout = 120
innodb_flush_method=O_DIRECT #UFS only

```

Caution: You can view contents of the back-end store by using standard MySQL tools. Do not use MySQL tools to modify your data.

Tuning Oracle Solaris CMT Server

This section provides tuning recommendations for Chip Multi-threading (CMT) architectures such as Sun servers with CoolThreads technology.

Set the following parameters in the `/etc/system` file.

```

set rlim_fd_max=260000
set hires_tick=1
set sq_max_size=0
set ip:ip_squeue_bind=0
set ip:ip_squeue_fanout=1
set ip:ip_soft_rings_cnt=16

```

TCP tuning:

```

ndd -set /dev/tcp tcp_time_wait_interval 60000
ndd -set /dev/tcp tcp_conn_req_max_q 3000
ndd -set /dev/tcp tcp_conn_req_max_q0 3000
ndd -set /dev/tcp tcp_max_buf 4194304
ndd -set /dev/tcp tcp_cwnd_max 2097152
ndd -set /dev/tcp tcp_xmit_hiwat 400000
ndd -set /dev/tcp tcp_recv_hiwat 400000

```

For Sun Fire T1000 and T2000 systems with 1.0GHz CPU, interrupt fencing by setting the following parameter:

```
psradm -i 1-3 5-7 9-11 13-15 17-19 21-23
```

Set the ZFS recordsize to 16 K (same as innoDB block size) by running the following commands:

```
zfs create rpool/data
zfs set recordsize=16K rpool/data
```

Tuning Reference

Refer to the following documentation for additional tuning information:

- MySQL:
http://www.solarisinternals.com/wiki/index.php/Application_Specific_Tuning
- Network:
<http://www.solarisinternals.com/wiki/index.php/Networks>
- GlassFish Server:
<http://download.oracle.com/docs/cd/E19159-01/819-3681/index.html>
- MySQL benchmarks:
<http://www.mysql.com/why-mysql/benchmarks/>
- Scaling MySQL, T5440, ZFS:
http://blogs.oracle.com/mrbenchmark/entry/scaling_mysql_on_a_256
- Spec:
<http://www.spec.org/jAppServer2004/results/jAppServer2004.html>

Backing Up and Restoring Calendar Server Files and Data

This chapter describes backing up and restoring files and data in Oracle Communications Calendar Server.

Topics:

- [Overview of Calendar Server Backup and Restore](#)
- [Calendar Server Backup and Restore Techniques](#)
- [MySQL Backup and Restore Techniques](#)
- [Oracle Database Backup and Restore Techniques](#)

Overview of Calendar Server Backup and Restore

Calendar store backup and restore is one of the most important administrative tasks for your Calendar Server deployment. You must implement a backup and restore policy for your calendar store to ensure that data is not lost if problems such as system crashes, hardware failures, or accidental deletion of information occur.

This information describes the two options for backing up and restoring the Calendar Server 7 calendar store (either MySQL database or Oracle Database, and the document store). You must understand the pros and cons of these solutions to make the proper choice for your deployment.

Note: You cannot back up the Calendar Server store by backing up the active calendar database and the Calendar Server **data** directory while Calendar Server is running. If you do so, bad data results. Thus, you must use one of the two methods described in this information.

Caution: You can view contents of the back-end store by using standard MySQL or Oracle Database tools. Do not use MySQL or Oracle Database tools to modify your data.

This information also assumes that you are backing up your LDAP Directory Server. Calendar Server stores user, group, and resource information in LDAP. Calendar Server uses the **davUniqueId** LDAP attribute to map each calendar entry (in LDAP) to a unique account in the calendar store. The unique identifier links various entries from different database tables for a user, group, and resource. You must use a unique identifier, and one that does not change, for user, group, and resource entries stored in

LDAP. For more information, see the topic on Calendar Server unique identifier in *Calendar Server Concepts*.

Calendar Server Backup and Restore Techniques

The section describes two ways to back up the Calendar Server data store.

Topics:

- [Using the davadmin db backup Command](#)
- [ZFS Snapshots](#)

Using the davadmin db backup Command

Calendar Server 7 provides the **davadmin db backup** command to back up the calendar server data.

Pros:

- Supports partial backup and restore.
- You can also use **backup** and **restore** to migrate data from one Calendar Server host to another.

Cons:

- The **davadmin db backup** command is relatively slow.
- The **davadmin db restore** command might take longer than the **backup** command, as it needs to rebuild the database and indexes.

ZFS Snapshots

Use Oracle Solaris ZFS snapshots to produce an atomic snapshot of the file system containing the MySQL database or Oracle Database and the attachment store. Then use **zfs send** or a third-party file system backup software to back up the snapshot. See *ZFS Administration Guide* for more information.

Pros:

- Performance is better than **davadmin db backup**.

Cons:

- This method does not support partial backup and restore.

MySQL Backup and Restore Techniques

The following methods back up the MySQL database only. For general information about MySQL backup and restore, see the MySQL documentation at:

<http://dev.mysql.com/doc/refman/5.1/en/backup-and-recovery.html>

MySQL Asynchronous Replication

Use MySQL asynchronous replication to replicate the databases. For more information, see the MySQL documentation at:

<http://dev.mysql.com/doc/refman/5.1/en/replication.html>

MySQL Database Dump

Use **mysqldump** to dump the databases for backup or transfer to another SQL server. For more information, see the MySQL documentation at:

<http://dev.mysql.com/doc/refman/5.1/en/mysqldump.html>

Point-In-Time Binlog Backup and Recovery

The binary log files provide you with the information you must use to replicate changes to the database. For more information, see the MySQL documentation at:

<http://dev.mysql.com/doc/refman/5.1/en/point-in-time-recovery.html>

Oracle Database Backup and Restore Techniques

For general information about Oracle Database backup and restore, see the backup and recovery documentation at:

http://docs.oracle.com/cd/E11882_01/nav/portal_14.htm#backup_and_recovery

Part II

Administering a High-Availability System

Part II contains the following chapters:

- [Configuring a High-Availability Database](#)
- [Configuring Calendar Server for Highly Availability](#)
- [Configuring Calendar Server on a GlassFish Server Cluster](#)

Configuring a High-Availability Database

Oracle Communications Calendar Server relies on native capabilities offered by MySQL Server and Oracle GlassFish Server for a high-availability solution. This chapter provides details on one such scenario: setting up replication of MySQL Server by using the JDBC Connector/J driver in GlassFish Server. If you want more sophisticated solutions for MySQL Server high availability, refer to the MySQL Server documentation, for example, the High Availability and Scalability chapter in *MySQL 5.5 Reference Manual*.

Other high availability solutions are also known in the MySQL Server community. You should decide which solution best fits your deployment and requirements.

Note: The examples in this information are intended for only one Calendar front end per Calendar back end.

Topics:

- [Overview of MySQL Server Asynchronous Replication](#)
- [MySQL Server Asynchronous Replication Example](#)
- [MySQL Server Two-Way Replication Example](#)
- [Replication Synchronization Issues](#)
- [Using the Multi-Host Failover Feature of JDBC Connector/J](#)
- [Test for MySQL Server Asynchronous Replication \(Manual\)](#)
- [Test for MySQL Server Two-Way Replication with Connector/J Failover](#)

Overview of MySQL Server Asynchronous Replication

MySQL Server, as well as third parties, offer a wide range of high availability options ranging from completely manual to high-end MySQL Server HA solutions. MySQL Server implements manual asynchronous replication, provided within the product itself. This has been in use for some time and is stable. Failover, failback, and resynchronizing nodes, and adding a node, are all done manually. See the topic on replication in *MySQL 5.5 Reference Manual* for more information.

MySQL Server 5.5 provides semi-synchronous replication in which the master node tries to sync with at least one other node before completing the request, subject to a timeout.

MySQL Server Asynchronous Replication Example

This section describes a simple example of MySQL Server asynchronous replication configuration for MySQL Server 5.5.8 consisting of one master and one slave.

To configure asynchronous replication:

1. Edit the `/etc/my.cnf` file for both master and slave MySQL Server hosts as follows:

```
[mysqld]
basedir = /opt/mysql/mysql
datadir = /var/opt/sun/comms/davserver/db
default-storage-engine = InnoDB
character-set-server = utf8
transaction-isolation = READ-COMMITTED
server-id=1
log-bin=mysql-bin
innodb_flush_log_at_trx_commit=1
sync_binlog=1
binlog-format=ROW
skip-name-resolve
```

Note the following:

- **server-id:** The unique id for each server numbered greater than 0. (The master here is 1, the slave is 2.)
 - **log-bin:** Turns on binary logging for replication.
 - **innodb_flush_log_at_trx_commit:** Recommended.
 - **sync_binlog:** Recommended.
 - **binlog-format:** Must be **ROW** because of transaction-isolation level used in Calendar Server and MySQL Server 5.5.8.
 - **skip-name-resolve:** This is a workaround if you experience the following slave error:

```
ERROR 1042 (HY000): Can't get hostname for your address.
```
 - Do not set **log-slave-updates**.
2. Follow the replication procedures described in the "Replication" chapter of *MySQL 5.5 Reference Manual* to complete the configuration.

In this example, both nodes have a **log-bin** and do not have **log-slave-updates**. Some procedures in the "Replication" chapter might include **log-slave-updates** for specific purposes, but they have been left out from these instructions, assuming most of the time a failed node retains its data and does not need the **log-slave-updates** data on the other node. These instructions also assume that the binary log removal over time might indicate a node that lost its data, and so is not able to get all data from the other node's logs anyhow.

MySQL Server Two-Way Replication Example

This section describes an example of a manually-controlled HA configuration. It is similar to the preceding asynchronous replication example, except that each node is set to be the slave of the other. Because both nodes have turned on binary logs, each node logs the data that comes to it. The configuration is the same as asynchronous (assuming the replication user exists on both nodes).

This is not very different from one-way synchronization considering that even in master-slave replication you can set up the master as a slave to the original slave when you are trying to resynchronize a failed master.

However, another way of using two-way replication is to assign specific client data to each node, and use the opposite node for the slave. The result is that each node replicates the other in parallel. It's important that any specific client data be assigned to only one master at a time or else inconsistencies might occur.

Replication Synchronization Issues

When managing MySQL Server replication, it is important to understand synchronization issues. If data arrives at one node in a different order as it does in a replicated node, replication might fail and halt.

For example, given one-way replication, imagine that the master receives data called "dog" and at the same time a client believing the master is not available sends "cat" to the slave. The master has "dog" in row 1, and the slave has cat in row 1. The data between the nodes has become inconsistent. When the slave receives "dog" from the master relay, it is not able to put it into row 1 as it exists on the master. The nodes become inconsistent and replication fails and halts.

MySQL Server has no internal mechanism to synchronize this automatically, and when the MySQL Server server comes up as an active slave, it accepts new connections and also tries to catch up with the master in parallel.

These issues need to be considered when executing manual replication procedures as well as when using automatic functions of any other program.

Using the Multi-Host Failover Feature of JDBC Connector/J

JDBC Connector/J for MySQL Server has various capabilities for load balancing and high availability. The example configuration in this section shows how to use JDBC Connector/J for MySQL Server for an automatic failover, assuming a manual resynchronization and fallback. This example assumes the use of MySQL Server 5.5.8.

1. On the GlassFish Server, make the following connector configuration:

Note: The following information pertains to configuring the Calendar Server database. You might also want to configure the same for the iSchedule database if it exists on the same host.

- Enable connection validation.
 - Enable on any failure close all connection.
 - Transaction isolation: Guaranteed read-committed
2. Use the following properties:

```
user mysql
password mysql
URL jdbc:mysql://masterhost:3306,slavehost:3306/caldav
autoReconnect true
failOverReadOnly false
autoReconnectForPools true
roundRobinLoadBalance false
secondsBeforeRetryMaster 2147483647
queriesBeforeRetryMaster 2147483647
```

3. For more information, refer to the "MySQL Server Connector/J" chapter on configuration parameters in *MySQL 5.5 Reference Manual*. The MySQL Server replication configuration is two-way with each node being both a master and slave to the other. The example shows a failure on **master1**, and **master1** failing over to **master2**.

Note: `secondsBeforeRetryMaster` and `queriesBeforeRetryMaster` are set to a very large value to prevent GlassFish Server from failing back to **master1** once it has experienced a failover. This prevents new data from being written to **master1** before **master1** has had a chance to catch up. Otherwise data might become inconsistent.

Failover and recovering of this example works as follows:

1. Fail over.

You need a way to be alerted that a node has gone down. Once you are alerted, you must address the situation quickly to control when **master1** comes back up and is resynchronizied and new connections are made to it.

2. Recover **master1**.

If **master1**'s data is damaged or lost, you must reload **master1** from **master2** described in the MySQL Server replication notes.

- Also, if **master1** has data that did not make it to **master2** before it failed over, it might be easier to reload **master1**. This situation is less likely if semi-synchronous replication is used.
- You can check the binary log position on **master1** with `show master status`, compared to **master2**'s `show slave status`, to determine if **master2** received all of **master1**'s data without error.

3. Bring back **master1**.

At this point it, might be useful to shut down GlassFish Server to make sure that **master1** does not receive new connections before it can resynchronize with **master2**. Other procedures are possible, but **master1** needs to resynchronize before it receives new GlassFish Server connections.

4. Bring up **master1**.

Verify it has resynchronized with **master2** by using `show master status` on **master2** versus `show slave status` on **master1**.

5. Once **master1** is caught up, you can restart GlassFish Server and it should return connections to **master1**, effectively failing back.

Note: Be sure to verify, test, and refine any HA procedure before putting it into production.

Test for MySQL Server Asynchronous Replication (Manual)

The test described in this section uses the following software and servers:

- MySQL Server 5.5.8 Enterprise Version
- JDBC Connector/J 5.1.5

- Two MySQL Server servers named **Master** and **Slave**
 - Oracle Solaris 10
 - GlassFish Server 2.1
1. On host **Master**, create a user named **mysql** that has replication permission on **Master**. In this case, user **mysql** is the same user name that Calendar Server uses itself to connect to MySQL Server.

```
GRANT REPLICATION SLAVE ON *.* TO 'mysql'@'%';
```

2. On both hosts **Master** and **Slave**, edit the `/etc/my.cnf` config file as follows.

```
[mysqld]
basedir = /opt/mysql/mysql
datadir = /var/opt/sun/comms/davserver/db
default-storage-engine = InnoDB
character-set-server = utf8
transaction-isolation = READ-COMMITTED
server-id=1
log-bin=mysql-bin
innodb_flush_log_at_trx_commit=1
sync_binlog=1
binlog-format=ROW
skip-name-resolve
```

3. Modify the GlassFish Server and JDBC Connector/J configuration to fail over from **Master** to **Slave**.

The following is for the **caldav** database. Do the same for the **iSchedule** database.

```
DataSource Classname: com.mysql.jdbc.jdbc2.optional.MysqlDataSource
Resource Type: javax.sql.DataSource
```

```
Enable 'connection validation'
Enable 'on any failure close all connection'
transaction isolation: Guaranteed
read-committed
```

```
user                mysql
password            mysql
URL                 jdbc:mysql://Master:3306,Slave:3306/caldav
autoReconnect       true
failOverReadOnly    false
autoReconnectForPools true
roundRobinLoadBalance false
secondsBeforeRetryMaster 2147483647
queriesBeforeRetryMaster 2147483647
```

secondsBeforeRetryMaster and **queriesBeforeRetryMaster** are set to a high value to prevent GlassFish Server and Connector/J from failing back if the master were to come back up.

4. Initialize both hosts **Master** and **Slave**.
 - a. Run the following command, if the servers were previously functioning as a slave:

```
stop slave;
```

- b. Remove all Calendar Server data from both hosts so that the databases are synchronized.

For example:

```
./davadmin db init -H localhost -t mysql -u mysql -d caldav
./davadmin db init -H localhost -t mysql -u mysql -d ischedule
```

Substitute your names for *caldav* and *ischedule*.

Caution: These commands completely remove the calendar data.

- c. On both **Master** and **Slave**, run the following command:

```
reset slave;
```

- d. On both **Master** and **Slave**, run the following command:

```
reset master;
```

5. Set up **Slave** to be synchronized with **Master**.

- a. Run the following command on **Master**:

```
show master status;
```

```
File = mysql-bin.000001
```

```
Position = 107
```

- b. Note the File and Position to set parameters on **Slave**.

6. Run the following command on **Slave**:

```
CHANGE MASTER TO
MASTER_HOST='Master',
MASTER_USER='mysql',
MASTER_PASSWORD='mysql',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=107;
```

7. Start **Slave**:

```
start slave;
```

8. Restart GlassFish Server to load Calendar Server.

9. Verify both **Master** and **Slave**.

```
use caldav;
select count(*) from Resources;
```

There should be one row after GlassFish Server starts.

10. Run data through Calendar Server (that is, use Calendar Server to create or change events, and so on, to cause calendar data to be stored).

11. Verify that data is accumulating on both **Master** and **Slave**.

For example, use the following MySQL commands to look at data in the tables:

```
use caldav;
select * from Resources;
```

You can also use this command:

```
select count(*) from Resources;
```

You can also use the following **status** commands:

```
show master status;
show slave status;
```

12. Stop **Master** and observe the failover.

The failover could take up to 60 seconds or more, as the connections time out.

13. Use the following command to observe that **Slave** is continuing to operate:

```
select count(*) from Resources;
```

Test for MySQL Server Two-Way Replication with Connector/J Failover

The test described in this section uses the following software and servers:

- MySQL Server 5.5.8 Enterprise Version
 - JDBC Connector/J 5.1.5
 - Two MySQL Server servers named **Master1** and **Master2**
 - Oracle Solaris 10
 - GlassFish Server 2.1 1
1. On both hosts **Master1** and **Master2**, create a user that has replication permission.

In this case, user **mysql** is the same user name that Calendar Server uses itself to connect to MySQL Server.

2. Run the following command both **Master1** and **Master2**.

```
GRANT REPLICATION SLAVE ON *.* TO 'mysql'@'%';
```

3. On both hosts **Master1** and **Master2**, edit the **/etc/my.cnf config** file as follows.

```
[mysqld]
basedir = /opt/mysql/mysql
datadir = /var/opt/sun/comms/davserver/db
default-storage-engine = InnoDB
character-set-server = utf8
transaction-isolation = READ-COMMITTED
server-id=1
log-bin=mysql-bin
innodb_flush_log_at_trx_commit=1
sync_binlog=1
binlog-format=ROW
skip-name-resolve
```

4. Modify the GlassFish Server and JDBC Connector/J configuration to fail over from **Master1** to **Master2**. The following is for the **caldav** database, you should also do the same for the iSchedule database.

```
DataSource Classname: com.mysql.jdbc.jdbc2.optional.MysqlDataSource
Resource Type: javax.sql.DataSource
```

```
Enable 'connection validation'
Enable 'on any failure close all connection'
transaction isolation: Guaranteed
read-committed
```

```
user          mysql
password     mysql
```

```

URL                jdbc:mysql://Master1:3306,Master2:3306/caldav
autoReconnect      true
failOverReadOnly   false
autoReconnectForPools true
roundRobinLoadBalance false
secondsBeforeRetryMaster 2147483647
queriesBeforeRetryMaster 2147483647
    
```

secondsBeforeRetryMaster and **queriesBeforeRetryMaster** are set to a high value to prevent GlassFish and Connector/J from failing back if the master were to come back up.

5. Initialize both hosts **Master1** and **Master2**:

a. Run the following command:

```
stop slave;
```

b. Remove all Calendar Server data from both hosts so that the databases are synchronized. For example:

```
./davadmin db init -H localhost -t mysql -u mysql -d caldav
./davadmin db init -H localhost -t mysql -u mysql -d ischedule
```

Substitute your names for *caldav* and *ischedule*.

Caution: These commands completely remove the calendar data.

c. On both **Master1** and **Master2**, run the following command:

```
reset slave;
```

d. On both **Master1** and **Master2**, run the following command:

```
reset master
```

6. Run the following command to verify file and position on each master:

```
show master status;
both show: mysql-bin.000001 107
```

7. Run the following commands to set each master to be a slave to the other:

a. On **Master2**:

```
CHANGE MASTER TO
MASTER_HOST='Master1',
MASTER_USER='mysql',
MASTER_PASSWORD='mysql',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=107;
```

b. On **Master1**:

```
CHANGE MASTER TO
MASTER_HOST='Master2',
MASTER_USER='mysql',
MASTER_PASSWORD='mysql',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=107;
```

8. Start slave connection on both hosts **Master1** and **Master2**:

```
start slave;
```

9. Restart GlassFish Server to load Calendar Server.

10. Verify both hosts **Master1** and **Master2**:

```
use caldav;
select count(*) from Resources;
```

There should be one row after GlassFish Server starts.

11. Run data through Calendar Server (that is, use Calendar Server to create or change events, and so on, to cause calendar data to be stored).

12. Verify that data is accumulating on both **Master** and **Slave**.

For example, use the following MySQL commands to look at data in the tables:

```
use caldav;
select * from Resources;
```

You can also use this command:

```
select count(*) from Resources;
```

You can also use the following **status** commands:

```
show master status;
show slave status;
```

13. Stop **Master1** and observe the failover.

The failover could take up to 60 seconds or more, as the connections time out.

14. Use this to observe the slave continuing:

```
select count(*) from Resources;
```

15. Stop GlassFish Server (to stop incoming client connections and data).

16. Bring **Master1** online and allow it to sync as a slave to **Master2**.

17. Verify that **Master1** synced as a slave to **Master2**.

- On **Master1**:

```
show slave status
```

Make sure that there are no errors and note the slave position, for example, **mysql-bin.000001 1827176**.

- On **Master2**:

```
show master status
mysql-bin.000001 1827176
```

Verify this position with the one that you noted on **Master1**.

18. Verify that **Master2** is still synced with **Master1**.

- On **Master2**:

```
show slave status
```

Make sure that there are no errors and note the slave position, for example, **mysql-bin.000002 107**.

- On **Master1**:

```
show master status
mysql-bin.000002 107
```

Verify this position with the one that you noted on **Master2**.

19. Verify Row Count on both **Master1** and **Master2** by comparing the count from each machine:

```
select count(*) from Resources;
```

20. Start GlassFish Server and start incoming client data.

21. Verify That **Master1** is in action again, and that **Master2** is following.

- On **Master1**:

```
show master status;
```

Verify that the position is increasing as master.

- On **Master2**:

```
show master status;
```

Verify that the position is not increasing.

- On both **Master1** and **Master2**:

```
select count(*) from Resources;
```

Configuring Calendar Server for Highly Availability

Choices for making Oracle Communications Calendar Server highly available include Oracle GlassFish HA Cluster, MySQL Async Replication, and Oracle Data Guard. You can also configure the document store for high availability.

Topics:

- [Front End High Availability: Configuring Calendar Server in GlassFish HA Cluster](#)
- [Back End High Availability: MySQL Async Replication](#)
- [Back End High Availability: Oracle Data Guard](#)
- [Document Store High Availability](#)

Front End High Availability: Configuring Calendar Server in GlassFish HA Cluster

This section describes how to deploy and configure Calendar Server on a GlassFish Server cluster. The GlassFish Server cluster feature enables you to create highly available and scalable deployment architectures.

- [About This Deployment Example](#)
- [Prerequisites and Deployment Example Architecture](#)
- [Configuring a Calendar Server Cluster Deployment](#)
- [GlassFish Server Load Balancer Limitations](#)

About This Deployment Example

This example assumes that you are familiar with the following tasks:

- Installing and configuring Calendar Server. See *Calendar Server Installation and Configuration Guide* for more information.
- Setting up clusters using GlassFish Server. See *GlassFish Server High Availability Administration Guide* for more information.

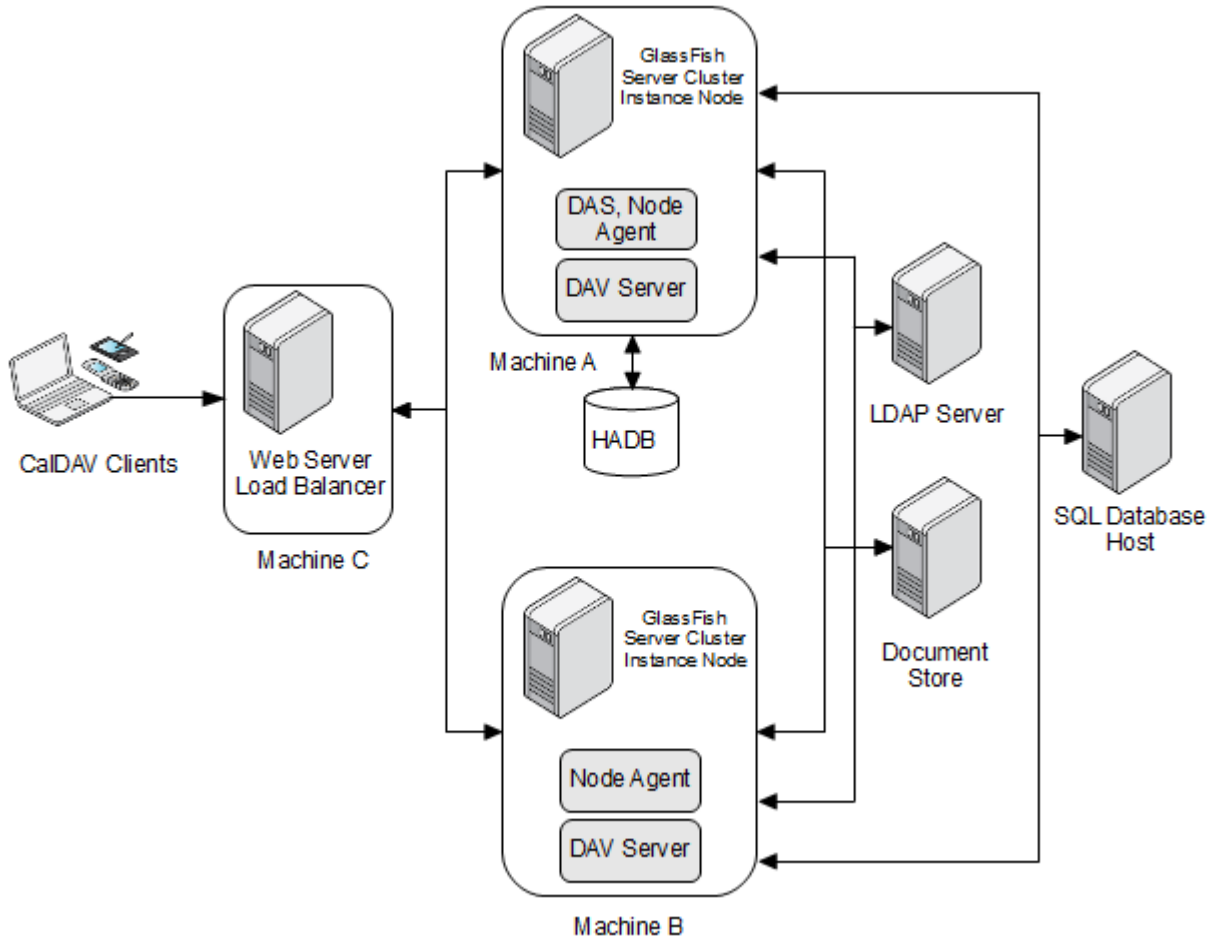
Prerequisites and Deployment Example Architecture

The hardware and software requirements for configuring Calendar Server in a GlassFish Server cluster are the same as configuring a default Calendar Server deployment. However, additionally you must install Oracle iPlanet Web Server

(formerly known as Sun Java System Web Server 7.0). In this example, Web Server functions as the load balancer.

Figure 13–1 shows the deployment architecture.

Figure 13–1 Calendar Server Cluster Deployment Architecture



Also, ensure that clocks are synchronized on the HADB hosts. See the topic on synchronizing system clocks for in *GlassFish Server High Availability Administration Guide* for more information.

Configuring a Calendar Server Cluster Deployment

For the following steps, refer to Figure 13–1.

Configuring Machine A Where DAS, Node Agent, and HADB Reside

1. Prepare shared memory configuration for HADB by editing the `/etc/system` file and adding the following entries:

```
set shmsys:shminfo_shmmax=0x80000000
set semsys:seminfo_semni=10
set semsys:seminfo_semmns=60
set semsys:seminfo_semmnu=600
```

2. Reboot the system:


```
reboot
```

Consult *GlassFish Server High Availability Administration Guide* for more shared memory configuration information.

3. Run the Glassfish Server installer and select the "High Availability Database Server," and "Domain Administration Server and Administration Tool" in the Component Selection screen.

4. Start the HADB agent.

```
GlassFish_home/hadb/4/bin/ma-initd start
```

5. Create an HADB Management Domain.

```
./hadbm createdomain --adminpassword=password machine-A-IP-address
export HADB_AGENT=machineA:1862
```

6. Give Node Supervisor Processes Root Privileges.

```
chmod u+s GlassFish_home/hadb/4/lib/server/clu_nsup_srv
chown root GlassFish_home/hadb/4/lib/server/clu_nsup_srv
```

7. Create a node agent for the instances in the cluster.

```
./asadmin create-node-agent --host machineA --port 4848 --user admin
--agentproperties remoteclientaddress=machineA dav-agent-1
./asadmin create-cluster --user admin dav-cluster
./asadmin create-instance --user admin --nodeagent dav-agent-1 --cluster
dav-cluster instance1
./asadmin create-instance --user admin --nodeagent dav-agent-2 --cluster
dav-cluster instance2
./asadmin set dav-cluster-config.jms-service.type=EMBEDDED
```

8. Go to ["Configuring Machine B Where Node Agent Resides"](#) and ["Configuring Machine C Where Web Server 7 Is Used as the Load Balancer"](#), then return and continue with the next step.

9. Create an HTTP load balancer.

```
./asadmin create-http-lb --user admin --devicehost machineC --deviceport
machineC-SSL-Port --autoapplyenabled=true --lbenableallinstances=true
--lbenableallapplications=true --target dav-cluster dav-lb
```

10. Log in to the GlassFish Server Administration Console on machine A and go to the load balancer dav-lb properties page and do "Test Connection" to make sure the SSL connection between DAS and Web Server 7 succeeds.

```
./asadmin configure-ha-cluster --user admin --hosts machineA,machineA
--haadminpassword password dav-cluster
```

11. Install and configure Calendar Server on machine A.

See *Calendar Server Installation and Configuration Guide* for instructions on installing and configuring Calendar Server. Make sure the web-app target is **dav-cluster**, and the MySQL host is the FQDN of the MySQL server, not the default localhost.

12. Enable HA on the deployed web app davserver, by editing the *GlassFish_home/domains/domain1/applications/j2ee-modules/davserver/WEB-INF/web.xml* file to add `<distributable/>` under `<web-app>`, and creating *sun-web.xml* in the preceding location with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc. DTD Application Server
```

```

9.0 Servlet 2.5//EN" ">http://www.sun.com/software/appserver/dtds/sun-web-app_
2_5-0.dtd"
<sun-web-app>
<session-config>
<session-manager persistence-type="ha">
<manager-properties>
<property name="persistenceFrequency" value="web-method"/>
</manager-properties>
<store-properties>
<property name="persistenceScope" value="session"/>
</store-properties>
</session-manager>
<session-properties/>
<cookie-properties/>
</session-config>
</sun-web-app>

```

13. Copy the contents of `/var/opt/sun/comms/davserver` from machine A to machine B, so that both instances in the **dav-cluster**, **instance1**, and **instance2** have the same configuration.

14. Enable load balancing for the davserver web app, and start the cluster.

```

./asadmin enable-http-lb-application -u admin --name davserver dav-cluster
./asadmin start-cluster dav-cluster

```

Configuring Machine B Where Node Agent Resides

1. Run the GlassFish Server installer and select Command Line Administration Tool Only in the Component Selection screen.
2. Create a node agent for the instance in the cluster and start the node agent.

```

./asadmin create-node-agent --host machineA --port 4848 --user admin
--agentproperties remoteclientaddress=machineB dav-agent-2
./asadmin start-node-agent dav-agent-2

```

Configuring Machine C Where Web Server 7 Is Used as the Load Balancer

1. Install Web Server 7 and create a symlink in the `WebServer_home` directory to the server instance directory.

```

ln -s /var/opt/SUNWwbsvr7/https-machineC /opt/SUNWwbsvr7/https-machineC

```

2. Start the Web Server 7 administrative server.

```

/var/opt/SUNWwbsvr7/admin-server/bin/startserv

```

3. Run the GlassFish Server installer, and select only the "Load Balancing Plugin" and use the path `/opt/SUNWwbsvr7/https-machineC` for the Web Server Location.
4. Synchronize Web Server instance configuration changes back to store, and start the instance.

```

wadm pull-config --config=machineC machineC
wadm deploy-config machineC
wadm start-instance --config=machineC machineC

```

5. Create an SSL http listener for use with load balancer configuration synchronization with GlassFish Server.

```

wadm create-selfsigned-cert --config=machineC --server-name=machineC
--nickname=cert-machineC

```

```
wadm create-http-listener --listener-port=machineC-SSL-Port --config=machineC
--server-name=machineC --default-virtual-server-name=machineC http-listener-ssl
wadm set-ssl-prop --config=machineC --http-listener=http-listener-ssl
enabled=true server-cert-nickname=cert-machineC client-auth=optional
wadm deploy-config machineC
wadm restart-instance --config=machineC
```

- From machine A, export the server certificate and copy the `s1as.rfc` file to `/var/tmp` on machine C.

```
GlassFish_home/lib/certutil -L -a -n s1as -d GlassFish_
home/domains/domain1/config > /var/tmp/s1as.rfc
```

- On machine C, import the GlassFish Server certificate to Web Server.

```
/opt/SUNWwbsvr7/bin/certutil -A -a -n s1as -t "TC" -i /var/tmp/s1as.rfc -d
/var/opt/SUNWwbsvr7/admin-server/config-store/machineC/config
```

- Add the following lines to the `default.acl` in the `/var/opt/SUNWwbsvr7/admin-server/config-store/machineC/config/default.acl` file. Change `/davserver` in the URI if it is different in your deployment.

```
acl "uri=/davserver";
allow (http_propfind,http_proppatch,http_mkcol,http_head,http_delete,http_
put,http_copy,http_move,http_lock,http_unlock,http_mkcalendar,http_report) user
= "anyone";
```

- Add the following line to the `/var/opt/SUNWwbsvr7/admin-server/config-store/machineC/config/magnus.conf` file:

```
Init fn="register-http-method" methods="MKCALENDAR"
```

- Deploy the configuration to the instance and restart the server instance.

```
wadm deploy-config machineC
wadm restart-instance --config=machineC
```

GlassFish Server Load Balancer Limitations

The GlassFish Server load balancer used in this deployment example can only do cookie-based stickiness routing, not IP-based routing. As a consequence, the session-based WCAP protocol does not work well with the load balancer. Requests from the same client are routed among the active instances that could be different than the authenticated session. Therefore, Convergence would not work against this HA deployment. To get around this limitation, an IP-based load balancer is required.

Back End High Availability: MySQL Async Replication

You can achieve a highly available, redundant MySQL back-end deployment by using asynchronous replication. You must configure GlassFish Server to use the replication driver, as explained in the *MySQL Connector/J Developer Guide* at:

<http://dev.mysql.com/doc/connector-j/en/connector-j-reference-configuration-properties.html>

See "[Configuring a High-Availability Database](#)" for instructions.

Note: MySQL Cluster with Calendar Server does not work reliably under load. Additionally, MySQL Cluster's data recovery mechanism is not satisfactory with Calendar Server. For these reasons, use MySQL asynchronous replication to achieve a redundant, highly available Calendar Server deployment.

Back End High Availability: Oracle Data Guard

For information on maximizing Oracle Database availability by using Oracle Data Guard and Advanced Replication, see the Oracle Database High Availability documentation at:

http://docs.oracle.com/cd/E11882_01/nav/portal_14.htm

Document Store High Availability

You can deploy a highly available document store. See "Configuring the Calendar Server Document Store" in *Calendar Server Installation and Configuration Guide* for details.

Configuring Calendar Server on a GlassFish Server Cluster

This chapter describes how to deploy and configure Oracle Communications Calendar Server on an Oracle GlassFish Server cluster. The GlassFish Server cluster feature enables you to create a collection of GlassFish Server instances that work together as one logical entity to provide Calendar Server with high availability through failure protection, scalability, and load balancing.

Note: This is an example deployment to illustrate the basics of setting up a GlassFish Server Cluster for Calendar Server.

Topics:

- [Prerequisites for Deploying Calendar Server on a GlassFish Cluster](#)
- [Example GlassFish Server Cluster Deployment Architecture](#)
- [Configuring a Calendar Server and GlassFish Server Cluster Deployment](#)
- [Limitations of This Deployment](#)

Prerequisites for Deploying Calendar Server on a GlassFish Cluster

This information assumes that you are familiar with the following tasks:

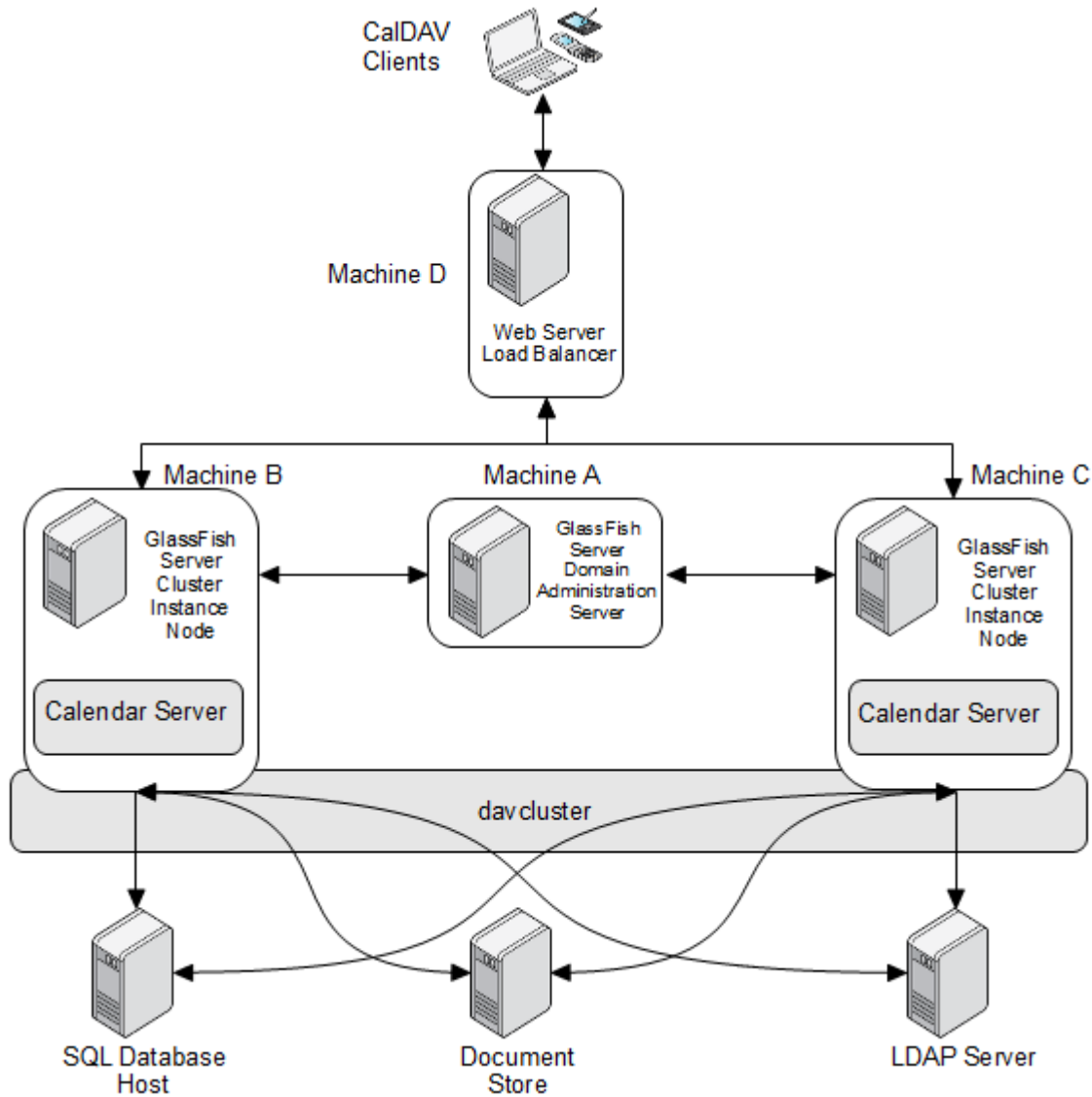
- Installing and configuring Calendar Server. See *Calendar Server Installation and Configuration Guide* for information about installing and configuring Calendar Server.
- Setting up clusters using GlassFish Server. See the following documents for more information:
 - *Oracle GlassFish Server High Availability Administration Guide*
 - *Oracle GlassFish Server Quick Start Guide*.

The hardware and software requirements for configuring Calendar Server in a GlassFish Server cluster are the same as configuring a non-cluster Calendar Server deployment. See the topic on system requirements in *Calendar Server Installation and Configuration Guide*. You also need to install a load balancer for the cluster. This example uses Oracle iPlanet Web Server (formerly known as Sun Java System Web Server 7.0). If you use a different load balancer, modify the appropriate steps in the procedures that follow.

Example GlassFish Server Cluster Deployment Architecture

Figure 14–1 shows an example cluster architecture consisting of two GlassFish Server nodes.

Figure 14–1 Calendar Server Cluster Deployment Architecture Example



This figure shows that a "davcluster" is formed from machines A and B, which are GlassFish Server cluster nodes for the Calendar Server front-end process. The Calendar Server front ends access the back-end database and document store, as well as the Directory Server. Machine A is the GlassFish Server domain administration server (DAS) that provides the shared location for the Calendar Server configuration information and data.

Configuring a Calendar Server and GlassFish Server Cluster Deployment

Configuring Calendar Server in a GlassFish Server cluster involves the following two high-level steps:

1. Creating a multi-instance GlassFish Server cluster
2. Deploying Calendar Server into the GlassFish Server cluster

The following section describes how to create a two-instance GlassFish Server cluster.

To Create a Two-Instance GlassFish Server Cluster

Follow the instructions described in "Deploying an Application to a Two-Instance Cluster" in *GlassFish Server Quick Start Guide* to set up a GlassFish Server cluster and load balancer. Instead of deploying the sample application in this example, you deploy Calendar Server. That is, replace the example section "To Deploy the Application and Configure the Load Balancer" with the next task, "[Deploying Calendar Server on a Two-Instance GlassFish Server Cluster.](#)"

Notes:

- If you plan on using a standalone domain administration server (DAS), and installing and configuring Calendar Server on one of the cluster nodes, make sure that you install GlassFish Server on the DAS machine in a network shared location accessible by the cluster nodes. Then, when you run the Calendar Server initial configuration script (**init-config**) on the cluster node and are prompted for the GlassFish Server installation path, use the network share path of the GlassFish Server 3 DAS installation.
 - If you plan to install and configure GlassFish as the **root** user, you must configure the network share so that no root squashing occurs to avoid permission issues. For example, on Solaris OS, use the **root=** option, and on Linux, use the **no_root_squash option** when sharing over NFS. However, if you plan to install Calendar Server on the DAS machine itself, then you do not need to install GlassFish Server on a network share location.
-
-

Deploying Calendar Server on a Two-Instance GlassFish Server Cluster

To deploy Calendar Server on a two-instance GlassFish Server cluster:

1. Following the "[Example GlassFish Server Cluster Deployment Architecture](#)" described previously, install Calendar Server on cluster node Machine B.

You must install the Unified Communications Suite distribution on the machine, run the **commpkg install** command, then select Calendar Server. For more information, see *Calendar Server Installation and Configuration Guide*.

2. Create an NFS share on DAS Machine A.

For both cluster nodes to share the same Calendar Server configuration and data, create a directory on the DAS Machine A, for example, **/ocucs**, and share it over NFS.

3. On Machine B and Machine C, mount the **/ocucs** share on Machine A to **/ocucs** so all machines can access the **/ocucs** path.

Note: If you run GlassFish Server as **root** user, configure the network share so no root squashing occurs.

4. On Machine B, configure the Calendar Server instance by running the Calendar Server **init-config** command.

For more information, see *Calendar Server Installation and Configuration Guide*.

5. When prompted by the **init-config** command, reply as follows:
 - a. To the prompt "Location to store application data and config (datadir)," use the network shared location **/ocucs/var/opt/sun/comms/davserver**.
 - b. To the prompt "Runtime user (cs.user)" and "Runtime group (cs.group)," specify the same user and group under which the GlassFish Server instance runs.
 - c. To the prompt "Application server install directory (appsrv.dir.install)," answer according to whether are configuring Calendar Server on the DAS machine, or a cluster node machine.

In this example, because you are configuring from the Machine B, use the network path of the GlassFish Server installation on the DAS machine A to which the current user has read and write access. That is, use **/home/gfuser/glassfish3**, assuming **/home** is the automounted network home directory for users.

- d. To the prompt "Application server target instance name (appsrv.target)," use the name of the cluster that was previously created in "[To Create a Two-Instance GlassFish Server Cluster](#)", for example, **davcluster**.
 - e. To the prompt "Calendar server access host (appsrv.http.host)," use the load balancer host for the cluster, for example, Machine D.
 - f. To the prompt "Calendar server access port (appsrv.http.port)," use the load balancer port for the cluster.
 - g. To the prompt "Application server admin server host (appsrv.admin.host)," use the DAS machine host name, for example, Machine A.
6. Answer the remaining **init-config** prompts.
7. Click **Configure Now** to configure Calendar Server.

8. Configure the load balancer on Machine D by adding the following lines to the *webserver7base/admin-server/config-store/machineD/default.acl* file:

```
acl "uri=/davserver";
allow (http_propfind,http_proppatch,http_mkcol,http_head,http_delete,http_
put,http_copy,http_move,http_lock,http_unlock,http_mkcalendar,http_report) user
= "anyone";
```

9. Add the following lines to the *webserver7base/admin-server/config-store/machineD/magnus.conf* file:

```
Init fn="register-http-method" methods="MKCALENDAR"
```

10. Run the Web Server **wadm deploy-config** command to deploy the changed configuration.

Note: If you install and configure Calendar Server on the DAS machine, when you run the **davadmin** command, use the **-H cluster-node** option for the command to succeed, where *cluster-node* can be any one of the nodes in the cluster.

Limitations of This Deployment

The GlassFish Server load balancer used in this deployment example can only do cookie-based stickiness routing, not IP-based routing. As a consequence, the session-based WCAP protocol does not work well with the load balancer. Requests from the same client are routed among the active instances that could be different than the authenticated session. Therefore, Convergence would not work against this HA deployment. To get around this limitation, use an IP-based load balancer.

Part III

Calendar Server Reference

Part III contains the following chapters:

- [Calendar Server Configuration Reference](#)
- [Calendar Server Command-Line Utilities](#)
- [Calendar Server Configuration Parameters](#)
- [Time Zone Database](#)

Calendar Server Configuration Reference

This chapter describes configuration files used by Oracle Communications Calendar Server. By default, these files are located in the *Calendar_home/config* directory.

Topics:

- [davserver.properties File](#)
- [davservercreds.properties File](#)
- [Document Store Server Configuration File](#)
- [certmap.conf File](#)
- [davadmin.properties File](#)
- [Notification Templates](#)

davserver.properties File

The **davserver.properties** file contains the main configuration settings. It consists of configuration parameters and their current values.

Caution: Do not edit this file by hand. Always use the **davadmin** command to set configuration parameters.

The format of the **davserver.properties** file is:

```
parameter=value  
parameter=value  
:  
:
```

davservercreds.properties File

The **davservercreds.properties** file contains the password configuration settings. It consists of configuration parameters that are passwords and their current values.

Caution: Do not edit this file by hand. Always use the **davadmin** command to set configuration parameters.

The format of the **davservercreds.properties** file is:

```
password_parameter=value
```

```
password_parameter=value
:
:
```

Document Store Server Configuration File

The `ashttpd.properties` file contains the document store configuration parameters.

[Table 15–1](#) describes the parameters in the `ashttpd.properties` file.

Table 15–1 *ashttpd.properties* File Parameters

Parameter	Description	Default Value
<code>service.host</code>	Server host	*
<code>service.port</code>	Server port number	8008
<code>store.datadir</code>	Data directory	<code>/var/opt/sun/comms/davserver</code>
<code>store.lockdir</code>	Lock directory	<code>/var/opt/sun/comms/davserver/lock</code>
<code>store.loglevel</code>	Log level	INFO
<code>store.sslkeystorepath</code>	Keystore for the server private key	<code>/var/opt/sun/comms/nabserver/config/dskeystore.jks</code>
<code>store.sslprotocols</code>	SSL protocols	TLSv1 TLSv1.1 TLSv1.2
<code>store.usessl</code>	Use SSL to communicate with document store client.	false

The format of the `ashttpd.properties` file is:

```
key=value
key=value
:
:
```

Each line in the `ashttpd.properties` file stores a single property. There is no space before and after *key* and *value*. If there are multiple network interfaces on the host and only one the server should bind to, specify that interface with the `service.host` config.

certmap.conf File

The `certmap.conf` file configures how a certificate is mapped to an LDAP entry.

The format of the `certmap.conf` file is:

```
certmap=name, name2
name.prop1=val1
name.prop2=val2
```

For more information on how to use this file, see the topic on the certificate mapper at:

<https://wikis.oracle.com/display/CommSuite/How+to+Set+Up+Certificate-based+Authentication+for+Convergence>

davadmin.properties File

You can provide options to the `davadmin` command by including them in the `davadmin.properties` file.

Table 15–2 describes the parameters in the `davadmin.properties` file.

Table 15–2 *davadmin.properties* File Parameters

Parameter	Description
<code>userid</code>	Specifies the GlassFish Administrator user ID.
<code>hostname</code>	Specifies the GlassFish Server host name.
<code>port</code>	Specifies the GlassFish administration port (JMX connector port).
<code>secure</code>	Specifies the path to the truststore file used for a secure connection (HTTPS) to Glassfish Server.
<code>dbtype</code>	Specifies the type of database, either <code>mysql</code> or <code>oracle</code> .
<code>dbhost</code>	Specifies the database host.
<code>dbport</code>	Specifies the database port.
<code>dbuserid</code>	Specifies the MySQL Server or Oracle Database user ID for database commands.
<code>sslprotocols</code>	Specifies the supported SSL protocols (<code>TLSv1</code> , <code>TLSv1.1</code> , and <code>TLSv1.2</code>) for the JMX proxy to communicate with management beans in the server.

The format of the `davadmin.properties` file is:

```
parameter=value
parameter=value
:
:
```

Notification Templates

Notification templates are files that contain pre-formatted notification messages. For example, `request.fmt` is used for scheduling request notification email message, while `sms.fmt` contains a short template for alarm SMS messages. See ["Using Calendar Server Notifications"](#) for more information.

Calendar Server Command-Line Utilities

This chapter provides information about the Oracle Communications Calendar Server command-line utilities.

Topics:

- [Overview of the Command-Line Utilities](#)
- [davadmin Utility](#)
- [davadmin account](#)
- [davadmin backend](#)
- [davadmin cache](#)
- [davadmin calendar](#)
- [davadmin calcomponent](#)
- [davadmin config](#)
- [davadmin db](#)
- [davadmin ldappool](#)
- [davadmin migration](#)
- [davadmin passfile](#)
- [davadmin vscan](#)
- [JConsole](#)
- [Summary of davadmin Changes by Release](#)
- [Deprecated Options](#)

Overview of the Command-Line Utilities

You use the **davadmin** command to administer Calendar Server. The **davadmin** command is installed in the *CalendarServer_home*/sbin directory with user or group bin/bin permissions.

Note: The **davadmin** command-line command administers aspects of the server and does not affect any LDAP entries.

davadmin Security

The **davadmin** command requires you to authenticate with a user name and password to be able to communicate with the server or database. You can use the **davadmin passfile** operation to store the necessary passwords in an encrypted wallet for use by subsequent **davadmin** commands. If you do not store passwords in the wallet, then you must enter them by using a no-echo prompt on the command line. See "[davadmin passfile](#)" for more information on how to create a file to store passwords.

Environment Variables

[Table 16–1](#) describes the environment variables that you can use with the various **davadmin** commands.

Table 16–1 *davadmin Environment Variable*

Environment Variable	Description
DAVADMIN_CLIFILE	Specifies the path to the bootstrap file. Can be used instead of the -F option.
DAVADMIN_ACCOUNT	Specifies the account information. Can be used instead of the -a account option.

davadmin Utility

Use the **davadmin** utility to administer Calendar Server.

Location

CalendarServer_homel/sbin

General Syntax

```
davadmin [ operation [ action ] ] [ option1 ] [ option2 ] ...
```

where:

- *operation* is the **davadmin** operation to run. See "[davadmin Operations](#)" for more information.
- *action* is the action that the specified operation performs, such as **create**, **delete**, **list**, and **modify**. Specifying an action is optional for certain operations.
- *option* is one or more command-line options that identify information that the operation needs and the specifics of what the operation does. For example, some options provide connection parameters, and the **-o** option specifies a configuration parameter that the **config** operation may list or modify. All options are optional if the **clifile** is used and accessed through the environment variable **DAVADMIN_CLIFILE**.

You can abbreviate an *operation*, an *action*, or both as long as they are unique in the command. For example, for the command **davadmin config list**, you can enter **davadmin c l**.

The default *action* for most commands is **list**. The default is used when you do not specify the *action*. For example, the following command lists the value of the **base.ldapinfo.cachesize** configuration parameter.

```
davadmin config -o base.ldapinfo.cachesize
```

Note: All words used for operations, actions, and options, and the components of *property=value*, are case-sensitive, typically lower-case.

Ways to Provide Options

You can provide options to the **davadmin** command by:

- Using the command line
- Using the clifile
- Including them in **davadmin.properties** file

Any user can create a **clifile**. Only the administrative user can use the **davadmin.properties** file. The **davadmin.properties** file is installed in the *CalendarServer_home/config* directory.

When you run the **davadmin** command, any option that you include on the command line takes precedence over any like option in the **clifile** or the **davadmin.properties** file. Use of the **clifile** or the **davadmin.properties** file is mutually exclusive. If you use the **clifile**, use it for any option that is not on the command line. If you run the **davadmin** command as the administrative user and do not supply a **clifile**, the **davadmin.properties** file is used for any option that is not on the command line.

The **davadmin.properties** file contains options for **userid**, **hostname**, **port**, **secure**, **dbhost**, **dbport**, and **dbuserid**.

Clifile Properties

Table 16–2 describes the possible properties in the bootstrap file (**clifile**).

Table 16–2 Clifile Properties

Property	Description
userid	GlassFish Administrator user ID.
usepasswordfile	Use the password file. Unless this property is empty, 'n', 'no' or 'false', the password file is used.
hostname	Server host name.
port	Glassfish administration port (JMX port).
secure	Path to the truststore file used for a secure connection (HTTPS).
dbuserid	MySQL Server or Oracle Database user ID.
dbhost	Host name where the database server resides.
dbhostname	Host name where the database server resides.
dbport	Port on dbhost for access to the database.
database	Specifies the name of the DAV store to be saved or updated.
docstore	Specifies the document store (remote store specified as <i>host:port</i> or local store by fully qualified path to root of document store)
migrationadminuser	Administrative user to authenticate to Calendar Server 6 host.
migrationserverport	Server and port information to connect to the Calendar Server 6 host from which data needs to be migrated. The format is <i>host:port</i> .

Common Options

Table 16–3 describes the options that are common to all **davadmin** operations.

Table 16–3 Common Options

Short Option	Long Option	Description	Required or Optional
-u <i>adminuserid</i>	--userid	MySQL Server or Oracle Database user ID for db commands, GlassFish Administrator user ID for all other commands.	Required unless you provide it through a CLI file by using the -F option, or you are displaying usage by using the -h option.
-W	--usepasswordfile	Get passwords from the password file. You use the The davadmin passfile command to create the password file. You can add passwords for the GlassFish Server administrative user, the migration server user, the database, and the document store.	Optional. If the password file does not exist or does not contain the needed password, you are prompted for the password.
-F <i>file</i>	--clifile	File with bootstrap information that you use to specify command-line options so that they do not have to be entered at the command line. Each line in the bootstrap file is in the form <i>property=value</i> . All property names and values are case-sensitive, typically lower-case. Some commands also have a -f , --file option, which provides additional batch input specific to those commands. For possible properties see the " Clifile Properties " table.	Required unless all necessary information is provided on the command line or in the davadmin.properties file. See " Ways to Provide Options " for more information on priority order of options, the clifile and the davadmin.properties file. A path to the clifile file can also be specified by the DAVADMIN_CLIFILE environment variable.
-H <i>host</i>	--hostname	Server host name.	Optional. Defaults to localhost .
-p <i>port</i>	--port	GlassFish administration port (JMX connector port) and MySQL Server or Oracle Database port for db commands. The GlassFish administration port can be found in the domain's domain.xml file or in the Administration Console (Configuration->Admin Service->system).	Optional. Defaults to 3306 for db commands and 8686 for other commands.
-s <i>path</i>	--secure	Path to the truststore file used for a secure connection (HTTPS).	Optional. Required if GlassFish is running in secure mode. Not applicable for db commands.
-e	--detail	Verbose output. Mostly used if a command returns an error.	Optional.
-q	--quiet	Quiet mode for scripts.	Optional.
-h	--help	Help for that particular operation.	Optional.
-V	--version	Lists version of davadmin utility. (Checks the local package version on disk, which could be different than what has been deployed to GlassFish Server, for example, in the case where a patch was added but the init-config command has not yet been run.)	Optional. Usable only by itself and not with other options.

Each operation also has its own specific options, as shown in the following sections.

davadmin Operations

Table 16–4 describes the **davadmin** operations.

Table 16–4 *davadmin Class of Operations*

Argument	Description
version	Displays version of the server. The GlassFish Server is queried for the version of Calendar Server deployed.
account	Performs operations that affect the entire user or resource account.
backend	Adds information for an additional back-end calendar store.
cache	Performs operations on various Calendar Server caches.
calendar	Performs calendar collection operations, such as create a collection, modify a collection, or delete a collection.
calcomponent	Performs resource operations, such as listing resources that meet a specified criteria, importing resources, or deleting resources.
config	Performs configuration operations, such as print a particular option, set a particular option, or list all options. Some configuration operations require that you restart Calendar Server. The davadmin config modify command informs you if the change requires you to restart Calendar Server to take effect. To stop and start Calendar Server, see "Stopping and Starting Calendar Server Services" for details.
db	Performs database related operations, like backing up and restoring the database.
ldappool	Performs ldappool operations, including creating, listing, and modifying LDAP pools.
migration	Performs migration of Calendar Server 6 data to Calendar Server Server 7 Update 1 and greater.
passfile	Creates, deletes, lists, or modifies passwords in the password file.
vscan	Performs virus scanning operations.

Each operation takes various command-line options. The common options used by all **davadmin** operations are described in [Table 16–3](#).

Note: Any option value that contains special characters or spaces must be enclosed in quotes (""") so that it is passed "as is" to the **davadmin** command. For example:

```
davadmin config modify -o base.ldapinfo.ugldap.binddn -v
"cn=Directory Manager"
```

Note: If a portion of an option that is enclosed in quotes also needs to be quoted, you must use single quotes around that portion. For example:

```
davadmin calendar modify -n calendar -y "displayname='A new
calendar name',acl=@:r"
```

Tool-Only Options

Two options, **-V** and **-h**, can be used without any operation specified. The **-V** option prints the version of the command-line utility. The **-h** option prints the general usage.

Exit Code

The tool exits with exit code 0 on success and 1 on failure.

davadmin account

Use this command to perform operations that affect the entire user or resource account.

Syntax

```
davadmin account [ create | delcomponents | delete | list | modify |
                 repair | subscribe | unsubscribe | upgrade ]
                 [-p port] [-s path] [-a account] [-g uniqueid (delete only)]
                 [-y property=value[,property=value...]] [-f file]
                 [-B ldapbaseuri] [-R ldapfilter] [-d days]
                 [-c collection_path | -C collections_file_path]
                 [-m] [-o] [-v (list only)] [-e] [-r] [-q] [-h]
```

account Operation

Table 16–5 describes the actions for the account operation.

Table 16–5 Actions for account Operation

Command	Description
create	Creates an account for user who has been provisioned in the LDAP Directory Server. The user must have an email address.
delcomponents	Deletes components from all of the calendars belonging to an account or a set of accounts. Use the -d option to specify deletion of all components older than this number of days.
delete	Deletes an account.
list	Lists properties of an account. The list command displays managed calendars for an account. These are all the calendars for which the account is the owner or has "all" rights. Also, list displays the users' subscribed calendars list. list is the default action, if it is not included on the command line, for most commands. You can use the davadmin account list command without the -a option to list all current users in the Calendar Server database. You can get either a simple list, which contains one user per line, or a detailed list, which contains complete information about the user's account. The options affected by this change are -a , -f , -B , and -v .
modify	Modifies an account.
repair	Repairs the user's email address in the database entries after an LDAP email change occurs. When used with the -o option, repair updates the owner lists of all accounts.
subscribe	Subscribe to a calendar belonging to another user. That other user must grant the requesting user access before this can be done.
unsubscribe	Remove a calendar from a user's subscription list.

Options for account Operation

Table 16–6 describes the options for the **account** operation.

Table 16–6 Options for account Operation

Short Option	Long Option	Description
-a <i>account</i>	--account	Required. Principal account information provided as email address. You can also supply the account information with the DAVADMIN_ACCOUNT environment variable.
-y <i>property</i>	--property	Comma-separated list of all <i>property=value</i> options for the specified calendar. Possible properties include: acl - The scheduling privileges set on the account. See " Administering Calendar Server Access " for more information about ACLs. set-ace - Sets one or more individual ACEs in the ACL. A semicolon separated list of ACEs. remove-ace - Removes one or more individual ACEs from the ACL. A semicolon separated list of ACE principals. ACE principals are in the form: @, @domain,group@domain, or user@domain. notifemail - Email notification enable flag. 0 = disabled, 1 = enabled notifrecipients - Recipients of email notifications. Multiple values are separated by a space. delegate_notifaddr - Accounts that are delegates for this account. Multiple values are separated by a space. owner - The new owner of the resource. This option is not available for user owned accounts. owner updates the owner lists of the old owner and the new owner with the right list of resource accounts they own. attendanceflag - Flag controlling behavior on invitation. Possible values are: 0 - no autoaccept, no booking conflict check, no recurrence check on invitations. 1 - autoaccept invitations 2 - autodecline if invitation results in booking conflict. 3 - autoaccept invitation and autodecline on booking conflict. 4 - autodecline recurring meeting invitations. 5 - autoaccept invitations and autodecline recurring meeting invitations 6 - autodecline recurring invitations and invitations that cause a booking conflict. 7 - autoaccept invitations, autodecline recurring invitations and invitations that cause a booking conflict.
-f	--file	Local input file with one line for each account, for batch operation. Each line has the format <i>user};properties</i> , where <i>properties</i> is a comma-separated list of property settings as specified in the -y option.
-B	--ldapbaseuri	Base URI in LDAP.
-R	--ldapfilter	User search filter in LDAP. Default is (objectClass=icsCalendarUser)

Table 16–6 (Cont.) Options for account Operation

Short Option	Long Option	Description
-r	--force	Force the operation (do not prompt for confirmation).
-h	--help	Displays davadmin account usage help.
-c	--collectionuri	The full path of a collection to be added to a user's subscription list. For example, <code>/home/user@example.com/mycal1/</code> . Be sure to include the <code>/</code> at the end of the path.
-C	--collectionuris	The full path to a file which holds full paths of collections to be added to a user's subscription list. Each line is a path. For example: <code>/home/user2@example.com/cal1/</code>

Table 16–7 describes the options for the **delete** operation.

Table 16–7 Options for delete Operation

Short Option	Long Option	Description
-a <i>account</i>	--account	Required. Principal account information provided as email address. You can also supply the account information with the <code>DAVADMIN_ACCOUNT</code> environment variable.
-d	--days	Number of days. Delete the components older than these many number of days. Applies only to the davadmin account delcomponents command.
-g <i>uniqueID</i>	NA	The principal account described by the database uniqueID , if -a fails. Normally you run davadmin account delete while the user is still defined in LDAP, so the higher level delete functionality can identify the user. In the incorrect case where the user is no longer in LDAP and the normal command fails due to User Not Found, you can delete the user's database data by specifying -g uniqueID , where <i>uniqueID</i> is the user's old LDAP uniqueID. Only use -g when users are no longer defined in LDAP.

Table 16–8 describes the options for the **repair** operation.

Table 16–8 Options for repair Operation

Short Option	Long Option	Description	Default?
-m	--email	Repairs the user's email address after an email change. Valid only for the repair action. Specify users with either the -a or -f options.	Yes
-o	--ownerlists	Updates the owner lists of all accounts. Valid only for the repair action.	No

Table 16–9 describes the options for the **davadmin account list** command.

Table 16–9 davadmin account list Options

Short Option	Long Option	Output	Comments
-a	--account	The detailed account information for this user. If the user is not in the database, the system displays an "Unknown user:" message.	The DAVADMIN_ACCOUNT environment variable, if set, is not used in place of the -a option. If -a is not supplied on the command line, a list of all users in the database will be displayed.
-f	--file	A list of the users in the file. The system displays an "Unknown user:" tag before the names of users in the file that are not in the database.	No comments.
-Buri	--ldapbaseuri	Base URI in LDAP. Searches LDAP for a set of users and then displays the users from that list that exist in the database.	No comments.
-v	--verbose	Detailed information is displayed about each of the users in the database.	Used with the -f and -B options.

davadmin account Examples

- To list the account for a user:

```
davadmin account list -a john.smith@example.com
```

Note: If a calendar name contains a special character, such as a space, **account list** does not display the expanded character, but **calendar list** does. For example, for a calendar named "a b c," **account list** displays the calendar as "a b c," but **calendar list**, because it uses the URI, displays `"/home/user@example.com/a%20b%20c"`. In this case, the expanded character "%20" that represents the space is displayed.

- To create an account for **user1@example.com**:

```
davadmin account create -a user1@example.com
```

- To create an account for **user1** under the LDAP base **o=isp** (the user has to be previously provisioned in LDAP):

```
davadmin account create -B "o=isp" -R "uid=user1"
```

- To create an account for all users whose uid starts with "user1" (the users have to be previously provisioned in LDAP) and have all of their notifemail properties set to disabled:

```
davadmin account create -B "o=isp" -R "uid=user1*" -y "notifemail=0"
```

- To create the calendar account with default calendar for a provisioned resource:

```
davadmin account create -a resource1@example.com
```

- To delete an account:

```
davadmin account delete -a john.smith@example.com
```

Note: This deletes the account from the calendar database. To completely remove the account from LDAP, see "[Removing Calendar Users.](#)"

- To delete a user's calendar entries, with all events and todos prior to and including today:

```
davadmin account delcomponents -a caluser31@example.com -d 0
```

- To set the scheduling rights on John Smith's account to allow Jane Doe to schedule events and all other users to just do free busy checks:

```
davadmin account modify -a john.smith@example.com -y  
acl="jane.doe@example.com:s;@:f"
```

- To clear a resource's owner field:

```
davadmin account modify -a resource1@example.com -y owner=""
```

After running this command, the resource then has no owner.

- To repair the owner list for a resource account:

```
davadmin account repair -o -a calresource@example.com
```

- To set the value of two individual ACEs in the ACL:

```
davadmin account modify -a user30@example.com -y  
set-ace="user19@example.com:s;user20@example.com:f"
```

- To remove an individual ACE from the ACL:

```
davadmin account modify -a user30@example.com -y remove-ace=user19@example.com
```

- To create two accounts and set their properties by using an input file:

Input File:

```
user1@example.com:notifemail=0,attendanceflag=5  
user2@example.com:notifemail=1,notifrecipients=user4@example.com;user3@example.  
com
```

Command:

```
davadmin account create -f input_file.txt
```

- To modify the previous two accounts and set their properties by using an input file:

```
davadmin account modify -f input_file.txt
```

davadmin backend

Use this command to add information for an additional back-end calendar store.

Syntax

```
davadmin backend [ create | list | purge ]
```

```

[-u id] [-W] [-F clifile] [-H hostname]
[-p port] [-s path] [-n name] [-j jndiname]
[-d dbdir] | [-S ashost] [-P asport]
[-e] [-q] [-h]

```

backend Operation

Table 16–10 describes the actions for the **backend** operation.

Table 16–10 Actions for backend Operation

Command	Description
create	Configures a new back-end calendar store configuration on the front end.
list	Lists the back-end calendar store(s). This is the default action if not included on the command line.
purge	Immediately purges calendar data marked for expiration from Calendar Server back-end database(s).

Options for backend Operation

Table 16–11 describes the options for the **backend** operation.

Table 16–11 Options for backend Operation

Short Option	Long Option	Description	Required or Optional
-n	--name	Name of the backend.	Required for create command.
-j	--jndiname	The JNDI name of the JDBC resource of the back end.	Required for create command.
-d	--dbdir	The path to the local document store directory.	Required for create command and if document store is local.
-S	--ashost	The host name of the remote document store.	Required for create command and if document store is remote.
-P	--asport	The port number of the remote document store.	Required for create command and if document store is remote.

davadmin backend Examples

- To list the back ends:


```
davadmin backend list -u admin
```
- To create a new back end with a local document store:


```
davadmin backend create -u admin -n store1 -j jdbc/store1 -d /var/cs7/store1
```
- To create a new back end with a remote document store:


```
davadmin backend create -u admin -n store2 -j jdbc/store2 -S store-2.example.com -P 8008
```

Caution: The **davadmin backend create** command alone is not enough to completely configure a new back-end store. See ["Monitoring and Managing Calendar Server"](#) for more information on configuring multiple Calendar Server back-end hosts.

- To immediately purge calendar data that has been marked for expiration from the default back end:

```
davadmin backend purge -u admin -n defaultbackend
```

davadmin cache

Use this command to perform operations on various Calendar Server caches.

Syntax

```
davadmin cache [ clear ]
                [-u id] [-W] [-F clifile] [-H hostname]
                [-p port] [-s path] [-t cache]
                [-d dbdir] [-S ashost] [-P asport]
                [-e] [-q] [-h]
```

cache Operation

[Table 16–12](#) describes the action for **davadmin cache** command.

Table 16–12 Action for cache Operation

Command	Description
clear	Clears the various Calendar Server caches.

Options for the cache Operation

[Table 16–13](#) describes the option for the **cache** operation.

Table 16–13 Option for cache Operation

Short Option	Long Option	Description
-t	--cachelist	Optional. Comma-separated list of caches, possible values are: <ul style="list-style-type: none"> ■ acl - ACL string cache corresponding to a URI and LDAP subject entry corresponding to each calendar collection. Otherwise cleared according to the configuration options of davcore.acl.aclcachesize and davcore.acl.aclcachettl. ■ domainmap - Cache of information on domains retrieved from LDAP. Otherwise cleared according to the configuration options of base.ldapinfo.cachesize and base.ldapinfo.cachettl. ■ ldapauth - Cache of logged-in principals' login ID and passwords (encrypted). Otherwise cleared according to the configuration options of base.ldapinfo.cachesize and base.ldapinfo.cachettl. ■ uri - Cache mapping LDAP subjects and URIs. Otherwise cleared according to the configuration options of davcore.uriinfo.ldapcachesize and davcore.uriinfo.ldapcachettl.

davadmin calendar

Use this command to perform calendar collection operations, such as creating a collection, modifying a collection, or deleting a collection.

Syntax

```
davadmin calendar [ create | modify | delete | list ]
                  [-u id] [-W] [-F clifile] [-H hostname]
                  [-p port] [-s path] [-a account] [-n name] [-v]
                  [-y property=value[,property=value...]] [-f file]
                  [-r] [-e] [-q] [-h]
```

calendar Operation

Table 16–14 describes the actions for the **calendar** operation.

Table 16–14 Actions for calendar Operation

Command	Description
create	Creates a calendar collection. Autocreates the account, if it does not exist.
modify	Modifies a calendar collection.
delete	Deletes a calendar collection.
list	Lists an account's calendars or details of a particular calendar (if the -n option is provided). This is the default action if not included on the command line.

Options for calendar Operation

Table 16–15 describes the options for the **calendar** operation.

Table 16–15 Options for calendar Operation

Short Option	Long Option	Description
-a <i>account</i>	--account	Required. Principal account information provided as email address. You can also supply the account information with the DAVADMIN_ACCOUNT environment variable.
-n <i>collection</i>	--name	The unique calendar collection name that corresponds to the last part of the calendar collection URI. For the default calendar, <i>collection</i> it is always calendar . This name might not always correspond with the calendar's display name. If you are unsure of the calendar name, use the davadmin calendar list command to list all calendars and find out the calendar names.

Table 16–15 (Cont.) Options for calendar Operation

Short Option	Long Option	Description
<code>-y property</code>	<code>--property</code>	<p>Comma-separated list of all <i>property=value</i> options for the specified calendar. Possible properties include:</p> <p>set-ace - Specifies a semicolon separated list of ACEs to add or modify to the calendar permissions (ACL).</p> <p>remove-ace - Specifies a semicolon separated list of ACE principals that are to be removed from the calendar permissions (ACL). The ACE principal is the user, group, domain, or all portion of the ACE not including the ":" and permission.</p> <p>displayname - The calendar name. Defaults to the name given with the <code>-n</code> option.</p> <p>calendar-description - Description string. No default.</p> <p>supported-calendar-component-set - Space-separated list of supported components. The default is VEVENT VTOD VFREEBUSY. This option is only available for creation of secondary calendars. It cannot be used for creation of the default calendar.</p> <p>wcaptzid - The time zone tzid set on the calendar, for example, America/Los_Angeles.</p> <p>acl - The access control string set on the calendar. See "Administering Calendar Server Access" for more information about ACLs.</p>
<code>-f file</code>	<code>--file</code>	<p>Local commands input file for batch operation. Each line has colon-separated entries for account information, calendar name, and property list. For example:</p> <pre>user1@example.com:testcal:calendar-description=user1's test cal</pre>
<code>-h</code>	<code>--help</code>	Displays davadmin calendar usage help.

davadmin calendar Examples

- To create an additional calendar with the given name for the specified user account:

```
davadmin calendar create -a john.smith@example.com -n mypersonalcalendar
```

The name, which is a required parameter, builds the new calendar's URI and sets its display name. This is the name that would be used for the `-n` option for any further **davadmin calendar** commands. This cannot be changed. The display name can be modified later by using the **davadmin calendar modify** command with the `-y displayname` option.

- To list a summary of the calendar specified by name:

```
davadmin calendar list -a john.smith@example.com -n mypersonalcalendar
```

- To delete a calendar specified by name:

```
davadmin calendar delete -a john.smith@example.com -n mypersonalcalendar
```

- To set the access rights on John Smith's default calendar to give Jane Doe all rights and only read rights to everyone else:

```
davadmin calendar modify -a john.smith@example.com -n calendar -y
acl="jane.doe@example.com;a;@:r"
```

davadmin calcomponent

Use this command to perform resource operations, such as listing resources that meet a specified criteria, importing resources, or deleting resources.

Syntax

```
davadmin calcomponent [ list | delete | import | export ]
                    [-u id] [-W] [-F clifile] [-H hostname]
                    [-p port] [-s path] [-a account] [-n name]
                    [-y property=value[,property=value...]] [-i uri]
                    [-m path | -x path] [-l logpath] [-t yes | no ]
                    [-e] [-r] [-q] [-h]
```

calcomponent Operation

Table 16–16 describes the actions for the **calcomponent** operation.

Table 16–16 Actions for calcomponent Operation

Command	Description
list	Displays a summary of all of the resources in a calendar or the specifics of one resource. This is the default action if not included on the command line.
delete	Deletes a resource or all of the resources in a calendar.
import	Imports resource data into a calendar.
export	Exports resource data from a calendar.

Options for calcomponent Operation

Table 16–17 describes the options for the **calcomponent** operation.

Table 16–17 Options for calcomponent Operation

Short Option	Long Option	Description
-a <i>account</i>	--account	Required. Principal account information provided as email address. You can also supply the account information with the DAVADMIN_ACCOUNT environment variable.
-n <i>collection</i>	--name	The unique calendar collection name that corresponds to the last part of the calendar collection URI. The default value used is the principal's default calendar. If you are unsure of the calendar name, use the davadmin calendar list command to list all calendars and find out the calendar names.
-y <i>property</i>	--property	Comma-separated list of all <i>property=value</i> options for specified calendar. Possible properties include: type - The component type or types. Possible values are VEVENT and/or VTODO . If you use both VEVENT and VTODO , enclose them in double quotes and separate them with a space. start - The start of a time range used in the search. The format of this value is <i>yyyymmddThhmmssZ</i> . This value is in Zulu time. (The T is a separator between the day and time.) end - The end of a time range used in the search. The format of this value is <i>yyyymmddThhmmssZ</i> . This value is in Zulu time. (The T is a separator between the day and time.)
-h	--help	Displays davadmin calcomponent usage help.
-i	--uri	URI of resource to request entire content.

Table 16–17 (Cont.) Options for calcomponent Operation

Short Option	Long Option	Description
-r	--force	Forces a delete operation so that you are not prompted for confirmation. This option is generally needed for scripts.
-m	--import-path	Path to the file on the server machine, containing data to be imported.
-x	--export-path	Path to the file where the exported data is to be stored.
-l	--logpath	Path to where the log directory is located. The davadmin calcomponent import command enables the import to continue even if an error occurs on an item being imported.

davadmin calcomponent Examples

- To list the calendar resources in the user's default calendar:


```
davadmin calcomponent list -a john.smith@example.com
```
- To display the contents of a particular calendar resource:


```
davadmin calcomponent list -a john.smith@example.com -i 23454-333-3-3333.ics
```
- To list only a calendar's tasks:


```
davadmin calcomponent list -a john.smith@example.com -y type=VTOD0
```
- To list all calendar resources from March 3, 2009 through March 4, 2009:


```
davadmin calcomponent list -a john.smith@example.com -y
start=20090303T070000Z,end=20090305T065959Z
```
- To delete the event resources from March 3, 2009 through March 4, 2009, assuming that the local time zone is Pacific Time:


```
davadmin calcomponent delete -a john.smith@example.com -y
type=VEVENT,start=20090303T070000Z,end=20090305T065959Z
```
- To delete a user's calendar entries, with some start/end date range:


```
davadmin calcomponent delete -a caluser31@example.com -y
start=20090701T000000Z,end=20090720T000000Z
```

davadmin config

Use this command to perform configuration operations, such as display a particular parameter, set a particular parameter, or list all parameters. Some configuration operations require that you restart Calendar Server. The **davadmin config modify** command informs you if the change requires you to restart Calendar Server to take effect.

Syntax

```
davadmin config [ list | modify ]
                [-u id] [-W] [-F clifile] [-H hostname]
                [-p port] [-s path] [-o property] [-v value]
                [-d] [-f file] [-e] [-q] [-h] [-M]
```


config Operation

Table 16–18 describes the actions for the **config** operation.

Table 16–18 Actions for config Operation

Command	Description
list	Lists all configuration settings. This is the default action if not included on the command line.
modify	Modifies a configuration setting.

Options for config Operation

Table 16–19 describes a list of options for **config** operations that can be provided, unless you are displaying usage by using the **-h** option. See "Calendar Server Configuration Parameters" for the complete list of configuration parameters.

Table 16–19 Options for config Operation

Short Option	Long Option	Description
-o <i>option</i>	--option	Configuration option name. Gets the optional value if specified without -v . Sets the option value if specified with a -v .
-v <i>value</i>	--value	Configuration option value.
-f <i>file</i>	--file	Local file with list of configuration <i>option=value</i> entries for setting. Pay attention to backslashes included in this input file. Backslashes are treated as an escape character for the next character in the line. For a single backslash to be properly interpreted in a string, you must precede each backslash with another backslash; that is, use an additional backslash. For example, to include the string <code>"^/principals/\z"</code> , you would use <code>"^/principals/z"</code> . This is due to the way that Java reads in properties files. For more information, see the <code>load(Reader reader)</code> method of the <code>java.util.Properties</code> class at http://docs.oracle.com/javase/6/docs/api/index.html
-M	--modonly	Lists the modified configuration properties (non-default values).
-d	--default	Sets the value to the default when used with the modify action. Lists the default value when used with the list action.
-h	--help	Description of config option if specified with -o . Otherwise, usage of davadmin config .

davadmin config Examples

- To show all configuration parameters:


```
davadmin config list
```
- To show all configuration parameters (prior to Calendar Server 7 Update 2):


```
davadmin config -l
```

davadmin config (since list is default)
- To show the current setting for the error log:

```
davadmin config -o log.dav.errors.loglevel
```

- To set the error log to accept "finest" messages:

```
davadmin config modify -o log.dav.errors.loglevel -v FINEST
```

- To list the default setting:

```
davadmin config list -o davcore.acl.defaultschedulingacl -d -u admin
Enter Admin password:
davcore.acl.defaultschedulingacl: @:s
```

- To modify to the default setting:

```
davadmin config modify -o davcore.acl.defaultschedulingacl -d -u admin
Enter Admin password:
davadmin config list -o davcore.acl.defaultschedulingacl -u admin
Enter Admin password:
davcore.acl.defaultschedulingacl: @:s
```

davadmin db

Use this command to perform database related operations, such as backing up and restoring the database, and upgrading the database schema.

Unlike other **davadmin** commands that communicate with GlassFish Server, the **davadmin db** commands communicate directly with the back-end database, and thus require that you specify the database host name, port, and password.

Although the **davadmin db** commands are not related to GlassFish Server like the other **davadmin** commands, **davadmin db** commands do still use parameter values in the **davadmin.properties** file if applicable.

Because each database back end is associated with a database host name, port, document store, and so on, in a multiple back-end deployment, use a unique **clifile** (specified with the **-F** option) for each back end in the deployment.

In a non-default deployment or multiple back-end deployment, properly define options such as (**-d database**) and (**-u dbuser**), which might need to use specific and not default values.

Syntax

```
davadmin db [ backup | init | list | restore | schema_version |
             schema_fullupgrade | schema_preupgrade ]
             [-h] [-e] [-W] [-t dbtype] [-H dbhost] [-p dbport] [-F clifile]
             [-u dbuserid] [-d database] [-s truststore] [-b blockfactor]
             [-D domain] [-a account_mail] [-T token] [-O] [-i path]
             [-c] [-A docstore] [-z preupgradefunction] [-k backup_file]
```

db Operation

Table 16–20 describes the actions for the **davadmin db** operation.

Table 16–20 Actions for db Operation

Command	Description
backup	Backs up a database.

Table 16–20 (Cont.) Actions for db Operation

Command	Description
init	Completely initializes the database. Caution: All data will be lost.
list	List contents of a backup file. This is the default action if not included on the command line.
restore	Restores the contents of a database.
schema_version	Displays version information for the database, connector, and product schema number.
schema_fullupgrade	Provides an optional way to perform a full upgrade of the database schema. For more information about upgrading database schema and upgrading Calendar Server, see "Upgrading Calendar Server" in <i>Calendar Server Installation and Configuration Guide</i> .
schema_preupgrade	Provides an optional way to perform a pre-upgrade on the database schema. For more information about upgrading database schema and upgrading Calendar Server, see "Upgrading Calendar Server" in <i>Calendar Server Installation and Configuration Guide</i> .

Caution: Do not run either the **schema_fullupgrade** or **schema_preupgrade** without fully understanding the impact on your Calendar Server deployment.

The **davadmin db backup**, **list**, and **restore** commands require that you specify the associated document store by using the **-A** option, or the **docstore** option in the **clifile**.

Note: If you are using a remote document store, you must set the document store password on the Calendar Server host by using the **davadmin passfile** command and that password must match the one set for the remote document store. This password is used whenever the backup or restore commands access the remote document store.

Options for db Operation

Table 16–21 describes the options for the **db** operation (in addition to the common options).

Table 16–21 Options for db Operation

Short Option	Long Option	Description	Available for Following Actions
-d	--database	Specifies the name of the DAV store to be saved or updated. The default is caldav . For MySQL Server, this is the database name. For Oracle Database, this is the network service name (not SID nor pdb name).	backup, restore, list
-H	--dbhost	Specifies the database host. The default is localhost .	All
-p	--dbport	Specifies the database port. The default is 3306.	All

Table 16–21 (Cont.) Options for db Operation

Short Option	Long Option	Description	Available for Following Actions
-u	--dbuserid	Specifies the database user. For MySQL Server, this is the connecting user name. For Oracle DB, this is the user/schema name.	All
-k	--bkfile	Specifies the path of the file where the database information is to be saved. Required.	backup, restore, list
-b	--bkfactor	Specifies blocking factor used during backup. The default is 20.	backup, restore, list
-T	--token	Specifies the incremental backup token or start time in milliseconds.	backup
-D	--domain	Domain name for per domain backup.	backup
-a	--account	User account email value for per user backup.	backup
-i	--ipath	Specifies the internal path for partial list or restore.	restore, list
-c	--contents	Lists the resources and header.	list
-A	--docstore	Specifies the document store (remote store specified as <i>host:port</i> or local store by fully qualified path to root of document store).	backup, restore, list
-t	--dbtype	Specifies the type of database, either mysql or oracle . The default is mysql .	All

Table 16–21 (Cont.) Options for db Operation

Short Option	Long Option	Description	Available for Following Actions
-O	--overwrite	Overwrites existing data.	backup, restore
-s	--dbsecure	Supplies the path to the trustStore file that contains the SSL certificate for secure communications with the remote document store.	backup, restore
-z	--dbupgradefunction	<p>Specifies to run the pre-upgrade function(s) on the database.</p> <p>Caution: Do not run schema_preupgrade without fully understanding the impact on your Calendar Server deployment. For more information, see "Upgrading Calendar Server" in <i>Calendar Server Installation and Configuration Guide</i>.</p> <p>The pre-upgrade functions are:</p> <ul style="list-style-type: none"> ▪ services-up - Executes all pre-upgrade functions that can be run with old services up. (Online DDL) Otherwise and most commonly, pre-upgrade functions must be run with services shut down. ▪ services-down - Executes all pre-upgrade functions that cannot be run with services up. ▪ all - Executes all available pre-upgrade functions. Services should be shut down. <p>For a list of available functions by release, see "Preupgrade Functions" in <i>Calendar Server Installation and Configuration Guide</i>.</p> <p>Unless otherwise specified, never run pre-upgrade functions with services up. In addition, always back up your database before upgrading.</p> <p>Preupgrade functions are listed for each release. Some function names execute multiple preupgrade functions.</p>	schema_preupgrade

davadmin db Examples

- To perform a full database backup:
`davadmin db backup -k backup_file`
- To perform a full backup for a particular user:
`davadmin db backup -k backup_file -a john.smith@example.com`
- To perform an incremental backup:
`davadmin db backup -k backup_file -T token obtained from last full backup`
- To perform a full backup for a particular domain:
`davadmin db backup -k backup_file -D sesta.com`

- To list the contents of the backup file:

```
davadmin db list -c backup_file
```

When the **davadmin db list -c** command retrieves backup file content, it goes through the checksums and is thus a way to verify the structure of the backup file itself.

- To perform a restore from a backup file:

```
davadmin db restore -k backup_file
```

- To restore from a backup file and overwrite a calendar:

```
davadmin db restore -O -e -W -k /export-filepath -i  
"hosted.domain/mail:given.surname@hosted.domain/" -H mysqlcalhost -A matching_  
document_store_host:8007 > /log_output_file
```

- To restore only the default 'calendar':

```
davadmin db restore -O -e -W -k /export-filepath -i  
"hosted.domain/mail:given.surname@hosted.domain/calendar/" -H mysqlcalhost -A  
matching_document_store_host:8007 > /log_output_file
```

- To restore only a calendar named **Soccer**:

```
davadmin db restore -O -e -W -k /export-filepath -i  
"hosted.domain/mail:given.surname@hosted.domain/Soccer/" -H mysqlcalhost -A  
matching_document_store_host:8007 > /log_output_file
```

- To back up using SSL and the trustStore file:

```
davadmin db backup -k /tmp/backup_file -O -A docstore_host.example.com:8008 -s  
/my_home/my_truststore -u mysql
```

- To execute a database schema preupgrade:

```
davadmin db schema_preupgrade -z preupgrade_function
```

This command executes one preupgrade function. A preupgrade function is an upgrade change to the database, which can be run before the formal upgrade. This command does not change the database schema version.

- To execute all available preupgrade functions:

```
davadmin db schema_preupgrade -z all
```

Prior to running this command, ensure that all services are shut down.

- To execute all preupgrade functions that cannot be run with services down:

```
davadmin db schema_preupgrade -z services-down
```

davadmin ldappool

Use this command to perform LDAP pool operations, including creating, listing, and modifying LDAP pools.

Syntax

```
davadmin ldappool [ create | delete | list | modify ]  
[-u id] [-W] [-F clifile] [-H hostname]  
[-p port] [-s path] [-n poolname]
```

```
[-y property=value [,property=value...]] [-f file]
[-r] [-h]
```

ldappool Operations

Table 16–22 describes the actions for the **ldappool** operation.

Table 16–22 Actions for ldappool Operation

Command	Description
create	Creates an LDAP pool and sets its configuration parameters.
modify	Modifies the LDAP pool's configuration parameters.
delete	Deletes an LDAP pool.
list	Lists an LDAP pool's configuration, or all LDAP pools' configuration. (This is the default action.)

Options for ldappool Operation

Table 16–23 describes the options for the **ldappool** operation.

Table 16–23 Options for ldappool Operation

Short Option	Long Option	Description
-n <i>poolname</i>	--name	The name of the LDAP pool.
-y <i>property</i>	--property	Comma-separated list of all <i>property=value</i> options for the specified LDAP pool. Properties are appended to base.ldappool.name to produce the configuration parameters for the LDAP pool. Possible properties include: ldaphost - Space-delimited list of host names. Each host name can include a trailing colon and port number. ldapport - Port number to which to connect. Ignored for any host name which includes a colon and port number. ldapusessl - Use SSL to connect to the LDAP host. Value can be true or false . binddn - Distinguished name to use when authenticating. bindpassword - Password to use when authenticating. ldappoolsize - Maximum number of connections for this pool. ldaptimeout - Timeout, in seconds, for all LDAP operations. ldappoolrefreshinterval - Length of elapsed time, in minutes, until the failover Directory Server reverts back to the primary Directory Server. If set to -1 , no refresh occurs.
-f <i>file</i>	--file	Local input file with one line for each account, for batch operation, containing lines in the form <i>pool_name:property_list</i> . The properties are the same ones available for the -y option. For delete operations, only <i>pool_name</i> is used.
-r	--force	Force the operation (do not prompt for confirmation).
-h	--help	Displays davadmin ldappool usage help.

davadmin ldappool Examples

- To create an LDAP pool named **myldap**:

```
davadmin ldappool create -n myldap -y
"ldaphost=host1.example.com,ldapport=389,binddn='cn=Directory
```

```
Manager',bindpassword=mypassword"
```

- To update an LDAP pool by using properties from a file:

```
davadmin ldappool modify -n myldap -f /tmp/update_pool.input
```

- To delete an LDAP pool:

```
davadmin ldappool delete -n myldap
```

- To list all existing LDAP pools:

```
davadmin ldappool list
```

- To list the configuration parameters of a specific LDAP pool:

```
davadmin ldappool list -n myldap
```

davadmin migration

Use this command to performs migration of Calendar Server 6 data to Calendar Server 7.

For more information on migrating from Sun Java System Calendar Server 6 to Oracle Calendar Server 7, see *Calendar Server Installation and Configuration Guide*.

Syntax

```
davadmin migration [ migrate | status ]
                    [-u id] [-W] [-H hostname]
                    [-p port] [-s path] [-a account]
                    [-X migrationadminuser] [-F clifile] [-f file]
                    [-L migrationserverport] [-S] [-B ldapbaseuri] [-R ldapfilter]
                    [-T starttime] [-l logpath] [-c] [-G tag] [-h]
```

migration Operation

[Table 16–24](#) describes the actions for the **migration** operation.

Table 16–24 Actions for migration Operation

Action	Description
migrate	Migrates the specified user(s).
status	Gets the current status of the migration operation.

The **migration** option supports all **davadmin** common options. The default action for **migration** is **migrate**.

Options for migration Operation

[Table 16–25](#) describes the options for the **migration** operation.

Table 16–25 Options for migration Operation

Short Option	Long Option	Description	Required
-a	--account	Principal account information of the user to be migrated, provided as email address.	Required unless batch mode is used and account information provided in files, or ldapfilter used.
-X	--migrationadminuser	Administrative user to authenticate to Calendar Server 6 host.	Required unless information is provided in clifile .
-L	--migrationserverport	Server and port information to connect to the Calendar Server 6 host from which data needs to be migrated. The format is <i>server:port</i> .	Required unless information is provided in clifile .
-l <i>log-directory</i>	--logpath	Logs information about the migration status.	Optional. Defaults to the Calendar Server log directory.
-f	--file	Local input file for batch operation. Each line contains the email address for an account.	Optional if using the -a option for single user migration, or an LDAP base URL is provided by using the -B option.
-S	--clientssl	Use SSL when making client connections.	Optional.
-B	--baseuri	Base URL in LDAP. All users under the URL are migrated.	Required if -a or -f options are not specified.
-R	--ldapfilter	User search filter in LDAP. Default is objectclass=icsCalendarUser .	Optional.
-T	--starttime	Start date for events and tasks to be migrated. The format of this value is <i>yyyymmddThhmmssZ</i> . This value is in Zulu time. (The T is a separator between the day and time.)	Optional.
-c	--capture	Captures trace information and details regarding the migration.	Optional. Useful if migration fails but produces a large amount of output.
-G	--tag	Log tag to use to check status. This is the path to the master log file that is output when the migration command is executed.	Required for status check.
-h	--help	Usage of davadmin migration .	Optional.

For more information, see the topic on migration logging in *Calendar Server Installation and Configuration Guide*.

The **clifile** that is provided through the **-F** option can be used to provide entries for **migrationadminuser**, **migrationadminpassword**, and **migrationserverport**. The long option for **-x** is **--migrationadminpasswordpath**, a path to the password file, but the entry in the **clifile** is **migrationadminpassword**, because it is just a password.

davadmin migration Examples

- To perform a migration of **user1**'s calendar (prior to Calendar Server 7 Update 2):

```
davadmin migration -X calmaster -x /admin/calmaster_pwd -L
cs6host.example.com:8080 -a user1@example.com -u admin -W /admin/appserver_pwd
-s /admin/truststore -t /admin/truststore_pwd
```

- To perform a migration of a list of users using the **clifile** for most of the input values (prior to Calendar Server 7 Update 2):

```
davadmin migration -f /admin/user_list -F /admin/mig_clifile
```

Where **user_list** contains:

```
user1@example.com
user2@example.com
user300@example.com
```

and the **mig_clifile** contains:

```
userid=admin
hostname=localhost
port=8686
secure=/admin/truststore
migrationadminuser=calmaster
migrationserverport=cs6host.example.com:8080
```

- To find the users to migrate based on an LDAP base URI and an LDAP filter (uid):

```
davadmin migration migrate -B "o=isp" -R "uid=c*" -X calmaster -L
cs6host.example.com:8080 -u admin
```

- To find the users to migrate based on an LDAP base URI and an LDAP filter (object class):

```
davadmin migration migrate -B "ou=people,o=example.com,o=isp" -R
"objectclass=icscalendaruser" -X calmaster -L cs6host.example.com:8080 -u admin
```

davadmin passfile

Use this command to create, delete, list, or modify passwords in the password file.

When running the **davadmin** command, instead of having to enter passwords at the no-echo prompt, you can supply passwords by using the **password** file. The **password** file is an encrypted "wallet," which holds all passwords that **davadmin** might use.

Syntax

```
davadmin migration [ create | delete | list | modify ]
[-u id] [-W] [-F clifile] [-H hostname]
[-p port] [-s path] [-h] [-O]
```

passfile Operation

Table 16–26 describes the actions for the **davadmin passfile** operation.

Table 16–26 Actions for passfile Operation

Action	Description
create	Creates the password file. If it already exists, modifies it.
delete	Deletes passwords in the password file. For each password, you are asked if it should be removed.
list	Displays all passwords in the password file.
modify	Modifies passwords in the password file.

The default action is **list**.

Options for passfile Operation

Table 16–27 describes the option for the **passfile** operation.

Table 16–27 Option for passfile Operation

Option	Description
-O	Run the passfile command in standalone mode when access to the Calendar Server is not needed. This is used when setting, deleting, and listing the document store password and SSL passwords on the remote document store host.

The **passfile** operation is available for the **create**, **delete**, **list**, and **modify** actions.

davadmin passfile Examples

- To modify the migration administrative password and add the document store password:

```
./davadmin passfile modify
Enter the Password File password:
Do you want to set the app server admin user password (y/n)? [n] n
Do you want to set the database password (y/n)? [n]
Do you want to set the migration server user password (y/n)? [n] y
Enter the migration server user password:
Reenter the migration server user password:
Do you want to set the document store password (y/n)? [n] y
Enter the document store password:
Reenter the document store password:
Do you want to set the document store SSL passwords (y/n)? [n]
Set new value for store.document.password. A server restart is required for
this change to take effect.
```

- To remove the database administrative password:

```
./davadmin passfile delete
Enter the Password File password:
Do you want to remove the app server admin user password (y/n)? [n]
Do you want to remove the database password (y/n)? [n] y
Do you want to remove the migration server user password (y/n)? [n]
Do you want to remove the document store password (y/n)? [n]
Do you want to remove the document store SSL keystore password (y/n)? [n]
Do you want to remove the document store SSL certificate password (y/n)? [n]
```

- To change the password for the remote document store on the remote host. This command must be run on the remote host:

```
./davadmin passfile modify -O
Enter the Password File password:
Do you want to set the document store password (y/n)? [n] y
Enter the document store password:
Reenter the document store password:
Do you want to set the document store SSL passwords (y/n)? [n]
```

- To list all of the passwords:

```
./davadmin passfile list
Enter the Password File password:
The app server admin user password: theadminpass
The migration server user password:
The database password: thesqlpass
The document store password: thedocstorepass
The document store SSL keystore password:
The document store SSL certificate password:
```

- To set the document store passwords used for SSL communications:

```
./davadmin passfile modify -O
Enter the Password File password:
Do you want to set the document store password (y/n)? [n]
Do you want to set the document store SSL passwords (y/n)? [n] y
Enter the document store SSL keystore password:
Reenter the document store SSL keystore password:
Enter the document store SSL certificate password:
Reenter the document store SSL certificate password:
```

davadmin vscan

Use this command to performs virus scanning operations.

Syntax

```
davadmin vscan [ scan ]
                [-u id] [-W] [-H hostname]
                [-p port] [-s path]
                [-F clifile]
                [-n backendID] [-a account] [-B uri] [-R filter]
                [-T time] [-r] [-h]
```

vscan Operation

[Table 16–28](#) describes the action for the **vscan** operation.

Table 16–28 Action for vscan Operation

Action	Description
scan	Scans calendar data for viruses.

The **scan** action is the default.

Options for vscan Operation

Table 16–29 describes the options for vscan operations.

Table 16–29 Options for vscan Operation

Short Option	Long Option	Description
-u <i>id</i>	--userid	The GlassFish Server administrator's user name. Required unless you provide it through a CLI file by using the -F option, or you are displaying usage by using the -h option.
-F <i>file</i>	--clifile	File with bootstrap information that you use to specify command-line options so that they do not have to be entered at the command line. Each line in the bootstrap file is in the form property=value . For possible properties, see Table 16–2. Required unless all necessary information is provided on the command line or in the davadmin.properties file. See Options Precedence for more information on priority order of options, the clifile and the davadmin.properties file. A path to the clifile file can also be specified by the DAVADMIN_CLIFILE environment variable.
-H <i>host</i>	--hostname	Host name of the server. Optional, defaults to localhost.
-p <i>port</i>	--port	GlassFish administration port (JMX connector port). The GlassFish administration port can be found in the domain's domain.xml file or in the Administration Console (Configuration->Admin Service->system. Optional. Defaults to 8686.
-s <i>path</i>	--secure	Path to the truststore file used for a secure connection (HTTPS). Optional. Required if GlassFish is running in secure mode.
-a <i>account</i>	--account	The account information (email address) of the user to be scanned.
-n <i>backendid</i>	--name	The name of the target backendID.
-B <i>uri</i>	--ldapbaseuri	Base URI in LDAP.
-R <i>filter</i>	--ldapfilter	User search filter in LDAP. Default is (objectClass=icsCalendarUser) .
-T <i>time</i>	--starttime	Scan data entered into the server after this time. Format: <i>yyyymmddThhmmssZ</i>
-r	--force	Force delete any data found as a positive hit during the virus scan. This overrides the davcore.virusscan.clivirusaction variable. So with davcore.virusscan.clivirusaction set to empty string (no delete) viruses are listed in the scan log after a scan. Then you can add a -r to the scan to delete offending data after review, without needing to change the virus scan configuration parameters.
-h	--help	Help for that particular operation. Optional.

For more information about how to set up and manage virus scanning, see ["Configuring and Managing Virus Scanning."](#)

davadmin vscan Examples

The **davadmin vscan** command operates through the GlassFish Server, and can thus operate on any of the back ends configured with the specific Calendar Server. (There may very well only be one.)

- To list the back ends:

```
davadmin backend list -u admin
```

```
defaultbackend  
ischedulebackend
```

Normally you would want to scan the "defaultbackend" since that is where calendar user's events and attachments are stored.

- To scan the entire default back end:

```
davadmin vscan scan -u admin -n "defaultbackend"
```

- To scan a single user's data given their calendar server registered email address:

```
davadmin vscan scan -u admin -a joe.smith@example.com
```

- To use LDAP base and filter to specify one or more users to scan:

```
davadmin vscan scan -u admin -n defaultbackend -B "o=dav" -R "uid=caluser12"  
davadmin vscan scan -u admin -n defaultbackend -B "o=dav" -R  
"(|(uid=caluser222)(uid=caluser111))"
```

In this example, using just a uid filter might not be specific enough for multiple domains. Perhaps use **ldapsearch** to test filters if needed.

- To scan data at or beyond February 14th, 2011, 1am Zulu:

```
davadmin vscan scan -u admin -n defaultbackend -T 20110214T010000Z
```

Specifying a **-T** only scans data at the specified time and later, and is a big time saver for ignoring older data already scanned. In the scan log, the time just before the scan began is printed at the end of the run so it can be used with the **-T** option in the next scan if no new virus rules are relevant.

Note: The **davadmin vscan** command uses the same virus scan configuration as online virus scan, however it does not use the **onlineenable** variable. Thus, you can run command-line scans without needing to affect incoming data if desired.

JConsole

The data and operations exposed by the MXBeans in the CalDAV server are accessible and modifiable by using JConsole. All Admin Beans can be found under **com.sun.comms.davserver.adminutil**.

AdminAccountMXBean Operation

Provides **createAccounts**, **deleteAccounts**, **listAccounts**, **modifyAccounts**, **deleteCalComponents** and **fixAccountMail** operations.

AdminBackendMXBean Operation

Provides **createBackend** and **getBackends** operations.

AdminCalComponentMXBean Operation

Provides **getCalComponentInfo**, **getCalComponents**, **deleteCalComponents**, **importCalComponents** and **exportCalComponents** operations.

AdminCalendarMXBean Operation

Provides `createCalCollection`, `modifyCalCollection`, `deleteCalCollection` and `getCalCollections` operations.

AdminConfigMBean Operation

You use the `getConfigParam` and `setConfigParam` operations to get and set configuration options. The `AllConfigParams` operation provides a list of the configuration parameters. The value in JConsole is displayed as a "`java.lang.String[]`" array and double-clicking this field shows the individual parameters. The `getConfigParamDescription` operation is used to get a detailed description of a parameter.

AdminMigrationMXBean Operation

Provides `checkStatus` and `migrate` operations.

AdminMiscMXBean Operation

The version attribute of this MBean provides the server version.

AdminUtilMXBean

This is the super class for all Admin...MXBeans. It provides the `checkConnection` operation.

Starting GlassFish Server in Secure Mode

If GlassFish Server is running in a secure mode, JConsole needs to be started with the `truststorepath` (`-Djavax.net.ssl.trustStore`) and optionally `truststorepassword` (`-Djavax.net.ssl.trustStorePassword`) passed in.

Summary of davadmin Changes by Release

Topics in this section:

- [Changes in Calendar Server 7 Update 1](#)
- [Changes in Calendar Server 7 Update 2](#)
- [Changes in Calendar Server 7 Update 2 Patch 5](#)
- [Changes in Calendar Server 7 Update 3](#)
- [Changes in Calendar Server 7.0.4.14.0](#)

Changes in Calendar Server 7 Update 1

- The `calresource` operation has been renamed to `calcomponent`.
- The `migration` operation has been added for migration of data from Calendar Server 6.3 to Calendar Server 7.
- The `account` operation has been added to enable listing, deletion, and properties modification of user accounts.
- The `calendar` operation has been enhanced to enable setting of more calendar properties.

Changes in Calendar Server 7 Update 2

- The **davadmin** command has been made more secure in Calendar Server 7 Update 2 by the removal of the capability to "pass in" passwords by using a password file. All **davadmin** passwords must now be entered by typing in to a no-echo prompt.
- The **backend** and **vscan** arguments have been added.
- The **dbhost** property replaces the **dbhostname** property.
- The **create**, **delcomponents**, and **repair** actions have been added to the **account** operation.
- The **config -l** option has been removed. Use **config list** now.
- The **-t** option has been added to the **davadmin db** command.
- The **list** and **modify** actions have been added to the **davadmin config** command.

Changes in Calendar Server 7 Update 2 Patch 5

- The **davadmin** command has also been updated to list calendars belonging to resource accounts owned by a user.
- To clear a resource's owner field, run the **davadmin account modify -a resource -y owner=""** command.
- The **repair** operation has been enhanced to include the **-m** option, to repair the user's email address after an email change, and the **-o** option, to update the owner lists of all accounts.
- The **list** operation displays managed calendars for an account.
- The **davadmin calcomponent import** command enables the import to continue even if an error occurs on an item being imported.
- You can create a **password** file for use with the **davadmin** command to store administrator passwords for the GlassFish Server administrative user, the migration administrative user, and the database user.

Changes in Calendar Server 7 Update 3

- The **passfile** option has been updated to accommodate setting a password on the local and remote document store.
- A new command, **davadmin ldappool**, has been added to support LDAP pools (which are used in configuring external Directory Server authentication).
- The **davadmin account list** command now displays a list of all users in the database and their details.

Changes in Calendar Server 7.0.4.14.0

- The **-v** option to **davadmin account list** displays the details of each account at the same time.
- The **davadmin account** command takes **subscribe** and **unsubscribe** actions, so that a Calendar Server administrator can subscribe or unsubscribe calendars for a user. The **subscribe** and **unsubscribe** actions take either a single collection path on the command line, specified by **-c**, or a set of collection paths in a file, specified by **-C**.
- The **davadmin config list -M** command lists changed options only.

- The **davadmin config -d** option sets the value to the default when used with the **modify** action. Additionally it lists the default value when used with the **list** action.
- The **davadmin account -y** operation and **davadmin -y calendar** operation take the **set-ace** and **remove-ace** properties.
- The **davadmin db -s** operation supplies the path to the **trustStore** file that contains the SSL certificate for secure communications with the remote document store.
- You can now set account properties with the new **account** operation option by using an input file (-f option). Previously, the -f option used to only allow a user name per line. Now it allows a user name followed by properties for that user.
- The **davadmin account upgrade** operation sets the next presence triggers for all existing events in the future. You must run **davadmin account upgrade** after upgrading from Calendar Server 7 Update 3 or prior releases for existing future events to have their presence triggers set.
- The **davadmin db backup**, **list**, and **restore** commands now require that you specify the associated document store. You specify the document store by using the **-A** option, or the **docstore** option in the CLI file.

Changes in Calendar Server 7.0.4.16.0

- The **-v** option to the **davadmin calendar list -a user** command displays a summary for all calendars belonging to the user.

Changes in Calendar Server 7.0.5.17.0

- The **davadmin db** command now takes the **schema_version**, **schema_fullupgrade**, and **schema_preupgrade** operations.

Deprecated Options

Table 16–30 describes the deprecated **davadmin** common options and in what release the option was deprecated.

Table 16–30 *Deprecated Common Option*

Short Option	Long Option	Description	Required or Optional
-W <i>passfile</i> Removed in Calendar Server 7 Update 2 . You are now prompted to enter the administrative password.	--passwordfile	File containing MySQL password for db commands, GlassFish Administrator password for all other commands.	Required unless you provide the password by using the -F option or by displaying usage by using the -h option. If you don't provide this information, you are prompted for the password.

Table 16–31 describes the deprecated **clifile** properties and in what release the property was deprecated.

Table 16–31 *Deprecated Clifile Properties*

Property	Description
userid	GlassFish Administrator user ID.

Table 16–31 (Cont.) Deprecated Clifile Properties

Property	Description
password Removed in Calendar Server 7 Update 2 . You are now prompted to enter the administrative password.	GlassFish Administrator password.
dbpassword Removed in Calendar Server 7 Update 2 . You are now prompted to enter the administrative password.	MySQL database user password.
migrationadminpassword Removed in Calendar Server 7 Update 2 . You are now prompted to enter the administrative password.	The Calendar Server 6 administrative password.

[Table 16–32](#) describes the deprecated option for **config** operation option and in what release the option was deprecated.

Table 16–32 Deprecated Option for config Operation

Short Option	Long Option	Description
-l Removed in Calendar Server 7 Update 2 . See the list action.	--list	Lists all configuration options.

[Table 16–33](#) describes the deprecated option for **migration** operation option and in what release the option was deprecated.

Table 16–33 Deprecated Option for migration Operation

Short Option	Long Option	Description	Required
-x Removed in Calendar Server 7 Update 2 . You are now prompted to enter the administrative password.	--migrationadminpasswordpath	Path to file that contains the Calendar Server 6 administrative password.	Required unless information is provided in clifile .

Calendar Server Configuration Parameters

Table 17-1 describes the configuration parameters and descriptions for Calendar Server. See "[davadmin config](#)" for information on how to update or change configuration parameters.

Table 17-1 Calendar Server Configuration Parameters

Parameter	Description
<code>base.ldapinfo.cachesize</code>	Size of the LDAP Authentication cache. Syntax: integer Minimum: 1 Maximum: 1000000 Default: 1000
<code>base.ldapinfo.cachettl</code>	Time to live (in seconds) of cached LDAP Authentication info. Syntax: integer Unit: seconds Minimum: 1 Maximum: Maximum int value Default: 60
<code>base.ldapinfo.dcroot</code>	Root of DC tree (Schema 1) or of the domain and users tree (Schema 2) in Directory Server Syntax: string Default: o=isp
<code>base.ldapinfo.defaultdomain</code>	Default domain Syntax: string Default: demo.example.com
<code>base.ldapinfo.domainattrs</code>	Space separated list of LDAP attributes to use when retrieving domain information. Syntax: string Default: icsStatus icsDomainNames icsDomainAcl externalAuthPreUrlTemplate externalAuthPostUrlTemplate corpDirectoryUrl
<code>base.ldapinfo.loginseparator</code>	Character(s) to be used as login separator (between user ID and domain) Syntax: string Default: @

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
base.ldapinfo.schemalevel	Schema level. Syntax: integer Minimum: 1 Maximum: 2 Default: 2
base.ldapinfo.searchfilter	This is the search filter used to look up users during authentication when one is not specified in the inetDomainSearchFilter for the domain. The syntax is the same as inetDomainSearchFilter . See <i>Communications Suite Schema Reference</i> for more information. Syntax: string Default: (uid=%U)
base.ldapinfo.serviceadmin	Distinguished name of single administrator in LDAP in absence of admin group. Syntax: string Default: None.
base.ldapinfo.serviceadmins	Distinguished Name of service admins group in LDAP. Syntax: string Default: None.
base.ldapinfo.userattrs	Space-separated list of LDAP attributes to retrieve from user entries during the authentication phase. Syntax: string Default: mail ismemberof
base.ldapinfo.authldap.binddn	Distinguished name to use when authenticating. Syntax: string Default: None.
base.ldapinfo.authldap.bindpassword	Password to use when authenticating. Syntax: password Default: None
base.ldapinfo.authldap.ldaphost	Space-delimited list of host names. Each host name may include a trailing colon and port number. Syntax: string Default: localhost:389
base.ldapinfo.authldap.ldappoolrefreshinterval	Length of elapsed time until the failover Directory Server host reverts back to the primary Directory Server host. If set to -1 , does not refresh. Syntax: integer Unit: minutes Minimum: 1 Maximum: 60 Default: 1

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
base.ldapinfo.authldap.ldappoolsize	Maximum number of connections for this pool. Syntax: integer Minimum: 1 Maximum: 100 Default: 10
base.ldapinfo.authldap.ldapport	Port number to which to connect. Ignored for any host name which includes a colon and port number. Syntax: integer Minimum: 0 Maximum: 65535 Default: 389
base.ldapinfo.authldap.ldaptimeout	Timeout for all LDAP operations. Syntax: integer Unit: seconds Minimum: 1 Maximum: 3600 Default: 60
base.ldapinfo.authldap.ldapusessl	Use SSL to connect to the LDAP host. Syntax: boolean Default: false
base.ldapinfo.authldap.sslprotocols	Specifies a space-delimited list of the supported SSL protocols to communicate with the LDAP back-end service. Syntax: string Default: TLSv1 TLSv1.1 TLSv1.2
base.ldapinfo.ugldap.binddn	Distinguished name to use when authenticating. Syntax: string Default: None.
base.ldapinfo.ugldap.bindpassword	Password to use when authenticating. Syntax: password Default: None.
base.ldapinfo.ugldap.ldaphost	Space-delimited list of host names. Each host name may include a trailing colon and port number. Syntax: string Default: localhost:389
base.ldapinfo.ugldap.ldappoolrefreshinterval	Length of elapsed time until the failover Directory Server host reverts back to the primary Directory Server host. If set to -1 , do not refresh. Syntax: integer Unit: minutes Minimum: 1 Maximum: 60 Default: 1

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
base.ldapinfo.ugldap.ldappoolsize	Maximum number of connections for this pool. Syntax: integer Minimum: 1 Maximum: 100 Default: 10
base.ldapinfo.ugldap.ldapport	Port number to which to connect. Ignored for any host name that includes a colon and port number. Syntax: integer Minimum: 0 Maximum: 65535 Default: 389
base.ldapinfo.ugldap.ldaptimeout	Timeout for all LDAP operations. Syntax: integer Unit: seconds Minimum: 1 Maximum: 3600 Default: 60
base.ldapinfo.ugldap.ldapussl	Use SSL to connect to the LDAP host. Syntax: boolean Default: false
base.ldapinfo.ugldap.sslprotocols	Specifies a space-delimited list of the supported SSL protocols to communicate with the back-end LDAP service. Syntax: string Default: TLSv1 TLSv1.1 TLSv1.2
base.ldappool.*.binddn	Distinguished name to use when authenticating. Syntax: string Default: None.
base.ldappool.*.bindpassword	Password to use when authenticating. Syntax: password Default: None.
base.ldappool.*.ldaphost	Space-delimited list of host names. Each host name may include a trailing colon and port number. Syntax: string Default: localhost:389
base.ldappool.*.ldappoolrefreshinterval	Length of elapsed time until the failover Directory Server host reverts back to the primary Directory Server host. If set to -1 , do not refresh. Syntax: integer Unit: minutes Minimum: 1 Maximum: 60 Default: 1

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
base.ldappool.*.ldappoolsize	Maximum number of connections for this pool. Syntax: integer Minimum: 1 Maximum: 100 Default: 10
base.ldappool.*.ldapport	Port number to which to connect. Ignored for any host name that includes a colon and port number. Syntax: integer Minimum: 0 Maximum: 65535 Default: 389
base.ldappool.*.ldaptimeout	Timeout for all LDAP operations. Syntax: integer Unit: seconds Minimum: 1 Maximum: 3600 Default: 60
base.ldappool.*.ldapusessl	Use SSL to connect to the LDAP host. Syntax: boolean Default: false
base.ldappool.*.sslprotocols	Specifies a space-delimited list of the supported SSL protocols for the LDAP pool to communicate with the back-end LDAP service. Syntax: string Default: TLSv1 TLSv1.1 TLSv1.2
davcore.acl.aclcachesize	Maximum number of ACL entries kept in cache. Entries are removed from the cache only when this maximum is reached or when any of the acl configuration parameters are changed. Can be set to 0 , indicating no cache. Syntax: integer Minimum: 0 Maximum: Maximum int value Default: 1000
davcore.acl.aclcachettl	Maximum amount of time (in seconds) that an ACL entry can be kept in cache. Syntax: integer Unit: seconds Minimum: 1 Maximum: Maximum int value Default: 60
davcore.acl.appleprivateevent	Enables or disables Apple Private/Confidential Events extension support. Syntax: boolean Default: false

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.acl.calendaranonymousall	If true, map '@'(all) in calendar acls to authenticated and unauthenticated users. If false, map '@'(all) in acl to just authenticated users. Syntax: boolean Default: true
davcore.acl.defaultcalendaracl	ACL to set when creating a new calendar collection (using the WCAP format). Syntax: string Default: None.
davcore.acl.defaultresourcecalendaracl	ACL to set when creating a new calendar collection for a resource (using the WCAP format). Syntax: string Default: @:r
davcore.acl.defaultresourceschedulingacl	ACL for user permissions to be set when creating a new calendar inbox collection for a resource (using the WCAP format). Syntax: string Default: @:s
davcore.acl.defaultschedulingacl	ACL for user permissions to be set when creating a new calendar inbox collection (using the WCAP format). Syntax: string Default: @:s
davcore.acl.schedulinganonymousall	If true , map '@'(all) in scheduling ACLs to authenticated and unauthenticated users. If false , map '@'(all) in acl to just authenticated users. Syntax: boolean Default: true
davcore.attachment.enable	Enables or disables attachments. Syntax: boolean Default: true
davcore.auth.cert.enable	Enables certificate-based client authentication. Syntax: boolean Default: false
davcore.auth.cert.fallback	Fallback to user name and password authentication. Syntax: boolean Default: true

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
<p>davcore.autocreate.calattendantresourceflags</p>	<p>Calendar attendant flags to set on resource scheduling inbox. Default setting used on autocreation.</p> <p>This value can be altered by the presence of the icsAutoAccept and icsDoubleBooking attributes. The value is a bitmask of the following:</p> <ul style="list-style-type: none"> 0 - No automatic accept, no booking conflict check, no recurrence check on invitations. 1 - Automatically accept invitations. 2 - Automatically declines if invitation results in booking conflict. 3 - Automatically accepts invitation and automatically declines on booking conflict. 4 - Automatically declines recurring meeting invitations. 5 - Automatically accepts invitations and automatically decline recurring meeting invitations. 6 - Automatically declines recurring invitations and invitations that cause a booking conflict. 7 - Automatically accepts invitations, automatically declines recurring invitations and invitations that cause a booking conflict. <p>Syntax: long Minimum: 0 Maximum: 7 Default: 3</p>
<p>davcore.autocreate.calattendantuserflags</p>	<p>Calendar attendant flags to set on users scheduling inbox. Default setting used on autocreation.</p> <p>This value can be altered by the presence of the icsAutoAccept and icsDoubleBooking attributes.</p> <p>The value is a bitmask of the following:</p> <ul style="list-style-type: none"> 0 - No automatic accept, no booking conflict check, no recurrence check on invitations. 1 - Automatically accepts invitations. 2 - Automatically declines if invitation results in booking conflict. 3 - Automatically accepts invitation and automatically declines on booking conflict. 4 - Automatically declines recurring meeting invitations. 5 - Automatically accepts invitations and automatically declines recurring meeting invitations. 6 - Automatically declines recurring invitations and invitations that cause a booking conflict. 7 - Automatically accepts invitations, automatically declines recurring invitations and invitations that cause a booking conflict. <p>Syntax: long Minimum: 0 Maximum: 7 Default: 0</p>

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.autocreate.calcomponents	Supported calendar components set for a new user calendar on autocreation. Default setting used on autocreation. Syntax: string Default: VEVENT VTOD VFREEBUSY
davcore.autocreate.displaynameattr	LDAP attribute, whose value is used to set Display Name, during autocreation. Default setting used on autocreation. Syntax: string Default: cn
davcore.autocreate.emailnotificationaddressattr	LDAP attribute, whose value is used to set Email Notification address, during autocreation. Default setting used on autocreation. Syntax: string Default: mail
davcore.autocreate.enableautocreate	Enable autocreate operation. Default setting used on autocreation. Syntax: boolean Default: true
davcore.autocreate.enableemailnotification	Enable email notification. Default setting used on autocreation. Syntax: boolean Default: true
davcore.autocreate.rescalcomponents	Supported calendar components set for a new resource calendar on autocreation. Default setting used on autocreation. Syntax: string Default: VEVENT
davcore.homeuri.*.backendid	Once it is determined that a URI matches the pattern, this backendid template is used to identify the back end server hosting this resource. The template can reference the \$1, \$2,... variables saved during the pattern matching, as well as references to LDAP attributes of the subject matching the subjectfilter , using the \${attrname} syntax or the \${attrname,defaultvalue} syntax. If this parameter is not set, the uriinfo.backendidtemplate is used. Syntax: string Default: None.
davcore.homeuri.*.rank	When multiple URI patterns are configured, this value determines in which order those URI patterns are evaluated. A lower number indicates that this pattern should be evaluated first. Syntax: integer Minimum: 0 Maximum: Maximum int value Default: 1

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.homeuri.*.subjectdomain	<p>Once it is determined that a URI matches the pattern, this domain template is used to identify the subject owning with this resource.</p> <p>The template can reference the \$1, \$2, and so variables saved during the pattern matching. For example, if the subjectdomain is set to \$2, and using the URI in the uripattern example, the domain of the subject will be example.com.</p> <p>Can be empty indicating the default domain.</p> <p>Syntax: string</p> <p>Default: \$2</p>
davcore.homeuri.*.subjectfilter	<p>Once it is determined that a URI matches the pattern, this LDAP filter template is used to identify the subject owning with this resource.</p> <p>The template can reference the \$1, \$2, and so on variables saved during the pattern matching. For example, if the subjectfilter is set to (mail=\$1@\$2), and using the URI in the uripattern example, the LDAP filter becomes (mail=john@example.com).</p> <p>Can be empty, indicating that this namespace is not associated with a particular subject.</p> <p>Syntax: string</p> <p>Default: (mail=\$1@\$2)</p>
davcore.homeuri.*.uripattern	<p>Regex pattern to be matched by the URI.</p> <p>This pattern can contain regex groups (identified by () parenthesis) which will be saved into \$1, \$2, and so on.</p> <p>The last regex group identifies the local path if there is any.</p> <p>For example, if the pattern is ^/home/([^/+]@([^/+])(/[^/]*)), the URI /home/john@example.com/calendar/ matches that pattern. \$1 will be set to the value john, \$2 will be set to the value example.com, and the local path will be /calendar.</p> <p>Syntax: string</p> <p>Default: ^/home/([^/+]@([^/+])(/[^/]*))</p>
davcore.ldapattr.commonname	<p>Common name attribute.</p> <p>Syntax: string</p> <p>Default: cn</p>
davcore.ldapattr.corpdirectoryurl	<p>LDAP attribute to locate a custom external corporate directory for this domain.</p> <p>Syntax: string</p> <p>Default: corpDirectoryUrl</p>
davcore.ldapattr.davstore	<p>Logical back-end ID attribute.</p> <p>Syntax: string</p> <p>Default: davStore</p>
davcore.ldapattr.defaultresourcetype	<p>Default CUTYPE value to use when the resource type LDAP attribute of a calendar resource is not present.</p> <p>Syntax: string</p> <p>Default: ROOM</p>

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.ldapattr.dngroupleader	Attributes for members in an LDAP group. Syntax: string Default: uniquemember
davcore.ldapattr.externalauthposturltemplate	LDAP attribute to use to determine whether external authentication should do a post=auth lookup against this domain. Syntax: string Default: externalAuthPostUrlTemplate
davcore.ldapattr.externalauthpreurltemplate	LDAP attribute to use to determine whether external authentication should be used against this domain. Syntax: string Default: externalAuthPreUrlTemplate
davcore.ldapattr.groupobject	Space separated list of object class values indicating an LDAP group. Syntax: string Default: groupofuniquenames groupofurls inetmailgroup
davcore.ldapattr.icsautoaccept	LDAP attribute to use to determine whether autoaccept should be enabled. The attribute value can be 1 (autoaccept) or 0 (no autoaccept). It is used only during autcreate. Syntax: string Default: icsAutoAccept
davcore.ldapattr.icsdoublebooking	LDAP attribute to use to determine whether double booking is allowed. The attribute value can be 1 (double booking allowed) or 0 (no double booking allowed). It is used only during autcreate. Syntax: string Default: icsDoubleBooking
davcore.ldapattr.icsstatus	Calendar Service status attribute. Syntax: string Default: icsstatus
davcore.ldapattr.mail	Mail attribute. Syntax: string Default: mail
davcore.ldapattr.mailalternateaddress	Space-separated list of alternate mail attributes. Syntax: string Default: mailAlternateAddress
davcore.ldapattr.mailgroupmember	Attributes for members in an LDAP group. Syntax: string Default: mgrpfc822mailmember

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.ldapattr.memberattr	LDAP attribute listing the groups of which the entry is a member. Syntax: string Default: ismemberof
davcore.ldapattr.preferredlang	Language attribute. Syntax: string Default: preferredLanguage
davcore.ldapattr.resourceobject	Space-separated list of object class values indicating an LDAP resource. Syntax: string Default: icsCalendarResource
davcore.ldapattr.resourceowner	LDAP attribute to use to determine the owner of a calendar resource (for example, conference room). The attribute value must be a distinguished name. It is used only during autcreate. Syntax: string Default: owner
davcore.ldapattr.resourcetype	LDAP attribute to use to determine the CUTYPE (ROOM versus RESOURCE) of a calendar resource. The CUTYPE of users and groups is not based on this attribute. The attribute value can take the following values: <ul style="list-style-type: none"> ▪ Location and room are mapped to a CUTYPE of ROOM. ▪ Thing and resource are mapped to a CUTYPE of RESOURCE. ▪ Other values are mapped to RESOURCE. Syntax: string Default: kind
davcore.ldapattr.uid	User ID attribute. Syntax: string Default: uid
davcore.ldapattr.urlgroupmember	Attributes for members in an LDAP group. Syntax: string Default: memberurl
davcore.ldapattr.userobject	Space separated list of object class values indicating a calendar user. Syntax: strin Default: None.

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.otheruri.*.backendid	<p>Once it is determined that a URI matches the pattern, this backendid template is used to identify the back end server hosting this resource.</p> <p>The template can reference the \$1, \$2, and so on variables saved during the pattern matching, as well as refer to LDAP attributes of the subject matching the subjectfilter, using the \${attrname} syntax or the \${attrname,defaultvalue} syntax. If this parameter is not set, the uriinfo.backendidtemplate is used.</p> <p>Syntax: string Default: None.</p>
davcore.otheruri.*.rank	<p>When multiple URI patterns are configured, this value determines in which order those URI patterns should be evaluated. A lower number indicates that this pattern should be evaluated first.</p> <p>Syntax: integer Minimum: 0 Maximum: Maximum int value Default: 1</p>
davcore.otheruri.*.subjectdomain	<p>Once it is determined that a URI matches the pattern, this domain template is used to identify the subject owning with this resource.</p> <p>The template can reference the \$1, \$2, and so on variables saved during the pattern matching. For example, if the subjectdomain is set to \$2, and using the URI in the uripattern example, the domain of the subject will be example.com.</p> <p>Can be empty indicating the default domain.</p> <p>Syntax: string Default: \$2</p>
davcore.otheruri.*.subjectfilter	<p>Once it is determined that a URI matches the pattern, this LDAP filter template is used to identify the subject owning with this resource.</p> <p>The template can reference the \$1, \$2, and so on variables saved during the pattern matching. For example, if the subjectfilter is set to (mail=\$1@\$2), and using the URI in the uripattern example, the LDAP filter becomes (mail=john@example.com).</p> <p>Can be empty, indicating that this namespace is not associated with a particular subject.</p> <p>Syntax: string Default: (mail=\$1@\$2)</p>

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
<p>davcore.otheruri.*.uripattern</p>	<p>Regex pattern to be matched by the URI.</p> <p>This pattern can contain regex groups (identified by () parenthesis) which is saved into \$1, \$2, and so on.</p> <p>The last regex group identifies the local path if there is any.</p> <p>For example, if the pattern is <code>^/home/([^/+]@([^/+])(/\z /.*)</code>, the URI <code>/home/john@example.com/calendar/</code> will match that pattern. \$1 will be set to the value <code>john</code>, \$2 will be set to the value <code>example.com</code> and the local path will be <code>/calendar</code>.</p> <p>Syntax: string</p> <p>Default: <code>^/home/([^/+]@([^/+])(/\z /.*)</code></p>
<p>davcore.presence.advancepresencetriggerinterval</p>	<p>Specifies the number of seconds in advance to trigger for presence update. That is, how long before event start/end is the made. Changes to this value do not affect existing event triggers.</p> <p>Syntax: long</p> <p>Unit: seconds</p> <p>Minimum: 0</p> <p>Maximum: Maximum long value</p> <p>Default: 0</p>
<p>davcore.presence.enable</p>	<p>Enables or disables presence information publication.</p> <p>Syntax: boolean</p> <p>Default: true</p>
<p>davcore.principalsuri.*.backendid</p>	<p>Once it is determined that a URI matches the pattern, this backendid template is used to identify the back-end server hosting this resource.</p> <p>The template can reference the \$1, \$2, and so on variables saved during the pattern matching, as well as refers to LDAP attributes of the subject matching the subjectfilter, using the <code>#{attrname}</code> syntax or the <code>#{attrname,defaultvalue}</code> syntax. If this parameter is not set, the <code>uriinfo.backendidtemplate</code> is used.</p> <p>Syntax: string</p> <p>Default: None.</p>
<p>davcore.principalsuri.*.rank</p>	<p>When multiple URI patterns are configured, this value determines in which order those URI patterns should be evaluated. A lower number indicates that this pattern should be evaluated first.</p> <p>Syntax: integer</p> <p>Minimum: 0</p> <p>Maximum: Maximum int value</p> <p>Default: 1</p>

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.principalsuri.*.subjectdomain	<p>Once it is determined that a URI matches the pattern, this domain template is used to identify the subject owning with this resource.</p> <p>The template can reference the \$1, \$2, and so on variables saved during the pattern matching.</p> <p>For example, if the subjectdomain is set to \$2, and using the uri in the uripattern example, the domain of the subject will be example.com.</p> <p>Can be empty indicating the default domain.</p> <p>Syntax: string Default: \$2</p>
davcore.principalsuri.*.subjectfilter	<p>Once it is determined that a URI matches the pattern, this LDAP filter template is used to identify the subject owning with this resource.</p> <p>The template can reference the \$1, \$2, and so on variables saved during the pattern matching.</p> <p>For example, if the subjectfilter is set to (mail=\$1@\$2), and using the URI in the uripattern example, the LDAP filter becomes (mail=john@example.com).</p> <p>Can be empty, indicating that this namespace is not associated with a particular subject.</p> <p>Syntax: string Default: (mail=\$1@\$2)</p>
davcore.principalsuri.*.uripattern	<p>Regex pattern to be matched by the URI.</p> <p>This pattern can contain regex groups (identified by () parenthesis) which will be saved into \$1, \$2, and so on.</p> <p>The last regex group identifies the local path if there is any.</p> <p>For example, if the pattern is ^/home/([^/+]@([^/+])(/\z /.*), the URI /home/john@example.com/calendar/ will match that pattern. \$1 will be set to the value john, \$2 will be set to the value example.com and the local path will be /calendar.</p> <p>Syntax: string Default: ^/home/([^/+]@([^/+])(/\z /.*)</p>
davcore.reverseuri.*.backendid	<p>Back-end ID on which to apply this reverse mapping. There should be only one mapping per backend.</p> <p>Syntax: string Default: None.</p>

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.reverseuri.*.uritemplate	<p>Canonical form of the URI prefix for this back end.</p> <p>This template should have a corresponding uripattern. It should not end with a slash.</p> <p>The template can reference LDAP attributes of the subject, using the \${attrname} syntax or the \${attrname,defaultvalue} syntax. The \${domain} syntax can be used to reference the domain of the subject.</p> <p>If no template is defined for a given back end, the uriinfo.defaulthomeuritemplate configuration parameter is used.</p> <p>Syntax: string Default: None.</p>
davcore.scheduling.allowownerdoublebooking	<p>If set, owners of resource calendars can double book even if the resource account prevents double booking.</p> <p>Syntax: boolean Default: false</p>
davcore.scheduling.calendarinboxexpirytime	<p>Specifies the number of seconds after which the resources in calendar-inbox will be deleted.</p> <p>Syntax: long Unit: seconds Minimum: 0 Maximum: Maximum long value Default: 2592000</p>
davcore.scheduling.calendaroutboxexpirytime	<p>Specifies the number of seconds after which the resources in calendar outbox are deleted.</p> <p>Syntax: long Unit: seconds Minimum: 0 Maximum: Maximum long value Default: 604800</p>
davcore.scheduling.includememberinattendee	<p>If set, scheduling messages would include the MEMBER=group attribute in the ATTENDEE property, where group is the LDAP group of which this attendee is a member. The default is set to false to accommodate compatibility with Leopard Apple iCal.</p> <p>Syntax: boolean Default: false</p>
davcore.scheduling.ischedulebackendid	<p>iSchedule back-end identifier. If not set, incoming iSchedule requests are disabled.</p> <p>Syntax: string Default: None.</p>

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
<p>davcore.scheduling.itiptransmittablexprops</p>	<p>Space separated list of iCalendar X- properties that are transmittable through iTIP.</p> <p>All other X- properties are not transmitted between organizer and attendees.</p> <p>Syntax: string</p> <p>Default: X-S10C-OWNER-APPT-ID X-S10C-TZID X-S10C-OUTLOOK-SENDER-EMAIL X-S10C-OUTLOOK-SENDER-CN X-S10C-OUTLOOK-ACCEPTED-BY-NAME X-S10C-OUTLOOK-EVENT-REPLY-TIME</p>
<p>davcore.scheduling.localuserattr</p>	<p>Local Calendar Server identifier attribute in LDAP. If not set, all users found in LDAP are considered local. For deployments with multiple servers, that can only partially interoperate, this option must be set to a valid LDAP attribute. (For example: davStore). Users with a valid value for that attribute in LDAP are considered local. Others are considered remote.</p> <p>Make sure the same attribute is added to davcore.uriinfo.subjectattributes values.</p> <p>Syntax: string</p> <p>Default: None.</p>
<p>davcore.scheduling.maxattendeesforrefresh</p>	<p>Above this limit of attendees, attendee's reply are only sent to the organizer and are no longer propagated to the other attendees.</p> <p>Syntax: long</p> <p>Minimum: 0</p> <p>Maximum: Maximum long value</p> <p>Default: 50</p>
<p>davcore.scheduling.maxbookingwindow</p>	<p>Specifies the number of days calendars that do not allow double booking can be booked in advance. A valid range is [0, 2G].</p> <p>Syntax: integer</p> <p>Unit: days</p> <p>Minimum: 1</p> <p>Maximum: Maximum int value - 1</p> <p>Default: 365</p>
<p>davcore.scheduling.maxretry</p>	<p>Specifies maximum number of attempts to deliver a scheduling message (for example, when the SMTP server or remote iSchedule server is temporarily down).</p> <p>Syntax: integer</p> <p>Minimum: 0</p> <p>Maximum: 999</p> <p>Default: 24</p>

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.scheduling.minbookingwindow	<p>Specifies the start of a booking window in days from the time of scheduling that a calendar can be booked in advance. A valid range is [0, 2G]. A negative integer value indicates minimum booking window is not honored during the free busy check.</p> <p>Syntax: integer Unit: days Minimum: -1 Maximum: Maximum int value - 1 Default: -1</p>
davcore.scheduling.rejectinactiverecipients	<p>When set to true, recipients of scheduling messages who have their icsStatus set to inactive are treated as unknown recipients. Otherwise, those recipients are treated as iMIP recipients.</p> <p>Syntax: boolean Default: false</p>
davcore.scheduling.rejectdeletedrecipients	<p>When set to true, users or groups that have icsstatus set to 'deleted' are treated as external addresses and an iMIP invitation is sent.</p> <p>Syntax: boolean Default: false</p>
davcore.scheduling.retryinterval	<p>Specifies the number of seconds to wait between two attempts to deliver a scheduling message (for example, if the SMTP server or remote iSchedule server is temporarily down).</p> <p>Syntax: long Unit: seconds Minimum: 0 Maximum: Maximum long value Default: 3600</p>
davcore.scheduling.schedulinglogfilter	<p>Space-separated list of specific user email addresses. Logging is done at a more detailed level for any user in this list who is the organizer or an attendee of an event used for scheduling.</p> <p>Syntax: string Default: None.</p>
davcore.scheduling.synchronousdelivery	<p>If set, scheduling messages (invitations, replies, cancel, and so on) are delivered synchronously on submit. Do not use this option during under normal operation.</p> <p>Syntax: boolean Default: false</p>
davcore.serverdefaults.exportconfigdir	<p>Directory path for exported XSL transformation files.</p> <p>Syntax: filepath Default: config/export</p>
davcore.serverdefaults.importconfigdir	<p>Directory path for imported properties and translation files.</p> <p>Syntax: filepath Default: config/import</p>

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.serverdefaults.jsonprefix	Default prefix to append to all JSON output. Syntax: string Default: &&
davcore.serverdefaults.sslprotocols	Specifies a space-delimited list of the supported SSL protocols as the default for the various back-end services' sslprotocols configuration. That is, if the specific sslprotocols parameter is not set, it is set to the value of davcore.serverdefaults.sslprotocols . Syntax: string Default: TLSv1 TLSv1.1 TLSv1.2
davcore.serverdefaults.tzid	Default TZID to return when a calendar collection does not have one explicitly set. Syntax: string Default: None.
davcore.serverlimits.httpconnecttimeout	HTTP connection timeout value (in milliseconds), when connecting to another server. Syntax: integer Minimum: 500 Maximum: 100000 Default: 5000
davcore.serverlimits.httpsockettimeout	HTTP Socket timeout value (in milliseconds), when connecting to another server, and waiting for data. Syntax: integer Minimum: 500 Maximum: 100000 Default: 5000
davcore.serverlimits.maxaddressbookcontentlength	Maximum size of an address book resource. Syntax: long Unit: bytes Minimum: 0 Maximum: Maximum long value Default: 10000000
davcore.serverlimits.maxattendeesperinstance	Maximum number of ATTENDEE properties in any instance of a calendar object resource stored in a calendar collection. Syntax: long Minimum: 0 Maximum: Maximum long value Default: 1000
davcore.serverlimits.maxcalendarcontentlength	Maximum size of a calendar resource. Syntax: long Unit: bytes Minimum: 0 Maximum: Maximum long value Default: 10000000

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.serverlimits.maxcontentlength	Maximum size of a resource. Might be overwritten for certain types of content (for example, text/calendar). Syntax: long Unit: bytes Minimum: 0 Maximum: Maximum long value Default: 10000000
davcore.serverlimits.maxgrouexpansion	Maximum nested level of group expansion. Syntax: integer Minimum: 0 Maximum: -1 Default: 3
davcore.serverlimits.maxhttpredirects	Maximum number of HTTP redirects to follow, when connecting to another server. Syntax: integer Minimum: 0 Maximum: 10 Default: 3
davcore.serverlimits.maxischedulecontentlength	Maximum size when posting ischedule requests. This affects iSchedule freebusy and scheduling requests. Syntax: long Unit: bytes Minimum: 0 Maximum: Maximum long value Default: 10000000
davcore.serverlimits.maxmigrationthreads	Maximum number of threads to create when running migration. Syntax: integer Minimum: 1 Maximum: 20 Default: 2
davcore.serverlimits.maxnumberofresourcesincollection	Maximum number of resources allowed in a collection. A value of -1 means no limit. Syntax: long Unit: bytes Minimum: -1 Maximum: Maximum long value Default: 10000

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.serverlimits.maxresults	<p>Maximum number of resources returned by a single fetch operation (WebDAV PROPFIND, CalDAV Reports, WCAP fetch or export, and so on).</p> <p>A value of 0 means no limit.</p> <p>Admins are not affected by this limit.</p> <p>Syntax: integer</p> <p>Minimum: 0</p> <p>Maximum: Maximum int value</p> <p>Default: 10000</p>
davcore.serverlimits.maxsearchtimerange	<p>Maximum bounded search range in days.</p> <p>Syntax: long</p> <p>Unit: days</p> <p>Minimum: 0</p> <p>Maximum: 366000</p> <p>Default: 3660</p>
davcore.serverlimits.maxuploadcontentlength	<p>Maximum size when uploading data. This affects operations that let you create multiple resources in one request (for example, import). It does not affect a regular PUT command.</p> <p>Syntax: long</p> <p>Unit: bytes</p> <p>Minimum: 0</p> <p>Maximum: Maximum long value</p> <p>Default: 20000000</p>
davcore.serverlimits.migrationtimeout	<p>Maximum number of hours to wait before terminating a migration.</p> <p>Syntax: integer</p> <p>Minimum: 1</p> <p>Maximum: 100</p> <p>Default: 8</p>
davcore.serverlimits.minsearchcharacters	<p>Minimum number of characters allowed in a text filter search.</p> <p>Syntax: integer</p> <p>Minimum: 0</p> <p>Maximum: 256</p> <p>Default: 3</p>
davcore.serverlimits.templockretry	<p>Maximum number of attempts to acquire a temporary lock when doing write operations.</p> <p>Syntax: integer</p> <p>Minimum: 1</p> <p>Maximum: Maximum int value</p> <p>Default: 20</p>

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.serverlimits.templocktimeout	<p>Maximum amount of time to wait for a temporary lock when doing write operations.</p> <p>Syntax: integer</p> <p>Unit: seconds</p> <p>Minimum: 1</p> <p>Maximum: Maximum int value</p> <p>Default: 60</p>
davcore.serverlimits.templockusebackend	<p>If true, temporary locks are ensured at the back-end level instead of staying local to a server instance.</p> <p>Syntax: boolean</p> <p>Default: false</p>
davcore.uriinfo.backendidtemplate	<p>The backendid template is used to identify the back-end server hosting the home of a given subject.</p> <p>The template can reference LDAP attributes of the subject, using the {attrname} syntax or the {attrname,defaultvalue} syntax.</p> <p>Syntax: string</p> <p>Default: {davStore,defaultbackend}</p>
davcore.uriinfo.defaultdavuriprefix	<p>Canonical form of DAV URI prefix for WebDAV-based protocols.</p> <p>This prefix corresponds to one of the the DavServlet specific path (for example, /dav) as defined in web.xml.</p> <p>It should not end with a slash.</p> <p>Syntax: string</p> <p>Default: /dav</p>
davcore.uriinfo.defaulthomeuritemplate	<p>Canonical form of a subject home URI prefix.</p> <p>This template should have a corresponding uripattern. It should not end with a slash</p> <p>The template can reference LDAP attributes of the subject, using the {attrname} syntax or the {attrname,defaultvalue} syntax. The {domain} syntax can be used to reference the domain of the subject.</p> <p>Syntax: string</p> <p>Default: /home/{mail}</p>
davcore.uriinfo.defaultprincipaluritemplate	<p>Canonical form of a subject principal URI prefix.</p> <p>This template should have a corresponding uripattern. It should not end with a slash.</p> <p>The template can reference LDAP attributes of the subject, using the {attrname} syntax or the {attrname,defaultvalue} syntax. The {domain} syntax can be used to reference the domain of the subject.</p> <p>Syntax: string</p> <p>Default: /principals/{mail}</p>

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.uriinfo.defaultresturiprefix	<p>Canonical form of REST URI prefix for WebDAV-based protocols.</p> <p>This prefix corresponds to one of the RESTfulServlet specific path as defined in the web.xml file.</p> <p>It should not end with a slash.</p> <p>Syntax: string</p> <p>Default: /rest</p>
davcore.uriinfo.directoryrootcollection	<p>Defines the root collection of all directory collections (without any prefix).</p> <p>Syntax: string</p> <p>Default: /directory/</p>
davcore.uriinfo.emailsearchfiltertemplate	<p>LDAP Filter used when searching a subject by email address. The %s token is replaced by the email value to search.</p> <p>Syntax: string</p> <p>Default: !(mail=%s)(mailalternateaddress=%s)</p>
davcore.uriinfo.fulluriprefix	<p>Full URL prefix to use wherever a full URL is required. It should not end with a slash.</p> <p>This prefix is used to construct attachment URLs embedded in calendar resources.</p> <p>Modifying this parameter does not change full URLs in already existing calendar resources.</p> <p>If SSL is used, the host name part of this prefix should match the host name associated with the certificate.</p> <p>Syntax: string</p> <p>Default: http://localhost</p>
davcore.uriinfo.ldapcachesize	<p>Maximum number of subjects (LDAP users, resources and groups) kept in cache when mapping URIs and subjects. Entries are removed from the cache only when this maximum is reached or when any of the uriinfo configuration parameters are changed. Can be set to 0, indicating no cache.</p> <p>Syntax: integer</p> <p>Minimum: 0</p> <p>Maximum: Maximum int value</p> <p>Default: 1000</p>
davcore.uriinfo.ldapcachettl	<p>Maximum time (in seconds) that subjects (LDAP users, resources and groups) are kept in cache when mapping URIs and subjects.</p> <p>Syntax: integer</p> <p>Unit: seconds</p> <p>Minimum: 1</p> <p>Maximum: Maximum int value</p> <p>Default: 60</p>

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.uriinfo.permanentuniqueid	<p>Name of an LDAP attribute present in the LDAP entry of all subjects (users, groups, resources, and so on) that defines a permanent and unique identifier for each subject.</p> <p>The attribute value is used internally to do the mapping between the subject LDAP entry and its repository. As such, it should remain constant for the lifetime of the subject LDAP entry and it should be unique (at least within the subject domain).</p> <p>Warning: Changing this configuration parameter results in data loss once the user's repository has been created.</p> <p>Syntax: string</p> <p>Default: davuniqueid</p>
davcore.uriinfo.principalsrootcollection	<p>Defines the root collection of all principals in their canonical form. (without any prefix). This parameter is used to return the WebDAV DAV:principal-collection-set property.</p> <p>Syntax: string</p> <p>Default: /principals/</p>
davcore.uriinfo.subjectattributes	<p>Space-separated list of LDAP attribute names to retrieve when doing a search for users, group or resources.</p> <p>Syntax: string</p> <p>Default: cn davstore icsstatus mail mailalternateaddress nsuniqueid owner preferredlanguage uid objectclass ismemberof uniquemember memberurl mgrprfc822mailmember kind</p>
davcore.uriinfo.subjectsearchfilter	<p>LDAP filter used when a user is searching for other users. The %s token is replaced by the search string.</p> <p>Syntax: string</p> <p>Default: (!(uid=%s*)(cn=%s*)(mail=%s*))</p>
davcore.uriinfo.subjectsearchfilterminimum	<p>The minimum number of characters allowed for the search string.</p> <p>Syntax: integer</p> <p>Minimum: -2147483648</p> <p>Maximum: Maximum int value</p> <p>Default: 3</p>
davcore.uriinfo.uricachesize	<p>Maximum number of resolved URIs kept in cache. Entries are removed from the cache only when this maximum is reached or when any of the uriinfo configuration parameters are changed.</p> <p>Can be set to 0, indicating no cache.</p> <p>Syntax: integer</p> <p>Minimum: 0</p> <p>Maximum: Maximum int value</p> <p>Default: 10000</p>

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.uriinfo.uricachettl	Maximum time (in seconds) that resolved URIs are kept in cache. Syntax: integer Unit: seconds Minimum: 1 Maximum: Maximum int value Default: 60
davcore.uriinfo.useldaproxyauth	If set to true , uses proxy authorization for any LDAP search on behalf of a user. If set to false , uses administrator credentials for all LDAP searches. Syntax: boolean Default: true
davcore.virusscan.auth	The virus scan connection should use user and password authorization. Syntax: boolean Default: false
davcore.virusscan.clivirusaction	Action to be performed when a virus is detected during command-line operation. Value is empty or delete . Syntax: string Default: None.
davcore.virusscan.debug	Enables or disables debugging on the virus SMTP connection. Syntax: boolean Default: false
davcore.virusscan.emailaddress	Email recipient address that the MTA is configured to use to trigger a custom virus scan. (Requires MTA configuration). Syntax: string Default: None.
davcore.virusscan.host	MTA host name configured to accept virus scans. Syntax: string Default: None.
davcore.virusscan.onlineenable	Enables or disables online virus scan. Syntax: boolean Default: false
davcore.virusscan.onlinefailureaction	Action to be performed when virus service fails during an online submission. Value is empty or reject . Syntax: string Default: None.
davcore.virusscan.onlinevirusaction	Action to be performed when a virus is detected during an online submission. Value is empty or reject . Syntax: string Default: None.

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
davcore.virusscan.pass	The SMTP authorization password for the SMTP virus scan connection. Syntax: password Default: None.
davcore.virusscan.port	MTA host port configured to accept virus scans. Syntax: string Default: 25
davcore.virusscan.starttls	The virus scan connection should use starttls. Syntax: boolean Default: false
davcore.virusscan.timeout	Timeout value (in milliseconds) for the connection to the MTA during a virus scan operation. Syntax: string Default: 10000
davcore.virusscan.user	The SMTP user authorized for the SMTP virus scan connection. Syntax: string Default: None.
davcore.virusscan.usessl	The virus scan connection should use SSL. Syntax: boolean Default: false
log.dav.commands.logdateformat	Specifies the date format pattern for the log. Syntax: logdateformat Default: yyyy-MM-dd'T'HH:mm:ss.SSSZ
log.dav.commands.logdir	Specifies the directory path for log files. Syntax: filepath Default: logs
log.dav.commands.loglevel	Specifies the log level. Valid levels are OFF (no information is logged), SEVERE , WARNING , INFO , CONFIG , FINE , FINER , FINEST , ALL (all information is logged). The FINEST and ALL levels produce a large amount of data. Syntax: loglevel Default: INFO
log.dav.commands.logtoparent	Flag to enable logging to the GlassFish Server log file, in addition to the Calendar Server logs. Syntax: boolean Default: false

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
log.dav.commands.maxlogfiles	Maximum number of log files. Syntax: integer Minimum: 1 Maximum: 100 Default: 10
log.dav.commands.maxlogfilesize	Maximum size of each log file. Syntax: integer Unit: bytes Minimum: 2097152 Maximum: Maximum int value Default: 2097152
log.dav.errors.logdateformat	Specifies the date format pattern for the log. Syntax: logdateformat Default: yyyy-MM-dd'T'HH:mm:ss.SSSZ
log.dav.errors.logdir	Specifies the directory path for log files. Syntax: filepath Default: logs
log.dav.errors.loglevel	Specifies the log level. Valid levels are OFF (no information is logged), SEVERE , WARNING , INFO , CONFIG , FINE , FINER , FINEST , ALL (all information is logged). The FINEST and ALL levels produce a large amount of data. Syntax: loglevel Default: INFO
log.dav.errors.logtoparent	Flag to enable logging to the GlassFish Server log file, in addition to the Calendar Server logs. Syntax: boolean Default: false
log.dav.errors.maxlogfiles	Maximum number of log files. Syntax: integer Minimum: 1 Maximum: 100 Default: 10
log.dav.errors.maxlogfilesize	Maximum size of each log file. Syntax: integer Unit: bytes Minimum: 2097152 Maximum: Maximum int value Default: 2097152

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
log.dav.scan.logdateformat	Specifies the date format pattern for the log. Syntax: logdateformat Default: yyyy-MM-dd'T'HH:mm:ss.SSSZ
log.dav.scan.logdir	Specifies the directory path for log files. Syntax: filepath Default: logs
log.dav.scan.loglevel	Specifies the log level. Valid levels are OFF (no information is logged), SEVERE , WARNING , INFO , CONFIG , FINE , FINER , FINEST , ALL (all information is logged). The FINEST and ALL levels produce a large amount of data. Syntax: loglevel Default: INFO
log.dav.scan.logtoparent	Flag to enable logging to the GlassFish Server log file, in addition to the Calendar Server logs. Syntax: boolean Default: false
log.dav.scan.maxlogfiles	Maximum number of log files. Syntax: integer Minimum: 1 Maximum: 100 Default: 10
log.dav.scan.maxlogfilesize	Maximum size of each log file. Syntax: integer Unit: bytes Minimum: 2097152 Maximum: Maximum int value Default: 2097152
log.dav.scheduling.logdateformat	Specifies the date format pattern for the log. Syntax: logdateformat Default: yyyy-MM-dd'T'HH:mm:ss.SSSZ
log.dav.scheduling.logdir	Specifies the directory path for log files. Syntax: filepath Default: logs
log.dav.scheduling.loglevel	Specifies the log level. Valid levels are OFF (no information is logged), SEVERE , WARNING , INFO , CONFIG , FINE , FINER , FINEST , ALL (all information is logged). The FINEST and ALL levels produce a large amount of data. Syntax: loglevel Default: INFO

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
log.dav.scheduling.logtoparent	Flag to enable logging to the GlassFish Server log file, in addition to the Calendar Server logs. Syntax: boolean Default: false
log.dav.scheduling.maxlogfiles	Maximum number of log files. Syntax: integer Minimum: 1 Maximum: 100 Default: 10
log.dav.scheduling.maxlogfilesize	Maximum size of each log file. Syntax: integer Unit: bytes Minimum: 2097152 Maximum: Maximum int value Default: 2097152
log.dav.telemetry.logdateformat	Specifies the date format pattern for the log. Syntax: logdateformat Default: yyyy-MM-dd'T'HH:mm:ss.SSSZ
log.dav.telemetry.logdir	Specifies the directory path for log files. Syntax: filepath Default: logs
log.dav.telemetry.loglevel	Specifies the log level. Valid levels are OFF (no information is logged), SEVERE , WARNING , INFO , CONFIG , FINE , FINER , FINEST , ALL (all information is logged). The FINEST and ALL levels produce a large amount of data. Syntax: loglevel Default: INFO
log.dav.telemetry.logtoparent	Flag to enable logging to the GlassFish Server log file, in addition to the Calendar Server logs. Syntax: boolean Default: false
log.dav.telemetry.maxlogfiles	Maximum number of log files. Syntax: integer Minimum: 1 Maximum: 100 Default: 10

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
log.dav.telemetry.maxlogfilesize	Maximum size of each log file. Syntax: integer Unit: bytes Minimum: 2097152 Maximum: Maximum int value Default: 2097152
notification.dav.configdir	Specifies the directory path for notification configuration files or format files. Syntax: filepath Default: config/templates
notification.dav.dateformat	Specifies the date format pattern for notification. For example, EEE MMMMM dd, yyyy . Syntax: dateformat Default: EEE MMMMM dd, yyyy
notification.dav.enableemailnotif	Enables or disables server-wide email notification. Syntax: boolean Default: true
notification.dav.enableimip	Enables or disables server-wide iMIP scheduling. Syntax: boolean Default: true
notification.dav.enableimipemailnotif	Enables or disables server-wide scheduling email notification that contains iMIP data. Syntax: boolean Default: false
notification.dav.enablejmsnotif	Enables or disables server-wide Java Message Service (JMS) notification. Syntax: boolean Default: true
notification.dav.maxpayload	Maximum payload size in bytes. Syntax: integer Minimum: -2147483648 Maximum: Maximum int value Default: 1000000
notification.dav.smtpauth	SMTP-AUTH access control mechanism flag. Syntax: string Default: false
notification.dav.smtpdebug	SMTP debug flag. Syntax: string Default: false
notification.dav.smtphost	SMTP host. Syntax: string Default: None.

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
notification.dav.smtppassword	SMTP password. Syntax: password Default: None.
notification.dav.smtpport	SMTP port. Syntax: string Default: 25
notification.dav.smtpstarttls	SMTP starttls flag. Syntax: string Default: true
notification.dav.smtpuser	SMTP user. Syntax: string Default: user
notification.dav.smtpusessl	SMTP use SSL flag. Syntax: string Default: false
notification.dav.timeformat	Specifies the time format pattern for notification. Use 'a' for AM/PM marker. For example, hh:mm:ss aaa . Syntax: timeformat Default: hh:mm:ss aaa
notification.dav.timezoneformat	Specifies the time zone format pattern for notification. Use 'z' for general time zone, or 'Z' for RFC822 time zone. Syntax: timezoneformat Default: z
service.dav.blacklist	List of CalDAV clients to be denied of service, expressed as a space separated list of regular expressions. Any client whose User-Agent HTTP header contains any of the regex is denied access. Syntax: string Default: None.
service.dav.ischedulewhitelist	List of hosts that are allowed to send iScheduling POST requests. Space-separated list of single host IP addresses and/or Classless Inter-Domain Routing (CIDR) entries. A CIDR entry is a base IP address followed by a number indicating how many upper bits to mask. For example, 10.20.30.0/24 matches all addresses in the range 10.20.30.0 - 10.20.30.255. If the entry is 0.0.0.0/0, all requests are allowed. If the list is empty, all requests are denied, except for those from "localhost". Syntax: string Default: None
service.dav.propfinddavheadervalue	Value of the HTTP Dav header value to return in all PROPFIND responses. Syntax: string Default: 1, 3, access-control, calendar-proxy, calendarserver-principal-property-search, calendar-access, calendar-auto-schedule, addressbook

Table 17–1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
<code>service.dav.telemetry.filter</code>	Space-separated list of request URIs that a particular request should match (start with) to be logged by telemetry (for example, <code>/dav/home/jsmith/calendar/</code> <code>/dav/home/jdoe/calendar/</code>). Syntax: string Default: None.
<code>service.dav.telemetry.forcetelemetry</code>	Force telemetry for all users. Warning: Setting this parameter to true generates a lot of data and should not be used on a production system. Syntax: boolean Default: false
<code>service.wcap.blacklist</code>	List of WCAP clients to be denied of service, expressed as a space-separated list of regular expressions. Any client whose User-Agent HTTP header contains any of the regex is denied access. Syntax: string Default: None.
<code>service.wcap.maxsessions</code>	Maximum number of WCAP session IDs stored in the sessions cache. Entries are removed from the cache only when this maximum is reached, or when a logout command is executed against the sessionid, or the entry has been in the cache for as long as the session timeout allows. Syntax: integer Minimum: 0 Maximum: Maximum int value Default: 10000
<code>service.wcap.sessiontimeout</code>	Number of seconds before expiring a session. Syntax: integer Unit: seconds Minimum: 1 Maximum: Maximum int value Default: 1800
<code>store.corpdir.defaultcorpdirectoryurl</code>	Default corporate directory information to use when doing searches. Can be overwritten by domain specific information (corpDirectoryUrl LDAP attribute in the domain entry). If no baseDN is provided, the user's domain baseDN for users and group is used. The list of attributes to retrieve is ignored. Syntax: string Default: ldap://ugldap??sub?(objectclass=*)
<code>store.corpdir.enablecorpdir</code>	Enables or disables corporate directory lookups. Syntax: boolean Default: true

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
store.corpdir.useldaproxyauth	<p>If set to true, uses LDAP proxy authorization to issue LDAP searches on behalf of the logged in user.</p> <p>If set to false, uses the LDAP Pool credentials for all LDAP searches.</p> <p>This parameter applies only to the default corporate directory configuration.</p> <p>Syntax: boolean Default: true</p>
store.dav.*.attachstorehost	<p>Document store host.</p> <p>Syntax: string Default: None.</p>
store.dav.*.attachstoreport	<p>Document store port.</p> <p>Syntax: integer Minimum: -2147483648 Maximum: Maximum int value Default: 8008</p>
store.dav.*.backendid	<p>Back-end identifier.</p> <p>Syntax: string Default: None.</p>
store.dav.*.dbdir	<p>Specifies the directory path for Calendar Server store.</p> <p>Syntax: filepath Default: data/db</p>
store.dav.*.jndiname	<p>JNDI name pointing to this back-end host's JDBC DataSource, as defined in the J2EE container (for example, jdbc/defaultbackend).</p> <p>Syntax: string Default: None.</p>
store.dav.*.purgedelay	<p>Sets the delay between deletion of a resource and its actual removal (purge) from the back-end database. Setting this value too low may cause synchronization clients to do a full resynchronization too often.</p> <p>Syntax: long Unit: seconds Minimum: 0 Maximum: Maximum long value Default: 2592000</p>

Table 17-1 (Cont.) Calendar Server Configuration Parameters

Parameter	Description
store.document.password	Password to use when authenticating to a remote document store server. Syntax: password Default: None.
store.document.timeout	The HTTP(S) connection and read timeout value. Syntax: integer Minimum: -2147483648 Maximum: Maximum int value Default: 10000
store.document.usessl	Use SSL for communications with the remote document store server. Syntax: boolean Default: false



Time Zone Database

This chapter lists the Time Zone Database (often called **tz** or **zoneinfo**) IDs that are supported by Oracle Communications Calendar Server. This list is not necessarily what shows up in WCAP clients. To add any of the following time zones to the WCAP client list, see "[Administering Time Zones Support](#)" for details.

The database includes the following time zones:

- [Africa](#)
- [America](#)
- [Antarctica](#)
- [Arctic](#)
- [Asia](#)
- [Atlantic](#)
- [Australia](#)
- [Europe](#)
- [Indian](#)
- [Pacific](#)

Africa

```
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Bamako
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
Africa/Brazzaville
Africa/Bujumbura
Africa/Cairo
Africa/Casablanca
Africa/Ceuta
Africa/Conakry
Africa/Dakar
Africa/Dar_es_Salaam
Africa/Djibouti
```

Africa/Douala
Africa/El_Aaiun
Africa/Freetown
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Lome
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
Africa/Malabo
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Monrovia
Africa/Nairobi
Africa/Ndjamena
Africa/Niamey
Africa/Nouakchott
Africa/Ouagadougou
Africa/Porto-Novo
Africa/Sao_Tome
Africa/Tripoli
Africa/Tunis
Africa/Windhoek

America

America/Adak
America/Anchorage
America/Anguilla
America/Antigua
America/Araguaina
America/Aruba
America/Asuncion
America/Atikokan
America/Bahia
America/Bahia_Banderas
America/Barbados
America/Belem
America/Belize
America/Blanc-Sablon
America/Boa_Vista
America/Bogota
America/Boise
America/Cambridge_Bay
America/Campo_Grande
America/Cancun
America/Caracas
America/Cayenne
America/Cayman
America/Chicago
America/Chihuahua

America/Costa_Rica
America/Cuiaba
America/Curacao
America/Danmarkshavn
America/Dawson
America/Dawson_Creek
America/Denver
America/Detroit
America/Dominica
America/Edmonton
America/Eirunepe
America/El_Salvador
America/Fortaleza
America/Glace_Bay
America/Godthab
America/Goose_Bay
America/Grand_Turk
America/Grenada
America/Guadeloupe
America/Guatemala
America/Guayaquil
America/Guyana
America/Halifax
America/Havana
America/Hermosillo
America/Inuvik
America/Iqaluit
America/Jamaica
America/Juneau
America/La_Paz
America/Lima
America/Los_Angeles
America/Maceio
America/Managua
America/Manaus
America/Marigot
America/Martinique
America/Matamoros
America/Mazatlan
America/Menominee
America/Merida
America/Metlakatla
America/Mexico_City
America/Miquelon
America/Moncton
America/Monterrey
America/Montevideo
America/Montreal
America/Montserrat
America/Nassau
America/New_York
America/Nipigon
America/Nome
America/Noronha
America/Ojinaga
America/Panama
America/Pangnirtung
America/Paramaribo
America/Phoenix
America/Port-au-Prince

America/Port_of_Spain
America/Porto_Velho
America/Puerto_Rico
America/Rainy_River
America/Rankin_Inlet
America/Recife
America/Regina
America/Resolute
America/Rio_Branco
America/Santa_Isabel
America/Santarem
America/Santiago
America/Santo_Domingo
America/Sao_Paulo
America/Scoresbysund
America/Shiprock
America/Sitka
America/St_Barthelemy
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Swift_Current
America/Tegucigalpa
America/Thule
America/Thunder_Bay
America/Tijuana
America/Toronto
America/Tortola
America/Vancouver
America/Whitehorse
America/Winnipeg
America/Yakutat
America/Yellowknife
America/Argentina/Buenos_Aires
America/Argentina/Catamarca
America/Argentina/Cordoba
America/Argentina/Jujuy
America/Argentina/La_Rioja
America/Argentina/Mendoza
America/Argentina/Rio_Gallegos
America/Argentina/Salta
America/Argentina/San_Juan
America/Argentina/San_Luis
America/Argentina/Tucuman
America/Argentina/Ushuaia
America/Indiana/Indianapolis
America/Indiana/Knox
America/Indiana/Marengo
America/Indiana/Petersburg
America/Indiana/Tell_City
America/Indiana/Vevay
America/Indiana/Vincennes
America/Indiana/Winamac
America/Kentucky/Louisville
America/Kentucky/Monticello
America/North_Dakota/Beulah
America/North_Dakota/Center
America/North_Dakota/New_Salem

Antarctica

Antarctica/Casey
Antarctica/Davis
Antarctica/DumontDURville
Antarctica/Macquarie
Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
Antarctica/Rothera
Antarctica/South_Pole
Antarctica/Syowa
Antarctica/Vostok

Arctic

Arctic/Longyearbyen

Asia

Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
Asia/Aqtau
Asia/Aqtobe
Asia/Ashgabat
Asia/Baghdad
Asia/Bahrain
Asia/Baku
Asia/Bangkok
Asia/Beirut
Asia/Bishkek
Asia/Brunei
Asia/Choibalsan
Asia/Chongqing
Asia/Colombo
Asia/Damascus
Asia/Dhaka
Asia/Dili
Asia/Dubai
Asia/Dushanbe
Asia/Gaza
Asia/Harbin
Asia/Hebron
Asia/Ho_Chi_Minh
Asia/Hong_Kong
Asia/Hovd
Asia/Irkutsk
Asia/Istanbul
Asia/Jakarta
Asia/Jayapura
Asia/Jerusalem
Asia/Kabul
Asia/Kamchatka
Asia/Karachi
Asia/Kashgar
Asia/Kathmandu
Asia/Kolkata

Asia/Krasnoyarsk
Asia/Kuala_Lumpur
Asia/Kuching
Asia/Kuwait
Asia/Macau
Asia/Magadan
Asia/Makassar
Asia/Manila
Asia/Muscat
Asia/Nicosia
Asia/Novokuznetsk
Asia/Novosibirsk
Asia/Omsk
Asia/Oral
Asia/Phnom_Penh
Asia/Pontianak
Asia/Pyongyang
Asia/Qatar
Asia/Qyzylorda
Asia/Rangoon
Asia/Riyadh
Asia/Sakhalin
Asia/Samarkand
Asia/Seoul
Asia/Shanghai
Asia/Singapore
Asia/Taipei
Asia/Tashkent
Asia/Tbilisi
Asia/Tehran
Asia/Thimphu
Asia/Tokyo
Asia/Ulaanbaatar
Asia/Urumqi
Asia/Vientiane
Asia/Vladivostok
Asia/Yakutsk
Asia/Yekaterinburg
Asia/Yerevan

Atlantic

Atlantic/Azores
Atlantic/Bermuda
Atlantic/Canary
Atlantic/Cape_Verde
Atlantic/Faroe
Atlantic/Madeira
Atlantic/Reykjavik
Atlantic/South_Georgia
Atlantic/St_Helena
Atlantic/Stanley

Australia

Australia/Adelaide
Australia/Brisbane
Australia/Broken_Hill
Australia/Currie

Australia/Darwin
Australia/Eucla
Australia/Hobart
Australia/Lindeman
Australia/Lord_Howe
Australia/Melbourne
Australia/Perth
Australia/Sydney

Europe

Europe/Amsterdam
Europe/Andorra
Europe/Athens
Europe/Belgrade
Europe/Berlin
Europe/Bratislava
Europe/Brussels
Europe/Bucharest
Europe/Budapest
Europe/Chisinau
Europe/Copenhagen
Europe/Dublin
Europe/Gibraltar
Europe/Guernsey
Europe/Helsinki
Europe/Isle_of_Man
Europe/Istanbul
Europe/Jersey
Europe/Kaliningrad
Europe/Kiev
Europe/Lisbon
Europe/Ljubljana
Europe/London
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Mariehamn
Europe/Minsk
Europe/Monaco
Europe/Moscow
Europe/Nicosia
Europe/Oslo
Europe/Paris
Europe/Podgorica
Europe/Prague
Europe/Riga
Europe/Rome
Europe/Samara
Europe/San_Marino
Europe/Sarajevo
Europe/Simferopol
Europe/Skopje
Europe/Sofia
Europe/Stockholm
Europe/Tallinn
Europe/Tirane
Europe/Uzhgorod
Europe/Vaduz
Europe/Vatican

Europe/Vienna
Europe/Vilnius
Europe/Volgograd
Europe/Warsaw
Europe/Zagreb
Europe/Zaporozhye
Europe/Zurich

Indian

Indian/Antananarivo
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte
Indian/Reunion

Pacific

Pacific/Auckland
Pacific/Chatham
Pacific/Chuuk
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofu
Pacific/Fiji
Pacific/Funafuti
Pacific/Galapagos
Pacific/Gambier
Pacific/Guadalcanal
Pacific/Guam
Pacific/Honolulu
Pacific/Johnston
Pacific/Kiritimati
Pacific/Kosrae
Pacific/Kwajalein
Pacific/Majuro
Pacific/Marquesas
Pacific/Midway
Pacific/Nauru
Pacific/Niue
Pacific/Norfolk
Pacific/Noumea
Pacific/Pago_Pago
Pacific/Palau
Pacific/Pitcairn
Pacific/Pohnpei
Pacific/Port_Moresby
Pacific/Rarotonga
Pacific/Saipan
Pacific/Tahiti
Pacific/Tarawa
Pacific/Tongatapu

Pacific/Wake
Pacific/Wallis

